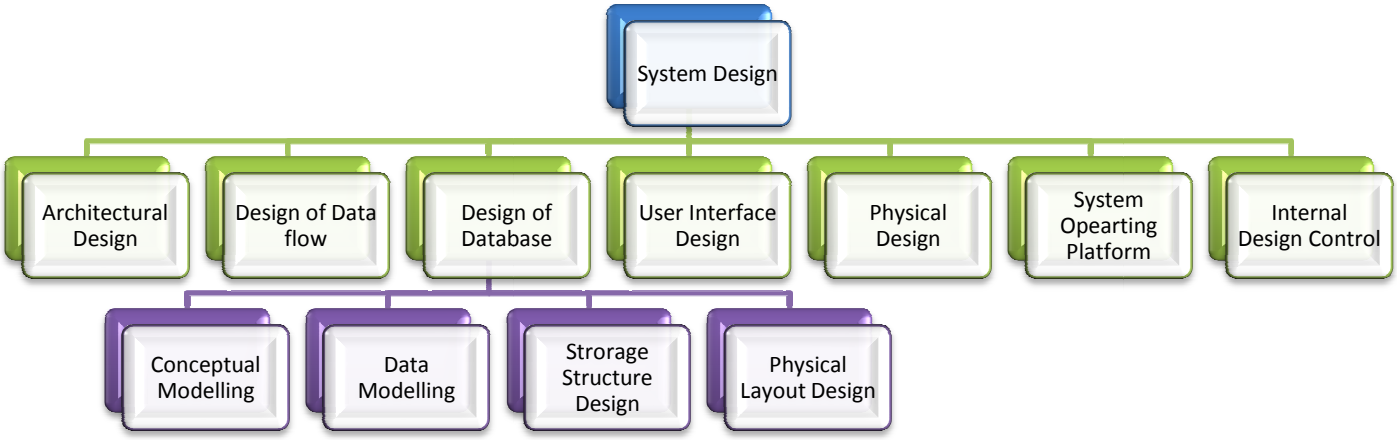
5.5.3 System Designing *(Making a blue print like an architect makes it for building)*

- After the completion of requirements analysis for a system, systems **designing activity takes place for the most feasible and optimal alternative**, which is selected by management.
- The *objective is to design an Information System that best satisfies the users/managerial requirements.* It describes the parts of the system and their interaction.

- Design phase documents/deliverables include a 'blueprint' for the design with the necessary specifications for the hardware, software, people and data resources.
- Design specifications guide the programmers about 'what the system should do and how to implement'. The programmers, in turn, write the programs that accept input from users, process data, produce the reports, and store data in the files.

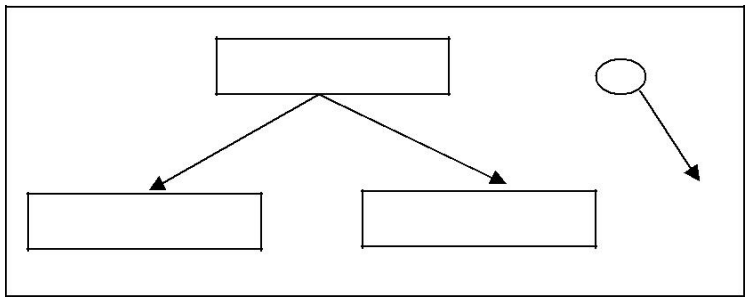
- Once the detailed design is completed, the design is then distributed to the system developers for coding.
- The design phase activities includes Architectural Design; Design of the Data / Information Flow; Design of the Database; Design of the User-interface; Physical Design; and Design and acquisition of the hardware/system software platform', which are described briefly as follows:



(a) Architectural Design:

- Architectural design **deals with the organization of applications in terms of hierarchy of modules and sub-modules.**
- At this stage, we identify major modules; functions and scope of each module; interface features of each module; modules that each module can call directly or indirectly and Data received from / sent to / modified in other modules.
- The architectural **design is made with the help of a tool called Functional Decomposition,** which can be used to represent hierarchies as shown in Figure below.
- It has three elements – Module, Connection, and Couple.

- The module is represented by a box and connection between them by arrows. Couple is data element that moves from one module to another and is shown by an arrow with circular tail.



Functional Decomposition Tool

(b) **Design of Data/Information flow:**

- The design of the data and information flow is a **major step in the conceptual design** of the new system.
- In designing the data / information flow for the proposed system, the inputs that are required are - existing data / information flows, problems with the present system, and objective of the new system. All these have been identified in the analysis phase and documented in Software Requirements Specification (SRS).

(c) Design of Database:

- Design of the database **involves determining its scope ranging from local to global structure**.
- The **scope is decided on the basis of interdependence** among organizational units.
- The design of the database involves four major activities.

May 12: What are the major activities involved in design of Database?

- **Conceptual Modeling:**
  In this step database administrator **design the database by creating Entity-Relationship Diagram** via entities/objects, attributes of these entities/objects and static and dynamic constraints on these entities/objects, and their relationships.

- **Data Modeling:** This is next step of database design. Here **conceptual models are translated into data models** so that they can be accessed and manipulated by both high-level and low- level programming languages.*(E-R diagram ek chart hota ha usko table me cobvert karte ha yaha so that program aur application ye table ko access kar sake)*

- **Storage Structure Design:** Decisions must be made on how to **linearize and partition**

the data structure so that it can be stored on some device *(Here we convert data model into a design that can be stored on hard disk)*. For example- tuples (row) in a relational data model must be assigned to records, and relationships among records might be established via symbolic pointer addresses.

- **Physical Layout Design:** Decisions must be made on **how to distribute the storage structure** across specific storage media and locations for example, the cylinders, tacks, and sectors on a disk and the computers in a LAN or WAN.(Axis bank can either maintain data at head office or different branches can keep their own data)

(d) User Interface Design:
It involves determining the ways in which users will interact with a system.

