MU

es easy-solutions

Mumbai University Paper Solutions

easy-solutions

Strictly as per the New Revised Syllabus of
Mumbai University
w.e.f. academic year 2014-2015

SOFTWARE ENGINEERING

Semester VI - Computer Engineering

EME93B Price ₹ 40/-

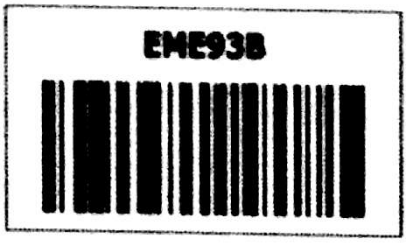
B-40

easy – solutions

Strictly as per new revised syllabus of Mumbai University w.e.f. academic year 2014-2015

Software Engineering

Semester VI - Computer Engineering



INDEX

Table of Contents

□□□

Syllabus

Module 1

Introduction :

1.1 Software Engineering Process Paradigms

1.2 Process Models – Incremental and Evolutionary models,

1.3 Typical Application for each model,

1.4 Agile methodology

1.5 Process and Project Metrics.

Module 2

Software project scheduling, Control & Monitoring :

2.1 Software estimation – Empirical estimation models – Cost/Effort estimation

2.2 Planning – Work breakdown Structure, Gantt Chart. Discuss schedule and cost slippage.

Module 3

Risk Management :

3.1 Risk Identification, Risk Assessment, Risk Projection, RMMM

Module 4

Software Configuration Management :

4.1 Software Configuration items, SCM process, Identification of objects in software configuration, version and change control, configuration audit , status reporting, SCM standards and SCM issues.

Module 5

Software Design Specification :

5.1 Software Design – Abstraction , Modularity

5.2 Software Architecture – Effective modular design, Cohesion and Coupling, Example of code for cohesion and coupling.

5.3 User Interface Design – Human Factors, Interface standards, Design Issues – User Interface Design Process.

Module 6

Software Quality

6.1 Software Quality Assurance – Software standards , Quality metrics Software Reliability, Quality Measurement and Metrics.

Module 7

Software Testing

7.1 Basic concept and terminology, Verification & validation, White Box Testing-Path Testing, Control Structures Testing , DEFUSE testing,

7.2 Black Box Testing –BVA Integration, Validation and system testing.

7.3 OO testing methods-Class Testing, Interclass testing, testing architecture, Behavioral testing.

7.4 Software Maintenance – Reverse Engineering.

Module 8

Web Engineering

8.1 For web based applications – attributes, analysis and design, testing.

8.2 Security Engineering,

8.3 Service-Oriented Software Engineering.

8.4 Test Driven Development

8.5 Software engineering with aspects.

□□□

Software Engineering

Statistical Analysis

Chapter No.	May 2015	Dec. 2015	May 2016	Dec. 2016
Chapter 1	30 Marks	35 Marks	40 Marks	60 Marks
Chapter 2	10 Marks	-	10 Marks	10 Marks
Chapter 3	10 Marks	15 Marks	10 Marks	10 Marks
Chapter 4	10 Marks	10 Marks	10 Marks	20 Marks
Chapter 5	20 Marks	25 Marks	20 Marks	20 Marks
Chapter 6	10 Marks	10 Marks	-	-
Chapter 7	30 Marks	20 Marks	40 Marks	20 Marks
Chapter 8	–	10 Marks	10 Marks	-
Repeated Questions	–	30 Marks	60 Marks	75 Marks

May 2015

Chapter 1 : Software Engineering Process [Total Marks - 30]

Q. 1(a) Write suitable applications of different software models. **(5 Marks)**

Ans. : Applications of different software models :

(1) **The Waterfall model :** The best example for a waterfall model or the classic model can be an automobile industry. For example : Car or Bike production. First the requirements are drafted. Once they are finalized then the design phase starts. Now the requirements do not change unless the bike or the car is completely ready. Then the next step of testing and further more.

(2) **Incremental process models :** The Incremental Process Models are used in application where basic requirement doesn't change but with every increment, some functionality is added. For example the word processing software like Microsoft Word. With each release, more functions are added.

(3) **Evolutionary process models :** The Evolutionary process models are mostly used in large projects. The military had adopted the spiral model for its Future Combat Systems program. The Evolutionary process models are also used in applications where business goals are unstable but the architecture must be realized.

(4) **The Specialized process models :** Special Process Models tend to be used when a narrowly defined Software Engineering Approach is chosen. Formal Methods Development Model are used more where safety-critical software is developed, e.g., aircraft avionics, medical devices, etc.

Q. 2(a) What is Agile methodology ? Explain it with the principles used and give example of any one such software model. **(10 Marks)**

Ans. : Agile methodology :

An agile process model includes the concept of development along with a set of guidelines necessary for the development process. The conceptual design is necessary for the satisfaction of the customer. The concept is also necessary for the incremental delivery of the product. The concept or the philosophy makes development task simple.

The agility team must be highly motivated, updated with latest skill sets and should focus on development simplicity. Agile development is an alternative approach to the conventional development in certain projects. The development guidelines emphasize on analysis and design activities and continuous communication between developers and customers. An agile team quickly responds to changes. The changes may be in development, changes in the team members, and changes due to new technology. The agility can be applied to any software process. It emphasizes rapid delivery of operational software.

Principle of agile alliance :

The agile alliance has given twelve agility principles as follows :

1. Satisfy customer through early and continuous delivery.
2. Accommodate changing requirements.
3. Deliver the working software frequently in shorter time span.
4. The customer, business people and developers must work together on daily basis during entire development.
5. Complete the task with motivated developers and facilitate healthy and friendly environment.
6. Convey the message orally, face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile process promotes sustainable development.
9. Achieve technical excellence and good design.
10. There must be simplicity in development.
11. The best architecture, requirements and design emerge from self-organizing teams.
12. Every team should think how to become more effective. It should review regularly and adjust its behavior accordingly.

Thus the principles of agility may be applied to any software process. To achieve agile development, the team must obey all the principles mentioned above and conduct proper planning.

Example : Extreme Programming (XP) :

The Extreme Programming is one of the most commonly used agile process models. All the agile process models obey the principles of agility and the manifesto of agile software development. The XP uses the concept of object oriented programming. This approach is preferred development paradigm.

As in conventional approach, a developer focuses on the framework activities like planning, design, coding and testing, the XP also has set of rules and practices.

Fig. 1 shows the Extreme Programming process and all the key XP activities.

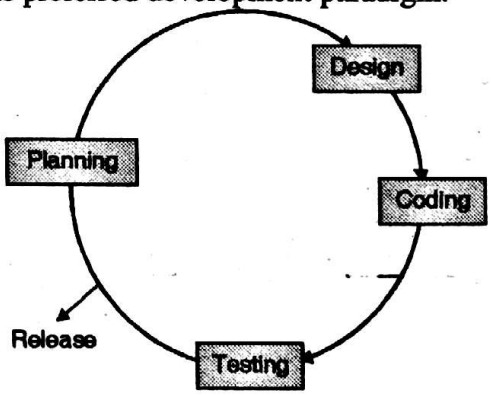


Fig. 1 : Extreme programming process

XP values :

Following are a set of five values that establish a foundation for all work performed in context with XP :

1. Communication 2. Simplicity
3. Feedback
4. Courage 5. Respect

For any development process, there must be a regular meeting of developer and the customer. There should be a proper communication for requirement gathering and discussion of the concepts. The simple design can always be easily implemented in code.

The feedback is an important activity which leads to the discipline in the development process. In every development projects, there is always a pressure situation. The courage or the discipline will definitely make the task easy. In addition to all the XP values, the agile should inculcate respect among all the team members, between other stake holder and with customers.

Q. 3(a) Explain size oriented software engineering metrics. **(5 Marks)**

Ans. :

Size-oriented metrics :

Project	LOC	Effort	$(000)	Pp. doc.	Errors	Defects	People
alpha	12,100	24	168	365	134	29	3
beta	27,200	62	440	1224	321	86	5
gamma	20,200	43	314	1050	256	64	6
.	.	.	.	.	.		
.	.	.	.	.	.		
.	.	.	.	.	.		

Fig. 2 : Size-oriented metrics

Size-oriented software metrics are derived by normalizing quality and productivity measures. This metric also consider the size of the software product. If any software organization keeps the simple records of software development process, then a table containing the size-oriented measures can be created. The example of such a table is shown in Fig. 2. This table has the list of each software development projects that are already completed over the past few years and it keeps the record of their corresponding measures for project. To develop metrics for a project that has the similar metrics compared to other projects, the developer can select the lines of codes as a standard normalization value. In fact, the size-oriented metrics are not considered as foolproof metrics and it is globally not accepted as the best metrics for software process.

Table 1 : Estimating information domain values

Information domain value	Opt.	Likely	Pess.	Est. count	Weight	FP count
Number of external inputs	20	24	30	24	4	97
Number of external outputs	12	15	22	16	5	78
Number of external inquiries	16	22	28	22	5	88
Number of internal logical files	4	4	5	4	10	42
Number of external interface files	2	2	3	2	[illegible]	15
Count total						**320**

Decomposition for FP-based estimation focuses on information domain values rather than software functions. Referring to the table presented in Table 1, the project planner estimates external inputs, external outputs, external inquiries, internal logical files, and external interface files for the CAD software. Each of the complexity weighting factors is estimated and the value adjustment factor is computed.

Factor		Value
1.	Backup and recovery	4
2.	Data communications	2
3.	Distributed processing	0
4.	Performance critical	4
5.	Existing operating environment	3
6.	On-line data entry	4
7.	Input transaction over multiple screens	5
8.	ILFs updated online	3
9.	Information domain values complex	5
10.	Internal processing complex	5
11.	Code designed for reuse	4
12.	Conversion/installation in design	3
13.	Multiple installations	5
14.	Application designed for change	5
	Value adjustment factor	**1.17**

Finally, the estimated number of FP is derived:

$$FP_{estimated} = count - total \times [0.65 + 0.01 \sum(F_i)]$$

$$FP_{estimated} = 375$$

The organizational average productivity for systems of this type is 6.5 FP/pm.

Q. 5(a) What is SRS document ? Build an SRS document for Online Student Feedback System. **(10 Marks)**

Ans. : Software requirement specification :

SRS or software requirement specification is an official document or the statement of what the system developers should implement. It includes both :

1. System requirement. 2. User requirement.

The SRS has a set of users like senior management of the organization, engineers responsible for developing the software.

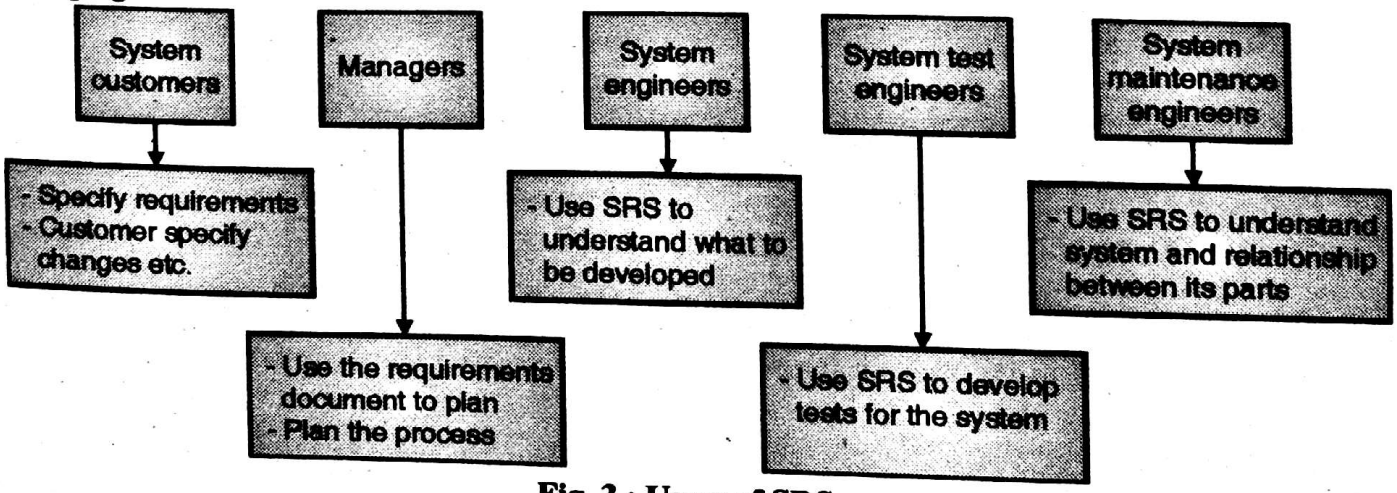


Fig. 3 : Users of SRS

The details which are included in SRS depends on the type of system that is being developed and the type of the development process. A number of large organizations, such as IEEE have defined standards for software requirements document. The most widely used standard is IEEE/ANSI 830-1998.

Online Students Feedback System :

The SRS document for Online Students Feedback System, according to IEEE/ANSI 830-1998 standard is as follows :

1. Introduction to Online Student Feedback System
   - (i) Purpose of Online Student Feedback System
   - (ii) Scope of Application.
   - (iii) Definitions, acronyms, abbreviations.
   - (iv) References
   - (v) Overview
2. General description
   - (i) Perspective of Online Student Feedback System
   - (ii) Functions of Online Student Feedback System
   - (iii) User characteristics
   - (iv) General constraints
   - (v) Assumptions and dependencies
3. Specific requirements
4. Appendices
5. Index.

Chapter 2 : Software Project Scheduling, Control and Monitoring [Total Marks - 10]

Q. 1(c) Explain COCOMO Model. **(5 Marks)**

Ans. : COCOMO model :

Barry Boehm has devised a software estimation models hierarchy having the name as COCOMO i.e. COnstructive COst MOdel. The COCOMO model has been evolved into more comprehensive model called COCOMO II. It is actually having a hierarchy of estimation models that focus the following areas :

1. **Application composition model :** It is used in the early stages of development, when user interfaces, system interaction, performance assessment and technology evaluation are prime important.
2. **Early design stage model :** Once the requirements are stabilized and basic architecture is constructed, then early design stage model is used.
3. **Post-architecture stage model :** It is used during development of the software.

COCOMOII models also require sizing information like other estimation models for the software. Following three sizing options are available :

1. Object points 2. Function points and 3. Lines of source code

The COCOMO II model uses object points that are an indirect software measure. They are is computed using the counts of :

1. Screenshots taken at user interface
2. Reports and

3. Components required in developing the application.

The screenshots or the reports are classified into any of the following three complexity levels as, Simple, Medium and Difficult

The client and server data tables are required to create the screenshots and reports. The complexity is defined as the function of these reports.

Table 2(a) : Complexity weighting for object types

Object type	Complexity weight		
	Simple	Medium	Difficult
Screen	1	2	3
Report	2	5	8
3GL component			10

After the determination of complexity, the number of screenshots, reports, and components object point are weighted as per the table shown in Table 2(a). In component-based development when software reuse is implemented, then the percent of reuse (% reuse) is estimated and the object point count is adjusted according to the following equation :

$$NOP = (\text{object points}) \times [(100 - \% \text{ reuse})/100]$$

where NOP → New Object Points.

Table 2(b) : Productivity rate for object points

Developer's experience/capability	Very low	Low	Nominal	High	Very high
Environment maturity / capability	Very low	Low	Nominal	High	Very high
PROD	4	7	13	25	50

In order to derive the estimate of effort, a "productivity rate" should be derived first. They are based on the computed NOP value. The Table 2(b) shown above presents the productivity rate. These productivity rates are illustrated for various levels of developer's experience as well as various levels of environment maturity.

$$PROD = NOP/\text{person-month}$$

After determining the productivity rate, an estimate of project effort can be derived as follows :

$$\text{Estimated effort} = NOP/PROD$$

Q. 3(a) Find function points for an e-commerce application with following data,

Number of user Inputs	50
Number of user Outputs	40
Number of user Inquiries	35
Number of user Files	06
Number of External Interfaces	04

Assume suitable complexity adjustment factors and weighting factors. **(5 Marks)**

Ans. : The method to calculate the Function Point is as follows :

Counting function points :

Table 3 : Functional units with weighting factors

Functional Units	Weighting factors		
	Low	Average	High
External Inputs (EI)	3	4	6
External output (EO)	4	5	7
External Inquiries (EQ)	3	4	6
External logical files (ILP)	7	10	15
External interface files (EIF)	5	7	10

By using the Table 3, we estimate the weighting factor. Then functional units are multiplied with the weighting factors.

