INFO6030
System Analysis and Design

Welcome

Week 10

Lecture Slide Content

- These lecture slides are the basis of the lecture.
- In other words, the actual lecture may cover more than what is contained in these slides.
- **Why**
    - A question may be asked that is related to an assessment
    - A point may be raised and then we chat about that idea
    - I may ask questions in the lecture and chat about the responses
    - I may related the content to assessments or current events
    - Example/s may be discussed due to the above actions

Indicative Course Schedule

Week	Week Begins	Topic	Learning Activity	Assessment Due
1	13 May	Introduction This is an indicative course schedule	Lab: Review Questions Team formation	
2	20 May	Unified Process (UP) and Models	Review Questions Case Study Team formation	
3	27 May	UP Information Gathering and Modelling	Review Questions Case Study	
4	3 June	Modelling Use Cases and Class Diagrams	Review Questions Case Study	
5	10 June	Feasibility	Review Questions	Quiz 1 (in lab)
6	17 June	Revision	Review Questions	Assessment 1: Friday 11.59pm
7	24 June	State Diagrams	Short Presentation (each team) Case Study	
8	1 July	Design Class Diagrams	Review Questions Case Study	
9	8 July	Sequence Diagrams	Review Questions	Quiz 2 (in lab)
10	**15 July**	**Testing Controls and HCI and Revision**	**Review Questions** **Case Study**	
11	22 July	Implementation and Deployment	Review Questions	
12	29 July	Revision	Short Presentation (each Team)	Assignment 2: Friday 11.59pm

Contact Information

Course Co-Ordinator: eugene

Lecturer: eugene Lutton

Email: eugene.lutton@newcastle.edu.au

Email Subject header:

- INFO6030 / Location / Reason
- For example: INFO6030 / Online / Quiz 2
- For example: INFO6030 / Callaghan / Video week 9

- **Consultation:** *Please email for appointment*

(Metal Deploye Resistor, n.d.)

Testing

(360Logica, n.d.)

Lecture Overview

Activities from Use Case Development are needed to bring a system into being:

- Implementation
- **Testing**
- Deployment
- Configuration and change management