Factors affecting Input / Output Form Designs

Nov 2000: Discuss the various objectives and factors that should be kept in mind while designing output for information system.
May 2001/May 2004: Discuss the various factors which a system analyst should consider while designing user output.
Nov 2001: Discuss briefly points to be considered while designing input for the computerized system.
Nov 02/Nov 05: What are the major factors to be considered in designing user inputs? Explain.
May 08: Discuss various issues that should be considered while designing system input.

Characteristic	Definition	Input Design	Output Design
Content	Refers to the actual pieces of data to be gathered to produce the required output to be provided to users.	The analyst is required to consider the types of data that are needed to be gathered to generate the desired user outputs. New documents for collecting such information may be designed.	The contents of a weekly output report to a sales manager might consist of sales person's name, sales calls made by each sales person during the week, and the amount of each product sold by each salesperson to each major client category.
Timeliness	Timeliness refers to when users need outputs, which may be required on a	Data needs to be inputted to computer in time because outputs cannot be	A sales manager, may be requiring a weekly sales report. Other users, such as

	regular, periodic basis – perhaps daily, weekly, monthly, at the of quarter or annually.	produced until certain inputs are available. Hence, a plan must be established regarding when different types of inputs will enter the system.	airline agents, require both realtime information and rapid response times in order to render better client service.
Format	Input format refers to the manner in which data are Physically arranged. Output format refers to the Arrangement referring to data output on a printed report or in a display screen.	After the data contents and media requirements are determined, input formats are designed on the basis of few constraints like – the type and length of each data field as well as any other special characteristics (number decimal places etc.).	Format of information reports for the users should be so devised that it assists in decision making, identifying and solving problems, planning and initiating corrective action and searching.
Media	Input-output medium refers to the physical device used for input, storage or output.	This includes the choice of input media and subsequently the devices on which to enter the data. Various user input alternatives may include display workstations, magnetic tapes, magnetic disks, keyboards, optical character recognition, pen-based computers and voice input etc. A suitable medium may be selected depending on the	A variety of output media are available in the market these days which include paper, video display, microfilm, magnetic tape/disk and voice output.

		application to be computerized.	
Form	Form refers to the way the information is inputted in the input form and the content is presented to users in various output forms - quantitative, non quantitative, text, graphics, video and audio.	Forms are pre-printed papers that require people to fill in responses in a Standardized way. Forms elicit and capture information required by organizational members that often will be input to the computer. Through this process, forms often serve as source documents for the data entry personnel.	The form of the output should be decided keeping in view the requirements for the concerned user. For example - Information on distribution channels may be more understandable to the concerned manager if it is presented in the form of a map, with dots representing individual outlets for stores.
Input Volume/ Output Volume	Input volume refers to the amount of data that has to be entered in the computer system at any one time. The amount of data output required at any one time is known as output volume.	In some decision support systems and many real-time processing systems, input volume is light. In batch-oriented transaction processing systems, input volume could be heavy which involves thousands of records that are handled by a centralized data entry department using key-to-tape or key-to-disk systems.	It is better to use high-speed printer or a rapid-retrieval display unit, which are fast and frequently used output devices in case the volume is heavy.

(e) Physical Design:

- For the physical design, the logical design is transformed into units, which in turn can be decomposed further into implementation units such as programs and modules.

- During physical design, the primary concern of the auditor is effectiveness and efficiency issues.
- **Some of the generic design principles being applied to develop the design of typical information systems include the following:**

  - There is a tendency to develop merely one design and consider it the final product. However, the recommended procedure is to design two or three alternatives and choose the best one on pre-specified criteria.
  - The design should be based on the analysis.
  - The software functions designed should be directly relevant to business activities.
  - The design should follow standards laid down. For instance, the user interface should have consistent color scheme, menu structure, location of error message and the like.
  - The design should be modular, with high cohesion and high coupling.

- Moreover, a module is a manageable unit containing data and instructions to perform a well-defined task.
- Interaction among modules is based on well- defined interfaces.
- Modularity is measured by two parameters: Cohesion and Coupling.
- Cohesion refers to the manner in which elements within a module are linked and interacting.
- Coupling is a measure of the interconnection between modules. It refers to the number and complexity of connections between 'calling' and 'called' modules.

(f) **System's Operating Platform:**

In some cases, the **new system requires an operating platform including hardware, network and system software not currently available in an organization.**

For example – a DSS might require high-quality graphics output not supported by the existing hardware and software.

The **new hardware/system software platform** required to support the application system **will then have to be designed** for requisite provisions.

If different hardware and software are not able to communicate with each, subsequent changes will have to be made and resources expanded in trying to make the hardware and software compatible to each other.

(g) Internal Design Controls:

From internal control point of view, this phase is also an important phase as all internal controls are placed in system during this phase. The key control aspects at this stage include the following:

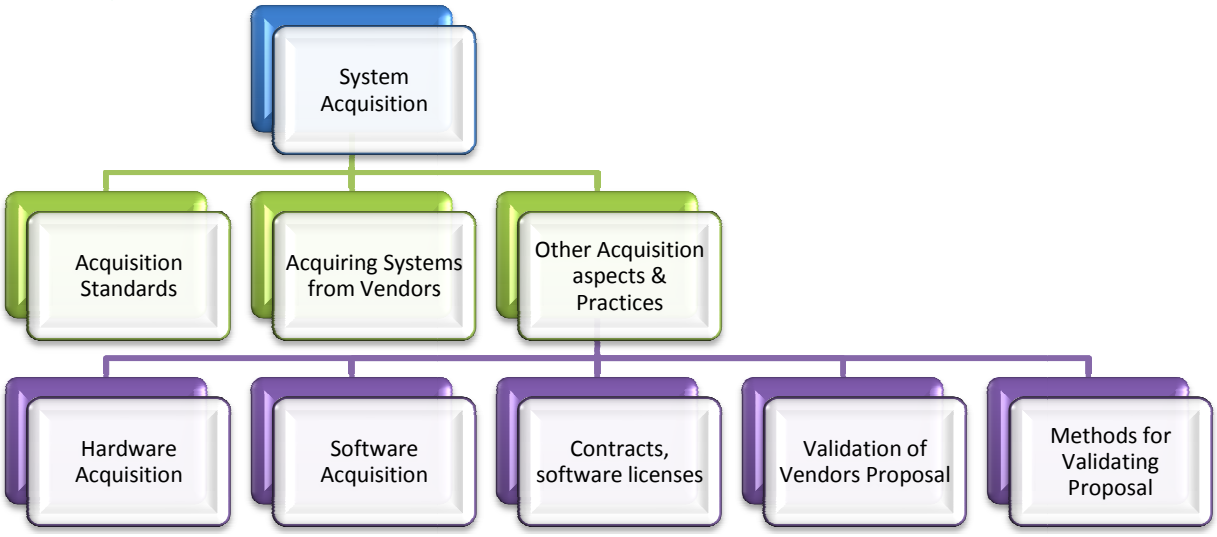
- Whether management reports of stage I and stage II, were referred by System Designer?
- Whether all control aspects have been properly covered?

