S.Y. B.Sc. (IT) : Sem. IV

Software Engineering

Time : 2½ Hrs.] Prelim Question Paper Solution [Marks : 75

Q.1 Attempt any THREE of the following : [15]

Q.1(a) Explain V-model along with waterfall. [5]

Ans.: The waterfall model, sometimes called the classic life cycle, suggests a systematic, sequential approach to software development that begins with customer specification of requirements and progresses through planning, modeling, construction, and deployment, culminating in ongoing support of the completed software. (Refer figure 1)

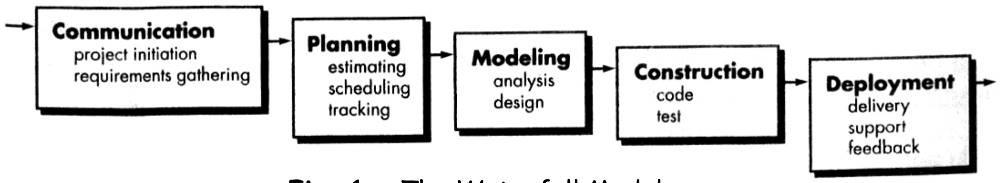


Fig. 1 : The Waterfall Model

A variation in the representation of the waterfall model is called the V-model. Represented in figure 2, the V-model [Buc99] depicts the relationship of quality assurance actions to the actions associated with communication, modelling, and early construction activities. As a software team moves down the left side of the V, basic problem requirements are refined into progressively more detailed and technical representations of the problem and its solution. Once code has been generated, the team moves up the right side of the V, essentially performing a series of tests (quality assurance actions) that validate each of the models created as the team moved down the left side. In reality, there is no fundamental difference between the classic life cycle and the V-model. The V-model provides a way of visualizing how verification and validation actions are applied to earlier engineering work.

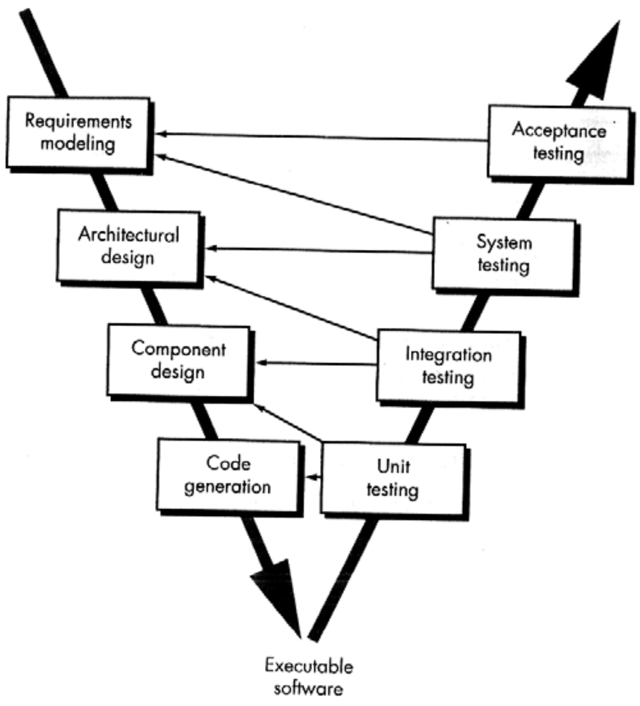


Fig. 2 : The V-model

The waterfall model is the oldest paradigm for software engineering. However, over the past three decades, criticism of this process model has caused even ardent supporters to question its efficacy.

Q.1(b) Explain extreme programming (XP)? [5]

Ans.: Extreme Programming (XP), the most widely used approach to agile software development. Although early work on the ideas and methods associated with XP occurred during the late 1980s, the seminal work on the subject has been written by Kent Beck [Bec04a]. More recently, a variant of XP, called industrial XP (IXP) has been proposed.

Feedback is derived from three sources : the implemented software itself, the customer, and other software team members. By designing and implementing an effective testing strategy, the software (via test results) provides the agile team with feedback. XP makes use of the unit test as its primary testing tactic. As each class is developed, the team develops a unit test to exercise each operation according to its specified functionality. As an increment is delivered to a customer, the user stories or use cases that are implemented by the increment are used as a basis for acceptance tests.

Extreme Programming uses an object-oriented approach as its preferred development paradigm and encompasses a set of rules and practices that occur within the context of four framework activities : planning, design, coding and testing. Following figure illustrates the XP process and notes some of the key ideas and takes that are associated with each framework activity. Key XP activities are summarized in the paragraphs that follow :

1. Planning
2. Design
3. Coding
4. Testing

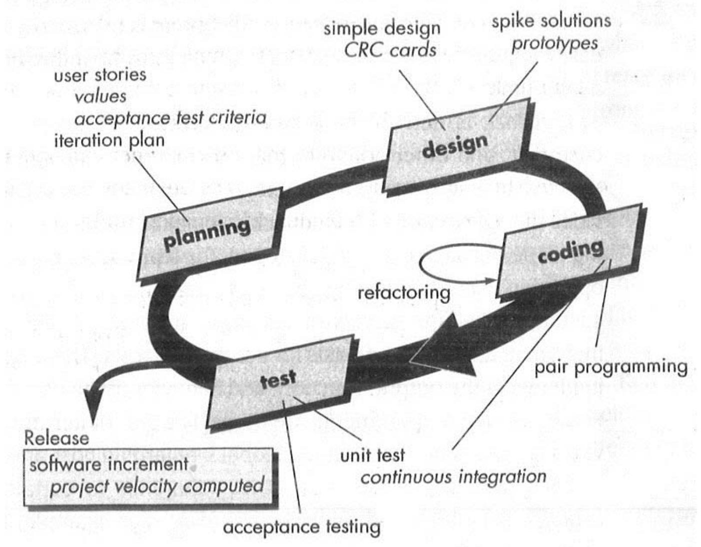


Fig.: The Extreme Programming Process

Q.1(c) Explain functional and non-functional requirements. [5]

Ans.: **Functional requirements :** The functional requirements part discusses the functionalities required from the system. The system is considered to perform a set of high-level functions $\{f_i\}$. The functional view of the system is shown in given figure. Each function f_i of the system can be considered as a transformation of a set of input data (i_i) to the corresponding set of output data (o_i). The user can get some meaningful piece of work done using a high-level function.

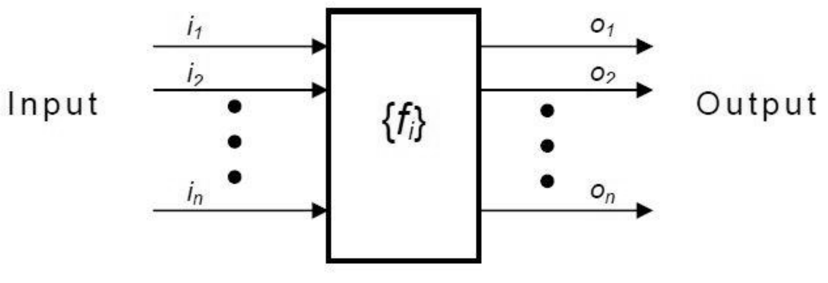


Fig.: View of a system performing a set of functions

Non-functional requirements : Nonfunctional requirements deal with the characteristics of the system which cannot be expressed as functions - such as the maintainability of the system, portability of the system, usability of the system, etc.

