Cal State Fullerton

# CPSC 254

Software Development With Open Source Systems

- Automated Testing

Instructor: Tejaas Mukunda Reddy

## Automated Testing

- Automated software testing is a process in which software tools execute pre-scripted tests on a software application before it is released into production.
- Software testing is a method of assessing the functionality of a  software program. There are many different types of software testing but the two main categories are dynamic testing and static testing.
- Dynamic testing is an assessment that is conducted while the program is executed; static testing, on the other hand, is an examination of the program's code and associated documentation. Dynamic and static methods are often used together
- The Five Essentials for Software Testing
- The following advice should help clarify your thinking about software testing and help you improve the effectiveness and efficiency of your testing. It is helpful to think about software testing in terms of five essential elements:

    1. A test strategy that tells you what types of testing and the amount of testing you think will work best at finding the defects that are lurking in the software
    2. A testing plan of the actual testing tasks you will need to execute to carry out that strategy
    3. Test cases that have been prepared in advance in the form of detailed examples you will use to check that the software will actually meet its requirements
    4. Test data consisting of both input test data and database test data to use while you are executing your test cases, and
    5. A test environment which you will use to carry out your testing.

## Test Strategy

- The purpose of testing is to find defects, not to pass easy tests. A test strategy basically tells you which types of testing seem best to do, the order in which to perform them, the proposed sequence of execution, and the optimum amount of effort to put into each test objective to make your testing most effective. A test strategy is based on the prioritized requirements and any other available information about what is important to the customers. Because you will always face time and resource constraints, a test strategy faces up to this reality and tells you how to make the best use of whatever resources you do have to locate most of the worst defects. Without a test strategy, you are apt to waste your time on less fruitful testing and miss using some of your most powerful testing options. You should create the test strategy at about the middle of the design phase as soon as the requirements have settled down.

## Testing Plan

- A testing plan is simply that part of your project plan that deals with the testing tasks. It details who will do which tasks, starting when, ending when, taking how much effort, and depending on which other tasks. It provides a complete list of all the things that need to be done for testing, including all the preparation work during all of the phases before testing. It shows the dependencies among the tasks to clearly create a critical path without surprises. You will be able to start filling in the details of your testing plan as soon as your test strategy is completed. Both your test strategy and testing plan are subject to change as the project evolves. Modify your strategy first, if you need to, and then your testing plan.

## Test Cases

- Your test cases (and automated test scripts if called for by your strategy) are prepared based on the strategy which tells you how much of each type of testing to do. Test cases are developed based on prioritized requirements and acceptance criteria for the software, keeping in mind the customer's emphasis on quality dimensions and the project's latest risk assessment of what could go wrong. Except for a small amount of ad hoc testing, all of your test cases should be prepared in advance of the start of testing. There are many different approaches to developing test cases. Test case development is an activity performed in parallel with software development. It is just as difficult to do a good job of coming up with test cases as it is to program the system itself. In addition to figuring out what steps to take to test the system, you need to know the requirements and business rules well enough to predict exactly what the expected results should be. Without expected results to compare to actual results, you will not be able to say whether a test will pass or fail. A good test case checks to make sure requirements are being met and has a good chance of uncovering defects.

## Test Data

- In addition to the steps to perform to execute your test cases, you also need to systematically come up with test data to use. This often equals sets of names, addresses, product orders, or whatever other information the system uses. Since you are probably going to test query functions, change functions and delete functions, you will most likely need a starting database of data in addition to the examples to input. Consider how many times you might need to go back to the starting point of the database to restart the testing and how many new customer names you will need for all the testing in your plan. Test data development is usually done simultaneously with test case development.