- Whether controls put in place in system, appear in the documentation done at this stage?
- Whether a separate review of design document has been done by internal auditor?

5.5.4 System Acquisition

After a system is designed either partially or fully, the **next phase of the systems development starts,** which relates to the acquisition of operating infrastructure including hardware, software and services.

Such **acquisitions are highly technical** and cannot be taken easily and for granted. **Thereby, technical specifications, standards etc. come to rescue.**



(a) Acquisition Standards:

Management should establish acquisition standards that address the security and reliability issues as per current state-of-the art development standards. *(Jitna security development standard me address karte utna hi acquisition standard me bhi address karna chaiye)*

Acquisition standards should <u>focus</u> on the following:

- **Ensuring security**, reliability, and functionality **already built** into a product;
- **Ensuring managers complete appropriate vendor contract**, and licensing reviews and acquiring products compatible with existing systems;
- **Invitations-to-tender** soliciting bids from vendors when acquiring hardware or integrated systems of hardware and software;
- **Request-for-proposals** soliciting bids when acquiring off-the-shelf or third-party developed software; and
- To **ensure functional**, security, and operational **requirements to be accurately identified and clearly detailed in request-for-proposals**.

(b) Acquiring Systems Components from Vendors:

- At the end of the design phase, the organization gets a reasonable idea of the types of hardware, software and services; it needs for the system being developed.
- Acquiring the appropriate hardware and software is critical for the success of the whole project. The organization can discover new hardware and software developments in various ways. Management also decides whether the hardware is to be purchased, leased from a third party or to be rented.
- A sub-committee of experts under the steering committee, referred to as 'System Acquisition Committee' is constituted. The sub-committee is mandated to ensure timely and effective completion of this stage.
- The next aspect is call for Request For Proposal (RFP) from vendors.
- This stage is one of the most critical phases for system acquisition; as well defined RFP leads to better acquisition. RFP, means asking vendors to submit proposals for the requirements mentioned.
- RFP process is the initiation of final stages for implementation.
- The requirements analysis and design phase have been completed, before starting of this phase.
- **The following considerations are valid for both acquisition of hardware and software:**

  - **Vendor Selection:**
    This step is a critical step for success of process of acquisition of systems. It is necessary to remember that vendor selection is to be done prior to sending RFP. The result of this process is that 'RFP are sent only to selected vendors'. For vendor selection, following things are kept in mind including the background and locational advantage of the vendor, the financial stability of vendor, the market feedback of vendor performance, in terms of price, services etc.

  - **Geographical Location of Vendor:**
    The issue to look for whether the vendor has local support persons. Otherwise, the proposals

submitted by vendor not as per RFP requirements need to rejected, with no further discussion on such rejected proposals. This stage may be referred to as 'technical validation', that is to check the proposals submitted by vendors, are technically complying with RFP requirements.

- **Presentation by Selected Vendors:**
  All vendors, whose proposals are accepted after "technical validation", are allowed to make presentation to the System Acquisition Team. The team evaluates the vendor's proposals by using techniques.

- **Evaluation of Users Feedback:**
  The best way to understand the vendor systems is to analyze the feedback from present users. Present users can provide valuable feedback on system, operations, problems, vendor response to support calls.

Besides these, some specific considerations for hardware and software acquisition are described as follows:

- The **benchmark tests** to be done for proposed machine. For hardware's, there are specified standard benchmark tests defined based on the nature of hardware. These need to be applied to proposed equipment.

- **Software considerations** that can be current applications programs or new programs that **have been designed to represent planned processing needs.**

- The benchmarking problems are oriented towards testing whether a computer offered by the vendor meets the requirements of the job on hand of the buyer.

- The benchmarking problems would then comprise long jobs, short jobs, printing jobs, disk jobs, mathematical problems, input and output loads etc., in proportion typical of the job mix.

- If the job is truly represented by the selected benchmarking problems, then this approach can provide a realistic and tangible basis for comparing all vendors' proposals. Tests should enable buyer to effectively evaluate cross performance of various systems in terms of hardware performance (CPU and input/output units), compiler language and operating system capabilities, diagnostic messages, ability to deal with certain types of data structures and effectiveness of software utilities.

- Benchmarking problems, however, suffer from a couple of disadvantages. It takes considerable time and efforts to select problems representative of the job mix which itself must be precisely defined. It also requires the existence of operational hardware, software and services of systems. Nevertheless, this approach is very popular because it can test the functioning of vendors' proposal. The manager can extrapolate in the light of the results of benchmarking problems, the

performance of the vendors' proposals on the entire job mix.

(c) Other Acquisition Aspects and Practices:

On addition to the above, there are several other acquisition aspects and practices also, which are given as follows:

(i) Hardware Acquisition:

In case of procuring such machinery as machine tools, transportation equipment, air conditioning equipment, etc., the management can normally **rely on the time tested selection techniques** and the objective selection criteria can be delegated to the technical specialist.

