Software Engineering Technical questions

Q. What is computer software?

A. Computer software is a complete package, which includes software program, its documentation and user guide on how to use the software.

Q. Can you differentiate computer software and computer program?

A. A computer program is piece of programming code which performs a well-defined task whereas software includes programming code, its documentation and user guide.

Q. What is software engineering?

A. Software engineering is an engineering branch associated with software system development.

Q. When you know programming, what is the need to learn software engineering concepts?

A. A person who knows how to build a wall may not be good at building an entire house. Likewise, a person who can write programs may not have knowledge of other concepts of Software Engineering. The software engineering concepts guide programmers on how to assess requirements of end user, design the algorithms before actual coding starts, create programs by coding, testing the code and its documentation.

Q. What is software process or Software Development Life Cycle (SDLC)?

A. Software Development Life Cycle, or software process is the systematic development of software by following every stage in the development process namely, Requirement Gathering, System Analysis, Design, Coding, Testing, Maintenance and Documentation in that order.

Q. What are SDLC models available?

A. There are several SDLC models available such as Waterfall Model, Iterative Model, Spiral model, V-model and Big-bang Model etc.

Q. What are various phases of SDLC?

A. The generic phases of SDLC are: Requirement Gathering, System Analysis and Design, Coding, Testing and implementation. The phases depend upon the model we choose to develop software.

Q. Which SDLC model is the best?

A. SDLC Models are adopted as per requirements of development process. It may very software-to-software to ensuring which model is suitable.

We can select the best SDLC model if following answers are satisfied -

- Is SDLC suitable for selected technology to implement the software?
- Is SDLC appropriate for client's requirements and priorities?
- Is SDLC model suitable for size and complexity of the software?
- Is the SDLC model suitable for type of projects and engineering we do?
- Is the SDLC appropriate for the geographically co-located or dispersed developers?

Q. What is software project management?

A. Software project management is process of managing all activities like time, cost and quality management involved in software development.

Q. Who is software project manager?

A. A software project manager is a person who undertakes the responsibility of carrying out the software project.

Q. What does software project manager do?

A. Software project manager is engaged with software management activities. He is responsible for project planning, monitoring the progress, communication among stakeholders, managing risks and resources, smooth execution of

development and delivering the project within time, cost and quality contraints.

Q. What is software scope?

A. Software scope is a well-defined boundary, which encompasses all the activities that are done to develop and deliver the software product.

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

Q. What is project estimation?

A. It is a process to estimate various aspects of software product in order to calculate the cost of development in terms of efforts, time and resources. This estimation can be derived from past experience, by consulting experts or by using pre-defined formulas.

Q. How can we derive the size of software product?

A. Size of software product can be calculated using either of two methods -

- Counting the lines of delivered code
- Counting delivered function points

Q. What are function points?

A. Function points are the various features provided by the software product. It is considered as a unit of measurement for software size.

Q. What are software project estimation techniques available?

A. There are many estimation techniques available. The most widely used are -

- Decomposition technique (Counting Lines of Code and Function Points)
- Empirical technique (Putnam and COCOMO).

Q. What is baseline?

A. Baseline is a measurement that defines completeness of a phase. After all activities associated with a particular phase are accomplished, the phase is complete and acts as a baseline for next phase.

Q. What is Software configuration management?

A. Software Configuration management is a process of tracking and controlling the changes in software in terms of the requirements, design, functions and development of the product.

Q. What is change control?

A. Change control is function of configuration management, which ensures that all changes made to software system are consistent and made as per organizational rules and regulations.

Q. How can you measure project execution?

A. We can measure project execution by means of Activity Monitoring, Status Reports and Milestone Checklists.

Q. Mention some project management tools.

A. There are various project management tools used as per the requirements of software project and organization policies. They include Gantt Chart, PERT Chart, Resource Histogram, Critical Path Analysis, Status Reports, Milestone Checklists etc.

Q. What are software requirements?

A. Software requirements are functional description of proposed software system. Requirements are assumed to be the description of target system, its functionalities and features. Requirements convey the expectations of users from the system.

Q. What is feasibility study?

A. It is a measure to assess how practical and beneficial the software project development will be for an organization. The software analyzer conducts a thorough study to understand economic, technical and operational feasibility of the project.