## Test Environment

- You will need a place to do the testing and the right equipment to use. Unless the software is very simple, one PC will not suffice. You will need all of the components of the system as close as possible to what it will eventually be. Test environments may be scaled-down versions of the real thing, but all the parts need to be there for the system to actually run. Building a test environment usually involves setting aside separate regions on mainframe computers and/or servers, networks and PCs that can be dedicated to the test effort and that can be reset to restart testing as often as needed. Sometimes lab rooms of equipment are set aside, especially for performance or usability testing. A wish list of components that will be needed is part of the test strategy, which then needs to be reality checked as part of the test planning process. Steps to set up the environment are part of the testing plan and need to be completed before testing begins.

## Conclusion

- If you want to improve your software testing, or if you are new to software testing, one very helpful thing you can do is make sure you have all five of these essentials well in place. Many testers struggle with inadequate resources, undocumented requirements, and lack of involvement with the development process early in the software development life cycle. Pushing for all five of the essentials and proper timing is one way to significantly improve the effectiveness of testing as an essential part of software engineering.
- https://www.techtarget.com/searchsoftwarequality/definition/automated-software-testing

## Types of Automated Tests

### Automated Unit Tests

- Automated Unit tests are written to test on code level. Bugs are identified in the functions, methods and routines written by developers.
- Some companies ask developers to do the unit testing themselves and some hire specialized test automation resources. These resources have access to source code and they write unit tests to break the production code. Due to the presence of unit tests, whenever the code compiles, all unit tests run and tell us the result that whether all functionality is working. If any unit test fails, that means there is now a bug present in the production code.
- Some of the most popular tools present in the market are NUnit and JUnit. Microsoft also provided its own framework for unit testing called MSTest.
- Go through the websites of these tools and they will provide examples and tutorials on how to write unit tests.

### Automated Web Service / API Tests

- An Application Programming Interface (API) makes it possible for software to talk to other software applications. Just like any other software, APIs need to be tested. In this type of testing, GUI is usually not involved.
- What we test here is usually the functionality, compliance and security issues. In web applications, we can test the Request and Response of our application that whether they are secure and encrypted or not.
- This is one of the examples where we can use API Testing. The most popular tool for API testing is SOAPUI which has both free and paid versions. There are other tools as well, which you can use according to your need.

## Types of Automated Tests

### Automated GUI Tests

- This type of automated testing is the toughest form of automation because it involves testing of a User interface of the application.
- It is tough because GUI's are highly subject to change. But this type of testing is also closest to what users will do with our application. Since the user will use the mouse and keyboard, automated GUI tests also mimic the same behavior by making use of mouse and keyboard to click or write to objects present on the user interface. Due to this, we can find bugs early and it can be used in many scenarios such as regression testing or filling up forms which takes too much time.
- The most popular GUI testing tools are QTP (Now called UFT), Selenium, Test Complete and Microsoft Coded UI (which is a part of Visual Studio ultimate and premium editions).
- **Some Misconceptions about Automation Testing**
- **Misconception #1**. Automation is here to replace manual testers.
- Test automation is for helping the testers to make testing faster and in a more reliable manner. It can never replace humans.
- Think of test automation as a car. If you walk, you will take around 20 minutes to reach your home. But if you use a car, you will reach in two minutes. The driver of the car is still you, a human, but.. the car helps the human to achieve his/her goal faster. Also, most of your energy is saved, because you didn't walk. So you can use this energy to perform more important things.
- Same goes with automation testing. You use it to quickly test most of your repeated, long and boring tests and save your time and energy to focus and test new and important functionality.

## Misconception

**As James Bach said a wonderful quote**:

- ***" Tools don't test. Only people test. Tools only perform actions that "help" people test. "***
- Tools can click on objects. But where to click will always be told by a manual tester. I think you get my point now.
- **Misconception #2.** Everything under the sun can be automated
- If you try to automate 100% of your test cases, maybe you will be able to do so, but if that you could do, then our first point becomes false. Because if everything is automated, what will manual tester do?
- Confused? Right?
- Actually, the point is, you cannot automate 100% of your test cases. Because of we, as testers, believe that no application can be 100% tested. There will always be some scenarios which we will miss. There will always be bugs that come only when your application will be used by clients.
- So if the application cannot be 100% tested, how you can promise 100% automation?
- Also, there is a very thin chance that you will be able to automate all of your existing test cases. There are always scenarios which are difficult to automate and are easier to do manually.
- **For example**, One user will enter the data, the second user will approve the data, the third user will view the data and the fourth user is prohibited to view the data. These scenarios can be automated, but they will take a lot of time and effort. So it will be easier if you just do that manually.
- Remember, we use cars to go distances, but there may be lengthy signals on the way, there will be fuel consumption, there will be parking space issues, parking charges and a lot more headache. In some scenarios, we just walk and reach to our destination So, you should not try to automate everything. Only automate those scenarios which are important and take a lot of time to do manually.