Q.1(d) Explain SDLC? [5]

Ans.: A software life cycle model (also called process model) is a descriptive and diagrammatic representation of the software life cycle. A life cycle model represents all the activities required to make a software product transit through its life cycle phases. It also captures the order in which these activities are to be undertaken. In other words, a life cycle model maps the different activities performed on a software product from its inception to retirement. Different life cycle models may map the basic development activities to phases in different ways.

Thus, no matter which life cycle model is followed, the basic activities are included in all life cycle models though the activities may be carried out in different orders in different life cycle models. During any life cycle phase, more than one activity may also be carried out.

Q.1(e) Explain incremental model? [5]

Ans.: The incremental model combines elements of linear and parallel process flows. Referring to following figure, the incremental model applies linear sequences in a staggered fashion as calendar time progresses. Each linear sequence produces deliverable "increments" of the software [McD93] in a manner that is similar to the increments produced by an evolutionary process flow.

For example, word-processing software developed using the incremental paradigm might deliver basic file management, editing, and document production functions in the first increment; more sophisticated editing and document production functions in the first increment; more sophisticated editing and document production capabilities in the second increment; spelling and grammar checking in the third increment; and advanced page layout capability in the fourth increment. It should be noted that the process flow for any increment can incorporate the prototyping paradigm.

When an incremental model is used, the first increment is often a core product. That is, basic requirements are addressed but many supplementary features (some known, others unknown) remain undelivered. The core product is used by the customer (or undergoes detailed evaluation). As a result of use and / or evaluation, a plan is developed for the next increment. The plan addresses the modification of the core product to better meet the needs of the customer and the delivery of additional features and functionality. This process is repeated following the delivery of each increment, until the complete product is produced.

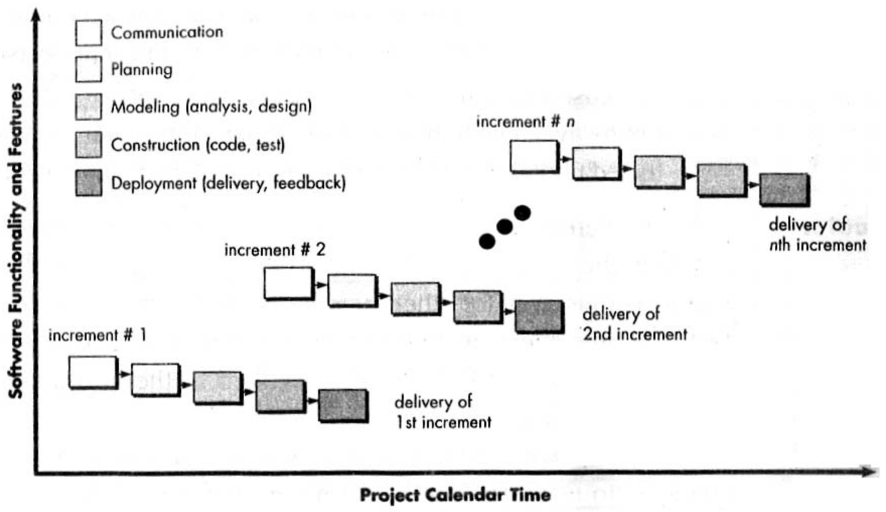

Communication
Planning
Modeling (analysis, design)
Construction (code, test)
Deployment (delivery, feedback)
Software Functionality and Features
increment # n
delivery of nth increment
increment # 2
delivery of 2nd increment
increment # 1
delivery of 1st increment
Project Calendar Time

Fig.: The Incremental Model

Q.1(f) Explain Rational unified process model? [5]

Ans.: The Unified Process is no exception. Figure below depicts the "phases" of the UP and relates them to the generic activities that have been discussed.

The inception phase of the UP encompasses both customer communication and planning activities. By collaborating with stakeholders, business requirements for the software are identified; a rough architecture for the system is proposed; and a plan for the iterative, incremental nature of the ensuing project is developed. Fundamental business requirements are described through a set of preliminary use cases that describe which features and functions each major class of users desires. Architecture at this point is nothing more than a tentative outline of major subsystems and the function and features that populate them. Later, the architecture will refined and expanded into set of models that will represent different views of the system. Planning identifies resources, assesses major risks, defines a schedule, and establishes a basis for the phases that are to be applied as the software increment is developed.

The elaboration phase encompasses the communication and modeling activities of the generic process model (Figure below). Elaboration refines and expands the preliminary use cases that were developed as part of the inception phase and expands the architectural representation to include five different views of the software – the use case model, the requirements model, the design model, the implementation model, and the deployment model. In some cases, elaboration creates an "executable architectural baseline".

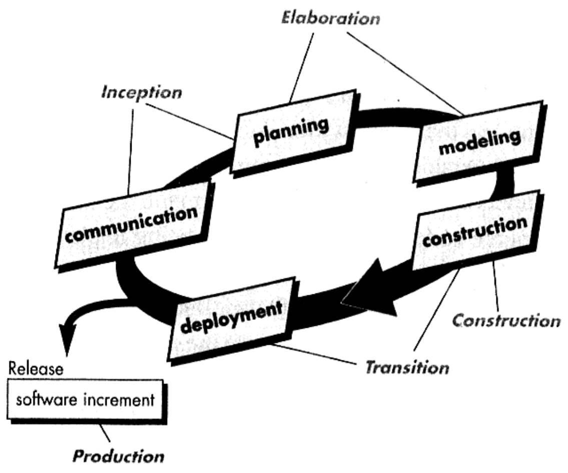


Fig.: Unified Process

Q.2 Attempt any THREE of the following: [15]

Q.2(a) Explain requirement Elicitation? [5]

Ans.: It certainly seems simple enough – ask the customer, the users, and others what the objectives for the system or products are, what is to be accomplished, how the system or product fits into the needs of the business, and finally, how the system or product is to be used on a day-to-day basis. But it isn't simple – it's very hard.

Christel and Kang [Cri92] identify a number of problems that are encountered as elicitation occurs.

- **Problems of scope :** The boundary of the system is ill-defined or the customers / users specify unnecessary technical detail that may confuse, rather than clarify, overall system objectives.
- **Problems of understanding :** The customers / users are not completely sure of what is needed, have a poor understanding of the capabilities and limitations of their computing environment, don't have a full understanding of the problem domain, have trouble communicating needs to the system engineer, omit information that is believed to be "obvious", specify requirements that conflict with the needs of other customers/users, or specify requirements that are ambiguous or untestable.

  Problems of volatility : The requirements change over time.

Q.2(b) Explain legacy systems? [5]

Ans.:
1. Legacy systems are software systems that are developed specially for an organization have a long lifetime.
2. Many software systems that are still in use were developed many years ago using technologies that are now obsolete.
3. These systems are still business critical that is, they are essential for normal functioning of the business.
4. They have been given the name legacy systems
   - Legacy systems have a complete specification.
   - Business processes are reliant on the legacy systems.
   - These systems may embed business rules that are not formally documented elsewhere.
   - New software development is risky and may not be successful.

Q.2(c) Explain emergent properties of a system? [5]

Ans.: Emergent properties are properties of the system as a whole rather than properties that can be derived from the properties of components of a system. Emergent properties are a consequence of the relationships between system components. They can therefore only be assessed and measured once the components have been integrated into a system.