- **Economic** - Resource transportation, cost for training, cost of additional utilities and tools and overall estimation of costs and benefits of the project.
- **Technical** - Is it possible to develop this system? Assessing suitability of machine(s) and operating system(s) on which software will execute, existing developers' knowledge and skills, training, utilities or tools for project.
- **Operational** - Can the organization adjust smoothly to the changes done as per the demand of project? Is the problem worth solving?

Q. How can you gather requirements?

A. Requirements can be gathered from users via interviews, surveys, task analysis, brainstorming, domain analysis, prototyping, studying existing usable version of software, and by observation.

Q. What is SRS?

A. SRS or Software Requirement Specification is a document produced at the time of requirement gathering process. It can be also seen as a process of refining requirements and documenting them.

Q. What are functional requirements?

A. Functional requirements are functional features and specifications expected by users from the proposed software product.

Q. What are non-functional requirements?

A. Non-functional requirements are implicit and are related to security, performance, look and feel of user interface, interoperability, cost etc.

Q. What is software measure?

A. Software Measures can be understood as a process of quantifying and symbolizing various attributes and aspects of software.

Q. What is software metric?

A. Software Metrics provide measures for various aspects of software process and software product. They are divided into –

- Requirement metrics: Length requirements, completeness
- Product metrics: Lines of Code, Object oriented metrics, design and test metrics
- Process metrics: Evaluate and track budget, schedule, human resource.

Q. What is modularization?

A. Modularization is a technique to divide a software system into multiple discreet modules, which are expected to carry out task(s) independently.

Q. What is concurrency and how it is achieved in software?

A. Concurrency is the tendency of events or actions to happen simultaneously. In software, when two or more processes execute simultaneously, they are called concurrent processes.

Example

While you initiate print command and printing starts, you can open a new application.

Concurrency, is implemented by splitting the software into multiple independent units of execution namely processes and threads, and executing them in parallel.

Q. What is cohesion?

A. Cohesion is a measure that defines the degree of intra-dependability among the elements of the module.

Q. What is coupling?

A. Coupling is a measure that defines the level of inter-dependability among modules of a program.

Q. Mentions some software analysis & design tools?

A. These can be: DFDs (Data Flow Diagrams), Structured Charts, Structured English, Data Dictionary, HIPO (Hierarchical Input Process Output) diagrams, ER (Entity Relationship) Diagrams and Decision tables.

Q. What is level-0 DFD?

A. Highest abstraction level DFD is known as Level 0 DFD also called a context level DFD, which depicts the entire information system as one diagram concealing all the underlying details.

Q. What is the difference between structured English and Pseudo Code?

A. Structured English is native English language used to write the structure of a program module by using programming language keywords, whereas, Pseudo Code is more close to programming language and uses native English language words or sentences to write parts of code.

Q. What is data dictionary?

A. Data dictionary is referred to as meta-data. Meaning, it is a repository of data about data. Data dictionary is used to organize the names and their references used in system such as objects and files along with their naming conventions.

Q. What is structured design?

A. Structured design is a conceptualization of problem into several well-organized elements of solution. It is concern with the solution design and based on 'divide and conquer' strategy.

Q. What is the difference between function oriented and object oriented design?

A. Function-oriented design is comprised of many smaller sub-systems known as functions. Each function is capable of performing significant task in the

system. Object oriented design works around the real world objects (entities), their classes (categories) and methods operating on objects (functions).

Q. Briefly define top-down and bottom-up design model.

A. Top-down model starts with generalized view of system and decomposes it to more specific ones, whereas bottom-up model starts with most specific and basic components first and keeps composing the components to get higher level of abstraction.

Q. What is the basis of Halstead's complexity measure?

A. Halstead's complexity measure depends up on the actual implementation of the program and it considers tokens used in the program as basis of measure.

Q. Mention the formula to calculate Cyclomatic complexity of a program?

A. Cyclomatic complexity uses graph theory's formula: $V(G) = e – n + 2$

Q. What is functional programming?

A. Functional programming is style of programming language, which uses the concepts of mathematical function. It provides means of computation as mathematical functions, which produces results irrespective of program state.

