Software Engineering (CEN5035)

Reading Review Exercises: Chapters 9 and 10

Name: Vikas Chaubey, **UFID:** 3511 5826, **Email:** vikas.chaubey@ufl.edu

Qns.1: 1. Explain why you can never be completely confident that all of the faults in a software product have been revealed by program testing?

Ans.1: Software testing is a process of verification of software functionality by using data which simulates user input**.** Basically, software testing checks whether or not a software is behaving as per expectations for a given user input. To perform software testing a tester designs test cases to test software functionality, the data used to perform testing for these test cases is called test data and the expected output for the test using test data is already known in advance. Test data is supposed to simulate the actual real-world user input data. A software passes the test if it produces the same output as the known value for different test cases. If it produces different results, then test was failed, and it confirms that software has some bugs which are supposed to be fixed. Once the bugs are identified then software is updated with bug fixes and the software testing is repeated. This process is repeated till the software is bug free.

However, it is not possible to find out all the faults and bugs in software with software testing because of following reasons:

1) It is very difficult to cover all the possibilities of real-world input data in form of test data to test all possible input data scenarios that a software might face, hence preparing test data sets which can provide hundred percent coverage is nearly impossible.
2) It is very hard to predict the unexpected data scenarios in advance which software might encounter during real time execution. Hence software behavior for this type of data is not known.
3) Another important factor is that design issues in the software cannot be tested completely because a software incorporate various design decision and assumptions which could prove very hard to cover completely and test it.
4) It is very difficult to test the each and every software statements with all possible user input data and combinations of all possible execution path combinations, hence software testing cannot identify all the faults in the software system.
5) Software Testing always involves some component of human contribution for example understanding the software specification and writing software test cases, hence there is always a fair possibility of a human error, that few important test cases have been missed.
6) Sometimes a software performance and its behavior also depend on surrounding environments for example a software system used in nuclear reactor is dependent on physical environmental constraints of the reactor itself, emulating those environments for testing purpose might prove to be very difficult and sometimes it cannot be done.
7) Another important factor is timing constraints, software testing done in industry is always time bound and done within pre decided deadlines, sometimes this time might not be sufficient to identify all possible faults in the software system.
8) User testing is performed to verify whether or not the software is usable and useful to end user, however the term end user is very wide, and it is not possible to verify the software for its usability for wide range of users on basis of feedback of only few people.
9) Security Testing is supposed to test the software against possible theft, security breach and damage. However, software cannot be tested for security against new unique ways which hackers might use to breach software security.
10) Performance and load testing are supposed to test the software performance under heavy load, emulating exact real time heavy loads which software might face in real time situations, is not possible during load testing.

Qns.2: Consider the following pseudo-code program:

```
p := 1
k := 0
while k<>n do     (Note: "<>" means "not equal to")
   p := 2*p
   k := k+1
end_while
```

Is the program correct? Carefully explain your answer.

Ans.2:

No, the program is not correct because the variable 'n' is not declared neither it is initialized hence program does not define its value, data type etc. Hence using it the condition check in while loop is not correct. If this pseudo code is translated to a program, then this program will throw error.

Qns.3: What is regression testing and why is it important? Explain why automated testing makes regression testing straightforward.

Ans.3:

Regression Testing: Regression testing is performed on a software when a software is updated or modified. A software might be updated or upgraded in order to incorporate new functionality, the software could also be modified in order to resolve an identified fault or bug. When such modifications are made to the software system then regression testing is performed. Regression testing is performed to make sure that modification in the program does not introduce any issues or bugs in program functionality. And it works as per expected and as per its specification.

Why Regression testing is important?

The main goal of regression testing is to verify that the already existing functionality of the software system is not affected by any update or modifications made to the software. When a code change is made to the software system then before the modified software is deployed in production , it has to be tested with regression testing first to verify that the changes made in the system have not introduced any functional issues or bugs and whether the software is working as per expected. If no issues are found in the system after testing, then the functionality testing of the build artifact is performed and if again there are no issues then the build is deployed in the production. If there are any issues found during regression testing, then functional testing is halted and build deployment is not done till the issues are resolved. The main advantage of regression testing is that it makes sure that modified software build which goes in production is stable and bug free. It makes sure that new changes made to the software system do not affect software functionality.

Regression Testing with Automation Testing: Automated testing is done by writing automated test case units to verify the functionality of the different code blocks in the program. Automated unit testing and integration testing are good examples of this. Automation testing makes regression testing very straightforward. So, when the code units are written then a developer can write automated tests to verify whether they function accordingly as expected. Various test units written for the multiple code blocks of the program can verify the functionality of the whole program.so when any change is made in the code, all the automated test cases could be run again , if any test case fails to pass then that means the code change has introduced a bug in the program. If all the test cases pass successfully then we know that old functionality of the program is intact without any bugs. That's the whole purpose of regression testing, to check for the possible bugs in the program after code modification. Automation of regression testing reduces testing effort significantly, with automation regression testing could be done faster. And that time could be utilized to perform other development activities.

