

# Week 1

Saturday, 22 February 2025 11:55 PM

## Introduction to Software Testing

### Unit Summary

- Computing is ubiquitous
- We test to avoid failures
  - o Bad publicity
  - o Loss of users
  - o Sued for damages
- Best Course of action is to follow a systematic process.

### Module Summary

- To find out what is software testing and software testing activities.
- To learn why software testing is important and why software fail?
- To examine some (in)famous failures in the discipline.
- To learn software testing terminology
- To learn classification of software testing techniques

### What is Software Testing?

- Definition: Testing is the process of executing a program with the intention of finding errors (Myers, 1976)
- Testing - an unnatural process as its aim to make the program fail.
  - o Errors, faults, failure
- To be successful in his/her testing activities, the tester must construct his/her test case such that; if faults are present in the software, these faults will be exposed.
- It is not possible to guarantee the absence of errors by testing the software using a large set of test cases.
- **The optimal result of testing - the maximum exposure of errors present.**
  - o Testing can demonstrate the presence of errors but not their absence
- Effective testing is a result of adequate preparations being made before hand:
  - o To arrange the system such that it is easy to test and
  - o Preparing a plan of testing so that the sequence of testing is well organised.

### Software Testing Activities.

- Steps in carrying out testing:
  - o Establish test objectives
  - o Design test cases and writing test cases.
  - o Evaluate the test cases
  - o Execute the test using the developed program.
  - o Examine test results (output from the previous step)

### Software Testing: why

- Testing is an important step in the development of programs
  - o Untested programs are likely to contain bugs, security flaws, logic errors or other undesirable behaviour.
    - Leading to downtime, lower sales, bad reputation, potential litigation, etc...
- Software failures are costly.
  - o 60 billion each year
  - o On average, 1 hour downtime (financial company) costs >6M
  - o System defects account for up to 40% of system failures.

### Causes of Software failure

- Incorrect memory usage (e.g segmentation fault)
- Deadlocks
  - o Two or more process are waiting on each other to release resources, resulting in circular

dependency. Leads to a complete standstill.

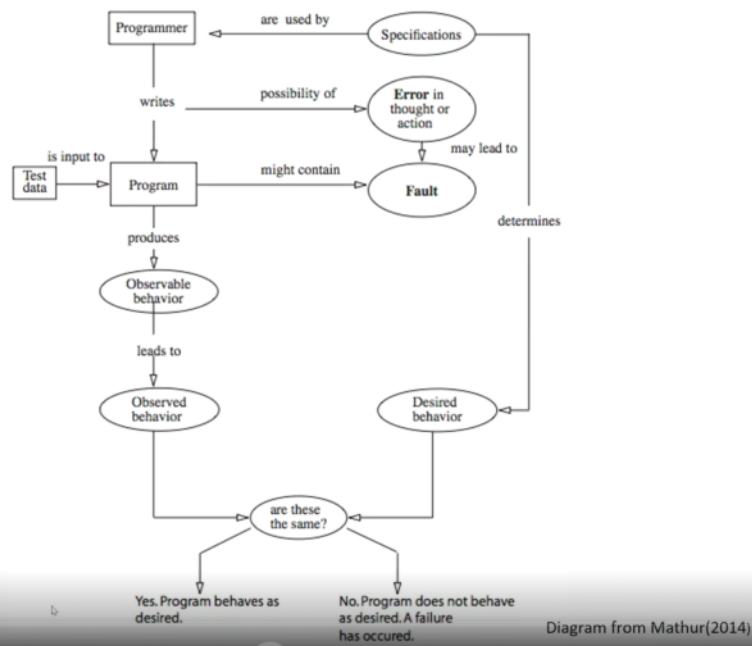
- Memory leaks
- Wrong implementation
  - o Misinterpret requirements
- Regression bugs
  - o Bugs in the system that weren't present when we deployed but became present after a new feature has been implemented
  - o Tricky to track down.

#### Famous Software Failures

- Therac-25
- Morris Worm
- Kerberos Random Number Generator
- Denver Baggage Handling System
- Ariane 5
- Apple's goto fail.
- OpenSSL's Heartbleed.

## Software Testing Terminology

### Error, Faults, Failures



- Human error occurs when writing the program. I.e a typo, miscalculation or misinterpretation of requirements.
- A fault, also known as a defect or a bug is a result of one or more of these errors in code.
- Failure occurs when the fault is executed. Causes the program to enter an incorrect state and return an incorrect output.
- When the actual behaviour doesn't match what we expect, that's a failure.