Q. Differentiate validation and verification?

A. Validation checks if the product is made as per user requirements whereas verification checks if proper steps are followed to develop the product.

Validation confirms the right product and verification confirms if the product is built in a right way.

Q. What is black-box and white-box testing?

A. Black-box testing checks if the desired outputs are produced when valid input values are given. It does not verify the actual implementation of the program.

White-box testing not only checks for desired and valid output when valid input is provided but also it checks if the code is implemented correctly.

Criteria	Black Box Testing	White Box Testing
Knowledge of software program, design and structure essential	No	Yes
Knowledge of Software Implementation essential	No	Yes
Who conducts this test on software	Software Testing Employee	Software Developer
baseline reference for tester	Requirements specifications	Design and structure details

Q. Quality assurance vs. Quality Control?

A. Quality Assurance monitors to check if proper process is followed while software developing the software.

Quality Control deals with maintaining the quality of software product.

Q. What are various types of software maintenance?

A. Maintenance types are: corrective, adaptive, perfective and preventive.

- **Corrective**

  Removing errors spotted by users

- **Adaptive**

  tackling the changes in the hardware and software environment where the software works

- **Perfective maintenance**

  implementing changes in existing or new requirements of user

- **Preventive maintenance**

  taking appropriate measures to avoid future problems

Q. What is software re-engineering?

A. Software re-engineering is process to upgrade the technology on which the software is built without changing the functionality of the software. This is done in order to keep the software tuned with the latest technology.

Q. What are CASE tools?

A. CASE stands for Computer Aided Software Engineering. CASE tools are set of automated software application programs, which are used to support, accelerate and smoothen the SDLC activities.

SDLC

Software Development Life Cycle, SDLC for short, is a well-defined, structured sequence of stages in software engineering to develop the intended software product.

SDLC Activities

SDLC provides a series of steps to be followed to design and develop a software product efficiently. SDLC framework includes the following steps:

Communication

This is the first step where the user initiates the request for a desired software product. He contacts the service provider and tries to negotiate the terms. He submits his request to the service providing organization in writing.

Requirement Gathering

This step onwards the software development team works to carry on the project. The team holds discussions with various stakeholders from problem domain and tries to bring out as much information as possible on their requirements. The requirements are contemplated and segregated into user requirements, system requirements and functional requirements. The requirements are collected using a number of practices as given -

- studying the existing or obsolete system and software,
- conducting interviews of users and developers,
- referring to the database or
- collecting answers from the questionnaires.

Feasibility Study

After requirement gathering, the team comes up with a rough plan of software process. At this step the team analyses if a software can be made to fulfil all requirements of the user and if there is any possibility of software being no more useful. It is found out, if the project is financially, practically and technologically feasible for the organization to take up. There are many algorithms available, which help the developers to conclude the feasibility of a software project.

System Analysis

At this step the developers decide a roadmap of their plan and try to bring up the best software model suitable for the project. System analysis includes Understanding of software product limitations, learning system related problems or changes to be done in existing systems beforehand, identifying and addressing the impact of project on organization and personnel etc. The project team analyses the scope of the project and plans the schedule and resources accordingly.

Software Design

Next step is to bring down whole knowledge of requirements and analysis on the desk and design the software product. The inputs from users and information gathered in requirement gathering phase are the inputs of this step. The output of this step comes in the form of two designs; logical design

and physical design. Engineers produce meta-data and data dictionaries, logical diagrams, data-flow diagrams and in some cases pseudo codes.

Coding

This step is also known as programming phase. The implementation of software design starts in terms of writing program code in the suitable programming language and developing error-free executable programs efficiently.

Testing

An estimate says that 50% of whole software development process should be tested. Errors may ruin the software from critical level to its own removal. Software testing is done while coding by the developers and thorough testing is conducted by testing experts at various levels of code such as module testing, program testing, product testing, in-house testing and testing the product at user's end. Early discovery of errors and their remedy is the key to reliable software.

Integration

Software may need to be integrated with the libraries, databases and other program(s). This stage of SDLC is involved in the integration of software with outer world entities.

Implementation

This means installing the software on user machines. At times, software needs post-installation configurations at user end. Software is tested for portability and adaptability and integration related issues are solved during implementation.