Some examples of emergent properties :

Property	Description
Volume	The volume of a system (the total space occupied) varies depending on how the component assemblies are arranged and connected.
Reliability	System reliability depends on component reliability but unexpected interactions can cause new types of failures and therefore affect the reliability of the system.
Security	The security of the system (its ability to resist attack) is a complex property that cannot be easily measured. Attacks may be devised that were not anticipated by the system designers and so may defeat built-in safeguards.
Repairability	This property reflects how easy it is to fix a problem with the system once it has been discovered. It depends on being able to diagnose the problem, access the components that are faulty, and modify or replace these components.
Usability	This property reflects how easy it is to use the system. It depends on the technical system components, its operators, and its operating environment.

Q.2(d) Explain system dependabilities. [5]

Ans.: The most important emergent property of a critical system is its dependability. It covers the related system attributes of availability, reliability, safety & security.

Importance of dependability :
- System that are unreliable, unsafe or insecure are often rejected by their users (refuse to the product from the same company).
- System failure costs may be very high. (reactor / aircraft navigation)
- Untrustworthy systems may cause information loss with a high consequent recovery cost.

Dependability Requirements
1. The system shall be available to deliver insulin when required to do so.
2. The system shall perform reliability and deliver the correct amount of insulin to counteract the current level of blood sugar.

3. The essential safety requirement is that excessive does of insulin should never be delivered as this is potentially life threatening.

The dependability of a system equates to its trustworthiness.
A dependable system is a system that is trusted by its users.

Principle dimensions of dependability :

- **Availability :** Probability that it will be up & running & able to deliver at any given time;
- **Reliability :** Correct delivery of services as expected by user over a given period of time;
- **Safety :** A judgement of how likely the system will cause damage to people or its environment;
- **Security :** A judgement of how likely the system can resist accidental or deliberate instructions;

Q.2(e) Explain oritical systems and its types. [5]

Ans.: If the system failure results in significant economic losses, physical damages or threats to human life than the system is called critical systems.

Objectives :

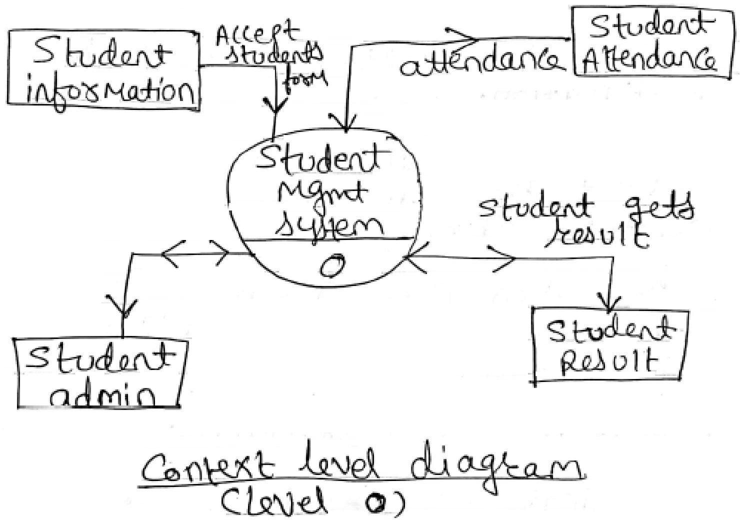
1. To explain what is meant by a critical system where system failure can have severe human or economic consequence.
2. To explain four dimensions of dependability : availability, reliability, safety and security.
3. To explain that, to achieve dependability, you need to avoid mistakes, detect and remove errors and limit damage cause by failure.

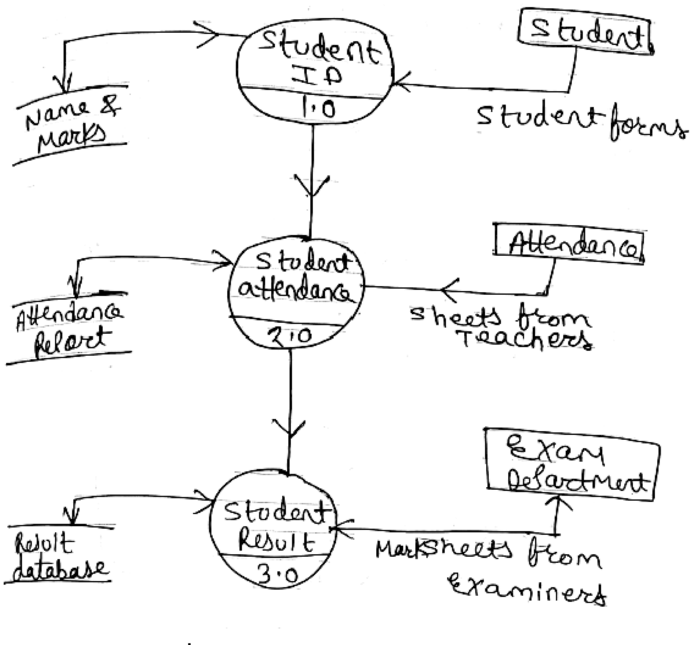
There are 3 types of critical systems in software engineering.

1. **Safety Critical Systems :**
   - Failure results in loss of life, injury or damage to the environment;
   - Chemical plant protection system.
2. **Mission Critical Systems :**
   - Failure results in failure of some goal-directed activity.
   - Spacecraft navigation system.
3. **Business Critical Systems :**
   - Failure results in high economic losses;
   - Customer accounting system in a bank;

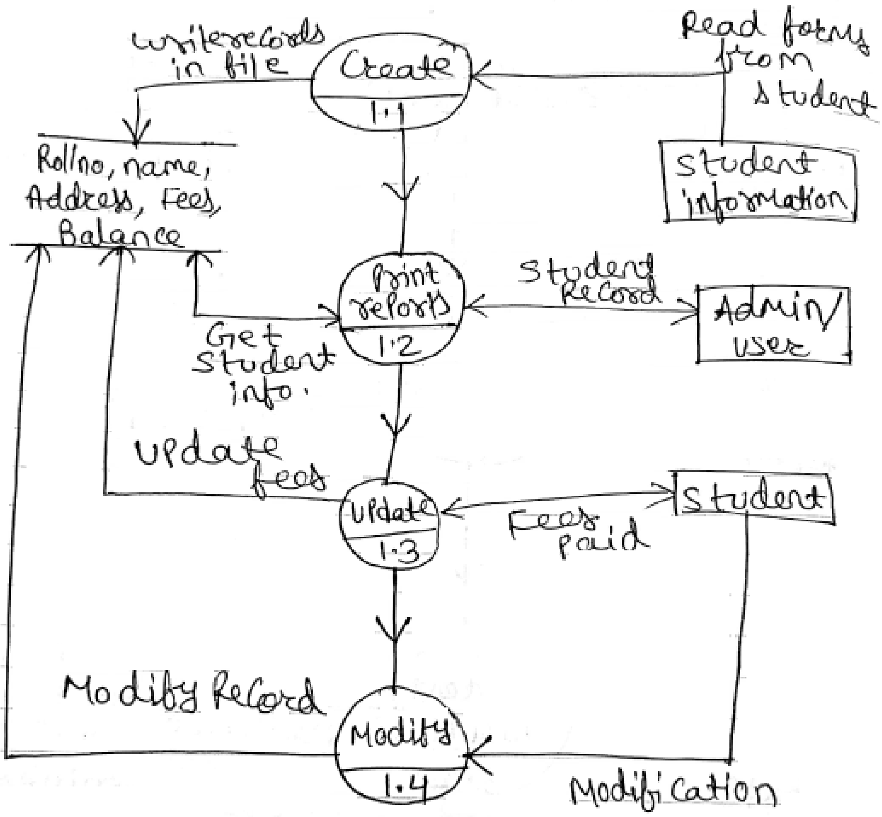
Q.2(f) Draw a data flow diagram (DFD) for collage management system by assuming your suitable data [5]

Ans.:





Level 1 DFD



Level 2 DFD Part I

Q.3 Attempt any THREE of the following: [15]

Q.3(a) Explain client server style? [5]

Ans.: Another very common style used to build systems today is the client-server style. Client-server computing is one of the basic paradigms of distributed computing and this architecture style is built upon this paradigm.

