CSE 470
Software Engineering
Testing and Unit Testing

Imran Zahid
Lecturer
Computer Science and Engineering, BRAC University

Intro

- Suppose you are being asked to lead the team to test the software that controls a new X-ray machine. Would you take that job?
- What if the contract says you'll be charged with murder in case a patient dies because of a malfunctioning of the software?

Software Testing

- 30-85 errors are made per 1000 lines of source code.
- Extensively tested software contains 0.5-3 errors per 1000 lines of source code.
- Testing is postponed, as a consequence: the later an error is discovered, the more it costs to fix it.
- Error distribution: 60% design, 40% implementation. 66% of the design errors are not discovered until the software has become operational.

Software Testing

- Check software correctness.
- Testing is the process of evaluating a system or its component(s) with the intent to find that whether it satisfies the specified requirements or not.
- Who does testing?
  - Software Tester
  - Software Developer
  - Project Lead/Manager
  - End User

Error, Fault, Failure

- An **error** is a human activity resulting in software containing a fault.
- A **fault (bug)** is the manifestation of an error.
- A fault may result in a **failure.**
- Error →Fault → Failure

Classification of testing techniques

- Classification based on the criterion to measure the adequacy of a set of test cases:
  - Coverage-based testing
  - Fault-based testing
  - Error-based testing
- Classification based on the source of information to derive test cases:
  - Black-box testing (functional, specification-based)
  - White-box testing (structural, program-based)
  - Grey-box testing

Comparison

	Black Box Testing	Grey Box Testing	White Box Testing
1.	The Internal Workings of an application are not required to be known	Somewhat knowledge of the internal workings are known	Tester has full knowledge of the Internal workings of the application
2.	Also known as closed box testing, data driven testing and functional testing	Another term for grey box testing is translucent testing as the tester has limited knowledge of the insides of the application	Also known as clear box testing, structural testing or code based testing
3.	Performed by end users and also by testers and developers	Performed by end users and also by testers and developers	Normally done by testers and developers
4.	-Testing is based on external expectations -Internal behavior of the application is unknown	Testing is done on the basis of high level database diagrams and data flow diagrams	Internal workings are fully known and the tester can design test data accordingly
5.	This is the least time consuming and exhaustive	Partly time consuming and exhaustive	The most exhaustive and time consuming type of testing
6.	Not suited to algorithm testing	Not suited to algorithm testing	Suited for algorithm testing
7.	This can only be done by trial and error method	Data domains and Internal boundaries can be tested, if known	Data domains and Internal boundaries can be better tested

Levels of Testing

- Functional Testing
  - Unit Testing
  - Integration Testing
  - System Testing
  - Regression Testing
  - Acceptance Testing
  - Alpha testing
  - Beta Testing

- Non-Functional Testing
  - Performance Testing (Load and Stress)
  - Usability Testing (Usability vs UI testing)
  - Security Testing
  - Portability Testing

Functional Testing

- **Unit Testing**
  - This type of testing is performed by the developers before the setup is handed over to the testing team to formally execute the test cases. Unit testing is performed by the respective developers on the individual units of source code assigned areas. The developers use test data that is separate from the test data of the quality assurance team.
  - The goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality.
- **Integration Testing**
  - The testing of combined parts of an application to determine if they function correctly together is Integration testing. There are two methods of doing Integration Testing Bottom-up Integration testing and Top Down Integration testing.
  - **Bottom-up integration** testing begins with unit testing, followed by tests of progressively higher-level combinations of units called modules or builds.
  - **Top-Down integration** testing, the highest-level modules are tested first and progressively lower-level modules are tested after that. In a comprehensive software development environment, bottom-up testing is usually done first, followed by top-down testing.
- **System Testing**
  - This is the next level in the testing and tests the system as a whole. Once all the components are integrated, the application as a whole is tested rigorously to see that it meets Quality Standards. This type of testing is performed by a specialized testing team.