## Misconception

- **Misconception #3**. Automation only involves recording and playback.
- Please don't live in a fantasy world. This fantasy is actually created by false advertisements from different automation tool vendors. They say you just record and playback your steps and your test cases will be automated. Well, that's a Big Lie!
- Automation is everything but recording and playback. Pure automation engineers normally don't use recording and playback feature at all. Recording and playback are generally used to get an idea how the tool is generating a script for our steps.
- Once we get to know the scripting, we always use scripting to create automated tests. Remember, you must know programming if you want to do test automation. On the other hand, don't be dis-heartened if you don't know programming. Because like any other task, the programming can also be learned with practice and dedication.
- I have known people, who are not even from Computer science background, but they learn to program and now they are awesome automation engineers. At Microsoft, they hire testers who can do programming. They are called SDET(Software development engineers for test). The first line of Job description says "SDET's write a lot of code….".

## 10 Best Practices and Strategies for Test Automation

- **#1. Hire a Dedicated Automation Engineer or Team**
    - This is a basic thing to do. Don't ask your manual testers to indulge in test automation. If you do want them to do test automation, then free them from manual testing work. Test automation is a full time job. For this you need dedicated resources.
    - I recommend building a test automation team consisting at least one automation architect. You can hire multiple automation engineers to work under the guidance of the test automation architect. The number of automation engineers depends on the number and size of your products.

- **#2. An automation tool is important, but it is not the solution of everything**
    - We talked about tool selection. But selecting the right tool is just the beginning. Some managers have the misconception that if they select a right tool, they can easily automate anything. Beware, automation tools do not give you everything. They make the process easier. But you need skilled resources to complete the process.
    - Often automation tools are buggy and they stuck in identifying complex objects on the application. The resources you hire, if they are skilled, come up with workaround which takes the process forward. Otherwise, if you don't hire good resources, Tool alone cannot guarantee the successful automation.

- **#3. Select the automation tool which is familiar to your resources**
    - If your resources are familiar with C# and your application to be tested is also developed in C#, then there is no point selecting the tool which does not offer C# to write scripts.
    - Language learning is time taking process. Avoid this learning curve by buying a tool which offers minimal learning curve.

## 10 Best Practices and Strategies for Test Automation

- **#4. Know the application being tested**
  - The tool selection depends heavily on the technologies used in your product. Know your product inside out before starting the automation.
  - If it is a web application, know the browsers it will support. Know the technologies being used in it. If it is a desktop application, know which language is built upon. What third-party controls are being used in the application. This will help you make the tool selection and future automation easier.

- **#5. Good Automation means good manual test case**
  - Nicely written strong manual test cases saves us from automating those test cases which are easy to automate but weak in finding defects.
  - Here is the quote from the book Lessons Learned in Software Testing:
  - "Automating without good test design may result in a lot of activity, but little value."
  - It is always advisable to first write the test case in manual form. Identify all prerequisites and test data. Write steps in a clear manner and write expected results in front of each step. The objective of one test case should be clear and it should be less dependent on the other test cases. Automation engineers should run this test case manually at least once to clearly decide what objects need to be identified and what will be the flow of navigation. Ask questions with manual testers.
  - This activity sometimes helps to identify bugs even before the automation script is written. Experts say that majority of bugs is identified in the test automation development phase rather than in actual execution phase.

