Iqra National University Peshawar Pakistan

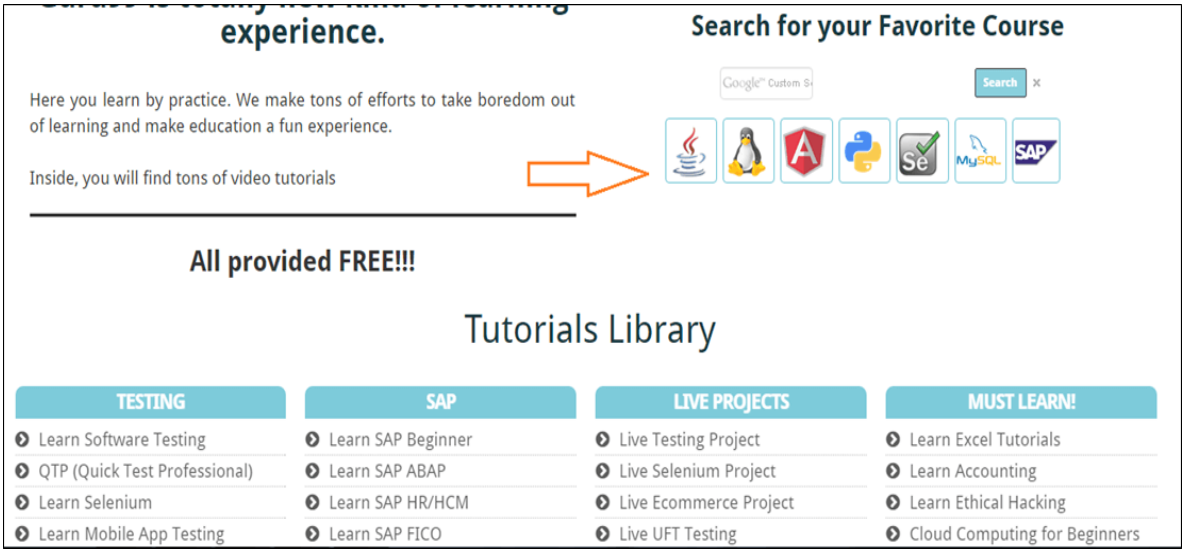
Department of Computer Science

Spring Semester, Mid-Assignment, April 2020

Subject:	**TSE**	Issue Date:	**20/April/2020**
Program:	**MS (CS)**	Submission Date:	**30/April/2020**
Teacher Name:	**Dr. Fazal-e-Malik**		
Student Name	**Rooh Ullah Jan**	ID	**6611**

Q.1 a) what is GUI testing?

GUI TESTING is a software testing type that checks the Graphical User Interface of the Application Under Test. GUI testing involves checking the screens with the controls like menus, buttons, icons, and all types of bars - toolbar, menu bar, dialog boxes, and windows, etc. The purpose of Graphical User Interface (GUI) Testing is to ensure UI functionality works as per the specification. A user does not see the source code. The interface is visible to the user. Especially the focus is on the design structure, images that they are working properly or not.



GUI Testing Tools are the tools used for verifying and validating the graphical user interface of the software application, for its cosmetic appearance, functional and non-functional properties of the application. The testing life cycle and defect life cycle are similar to any other type of testing methods applied on the software applications. Some of the widely used GUI Testing tools in the software industries are SWTBot, Selenium, Test Studio, TestComplete, TestPartner, QF-Test, RCP Testing Tool, Telerik, Tellurium, Watir, coded UI, cucumber, Smartbear LoadUI, QAliber, GTT, IcuTest, Test Anywhere, Test Studio, RIATest, SilkTest, CubicTest, Ranorex, CrossBrowserTesting, Abbot Java GUI Test Framework, AutoIt UI Testing, eggplant UI Automation Testing, FitNesse, etc.

b) Study open source Tools available to conduct GUI Testing and write notes on only two tools.

There are many tools use for GUI Testing. Two of them are given below.

1. **Selenium**

- Selenium is one of the most common and widely used Testing tools for functional and UI testing.
- It supports parallel Testing on various web browsers like Chrome, Mozilla Firefox, IE, Safari, etc.
- Test scripts for GUI Testing in Selenium can be written in various languages like Python, Java, C#, etc.
- It provides the special facility of record and play which is very helpful while performing the UI tests.
- Executing the test scripts in Selenium assures Synchronization. As the UI Testing is asynchronous but while using Selenium for Automation

Testing ensures that the execution is moved to the next page once the current page is loaded properly.