The Unadjusted Function Point (UFP) is calculated as follows :

$$UFP = \sum_{i=1}^{5} \times \sum_{j=1}^{3} (Z_{ij} W_{ij})$$

Where, i indicates the row and j indicates the column of the above table.

W_{ij} = entry of i^{th} row and j^{th} column of above table.

Z_{ij} = count of number of functional units of type i that have been classified corresponding to column j.

Now by using the following formula, Function Point is calculated :

$$FP = UFP \times CAF$$

Where, CAF is Complexity Adjustment Factor and it is equal to :

$$(0.65 + 0.01 \times \sum F_j)$$

Where, F_j (i = 1 to 14) are the degree of influence.

Now we assume that all complexity adjustment factors and weighting factors are average. Calculate Function Point as per the formula as follows :

Measurement parameter	Count	Weighting Factor(Average)	Total
Number of user inputs	50	4	= 200
Number of user outputs	40	5	= 200
Number of user enquiries	35	4	= 140
Number of user files	6	10	= 60
Number of external interfaces	4	7	= 28

$$UFP = \sum_{i=1}^{5} \times \sum_{j=1}^{3} (Z_{ij} W_{ij})$$

Now,

$$UFP = (50 \times 4) + (40 \times 5) + (35 \times 4) + (6 \times 10) + (4 \times 7)$$

$$UFP = 200 + 200 + 140 + 60 + 28 = 628$$

$$FP = UFP \times CAF$$

$$FP = UFP \times (0.65 + 0.01 \times \sum F_j)$$

$$FP = 628 \times (0.65 + 0.01 \times (14 \times 3))$$
$$FP = 628 \times (0.65 + 0.42)$$
$$FP = 628 \times (1.07)$$
$$FP = 672$$

Chapter 3 : Risk Management [Total Marks - 10]

Q. 5(b) What are Software risks? Write a note on RMMM for delayed projects. **(10 Marks)**

Ans. : Software risks :

When software development process is started, it is main job of a software manager to look into all types of possible risks involved in the entire process. It involves focusing on the possible risks that could affect the project development process :

1. Schedule of the development process
2. Quality of the application under construction

It is the responsibility of a project manager to look into the matter and take necessary action to avoid these risks. All the results of risk analysis and analysis of consequences of risk occurring should be well documented in the planning of the project itself. If the risk management is effective, then it becomes easier to handle all the problems and it is ensured that the project schedules and budget are within acceptable limits and there is no schedule slippage and ultimately no budget slippage. The always threaten the project development processes and the software under construction. Following are three important categories of risk :

1. **Project risks :** These are the risks that directly affect the schedule of the project and the resources involved in the development process.
   Example of project loss : Loss of an experienced developer and designer.
2. **Product risks :** The product risks affect the quality and performance of the application built.
   Example of product risks : Failure of a purchased component to perform as per expectation.
3. **Business risks :** The business risks are affecting the organization those develop and process the software. **Example of business risks :** A competitor of the organization introducing a new product.

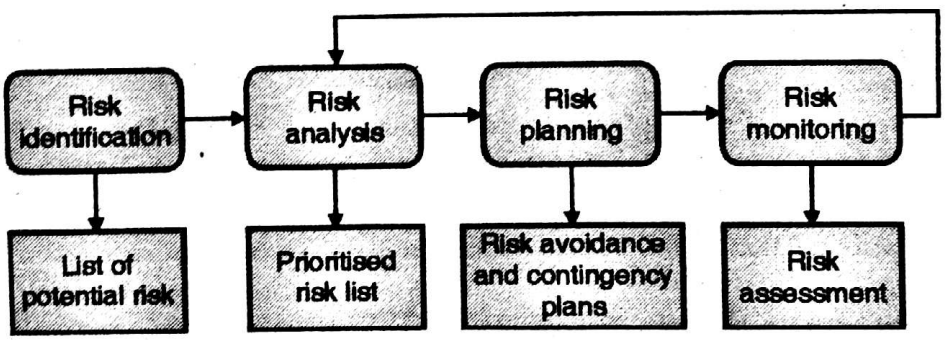


Fig. 4 : The Risks management process

Risk analysis actually helps the project development team to build a strategy to handle all possible risks.

RMMM (Risk Mitigation, Monitoring and Planning) :

Risk analysis actually helps the project development team to build a strategy to handle all possible risks. Following are three important issues (or steps) that must be considered for developing effective strategies :

1. Risk avoidance (i.e. Risk mitigation) 2. Risk monitoring

3. Risk management and planning.

In order to avoid the risk, the best approach is proactive approach. In the proactive approach, the development process starts with risk mitigation plan and it helps in avoiding possible risks. The step is risk monitoring that begins from the start of the development process. The project manager is generally responsible for keeping vigilance on the factors that can cause risk to occur. The project manager starts monitoring with the information received from the risk mitigation step. He scans all the documents prepared in the risk mitigation step. The third and final step is risk management and planning which assumes that the risk has occurred and accordingly the manager has to take necessary action. The Risk Mitigation, Monitoring and Management (RMMM) steps incur extra project cost.

Chapter 4 : Software Configuration Management [Total Marks - 10]

Q. 2(b) Explain Change Control and Version Control in SCM. **(10 Marks)**

Ans. : Version control : The version control actually controls the new releases or new versions. It combines the procedures and tools in order to control various versions of configuration objects. Any version control management system has four major capabilities that is integrated in the version control system itself :

1. The repository will store all the related configuration objects.
2. The repository will store all the versions of configuration objects.
3. It has a facility to provide the related information about configuration objects so that a software engineer will construct a new version based on those information.
4. To track all the issues in development process by using a special tracking facility in the version control.

Change control :

In a development of a larger software system, the changes may be uncontrolled and it leads to a complex situation. In such projects the change control is done partially by human and partially by some automated tools. In such a complex situation human intervention is very much necessary. The change control process is explained in the following Fig. 5. The change request is first submitted and then evaluated by a technical support staff by taking into consideration its potential side effects and the overall impact on other objects in the product. The other parameters to be evaluated are system functions, the cost of project etc. Based on the result of the evaluation treated as a change report, the implementation is taken into consideration. This report is submitted by change control authority i.e. CCA. The CCA is a person or a group who has the final authority to take decision on any changes and their priority.

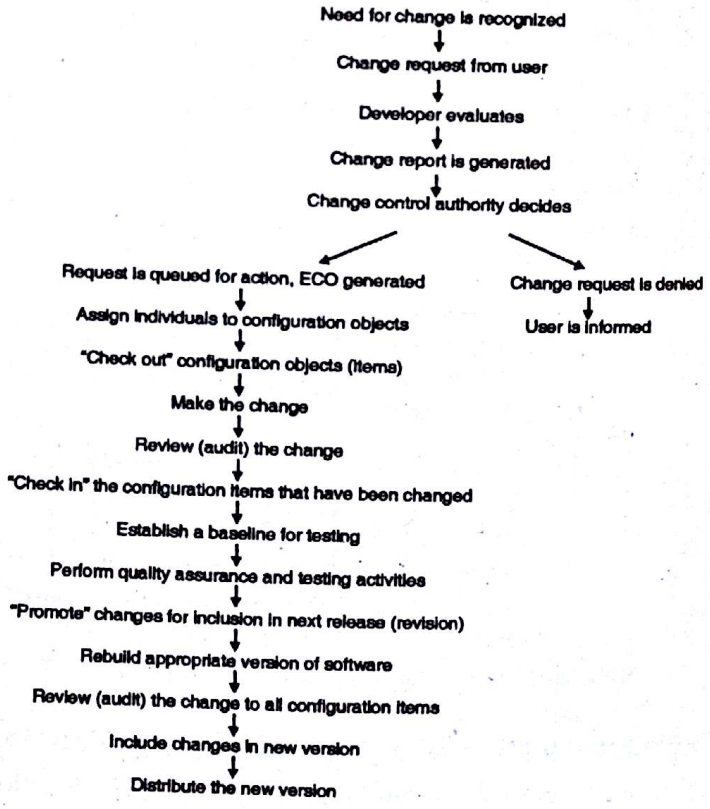


Fig. 5 : The change control process

After approval from CCA, a change order called ECO (Engineering Change Order) is generated for each of the change. The object to be changed can be placed in a directory that is controlled solely by the software engineer making the change. These version control mechanisms, integrated within the change control process, implement two important elements of change management :

1. Access control and 2. Synchronization control

Before an SCI become a baseline, the changes should be applied. The developer will look after whether the changes are justified or not by project. The technical requirement must check properly. After approval from CCA, a baseline may be created and change control is implemented. Once the final product is released, the formal changes must be instituted as per the Fig. 5. This formal change is outlined. The CCA plays an active role in second and third layers of change control based on size of the project.

Chapter 5 : Software Design Specification [Total Marks - 20]

Q. 3(b) What is Coupling and Cohesion? Explain different forms of it. **(10 Marks)**

Ans. : Cohesion :

Cohesion is an extension of the concept of information hiding. In cohesion, a module performs only one task within a software procedure i.e. requires only a little interaction with procedures that are in other parts of a program. Cohesion can be represented as a "spectrum". High collision is always recommended but mid-range of the spectrum is often acceptable. The cohesion scale is nonlinear. That is, low-end cohesiveness is worse than middle range, which is nearly same high-end cohesion. Designers are not required to be bothered about categorizing cohesion in a particular module. Rather, the overall concept should be understood properly and low-end cohesion must be avoided when modules are designed. At the low-end of the spectrum which is actually not desirable, we encounter a module that performs different tasks that are loosely related to each other. These modules are termed as coincidentally cohesive. A module that performs tasks that are related logically is logically cohesive. When a module contains tasks that are related by the fact that all must be executed with the same span of time, the module exhibits temporal cohesion. Moderate levels of cohesion are close to each other in the degree of module independence. Procedural cohesion exists when processing elements of a module are related and are executed in a specific order. When all the processing elements do concentrate on only one area of a data structure, then communicational cohesion is available. High-end cohesion is characterized by a module that performs one single and unique procedural task.

Coupling :

Coupling is a measure of inter-relationship among various modules in a software structure. Coupling always depends on the interface complexity among modules i.e. the point at which reference or entry is made to a module. It also depends on what data is passed across the interface among modules. In software design, we look for minimum possible coupling. If the connectivity between different modules of software is made simple then it is bit easier to understand.

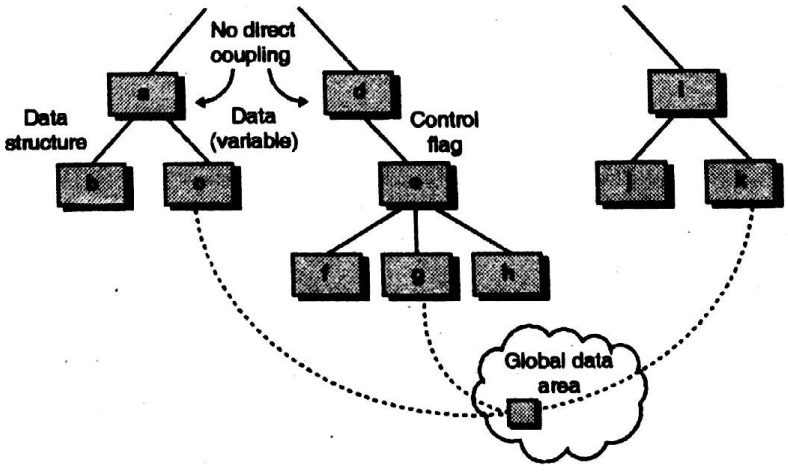


Fig. 6 : Types of coupling

The simple connectivity will also avoid "ripple effect" up to certain extent. The ripple effect is introduced when the errors occurring at one particular location in the system and propagating through a system.

Different forms of coupling :

In the Fig. 6, various examples of different types of module coupling are illustrated. Modules **a** and **d** are children of different modules. Each of them is not related to each other and thus there is no direct coupling occurs. Module **c** is child of module **a.** It is accessed through a conventional argument list. All the data are passed through this argument list. When the argument list is simple, the data can be passed and there is one to one correspondence between the items exist. The low data coupling or simply low coupling is demonstrated through this structure. When a part of data structure apart from simple argument, is passed through a module interface, then stamp coupling is exhibited. This is actually a variation of data coupling only. It is illustrated as shown in Fig. 6 and it occurs between **b** and **a**.

The coupling is characterized by passage of control between modules at moderate levels. In most of the software designs, control coupling is common and is exhibited in Fig. 6 where a "control flag" is passed between modules d and e. When modules are tied to some environments that are external to software, then the high levels of coupling may occur. Like an I/O can couple a module to some devices, formats, and different communication protocols. This is external coupling and is very essential. It must be limited to only a small number of modules in a structure. Like low coupling, high coupling my also occurs when different modules reference a global data area. This is called common coupling also, and is shown in Fig. 6 modules **c, g,** and **k** share their data item in a global data area. The module **c** initializes the items. After that the module **g** recomputes and later on updates these items. An error has been occurred and module **g** updates the item incorrectly.

In processing the module, **k** reads the wrong item and during its processing, it fails and ultimately the software is aborted. The reason behind this abort is module **k** and the actual cause is module **g** since it has updated the item incorrectly. Diagnosing the problems is time consuming and difficult. The diagnosis in structures is always with the common coupling. The use of global data is not always bad, but the developer may use them in coupling. The designer should know well the impacts of these couplings and care against them to protect. The coupling occurs when a module uses the data present in some another module. It can use control information also during the coupling. The concept of content coupling used when all the subroutines or data structures within a module are compulsory asked to involve in coupling. The content coupling must be avoided. The content coupling modes may occur due to design decisions made during development of structure.

For example : Compiler coupling ties source code to specific (and often nonstandard) attributes of a compiler; Operating System (OS) coupling ties design and resultant code to operating system "hooks" that can create havoc when OS changes occur.

Q. 4(a) What are the features of a good user Interface? Design and interface for Online Air Ticket Reservation System. **(10 Marks)**

Ans. :

Features of a good user interface :

1.	Clarity	2.	Concision	3.	Familiarity
4.	Responsiveness	5.	Consistency	6.	Aesthetics
7.	Efficiency	8.	Attractive	9.	Forgiveness

3. **Usability :** The software usability is defined as the degree to which the application software is easy to use. It defines the efforts taken to operate and learn, submit input and interpret output.
4. **Efficiency :** It is defined as the amount of various resources required to execute a given program. The computing resources are memory devices, processing devices and input output devices etc.
5. **Maintainability :** It is defined as the effort needed to uncover the errors and the problems in functioning of a program and resolve these errors.
6. **Integrity :** The integrity of a system is defined as the efforts taken to manage access authorization. The integrity of a system can be ensured by controlling all the accesses to it.
7. **Flexibility :** The ease with which a program can be modified and extended in future.
8. **Testability :** Amount of time and the efforts taken to test a program.
9. **Portability :** Shifting a program from one environment to another.
10. **Reusability :** How existing codes can be reused in future developments according to the scope of the software.
11. **Interoperability :** Ability to connect one system to the another system.

