Module : 1 (SLDC)

Q:1 What is shoftware? What is software engineering?

Ans.

Definition:

Software is set of instuction, data and information for computer system. This is in contrast to hardware, from which the system is built and which actually performs the work.

At the lowest programming leve;, eecutable code consists of machine language instructions supported by an individual processor - typically a central processing unit (CPU) or a graphics processing unit (GPU). Machine language consists of groups of binary values signifying processor instructions that change the state of the computer fromits preceding state.

The majority of software is written in high-level programming languages. They are easier and more efficient for programmers because they are closer to natural languages than machine languages. High-level languages are translated into machine language using a compiler or an interpreter or a combination of the two Software may also be written in a low-level assembly language, which has a strong correspondence to the computer's machine language instructions and is translated into machine language using an assembler.

Software engineering:

A software engineer is a person who applies the principles of software engineering to design, develop, maintain, test, and evaluate computer software. The term programmer is sometimes used as a synonym, but may also lack connotations of engineering education or skills.

Engineering techniques are used to inform the software development process which involves the definition, implementation, assessment, measurement, management, change, and improvement of the software life cycle process itself. It heavily uses software configuration management which is about systematically controlling changes to the configuration, and maintaining the integrity and traceability of the configuration and code throughout the system life cycle. Modern process use software versioning.

Q:2 Explain types of software.

Ans.

Types of softwre:

1. System software :These software programs are designed to run a computer's application progras and hardware. System software coordinates the activities and functions of the hardware and software. In addition, it controls the operations of the computer hardware and provides

an environment or platform for all the other types of software to work in. The OS is the best example of system software; it manages all the other computer programs. Other examples of system software include the firmware, computer language tranlators and system utilities.

2. Application software :The most common type of software, application software is a computer software package that performsd a specific function for a user, or in some cases, for another application. An application can be self-contained, or it can be a group of programs that run the application for the user. Example of modern applications include office suites, grafic software, databases and databases management programs, web browsers, word processors, software development tools, image editors and communication platforms.

3. Programming software :Computer programmers use programming software to write code. Programming software and programming tools enable developers to develop, write, test and debug other software programs. Examples of programming software include assemblers, compilers, debuggers and interpreters.

4. Driver software :Also known as device drivers, this software is often considered a type of system software. Device drivers control the devices and peripherals connected to a computer, enabling them to perform their specific tasks. Every device that is connected to a computer needs at least one device driver to function. Examples include software that comes with any nonstandard hardware, including special game controllers, as well as the software that enables standard hardware, such as USB storage devices, keyboards, headphones and printers.

5. Middleware :The term middleware describes software that mediates between application and system software or between two different kinds of application software. For example, middleware enables Microsoft Windown to talk to Excel and Word. It is also used to send a remote work request from an application in a computer that has one kind of OS, to an application in a computer with a different OS. It also enables newer applications to work legacy ones.

Q: 3 What is SDLC? Explain each phase of SDLC.

Ans.

SDLC:
The Software Development Life Cycle (SDLC) is a structure process that enables the production of high-quality, low-cost software, in the shortest possible production time. The goal of the SDLC is to produce superior software that meets

and exceeds all customer expectations and demands. The SDLC defines and outlines a detailed plan with stages, or phases, that each encompass their own process and deliverables. Adherence to the SDLC enhances development speed and minimizes project risks and costs associated with alternative methods of production.

Phases of SDLC :

Planning phase(Requirement gathering):

The planning phase encompasses all aspects of project and product management. This typically includes resource allocation, capacity planning, project scheduling, cost estimation, and providing.

During the planning phase, the development team collects input from stakeholders involved in the project; customers, sales, internal and external experts, and developers. This input is synthesized into a detailed definition of the requirements for creating the desired software. The team also determines what resources are required to satisfy the project requirements, and then infers the associated cost.

Expectations are clearly defined during this stage as well; the team determines not only what is desired in the software, but also what is NOT. The tangible deliverables produced from this phase includes project plans, estimated costs, projected scedules, and procurement needs.

