Assignment 1

3.1 Summery

The article explains why software testing is so hard and constant trade-off through a four phase approach. It explains the possible reasons for the bugs reported by the user. The bugs might have escaped in testing and caught when user testing would be the result from executing untested code, order of execution in actual user, untested input values and user's operating environment never tested.

The testers face problems and technical issues that must be solved before moving to next steps. The inherent difficulties in the process of software testing explained in four phases.

1.Modeling the software's Environment

2.Selecting test scenarios

3.Running and evaluating test scenarios

4.Measuring testing progress

Phase 1: Modeling the software's Environment

Tester needs to simulate each interface in the software system. The interfaces usually categorized into four types. Human interface, the people usually interacts with the system using GUI. Software interface, API that uses operating system, database and runtime libraries. Testers may neglect operating systems information like storage medium is full. File System Interface, to read and write the data to external files, which needs lots of error-checking code to check the appropriate data and formatting. Communication interface, interacts with physical medium and other embedded systems. The tester has to submit different combinations of commands and data to test it.

Interactions with system resources and files, that often needs a check falls outside the control of the testing software. Accessing shared files among multiple processes, interacting properly with system reboot. These often needs a critical consideration of input combinations. A combination of variables and input sequencing is important.

Phase 2: Selecting Test Scenarios

Test scenarios selection always depends on constraints like time and money. Only subset of all scenarios will be applied in any realistic software development schedule. Its need wise selection of test scenarios which have the coverage in terms of source code or input domains. Released products would have very few bugs if code and input coverage were sufficient. To achieve test desired coverage, it has two criteria.

Execution path test criteria:

It focuses on paths that cover control structures. Selecting a set of test in order to execute each source statement at least once. Selecting a set of tests in order to evaluate each branching structure (if, switch, while).

Data flow from one location to other location needs to checked. For that, selection of a set of test that initializes each data structure and use them. Fault seeding is also a method in which test scenarios designed to find intentionally inserted (seeded) errors in the source code.

Input domain test criteria:

Selecting a set of test that contains physical input and cause each interface control to be stimulated has simple coverage to more complex statistical measurement.

Discrimination criteria is to select random selection of input sequences rather than selecting set of tests with same statistics properties and paths likely to be executed by the user.

Phase 3: Running and Evaluating Test Scenarios

Test scenarios are turned into executable form to simulate the user action. To reduce the manual work, The automation will be used to test the scenarios. Data of test case execution gathered from the hooks placed in the code. However, these hooks are removed before releasing the software. The evaluation of the test execution results compared with expected results mentioned in the requirement document. The comparison to analyze the data is hard.

For evaluating the test, there are two approaches: formalism and embedded test code. Formalism is to compare the expected and actual behaviors based on SRS. In the absence of bugs, testers find only obvious bugs and it wastes significant time when tester reporting unspecified features as bugs Other approach is embedded test code. Testing the code that exposes internal data and other type is to check the system state after performing some routines and applying inverse routines.

Regression testing is to test the system through subsequent releases till we get stable version to release. Evaluation may have concerns like tester may not reproduce the same bug he encountered by running a random test case. It raises issue that test scenario re-execution in order to produce the bug or debug the issue.

Phase 4: Measuring Testing Progress

Testing progress often measured as count the tasks we perform while testing. No. of inputs applied, percentage of code covered and no. of failures we found. The structural and functional completeness will be helpful in determining when to stop testing and release the product.

Testability is the new concept proposed for measuring the testing progress. The product with high testability is easy to test and easier to find bugs in. Low testability requires more tests to get the same conclusion.

Reliability Models are mathematical models of test scenarios and failure data that attempt to test predict future failures. These models make some assumptions about the underlying probability distribution that handles failure occurrences. Many expressed skepticisms about these reliabilities based on assumptions.

3.2 Concepts that I learned

1. Statement coverage/line coverage

Process of covering each source line to be executed at least once by the selected test cases.