The management depends upon the vendor for support services, systems design, education and training etc., and expansion of computer installation for almost an indefinite period; therefore, this is **not just buying the machine and paying the vendor for it but it amounts to an enduring alliance** with the supplier.

(ii) Software Acquisition:

Once user output and input designs are finalized, the nature of the application software requirements must be assessed by the systems analyst.

This determination helps the systems development team to decide 'what type of application software products is needed' and consequently, the degree of processing that the system needs to handle.

This helps the system developers in deciding about the nature of the systems software and computer hardware that will be most suitable for generating the desired outputs, and also the functions and capabilities that the application software must possess.

At this stage, the system developers must determine whether the application software should be created in-house or acquired from a vendor.

(iii) Contracts, Software Licenses and Copyright Violations:

Contracts between an organization and a software vendor should **clearly describe the rights and responsibilities** of the parties to the contract. The contracts should be in writing with sufficient detail to provide assurances for performance, source code accessibility, software and data security, and other important issues.

Software license is a license that **grants usage permission** to do things with computer software. The usual goal is to authorize activities, which are prohibited by default by copyright law, patent law, trademark law and any other intellectual property rights. The reason for the license, essentially is that virtually all intellectual property laws were enacted to encourage disclosure of the intellectual property.

Copyright laws protect proprietary as well as open-source software. The use of unlicensed software or violations of a licensing agreement expose organizations to possible litigation.

(iv) Validation of Vendors' proposals:

The contracts and software licensing process consists of evaluating and ranking the proposals submitted by vendors and is quite difficult, expensive and time consuming, but in any case it has to

be gone through.
This problem is made difficult by the fact that vendors would be offering a variety of configurations. **The following factors have to be considered towards rigorous evaluation.**

- The Performance capability of each proposed System in Relation to its Costs;
- The Costs and Benefits of each proposed;
- The Maintainability of each proposed;
- The Compatibility of each proposed system with Existing Systems; and
- Vendor Support.

(v) Methods of Validating the proposal: Large organizations would naturally tend to adopt a sophisticated and objective approach to validate the vendor's proposal. Some of the validation methods are given as follows:

- **Checklists:**
  - It is the most **simple and a subjective method** for validation and evaluation.
  - The various **criteria are put in check list** in the form of suitable questions against which the responses of the various vendors are validated.
  - For example, Support Service Checklists may have parameters like Performance; System development, Maintenance, Conversion, Training, Back-up, Proximity, Hardware and Software.

- **(Nov 2003)Point-Scoring Analysis:**
  - Point-scoring analysis **provides an objective means** of selecting a final system.
  - There are **no absolute rules** in the selection process, **only guidelines for matching user needs** with software capabilities.
  - **Limitation:** Even for a small business, the evaluators must consider such issues as the company's data processing needs, its in-house computer skills, vendor reputations, software costs, and so forth.
    (Even for small business itna mehnat karna padenga)

Point Scoring Analysis List

Software Evaluation Criteria	Possible points	Vendo A	B	C
Does the software meet all mandatoryspecifications?	10	7	9	6
Will program modifications, if any, be minimal to meet company needs?	10	8	9	7

Does the software contain adequate controls?	10	9	9	8
Is the performance (speed, accuracy, reliability, etc.) adequate?	10	7	9	6
Are other users satisfied with the software?	8	6	7	5
Is the software user-friendly?	10	7	8	6

Public Evaluation Reports:

- Several consultancy as well as independent **agencies compare and contrast the hardware and software performance** for various manufacturers and publish their reports in this regard. This method has been **frequently and usefully employed** by several buyers in the past.
- For those criteria, however, **where published reports are not available, reports would have to be made to other methods of validation**.
- This method is particularly useful where the buying staff has inadequate knowledge of facts.

May 2000: Benchmarking Problems related Vendor's Solutions:

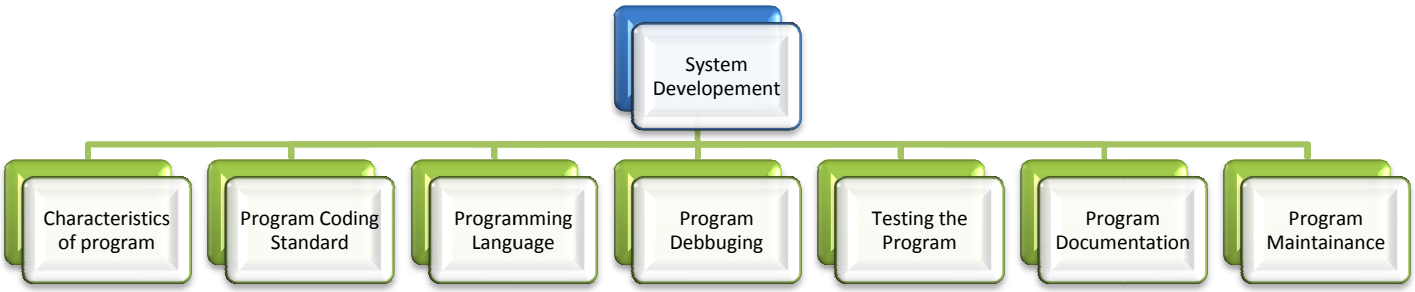
- Benchmarking problems for vendors' proposals are **sample programs** that represent at least a part of the buyer's primary work load.
- They **include software considerations** and can be current application programs or new programs that have been designed to represent planned processing needs.
- That is, benchmarking problems are **oriented towards testing** whether a solution offered by the vendor **meets the requirements of the job on hand of the buyer.**
- This approach is **very popular as it can test the functioning of** vendor's **proposal on the job on hand.**

Testing Problems:

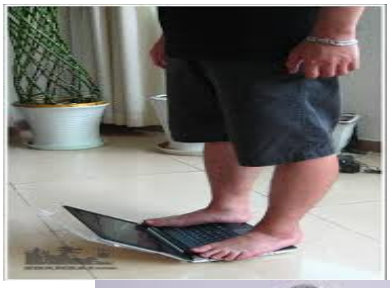
- Test problems disregard the actual job mix and are **devised to test the true capabilities** of the hardware, software or system.
- For example, test problems may be developed to evaluate
  - the time required to translate the source code (program in an assembly or a high level language) into the object code (machine language),
  - response time for two or more jobs in multi-programming environment,
  - overhead requirements of the operating system in executing a user program,
  - length of time required to execute an instruction, etc.
- The **results, achieved by the machine can be compared and price performance** judgment can be made.