Operation and Maintenance

This phase confirms the software operation in terms of more efficiency and less errors. If required, the users are trained on, or aided with the documentation on how to operate the software and how to keep the software operational. The software is maintained timely by updating the code according to the changes taking place in user end environment or technology. This phase may face challenges from hidden bugs and real-world unidentified problems.

Disposition

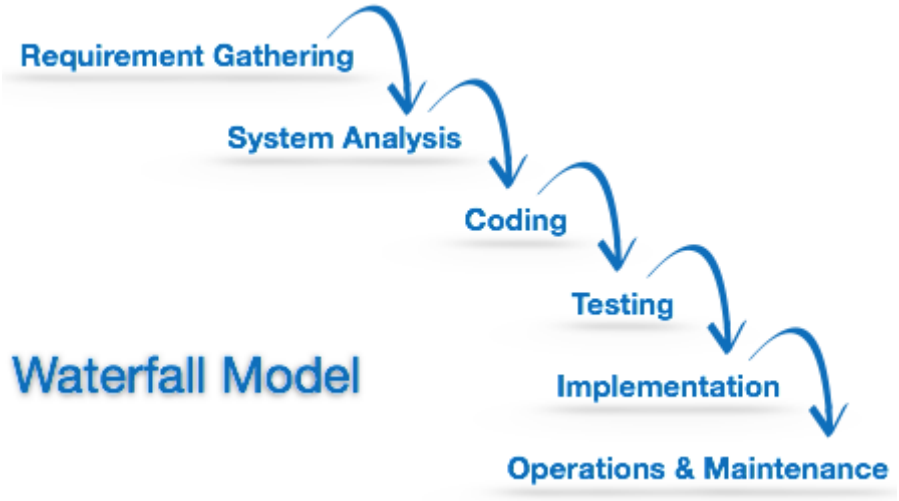
As time elapses, the software may decline on the performance front. It may go completely obsolete or may need intense upgradation. Hence a pressing need to eliminate a major portion of the system arises. This phase includes archiving data and required software components, closing down the system, planning disposition activity and terminating system at appropriate end-of-system time.

Software Development Paradigm

The software development paradigm helps developer to select a strategy to develop the software. A software development paradigm has its own set of tools, methods and procedures, which are expressed clearly and defines software development life cycle. A few of software development paradigms or process models are defined as follows:

Waterfall Model

Waterfall model is the simplest model of software development paradigm. It says the all the phases of SDLC will function one after another in linear manner. That is, when the first phase is finished then only the second phase will start and so on.

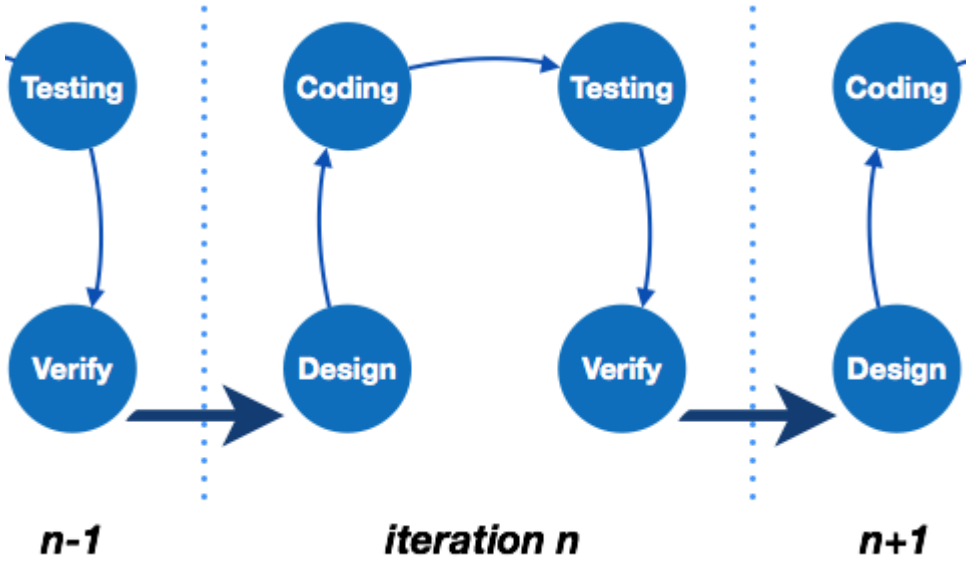


This model assumes that everything is carried out and taken place perfectly as planned in the previous stage and there is no need to think about the past issues that may arise in the next phase. This model does not work smoothly if there are some issues left at the previous step. The sequential nature of model does not allow us go back and undo or redo our actions.