Analysis phase:

* The analysis phase defines the requirements of the system, independent of how these Requirement will be accomplished.
* This phase defines the problem that the customer is trying to solve.
* The deliverable result at the end of this phase is a requirement document.
* Ideally, this document states in a clear and precise fashion what is to be built.
* This analysis represents the -what" phase.
* The requirement documentaries to capture the requirement phase and mapsthe requirements from the customer's perspective by defining goals.
* This phase starts with the requirement document delivered by the requirement phase and mapsthe requirements into architecture.
* The architecture defines the components, their interfaces and behaviors.
* The deliverable design document is the architecture.
* The phase represent the -how" phase.
* Details on computer programming languages and environments, machines, packages, application architecture, distributed architecture layering, memory size, platform, algorithms, data structures, global type definitions, interfaces, and many other engineering details are established.

* The design may include the usage of existing components.

Design phase:

* Design Architecture Document
* Implementation Plan
* Critical priority Analysis
* Performance Analysis
* Test plan
* The Design team can now expand upon the information established in the requirement Document.
* The requirement document must guide this decision process.
* Analyzing the trade-offs of necessary complexity ballows for many things to remain simple which, in turn, will eventually lead to a higher quality product. The architecture team also converts the typical scenarios into a test plan.

Implementation phase:

* In the implementation phase, the team builds the components either from scratch or by composition.
* Given the architecture document from the design phase and the requirement document from the analysis phase, the team should build exactly what has been requested, though there is still room for innovation and flexibility.
* For example, a component may be narrowly desiged for this practicular system, or the component may be made more general to satisfy a reusability guideline.
* Implementation - code
* Critical Error Removal
* The implementation phase deals with issues of quality, performance, baselines, libraries, and debugging.

The end deliverable is the product itself. There are already many establised techniques associated with implementation.

Testing phase:

* Simply stated, qualityis very important. Many companies have not learned thatn quality is important and deliver more claimed functionality but at a lower quality level.
* It is much easier to explain to a customer why there is a missing feature than to explain to a customer why the product lacks quality.
* A customer satisfied with the quality of a product will remain loyal and wait for new Functionality in the next version.
* Quality is a distinguishing attribute of a system indicating the degree of excellence.
* Regression Testing
* Internal Testing
* Unit Testing
* Application Testing
* Stress Testing

* The testing phase is a separate phase which is performed by a different team after the implementation is completed.

Maintenance Phase:

* Software maintenance is one of the activities in software engineering, and is the process of enhancing and optimizing deployed software ( software release), as well as fixing defects.
* Software maintenance is also one of the phases in the system Devlopment Life Cycle (SDLC), as it applies to software development. The maintenance phase is the phase which comes after development of the software into the field.
* The developing organization or team will have some mechanism to document and track defects and deficiencies.
* Configuration and version management
* reengineering (redesigning and refactoring)
* updating all analysis, design and user documentation
* Repeatable, automated tests enable evolution and refactoring

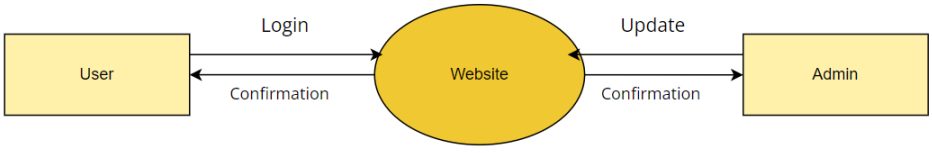
Maintenance is the process of changing a system after it has been deployed.
Corrective maintenance: identifying and repairing defects
Adaptive maintenance: adapting the existing solution to the new platforms.
Perfective Maintenance: implimanting the new requirements