5.5.5 System Development : Programming Techniques and Languages

This phase is supposed to **convert the design specifications into a functional system** under the planned operating system environments.
Application programs are written, tested and documented, conduct system testing. Finally it results into a fully functional and documented system.



A good coded program should have the following characteristics:
Nov 12: What are the characteristics of a good coded program?
Code: 3R's make use of efficiency accurately.




1. **Reliability:** It refers to the **consistence which a program provides** over a period of time. However poor setting of parameters and hard coding some data, subsequently could result in the failure of a program after some time.

2. **Robustness:** It refers to the process of **taking into account all possible inputs and outputs** of a program in case of least likely situations.

3. **Readability:** It refers to the **ease of maintenance** of program even in the absence of the program developer.

4. **Usability:** It refers to a **user-friendly interface** and easy-to-understand document required for any program.

5. **Efficiency:** It refers to the performance which should **not be unduly affected** with the increase in input values.

6. **Accuracy:** It refers not only to **what program is supposed to do**, but should also take care of what it should not do. The second part becomes more challenging for quality control personnel and auditors.

May 02: Explain briefly the stages through which program has to pass during development.
Nov 05/May 07: Discuss various stages through which an in-house creation of program has to pass.
Answer: Point (a) to (f)
(Tutorial Note: Students please don't write 7 phases of SDLC when this question is asked in exam- & stages given on pg. 20 are altogether different answer)

(a) Program Coding Standards:

- The **logic** of the program outlined in the flowcharts **is converted into program statements** or instructions at this stage.
- For each language, there are specific rules concerning format and syntax.
- Syntax means vocabulary, punctuation and grammatical rules available in the language manuals that the programmer has to follow strictly and pedantically*(meticulously)*.
- **Different programmers may write a program using different sets of instructions but each giving the same results.**
- **Therefore, the coding standards are defined**, which serves as a method of communication between teams, amongst the team members and users, thus working as a good control.
- Coding standards *minimize the system development setbacks due to programmer turnover*.

(b) Programming Language:

Application programs are coded in the form of statements or instructions(Source code) and the same is converted by the compiler to object code for the computer to understand and execute.
The programming languages*(for writing source code)* commonly used are given as follows :

- *High level general purpose programming* languages such as COBOL and C;
- *Object oriented* languages such as C++, JAVA etc.;
- *Scripting language* such as JAVAScript, VBScript; and
- *Decision Support or Logic Programming* languages such as LISP and PROLOG.

(Intentionally left blank)

(c) Program Debugging:

Nov 2001: What is program debugging:

- Debugging is the **most primitive form of testing** activity, which refers to **correcting programming language syntax***(gramettical mistakes)* **and diagnostic errors** so that the program compiles cleanly.
- A **clean compile means** that the program can be successfully converted from the source code written by the programmer into machine language instructions.

Debugging can be a tedious task consisting of following four steps:

1. Giving **input the source program** to the compiler,
2. Letting the **compiler to find errors** in the program,
3. **Correcting lines** of code **that are erroneous**, and
4. **Resubmitting the corrected source program** as input to the compiler.

(d) Testing the Programs:

- A careful and thorough testing of each program is **imperative to the successful installation** of any system.
- The programmer **should plan the testing to be performed**.
- The **test plan should require the execution of all standard processing logic** based on chosen testing strategy/techniques.
- A **log of test results** and all conditions successfully tested **should be kept**.

(e) Program Documentation:

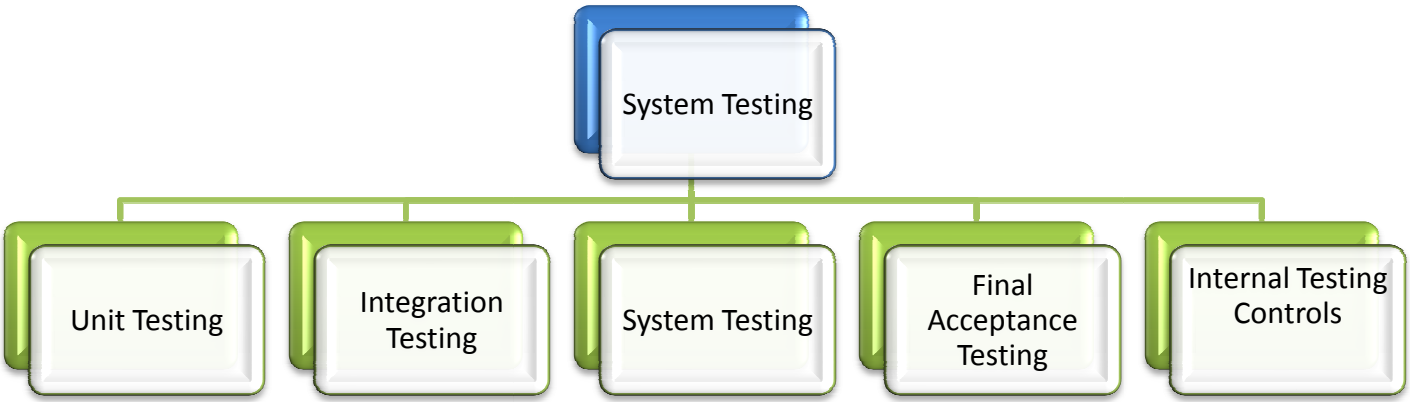
- The **writing of narrative procedures and instructions for people**, **who will use software** is done throughout the program life cycle.
- Managers and users should carefully **review documentation in order to ensure that the software and system behave as the documentation indicates.** If they do not, documentation should be revised.
- **User documentation should also be reviewed for understandability** i.e. the documentation should be prepared in such a way that the user can clearly understand the instructions.