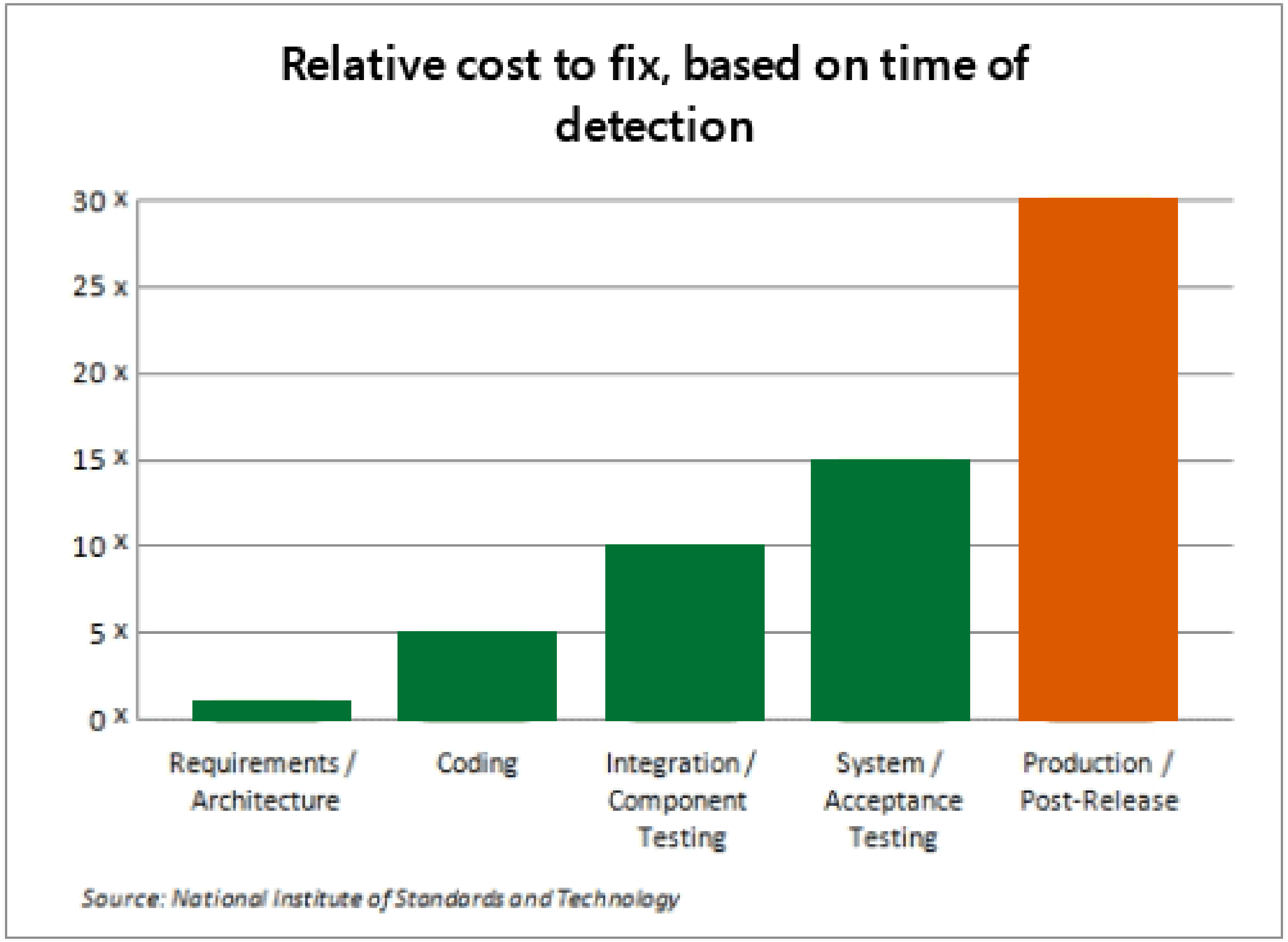
Testing Is Important

- It is not unusual for developers to spend 50% of the total project time on testing

- For life critical software (flight control) testing can cost 3 to 5 times as much as all other activities combined

- The *destructive* nature of testing requires that the developer discard preconceived notions of the *correctness* of his/her developed software

Testing Is Important



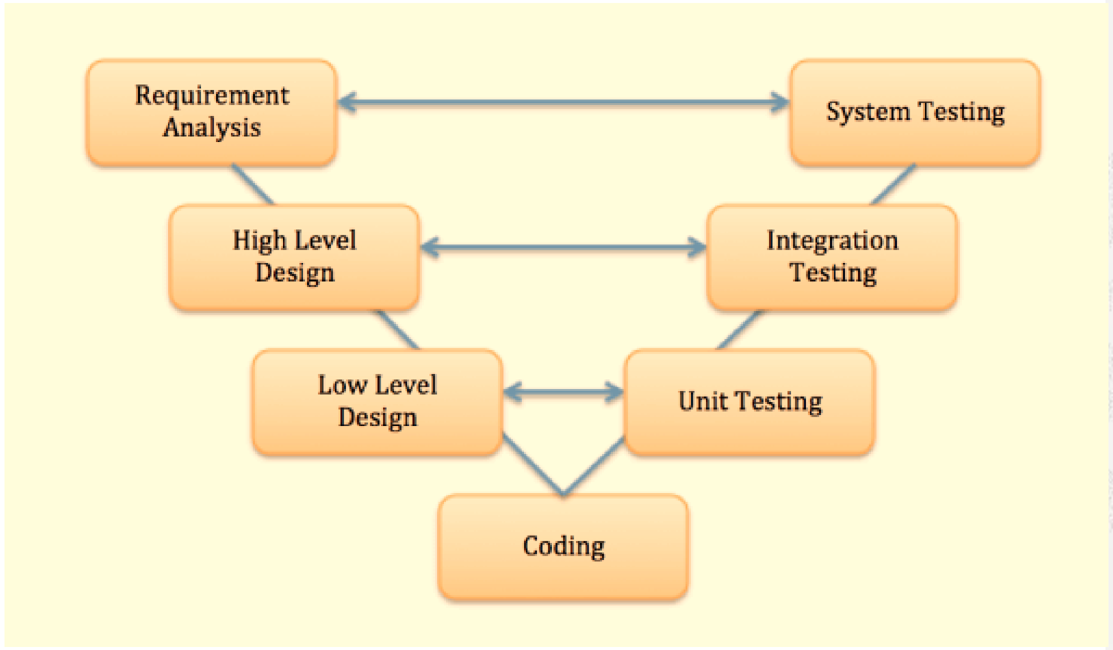
(Guru99, n.d.)