(II) ISO 9126 Quality factors :

The ISO 9126 standard was developed in an attempt to identify quality attributes for computer software. The standard identifies six key quality attributes :

1. **Reliability :** The software should be available for the use and should satisfy following key attributes :
   (i) Maturity
   (ii) Fault tolerance i.e. ability to withstand against failures
   (iii) Recoverability i.e. the system should regain its original state once the failure occurs. It should also ensure the data integrity and consistency while recovering from failure states.
2. **Portability :** Portability means the system should be able to work in different hardware and software environments according to the following attributes :
   (i) Installability (ii) Adaptability
   (iii) Conformance (iv) Replaceability
3. **Efficiency :** Efficiency means making the optimal use of the system resources like memory devices, processing devices and input output devices etc. It should take into consideration following attributes : Time behaviour and Resource behaviour.
4. **Maintainability :** Maintainability is defined as the ease with which the software may be repaired and ready to use once again. The maintainability should take into consideration following attributes :
   (i) Changeability (ii) Testability
   (iii) Stability and (iv) Analyzability
5. **Functionality :** The degree to which the application software satisfies the customer's needs and requirements according to the following attributes :
   (i) Accuracy of working model (ii) Suitability of the product
   (iii) Interoperability (i.e. the product can be connected to any system),
   (iv) Compliance and (v) Security
6. **Usability :** The software usability is defined as the degree to which the application software is easy to use according to the following usability attributes :
   (i) Understandability of the software by the end-uses

(ii) Learnability means leaning the use of the product and Operability.

(iii) The ISO 9126 factors do not support direct measurement of quality, but provides indirect measurement. It provides a very good checklist for evaluating the quality.

Chapter 7 : Software Testing [Total Marks - 30]

Q. 1(b) Compare verification and validation testing. **(5 Marks)**

Ans. : Comparison between verification and validation :

Sr. No.	Verification	Validation
1.	Verification is a set of activities that ensures that all customer's requirements are fulfilled by the software and all the functions are implemented and working correctly.	The validation ensures that the software is developed and it is traceable to all the customer's requirements.
2.	The Verification is basically one of the elements of software testing.	The Validation is also one of the elements of software testing.
3.	Verification means evaluating whether the developer build product in a right way ?	Validation means evaluating whether developer built right product ?
4.	Verification is static.	Validation is dynamic.
5.	Verification evaluates all the documentation, the planning, the coding, requirements and specifications.	Validation means the evaluation of the complete product itself.
6.	Verification takes place before validation.	Not vice versa.
7.	The inputs for verification are : Technical meetings, various issues, review meeting etc.	The input for validation are : the complete testing of the product itself.
8.	The output of verification : Flawless documents, perfect plans and nearly completed specifications along with requirement.	The output of validation : The perfect final product.

Q. 1(d) Explain the different types of software Maintenance. **(5 Marks)**

Ans. :

Types of software maintenance :

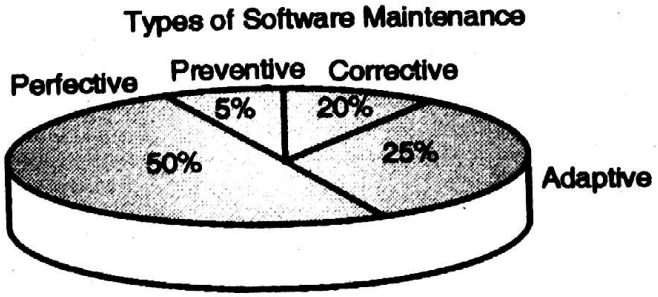


Fig. 9 : Types of maintenance

Following are four types of maintenance :

1. **Corrective maintenance :** The corrective maintenance is the type of maintenance in which the errors are fixed when it is observed during the use of the software. In this the errors may be caused due to faulty software design, incorrect logic and improper coding. All these errors are called as residual errors and they prevent the final specification. In case of system failure, the corrective maintenance approach is initiated to locate the original specification. Corrective maintenance normally used for more than 20% of all the maintenance activities.
2. Adaptive maintenance : The adaptive maintenance means the implementation of the modification in the system. The changes may be of hardware or the operating system environments.

   The examples of adaptive maintenance are :

   1. Business rules 2. Work patterns
   3. Government policies that have substantial impact on the software.

   Adaptive maintenance normally used for more than 25% of all the maintenance activities.
3. **Perfective maintenance :** Perfective maintenance deals with the modified and changed user requirements. The functional enhancements are taken into consideration in Perfective maintenance. In Perfective maintenance, the function and efficiency of the code is continuously improved.

   Following are the examples of perfective maintenance :

   1. Modifying the payroll program to add new union settlement
   2. Adding a new report in the sales analysis system

   Perfective maintenance normally used for more than 50% of all the maintenance activities and it is the largest among all.
4. **Preventive maintenance :** Preventive maintenance is used to prevent the possible errors to occur. Thus in this activity, the complexity is minimized and the quality of the program is enhanced. Adaptive maintenance normally used for 5% of all the maintenance activities and it is the smallest among all.

Q. 6(a) Compare Black box and White Box Testing. Find cyclomatic complexity of following code.

```
IF A = 10 THEN, IF B > C THEN
A=B ELSE A = C
END IF
END IF
PRINT A
PRINT B
PRINT C
```

(10 Marks)

Ans. : Comparison between white-box and black-box testing :

Sr. No.	White box testing	Black box testing
1.	White-box testing also called as glass-box testing. It employs the test case design concept that uses the control structures. These control structures are described as component-level design to derive the test cases.	Black-box testing is also known as behavioural testing. It focuses only on the functional requirements of the software.

Sr. No.	White box testing	Black box testing
2.	It guarantees the traversal of all independent paths in a module at least once. It refers the concept of traversal of tree i.e. each of the nodes in a tree must be traversed at least once.	The black-box testing helps the software developer to derive the sets of input conditions. These input conditions exercise all the functional requirements for a program.
3.	It evaluates all logical decisions and finds whether they are true or false. It evaluates all the loops to check their boundaries and operational bounds. It evaluates all internal data structures to confirm their validity.	It evaluates performance errors or the behavioural errors. It evaluates missing or incorrectly defined functions. It evaluates incorrect data structures and external database errors. It evaluates errors occurred during initialization and termination and interface errors also.
4.	The white box testing begins early in the testing process.	The black box testing is applied in the final stages of testing
5.	Different white-box testing methods are : Basis Path Testing, Control Structure Testing	Different black-box testing methods are : Graph-Based Testing Method, Equivalence Partitioning, Boundary Value Analysis, Orthogonal Array Testing

Cyclomatic complexity :

The Cyclomatic Complexity is calculated as follows :

Cyclomatic complexity = E – N + P

where,

E = Number of edges in the flow graph.

N = Number of nodes in the flow graph.

P = Number of nodes that have exit points

Flow graph :

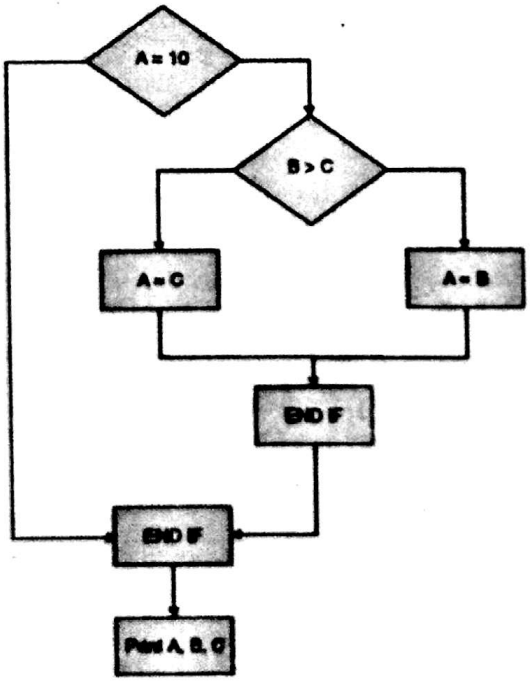


Fig. 10

Online Air Ticket reservation system :

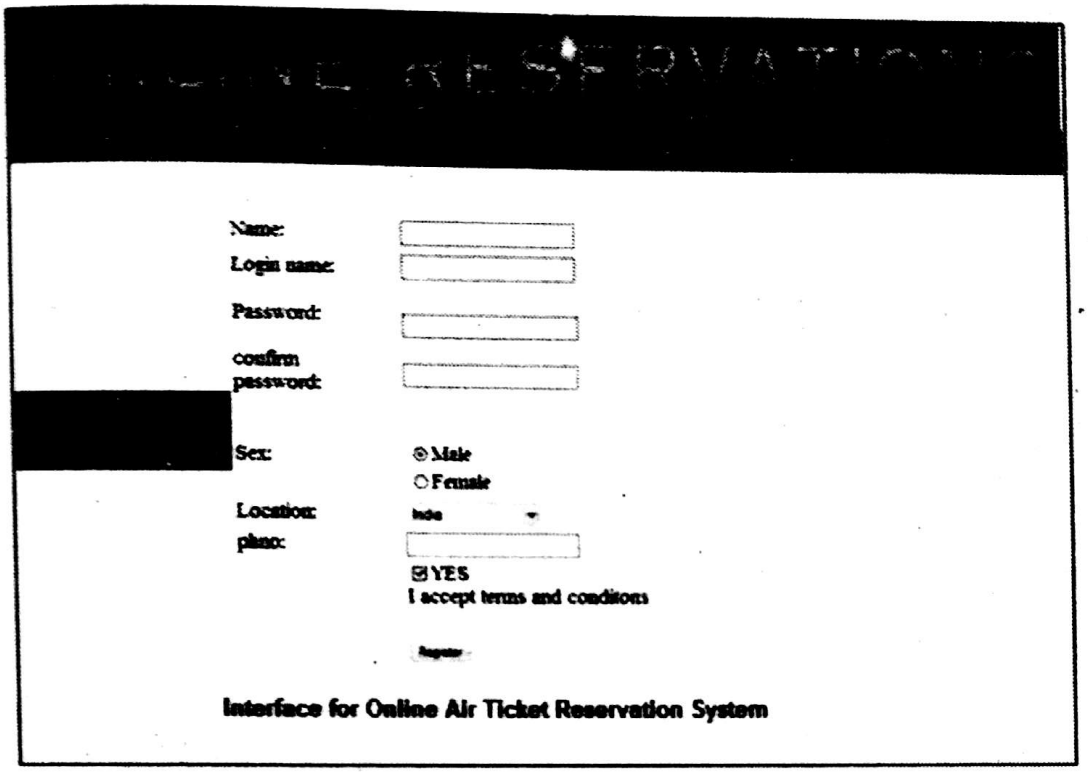


Fig. 7

Chapter 6 : Software Quality [Total Marks - 10]

Q. 4(b) Explain different metrics used for maintaining Software Quality. **(10 Marks)**

Ans. : Different metrics used for maintaining software quality :

(i) McCall's Quality factors :

The factors that affect software quality can be categorized in two broad groups :

Factors that can be directly measured (e.g., defects uncovered during testing) and Factors that can be measured only indirectly (e.g., usability or maintainability) The categorization of factors that affect software quality shown in Fig. 8, focus on three important aspects of a software product :

1. Its operational characteristics,
2. Its ability to undergo change and
3. Its adaptability to new environments.

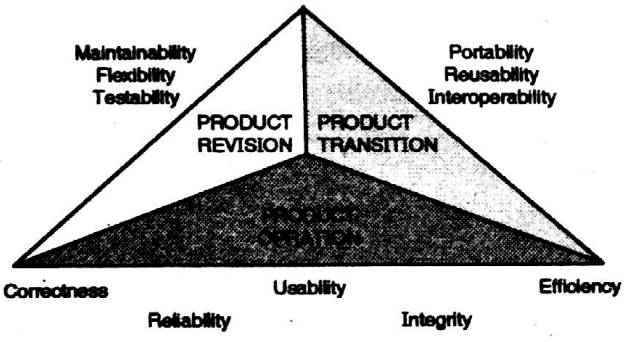


Fig. 8 : McCall's software quality factors

As shown in Fig. 8, provide the following factors :

1. **Correctness :** The correctness is the quality factor that assesses the software for its correct functioning according to the specification and customer's requirements.
2. **Reliability :** It is the quality attribute that evaluates the software for its working according to the desired precision.

The Cyclomatic complexity is calculated using the above control flow diagram that shows seven nodes (shapes) and eight edges (lines), hence the cyclomatic complexity is 8 - 7 + 2 = 3

Q. 6(b) Explain software Reverse Engineering in detail. **(10 Marks)**

Ans. : Software reverse engineering :

The Reverse Engineering is the discipline of software engineering, where the knowledge and design information is extracted from the source code or it is reproduced. The Reverse Engineering is a process where the system is analyzed at higher level of abstraction. It is also called as going backward through all the development cycles. Following are some important purposes of Reverse Engineering :

1. Security auditing
2. Enabling additional features
3. Used as learning tools
4. Developing compatible products cheaper than that are currently available in the market.

Following are three important parameters to be considered for of a reverse engineering process :

1. **Abstraction level :** In the abstraction level of a reverse engineering process, the design information is extracted from the source code. It is the highest level in the reverse engineering process. It is always expected that the abstraction level for any reverse engineering process must be high. When the level of abstraction is high, then it becomes easy for the software developer to understand the program.
2. **Completeness :** The completeness is nothing but the details available through the abstraction level of reverse engineering process. For example from the given source code it is very easy to develop the complete procedural design. .
3. **Directionality :** The directionality of a reverse engineering process is a method of extracting information from the source code and making it available to the software developers. The software developers can use this information during the maintenance activity. Following Fig. 11 exhibits the reverse engineering process.

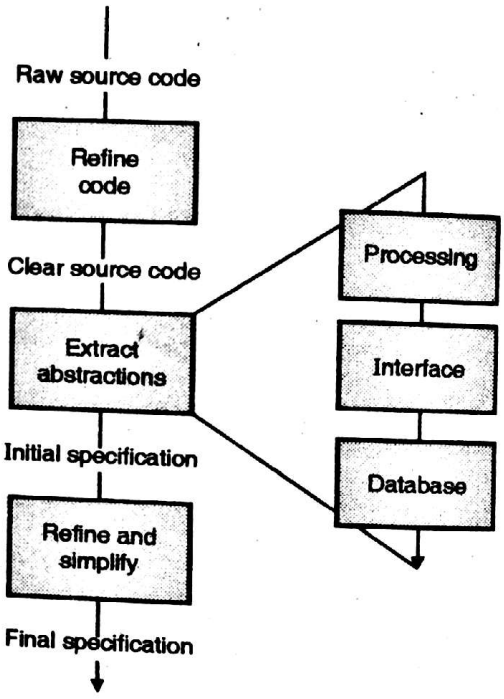


Fig. 11 : A Reverse Engineering Process

The reverse engineering process include the following steps :

1. The raw or dirty source code obtained from the abstraction level is taken as input.
2. This code is refined or restructured. (Clean code is obtained)
3. From the clean code, the abstraction is extracted.
4. From this initial specification, the code is refined and simplified
5. Now, we get the final specification.

Thus the final specification obtained from the reverse engineering process, is used as final specification by the software developers.

□□□

Dec. 2015

Chapter 1 : Software Engineering Process [Total Marks - 35]

Q. 1 A distance learning institute decides to use e-learning software to ease its regular functioning of the program. Through this e-learning tool students can register to various courses, appear for online exams, download study material, upload assignments online, view lecture videos etc. The faculty can upload study materials, conduct exams, teach one or many courses. The institute can check student and faculty information, collect fees, pay salary, display results and so on. Create an SRS for the institute that includes the following

1. Product perspective
2. Scope and objective
3. Functional requirements (at least 3)
4. Non-functional requirements

(20 Marks)

Ans. : The SRS for an e-Learning Institute can be as follows :

(a) **Product perspective :** The proposed system is a solution carry out online lecture videos for different courses with study material(notes), online exams.

(b) **Scope and objective :** The document is the one that describes the requirements along with interfaces for the system. It is meant for use by the developers and will be the basis for validating the final delivered system. The system helps in buying of goods, products and services online by choosing the listed products from website

(c) **Functional Requirements :**

1. **Product :** Product includes the information about different courses, related video lectures, study material, online test, assignments with this students and faculty information which are to be displayed on the website.
2. **Price :** Price deals with the fees of the course, discounts applicable for the particular course as per institute declared notice.
3. **Transactions :** All transactions undergoing in the website will be controlled and managed by this module. Transactions in the sense, fees collection and salary of faculty members .