- **#6. Identify opportunities with automation**
  - If you are handed over with a manual test case to automate, don't just automate that test case as it is. Instead, find further opportunities in your automation, to expand the scope of this test case.

## 10 Best Practices and Strategies for Test Automation

- **#6. Identify opportunities with automation**
  - If you are handed over with a manual test case to automate, don't just automate that test case as it is. Instead, find further opportunities in your automation, to expand the scope of this test case.
  - For example, if the manual test case requirement is you have to login to a web page. You can expand this test case by making it data-driven. List all the possible scenarios of login like invalid password, empty password, invalid username, invalid email, blank username, remember me checked, not checked, etc. List the possible scenarios along with their expected result in an excel file and put this excel file as a data source to your test case. Now this one manual test case, after being automated, can test all the possible scenarios in one go.
  - Always look for opportunities that can be done with automation, but difficult to do manually. Such as Load Testing scenarios, Performance Benchmarks, Same Tests under different environments with different configurations, Memory Leaks, High Precision Tests etc. These all are difficult scenarios for manual testers.

- **#7. You cannot automate everything**
  - Automation means running fewer tests more often. You have to start small by attacking your smoke tests first. Then cover your build acceptance tests. Then move onto your frequently performed tests, then move onto your time taking tests. But make sure every test you automate, it saves time for a manual tester to focus on more important things.
  - Automation is not here to replace manual testers. Nor it can. It is here to take the repeated work away from manual testers so that they can use their full focus and strength in finding new testing scenarios and bugs. (Read my article Misconceptions of test automation)
  - Automate few tests that are valuable and time savers or difficult to do for manual testers. If you did that, the task of automation is done.

# 10 Best Practices and Strategies for Test Automation

- **#8. Avoid GUI Automation when there is an alternate present**
  - GUI automation is always tougher than other types of automated tests. So if there is a situation when you can achieve your target by not automating the GUI, but by some other methods like command line inputs, then the best strategy is to avoid GUI automation.
  - For example, you want to test the installation of the application. The objective is to check whether application installed or not in a particular environment. One approach is to start the installation and click on “Next” button multiple times through your automation tool. It can be tricky, time-consuming and it is subject to maintenance if UI changes. The other approach is to initiate the application installation with a batch file giving silent arguments. The application will silently install showing no GUI. The objective will be achieved in less time and in a more reliable manner..

- **#9. Use Automation for other useful purposes as well**
  - Automation is such a fantastic thing. You can achieve such things from it that you don’t normally think about. Automation is not just about programming a manual test case. Instead, you can use automation to facilitate different operations in your organization.
  - For example, you can use automation to create master data and setup configurations automatically for manual testers. So that they can start their testing as early as possible.
  - I can give one example from my own company. We wanted to switch from our test case management tool. We were using “Test Director” (now HP ALM) and wanted to switch to TFS (Team Foundation Server). We had around 4000 manual Test Cases and Bugs in Test Director. Transferring them manually to TFS could take about a month. So my manager asked me to try some automation.
  - I dig those tools and found out that Test Director is using SQL server as its repository. For TFS, I found out a tool that can read test cases and bugs from an excel file, if they are written in a particular format, and can insert them in TFS. The Rest of the story is simple. I wrote an SQL query to fetch all test cases and bugs and exported them in an Excel File in the specific format. I then used that tool which reads all test cases and bugs from excel file and inserted them in TFS. The whole process took only 3 hours. My manager was very happy. I hope you get my point too.

## 10 Best Practices and Strategies for Test Automation

- **#10. Automation is software development**
  - If you develop a quality software, it needs best practices. It needs code reviews to write quality code. It needs a framework or design pattern to be followed. It needs constant maintenance.
  - Automation is basically software development. So all best practices you follow when you develop a software should be followed in doing automation. Automation Framework should be there. Code Reviews should be done. Bugs of automation should be reported in bug repository. Source Code of automation should be placed under a source control, etc. The more you treat it like software development, the more successful automation will be.