Testing: Terms

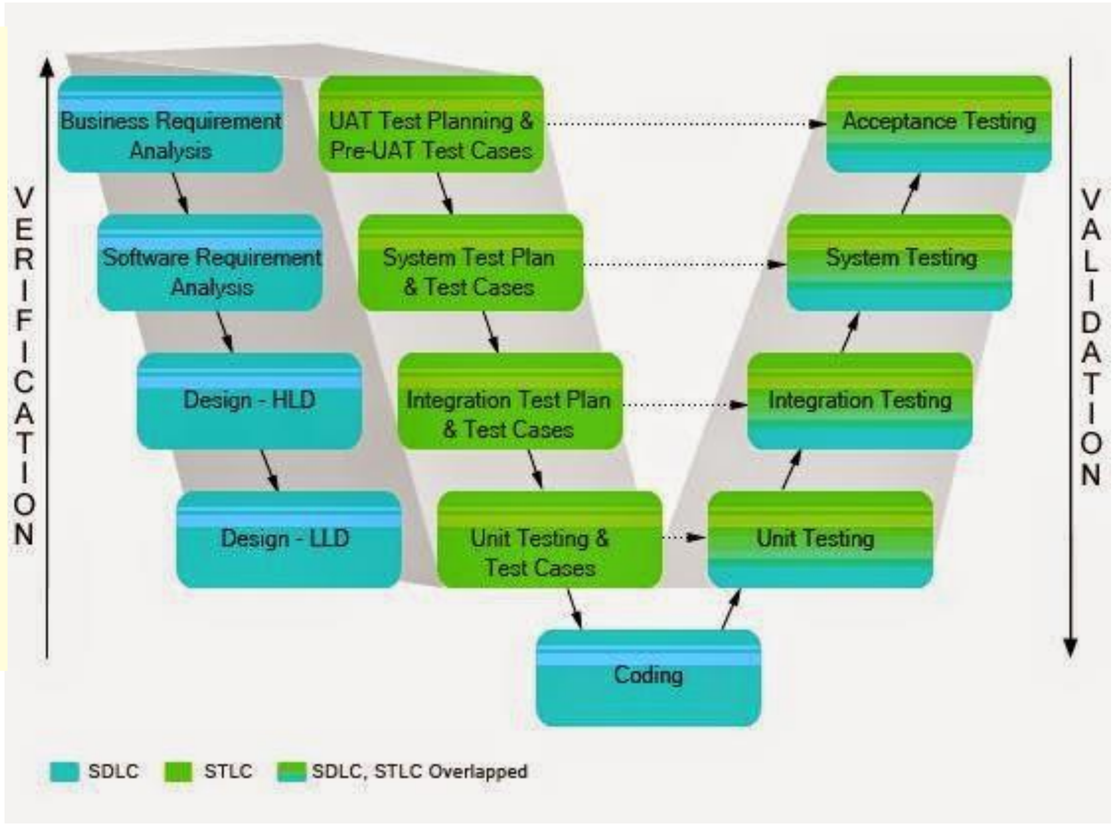
- Testing is a process of identifying defects
- Develop test Plan Test Cases and test data
  - ***Test Plan*** describes the test strategy, objectives, schedule, estimation, deliverables, and resources required to perform testing for a software product.
  - ***Test data*** is a set of ***starting states*** and ***events*** used to test a module, group of modules, or entire system
  - A ***test case*** is a formal description of
    - A starting state
    - One or more events to which the software must respond
    - The expected response or ending state
- See the weekly readings at end of lecture

SDLC and STLC: The V Model

(Guru99, n.d.)

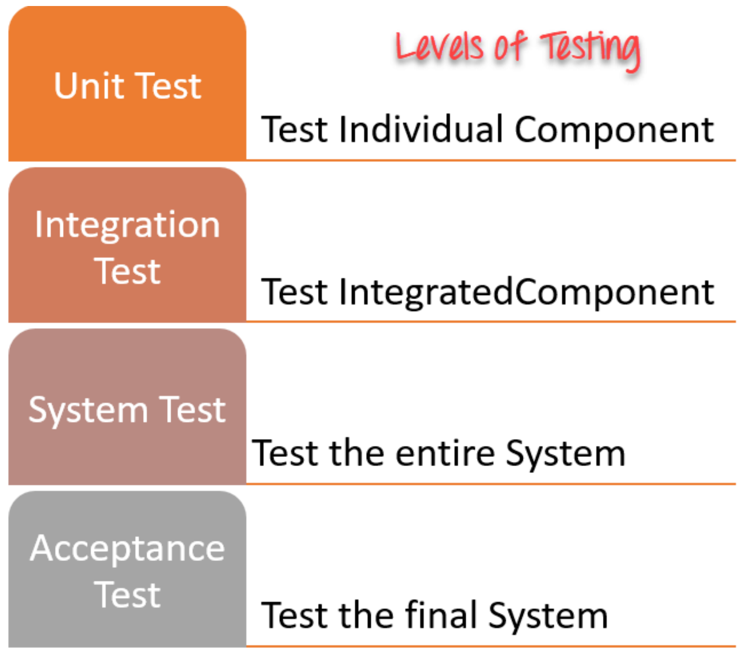


(Soni, 2015)



VERIFICATION
Business Requirement Analysis
UAT Test Planning & Pre-UAT Test Cases
Acceptance Testing
Software Requirement Analysis
System Test Plan & Test Cases
System Testing
Design - HLD
Integration Test Plan & Test Cases
Integration Testing
Design - LLD
Unit Testing & Test Cases
Unit Testing
Coding
VALIDATION
SDLC
STLC
SDLC, STLC Overlapped