In this style, there are two component types – clients and servers. A constraint of this style is that a client can only communicate with the server and cannot communicate with other clients. The communication between a client component and a server component is initiated by the client when the client sends a request for some service that the server supports. The server receives the request at its defined port, performs the service, and then returns the results of the computation to the client who requested the service.

There is one connector type in this style – the request/reply type. A connector connects a client to a server. This type of connector is asymmetric – the client end of the connector can only make requests (and receive the reply), while the server end can only send replies in response to the requests it gets through this connector. The communication is frequently synchronous – the client waits for the server to return the results before proceeding. That is, the client is blocked at the request until it gets the reply.

A general form of this style is an n-tier structure. In this style, a client sends a request to a server, but the server, in order to service the request, sends some request to another server. That is, the server also acts as a client for the next tier. This hierarchy can continue for some levels, providing an n-tier system. A common example of this is the 3-tier architecture. In this style, the clients that make requests and receive the final results reside in the client tier. The middle tier, called the business tier, contains the component that processes the data submitted by the clients and applies the necessary business rules. The third tier is the database tier in which the data resides. The business tier interacts with the database tier for all its data needs.

Most often, in a client-server architecture, the client and the server component reside on different machines. Even if they reside on the same machine, they are designed in a manner such that they can exist on different machines. Hence the connector between the client and the server is expected to support the request/result type of connection across different machines. Consequently, these connectors are internally quite complex and involve a fair amount of networking to support. Many of the client-server systems today use TCP ports for their connectors. The web uses the HTTP for supporting this connector.

Note that there is a distinction between a layered architecture and a tiered architecture. The tiered style is a component and connector architecture view in which each tier is component, and these components communicate with the adjacent ones through a defined protocol. A layered architecture is a module view providing how modules are organized and used. In the layered organization, modules are organized in layers with modules in a layer allowed to invoke services only of the modules in the layer below. Hence, layered and tiered represent two different views. We can have an n-tiered architecture in which some tier(s) has a layered architecture. For example, in a client-server architecture, the server might have a layered architecture, that is, modules that compose the server are organized.

Q.3(b) Explain UI design issues? [5]

Ans.: UI design issues

There are few issues that are to be considered while designing a good UI.
These issues should be taken care of so that they do not cause any technical problems.
Following are the design issues :

(i) Response time

This is the mean time between request and response of the software with desired output. It can be measured in length and variability.
The response time should not be too long and variability means deviation from the average response time.

(ii) Help facilities

The end user requires help in 2 cases :
(1) When he needs some information and he is not able to find it.
(2) When he is in a trouble.
Bothe of these requirements must be available all the time to the user.

(iii) Error handling
Poor error messages may result in rejecting the product rather than accepting it. The error message should describe the problem in words that are well understood by users and it should provide an solution to it.

(iv) Application accessibility
It states whether the application is simple to interact or not.
Special guidelines are given to the users such as visual, hearing etc.

(v) Internationalization
The application should be usable by any type of end users, located in different locations and speaking different languages. The interface design should be globalized rather than localized.

Q.3(c) Explain management activities? [5]

Ans.: Management activities
Many managers are responsible to execute some or all of the following management activities involved in project management.
(i) Project Proposal Writing
(ii) Project Planning and Scheduling
(iii) Project Costing
(iv) Project Monitoring and Reviews
(v) Personnel Selection and Evaluation
(vi) Report writing and Presentations

The first four activities are grouped under the project planning, whereas the last 2 are performed when running the project. All this activities are performed iteratively.

(i) Project Proposal Writing
It is the initial step and it contains the project objectives, plan to carry out the project, cost and schedule estimation.

(ii) Project Planning and Scheduling
It involves first identification of the activities, milestones, deliverables and then a plan is formed to develop the estimated project in time and with good quality.

(iii) Project Costing
It involves the estimation of resources, which are needed to successfully carry out the project plan.

(iv) Project Monitoring and Reviews
It is an ongoing activity and it is done by project manager, where he keeps examining the project progress and compare it with planned progress and costs.

(v) Personnel Selection and Evaluation
The project budget is low, so instead of highly paid staff, less experienced and less paid staff should be assigned to this project.
The appropriate experienced staff is assigned to other important project, so there is no need to hire new staff.

(vi) Report writing and Presentations
The project manager is responsible to write short and abstract information from detailed project review and use this information during the progress reviews.

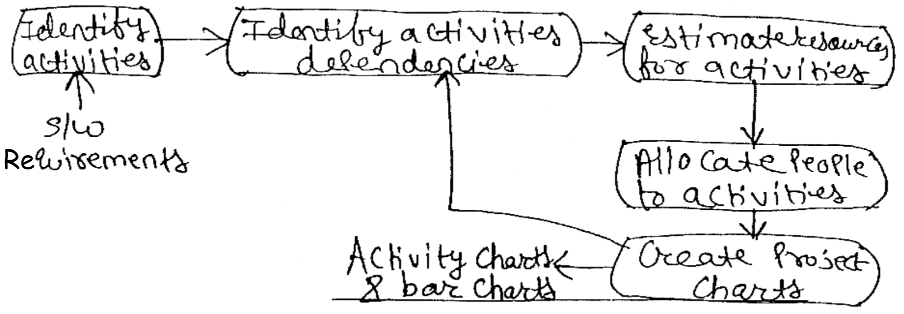
Q.3(d) Explain project scheduling? [5]

Ans.: Project Scheduling:

(i) Description :
1) The Project schedule is a Calendar that is used to allocate the tasks to be performed with the resources available to perform the tasks.
2) Before a Project schedule is estimated, the Manager designs a work breakdown structure (WBS), which is used to estimate the time to implement the task.

(ii) Project Scheduling Process:

1) Divide the Project into various tasks and estimate the duration and resources required to complete each task.
2) Arrange the tasks so as to make optimal use of workforce.
3) Reduce the task dependencies, so as to avoid delays that might be caused by some other task i.e. waiting for another task to complete.


Identify activities
Identify activities dependencies
Estimate resources for activities
S/W Requirements
Allocate People to activities
Create Project Charts
Activity Charts & bar Charts

(iii) Scheduling Problems:

1) Detecting the Complexity of the problems and accordingly estimating the cost of developing a solution is difficult.
2) Achieved Productivity is not Proportional to the number of people working on it.
3) Adding new staff in the late project development moves the delivery date further, because of the communication gap between old staff & new staff.
4) Some accidental events happen in the project development and these needs to be considered in the planning.

