Name: Imran Khan

I-D: 14281

Subject: OOSE

Q1.

Answer:

Developers usually distinguish between modeling and coding. Models are used to design systems, better understand them, define required functionality, and create documentation. The code is then written to implement the designs. Debugging, testing and maintenance are also done at the code level.

Most software developers, however, also create models. The concepts of pure coding are, in most cases, far removed from the requirements and the real domain of the problem. Models are used to increase the level of abstraction and to hide implementation details. In the traditional development process, the models, however, are kept completely separate from the code because there is no automatic change in the code from these models. Instead, developers read the models and interpret them while coding the application and developing workable software.

During implementation, models are not further updated and are often deleted after coding. This is simply because the cost of updating the models far outweighs the benefits of these models. Code and model The

cost of maintaining the same information in two places is high because it is manual, cumbersome and erroneous.

The following image shows the alignment code and model

During implementation, models are not further updated and are often deleted after coding. This is simply because the cost of updating the models far outweighs the benefits of these models. Code and model The cost of maintaining the same information in two places is high because it is manual, cumbersome and erroneous.

Models can also be used in reverse engineering: trying to understand software once it is designed and built. Although it is understandable to create documentation based on models after this, the concept of code can also be useful for trying to understand the program or to use it as elements in models while importing libraries or other constructs from the code. However, such models are not commonly used to implement, debug, or test software because we have the code.

One way to solve this problem is to use only one source, usually code, and show some of it in models. The classic example is that only a

fraction of the power of the class diagram is used. The part where the class diagram is exactly on the map of the class code.

In model-driven development, we use the model as a key model in the development process: we have source models instead of source code. Throughout this book, we argue that this approach should be applied whenever possible because it increases the level of abstraction and hides complexity. Truly model-driven development uses changes in the same way that a pure coding approach uses developers.

Once the models are ready, the target code can be generated and then compiled or interpreted for implementation. From a modeler's point of view, the generated code is complete and does not need to be modified from generation to generation. However, this means that "intelligence" is not only in the models but also in the code generator and infrastructure.

Question 2

Answer

Techniques:

The purpose of using a variety of testing methods in your development process is to ensure that your software can run successfully in multiple environments and across different platforms. They can usually be broken down into practical and passive testing. Functional testing involves examining an application against business requirements. It covers all types of tests developed to guarantee every part of the software piece,

which is expected to be done using cases provided by the design team or business analyst as expected. - These testing methods are usually performed in sequence and include:

Unit testing

Integration testing

System testing

cept acceptance test

Non-practical testing methods include all test types focused on the operational aspects of the software piece. These include:

formance performance testing

Security testing

ability test

ati compatibility check

Why are there numbers?

The key to releasing high quality software that your users can easily adopt is to build a robust testing framework that implements both practical and passive software testing procedures. The number is there.

Significance

Unit testing

Unit testing is the first level of testing and is often performed by the developers themselves. This is to ensure that the individual components of a piece of software at the code level are put into practice and work as they were designed. Developed in a test-based environment