(f) Program Maintenance:

- **The requirements** of business data processing applications **are subject to periodic change**.
- **This calls for modification** of various programs.

5.5.6 System Testing

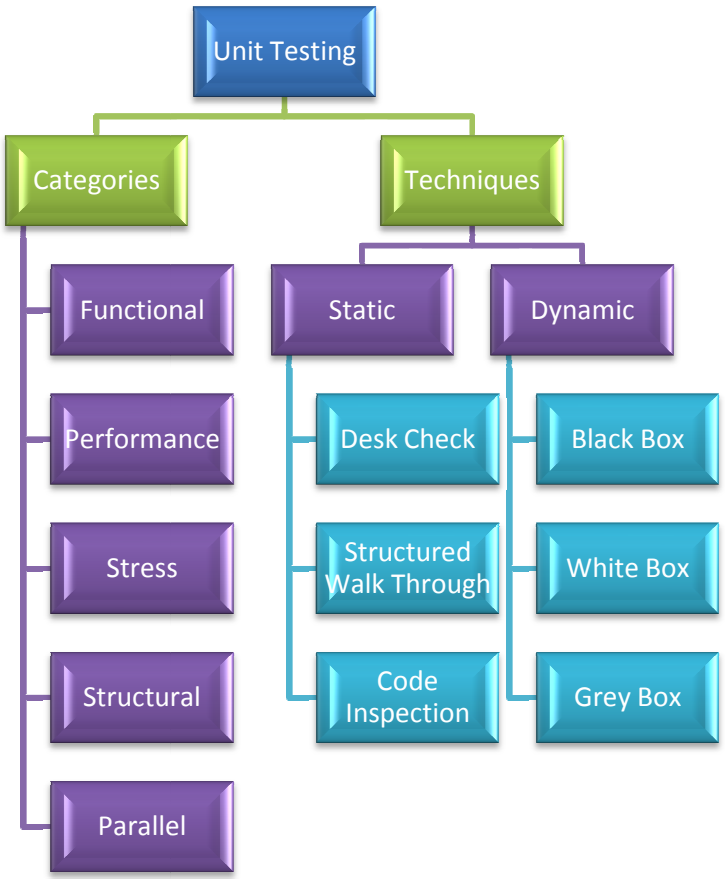
Testing is a process used to **identify the correctness, completeness and quality** of developed computer software. *(eg: paper checker)*



Testing should **systematically uncover different classes of errors** in a minimum amount of time and with a minimum amount of effort.
The data collected through testing can also provide an indication of the software's reliability and quality. However, **testing cannot show the absence of defect**, **it can only show that software defects are present.**

Different levels/facets of Testing are described as follows.

(i) Unit Testing:



- In computer programming, unit testing is software **verification and validation method** in which a programmer tests **if individual units** of source code are fit for use.
- A unit is the **smallest testable part** of an application which may be an individual program, function, procedure, etc.
- Unit tests are typically **written and run by software developers** to ensure that code meets its design and behaves as intended.
- The goal of unit testing is to **isolate each part** of the program **and show that the individual parts are correct.**

There are five Categories of tests that a programmer typically performs on a program unit. Such typical tests are described as follows: *('C' for cricketer c for Categoroies- this code is necessary so that students don't get confused with techniques given on pg. 57 of this notes)*



Functional Tests:

- Functional Tests check **'whether programs do what they are supposed** to do or not'. *(wicket keeping karna ha ya caption banna ha)*
- The test plan specifies operating conditions, input values, and expected results, and as per this plan programmer checks by inputting the values to see whether the actual result and expected result match.

Performance Tests: *(Virat performs with a centurty)*

- Performance Tests should be designed to **verify the response time, the execution time, the throughput, primary and secondary memory utilization** and the traffic rates on data channels and communication links.

Stress Tests: (*Dravid defensive khel khel k bowler ki line and length bigad denga)*

- Stress testing is a form of testing that is used to **determine the stability** of a given system or entity.
- It **involves testing beyond normal operational capacity**, often to a breaking point, in order to observe the results.
- These tests are designed to overload a program in various ways.

- The purpose of a stress test is **to determine the limitations** of the program.
- For example, during a sort operation, the available memory can be reduced to find out whether the program is able to handle the situation.

Structural Tests: *(leg ka haddi check kar raha ha i.e Internal Structure)*

- Structural Tests are concerned with **examining the internal processing logic** of a software system.
- For example, if a function is responsible for tax calculation, the verification of the logic is a structural test.

Parallel Tests:

- In Parallel Tests, the **same test data is used in the new and old system** and the output results are then compared.

In terms of techniques, Unit Testing is classified as Static Analysis Testing and Dynamic Testing. Such typical testing techniques are elaborated as follows:

(a) Static Testing:

Static Analysis Tests are conducted on source programs and do not normally require executions in operating conditions. Typical static analysis techniques include the following:

- **Desk Check:** *(Eg: Student reading his paper again after completing it)*

This is **done by the programmer him/herself**. S/he checks for logical syntax errors, and deviation from coding standards.

- **Structured Walk Through:** (*Eg: Student wrote a mock paper and telling his checker that ye ans iske bad likha ha etc etc)*

The application developer leads other programmers to scan through the text of the program and explanation to uncover errors.

- **Code Inspection:** *(Eg: Aur finally apne CA final ka paper likha jo ek third party check karenga)*

The program is reviewed by a formal committee. Review is done with formal checklists.