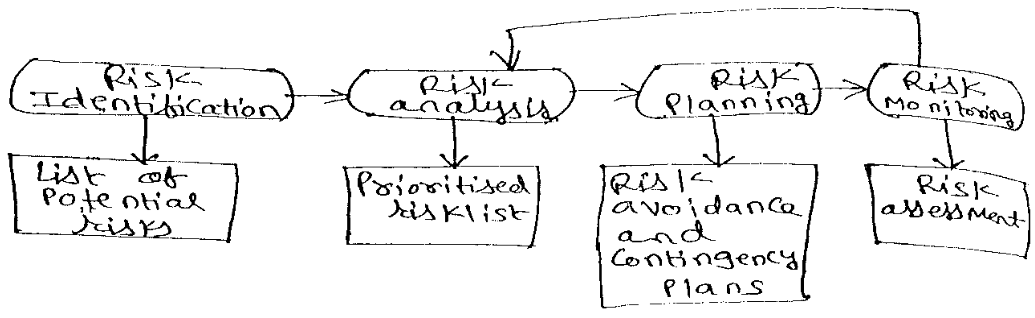
(iv) Basic Principles :

1) **Interdependency:** Few activities, actions or tasks occur in sequence where they are interdependent upon each other and while others and while others occur in parallel.
2) **Effort allocation:** The Project Manager must allocate number of people that must be present at any given point of time.
3) **Effort Validation:** The Project manager must monitor that, not more than the allocated number of people are present at any given point of time.
4) **Defined Responsibilities:** Every task that is scheduled must be assigned to a specific team member.
5) **Defined outcomes:** Every task that is scheduled must have a defined outcome.
6) **Defined Milestones:** Each set of task must be associated with a project milestone.
7) **Time allocation:** Each tack must be allocated some number of work units, defining their start date and completion date.

Q.3(e) Explain risk management? **[5]**

Ans.: Risk Management:

1) It is about identifying risks to ensure that these risks do not turn into major threats.
2) It is about identifying risks and deriving plans to reduce their effect on the project.
3) The process is shown as follows :



4) **There are 7 Principles of it :**
   (i) **Maintain a global Perspective:** View the s/w risks within the systems context and business problem.
   (ii) **Take a forward – looking view:** Think about the risks that may arise in the future and establish plans so that those risks can be managed.
   (iii) **Encourage Open–Communication:** Encourage all stakeholders and users to suggest risks at any time.
   (iv) **Integrate:** Identify and consider those risks, which have to be integrated into the s/w.
   (v) **Emphasize a Continuous Process:** Throughout the s/w process, the risk mgmt team should be careful enough to modify identified risks.
   (vi) **Develop a shared product vision:** Sharing the same vision of the software by all stakeholders can do better risk identification and assessment.
   (viii) **Encourage teamwork:** The talents, skills and knowledge of all stakeholders should be grouped together to get better idea about the risks involved.

Q.3(f) Explain six sigma strategy? [5]

Ans.: Six Sigma is the most widely used strategy for statistical quality assurance in industry today. Six Sigma strategy "is a rigorous and disciplined methodology that uses data and statistical analysis to measure and improve a company's operational performance by identifying and eliminating defects in manufacturing".

The Six Sigma strategy defines 3 core steps :

- Define customer requirements and project goals via well defined methods of customer communication.
- Measure the existing process and its output to determine current quality performance.
- Analyze defect metrics and determine the vital few causes.

If an existing software process is in place, but improvement is required, Six Sigma suggests 2 additional steps :

- Improve the process by eliminating the root causes of defects.
- Control the process to ensure that future work does not reintroduce the causes of defects.

Q.4 Attempt any THREE of the following: [15]

Q.4(a) Differentiate between software verification and validation. [5]

Ans.: **Verification:** The output of the system design phase, like the output of other phases in the development process, should be verified before proceeding with the activities of the next phase. Unless the design is specified in a formal, executable language, the design cannot be executed for verification. Other means for verification have to be used. The most common approach for verification is design reviews or inspections.

Validation: Due to the nature of the requirement specification phase, there is a lot of room for misunderstanding and committing errors, and it is quite possible that the requirements specification does not accurately represent the client's needs. The basic objective of the requirements validation activity is to ensure that the SRS reflects the actual requirements accurately and clearly. A related objective is to check that the SRS document is itself of "good quality".

Verification refers to the set of activities that ensure that software correctly implements a specific function.

Validation refers to a different set of activities that ensure that the software that has been built is traceable to customer requirements.

Boehm states this another way :
Verification : "Are we building the product right?"
Validation : "Are we building the right product?"

Verification and validation encompasses a wide array of SQA activities that include formal technical reviews, quality and configuration audits, performance monitoring, simulation, feasibility study, documentation review, database review, algorithm analysis, development testing, qualification testing, and installation testing.

Q.4(b) Explain component testing. [5]

Ans.:
- Component testing is one of the testing methodology in which the testing is carried out on every component of software system independently without combining it with other components.
- Component testing is also called as Module Testing because it perform testing on single module at a time.
- Normally software is made up of many components.
- Components Testing concentrate on testing these different components separately.
- Component testing is most frequently used black box testing technique which is performed by Quality Assurance (QA) Team
- Who does component testing?
  - Component testing is done by software testers.
  - The developer perform unit testing in which they check single functional module work as per requirement of end user.
  - After unit testing is performed, the step is to perform component testing which is done by the testers.
- When to perform Component testing:
  - Component testing is performed as soon as Unit Testing is performed by the developer and testing team got software for testing.
  - This software build is known as UT build (Unit Testing Build).
  - Major functionality of all the components are tested in this phase,
- For entry criteria in components testing:
  Minimum number of the components to be present in the Unit Testing build (UT) must be developed and unit testing should be performed on them by software developers.
- Exit criteria for component testing:
  - All the components present of Critical or High or Medium level severity (priority assigned to defect by system point of view) and priority (priority assigned to defect by user point of view) defect in defect log.

Test strategy of Component Testing
Depend on deepness of testing levels, Component testing can be classified as :

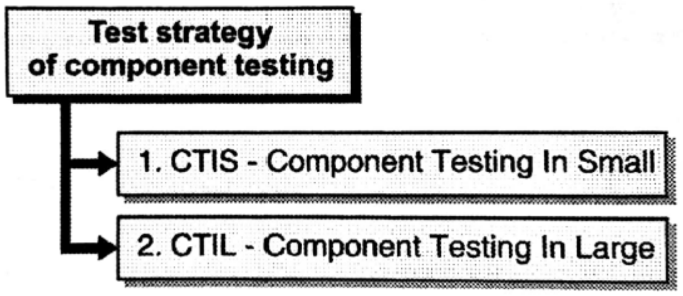

Test strategy of component testing
1. CTIS - Component Testing In Small
2. CTIL - Component Testing In Large

Fig.: Test strategy of component testing

1. CTIS – Component Testing In Small
- Component testing may be performed with or without separating a component from other components which are present in the software application under test.

- If component testing is done with separating component from other component, then that component testing is called as Component Testing in Small.
- For example assume a website which consist of 6 dissimilar web pages, then testing every webpage individually with separating that component from other components, is called as Component testing in Small.

2. **CTIL – Component Testing In Large**
   - Component testing performed without separating current testing component from other components in the software application under test is called Component Testing Large.
   - For example, assume there is an software which contains three components as Component X, Component Y and Component Z.
   - The developer develop the component Y and want to test it, but there is problem in testing Y because some functionality in Y depends on component X and some functionalities in B are depend on component Z. But the component X and component Z are not developed at that time. In this case, to test the component Y completely, we can use concept of stub and driver which are dummy programs and cover functionality from X and Z on which some functionality of Y is depend. Stub is used instead of component X and component Z is replaced by driver.
   - **Stub** : A stub is a program which is called by the software component under test. Normally called components or low level components are replaced by stub.
   - **Driver** : Calling components or high level components are replaced by driver while performing component testing.