(d) **Non-Functional Requirements :**

(1) **User Interfaces :** Each part of the user interface intends to be as user friendly as possible. The fonts and buttons used will be intended to be very fast and easy to load on web pages. The pages will be kept light in space so that it won't take a long time for the page to load.

(2) **Communication Interface :** The Website Order system shall send an e-mail confirmation to the students regarding fees, results, marketing of particular course and also to faculty members for salary conformation.

Q. 2(a) Define Software Engineering. Explain in brief the software process framework. **(5 Marks)**

Ans. : Software Engineering

The term software engineering is defined as :

"By using the principles of sound engineering and its establishment, software is developed that should be economical and should work efficiently on real machines." Software engineering is a systematic and disciplined approach towards developments of software. The approach must be systematic for operation of the software and the maintenance of it too.

Software engineering is considered as a layered technology.

These layers includes :

1. Quality focus
2. Process
3. Methods
4. Tools

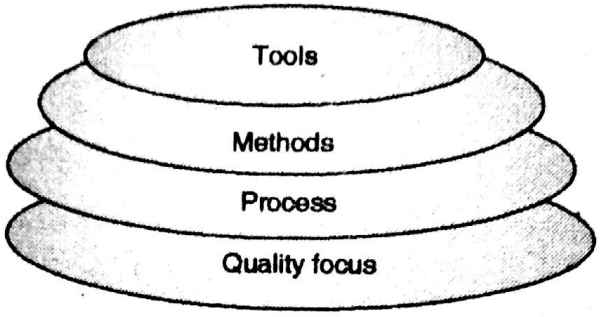


Fig. 1-Q. 2(a) : Software engineering layers

Software Process Framework :

A software process is collection of various activities. There are five generic process framework activities :

1. Communication
2. Planning
3. Modeling
4. Construction
5. Deployment

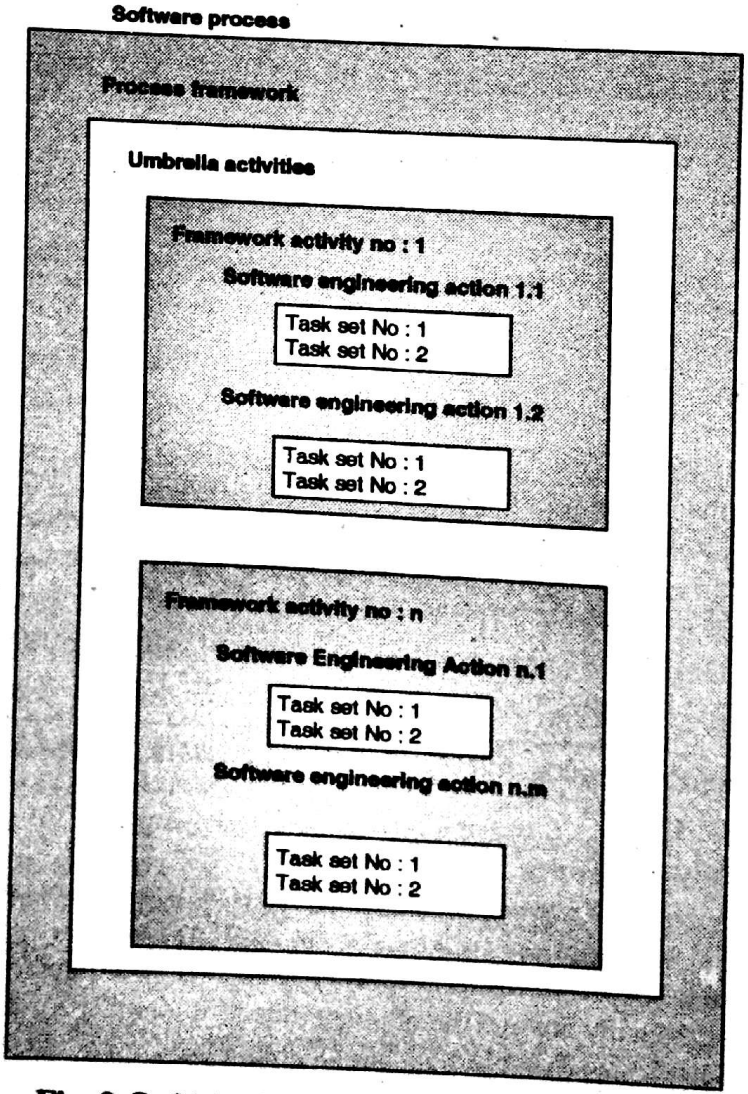


Fig. 2-Q. 2(a) : A software process framework

Q. 4(a) What is Agility in context of software engineering? Explain Extreme Programming (XP) with suitable diagram. **(10 Marks)**

Ans. : **Please refer Q. 2(a) of May 2015.**

Chapter 3 : Risk Management [Total Marks - 15]

Q. 2(d) Discuss the different categories of risk that help to define impact values in a risk table. **(5 Marks)**

Ans. : Risk Management

Whenever we start any business or any development process, we take into consideration the risks involved in accomplishing that task. Similarly when software development process is started, it is main job of a software manager to look into all types of possible risks involved in the entire process.

It involves focusing on the possible risks that could affect the project development process :

1. Schedule of the development process
2. Quality of the application under construction

It is the responsibility of a project manager to look into the matter and take necessary action to avoid these risks. All the results of risk analysis and analysis of consequences of risk occurring should be well documented in the planning of the project itself. If the risk management is effective, then it becomes easier to handle all the problems and it is ensured that the project schedules and budget are within acceptable limits and there is no schedule slippage and ultimately no budget slippage.

The always threaten the project development processes and the software under construction :

Following are three important categories of risk :

1. **Project risks :** These are the risks that directly affect the schedule of the project and the resources involved in the development process. Example of project loss : Loss of an experienced developer and designer.
2. **Product risks :** The product risks affect the quality and performance of the application built. Example of product risks : Failure of a purchased component to perform as per expectation.
3. **Business risks :** The business risks are affecting the organization those develop and process the software. Example of business risks : A competitor of the organization introducing a new product.

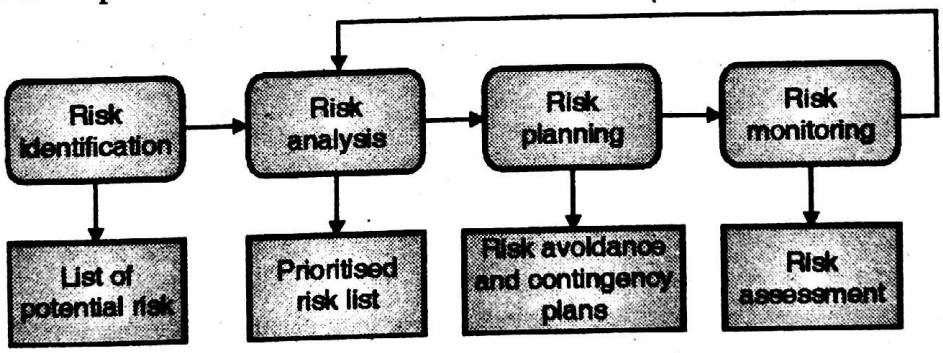


Fig. 1-Q. 2(d) : The Risks management process.

Developing a Risk Table

A risk table gives very useful technique to project managers for the risk projection. Table 1-Q. 2(d) is an example of risk table.

Table 1-Q. 2(d) : A sample risk table

Risks	Category	Probability	Impact	RMMM
Size estimate may be significantly low	PS	60%	2	
Larger number of users than planned	PS	30%	3	
Less reuse then planned	PS	70%	2	
End-users resist system	BU	40%	3	
Delivery deadline will be tightened	BU	50%	2	
Funding will be lost	CU	40%	1	
Customer will change requirement	PS	80%	2	
Technology will not meet expectations	TE	30%	1	
Lack of training on tools	DE	80%	3	
Staff inexperienced	ST	30%	2	
Staff turnover will be high	ST	60%	2	

In the above table the impact values are as follows :

1. For the disastrous situations :- - > 1
2. For some critical situations :- - > 2
3. For the average situations :- - > 3
4. For the negligible situations :- - > 4

In the above table the categories used are as follows :

1. PS used for project size risk
2. BU used for business risk
3. CU used loss of funds etc.

The risk table is prepared right in the beginning of the development process. It is very much useful in analyzing the various types of risks associated. The risks are categorized in different categories as shown in Table 1-Q. 2(d). The probability of occurrence of particular risks is also estimated and associated impact values are listed in the above table. The project manager goes through this table and gives a cutoff line. All the risks that fall below the cutoff line are evaluated again. Similarly the risks that are above the cutoff line are properly managed by the project manager.

Q. 5(a) Explain the various steps in Risk Management with suitable diagram. Identify the risks associated with delayed projects. **(10 Marks)**

Ans. : Various steps in Risk Management

Risk management is one of the most important jobs for a project manager. Risk management involves anticipating risk that might affect the project schedule or the quality of the software being developed, and then taking action to avoid these risks. Risk is something that prefer not to happen. Risks may threaten the project, the software that is being developed, or the organization.

Following are the various steps involved in Risk management :

1. **Risks identification :** Risk identification can be done by identifying the known and predictable risk.
2. **Risk analysis :** Risk can be analyze by assessing consequences of problem associated with risk.
3. **Risk planning :** Making plan to address the risk, either by avoiding it or minimizing its effect on the project.

Risk monitoring means regularly assessing the risk and plans for risk mitigation. The risk management process is an iterative process that continues throughout the project. Once initial risk management plan is drawn up, it will help to monitor the situation to detect the emerging risks. As more information about the risks become available, then it will be easier to analyze and decide if the risk priority has changed. Referring it you may then have to change your plans for risk avoidance and contingency management. All these steps can illustrated with the help of Fig. 1-Q. 5(a).

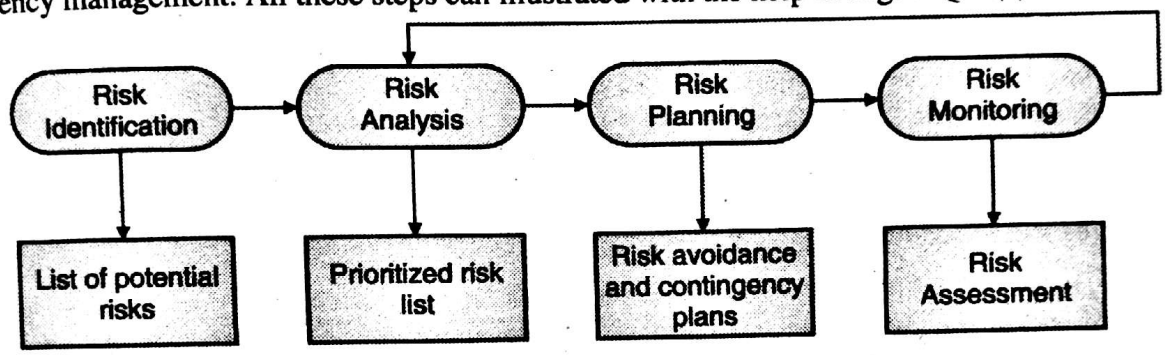


Fig. 1-Q. 5(a) : Steps in Risk Management

Risks associated with delayed projects

Customer will not satisfy with the service provided by service provider. Project development cost will be increased since maximum resources will be used for completion of the project. There might be chances of new requirement from customer because of delay in project delivery.

Chapter 4 : Software Configuration Management [Total Marks - 10]

Q. 6(a) Explain the change control and version control activities in SCM. **(10 Marks)**

Ans. : **Please refer Q. 2(b) of May 2015.**

Chapter 5 : Software Design Specification [Total Marks - 25]

Q. 2(b) Discuss on Modularity and Functional Independence fundamentals of design concepts. **(5 Marks)**

Ans. : Modularity : The modularity consists of software architecture and design patterns i.e. software is divided separately according to name and addresses of components and sometimes it is called as modules that are integrated to fulfill problem requirements. The modularity is defined as the modularization of a single attribute of software into number of small parts such that these parts can be easily managed. The number of variables, span of reference, number of control paths and overall complexity will make the understanding of the program nearly impossible.

Functional Independence fundamentals : The functional independence concept is a type of modularity and related to the concepts of abstraction and information hiding. In functional independence each module addresses a specific sub function of requirements and has a simple interface. This can be viewed from other parts of the program structure.

Independence is assessed using following two qualitative criteria :

1. Cohesion 2. Coupling

Cohesion is a relative functional strength of a module and coupling is the relative interdependence among modules.

1. **Cohesion :** Cohesion is an extension of the concept of information hiding. In cohesion, a module performs only one task within a software procedure i.e. requires only a little interaction with procedures that are in other parts of a program. Cohesion can be represented as a "spectrum".
2. **Coupling :** Coupling is a measure of inter-relationship among various modules in a software structure. Coupling always depends on the interface complexity among modules i.e. the point at which reference or entry is made to a module. It also depends on what data is passed across the interface among modules.

Q. 3(a) Explain in brief the different types of coupling and cohesion. Give one practical example of high cohesion and low coupling. **(10 Marks)**

Ans. : **Please refer Q. 3(b) of May 2015.**

Q. 5(b) Explain different architectural styles with suitable brief example for each. **(10 Marks)**

Ans. : Architectural Styles

The application software that is developed for computer-based systems also shows any one of the architectural styles that are available. Each of these styles explains the system category that consists of :

1. A set of components like a database or the computational modules that perform a function required by the system.
2. A set of connectors used for communication, coordination, and cooperation between the components.
3. The constraints used to define the integration of various components into system.
4. The semantic models used to understand the overall properties of a system with the help of analysis of its elements.

An architectural style is a transformation that may be applied on the design of overall system. The main purpose is to establish a structure for all the components of the system. In the case where architecture is to be redefined, the use of an architectural style will result in fundamental changes to the structure of the software including a reassignment of the functionality of components. An architectural pattern, like an architectural style, applies a transformation on the design of architecture.

However, a pattern may differs from a style in following fundamental ways :

1. The scope of a pattern is relatively small that focuses on only one aspect of the architecture rather the complete architecture.
2. A pattern may apply a rule on the architecture to describe the use of software to handle its functionality at the structural level. For example, concurrency.
3. The Architectural patterns address some specific behavioural issues in it's the context. For example, how the application software handles synchronization or interrupts occurring. These patterns are used with an architectural style to define the shape the overall structure of a system.

Chapter 6 : Software Quality [Total Marks - 10]

Q. 3(b) What is FTR in SQA? What are its objectives? Explain the steps in FTR. **(10 Marks)**

Ans. : FTR in SQA :

The FTRs (Formal Technical Reviews) should be carried out at the end of every life cycle phase. By doing this, the problems will be detected and corrected. It also checks that all the requirements are satisfied.

Examples of formal technical reviews are :

1. PDR (Preliminary Design Review)
2. CDR (Critical Design Review) and
3. TRR (Test Readiness Review).

A technical review takes into consideration overall structure of the software product being developed. All the desired requirements must be fulfilled. The technical reviews are the integral part of the development process. In FTR (formal technical reviews), the actual work completed is compared with standards and procedures already established by the software organization.

Objectives :

Following are the objectives of FTR :

1. To make project more manageable.
2. It is useful to uncover the error in logic, function and implementation for any representation of software.
3. FTR to ensure that software meets specified requirements.
4. To achieve software that is developed in a uniform manner.
5. It is also ensure that software is represented according to predefined standards.

Steps in FTR

1. **The review meeting :** Every review meeting should be conducted by considering the following constraints :
   1. Involvement of people : Between 3 and 5 people should be involve in the review.
   2. Advance preparation Advance preparation should occur but it should be very short that is at the most 2 hours of work for each person can be spent in this preparation
   3. Short duration : The short duration of the review meeting should be less than two hour.