- It allows adding the verifications through assertions for which inbuilt functions are available in Selenium.

2. TestingWhiz

TestingWhiz is a test automation tool with the code-less scripting by Cygnet Infotech, a CMMi Level 3 IT solutions provider. TestingWhiz tool's Enterprise edition offers a complete package of various automated testing solutions like web testing, software testing, database testing, API testing, mobile app testing, regression test suite maintenance, optimization, and automation, and cross-browser testing.

TestingWhiz offers various important features like:

- Keyword-driven, data-driven testing, and distributed testing
- Browser Extension Testing
- Object Eye Internal Recorder
- SMTP Integration
- Integration with bug tracking tools like Jira, Mantis, TFS and FogBugz
- Integration with test management tools like HP Quality Center, Zephyr, TestRail, and Microsoft VSTS
- Centralized Object Repository
- Version Control System Integration
- Customized Recording Rule.

Q.2 a) what is usability Testing?

Usability testing is a technique used in user-centered interaction design to evaluate a product by **testing** it on users. This can be seen as an irreplaceable **usability** practice, since it gives direct input on how real users use the system.

Also it is refers to evaluating a product or service by testing it with representative users. Typically, during a test, participants will try to complete typical tasks while observers watch, listen and takes notes. The goal is to identify any usability problems, collect qualitative and quantitative data and determine the participant's satisfaction with the product.

Usability is a delicate thing. Designers have to find creative ways to combine what they think is best with the user's perception of what is best. That touches on everything in the product: from visuals like the color scheme, to the navigation and interactions. But how can you understand what another individual prefers? Well – you gotta ask them.

The idea is that you get a representative group of users and see what is best in their opinion. Granted, simply "asking" may be simplifying it. It does involve a whole process of research and planning before you even get to the actual testing.

A usability test is the act of putting a prototype of your product in front of a user and carrying out an interview. This usually goes with a set of activities or tasks that participants are asked to do with the prototype.

b) Study usability test tools available to conduct Usability Test and write notes on only two tools.

1. UserZoom

UserZoom is a huge name in the usability testing tools market. It's no surprise, really. Reaching the top of the game seems like a natural thing for a tool that offers 537% return on investment. Yes, there is an actual study carried out by Forrester Research with UserZoom that found some incredible statistics.

But when it comes to features and functionality: UserZoom offers the option to use your own participants in studies or making use of their advanced search engine – which has about 120 million possible participants. Reviews left by users of the platform praise it for the quick feedback and ability to carry out several studies in fast succession.

A cool aspect of this platform is that you can customize the plan so it includes only the add-ons you need for your project or company – such as the audience search engine.

- Who is it for: startups and small/medium businesses.

- Price: available upon request

2. UsabilityHub

UsabilityHub is one of the usability testing tools on this list that aims for simplicity.

One of their features, for example, is simplicity in itself: participants are shown a page for 5 seconds. Afterwards, they are asked what they remember about the page – all answers are listed. In the end, users have a word cloud with the things participants found most memorable. This helps designers adjust the design to make a lasting impression.

Among the features of UsabilityHub that we love is their advanced heat maps, that go beyond simply marking of click-density. This platform actually gives information and data along with the heat map, such as time-to-click.

Like other usability testing tools, it gives users the option to draw from the participant database or recruit their own participants. For users who bring in their own participants, there is no limit of how many people can take part in the study. If you prefer to rely on their participant database, you'll be able to list out specific demographics that interest you.

- Who is it for: startups, small and medium businesses
- Pricing: plans vary from 79$/month to 396$/month

Q.3 a) what is clean room software engineering?

Clean Room Software Engineering:

The cleanroom software engineering process is a software development process intended to produce software with a certifiable level of reliability also Cleanroom software engineering is a process for developing high-quality software with certified reliability. Originally developed by Harlan Mills, the "cleanroom" name was borrowed from the electronics industry, where clean rooms help prevent defect during fabrication. In that sense, cleanroom software engineering focuses on defect

prevention, rather than defect removal, and formal verification of a program's correctness. The Cleanroom Reference Model provides guidelines for defining development teams and project roles, most importantly, the distinction between testers and developers. Having testing and development in the same group is a conflict of interest, while splitting them into separate groups allows for natural competition to push the project toward higher quality results.