Q:4 What is DFD? Create a DFD diagram on Flipkart

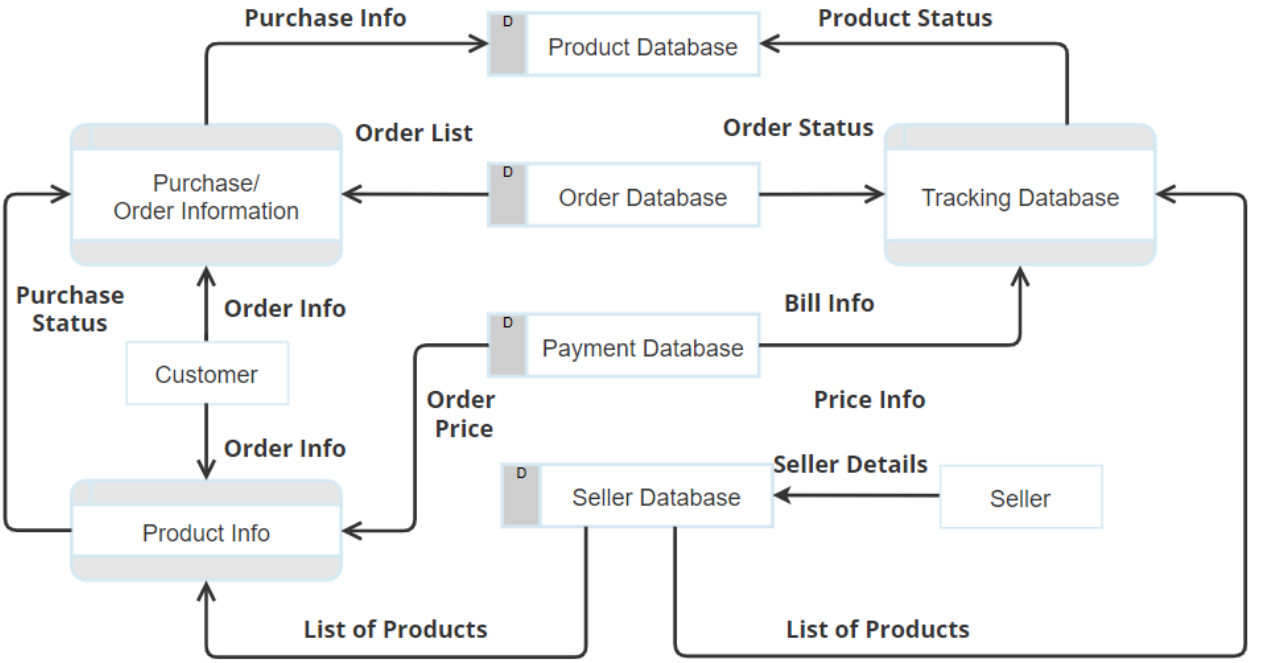
Ans.
A Data Flow Diagram(DFD) is a traditional way to visualize the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically.

It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination.While they work well for data flow software and system, they are less applicable nowdays to visualizing interactive, real-time or database-oriented software or system.

* 0 level DFD




* 1 or 2 level DFD




Q:5 What is Flow chart? Creat a flowchart to make addition of two numbers.

Ans.
A flowchart is a type of diagram that represents a workflow or process. The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows. Flowchart are used in analyzing, designing, documenting or managing a process or program in various fields.

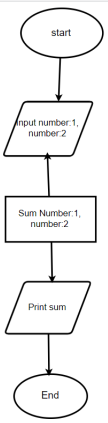
Flowcharts typically use the following main symbols:
* A process step, usually called an activity, is denoted as a rectangular box.
* A decision is usually denoted as a diamond.

Types of flowchart:

* Document flowchart, showing controlls over a document flow through a system
* Data flowcharts, showing controls over a data-flow in a system
* System flowcharts, showing controls at a physical or resource level
* Program flowchart, showing the controls in a program within a system

Notice that every type of flowchart focuses on some kind of control, rather than on the particular flow itself.

Flow chart of two sum:



Q:6 What is Use case Diagram? Create a use-case on bill payment on paytm.

Ans.

A use case diagaram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses. The actors are often shown as stick figures.

While a use case itself might drill into a lot of detail about evary possibility, a use-case diagram can help provide a higher-level view of the system. It has been said before that "Use case diagrams are the blueprints for your system".