   Rather than attempting to review the entire design walkthrough are conducted for modules or for small group of modules. The focus of the FTR is on 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 for product for its deadlines. The copies of product material is then distributed to reviewers. -The producer organises "walkthrough" the product, explaining the material, while the reviewers raise the issues based on theirs advance preparation. One of the reviewers become recorder who records all the important issues raised during the review. When error 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 modification.
2. **Review reporting and record keeping :** During the FTR, the reviewer actively record all the issues that have been raised. At the end of meeting these all raised issues are consolidated and review issue list is prepared. Finally, formal technical review summary report is produced.
3. **Review guidelines :** Guidelines for the conducting of formal technical review must be established in advance. These guidelines 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 producers.
2. Set an agenda of a review and maintain it.
   1. When certain issues are raised then debate or arguments should be limited. Reviews should not ultimately results in some hard feelings.
   2. Find out problem areas, but don't attempt to solve every problem noted.
   3. Take written notes (it is for record purpose)
   4. Limit the number of participants and insists upon advance preparation.
   5. Develop a checklist for each product that is likely to be reviewed.
   6. Allocate resources and time schedule for FTRs in order to maintain time schedule.
   7. Conduct meaningful trainings for all reviewers in order to make reviews effective.
   8. Reviews earlier reviews which serve as the base for the current review being conducted.

Chapter 7 : Software Testing [Total Marks - 20]

Q. 2(c) Explain cyclomatic complexity. How is it computed? **(5 Marks)**

Ans. : Cyclomatic complexity provides quantitative measure of the logical complexity of the program. The **Cyclomatic complexity** is a software metrics. In the context of the basis path testing, the value of Cyclomatic complexity is calculated and they are the independent paths in the basis set. The Cyclomatic complexity provides an upper bound for the test cases and it guarantees that all the statements are executed at least once. The Cyclomatic complexity is considered as a foundation in graph theory. It is calculated in one of the following three ways :

How many regions are related to the Cyclomatic complexity ?

Cyclomatic complexity V(G) for a graph G is defined as :

$$V(G) = E - N + 2$$

In the above equation, E = the number of flow graph edges, and

N = the number of flow graph nodes.

Cyclomatic complexity, V(G), for a flow graph, G, is also defined as :

$$V(G) = P + 1$$

In the above equation, P = the number of predicate nodes contained in the flow graph G.

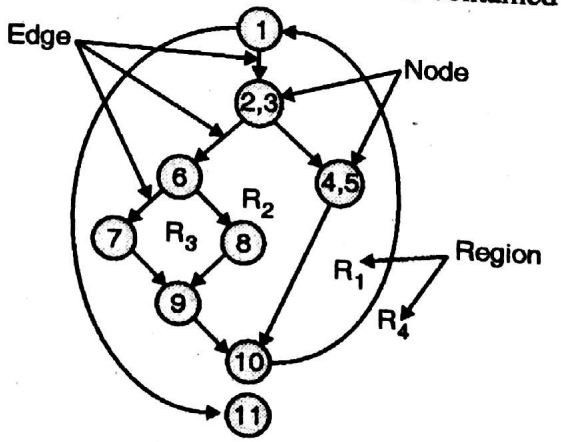


Fig. 1-Q. 2(c)

Referring once more to the flow graph in Fig. 1-Q. 2(c), the Cyclomatic complexity can be computed using each of the algorithms just noted :

1) The flow graph has four regions. 2) V(G) = 11 edges – 9 nodes + 2 = 4.
3) V(G) = 3 predicate nodes +1 = 4.

Q. 2(e) Briefly explain Unit and Integration Testing in the OO Context. **(5 Marks)**

Ans. : Unit Testing

The objective of any testing strategy is simply uncovering the maximum number of possible errors with optimum amount of efforts applied over a realistic time span. In conventional software, we use unit testing and in case of object oriented software, unit testing is referred as class testing. Whenever the object-oriented software is tested, the concept of the unit testing changes as compared to conventional approaches. Encapsulation is preferred in object oriented software and concept of classes is used. Each of classes and their instances are called as objects. The package attributes are the data and operations are nothing but the functions in object oriented context. An encapsulated class is the main focus of unit testing in object oriented context. Operations in each of the classes are the smallest units that can be unit tested.

Integration Testing in the OO Context

Since the object-oriented software does not have any fixed hierarchical control structure, the top-down and bottom-up integration strategies becomes meaningless. Also, the integrating operations one at a time into a class is impossible due to the direct and indirect interactions of the components of that class. Following are two different strategies for integration testing in OO context :

1. The **thread-based testing** combines the set of classes that are required to respond to the input for the system.
2. The **use-based testing** start with the construction of the system and it tests those classes only that uses very few server classes.

Q. 4(b) Explain different techniques in White Box Testing. **(10 Marks)**

Ans. : Different white-box testing methods are :

Basis Path Testing

Basis path testing use the concept of white-box testing. It enables the developer to design test cases in order to derive a logical complexity measure of a procedural design. To define a basis set of execution paths, the developer uses this complexity measure as a guide. In this, test cases designed are guaranteed to execute all the statements in the program at least once.

Following are the examples of basis path testing :

Flow Graph Notation : The flow graph depicts logical control flow using the notation illustrated in Fig. 1-Q. 4(b). Each structured construct has a corresponding flow graph symbol.

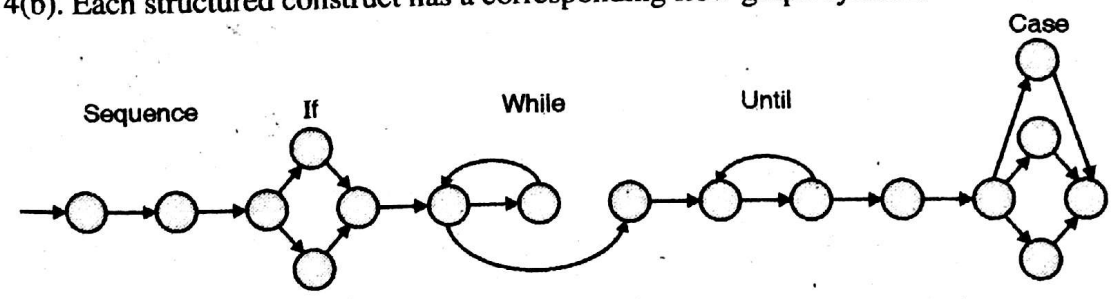
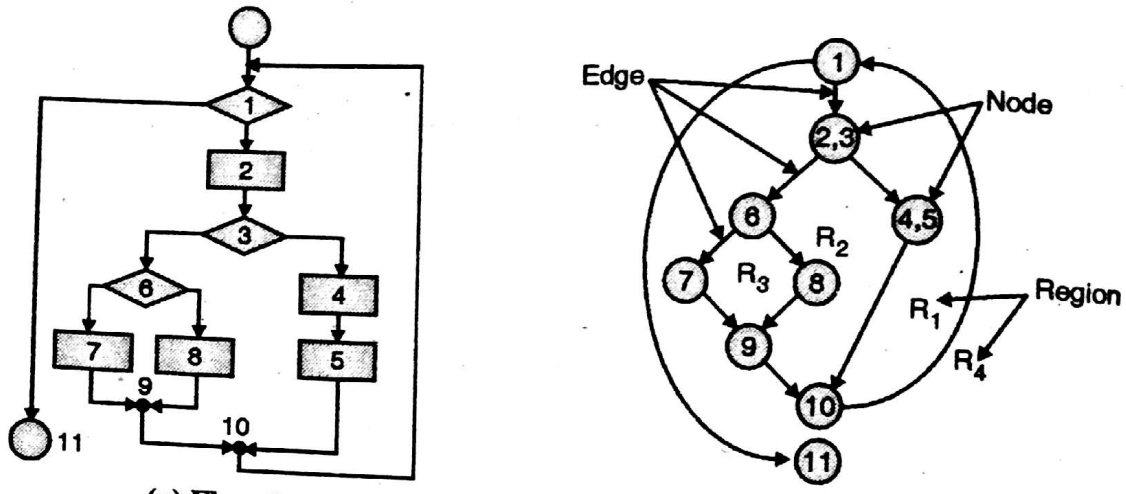


Fig. 1-Q. 4(b) : Flow graph notation

To illustrate the use of a flow graph, we consider the procedural design representation in the Fig. 2(a)- Q. 4(b).



(a) Flowchart **(b) Flow Graph**

Fig. 2-Q. 4(b)

In the Fig. 2-Q. 4(b), to show the program control structure, a flowchart is used. Fig. 2(b)-Q. 4(b) there is mapping of flowchart into a flow graph. Refer Fig. 2(b)-Q. 4(b), to show the flow graph node are represented by circles. These circles represent one or more procedural statements.

Independent Program Paths

An independent path is defined as a path in the program that introduces at least one new set of processing statements or introduces a new condition. When an independent path is stated in terms of a flow graph, it must move along at least one edge and that edge should not have been traversed before defining the path.

Deriving Test Cases

The basis path testing method can be applied to a procedural design or to source code. In this section, we present basis path testing as a series of steps.

The following steps can be applied to derive the basis set :

1. Using the design or code as a foundation, draw a corresponding flow graph.
2. Determine the Cyclomatic complexity of the resultant flow graph.
3. Determine a basis set of linearly independent paths.
4. Prepare test cases that will force execution of each path in the basis set.

One important point to remember is that some of the independent paths cannot be tested alone. But the combination of data is required to traverse the path. This cannot be achieved in the normal flow of the program. In these cases, the independent paths are tested as part of another path test.

Graph Matrices

A graph matrix is a tool that supports basis path testing. A graph matrix is defined as a square matrix whose size is equal to the number of nodes on the flow graph. The size of matrix means the number of rows and columns of the square matrix. The graph matrix is actually a tabular representation of a flow graph. By adding a link weight to each of the matrix entry, it becomes a powerful tool for testing program control structure.

Control Structure Testing

Basis path testing is one of the techniques for control structure testing.

Condition Testing

The condition testing is one of the test case design method. In condition testing, all the logical conditions in the program or the program module are tested. An example of a simple condition may be a Boolean variable or a relational expression. The expression may contain a single NOT operator. Any relational expression has the following simple form :

E_1 < the relational operator > E_2

Where, E_1 and E_2 are two arithmetic expressions and between them, any of the following relational operator may be present :

< → Less than

<= → less than or equal to

≠ → Non-equality

> → Greater than

≥ → Greater than or equal to

A condition may also be a compound condition and it consists of two or more simple conditions, Boolean operators and parentheses.

Data Flow Testing

In dataflow testing approach, a test paths of a program is selected based on the locations of definitions and uses of variables in the program. To explain the data flow testing method, consider that each of statement in a program is assigned a unique number. Also the functions and global variables do not modify their parameters.

Consider a statement S (its unique statement number) as follows :

DEFX(S) = {X | statement S contains a definition of X}

USEX(S) = {X | statement S contains a use of X}

If statement S is an "if" or "loop" statement, then its DEFX set is empty and its USEX set depends on the condition of statement S. The variable X at statement S is the live at statement S', if there is a path from statement S to statement S'. Also it should not contain any other definition of X.

Loop Testing

Loops are the important part of nearly all the programs in the software. Normally the developers pay very little attention this while conducting tests. The loop testing technique is in the category of a white-box testing. It focuses mainly on the validity of loop constructs. In the following section, we define four classes of loops :

1. **Simple loops :** Following tests can be applied to simple loops, where n is the maximum number of passes through the loop :
   Skip the whole loop. Only one pass allowed. Only two passes allowed. m passes allowed through the loop where m < n.
   Also, the possibilities of n – 1, n and n+1 passes allowed.
2. **Concatenated loops :** This concatenated loop approach can be implemented by using the approach discussed in simple loops. In this each of the loops is independent of other. When loops are not independent, this approach is not suitable and instead nested loop approach is suggested.
3. **Nested loops :** The nested loop approach is an extension of simple loop approach. By nesting the loops, the number of testing levels will be increased automatically. This might result in number of tests which may not be practically possible. Following are some approaches that are suggested to reduce the number of tests.

Start with the inner most loops and set all the other loops at minimum levels. Now conduct the simple loop approach to innermost loop while hold other with their minimum interaction value. Move outward to conduct simple test to the next loop and keeping its out loop value at minimum. Continue the same process till all the loops are exercised.

DEF-USE Testing

The DEF-USE (Definition-Use) testing is based on data flow mechanism. It performs testing on definitions and uses of variables in the program statements. The Definition-Use (DU) testing consists of all the definitions and uses of the variables as per the order of their occurrence in the source code. Generally the DU pair is needed in this form of testing. The DU chain is obtained by identifying the definitions and use pairs in a program structure. The DU pair can also be applied on a model such that a test case generation tools can generate the test cases in the model.

The DU pair can be used based on following three coverage criteria :

1. Use every possible definition 2. Get it to every use, and
3. Follow all the DU paths.

Thus all the test cases can be constructed to cover these DU pairs.

Following are some definitions in DU testing :

Definition node, DEF(v,n) is a node in the program where variable v is given its value.

1. Use node, USE(v,n) is a node in the program where the variable v is used.
2. Definition-Use path, or DU path for variable v is a path where DEF(v,i) and USE(v,e) are the initial and end nodes of that path.

Chapter 8 : Web Engineering [Total Marks - 10]

Q. 6(b) Explain TDD with its advantages. **(10 Marks)**

Ans. : Test Driven Development (TDD)

Test driven development (TDD) is a development approach in that testing and coding are combined together. Each of the increment is coded and at the same time it is tested too. Until and unless the last increment (version) is not fully tested, the next increment is not coded. The TDD was introduced as a part of agile software development process, for example extreme programming.

Fig. 1-Q. 6(b) exhibits a general test driven development process :

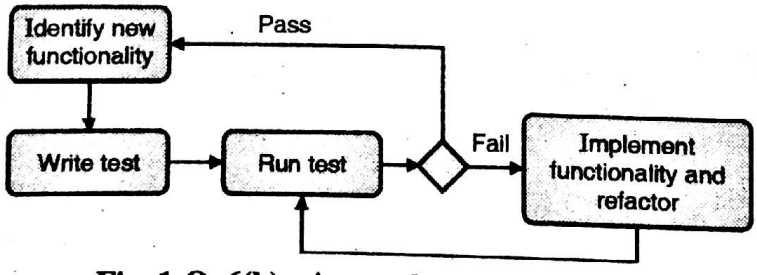


Fig. 1-Q. 6(b) : A sample test-driven development

Following are steps in the test driven development process :

1. Identify the functionality of the increment.
2. Write the proper test for the functionality identified and implement it.
3. Run the test written for the functionality.
4. Refactor the code to improve it.
5. After executing all the test, move to the next set of functionalities.

This may be implemented by using automated testing environment, e.g. JUnit environment that supports Java program testing.

Advantages of test driven development

Following are some important benefits of test driven development :

1. **Code coverage :** All the segments of code must have an associated test. Therefore the errors are detected early in the coding process itself.
2. **Regression testing :** The regression testing is used to find that the program has not introduced any new errors after conducting the test.
3. **Simplified debugging :** The test driven development process normally avoids the automated testing tools due to simplified debugging processes.
4. **System documentation :** The test plan itself acts as a document. After reading the test, one can easily understand the code.

❑❑❑

May 2016

Chapter 1 : Software Engineering Process [Total Marks - 40]

Q. 1 Develop the SRS for the following scenario : A school has one or more departments. Department offers one or more subjects. A particular subject will be offered by only one department. Department has instructors and instructors can work for one or more departments. Students can enrol in upto 5 subjects in a school. Instructor can teach up to 3 subjects. The same subject can be taught by the different instructors. Students can be enrolled in more than one school. SRS for the school should include the following :

a. Product perspective ;

b. Scope and objective ;

c. Functional requirements;

d. Non-functional requirements **(20 Marks)**

Ans. : The SRS for an school can be as follows :

(a) Product perspective : The proposed system is a solution carry out detailed information about departments, students, instructors of school.

(b) Scope and objective : The document is the one that describes the requirements along with interfaces for the system. The system helps to maintain the record of particular school. In that information about departments, different subjects respected instructors and students.

(c) Functional Requirements :

1. **Product :** Product includes the information about different departments, students, instructor, lectures, tests, assignments with this students and faculty information which are to be displayed on the website. Student can enroll in minimum one subject to maximum five subjects also instructors teach maximum three subjects.
2. **Price :** Price deals with the fees of the course, discounts applicable for the particular course as per school declared notice.
3. **Transactions :** All transactions undergoing in the website will be controlled and managed by this module. Transactions in the sense, fees collection, salary of instructors and other expenses which required for school.