Cleanroom differs from other formal methods in that it doesn't require mathematically defined requirements—those stated in plain English are adequate. These requirements are divided into tagged statements for traceability. The process of tagging requirements in small verifiable statements allows for tracing and verification of each requirement throughout the process. Moreover, since attempts to document requirements are likely to have errors, inconsistencies, and omissions, Cleanroom refines many of these through the "Box Structure Development Method," a process that treats software as a set of communicating state machines that separate behavioral and implementation concerns.

To illustrate the Cleanroom development process, I'll present a GUI that eliminates individual Save and Quit features. The project requirements are split between an abstract GUI and data-editing requirements. Partitioning the tagged requirements along lines of "separation of concerns" is the first step toward discovery of natural divisions in the software architecture.The source code implementation of this GUI.

b) Study some case studies in cleanroom software engineering and write summaries of some case studies.

Study of some case study are given below.

CASE STUDY OVERVIEW:

Experimental Software Engineering is needed to enable transfer of research results from the universities to industrial use. It is mostly not possible to take research results and apply them in a large experiment directly. Therefore, the technology transfer must be carried out in a number of steps: minor case study within a university environment, an enlarged experiment in industry and finally into real projects as the normal method or technique to use.

This case study focuses on reporting some results from the first step in the dissemination process. A comparative study between two development methods is reported. The two methods are: object-based Cleanroom and Object-Oriented Software Engineering. The study shows that the object-based Cleanroom produces higher quality software with approximately the same effort as the more widely spread method. The results are, due to the small study, not statistical significant, but they are promising for the future and they ought to encourage industry to perform a larger experiment in an industrial setting. Both quantitative and qualitative results are presented.

Introduction:

The transfer of new technology from research to practical application is difficult. New ideas and methods cannot be incorporated into existing software processes due to lack of evidence in their ability to improve cost and quality. Therefore, experimental software engineering is needed. This can be done in several ways, either by performing experiments in real projects or in laboratories. Both of these approaches have their drawbacks, in the first case it might be too expensive to experiment in a real environment and in the second case it might be hard to draw general conclusions from a limited experiment. This paper presents a limited experiment performed within a university environment. The objective is to get indications, which could form the basis for deciding whether or not to continue with a larger experiment or pilot project in an industrial setting.

Brief introduction to the two methods

Brief introduction to Object-Oriented Software Engineering:

As stated in [Jacobson92] Object-Oriented Software Engineering (OOSE) is an object-oriented process for developing large scale software systems. The process takes a global view of the system development and focuses on minimizing the life cycle cost of a system. An essential part of the method is the concept of use cases. A use case specifies a flow of actions that a specific actor invokes in the system. For example a description of a normal telephone call between two users is a typical use case. OOSE consists of three processes: analysis, construction and testing. In the work five different models are produced. During the analysis process the requirements model and the analysis model are produced. The result of the

construction process is the design model and the implementation model. Finally, the testing results in the test model.

The Analysis process – Here the constructor builds a picture of the system to be created. The requirements and the analysis models are produced in this process (see figure 1).

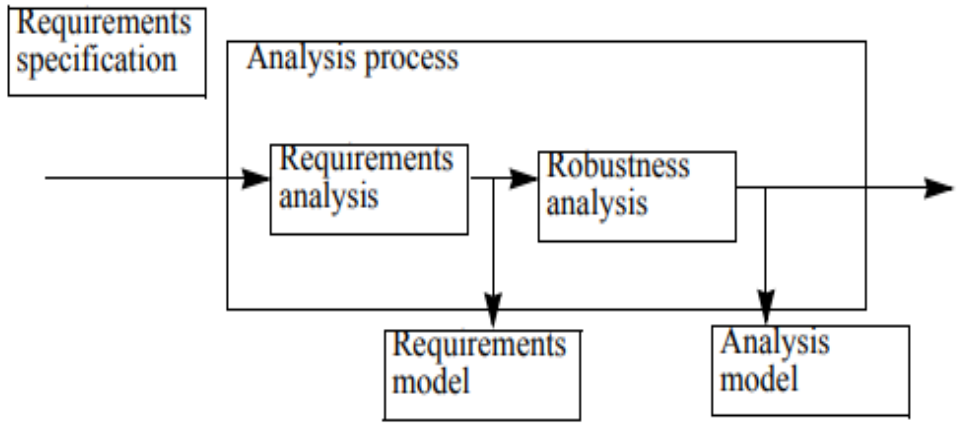


Figure 1. The Analysis process produces two models, the requirements model and the analysis model.