#### Software Testing Terminology: Software Quality

- **Static Quality Attributes:**
  - o Things that don't change once the software is built.
    - Structured, maintainable, testable code as well as the availability of correct and complete documentation.
    - A shit user manual is still shit, poor code and poor documentation makes it harder to modify and test etc.

- Dynamic quality attributes: software reliability, correctness, completeness, consistency, usability, and performance
  - Focus on how the software performs during execution.
  - Software reliability:
    - the probability of failure free operation of software in its intended environment
  - Correctness:
    - Whether the programs output is correct for a given input.
    - Establish correctness via testing would imply testing a program on all elements in the input domain. In most cases that are encountered in practice, this is impossible to accomplish.
    - Established via mathematical proofs of programs.
  - Testability:
    - The degree to which a system of component facilitates the establishment of test criteria and the performance of tests to determine whether those criteria have been met.
    - As software complexity increases, testability decreases
    - Make software as testable as possible./
  
- Program behaviour
  - When describing how a program behaves we can use English, but can lead to confusion due to misinterpretation.
  - More precise way is to use program states do define behaviour. Involves a state diagram.
  
  - Program States
    - States of a program is the set of current values of all its variables (state vector) and an indication of the next statement to be executed.
    - Program execution of its statement - causes the program to move from one state to the next. (line by line for example)
    - Program behaviour = a sequence of different program states .
  - To specify:
    - Simplest way - natural language (multiple interpretations)
    - State Diagram, formal mathematical specifications
  
- Operational Profiles.
  - An operational profile is a numerical description of how a program is used by different types of users .
  - Think of it as a numerical break down of how often specific functions or inputs in a program are utilised.
  - Systems often have multiple operational profiles. Each dependent on a single group of users interacting with a system. One profile might be for accountants who will generally input numbers all day, another may be managers who write reports all day consisting of strings.
  - By using operational profiles we can make the best use of our testing resources. We can test more rigorously , ensuring that the test cases that do pass are the most reliable for that group of users.
  
- For example if we know a system mainly handles numerical inputs, we should focus most of our inputs on the numerical test domain.

## Examples:

Operational profile #1		Operational profile #2	
Sequence	Probability	Sequence	Probability
Numbers only	0.9	Numbers only	0.1
Alphanumeric strings	0.1	Alphanumeric strings	0.9

## Software Testing Terminology

### Test/debug cycle

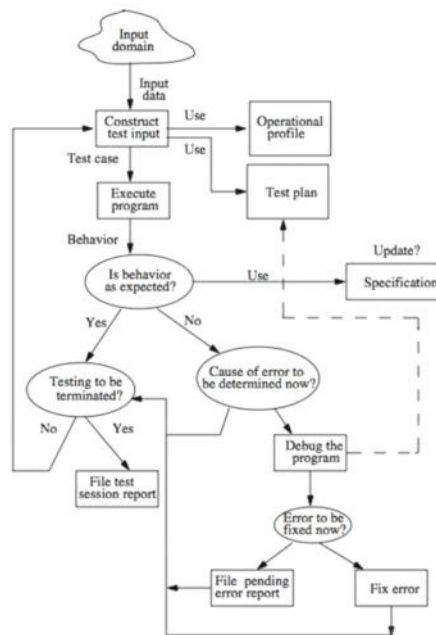


Diagram from Mathur(2014)

### Software Testing Terminology

#### - Test Oracle

- The entity that performs the task of checking the correctness of the observed behaviour of a program.
- Literally just checks the input vs the output.
-