Q.4(c) Explain automated testing process. **[5]**

Ans.: Following steps are followed in an Automation Process.

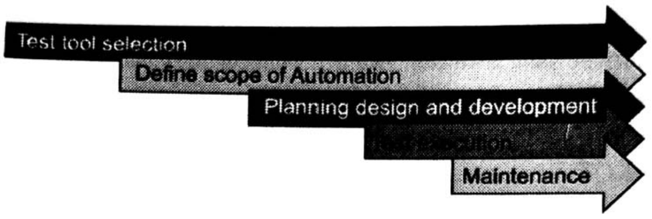


Fig.: Automated Testing Process

(1) Test tool selection : The selection of Test Tool is mainly based on the technology in which the Application Under Test is built on. For example QTP does not support Informatica, hence it cannot be used for testing Informatica applications.

(2) Define the scope of Automation : Scope of automation is nothing but the area of the specific Application which is Under Test for automation.

Following points are consider while determining scope:
- Important features for the business
- Scenarios having **huge data**
- **Common functionalities** among applications
- Feasibility in Technical point of view
- Extent to which the reuse of business components can be implemented
- **Complexity** of test cases
- Ability to use the common test cases for the purpose of cross browser testing

(3) Planning, Design and Development:

In this phase Automation strategy & plan is developed containing following details–
- The specific selected automation tools
- Framework design with its features
- In–Scope as well as Out–of–scope items of automation

- preparation of Automation test bed
- Schedule as well as deadlines of scripting and execution
- Deliverables of Automation Testing

(4) Test Execution

- In this phase the execution of Automation Scripts is done. Before setting the scripts to run, they need input test data. After execution, in detail test report is provided by them.
- For the purpose of execution, the automation tool can be used directly or the Test Management tool can be used which in turn invoke the automation tool.
- For Example, Quality center can be used which is a Test Management tool which can further call QTP for executing automation scripts. A single machine or a group of machines can be used to run scripts. To save time, such execution can be carried out at night.

(5) Maintenance : With each and every successive cycle, new functionalities are added in the System which is Under Test. For each release cycle, there is need of add, review and maintain the Automation Scripts. Maintenance is very essential for the purpose of improving effectiveness of Automation Scripts.

Benefits of Automation Testing:

- near about 70% faster as compared to the manual testing
- Wide range of application features
- Results are reliable
- Assurance of Consistency
- Time and Cost are saved
- Accuracy is improved
- No need of human Intervention while execution
- Efficiency is increased
- Speed of executing tests is better
- Test scripts are reusable
- Test Frequently and thoroughly

Automation Testing Tools:

- Selenium
- QTP (HP UFT)
- Rational Functional Tester
- WATIR
- SilkTest

Q.4(d) Explain size-oriented metrics? [5]

Ans.:
- Size oriented software metrics are in general derived by the process of normalizing quality and/or productivity measures by assuming the software size which has been produced.
- If a software organization maintains simple records, a table of size–oriented measures will be as shown in following Table.

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

- In Table, the details are given about all the software projects developed in last five years.
- **Protect alpha :** 12,100 lines is the total size of code with 24 person months of effort at a price of $$168,000.

- It is important that the effort and cost recorded in the table does not represent only coding, but it also represents various software engineering activities such as analysis, design and testing.
- The further information shows that the size of documentation is of 365 pages.
- Before the release of software the count or errors was 134. Within the first year after release to the customer, the count of defects was 29. The number of people involved in development was three.
- For the purpose of metrics development which can be assimilated with other same types of metrics from various projects, as our normalization value, the Line of code will be selected.
- As per the table information, a set of simple size–oriented metrics can be generated for all the projects:
  - Errors per KLOC (Thousands line of code)
  - Defects per KLOC,
  - $ Per KLOC,
  - Pages of documentation per KLOC
- The Size–oriented metrics are not considered as the best solution for measuring the software process.

Q.4(e) Explain cocomo model. [5]

Ans.: COCOMO means the COnstructive COst MOdel. In the COCOMO model, after determining the initial estimate, some other factors are incorporated for obtaining the final estimate. For the initial estimate (also called nominal estimate) the equation for an organic project is E = 39 (SIZE)91. COCOMO uses a set of 15 different attributes of a project called cost driver attributes. Examples of the attributes are required software reliability, product complexity, analyst capability, application experience, use of modern tools, and required development schedule. Each cost driver has a rating scale, and for each rating, a multiplying factor is provided. For example, for the reliability, the rating scale is very low, low, nominal, high and very high; the multiplying factors for these ratings are 0.75, 0.88, 1.00, 1.15 and 1.40 respectively. So if the reliability requirement for the project is judged to be low, then the multiplying factor is 0.75, while if it is judged to be very high, the factor is 1.40. The attributes and their multiplying factors for different ratings are shown in the following table.

Cost Drivers	**Ratings**				
	Very Low	**Low**	**Nominal**	**High**	**Very High**
Product Attributes					
RELY, required reliability	0.75	0.88	1.00	1.15	1.40
DATA, database size		0.94	1.00	1.08	1.16
CPLX, product complexity	0.70	0.85	1.00	1.15	1.30
Computer Attributes					
TIME, execution time constraint			1.00	1.11	1.30
STOR, main storage constraint			1.00	1.06	1.21
VITR, virtual machine volatility		0.87	1.00	1.15	1.30
TURN, computer turnaround time		0.87	1.00	1.09	1.15
Personnel Attributes					
ACAP, analyst capability	1.46	1.19	1.00	0.86	0.71
AEXP, application exp.	1.29	1.13	1.00	0.91	0.82
PCAP, programmer capability	1.42	1.17	1.00	0.86	0.70
VEXP, virtual machine exp.	1.21	1.10	1.00	0.90	
LEXP, prog. Language exp.	1.24	1.07	1.00	0.95	
Project Attributes					
MODP, modern prog. Practices	1.24	1.10	1.00	0.91	0.82
TOOL, use of SW tools	1.24	1.10	1.00	0.91	0.83
SCHED, development schedule	1.23	1.08	1.00	1.04	1.10

Q.4(f) Explain system testing. [5]

Ans.: System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system.

Recovery Testing :

Many computer based systems must recover from faults and resume processing within a prespecified time. In some cases, a system must be fault tolerant.

Recovery testing is a system test that forces the software to fail in a variety of ways and verifies that recovery is properly performed.

If recovery is automatic (performed by the system itself), reinitialization, checkpointing mechanisms, data recovery, and restart are evaluated for correctness.

If recovery requires human intervention, the mean-time-to-repair (MTTR) is evaluated to determine whether it is within acceptable limits.

Security Testing :

Any computer-based system that manages sensitive information or causes actions that can improperly harm (or benefit) individuals is a target for improper or illegal access.

Security testing attempts to verify that protection mechanisms built into a system will, in fact, protect it from improper penetration. The system's security must, of course, be tested for invulnerability from frontal attack—but must also be tested for invulnerability from flank or rear attack."

Stress Testing :

During earlier software testing steps, white-box and black-box techniques resulted in thorough evaluation of normal program functions and performance.

Stress testing executes a system in a manner that demands resources in abnormal quantity, frequency, or volume.
For example,

(i) special tests may be designed that generate ten interrupts per second, when one or two is the average rate,
(ii) input data rates may be increased by an order of magnitude to determine how input functions will respond,
(iii) test cases that require maximum memory or other resources are executed,
(iv) test cases that may cause thrashing in a virtual operating system are designed,
(v) test cases that may cause excessive hunting for disk-resident data are created. Essentially, the tester attempts to break the program.