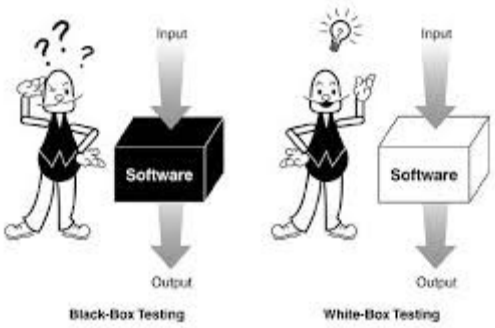
(b) Dynamic Analysis Testing:

Such testing is normally conducted through execution of programs in operating conditions. Typical techniques for dynamic testing and analysis include the following**:**

Black Box Testing:

- **Black Box Testing** takes an **external perspective** of the test object to derive test cases.
- These tests can be functional or non-functional *(Testing the application based on the clients and performance requirement)* though **usually functional** *(Functions are tested by feeding them input and examining the output, and internal program structure is rarely considered (not like in white-box testing). Functional Testing usually describes what the system does).*

- The test designer **selects valid and invalid inputs** and determines the correct output.
- There is **no knowledge of the test object's** internal structure.



- This method of test design is **applicable to all levels** of software testing: unit, integration, system and acceptance.
- The **higher the level**, hence the bigger and more complex the box, the **more one is forced to use black box testing to simplify.**
- While this method can uncover unimplemented parts of the specification, **one cannot be sure that all existent paths are tested.**
- If a **module performs a function which is not supposed** to, the black box **test does not identify it.**

White Box Testing:

- **White box testing** uses an **internal perspective** of the system to design test cases based on internal structure.
- It **requires programming skills** to identify all paths through the software.
- The tester **chooses test case inputs to exercise paths** through the code and determines the appropriate outputs.
- Since the tests are based on the actual implementation, **if the implementation changes, the tests probably will need to change, too.**
- It is **applicable at the unit, integration and system**, ~~acceptance~~ levels of the testing process, it is typically applied to the unit.
- While it normally tests paths within a unit, it can also test paths between units during integration, and between subsystems during a system level test. *(upar vali line ghuma fira k likhi k integration level aur system level pe bhi kam aa sakta ha)*
- **After obtaining a clear picture of the internal workings** of a product, **tests can be conducted** to ensure that the internal operation of the product conforms to specifications and all the internal components are adequately exercised.

Gray Box Testing:

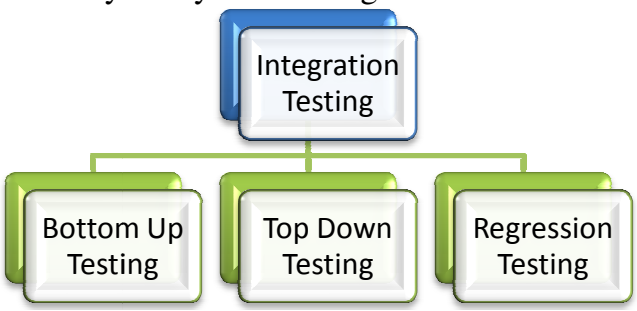
- **Gray box testing** is a software testing technique that **uses a combination of** black box testing and white box testing.
- In gray box testing, the tester applies a limited number of test cases to the internal workings of

the software under test.

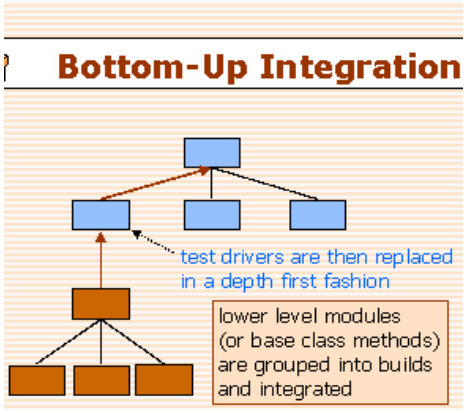
- In the remaining part of the gray box testing, one takes a black box approach in applying inputs to the software under test and observing the outputs.

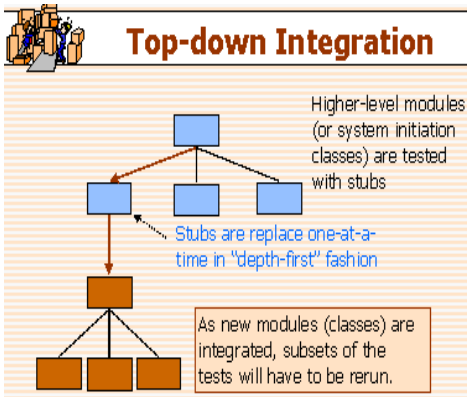
(ii)Integration Testing:

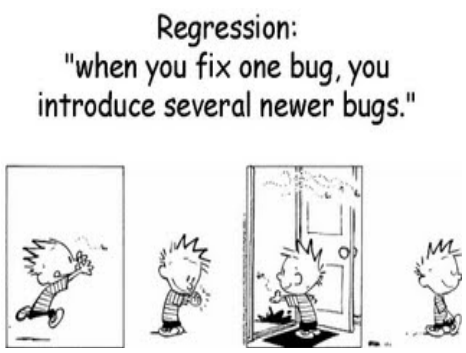
- Integration testing is an activity of software testing in which **individual software modules are combined and tested** as a group.
- It occurs after unit testing and before system testing with an **objective to evaluate the validity of connection of two or more components** that pass information from one area to another.
- Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.



This is carried out in the following two manners:







Bottom-up Integration:

- is the **traditional strategy** used to integrate the components of a software system into a functioning whole.
- It consists of unit testing, followed by sub-system testing, and then testing of the entire system.
- Bottom-up testing is **easy to implement** as at the time of module testing, tested subordinate modules are available.
- The disadvantage, however is that testing of **major decision/control points is deferred to a later period.**

Top-down Integration: *(opposite of bottom up)*

- **starts with the main routine**, and stubs are substituted, for the modules directly subordinate to the main module.