This model is best suited when developers already have designed and developed similar software in the past and are aware of all its domains.

Iterative Model

This model leads the software development process in iterations. It projects the process of development in cyclic manner repeating every step after every cycle of SDLC process.

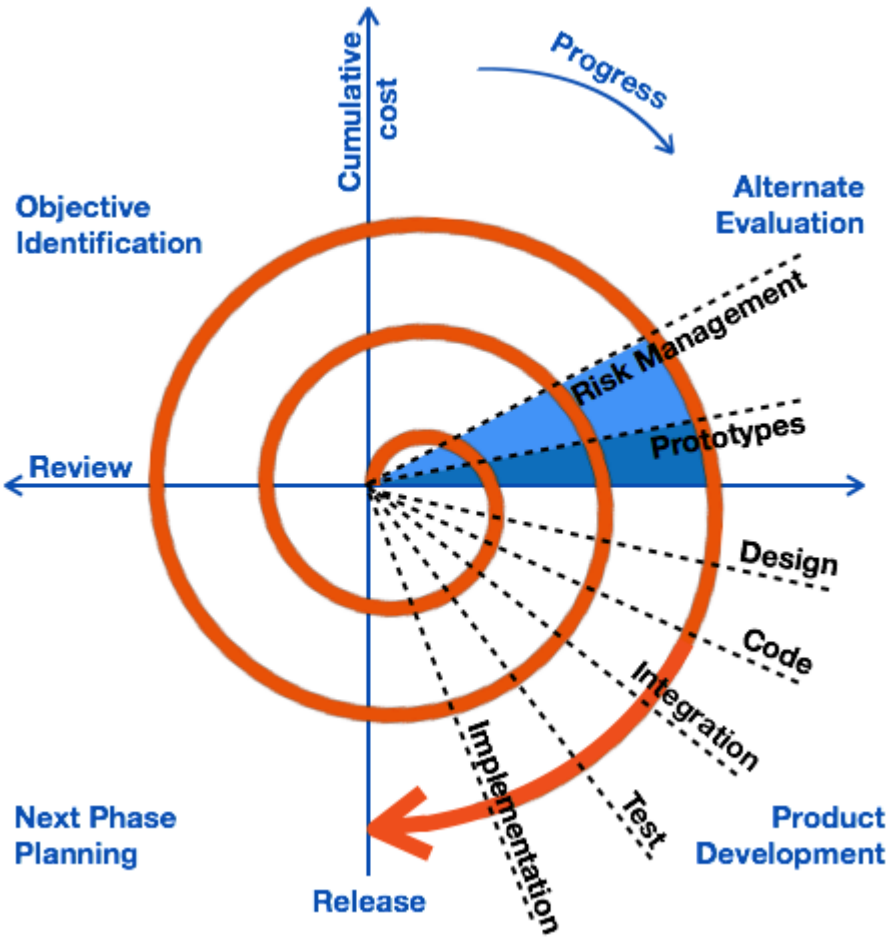


The software is first developed on very small scale and all the steps are followed which are taken into consideration. Then, on every next iteration, more features and modules are designed, coded, tested and added to the software. Every cycle produces a software, which is complete in itself and has more features and capabilities than that of the previous one.

After each iteration, the management team can do work on risk management and prepare for the next iteration. Because a cycle includes small portion of whole software process, it is easier to manage the development process but it consumes more resources.