Levels of Software Testing

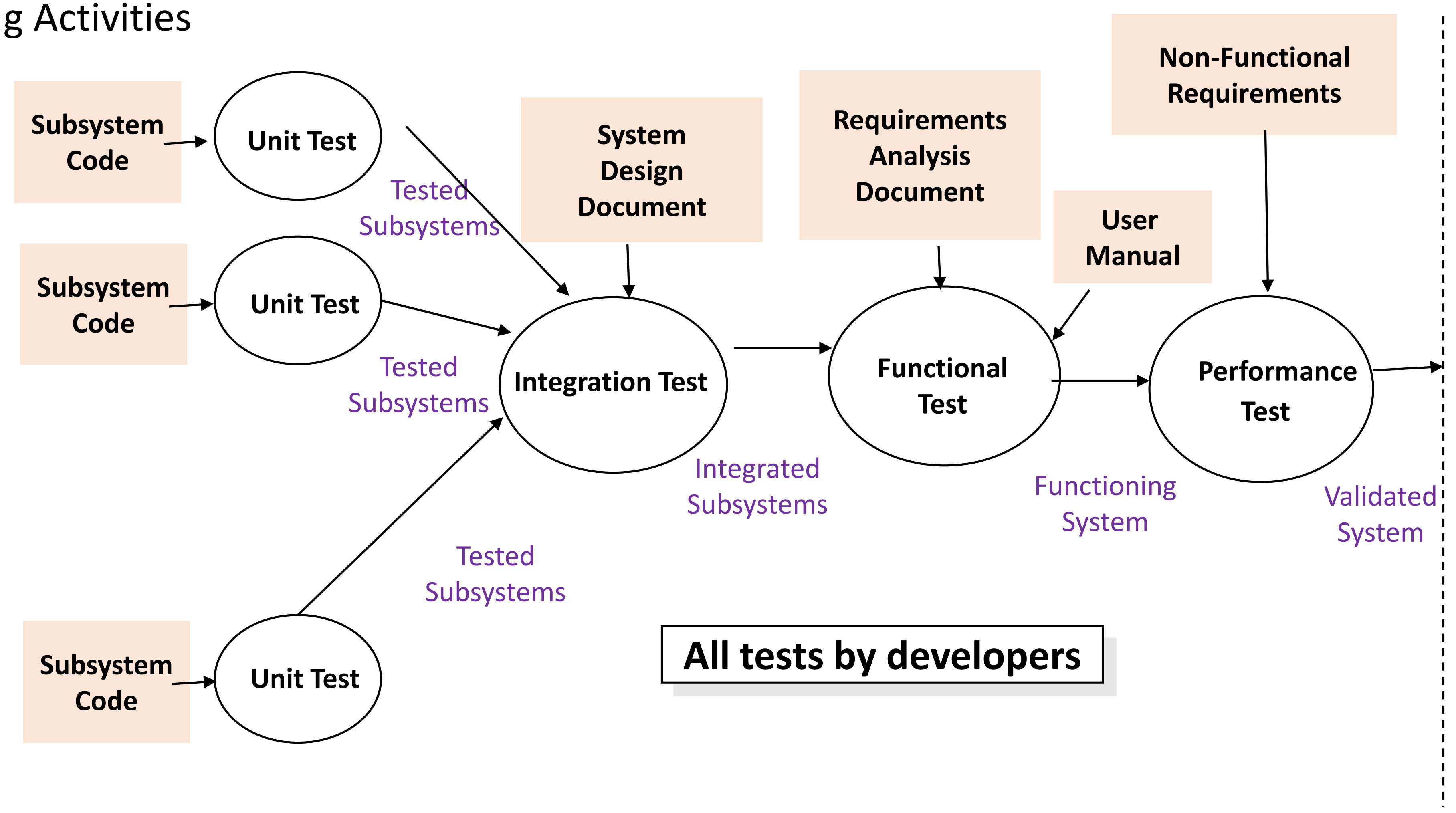


- Testing usually begins with ***functional*** (***black-box***) tests,
- supplemented by ***structural*** (***white-box***) tests,
    - working from the unit level toward the system level with one or more integration steps.

(Hamilton, 2023 May 13)