The requirements model – This model endeavors capturing the functional demands of the system, and hence all functionality expected in the system is specified here. The model contains three different parts: a use case model, an interface description and a problem domain model. In the use case model the functionality is described as use cases using natural language. The use case model is also the spine of the Construction and the Testing processes. The analysis model – This model focuses on giving the system a robust and easy to change object structure. The requirements model is the input to this part of the Analysis process, i.e. where the object structure of the system is laid. The Construction process – Both the requirements and the analysis models are inputs to this process. The design and the

implementation models are produced in this process. Therefore, the result of this process is a complete system (see figure 2).

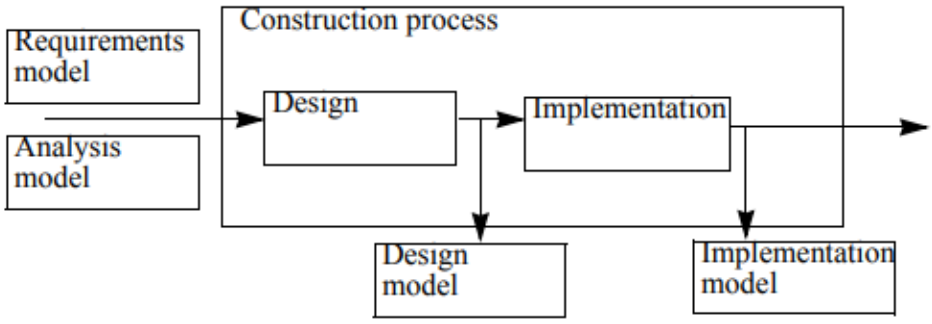


Figure 2. The Construction process produces two models, the design model and the Implementation model.

The design model – In the design model the object structure is adjusted to represent the implementation environment. The analysis model has been developed under idealistic conditions and must now be adjusted to the circumstances prevailing. Any changes in the design model that are of logical nature must be updated in the analysis model. The types of objects used in this model are called blocks. It is important to remember that the blocks are not the same type of objects as those in the analysis model. An interaction diagram is developed for each use case. It contains a detailed description of the different signals sent and received by the different blocks.

The implementation model – This model is the actual implementation of the system. It contains the source code. One does not necessarily have to use an object oriented programming language, but it is preferable. The design model is the base for the implementation. The Testing process – In the Testing process the implementation model is tested, the test model is produced and a decision is taken whether or not the system is ready for delivery. The test model – This model endeavors verifying the system.

Brief introduction to object-based Clean room Object-Based Analysis and Design (COBAD) [Cosmo94a, Cosmo94b] is an object based process for developing large scale software systems. The process focuses on minimizing the life cycle cost of a system by producing high reliability software. This is obtained by making the correct thing from the beginning and by making extensive verifications of the produced material early in the development. Each step in the development is verified against the previous step. An essential part of the method is the concept of Stimulus-Response Diagrams (SRD) [Cosmo94a], which is a formal definition of a use case. An SRD specifies a flow of actions that a specific actor invokes in the system. For example a description of a normal telephone call between two users is a typical event that can be described in one SRD. Development in COBAD is done through three processes: Specification and analysis, Design and implementation, and Certification. The work results in a number of models. During the Specification and analysis process, three models of the system are produced: the use case model, the usage model and the object model. During the Design and implementation process, five models are produced for each object: the interaction model, the use case model, the usage model and the state box model and the implementation model. Finally the Certification process results in the test model. The names of the models are the same for some of the models in the Specification and analysis process and the Design and implementation process, which is due to that they describe the same aspects although on different system levels. The Specification and analysis process – Here the constructor builds a picture of the external behavior of the system. The use case model, the usage model and the object model of the system are produced in this process. The use case model – This model focuses on capturing the functional demands of the system, i.e. all functionality expected in the system is specified here. The model contains three parts: a scenario model, a black box model and an interface description. In the black box model the functionality is based on use cases (or scenarios), using the Black Box Description Language [Cosmo94a]. In this study the black box model is not defined formally. In the interface description the actors of the system and the interface of the system are described. The usage model – This model should capture the usage of the system.