Spiral Model

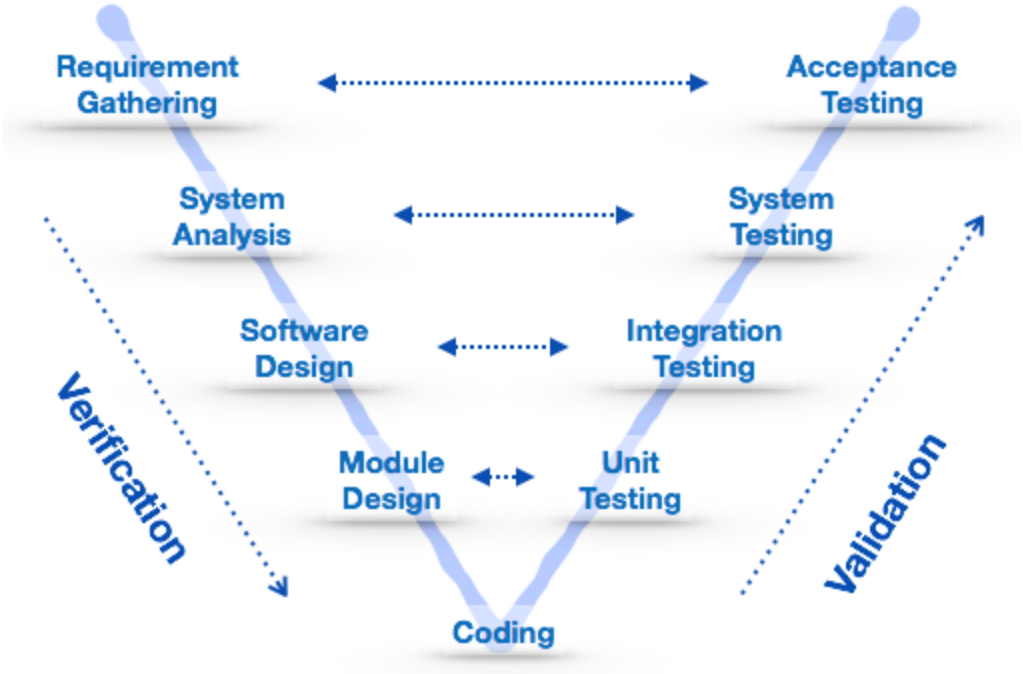
Spiral model is a combination of both, iterative model and one of the SDLC model. It can be seen as if you choose one SDLC model and combine it with cyclic process (iterative model).



This model considers risk, which often goes un-noticed by most other models. The model starts with determining objectives and constraints of the software at the start of one iteration. Next phase is of prototyping the software. This includes risk analysis. Then one standard SDLC model is used to build the software. In the fourth phase of the plan of next iteration is prepared.

V – model

The major drawback of waterfall model is we move to the next stage only when the previous one is finished and there was no chance to go back if something is found wrong in later stages. V-Model provides means of testing of software at each stage in reverse manner.



At every stage, test plans and test cases are created to verify and validate the product according to the requirement of that stage. For example, in requirement gathering stage the test team prepares all the test cases in correspondence to the requirements. Later, when the product is developed and is ready for testing, test cases of this stage verify the software against its validity towards requirements at this stage.

This makes both verification and validation go in parallel. This model is also known as verification and validation model.

Big Bang Model

This model is the simplest model in its form. It requires little planning, lots of programming and lots of funds. This model is conceptualized around the big bang of universe. As scientists say that after big bang lots of galaxies, planets and stars evolved just as an event. Likewise, if we put together lots of programming and funds, you may achieve the best software product.

For this model, very small amount of planning is required. It does not follow any process, or at times the customer is not sure about the requirements and future needs. So the input requirements are arbitrary.

This model is not suitable for large software projects but good one for learning and experimenting.

Data Flow Diagram(DFD)

Data flow diagram is graphical representation of flow of data in an information system. It is capable of depicting incoming data flow, outgoing data flow and stored data. The DFD does not mention anything about how data flows through the system.

There is a prominent difference between DFD and Flowchart. The flowchart depicts flow of control in program modules. DFDs depict flow of data in the system at various levels. DFD does not contain any control or branch elements.

Types of DFD

Data Flow Diagrams are either Logical or Physical.

- **Logical DFD** - This type of DFD concentrates on the system process, and flow of data in the system. For example, in a Banking software system, how data is moved between different entities.
- **Physical DFD** - This type of DFD shows how the data flow is actually implemented in the system. It is more specific and close to the implementation.