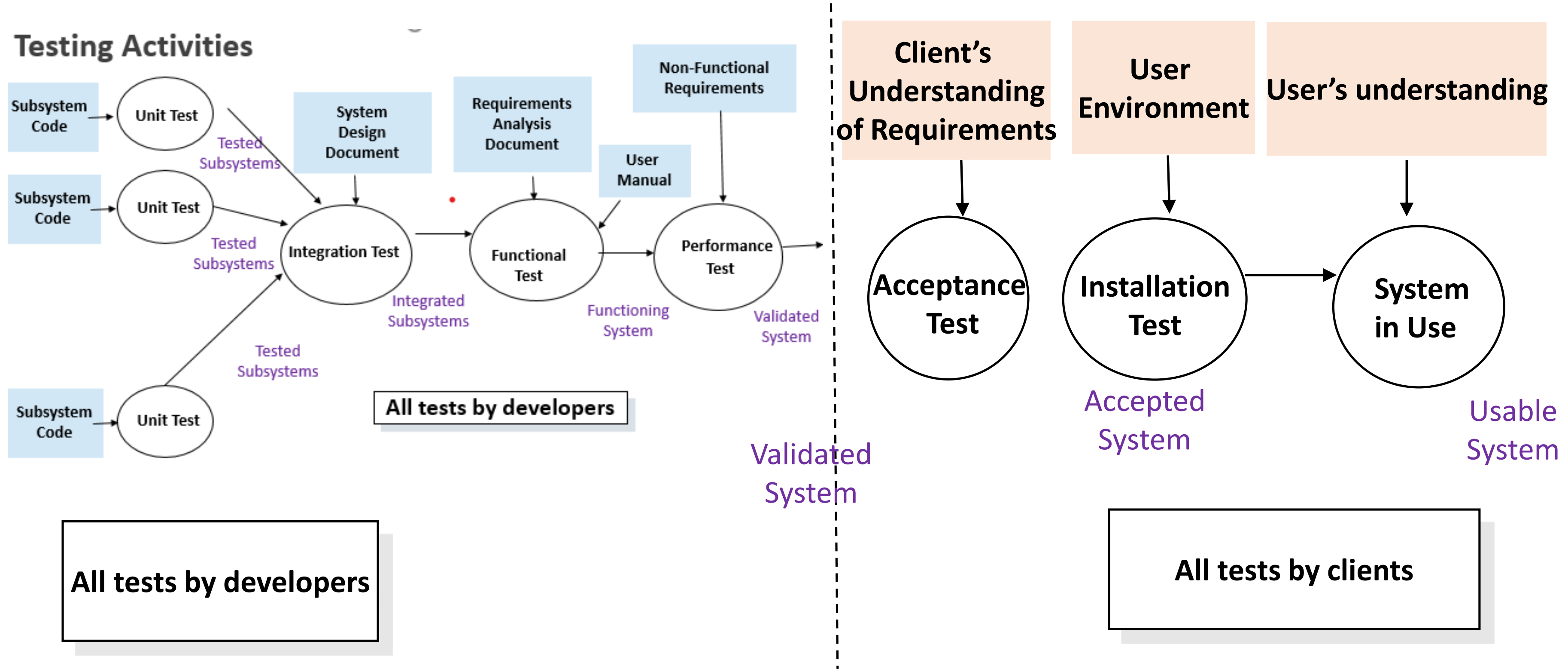
Levels of Software Testing

- Testing Activities



Levels of Software Testing

- Testing Activities



Levels of Testing: Unit

- **Unit testing:**
- A Unit is a smallest testable portion of system or application which can be compiled, liked, loaded, and executed. This kind of testing helps to test each module separately.
- The aim is to test each part of the software by separating it. It checks that component are fulfilling functionalities or not. This kind of testing is performed by developers.

Look for errors in Individual subsystem or object.

Carried out by developers.

Black Box Testing: check outputs, tests based on use cases, uses developer knowledge of functionality.

White Box Testing: check logical pathways through code including decision points, loops and boundary values.

Goal: Confirm that subsystems is correctly coded and carries out the intended functionality.

(Hamilton, 2023 May 13)

Levels of Testing: Unit example

- Types of Testing: Unit Testing Test Case

Test Case 1: Deposit Money

Input: BankAccount account = new BankAccount(100);, double amount = 50;
Expected Output: account.getBalance() returns 150.0
Test Code: assertEquals(150, account.getBalance());
Test Method: testDepositMoney()

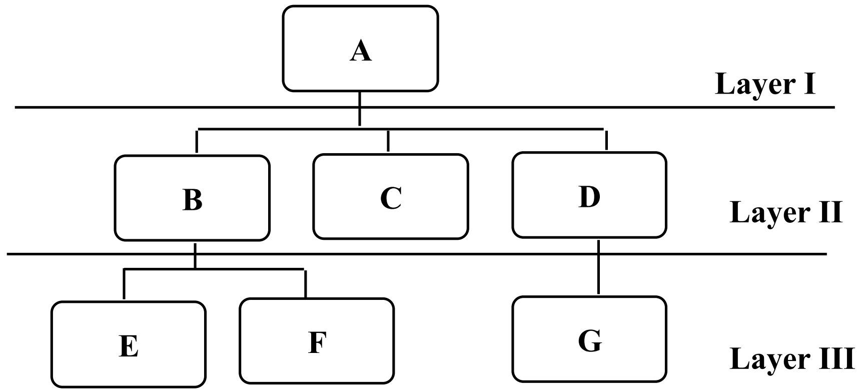
Levels of Testing: Unit example

- Types of Testing: Unit Testing

```
import org.junit.Test;
import static org.junit.Assert.assertEquals;

public class   MyTest {

    @Test public void testAddition() {
        Addition  addition = new Addtion ();
        int result =   addtion.calculate(2, 3);
        assertEquals(5, result);
    }
}
```

Levels of Testing: Integration

- **Integration testing:**
- Integration means combining. For Example, In this testing phase, different software modules are combined and tested as a group to make sure that integrated system is ready for system testing.
- Integrating testing checks the data flow from one module to other modules. This kind of testing is performed by testers.
    - Find errors with connecting subsystems together.
    - Carried out by developers.
    - May employ a top down and bottom up approach.
    - May need to write stubs.
    - **Goal:** Test the interface among the subsystem.