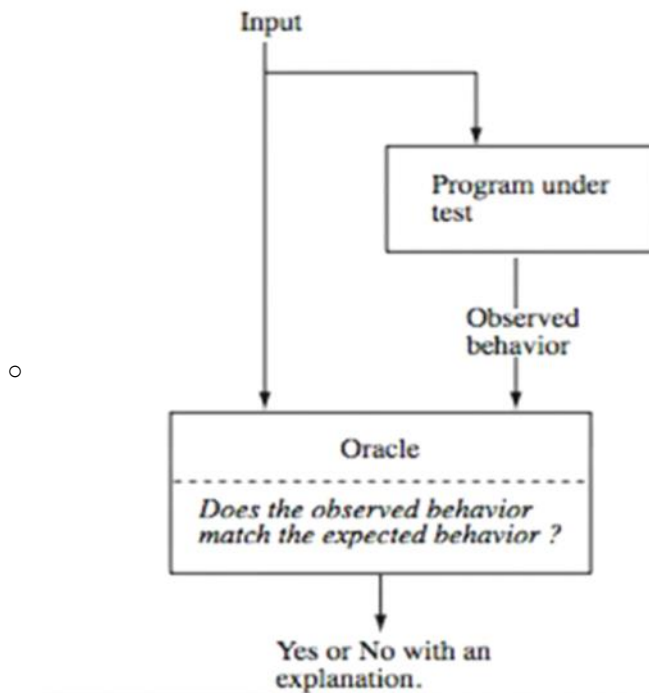


Diagram from Mathur(2014)

## Classification of Software: Testing Techniques

### One Possible classification:

- C1: Source of test generation
  - o Where do our test cases come from? Are they manually generated or derived from specification?
- C2: Lifecycle phase in which testing takes place
  - o Are we testing during development? After development or during maintenance?
- C3: goal of a specific testing activity
  - o What are we trying to achieve? Measure performance or find defects?
- C4: Characteristics of the artifact under test.
  - o What is actually being tested? A single module or the entire code base.

# Classification of Software Testing Techniques: C1

Artifact	Technique	Example
Requirements (informal)	Black-box	Ad-hoc testing
		Boundary value analysis
		Category partition
		Classification trees
		Cause-effect graphs
		Equivalence partitioning
		Partition testing
		Predicate testing
Code	White-box	Random testing
		Adequacy assessment
		Coverage testing
		Data-flow testing
		Domain testing
		Mutation testing
		Path testing
Requirements and code	Black-box and White-box	Structural testing
		Test minimization using coverage
Formal model:	Model-based	Statechart testing
Graphical or mathematical specification	Specification	FSM testing
		Pairwise testing
		Syntax testing
Component interface	Interface testing	Interface mutation
		Pairwise testing

Taken from  
Mathur (2014)

- Black box testing:
  - o Focuses on testing from a user perspective .
  - o Not concerned with the internals of the code but rather testing the inputs and outputs.
- Adhoc testing
  - o Random testing
- Boundary value analysis
  - o Testing at the edges of input ranges
- White box testing:
  - o Involves looking at the code, examine structure, security and performance.
- Finding security issues and poorly written code.


## Classification of Software Testing Techniques: C2

Phase	Technique
Coding	Unit testing
Integration	Integration testing
System integration	System testing
Maintenance	Regression testing
Post system, pre-release	Beta-testing

Taken from  
Mathur (2014)

- We can test during each phase of the lifecycle.
- Unit testing:
  - o Focus on individual classes or functions. Ensure that each unit works directly in isolation

- Integration testing:
  - o Combining two or more units (subsystem). Verify that these units work together.
- System testing
  - o Ensure that all subsystems work together.
- Regression testing
  - o After the system is deployed. Ensure that new updates don't introduce new faults or break functionality.




### Classification of Software Testing Techniques: C3

Goal	Technique	Example
Advertised features	Functional testing	
Security	Security testing	
Invalid inputs	Robustness testing	
Vulnerabilities	Vulnerability testing	
Errors in GUI	GUI testing	Capture/plaback Event sequence graphs Complete Interaction Sequence Transactional-flow
Operational correctness	Operational testing	
Reliability assessment	Reliability testing	
Resistance to penetration	Penetration testing	
System performance	Performance testing	Stress testing
Customer acceptability	Acceptance testing	
Business compatibility	Compatibility testing	Interface testing Installation testing
Peripherals compatibility	Configuration testing	

Taken from Mathur (2014)

- Goal directed testing. Focuses on specific areas. Ensure that software behaves towards specifications.



### Classification of Software Testing Techniques: C4

Characteristics	Technique
Application component	Component testing
Client and server	Client-server testing
Compiler	Compiler testing
Design	Design testing
Code	Code testing
Database system	Transaction-flow testing
OO software	OO testing
Operating system	Operating system testing
Real-time software	Real-time testing
Requirements	Requirement testing
Software	Software testing
Web service	Web service testing

Taken from Mathur (2014)

Software Testing Tools:  
Examples

## Python Testing Tools

- Unittest, Pytest
- Check out the tools at: <https://pythonhosted.org/testing/>

## Java Testing Tools

- Junit (<http://junit.org>)
- TestNG (<http://testng.org/doc/index.html>)

## Defects Tracking

- Bugzilla (<https://www.bugzilla.org/>)