Name : Aman ullah
Id: 14303

Subject: OOSE

Programe: BS(SE)05

Question #1

In Software Engineering, there is not a single answer to the question "What should be done first, Coding or Modeling?". Elaborate different scenarios in which all the answers to this questions are justified.

Answer

Generally model is created before programming and it is highly recommended first model and then do programming.

It all depends….

- **Forward Engineering**
  - Creation of code from a model
  - Start with modeling
  - Greenfield projects
- **Reverse Engineering**
  - Creation of a model from existing code

- Interface or reengineering projects

Roundtrip Engineering

- Move constantly between forward and reverse engineering
- Reengineering projects
- Useful when requirements, technology and schedule are changing frequently.

Modeling Principles

In software engineering work, two classes of models can be created:

- Requirements models (also called analysis models) represent the customer requirements by depicting the software in three different domains: the information domain, the functional domain, and the behavioral domain.
- Design models represent characteristics of the software that help practitioners to construct it effectively: the architecture, the user interface, and component-level detail.

Design Modeling Principles

- Principle #1. Design should be traceable to the requirements model. The design model translates requirement model into an architecture.
- Principle #2. Always consider the architecture of the system to be built. Skeleton of the system
- Principle #3. Design of data is as important as design of processing functions. The way the data objects are realized.

- Principle #4: Interfaces (internal and external) must be designed with care. This makes integration and testing easier.
- Principle #5. User interface design should be tuned to the needs of the end-user. However, in every case, it should stress ease of use.
- Principle #6. Component-level design should be functionally independent. Focus on one and only one function.
- Principle #7. Components should be loosely coupled to one another and to the external environment. To reduce error propagation and increase maintainability.
- Principle #8. Design representations (models) should be easily understandable. For coders, testers, etc.
- Principle #9. The design should be developed iteratively. With each iteration, the designer should strive for greater simplicity. Like all activities for refinement.

Coding Principles

- As you begin writing code, be sure you: Constrain your algorithms by following structured programming [Boh00] practice.
- Consider the use of pair programming
- Select data structures that will meet the needs of the design.
- Understand the software architecture and create interfaces that are consistent with it.
- Keep conditional logic as simple as possible.
- Create nested loops in a way that makes them easily testable.
- Select meaningful variable names and follow other local coding standards.

- Write code that is self-documenting.
- Create a visual layout (e.g., indentation and blank lines) that aids understanding.

Question #2

When carrying out Testing of a Software, a number of techniques are used. Why are they so many in number? Name a few popular Testing Techniques in Software Engineering and state the importance of each one.

Answer

Software engineering techniques are so many in number because there is still a list of more than 100+ types of testing, but all testing types are not used in all types of projects.

Software testing is very important because of the following reasons:

1. Software testing is really required to point out the defects and errors that were made during the development phases.
    - Example: Programmers may make a mistake during the implementation of the software. There could be many reasons for this like lack of experience of the programmer, lack of knowledge of the programming language, insufficient experience in the domain, incorrect implementation of the algorithm due to complex logic or simply human error.

2. It’s essential since it makes sure that the customer finds the organization reliable and their satisfaction in the application is maintained.
3. Testing is necessary in order to provide the facilities to the customers like the delivery of high quality product or software application which requires lower maintenance cost and hence results into more accurate, consistent and reliable results

Name of Different Software engineering techniques:

Functional Testing types include:

- Unit Testing
- Integration Testing
- System Testing
- Sanity Testing
- Smoke Testing
- Interface Testing
- Regression Testing
- Beta/Acceptance Testing

Non-functional Testing types include:

- Performance Testing
- Load Testing
- Stress Testing
- Volume Testing
- Security Testing
- Compatibility Testing
- Install Testing
- Recovery Testing
- Reliability Testing
- Usability Testing
- Compliance Testing
- Localization Testing

Integration Testing

Testing of all integrated modules to verify the combined functionality after integration is termed as Integration Testing.

Recovery Testing

Recovery Testing determines if the system is able to continue the operation after a disaster. Assume that application is receiving data through the network cable and suddenly that network cable has been unplugged.

Regression Testing

Testing an application as a whole for the modification in any module or functionality is termed as Regression Testing. It is difficult to cover all the system in Regression Testing, so typically Automation Testing Tools are used for these types of testing.

Smoke Testing

it checks that no show stopper defect exists in the build which will prevent the testing team to test the application in detail.

Acceptance Testing

An Acceptance Test is performed by the client and verifies whether the end to end the flow of the system is as per the business requirements or not and if it is as per the needs of the end-user. Client accepts the software only when all the features and functionalities work as expected.

Sanity Testing

Sanity Testing is done to determine if a new software version is performing well enough to accept it for a major testing effort or not. If an application is crashing for the initial use then the system is not stable enough for further testing. Hence a build or an application is assigned to fix it.

Security Testing

It is done to check how the software or application or website is secure from internal and external threats. This testing includes how much software is secure from the malicious program, viruses and how secure and strong the authorization and authentication processes are.

System Testing

Under System Testing technique, the entire system is tested as per the requirements. It is a Black-box type Testing that is based on overall requirement specifications and covers all the combined parts of a system.

Unit Testing

Testing of an individual software component or module is termed as Unit Testing. It is typically done by the programmer and not by testers, as it requires detailed knowledge of the internal program design and code. It may also require developing test driver modules or test harnesses.

Comparison Testing

Comparison of a product's strength and weaknesses with its previous versions or other similar products is termed as Comparison Testing.

Load Testing

It help to find the maximum capacity of the system under specific load and any issues that cause software performance degradation. Load testing is performed using tools like JMeter, LoadRunner, WebLoad, Silk performer, etc.

Stress Testing

This testing is done when a system is stressed beyond its specifications in order to check how and when it fails. This is performed under heavy load like putting large number beyond storage capacity,

complex database queries, continuous input to the system or database load.

THE END