(Hamilton, 2023 May 13)

Levels of Testing: Integration Example

- Types of Testing: Integration Calculator.java
- Test Case: Verify that the BankingSystem class can deposit and withdraw funds correctly from the BankAccount class.

- Preconditions:
  - The BankAccount class has been initialized with a balance of 100.
  - The BankingSystem class has been initialized with the BankAccount object.
- Steps:
  - Deposit 50 into the account using the BankingSystem class.
  - Verify that the balance of the BankAccount class has increased by 50.
  - Withdraw 20 from the account using the BankingSystem class.
  - Verify that the balance of the BankAccount class has decreased by 20.
  - Verify that the balance of the BankAccount class is still greater than 0.
- Expected Results:
  - The balance of the BankAccount class is 150 after depositing 50.
  - The balance of the BankAccount class is 80 after withdrawing 20.
  - The balance of the BankAccount class is still greater than 0.

Levels of Testing: Integration Example

Types of Testing: Integration Calculator.java

```
public class Calculator {
    public int add(int a, int b) {
        return a + b;
    }
}

public class Logger {
    public void log(String message) {
        System.out.println(message);
    }
}
```

Levels of Testing: Integration Example

- Types of Testing: Types of Testing: Integration Calculator.java

```
public class CalculatorIntegrationTest {
  @Test
  public void testAddition() {
    Calculator calculator = new Calculator();
    Logger logger = new Logger();
    // Call the calculator method and log the result
    int result = calculator.add(2, 3);
    logger.log("Result: " + result);
    // Verify that the result was logged correctly
    assertEquals("Result: 5", logger.getLog());
  }
}
```

Levels of Testing: Integration Example

- Types of Testing: Integration Calculator.java

  - Create instances of the Calculator and Logger classes, and then call the add method on the Calculator instance to perform an addition operation.
  - Log the result to the console using the Logger instance.
  - Verify that the result was logged correctly by checking the output of the Logger instance using the assertEquals method.
  - "Result: 5", which is the result that is expected of the addition operation.

Levels of Testing: System

- **System Testing**
- System testing is performed on a complete, integrated system. It allows checking system's compliance as per the requirements. It tests the overall interaction of components. It involves load, performance, reliability and security testing.
- System testing most often the final test to verify that the system meets the specification. It evaluates both functional and non-functional need for the testing.

- Test entire system behavior as a whole, with respect to scenarios and requirements (treat the system as a black box).
- Carried out by developers

- Functional testing for all system functions

- Non functional testing – performance testing.

- **Goal:** Determine if the system meets both functional and non-functional requirements.

(Hamilton, 2023 May 13)

Levels of Testing: System

- Types of Testing: System Testing
- This is a little more complicated but in general we are testing that the system transforms inputs/outputs or system functionality as required.
- What can we test for the Fire Management System?

Levels of Testing: System Example

Types of Testing: System Testing

- Test Case 1: Login and Logout
- Precondition: User is not logged in
  - Step 1: Enter valid username and password
  - Step 2: Verify successful login
  - Step 3: Click on logout button
  - Step 4: Verify successful logout
- Expected Result: User is logged out and cannot access the system

- Test Case 2: Deposit and Withdrawal
- Precondition: User has a valid account
  - Step 1: Enter valid deposit amount
  - Step 2: Verify successful deposit
  - Step 3: Enter valid withdrawal amount
  - Step 4: Verify successful withdrawal
- Expected Result: User's account balance is updated correctly

Levels of Testing: Acceptance

- **Acceptance testing:**
- Acceptance testing is a test conducted to find if the requirements of a specification or contract are met as per its delivery. Acceptance testing is basically done by the user or customer. However, other stockholders can be involved in this process. This is blackbox testing
- Evaluates the system delivered by developers
- Carried out by the client. Executing typical transactions in the lab with developer present (Alpha testing) or on site without developer (Beta testing)
- Regression testing carried out on site as part of post delivery maintenance
- **Goal:** Demonstrate that the system meets customer requirements and is ready to use

(Hamilton, 2023 May 13)

Levels of Testing: Acceptance

- Types of Testing: Acceptance Testing

- Test Case 1: Deposit and Withdrawal
- Test Case ID: 1.1
- Description: The system should allow users to deposit and withdraw funds.
- Steps:

- Login to the system with a valid user account.
  - Select the "Deposit" option.
  - Enter the amount to deposit (e.g. $100).
  - Click the "Submit" button.
  - Verify that the account balance is updated to reflect the deposit.
  - Select the "Withdraw" option.
  - Enter the amount to withdraw (e.g. $50).
  - Click the "Submit" button.
  - Verify that the account balance is updated to reflect the withdrawal.