Qns.4: a). Briefly outline the code review process.
b). Give three reasons why you should use code reviews as well as (machine based) testing when you are developing software.
Ans.4:

a) **Code Review Process:** In the process of software development code reviews are essentially important. Software development requires attention to details, because even a small mistake made by a software developer can result into serious software issues and errors. While writing code it is natural for developers to miss important details and make mistakes. Hence a code review process done by peers in the team can help in identifying those mistakes. Various issues like design problems, wrong syntax, wrong development practices and even software bugs could be identified in advance during development process before the software enters the quality assurance phase. Code review practices fasten the code development process by reducing the testing effort and code review encourages good software development practices among developers in the team. It facilitates knowledge sharing among peers.

A code review is not just intended to identify bugs and errors. Code reviews can also help in refining software design and Implementing new software features as well. It makes sure that the written code by different developers follow the same standard set by organization. It is very important to set the goals for code review process. So that all peers can work together to achieve these goals. A programmer can setup a code review with another peer initially and then prepares the code for review. Then programmer send the code to the reviewer and then it is reviewed. After review, reviewer prepares a report. After that reviewer and programmer can discuss the different issues and programmer can work on to resolve them afterwards. Various organization use ticketing systems during code review process to, it could be used for commenting and discussions. For every identified issue, a ticket could be raised in the system. Sometimes running some tests on the piece of code which is under evaluation also help identifying the bugs faster than just looking at the lines of code. After an overview an in-depth evaluation is done to identify any issue in the program and final evaluation is posted.

b) Three reasons to use code reviews as well as machine-based testing while developing software:

1. Code reviews help in identifying the bugs and faults in the program in advance before the testing phase starts, this could fasten the development process and reduce the testing time significantly. On the other hand, machine-based testing helps in testing the code thoroughly which is not possible to do, during code reviews when both of these are done together software quality is improved significantly.
2. Code review encourages developers to follow effective software design which is result of multiple code reviews and discussions among peers, though machine-based testing can ensure the validity of code design but it does not provide a mechanism to improve software design hence using both together these two boxes could be checked.
3. Code reviews ensure that teams involved in software development are following practices and standard set by organization, this improves code more standard, readable and qualitative. This helps in simplifying automated machine-based test units as well which check the code for correctness. Both of these processes complement each other and together increase program design and code quality.

Qns.5: Consider the program specification: If the input is an ordered pair of numbers, (x, y), a message is output stating whether they are in ascending order, descending order, or equal. If the input is other than an ordered pair of numbers, an error message is output and associated equivalence classes:

{ (x, y) | x<y } (Valid input class)
{ (x, y) | x>y } (Valid input class)
{ (x, y) | x=y } (Valid input class)
{ input is other than an ordered pair of numbers } (Invalid, i.e. "incorrect input" class)

Indicate whether or not the underlying assumption of partition testing (that all the inputs associated with each equivalence partition/class will be treated in the say way in your code -- i.e., either mapped to a correct output or mapped to an incorrect output) would hold for each of the following program designs. Indicate "yes" (the assumption would hold) or "no" (the assumption would not hold) as appropriate for each.

Ans.5:

a) **output ("hello world")**

Ans: YES

b)
```
if (input is an ordered pair of numbers)
  then output("equal")
else
  output ("invalid input")
end_if_else
```

Ans. YES

c)
```
if (input is other than an ordered pair of numbers)
  then output ("invalid input")
else
  if (x<y)
    then output ("ascending order")
  else
    output ("descending order")
  end_if_else
end_if_else
```

Ans. YES

Qns.6: Why is it desirable to design test cases that cover "invalid equivalence classes" (Sommerville calls these "incorrectness partitions") ONE CLASS AT A TIME?
Ans.6: In order to test the program efficiently the testing input data should be divided into sets which would be handled by the code in similar fashion. These input data sets are called equivalence partitions or classes. In order to test the software system thoroughly it is important to test the system with correct and valid input data sets ,here valid input data means the kind of input which the software expects, but it is also very important to test the software system with invalid input data. This is important because software system's behavior should be checked against unexpected data, Software system should be tested with unexpected data sets to make sure that software handles the unexpected data in an expected manner without any failures, these type of testing data sets are called invalid equivalence partitions or classes. When the test cases are designed for invalid equivalence classes then it is preferred to do it one class at a time because the expected behavior of software for a specific input data partition is similar hence test cases could be designed to test that specific behavior of the software in detail. Designing test cases for one class at a time keeps the testing approach very modular and focused to test specific behavior of software system as per input data sets. This helps in testing the software thoroughly and identify as many bugs as possible.