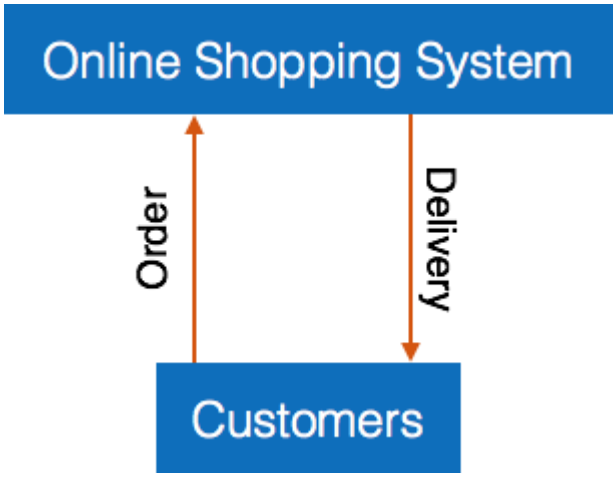
DFD Components

DFD can represent Source, destination, storage and flow of data using the following set of components -

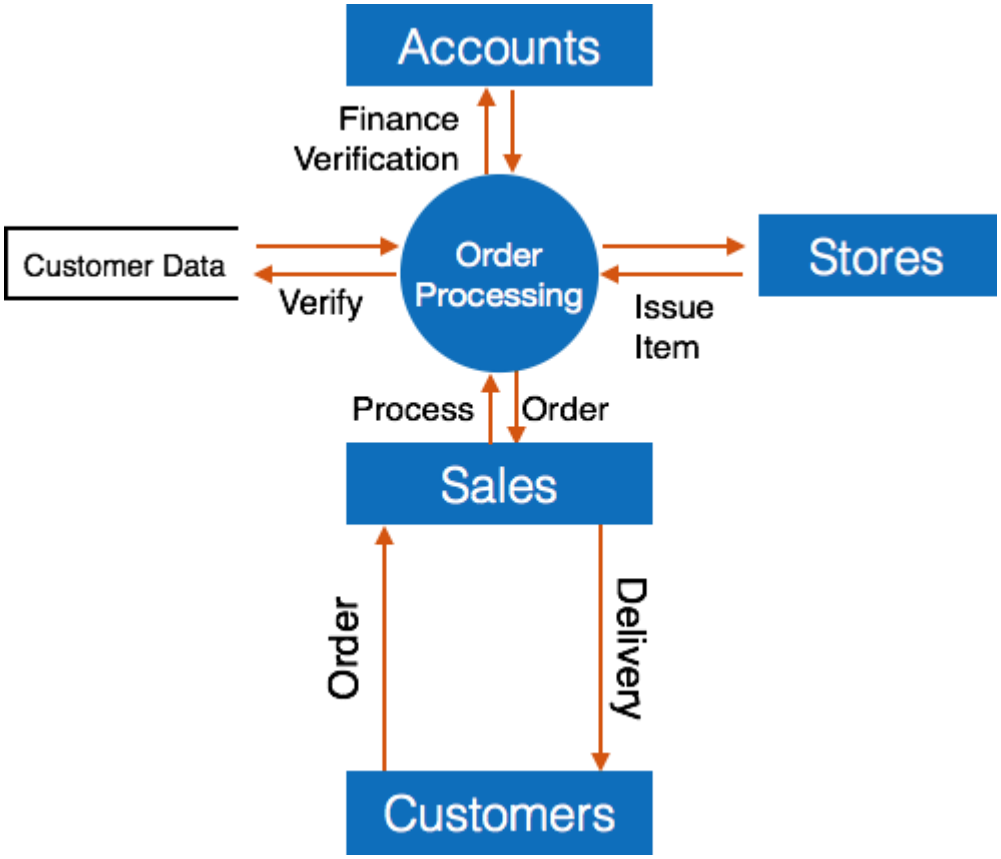
- **Entities** - Entities are source and destination of information data. Entities are represented by a rectangle with their respective names.
- **Process** - Activities and action taken on the data are represented by Circle or Round-edged rectangles.
- **Data Storage** - There are two variants of data storage - it can either be represented as a rectangle with absence of both smaller sides or as an open-sided rectangle with only one side missing.
- **Data Flow** - Movement of data is shown by pointed arrows. Data movement is shown from the base of arrow as its source towards head of the arrow as destination.