Non-Functional Testing

- **Performance Testing**
  - It is mostly used to identify any bottlenecks or performance issues rather than finding the bugs in software. There are different causes which contribute in lowering the performance of software:
    - Network delay.
    - Client side processing.
    - Database transaction processing.
    - Load balancing between servers.
    - Data rendering.
- Performance testing is considered as one of the important and mandatory testing type in terms of following aspects:
  - Speed (i.e. Response Time, data rendering and accessing)
  - Capacity
  - Stability
  - Scalability
- It can be either qualitative or quantitative testing activity and can be divided into different subtypes such as Load testing and Stress testing.

Non-Functional Testing

- **Load Testing**
  - A process of testing the behavior of the Software by applying maximum load in terms of Software accessing and manipulating large input data. It can be done at both normal and peak load conditions. This type of testing identifies the maximum capacity of Software and its behavior at peak time.
  - Most of the time, Load testing is performed with the help of automated tools such as Load Runner, AppLoader, IBM Rational Performance Tester, Apache JMeter, Silk Performer, Visual Studio Load Test etc.
- **Stress Testing**
  - This testing type includes the testing of Software behavior under abnormal conditions. Taking away the resources, applying load beyond the actual load limit is Stress testing.
  - The main intent is to test the Software by applying the load to the system and taking over the resources used by the Software to identify the breaking point. This testing can be performed by testing different scenarios such as:
  - Shutdown or restart of Network ports randomly.
  - Turning the database on or off.
  - Running different processes that consume resources such as CPU, Memory, server etc.

Test-Driven Development (TDD)

- First write the tests, then do the design/implementation
- Part of agile approaches like XP
- Supported by tools, eg. JUnit
- Is more than a mere test technique; it subsumes part of the design work

Steps of TDD

1. Add a test
2. Run all tests, and see that the system fails
3. Make a small change to make the test work
4. Run all tests again, and see they all run properly
5. Refactor the system to improve its design and remove redundancies

Testing

Manual Testing

- Done by Users
- Refers to using the software and identifying problems
- Generally uses excel sheet or similar tools to keep track of traces

Automated Testing

- Done by Development team – developers, testers
- Refers to writing and running test cases and to identify problems
- Can be done in many ways – testing libraries, automated testing tools

Manual Testing Example

Test Case ID	Test Title	Testing Steps	Test Data	Expected Results	Actual Results	Pass/Fail
TU01	Check Customer Login with valid Data	1. Go to site http://mySampleProject.com	Userid = guru99 Password = pass99	User should Login into application	As Expected	Pass
		2. Enter UserId				
		3. Enter Password				
		4. Click Submit				
TU02	Check Customer Login with invalid Data	1. Go to site http://mySampleProject.com	Userid = guru99 Password = glass99	User should not Login into application	As Expected	Pass
		2. Enter UserId				
		3. Enter Password				
		4. Click Submit				

Unit Testing

- Unit testing is a type of automated testing to check if the small piece of code is doing what it is supposed to do.
- Unit testing checks a single component of an application. The components are methods / functions
- The scope of Unit testing is narrow, it covers the Unit or small piece of code under test. Therefore while writing a unit test shorter codes are used that target just a single class.
- Unit testing comes under White box testing type.

How To Write Unit Tests

- *A TEST CASE is a set of conditions or variables or instructions under which a tester will determine whether a system under test satisfies requirements or works correctly.*
- Setup / pre-conditions are usually defined as a separate property of the test case. Are generally written under the setup method.
- Test steps: instructions / statements written with in the test method.
- A predefined expected result is required for invoking the unit.

Test Case ID	TC_001
Test Case Title	Verify valid user login
Project	User Authentication System
Module	Login/Authentication
Objective	Verify login with valid credentials
Preconditions	Valid account exists, system running
Test Steps	1. Open the login page 2. Enter valid username and password 3. Click "Login"
Expected Result	User successfully logs in and is redirected to the dashboard
Postconditions	User is logged in and on the homepage/dashboard
Test Data	Username: `testuser@example.com` Password: `password123`
Environment	Browser: Chrome OS: Windows 10
Pass/Fail Criteria	Pass: Successful login Fail: Error message or failed login

Thank you