Qns.7: Explain why adopting DevOps provides a basis for more efficient and effective software deployment and operation.
Ans.7: Adopting DevOps makes the deployment and operations more efficient with following benefits:

1) Introduction of DevOps removes the time delays and overheads associated with traditional "Development- Release - Support" Model in software development process. In the traditional process all these components of software development are handled by separate teams which introduce significant delays in the software life cycle. In DevOps all these tasks are handled by one single team hence these overheads are removed, hence software development and delivery become fast.
2) DevOps encourages release of new software updates frequently in incremental form. Hence the software updates are deployed in production in form of small releases. Hence there are less chances of those small new updates causing big failures in the software system in the production environment. Even if the there are any issues after releasing new updates, then the software system could be easily rolled back to previous stable version.
3) In DevOps the software development and support are handled by the same team hence software bug fixing during support is more efficient. If production environment faces any issues then these issues could be fixed very fast, which is much needed. The downtime of software service because of any issue or error should be minimal. DevOps makes software bug fixing faster and efficient.
4) DevOps methodology results into more productivity in the teams. The software development process could be effectively managed because all the tasks are handled by a single team.
5) Automated methodologies like continuous integration, continuous deployment etc. make deployment processes more reliable compared to traditional release methodologies because they involved human interference.
6) DevOps provide more time for teams to be more creative and innovative in software development process instead of just spending more time in mundane tasks.
7) DevOps has various business benefits such as more stable operating environments and improved communication and collaboration among team members.

Qns.8: What is issue management and why is it important for software product development?
Ans.8: It is important to understand what an issue means in software development process. An issue is basically the deviance of the software system from its expected behavior or functionality. Hence the main focus of the whole software development process is to avoid the software issues or fix them when they are encountered so that software works as per its required specifications. This is where the issue management comes into picture in the software development process and that's why issue management is very important to the development process.

The issue management is a range of processes which involve identification of any issues in the software system to resolving the issues. This management includes functions which include methodologies to identify issues in the software system, documenting the issues, then allocating tasks to team members to resolve the issues, the steps involved for the resolution of the identified problems and finally documenting all encountered issues in the life cycle so that issue resolution process could be optimized for future. Some best practices in Issue management are as follow:

1) Recording the issues as they occur in the software system
2) Documenting the details of the software issue for example issue type, whether its functional, or related to performance or User interface etc.
3) Categorizing issues based on priority low, medium, high etc.
4) Maintaining issue log to track the progress mand monitoring the status such as new issue, assigned for resolution, reopen issue, retest issue, resolved etc.
5) Reporting the issue to the stakeholders and people involved in development process
6) Establishing and sticking to guidelines of issue management

Qns.9: Why does the use of continuous integration make it easier to find bugs in the software that you are developing?
Ans.9: When a small change is made in software system or the software system is updated and then these changes are pushed to the repository. In that scenario integration means that these new changes made to the software system are integrated with existing software and then this whole code with new functionality is tested. Continuous integration means that every small change made in the software system is sent to the integration server for testing continuously as soon as new changes are available. This is the concept of continuous integration.

Why continuous integration is effective?
Continuous integration makes it very easy to find the bugs in the software system mainly because for every small update or change in the system, the stable system is tested with these small update, so if there are any bugs found during the testing then it is very easy to localize the issue , because then it's very clear that the issue is arising because of the small update which was made to the system. That's why localization of issue in the software code is very easy hence the bugs could be found and fixed easily. On the contrary when the large updates are tested with software system then localization of issues or bugs in the code base becomes difficult. Hence continuous integration is more effective to find bugs in the system than normal integration which is done sporadically.

Qns.10: What are the differences between continuous integration, continuous delivery and continuous deployment?
Ans.10: The differences are as follow:

1) **Continuous Integration:** In continuous integration whenever a new code update is available , a developer merges that new functionality with the original code base, a build is created and then this build is tested using automation testing, already written automated test cases check whether the new added change to the software effect the software functionality in any way or not, means that every small change made in the software system is sent to the integration server for testing continuously as soon as new changes are available. This is the concept of continuous integration. Continuous integration mainly makes use of automation testing to test the build.

2) **Continuous Delivery:** Continuous delivery is an extended form of continuous integration. Continuous delivery adds an extra layer of through testing on top of continuous integration automation testing. The goal of continuous delivery is to make the modified software production ready with through testing once it clears continuous integration automation testing. That means after the build passes the continuous integration phase then the modified software is tested with regression, UI and performance testing extensively to check build stability and make it production ready. Then If the new modifications are bug free then an deployment pipeline is triggered, and the new build is deployed to testing or production environments. These deployment pipelines could trigger automatically on the basis of artifact availability, or on scheduled basis or manually whenever required.

3) **Continuous Deployment:** Continuous deployment takes this model one step ahead by adding an automatic deployment pipeline which is responsible to deploy the new available builds to the production environment. The goal of continuous deployment is to deploy the new code commits or modifications as soon as they are available and tested. While in continuous delivery deployment model is on demand, in case of continuous deployment the whole process is automated hence as soon as new code changes are available , it passes through the CD pipeline and if the code build passes all the pipeline phases then the build is automatically deployed to the production server. These deployment pipelines could trigger automatically on the basis of artifact availability