A variation of stress testing is a technique called sensitivity testing.
Sensitivity testing attempts to uncover data combinations within valid input classes that may cause instability or improper processing.

Performance Testing :

For real-time and embedded systems, software that provides required function but does not conform to performance requirements is unacceptable.
Performance testing is designed to test the run-time performance of software within the context of an integrated system.

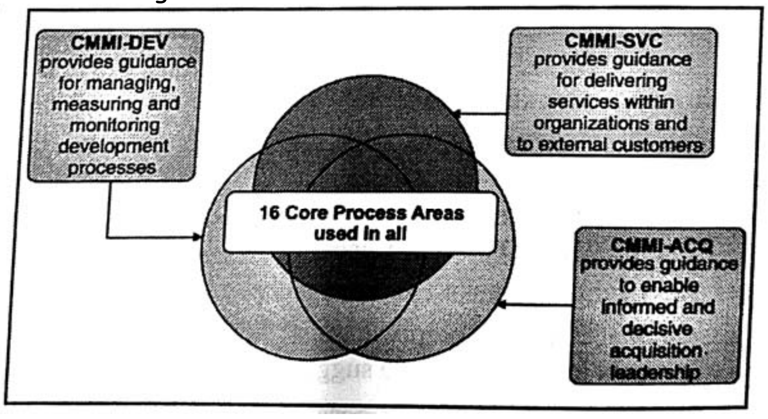
Performance testing occurs throughout all steps in the testing process. Even at the unit level, the performance of an individual module may be assessed as white-box tests are conducted.

Performance tests are often coupled with stress testing and usually require both hardware and software instrumentation. That is, it is often necessary to measure resource utilization (e.g., processor cycles) in an exacting fashion.

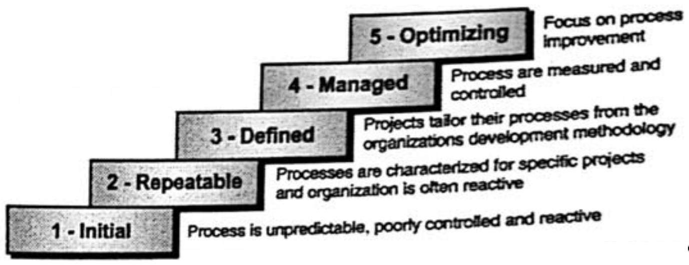
Q.5 Attempt any THREE of the following: [15]

Q.5(a) Explain CMMI process improvement framework? [5]

Ans.:
- Capability Maturity Model Integration (CMMI) is a capability maturity model.
- CMMI framework consists of a collection of computer programs based on knowledge system engineering, software engineering, integrated product and process development and provider sourcing.


CMMI-DEV provides guidance for managing, measuring and monitoring development processes
CMMI-SVC provides guidance for delivering services within organizations and to external customers
16 Core Process Areas used in all
CMMI-ACQ provides guidance to enable informed and decisive acquisition leadership

- CMMI framework has three groups as :
  1. CMMI for Development (CMMI - DEV)
  2. CMMI for Service (CMMI - SVC)
  3. CMMI for Acquisition (CMMI - ACQ)
- These three groups forms model components which are uniquely designed for particular business process and they may contain some core processes as a part of them which will be same among these groups.
- It is also possible to extend CMMI framework by making addition of model component to it.
- In general CMMI has following maturity levels :
  1. Initial
  2. Managed
  3. Defined
  4. Quantitatively managed
  5. Optimizing


5 - Optimizing
Focus on process improvement
4 - Managed
Process are measured and controlled
3 - Defined
Projects tailor their processes from the organizations development methodology
2 - Repeatable
Processes are characterized for specific projects and organization is often reactive
1 - Initial
Process is unpredictable, poorly controlled and reactive

- At Initial stage processes are created for particular purpose and chaotic. Environments in organization are unstable. Thus organizational success is only because of skills and engagements of team members instead of well defined processes.
- At managed level of maturity an organization have all processes as well defined and executed according to rules and guidelines defined by customers or stakeholders. All processes achieves their goals.
- At level defined an organization have well defined or well structured processes which gives detail information about standards, procedures, tools and methods.

- At quantitatively managed maturity level some processes are selected from well defined processes and are analyzed for performance by using various process measurement techniques.
- At the last level of maturity that is optimizing level an organization is continuously improving its processes to make them best in order to improve the efficiency of process.

Q.5(b) Explain service oriented software engineering (SOSE)? [5]

Ans.:
- **Service Oriented Software Engineering (SOSE)** is a well defined approach used for development of software system.
- It encompasses methodology for building software system by combining two or more predefined and reusable services provided by third party.
- Nowadays SOSE is standard approach for software development because of its advantages like reduction in cost and time efforts, extensibility etc.
- SOSE mainly uses Service Oriented Architecture (SOA) for developing the software systems.
- Service oriented architecture (SOA) is architecture for software development process in which services are provided to system components by un-hosted components through the use of network communication.
- Basic principle of SOA is that the services provided by third party are independent of vendors, products and technologies.
- Service may contain other services; it represents organizational activity with required output. It is self contained task and acts like black box for its consumer.
- SOA has following components :

1) **Service Provider :** Service provider is a third party channel which develops reusable service and display its information if service registry so that others will be aware of it. They decides which service to expose and rate offered for services.
2) **Service Registry or Service Repository :** It is list of services describing its purpose and characteristics. Each service must be registered in service registry so that user will be aware of it.
3) **Service Requester or consumer :** It is an organization or development team which want to develop system by using service from the service registry.

Q.5(c) Write note on reuse landscape. [5]

Ans.:
- The reuse landscape means number of possible ways of implementing software reuse.
- Software can be reused in many different ways as shown in Figure.

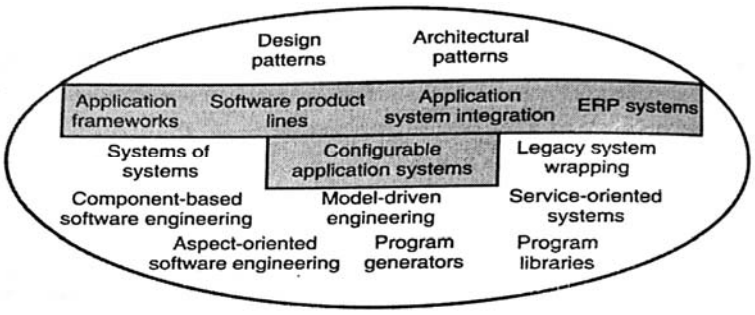


Fig.: The reuse landscape.

They can be described as follow :

- **Design Pattern :** A solution to general problems that can be reused in software development is known as design pattern in software engineering.
- **Architectural Pattern :** Architectural pattern means design pattern with larger scope. It gives solution for variety of issues in software engineering.