- ***An incomplete portion of a program code that is put under a function in order to allow the function and the program to be compiled and tested, is referred to as a stub.***

- **Once the main module testing is complete, stubs are substituted with real modules** one by one, and these modules are tested with stubs.
- This **process continues till the atomic modules** are reached.
- Since **decision- making processes are likely to occur in the higher levels** of program hierarchy, the top-down strategy emphasizes on major control decision points encountered in the earlier stages of a process and detects any error in these processes.
- The **difficulty arises in the top-down method, because the high-level modules are tested**, not with real outputs from subordinate modules, but **from stubs.**

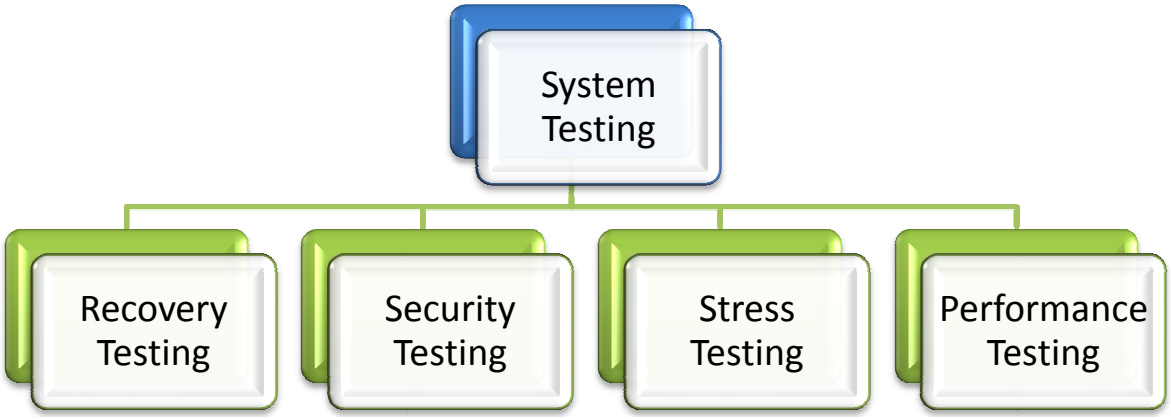
Regression Testing: *(Eg: Ghar me 3 family members ache se rehte the, ek aur member k ane se ho sakta ha existing members ko bich jagda hone lage)*

- **Each time a new module is adde**d as part of integration testing, the **software changes**.
- New data flow paths are established, new I/O may occur and new control logic is invoked.
- These **changes may cause problems with functions that previously worked flawlessly.**
- In the context of the integration testing, the regression tests ensure that changes or corrections have not introduced new errors.
- The data used for the regression tests should be the same as the data used in the original test.

(iii) System Testing:

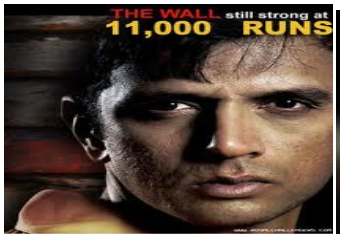
May 01/Nov 13: Write a short note on system testing

- It is a process in which **software and other system elements are tested AS A WHOLE**.
- System testing **begins** *either when* **the software as a whole is operational** *or when* the **well-defined subsets of the software's functionality have been implemented**.
- The **purpose** of system testing **is** to ensure that the new or modified **system functions properly.**
- These test procedures are **often performed in a NON-production test environment**.



The types of testing that might be carried out are as follows:





- **Recovery Testing:**
  - This is the activity of testing '**how well** the application is **able to recover from crashes, hardware failures** and other similar problems'.
  - Recovery testing is the **forced failure** of the software in a variety of ways **to verify that recovery is properly performed.**

- **Security Testing:**
  - This is the process to determine that an Information System **protects data and maintains functionality as intended or not.**
  - The six basic security concepts that need to be covered by security testing are – confidentiality, integrity, authentication, authorization, availability and non-repudiation.
  - This testing technique also ensures the **existence and proper execution of access controls** in the new system.

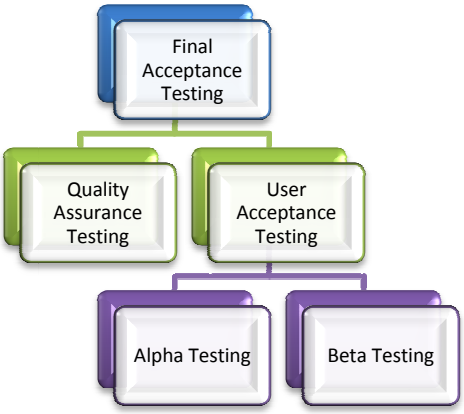
- **Stress or Volume Testing:**
  - Stress testing is a form of testing that is used to **determine the stability** of a given system or entity.
  - It involves **testing beyond normal operational capacity**, often to a breaking point, in order to observe the results.
  - Stress testing may be performed by **testing the application with large quantity of data during peak hours** to test its performance. *(large quantity of data se test karne ka vo bhi September me)*

- **Performance Testing:**
  - In the computer industry, software performance testing is used to **determine the speed or effectiveness** of a computer, network, software program or device.
  - This testing technique **compares** the new system's performance **with that of similar systems using well defined benchmarks***(sachin)*. *(every player will be compared with sachin)*

(Intentionally left blank)

(iv) Final Acceptance Testing:

It is conducted when the system is just ready for implementation. During this testing, it is ensured that the new system satisfies the quality standards adopted by the business and the system satisfies the users.



- **Quality Assurance Testing:**

It ensures that the new system satisfies the prescribed quality standards and the development process is as per the organization's quality assurance policy, methodology and prescriptions.

- **User Acceptance Testing:**

It ensures that the functional aspects expected by the users have been well addressed in the new system. There are two types of the user acceptance testing described as follows:

- **Alpha Testing:** This is the first stage, often performed by the users within the organization by the developers, to improve and ensure the quality/functionalities as per users satisfaction.

- **Beta Testing:** This is the second stage, generally performed after the deployment of the system. It is performed by the external users, during the real life execution of the project. It normally involves sending the product outside the development environment for real world exposure and receives feedback for analysis and modifications, if any.

(v) Internal Testing Controls: *(EXPECTED QUESTION)*

There are several controls that can be exercised internally to assure the testing phase quality and efficiency.

Though it varies from one organization to another, **some of the generic key control aspects appear to be addressed by the responses to following queries:**

- Whether the **test-suite** prepared by the testers **includes the actual business scenarios?**