Static Testing and Dynamic Testing

- **Static testing** is an important testing technique that takes the form of Business requirement review, Functional requirement review, design reviews, code walkthroughs and test documentation review. It is a continuous activity and not done just by testers.

- **Validation**, the dynamic testing part is more hands-on and happens on the product itself and not on an artifact or a representation of the product. A much formal process of test case/condition identification, coverage considerations, execution and defect reporting all mark the dynamic testing methods.

- **An example:** Say this is the Business Requirements Document (BRD) for a banking site that is big on security. There is a section in the BRD that talks about the password rules for the various users creating an account with the online banking site. One of the rules is: **A user cannot use a password that he/she uses for other accounts.** This is not do-able.

- Because, a site can merely suggest how the user should set login credentials but there is no way, this restriction can be imposed. So, this requirement is not feasible – in other words, cannot be accomplished through the software.

Software Testing Help (2019, 2 July)

Unit Testing

Please watch this video:

Title: What is Unit Testing? Why YOU Should Learn It + Easy to Understand Examples
Link: https://www.youtube.com/watch?v=3kzHmaeozDI
Duration: 10:42 minutes

Code Example at 4mins – 9mins

(Sterkowitz, 2018)

Black & White Box Testing

(AnimatedVideoAgency, 2012)

Please watch this short video:

(is dated but covers basics)

Title: Software Testing Tutorial - Animated Video **Duration:** 1:53 minutes
Link: https://www.youtube.com/watch?v=Y7Wg4508tHo

Black & White Box Testing

Testing for expected results (outputs) to specific inputs is referred to as ***Black-Box*** testing

- We are not concerned with the internal description of a module, only its external reactions to externally stimuli
- ***Functional*** Testing

Testing the internal logic and working of a module is referred to as ***White-Box*** testing

- ***Structural*** Testing

Black & White Box Testing

Black box testing attempts to derive sets of inputs that will fully exercise all the ***functional requirements*** of a system.

It is **not** an alternative to white box testing.

It attempts to find errors in the following categories:

- incorrect or missing functions,
- interface errors,
- errors in data structures or external database access,
- performance errors, and
- initialisation and termination errors.

White Box Testing

- Testing based on an analysis of internal logic (design, code, etc.). Also know as ***structural testing***.
- The *expected* results still come from requirements.
- White-box testing techniques apply *primarily* to lower levels of testing (e.g., unit and component).
- White box testing should be performed early in the testing process, while black box testing tends to be applied during later stages.

White Box Testing

Example: “Logic Coverage”

- ***Statements***: each statement is executed at least once
- ***Branches***: each branch is traversed (and every entry point taken) at least once
- ***Conditions***: each condition True at least once and False at least once
- ***Compound Conditions***: all <u>***combinations***</u> of condition values at every branch statement are covered (and every entry point taken)
- ***Path***: all program paths are traversed at least once

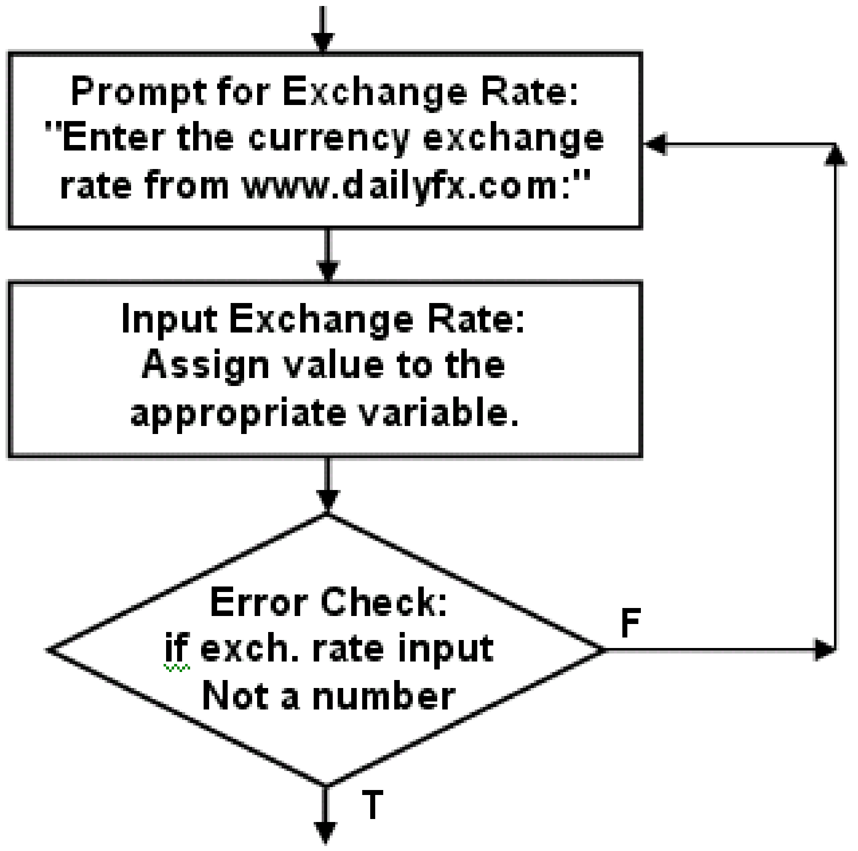
White Box Testing

Example: “Logic Coverage”