2.Branch coverage

Process of evaluating each branch structure (If, case, while) with each of test case possible values.

3.Functional testing

Regardless of source code structure, the selection of test scenarios (including test selection methods and test data adequacy criteria) should be based on attributes of the specification and not on attributes of the code.

Ex: Specification based testing, behavioral testing and black-box testing.

4. Structural testing

testing that requires that inputs are based solely on structure of the source code or its data structures.

Ex: Code based testing and white box testing.

5.Regression Testing

The testing which is carried way with tests that were run against earlier versions of the software. Regression is retest the software of current version n using the tests performed against previous version n-1.

3.3 Argument

As mentioned, Testability is the new proposed concept of measuring the testing progress. How the testability is defined in order to find the undiscovered ones. Any measurement should be based on requirements and test scenarios selected for testing the software. If predictive ability is the case for ensuring high testability, how can it be assuring the right results in case of dependent software upgrades happened.

3.4.

During the release of mobile application in Agile model, Our testing faced the issues like nor reproducing the issues they encountered while testing. Testers used to record their manual testing in order to capture issue which is unable to reproduce latter. We encountered issues mentioned in regression testing. While fixing the bugs reported by testers, the code change lead to all possible cases like bug not fixed & something else broken, bug fixed & something else broken. These things often led to lot of re-testing and rework. But we could be able to solve this problem by checking multiple commits we have done in control version git. Using Xcode tool, Testers used to place hooks in the code to gather test results by running automation profile in the tool.

3.5 Questions

1. what are the tools used for Functional and structural testing?

2.How testability adapts with current software model like Agile. How can we use it?

3.6 Experience

I have worked on Web and iOS mobile applications for 4 years. Having used objective-c as front end and php& MySQL as backend, developed aerospace services mobile application, especially designed for Pilots, which includes core services like Flight planning, weather, Runway analysis and Weight & Balance.

The web applications were built using Dreamweaver for web front end development, Tortoise git tool for code check-in and check-out, Win Scp for FTP server connection, Jira for Project test cases management tool and git for version control and configuration management.

We used XCode for iOS application development. XCode provides various tools like Static code analyzer, memory management, Zombie identification tool, automation tool and allocations. Using these tools, we used to test the application performance. We used Google Analytics for capturing User statistics like Usage of the application.

For Testing the mobile application, we have enterprise environment set up i.e Enterprise app store where the beta version of the application can be handed over to 100 members of the registered users in the Enterprise app store (Usually Apple does not allow the beta versions of the applications to be tested on any devices, it needs Device UDID).

We followed waterfall model for web application development. For Mobile application, we used to have scrum meetings as part of Agile model. Usually we met QA teams whenever we have releases on fortnightly. The QA team has process check in the release audit. They will thoroughly verify Code tags, SRS tags, unit test case document results, system test case document results, UAT test case document results. And, randomly they will pick one SRS and check the code flow along with all required tags in the code.

Once we would get approval from QA team. We push the code to the production and test the application. If we find any bugs in the production, we consider them PRD (post release defects). Based on priority that should need immediate attention and fix it.

3.7 Application Overview:

eCards Mobile Application:

This application is designed to make card less transactions. It digitalizes all banking cards like debit cards, credit cards, forex cards and gift cards. The customer does not need to carry banking cards in his wallet. The customer with valid customer id and password can login to the system and can able to see all cards belongs to him on the dashboard. Based on card type, the cards are categorized and displayed in well arranged. Upon selecting the card from the any category, the customer generates QR code by selecting card with sufficient amount in the code.

There is other application of the bank to scan the QR code generated on ecards mobile application. The QR code contains the card details and scanned against the scanner app. On successful scanning, the transaction is processed and the customer gets the notification of the transaction.

Along with this, the customer can be able to see the previous transaction record on any selected card. If any issue to be reported to the customer care, the customer services offers FaceTime and email options to report the issues.