Assignment 3

Student Name: Sahil Kaundal UID: 21BCS8197

Branch: CSE (Lateral Entry) Section/Group:20BCS-807\_B Semester: 4th Date of Performance: 05/05/2022

Subject Name: SE Subject Code: 20CSP-254

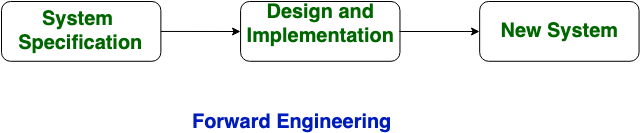
Q1.

**Differentiate between reverse engineering and forward engineering by considering example of any software.**

Answer:

Forward Engineering

* Forward engineering is a process of obtaining desired software from the specifications in hand which was brought down by means of reverse engineering. It assumes that there was some software engineering already done in the past.
* Forward engineering is the same as the software engineering process with only one difference it is carried out always after reverse engineering.
* The forward engineering process applies software engineering principles, concepts, and methods to recreate an existing application. In most cases, forward engineering does not simply create a modern equivalent of an older program.
* Rather, new user and technology requirements are integrated into the reengineering effort.
* The redeveloped program extends the capabilities of the older application.
* The example of forward engineering is the construction of electronic kit, construction of DC MOTOR, etc.
* Example of Forward Engineering could be any newly developing application or system that is started or about to start based on given requirements.



Reverse Engineering

* Reverse engineering can extract design information from source code, but the abstraction level, the completeness of the documentation, the degree to which tools and a human analyst work together.
* The directionality of the process is highly variable.
* The *abstraction*level of a reverse engineering process and the tools used to effect it refers to the sophistication of the design information that can be extracted from source code.
* Ideally, the abstraction level should be as high as possible.
* That is, the reverse engineering process should be capable of deriving procedural design representations (a low-level abstraction), program and data structure information (a somewhat higher level of abstraction), object models, data and/or control flow models (a relatively high level of abstraction), and entity-relationship models (a high level of abstraction).
* As the abstraction level increases, you are provided with information that will allow an easier understanding of the program
* The *completeness*of a reverse engineering process refers to the level of detail that is provided at an abstraction level. In most cases, the completeness decreases as the abstraction level increases.
* *Interactivity*refers to the degree to which the human is “integrated” with automated tools to create an effective reverse engineering process.
* In most cases, as the abstraction level increases, interactivity must increase or completeness will suffer.
* The *directionality*of the reverse engineering process is one-way, all information extracted from the source code is provided to the software engineer who can then use it during any maintenance activity.
* An example of backward engineering is research on Instruments etc.
* Reverse engineering includes mainly research and RND work which could get implemented in the already developed application or system for providing more efficient performance with more functionality.



Q2.

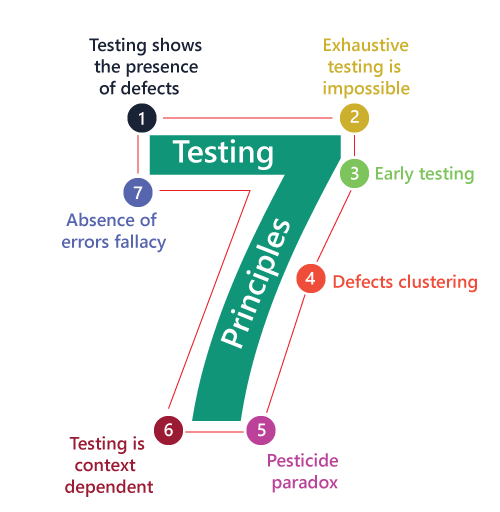
What are the testing principles the software engineer must apply while performing the software testing?

Answer:

Software testing is a procedure of implementing software or the application to identify the defects or bugs. For testing an application or software, we need to follow some principles to make our product defects free, and that also helps the test engineers to test the software with their effort and time.

The seven essential principles of software testing.

* Testing shows the presence of defects
* Exhaustive Testing is not possible
* Early Testing
* Defect Clustering
* Pesticide Paradox
* Testing is context-dependent
* Absence of errors fallacy



* Testing shows the presence of defects

The test engineer will test the application to make sure that the application is bug or defects free. While doing testing, we can only identify that the application or software has any errors. The primary purpose of doing testing is to identify the numbers of unknown bugs with the help of various methods and testing techniques because the entire test should be traceable to the customer requirement, which means that to find any defects that might cause the product failure to meet the client's needs.

By doing testing on any application, we can decrease the number of bugs, which does not mean that the application is defect-free because sometimes the software seems to be bug-free while performing multiple types of testing on it. But at the time of deployment in the production server, if the end-user encounters those bugs which are not found in the testing process.

* Exhaustive Testing is not possible

Sometimes it seems to be very hard to test all the modules and their features with effective and non- effective combinations of the inputs data throughout the actual testing process.

Hence, instead of performing the exhaustive testing as it takes boundless determinations and most of the hard work is unsuccessful. So we can complete this type of variations according to the importance of the modules because the product timelines will not permit us to perform such type of testing scenarios.

* Early Testing

Here early testing means that all the testing activities should start in the early stages of the software development life cycle's requirement analysis stage to identify the defects because if we find the bugs at an early stage, it will be fixed in the initial stage itself, which may cost us very less as compared to those which are identified in the future phase of the testing process.

To perform testing, we will require the requirement specification documents; therefore, if the requirements are defined incorrectly, then it can be fixed directly rather than fixing them in another stage, which could be the development phase.

* Defect clustering

The defect clustering defined that throughout the testing process, we can detect the numbers of bugs which are correlated to a small number of modules. We have various reasons for this, such as the modules could be complicated; the coding part may be complex, and so on.

These types of software or the application will follow the Pareto Principle, which states that we can identify that approx. Eighty percent of the complication is present in 20 percent of the modules. With the help of this, we can find the uncertain modules, but this method has its difficulties if the same tests are performing regularly, hence the same test will not able to identify the new defects.

* Pesticide paradox

This principle defined that if we are executing the same set of test cases again and again over a particular time, then these kinds of the test will not be able to find the new bugs in the software or the application. To get over these pesticide paradoxes, it is very significant to review all the test cases frequently. And the new and different tests are necessary to be written for the implementation of multiple parts of the application or the software, which helps us to find more bugs.

* Testing is context-dependent

Testing is a context-dependent principle states that we have multiple fields such as e-commerce websites, commercial websites, and so on are available in the market. There is a definite way to test the commercial site as well as the e-commerce websites because every application has its own needs, features, and functionality. To check this type of application, we will take the help of various kinds of testing, different technique, approaches, and multiple methods. Therefore, the testing depends on the context of the application.

* Absence of errors fallacy

Once the application is completely tested and there are no bugs identified before the release, so we can say that the application is 99 percent bug-free. But there is the chance when the application is tested beside the incorrect requirements, identified the flaws, and fixed them on a given period would not help as testing is done on the wrong specification, which does not apply to the client's requirements. The absence of error fallacy means identifying and fixing the bugs would not help if the application is impractical and not able to accomplish the client's requirements and needs.