(d) Non-Functional Requirements :

(1) **User Interfaces :** Each part of the user interface intends to be as user friendly as possible. The fonts and buttons used will be intended to be very fast and easy to load on web pages. The pages will be kept light in space so that it won't take a long time for the page to load.

(2) **Communication Interface :** The Website system shall sends an e-mail of salary confirmation to the instructors and fees receipt to the students. The system also generates attendance report and sends to respective HOD of department and students along with there subjects.

Q. 2(a) Explain and compare FTR and walkthrough. **(10 Marks)**

Ans. :

FTR : In FTR (Formal Technical Reviews), the actual work completed is compared with standards and procedures already established by the software organization. It is one of SQA activities.

Walkthrough

A walkthrough is an informal meeting for evaluation or informational purposes. A walkthrough is also a process at an abstract level. It's the process of inspecting software code by following paths through the code (as determined by input conditions and choices made along the way). The purpose of code walkthroughs is to ensure the code fits the purpose. Walkthroughs also offer opportunities to assess an individual or team's competency.

FTR	Walkthrough
FTR is a formal meeting conducted by technical staff.	A walkthrough is an informal meeting for evaluation or informational purposes.
The purpose of meeting is to detect quality problems and suggest improvements.	The purpose of code walkthroughs is to ensure the code fits the purpose.
The technical staff focuses on quality from customer's point of view, i.e. what customer exactly wants.	Walkthroughs offer opportunities to assess an individual or team's competency.
It is planned activity. It requires lot of preparation before initiating.	Its unplanned activity. A walk through is an evaluation process which is an informal meeting, which does not require preparation.

Q. 3(b) What are agile process and its advantages ? Explain any one agile process. **(10 Marks)**

Ans. : Please refer Q. 2(a) of May 2015.

Chapter 2 : Software Project Scheduling, Control and Monitoring [Total Marks - 10]

Q. 2(b) Explain the process of CMM. **(10 Marks)**

Ans. : Process of CMM (Capability Maturity Model)

Capability Maturity Model is a bench-mark for measuring the maturity of an organization's software process. It is a methodology used to develop and refine an organization's software development process. CMM can be used to assess an organization against a scale of five process maturity levels based on certain Key Process Areas (KPA). It describes the maturity of the company based upon the project the company is dealing with and the clients. Each level ranks the organization according to its standardization of processes in the subject area being assessed. In CMMI models with a staged representation, there are five maturity levels designated by the numbers 1 through 5 as shown below :

1. Initial 2. Managed 3. Defined
4. Quantitatively 5. Managed 6. Optimizing

The CMMI presents two types of meta models :

1. As a Continuous model 2. As a Staged model

1. **As a Continuous model :** It describes the process in two dimensions as shown in Fig. 1-Q. 2(b). Each process area (e.g. project planning, requirement management) are assessed against specific goals and practices and is rated according to following capability levels:
2. **Level 0 : Incomplete :** The process area (e.g., requirements management) is either not performed or does not achieve all goals and objectives defined by the CMMI for level 1 capability.
3. **Level 1 : Performed :** All of the specific goals of the process area (as defined by the CMMI) have been satisfied. Work tasks required to produce defined work products are being conducted.

4. **Level 2 : Managed :** All level 1 criteria have been satisfied. In addition, all work associated with the process area conforms to an organizationally defined policy; all people doing the work have access to adequate resources to get the job done.
5. **Level 3 : Defined :** All level 2 criteria have been achieved. Also the process is tailored from organization's set of standard processes according to organization's guidelines.
6. **Level 4 : Quantitatively managed :** All level 3 criteria have been achieved. Also the process is controlled and improved using measurements and quantitative assessment.

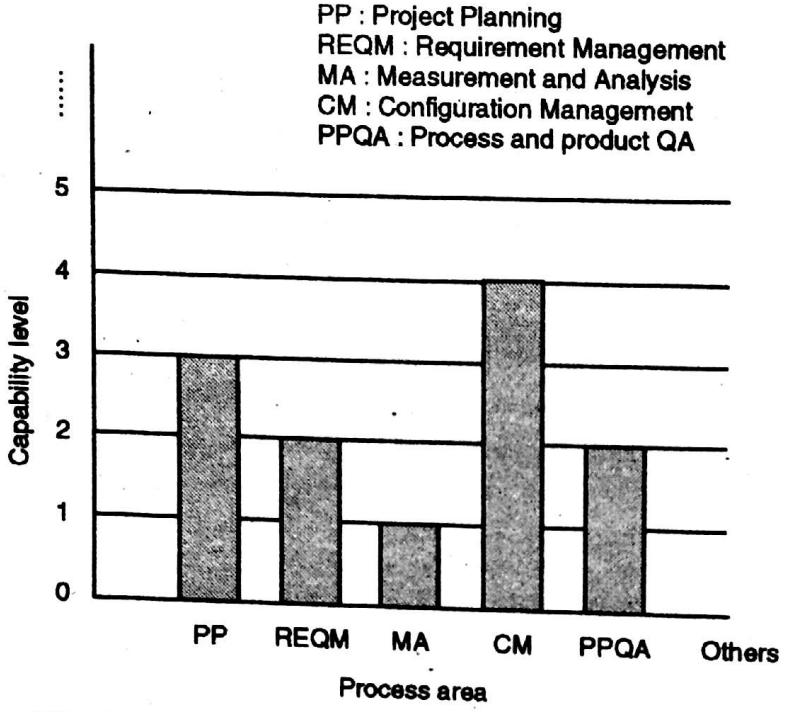


Fig. 1-Q. 2(b) : CMM process area capability profile

7. **Level 5 : Optimized :** All level 4 criteria have been achieved. Also the process area is adapted all standards to meet changing customer's needs.

Chapter 3 : Risk Management [Total Marks - 10]

Q. 6(a) Write short note on : Risk management. **(10 Marks)**

Ans. : **Please refer Q. 2(d) of Dec. 2015.**

Chapter 4 : Software Configuration Management [Total Marks - 10]

Q. 4(a) Explain the change control and version control activities in SCM. **(10 Marks)**

Ans. : **Please refer Q. 2(b) of May 2015.**

Chapter 5 : Software Design Specification [Total Marks - 20]

Q. 3(a) Explain coupling and cohesion. Explain different types of coupling and cohesion. **(10 Marks)**

Ans. : **Please refer Q. 3(b) of May 2015.**

Q. 5(b) What is user interface design process ? Explain with one example. **(10 Marks)**

Ans. : **User Interface Design :** Success of most of the software depends upon user interface design. It is part of human computer interaction. In user interface human controls the software.

As two different entities are trying to communicate with each other, one is human being and other is computer. We need to understand nature of human being designing user interface. Many factors are considered before designing user interface which includes type of user, age of user, education level, purpose of software, etc. User interface design is created by considering human type also like child, grown up person, senior citizen, educated person, non educated persons, etc. Good interface design is user friendly and user can communicate very easily with that software. User interface can be at the hardware and software level also. At hardware level, type of button provided can be considered as user interface. E.g. in washing machine, button provided, display provided can be considered as hardware user interface.

At software level we can provide user interface using programming. Different menus, options, radio button, list boxes, text boxes, button, etc are provided at the software for user interface. If the software is used by illiterate person then user interface should be in terms of pictures and some visual effects. If user is disabled then we need to design interface accordingly. E.g. for blind users we need to provide interface using sound effects. Different biometric factors can be used for user interface design for disabled users. User convenience is given at most priority in user interface design. If interface is designed for mobile users then also we need to consider different factors like type of mobile, type of user, internet bandwidth, battery life, processing power, screen size, type of input, etc. It needs good understanding of user needs. In this topic we focus on user interface design of software and not of the hardware. Various aspects of interface design has been discussed. Different principles, methods are discussed. Plenty of examples are taken for better understanding.

Type of User Interface

It means how user can communicate with computer systems. Two fundamental approaches are used to interface with the computer system. User can communicate with computer system using the command interface and graphical user interfaces. User can communicate with computer using different commands provided by the system. In another approach user interact by using user friendly graphical user interface. Graphical user interface provides the menu and icon based facility which can be used using keyboard and mouse.

1. Command Interpreter
2. Graphical User Interfaces

Characteristics of Good User Interface

There are different characteristics of good user interface :

1. Clarity
2. Concision
3. Familiarity
4. Responsiveness
5. Consistency
6. Aesthetics
7. Efficiency
8. Attractive
9. Forgiveness

Benefits of Good Interface Design

1. Higher revenue
2. Increased user efficiency and satisfaction
3. Reduced development costs
4. Reduced support costs.

Example of User Interface Design Process : (Case study : Ecommerce Grocery Stores)

Selling groceries online is different from selling electronics or apparels. Groceries are 'needs' while electronics/apparels etc. are 'wants'. This is the reason why "groceries" are least sold in e-commerce model compared to other products.

Innovative front end design which would revolutionize the front end design of e-commerce websites. Innovative options like 'You might have ran out of' and 'Most ordered by you' makes shopping very quick. A striking feature in this website would be "multiple products search". Customer can enter whole list of items he/she would like to buy. The site will be designed to show the search results for these items one by one.

This makes the shopping hassle-free unlike other e-commerce websites. The search will be designed for giving correct search results even if the user types the word in local language with English script. Placing Facebook profile pics to increase the loyalty factor. It gives a feel of being a part of the store. It depicts and validates the actual presence of the customer on the site.

Groceries are something that people buy on a regular basis and they are a necessity for daily life (needs) unlike products like t-shirts, perfumes, books etc. (which are wants). Therefore, people do not tend to spend so much time on buying groceries as they spend on other products. Categorization of products is necessary for easing the burden of searching a product, but in the current design we could not categorize the products to significantly simplify the search. We realized that there are about 4 levels of product categories in groceries.

1. Parent category -> Sub category -> Product type -> Product;
2. For example: Personal care -> Body care -> Soaps – >Lux

Consider the Fig. 1-Q. 5(b) that illustrates how sub-category icons filled in. On clicking an icon, a popup shows up with relevant products.

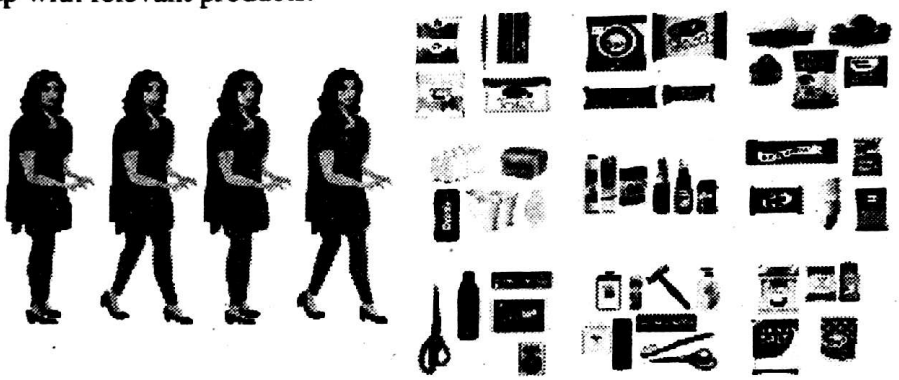


Fig. 1-Q. 5(b)

Chapter 7 : Software Testing [Total Marks - 40]

Q. 4(b) Differentiate between black box testing and white box testing. Explain in detail about any one testing tool. **(10 Marks)**

Ans. : Differentiation between White-box and Black-box Testing :

Please refer Q. 6(a) of May 2015.

Testing Tools (Win runner, Load runner)

We know that testing a software application is how tedious and painful. It consumes lot of time and man hours. Eventually heavy investment is required. Also there are possibilities of mistakes using manual testing. Sometimes it is quite impossible to test all the features of software thoroughly before the final delivery of the product to the client. Sometimes some serious bugs are undetected and uncorrected that causes failure of the product and ultimately failure of the business.

In order to control testing in a proper way, some automated tools are used to uncover the errors and correct them accordingly. The examples of these automated tools are as follows :

1. WinRunner 2. LoadRunner

By using these automated tools, all the problem that are encountered using manual testing can be resolved properly. This actually speeds up the process and minimizing man power involved in testing the application. It helps in reducing the development cost. With WinRunner and LoadRunner tools you can save time by running the test batches overnight.

Q. 5(a) What are the different types of maintenance and also explain steps for creating a maintenance log ? **(10 Marks)**

Ans. : Types of maintenance : Please refer Q. 1(d) of May 2015.

Steps for creating a maintenance log

A log file, in a computing context, is the automatically produced and time-stamped documentation of events relevant to a particular system. Virtually all software applications and systems produce log files. For maintenance activities in software applications, log files have its own significance. Following are some steps for creating a maintenance log file :

1. Make all entries of data exist in the database for the application under maintenance in the maintenance log file before start of maintenance activity.
2. Perform routine maintenance and make entry of each activity in the log file.
3. Keep electronic record of interactions that have occurred during maintenance activities.
4. Check database state and compare it with the database state before the maintenance initiated.
5. Restore the data to its original state for any inconsistent state.

Q. 6(b) Write short note on Reverse engineering. **(10 Marks)**

Ans. : Please refer Q. 6(b) of May 2015.

Q. 6(d) Write short note on Object oriented testing methods. **(10 Marks)**

Ans. : Oriented Testing Methods :

The Object object-oriented testing approach is quite similar to the conventional testing approaches in strategy, but it is different in tactics. Since object oriented analysis and design models are quite similar in structure and their content to the final object oriented program, the testing can be started with review of these models. After the generation of code, the object oriented testing begins. Object-oriented testing mainly focuses on designing and appropriate sequences of operations to cover the states of a class.

The Test Cases and the Class Hierarchy

Through the analysis and design models, if a class is evolved then it becomes a target for test case designs. Since the attributes and operations are encapsulated in a class, the testing operations outside of this class are supposed to be the non-productive. Even though the encapsulation is an important concept for object oriented systems, it can cause a small obstacle while testing. Testing always requires reporting on the concrete and abstract state of the object. But encapsulation may cause some difficulty to obtain this information. If built-in operations provide the values for class attributes to the report, then the state of an object can be acquired. Due to inheritances, the software developer may face challenges in designing the test cases. Any new context o usage requires testing the system again despite reuse has been achieved. As inheritance creates the problem to the developer, the multiple inheritances complicates the testing process more by increasing the number of contexts for which testing is required. The set of test cases can be derived from the super-class, if a sub-class is instantiated from the super-class in the same program. Thus we can use the same set of test cases for the sub-class also that already been used by the super-class. But if the subclass is used in some different context, then the super-class test cases may not be much useful for the sub-classes.

Applicability of Conventional Test Case Design Methods

Usually black-box testing methods those are appropriate for the systems that are developed using conventional development approaches are also appropriate for objected systems also. All the use-cases may provide the useful input in the design of black-box testing and also for the state-based tests.

Fault-Based Testing

The main objective of fault-based testing in object oriented context is to design test cases that have a probability of detecting high number of errors and faults. It is always recommended that the system or the product should be designed in such a way that it satisfies all the customer's requirements. This is the reason why preliminary planning must start with the analysis model. In this technique, it is required to perform fault-based testing. The developer who tests always look for greater number of possible errors and faults. To determine whether the faults exist or not, the test cases are designed to assess the design or code. Of course, the effectiveness of these techniques depends on how testers perceive a plausible fault. If the real faults are detected in an object oriented system, then it should be applauded, since it is perceived that the faults are not much. This particular approach is really better than all the random testing techniques. If the analysis and design models provide some ideas that where it went wrong, then fault-based testing can be proved to be the successful approach. It can provide some good results with less amount of efforts and expenditures.

Scenario-Based Testing

Fault-based testing misses two main types of errors :

1. Incorrect specifications and 2. Interactions among subsystems.

When errors associated with incorrect specifications occur, the product doesn't do what the customer wants. It might do the wrong thing, or it might omit important functionality. But in either circumstance, quality suffers. Errors associated with subsystem interactions occur when the behaviour of one subsystem creates circumstances that cause another subsystem to fail. Scenario-based testing concentrates on what the user does, not what the product does. This means capturing the tasks that the user has to perform, then applying them and their variants as tests. Scenarios uncover interaction errors.