The model describes how the actors may use the system. Each possible sequence of stimuli into the system (the alphabet of the system) must be specified. This can

be done as defined in [Runeson92] or in abstract grammar as it has been done in the study. The usage model is the input to the certification model. The object model – The first step in analyzing the requirements, to obtain the first internal description of the system, is to construct an object model. The object model shows the static data structure of the real-world system and organizes it into workable pieces.

The object model describes real-world object classes and their relationships to each other. The model focus is on the static structure as it is usually better defined, less dependent on application details, more stable as the solution evolves and it is easier for humans to understand.

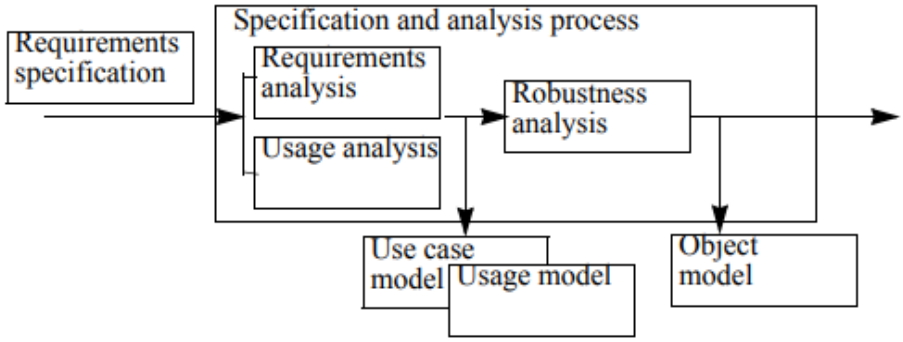


Figure 3. The Specification and analysis process produces three models.

The Design and implementation process – The three models from the Specification and analysis process are input to this process. The interaction model, the use case model of the objects, the usage model of the objects, the state box model and the implementation model are produced. The result of this process is hence a complete system. The interaction model - An interaction diagram is developed for each use case. It contains a detailed description of the different signals sent and received by the different blocks. In the model the responsibilities of the system are distributed to objects in the object model. The use case model of the objects – In this model the object structure is adjusted to represent the implementation environment and to capture the functional demands of the object. The model contains three parts: a scenario model, a black box model and an interface description. In the black box model the functionality is described as use cases (or scenarios), using Black Box Description Language.

The model is not defined formally within this study. In the interface description, the actors of the object and the interface of the object are described. The usage model of the objects – This model must capture the usage of each object of the system. The model describes how the actors may use an object. This is described using abstract grammar. The state box model – In this model the focus is on distributing the responsibilities of the objects to data in the objects. The role of the state box is to open up the black box model of the object one step by making its data visible.

The implementation model – This model is the implementation of the system. It contains the source code. In this study, SDL [CCITT88] was used as implementation language. The state box model of each object is the base for the implementation model.

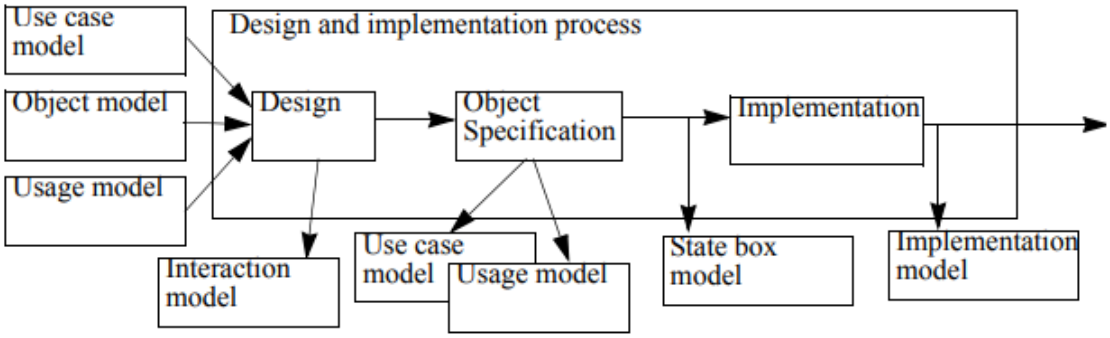


Figure 4. The Design and implementation process produces five models.

The Certification process – The usage models produced for the system and the objects are input to this process. These models are the basis for certifying the implementation model. The test model is produced and a decision taken whether or not the system is ready for delivery. The test model – This model documents the verification of the system.