Levels of DFD

- **Level 0** - Highest abstraction level DFD is known as Level 0 DFD, which depicts the entire information system as one diagram concealing all the underlying details. Level 0 DFDs are also known as context level DFDs.



- **Level 1** - The Level 0 DFD is broken down into more specific, Level 1 DFD. Level 1 DFD depicts basic modules in the system and flow of data among various modules. Level 1 DFD also mentions basic processes and sources of information.



- **Level 2** - At this level, DFD shows how data flows inside the modules mentioned in Level 1.

  Higher level DFDs can be transformed into more specific lower level DFDs with deeper level of understanding unless the desired level of specification is achieved.

Q#1. What is Agile Testing?

Ans. Agile Testing is a practice that a QA follows in a dynamic environment where testing requirements keep changing according to the customer needs. It is done parallel to the development activity where the testing team receives frequent small codes from the development team for testing.

Q#2. What is the difference between burn-up and burn-down chart?

Ans. Burn-up and burn-down charts are used to keep track of the progress of the project.

Burn-up charts represent how much work has been completed in any project whereas Burn-down chart represents the remaining work in a project.

Q#. What are the different types of Agile Methodologies?

There are several types of agile development methodology. Scrum is one of the most popular and widely used agile methods. Other types of agile development methodology are; development like Crystal Methodology, DSDM (Dynamic Software Development Method), Feature-driven development(FDD), Lean software development and Extreme Programming(XP).

Q#. Difference between extreme programming and scrum?

Scrum teams usually have to work in iterations which are known as **sprints** which generally last up to two weeks to one month long while XP team works in the iteration that lasts for one or two weeks.
XP teams are more flexible as they can change their iterations while Scrum teams do not allow any change in their iterations.
The product owner prioritizes the product backlog but the team decides the sequence in which they will develop the backlog items in scrum methodology. Whereas XP team works in strict priority order, features developed are prioritized by the customers.

Q#3. Define the roles in Scrum?
Ans. There are mainly three roles that a Scrum team have:

1. Project Owner – who has the responsibility of managing the product backlog. Works with end users and customers and provide proper requirement to the team to build the proper product.
2. Scrum Master – who works with scrum team to make sure each sprint gets complete on time. Scrum master ensure proper work flow to the team.
3. Scrum Team – Each member in the team should be self-organized, dedicated and responsible for high quality of the work.

Q#4. What is Product backlog & Sprint Backlog?

Ans. Product backlog is maintained by the project owner which contains every feature and requirement of the product.
Sprint backlog can be treated as subset of product backlog which contains features and requirements related to that particular sprint only.

Q#5. Explain Velocity in Agile?
Ans. Velocity is a metric that is calculated by addition of all efforts estimates associated with user stories completed in a iteration. It predicts how much work Agile can complete in a sprint and how much time will require to complete a project.

Q#6. Explain the difference between traditional Waterfall model and Agile testing?
Ans. Agile testing is done parallel to the development activity whereas in traditional waterfall model testing is done at the end of the development.
As done in parallel, agile testing is done on small features whereas in waterfall model testing is done on whole application.

Q#7. Explain Pair Programming and its benefits?
Ans. Pair programming is a technique in which two programmer works as team in which one programmer writes code and other one reviews that code. They both can switch their roles.
Benefits:

1. Improved code quality: As second partner reviews the code simultaneously, it reduces the chances of mistake.
2. Knowledge transfer is easy: One experience partner can teach other partner about the techniques and codes.

Q#8. What is re-factoring?
Ans. Modification of the code without changing its functionality to improve the performance is called re-factoring.

Q#9. Explain the Iterative and Incremental Development in Agile?

Ans. Iterative Development: Software is developed and delivered to customer and based on the feedback again developed in cycles or release and sprints. Say in Release 1 software is developed in 5 sprints and delivered to customer. Now customer wants some changes, then development team plan for 2nd release which can be completed in some sprints and so on.

Incremental Development: Software is development in parts or increments. In each increment a portion of the complete requirement is delivered.