Testing Surface Structure and Deep Structure

The surface structure is the external structure of an object oriented program. This is the structure that is visible to an end-user. In object oriented systems, instead of performing functions, the users are given the task of manipulating the objects in some way. Whichever interface is used, but all the tests depend on user task only. In order to capture these tasks, the developer has to understand, watch, and talk with some selected representative users. The best tests results are obtained when the developer looks at the product or software in some new or unconventional way. The **deep structure** is the internal technical details of an object oriented program. The deep structure is understood by assessing the design and code. Generally the deep structure testing is designed to evaluate all the dependencies, communication mechanisms, and behaviours established for object oriented software. The analysis and design models are usually provide the basis for deep structure testing.

Chapter 8 : Web Engineering [Total Marks - 10]

Q. 6(c) Write short note on Service-oriented software engineering. **(10 Marks)**

Ans. : Service-Oriented Software Engineering

Consider an example service-oriented architecture as depicted in Fig. 1-Q. 6(c).

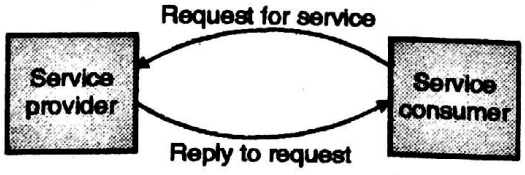


Fig. 1-Q. 6(c) : An example service-oriented architecture

It consists of service provider and service consumer. The consumer requests a service by sending a request message and the service provider come up with a reply or a response message to the service consumer. In some cases, the service provider may also be a service consumer. **Service-Oriented Software Engineering development** is much similar to object-oriented development and component-based development. Following are some points that exhibit this similarity :

Similar to objects and components, a service is also a building block that hides the complexities from the outside word and gives a single interface. The objects use abstract data types, and services have the similar adaptability. Like objects and components, services can be viewed as hierarchy or class. An important term associated to SOA is loose coupling. The loose coupling means the software components have minimum built-in knowledge, rather they discover the information when needed. A client once gets a service, then he discovers all its features like capability, policy etc. Loose coupling has following benefits :

1. **Flexibility** : means services can be available on any server.
2. **Scalability** : means services can be added or removed.
3. **Replaceability** : means updated services may be introduced.
4. **Fault tolerance** : means services are available despite failures from alternate services.

SOA and Web Services

Web services are available to the users through an application-oriented interface. They are the distributed software component that offers the services and functionalities to its users. The information is organized in needed or required manner by using XML (Extensible Markup Language). The XML is widely used web tool for processing the unstructured data and display a formatted data. **For example Online shopping :** A user can add multiple items to its shopping cart. These items may be available on different servers. XML provides a simple interface to access data from different servers located at different places and present a simple or abstract view of overall data. Web services are widely used in a distributed heterogeneous environment. i.e. various platforms they run on and they are written in different languages.

Following are some important specifications used by the web services :

1. XML used for formatting unstructured data.
2. SOAP (Simple Object Access Protocol)
3. WSDL (Web Services Description Language) is an XML based language.
4. The most commonly used communication protocol for web services is SOAP over HTTP or HTTPS.

SOA Design and Development

1. The SOA design uses the general design concepts like :
2. Encapsulation
3. Modularization
4. Separation of the interfaces from their implementation.
5. In addition to this general design concepts, the SOA also introduces the concept of service choreography, service repositories and interaction via messages and events.

Following is the list of steps followed to implement SOA system :

1. Break business logic into different pieces or units.
2. Identify the work to be performed by each unit of business logic.
3. Explain the functionality of each unit in terms of services.

4. Implement the services.
5. Identify the infrastructure needed by services.
6. Identify the link of common functionality between various services.
7. Assign meaningful names to the services to ease in reusability.
8. Define the events for different services.
9. Show the workflows to enable service choreography.
10. Finally publish the services in a registry so that they can be discovered and made available.

6. SOA systems are designed and developed using various SOA standards like WSDL, UDDI etc.

Advantages and the Future of SOA

Based on two major categories i.e. technical and business, the advantages of SOA are illustrated as follows :

1. SOA offers autonomy and location transparency in their services.
2. The architecture is loosely coupled and promotes late binding of services.
3. Modularity of services to decompose complex problems.
4. Reusability of services due to modularity and it is possible to reuse the existing code and minimize the development cost.
5. Because of its agility, SOA enables faster time to market.
6. It is process driven.
7. It is business driven.
8. It is adaptable to changing business needs.

❑❑❑

Dec. 2016

Chapter 1 : Software Engineering Process [Total Marks - 60]

Q. 1(a) Develop the SRS for an Online Shopping Portal.

A customer visits the online shopping Portal. He may buy item or just visit the page and logout. The customer can select segment, then a category and brand to get different products in the desired brand. The customer can select product for purchasing. The process can be repeated for more items. Once the customer finishes the selecting product/s, the cart can be viewed. If the customer wants to edit the final cart it can be done. For completing the process of purchase and payment the customer has to login the portal. If the customer is visiting for first time, he should do the registration process first, else he can directly login to continue. Final is submitted for payment and the delivery address is confirmed by the customer. Confirmation is given to the customer through shipment Id and products list. SRS for this should include:

a. Product perspective
b. Scope and objective
c. Functional Requirements
d. Non-Functional Requirements **(20 Marks)**

Ans. : The SRS for an Online Shopping Portal can be as follows :-

(a) **Product perspective :** The proposed system is a solution carry out buying/ selling products online.

(b) **Scope and objective :** The document is the one that describes the requirements along with interfaces for the system. It is meant for use by the developers and will be the basis for validating the final delivered system. The system helps in buying of goods, products and services online by choosing the listed products from website

(c) **Functional Requirements :**

1. **Product :** Product includes the information about particular product, such as product number, item, name, category, images of products, description, features, brands, constraints of products, which are to be displayed on the website.
2. **Price :** Price deals with the cost of the product, discounts applicable for the particular product of a vendor/seller.
3. **Transactions :** All transactions undergoing in the website will be controlled and managed by this module. Transactions in the sense, Shopping Cart management.
4. **Order Report :** Order Report will have the list of products ordered and the customer details who have bought that product, which are undelivered.
5. **Delivery Report :** Delivery Reports will generate products list, which are delivered to customers. After confirmation of delivery address is confirmed by the customer.

(d) **Non-Functional Requirements :**

(1) **User Interfaces :** Each part of the user interface intends to be as user friendly as possible. The fonts and buttons used will be intended to be very fast and easy to load on web pages. The pages will be kept light in space so that it won't take a long time for the page to load.

(2) **Communication Interface :** The Website Order system shall send an e-mail confirmation to the customer that the items they ordered will be delivered to the shipping address along with user identification.

Q. 4(a) What do you mean by requirements ? Explain Functional and Non- Functional requirements in details. **(10 Marks)**

Ans. : Requirements :

In product development and process optimization, a requirement is a singular documented physical and functional need that a particular design, product or process must be able to perform. It is most commonly used in a formal sense in systems engineering. It is a statement that identifies a necessary attribute, capability, characteristic, or quality of a system for it to have value and utility to a customer, organization, internal user, or other stakeholder. Requirements convey the expectations of users from the software product. The requirements can be obvious or hidden, known or unknown, expected or unexpected from client's point of view.

1. **Functional Requirements :**
   1. **Product :** Product includes the information about particular product, such as product number, item, name, category, images of products, description, features, brands, constraints of products, which are to be displayed on the website.
   2. **Price :** Price deals with the cost of the product, discounts applicable for the particular product of a vendor/seller.
   3. **Transactions :** All transactions undergoing in the website will be controlled and managed by this module. Transactions in the sense, Shopping Cart management.
   4. **Order Report :** Order Report will have the list of products ordered and the customer details who have bought that product, which are undelivered.
   5. **Delivery Report :** Delivery Reports will generate products list, which are delivered to customers. After confirmation of delivery address is confirmed by the customer.
2. **Non-Functional Requirements :**
   1. **User Interfaces :** Each part of the user interface intends to be as user friendly as possible. The fonts and buttons used will be intended to be very fast and easy to load on web pages. The pages will be kept light in space so that it won't take a long time for the page to load.
   2. **Communication Interface :** The Website Order system shall send an e-mail confirmation to the customer that the items they ordered will be delivered to the shipping address along with user identification.

Q. 5(a) Discuss Incremental model and prototype model for software development with merits and demerits. **(10 Marks)**

Ans. : Incremental Model :

The incremental model combines the elements of waterfall model applied in an interactive fashion. The first increment is generally a core product. Each increment produces the product, which is submitted to the customer. The customer suggests some modifications. The next increment implements customer's suggestions and some additional requirements in previous increment. The process is repeated until the product is finished. Consider the development of word processing software (like MS Word or Word Star) using incremental model.

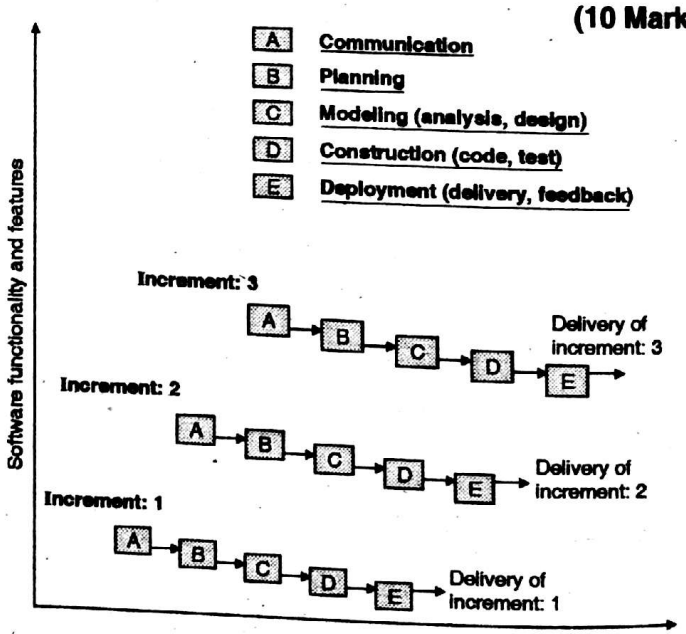


Fig. 1-Q. 5(a) : Project calendar time

Increment 1 : Basic file management functions like : New, Open, Save, Save as etc. can be implemented in first increment.

Increment 2 : More sophisticated editing and document production capabilities can be implemented in second increment. For example rulers, margins and multiple font selection can be implemented.

Increment 3 : Spelling, grammar checking and auto correction of words is implemented in third increment.

Increment 4 : The final advanced page layout capabilities are developed in fourth increment. And so on till the product is finished.

Merits / Advantages :

1. The incremental model is useful when the size of development team is small in the beginning or some team members are not available. Thus early increments can be implemented with small size of team.
2. If the product satisfies the client, then the size of team can be increased.
3. Also increments can be properly planned to handle all types of technical risks. For example some application may require a new hardware that is presently not available. In this situation the increments can be planned to avoid the use of that hardware.
4. The initial product delivery is faster and cost of development is low.
5. The customers can respond to the functionalities after each increment and come up with feedback.

Demerits / Disadvantages :

1. The cost of finished product may be increased in the end beyond the cost estimated.
2. After each increment, the customer can demand for additional functionalities that may cause serious problems to the system architecture.
3. It needs a very clear and complete planning and design before the whole system is broken into small increments.

Prototype model :

The **prototyping paradigm** offers best approach for human machine interaction. Ideally prototype serves as a mechanism for identifying software requirements. The prototyping approach is often called as **throwaway prototyping**. By using this approach, there is a rough demonstration of the customer's requirements i.e. the prototype used in demonstration is just a throwaway prototype. If working prototype is built, developer can use software tools if required (e.g. report generators). This produces working program quickly. Thus prototype can be served as 'first system'.

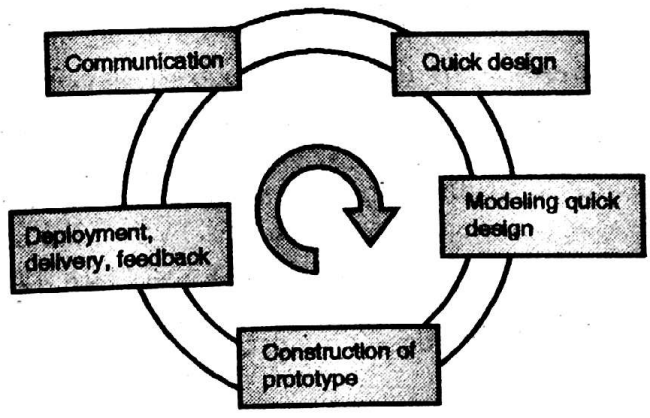


Fig. 2-Q. 5(a) : The prototyping model

Different phases of prototyping model are :

1. **Communication :** The software prototyping paradigm starts with the communication between the developer and the customers. The software engineer and the customer meet regularly to define the overall objectives of the desired software and to identify it's requirements.
2. **Quick design :** Quick design focuses on those aspects of software that are visible to the customer. It includes clear input, output formats and human machine interfaces.
3. **Modelling quick design :** The model of software is now built that gives clear idea of the software to be developed. This enables the developer to better understand the exact requirements.
4. **Construction of prototype :** The concept of modeling the quick design leads to the development of prototype. This construction of prototype is deployed by the customer and it is evaluated by the customer itself.
5. **Deployment, Delivery, Feedback :** As picture of software to be built is very clear, customer can give his suggestions as a feedback. If result is satisfactory, the model is implemented otherwise process is repeated to satisfy customers' all requirements.

Merit / Advantage :

Prototyping makes requirements more clear and system more transparent.

Demerits / Disadvantages :

The software developer generally compromises in the quality to get the working prototype quickly. Due to this reason, sometimes inappropriate operating system or programming language may be selected. He may use and implement inefficient algorithm.

Q. 5(b) Explain size oriented software engineering metrics. **(10 Marks)**

Find function points for an E-commerce application with the following data.

No. of User inputs	50
No. of User outputs	30
No. of User inquiries	35
No. of User files	06
No. of external interfaces	04

Assume suitable complexity adjustment factors and weighting factors.

Ans. : Size oriented software engineering metrics : Please refer Q. 3(a) of May 2015.

Characteristics	Count	X	Average Value	
No. of User inputs	50	X	4	200
No. of User outputs	30	X	5	150
No. of User inquiries	35	X	4	140
No. of User files	06	X	10	60
No. of external interfaces	04	X	7	28

Total Number of Unadjusted Function Point = 578

The value adjustment factor (VAF) is based on 14 general system characteristics (GSC's) that rate the general functionality of the application being counted.

Once all the 14 GSC's have been answered, they should be tabulated using the International Function Point User Group (IFPUG) Value Adjustment Equation (VAF) --

14 where : Ci = degree of influence for each General System Characteristic

VAF = 0.65 + [(Ci) / 100].i = is from 1 to 14 representing each GSC

The final Function Point Count is obtained by multiplying the VAF times the Unadjusted Function Point (UAF).

FP = UAF * VAF

Therefore, the Function Point = 375FP

Q. 6(c) Write short notes on Agile Methodology. **(10 Marks)**

Ans. : Please refer Q. 2(a) of May 2015.

Chapter 2 : Software Project Scheduling, Control and Monitoring [Total Marks - 10]

Q. 6(b) Write short notes on COCOMO Model. **(10 Marks)**

Ans. : Please refer Q. 1(c) of May 2015.

Chapter 3 : Risk Management [Total Marks - 10]

Q. 6(d) Write short notes on Risk Management. **(10 Marks)**

Ans. : Please refer 2(d) of Dec. 2015.

Chapter 4 : Software Configuration Management [Total Marks - 20]

Q. 2(b) Explain Software Configuration item identification. **(10 Marks)**

Ans. : Software Configuration Items

Basically SCI i.e. software configuration item is the integral part of software engineering development process. It is a part of large specification or we can say that one test case among large suite of test cases. In fact the SCI is a document or the program component like C++ functions or a Java applet.