- **Application framework :** Software can be reused as an application framework by combining one or more abstract and concrete classes modify them and extend them to create new software application.
- **Software Product Lines :** Software product lines defines common architecture for an software application which can be easily modified to different customers.
- **Application System Integration :** More than one application systems can be combined to develop software with additional functionality.
- **ERP Systems :** ERP systems are widely used systems in business domain they provide various functions generic to many different business domains.
- **Systems of Systems :** As the name suggest it is a collection two or more different distributed systems implemented as a new system to provide required functionality.
- **Configurable Application Systems :** These are the software systems which are designed according to specific domain and can be easily configured to satisfy the requirements of particular customers.
- **Legacy System Wrapping :** Legacy systems are systems which are outdated but still in use since they cannot be replaced. Legacy System wrapping is a process of defining set of common interfaces which gives access to legacy systems.
- **Component Based Software Engineering :** It is an approach to develop to a software by reusing various predefined components.
- **Model Driven Engineering :** In this approach models for particular domain are developed which can be used to build implementation independent modules and code. Such a module and code can be reused to develop various software applications.
- **Service Oriented Systems :** Common services can be shared among many systems which can be provided externally. Systems which are developed by linking shared services are called service oriented systems.
- **Aspect Oriented Software Engineering :** In this approach software is constructed by using shared components at different places when it is compiled. Shared components are components which addresses various limitations of other essential systems.
- **Program Generator :** It is a system which has knowledge about an application and can generate program in respective domain by using knowledge embedded in it.
- **Program Libraries :** As the name suggest program library is a collection of various classes and functions that can be reused in many applications to provide adequate functionality.

Q.5(d) Differentiate between COTS-integrated systems v/s COTS-solution systems. [5]
Ans.:

	COTS-integrated systems	**COTS-solution systems**
1)	Integrated several heterogeneous system products provide customized functionality.	Single product that provides the functionality required by a client.
2)	Flexible solutions may be developed for customer processes.	Based around a generic solution and standardized processes.
3)	Development focus is on system integration, so it is also called as integrated application system.	Development focus is on system configuration, so it is also called as configurable application system.
4)	Owner of the system is responsible for maintenance.	Vendor of the system is responsible for maintenance.
5)	Owner of the system provides the platform for the system.	Vendor of the system provides the platform for the system.

Q.5(e) Write short note on web application frameworks. (WAFs) [5]
Ans.:
- Nowadays most of the software domains are using an important type of framework known as Web Application Framework (WAF).

- As the name indicates these frameworks helps to build dynamic web sites.
- Today most of the IT companies makes use of Model-View-Controller (MVC) architecture of a WAF.
- General format of MVC is as shown in below Figure

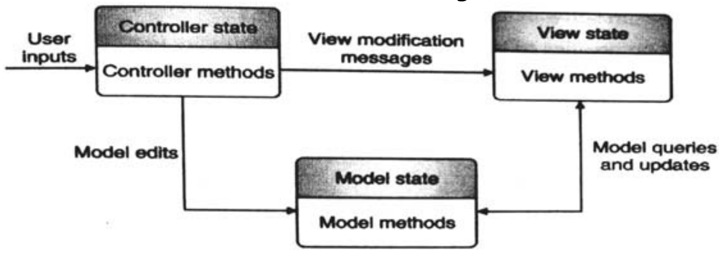


Fig.: Model-View-Controller (MVC) pattern.

Features of Web application frameworks :

1) Security is one of the essential concerns with respect to use of any system. Web Application Framework includes basic functionality for authorization and authentication. This functionality enables restricted and authorize access to system.
2) Now, many web application frameworks have built in capability of web-cache. Responsibility of web-cache is to store copies of accessed documents in order to reduce traffic load on server and its bandwidth usage.
3) Many web application frameworks also support a useful feature called scaffolding. Scaffolding is a programming technique used for writing descriptions that specifies use of database. They are used by application in order to maintain database values in a system.
4) WAFs also supports URL mapping feature which helps in creating user friendly URLs.

Q.5(f) What are the issues in distributed systems. [5]

Ans.: There are several issues while designing the distributed system. These are given in Figure.

1) Heterogeneity :

- The Internet has heterogeneous collection of computers and networks. In a network, the Heterogeneous computers are those which have different hardware components, OSes, or different architectures. The Internet allows users to access services and executes the applications on heterogeneous networks.
- Heterogeneity can be applied to: networks, computer hardware, operating systems, programming languages, and implementations by different developers.

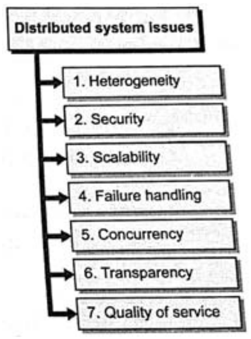


Fig.: Distributed system issues.

2) Security :

- In distributed system several information resources which are accessed and manipulated by users are important for them. So security of them is more important the 3 components of Security for information resources is given below:
- **Confidentiality** : It protects the information resource form leaking to unauthorized persons.
- **Integrity** : It protects the information resources from altering or corruption.
- **Availability** : It protects the information resource from obstacles during accessing the resources.

3) Scalability :

- Even though the great increase in the number of users & resources of a system is occurred, the system should be effectively handling this growth. This type of system is called as scalable system. The Distributed systems are scalable because they operate effectively and efficiently at several scales, varying from a small intranet to the Internet. As we know in the Internet number of computers and servers increases significantly.

- While designing a system for scalable system it should control the cost of physical resources which arc increased on demand. As the system resources grow to extend the system, the relative cost also grows, so the scalable system should handle them.

4) Failure handling :

- Due to occurrence of some faults in hardware or software, programs may generate wrong outputs or may terminate the execution. In the distributed system particular work is distributed over network so if Failure occur in a computer then only this computer goes down and other will work normally so it is difficult to detect that the failure is occur in the system and then failure recovery is done.
- Following are the techniques used for dealing with failures occurred in distributed system :
  (i) Failure detection (ii) Masking failures (iii) Tolerating failures
- When one of the components in a distributed system fails, then only the task given to the failed component is affected. So, if such failure occurs distributed system allows a user to move to the other computer and the server process is started on that newly assigned computer.

5) Concurrency :

- The applications and services offer resources, which are used by distributed systems' users in sharing. Therefore several clients can requests to access the particular shared recourse at a time.
- It can be handling a server which allows only one client to access that resource at a time, which will decrease the throughput of the system.
- So the applications and services allow the concurrent processing of several client requests. This concurrency leads the resources in inconsistent state.
- So while designing the distributed system, make sure that the shared resource of the system operates correctly in a concurrent environment.

6) Transparency :

- To make the distributed system more user friendly the software hides some of the details of the distribution of system resources.
- A distributed system that appears to its users & applications to be a single computer system is said to be transparent.
- Transparency in a system allows users & apps to access remote resources as they access local resources.
- Types of Transparency as given in Table :

Table : Different forms of transparency in a distributed system (ISO, 1995)

Transparency	Description
Access	Hide differences in data representation & resource access. Enables interoperability that indicates that using identical operations both the local as well as remote resources can be accessed.
Location	Hide location of resource. The resources can be accessed without knowing their physical or network locations (for example, IP address)
Migration	Hide possibility that a system may change location of resource. Allows moving the resources and clients in the system and it will not effect on the resource accessibility.
Replication	Hide the possibility that multiple copies of the resource exist. It increases the reliability and availability of resources by using multiple instances of resources, and hides the presence of replicas from user or application programmers.

Concurrency	Hide the possibility that the resource may be shared concurrently. Many different processes are operated concurrently using shared resources.
Failure	Hide failure and recovery of the resource. Even though the failure occurs in the system it will complete their task by recovering the failure.
Relocation	While a resource is in use it hides the movement of that resource form user and application.

- The two most important transparencies are access and location transparency together known as network transparency.

7) Quality of service :

The users of a system request a service to complete their work. The system should be able to give good quality of service to the user. The properties of systems that affect the quality of the service are: reliability, security and performance. While designing the distributed system the system should provide good quality of service.

❑ ❑ ❑ ❑ ❑