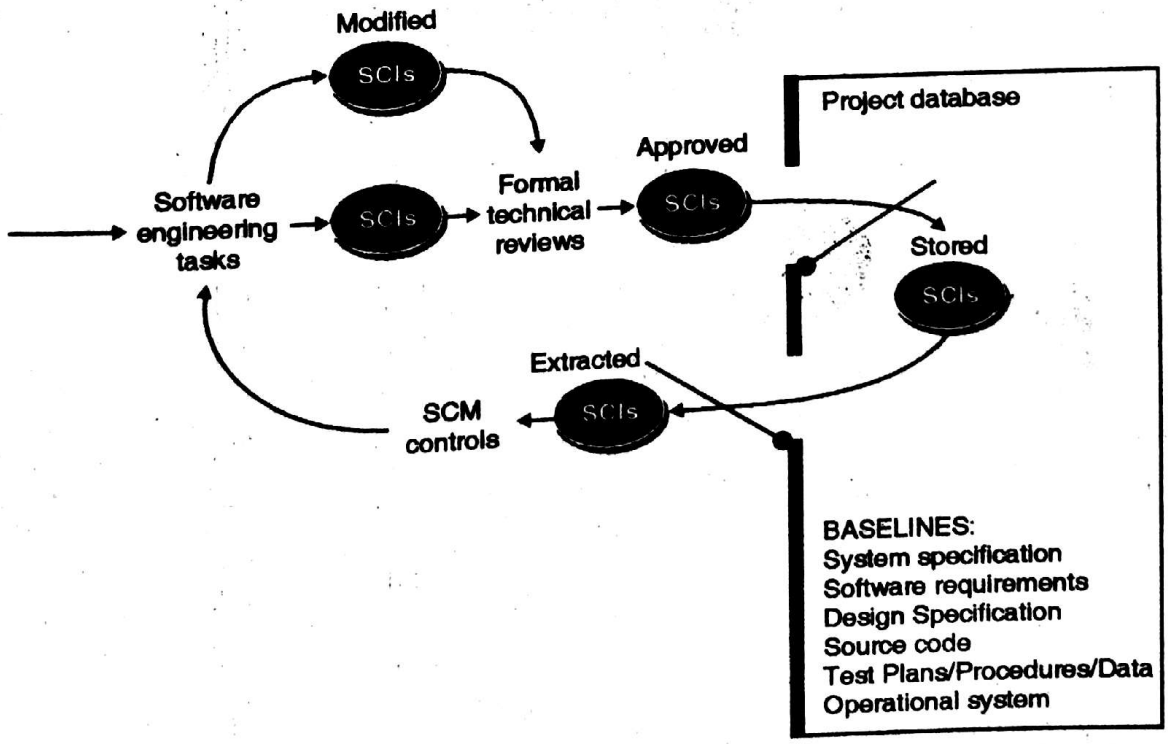


Fig. 1-Q.2 (b) : Base lined SCIs and the project database

Most of the organizations use software tools under configuration control to help development process. In fact the editors, compilers, browsers and various automated tools are the integral part of software configuration. In fact the SCIs are catalogued in the project database with a single name and they form configuration objects. These objects are configuration object and it has a name, attribute and it has certain relationship with other configuration objects. Referring to Fig. 2-Q. 2(b), the configuration objects, Design Specification, Data Model, Component N, Source Code and Test Specification are each defined separately.

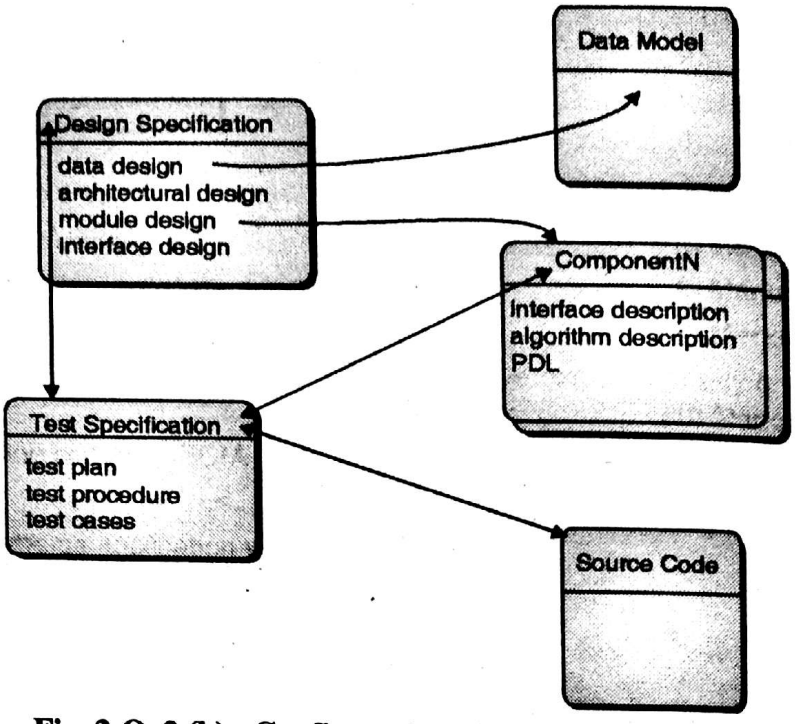


Fig. 2-Q. 2 (b) : Configuration objects

Q. 6(a) Write short notes on Version control and change control. **(10 Marks)**

Ans. : **Please refer Q. 3(b) of May 2016.**

Chapter 5 : Software Design Specification [Total Marks - 20]

Q. 2(a) What is user interface design process? Explain with one example. **(10 Marks)**

Ans. : **Please refer Q. 5(b) of Dec. 2016.**

Q. 3 (b) What is Coupling and Cohesion? Explain different forms of it. **(10 Marks)**

Ans. : **Please refer Q. 3(b) of May 2015.**

Chapter 7 : Software Testing [Total Marks - 20]

Q. 3 (a) What are the objectives of testing? Explain black box testing and integration testing. **(10 Marks)**

Ans. : Objectives of testing :

Software Testing has different goals and objectives. The major objectives of Software testing are as follows :

1. Finding defects which may get created by the programmer while developing the software.
2. Gaining confidence and providing information about the level of quality.
3. To prevent defects.
4. To make sure that the end result meets the business and user requirements.
5. To ensure that it satisfies the BRS that is Business Requirement Specification and SRS that is System Requirement Specifications.
6. Gain the confidence of the customers by providing them a quality product.
7. Software testing helps in finalizing the software application or product against business and user requirements. It is very important to have good test coverage in order to test the software application completely and make it sure that it's performing well and as per the specifications.

8. While determining the coverage the test cases should be designed well with maximum possibilities of finding the errors or bugs. The test cases should be very effective. This objective can be measured by the number of defects reported per test cases. Higher the number of the defects reported the more effective are the test cases.
9. Once the delivery is made to the end users or the customers they should be able to operate it without any complaints. In order to make this happen the tester should know as how the customers are going to use this product and accordingly they should write down the test scenarios and design the test cases.
10. This will help a lot in fulfilling all the customer's requirements. Software testing makes sure that the testing is being done properly and hence the system is ready for use. Good coverage means that the testing has been done to cover the various areas like functionality of the application, compatibility of the application with the OS, hardware and different types of browsers, performance testing to test the performance of the application and load testing to make sure that the system is reliable and should not crash or there should not be any blocking issues.
11. It also determines that the application can be deployed easily to the machine and without any resistance. Hence the application is easy to install, learn and use.

Black-Box Testing (Also called Behavioural Testing) :

Black-box testing is also known as behavioural testing. It focuses only on the functional requirements of the software. The black-box testing helps the software developer to derive the sets of input conditions. These input conditions exercise all the functional requirements for a program.

1. Sample code 2. int a, b 3. input (a, b) 4. if (a > b)
5. display ("a is bigger than b") 6. else display ("b is equal to or greater than a")

Black-box testing is a different approach and it can not be the alternative or the replacement of white-box testing approach. It is a complementary testing method that detects a different class of errors compared to white-box testing methods.

Following are categories of errors that can be detected by using black-box testing approach :

1. The performance errors or the behavioural errors
2. Missing or incorrectly defined functions
3. Data structures incorrectly defined and external data base access errors,
4. Errors occurred during initialization and termination.
5. Errors occurred during interface.

In contrast to white-box testing, where testing begins early in the testing process, the black box testing is applied in the final stages of testing.

Integration testing :

Since the object-oriented software does not have any fixed hierarchical control structure, the top-down and bottom-up integration strategies becomes meaningless. Also, the integrating operations one at a time into a class is impossible due to the direct and indirect interactions of the components of that class. Following are two different strategies for integration testing in OO context :

1. Thread-based testing 2. Use-based testing

The **thread-based testing** combines the set of classes that are required to respond to the input for the system. The **use-based testing** start with the construction of the system and it tests those classes only that uses very few server classes.

Q. 4(b) What are the different types of maintenance and also explain steps for creating maintenance log ? **(10 Marks)**

Ans. : **Please refer Q. 5(a) of May 2016.**

❑❑❑

May 2015

Q. 1 (a) Write suitable applications of different software models. C1 **(5 Marks)**

(b) Compare verification and validation testing. C7 **(5 Marks)**

(c) Explain COCOMO Model. C2 **(5 Marks)**

(d) Explain the different types of software Maintenance. C7 **(5 Marks)**

Q. 2 (a) What is agile methodology? Explain it with the principles used and give example of any one such software model. C1 **(10 Marks)**

(b) Explain change control and version control in SCM. C4 **(10 Marks)**

Q. 3 (a) Explain size oriented software engineering metrics. C1 **(10 Marks)**

Find function points for an e-commerce application with following data,

Number of user inputs	50
Number of user outputs	40
Number of user inquiries	35
Number of user files	06
Number of external interfaces	04

C2

Assume suitable complexity adjustment factors and weighting factors.

(b) What is coupling and cohesion? Explain different forms of it. C5 **(10 Marks)**

Q. 4 (a) What are the features of a good user interface? Design and interface for online air ticket reservation system. C5 **(10 Marks)**

(b) Explain different metrics used for maintaining software quality. C6 **(10 Marks)**

Q. 5 (a) What is SRS document? Build an SRS document for online student feedback system. C1 **(10 Marks)**

(b) What are software risks? Write a note on RMMM for delayed projects. C3 **(10 Marks)**

Q. 6 (a) Compare black box and white box testing. Find cyclomatic complexity of following code

```
IF A = 10 THEN
IF B > C THEN
        A=B
ELSE A= C
END IF
END IF
        PRINT A
        PRINT B
        PRINT C
```

C7 **(10 Marks)**

(b) Explain software reverse engineering in detail. C7 **(10 Marks)**

□□□

Dec. 2015

Q. 1 A distance learning institute decides to use e-learning software to ease its regular functioning of the program. Through this e-learning tool students can register to various courses, appear for online exams, download study material, upload assignments online, view lecture videos etc. The faculty can upload study materials, conduct exams, teach one or many courses. The institute can check student and faculty information, collect fees, pay salary, display results and so on. Create an SRS for the institute that includes the following **(20 Marks)**

1. Product perspective
2. Scope and objective
3. Functional requirements (at least 3)
4. Non-functional requirements

Q. 2 Attempt any four **(4 Marks)**

(a) Define Software Engineering. Explain in brief the software process framework. **(5 Marks)**

(b) Discuss on Modularity and Functional Independence fundamentals of design concepts. **(5 Marks)**

(c) Explain cyclomatic complexity. How is it computed? **(5 Marks)**

(d) Discuss the different categories of risk that help to define impact values in a risk table. **(5 Marks)**

(e) Briefly explain Unit and Integration Testing in the OO Context. **(5 Marks)**

Q. 3 (a) Explain in brief the different types of coupling and cohesion. Give one practical example of high cohesion and low coupling. **(10 Marks)**

(b) What is FTR in SQA? What are its objectives? Explain the steps in FTR. **(10 Marks)**

Q. 4 (a) What is Agility in context of software engineering? Explain Extreme Programming (XP) with suitable diagram. **(10 Marks)**

(b) Explain different techniques in White Box Testing. **(10 Marks)**

Q. 5 (a) Explain the various steps in Risk Management with suitable diagram. Identify the risks associated with delayed projects. **(10 Marks)**

(b) Explain different architectural styles with suitable brief example for each. **(10 Marks)**

Q. 6 (a) Explain the change control and version control activities in SCM. **(10 Marks)**

(b) Explain TDD with its advantages. **(10 Marks)**

❑❑❑

May 2016

Q. 1 Develop the SRS for the following scenario : **(20 Marks)**

A school has one or more departments. Department offers one or more subjects. A particular subject will be offered by only one department. Department has instructors and instructors can work for one or more departments. Students canenrol in up to 5 subjects in a school. Instructor can teach up to 3 subjects. The same subject can be taught by the different instructors. Students can be enrolled in more than one school.

SRS for the school should include the following :

(a) Product perspective (b) Scope and objective ;

(c) Functional requirements (d) Non-Functional requirements

Q. 2 (a) Explain and compare FTR and walkthrough. C1 **(10 Marks)**

(b) Explain the process of CMM. C2 **(10 Marks)**

Q. 3 (a) Explain coupling and cohesion. Explain different types of coupling and cohesion. → C5 **(10 Marks)**

(b) What are Agile process and its advantages ? Explain any one Agile process. C1 **(10 Marks)**

Q. 4 (a) Explain the change control and version control activities in SCM. C4 **(10 Marks)**

(b) Differentiate between black box testing and white box testing. Explain in detail about any one testing tool. C7 **(10 Marks)**

Q. 5 (a) What are the different types of maintenance and also explain steps for creating a maintenance log ? C7 **(10 Marks)**

(b) What is user interface design process ? Explain with one example. C5 **(10 Marks)**

Q. 6 Write short notes on (any two) : **(20 Marks)**

(a) Risk management C3

(b) Reverse Engineering & C7

(c) Service-oriented Software Engineering C8

(d) Object oriented testing methods. C7

Dec. 2016

Q. 1 (a) Develop the SRS for an Online Shopping Portal.

C1 A customer visits the online shopping Portal. He may buy item or just visit the page and logout. The customer can select segment, then a category and brand to get different products in the desired brand. The customer can select product for purchasing. The process can be repeated for more items. Once the customer finishes the selecting product/s, the cart can be viewed. If the customer wants to edit the final cart it can be done. For completing the process of purchase and payment the customer has to login the portal. If the customer is visiting for first time, he should do the registration process first, else he can directly login to continue. Final is submitted for payment and the delivery address is confirmed by the customer. Confirmation is given to the customer through shipment Id and products list. SRS for this should include:

a. Product perspective b. Scope and objective

c. Functional Requirements d. Non-Functional Requirements **(20 Marks)**

Q. 2 (a) What is user interface design process? Explain with one example. C5 **(10 Marks)**

(b) Explain Software Configuration item identification. C4 **(10 Marks)**

Q. 3 (a) What are the objectives of testing? Explain black box testing and integration testing. C7 **(10 Marks)**

(b) What is Coupling and Cohesion? Explain different forms of it. C5 **(10 Marks)**

Q.4 (a) What do you mean by requirements ? Explain Functional and Non- Functional requirements in details. **(10 Marks)**

(b) What are the different types of maintenance and also explain steps for creating maintenance log ? **(10 Marks)**

Q.5 (a) Discuss Incremental model and prototype model for software development with merits and demerits. **(10 Marks)**

(b) Explain size oriented software engineering metrics. **(10 Marks)**

Find function points for an E-commerce application with the following data.

No. of User inputs	50
No. of User outputs	30
No. of User inquiries	35
No. of User files	06
No. of external interfaces	04

Assume suitable complexity adjustment factors and weighting factors.

Q.6 Write short notes on (any two) :

(a) Version control and change control. **(10 Marks)**

(b) COCOMO Model. **(10 Marks)**

(c) Agile Methodology. **(10 Marks)**

(d) Risk Management. **(10 Marks)**

❑❑❑

Note