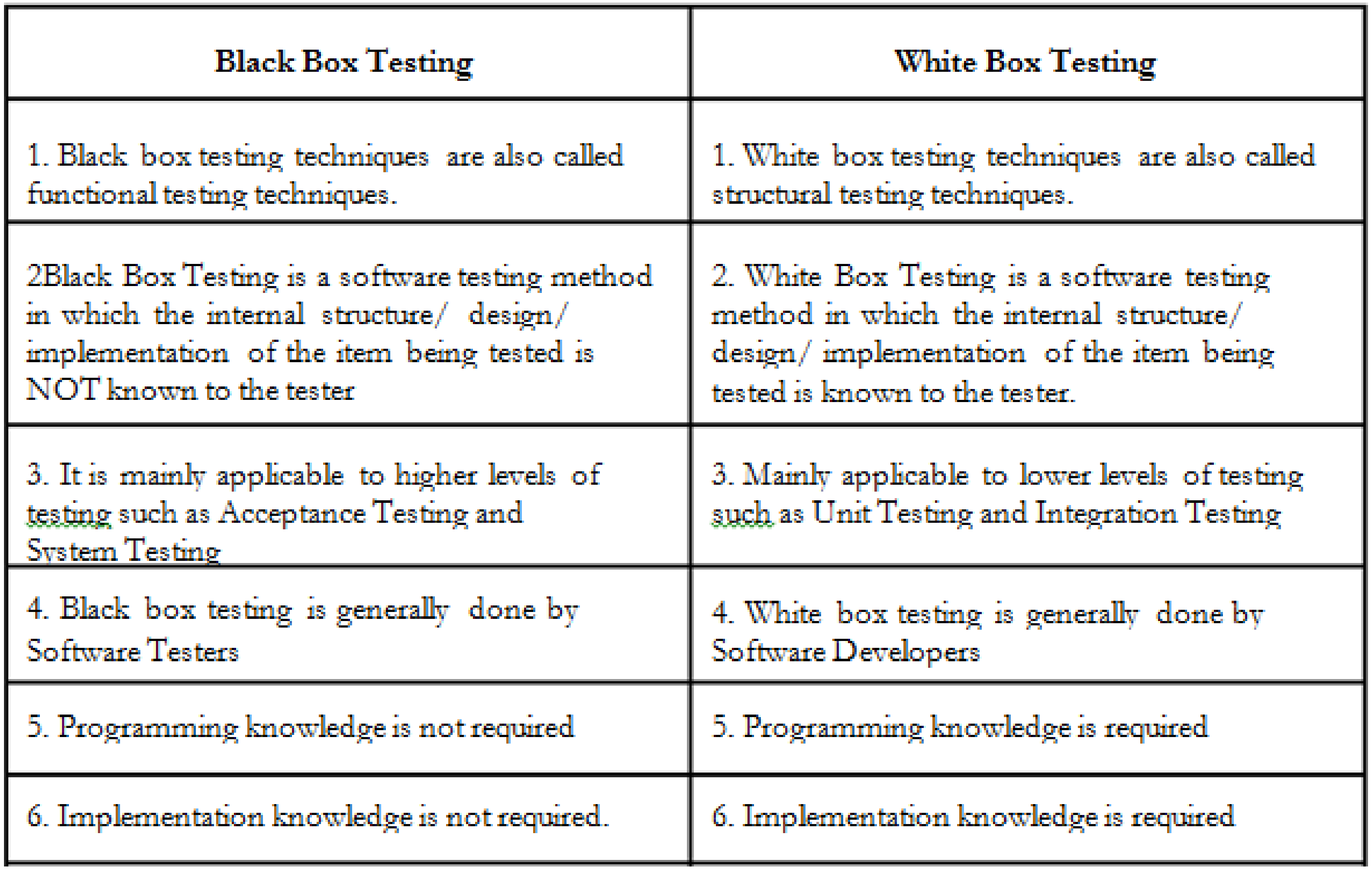
```
FindMean (FILE ScoreFile)
{   float SumOfScores = 0.0;
    int NumberOfScores = 0;
    float Mean=0.0; float Score;
    Read(ScoreFile, Score);
    while (! EOF(ScoreFile) {
        if (Score  > 0.0 ) {
                SumOfScores = SumOfScores + Score;
                NumberOfScores++;
                }

        Read(ScoreFile, Score);
    }
    /* Compute the mean and print the result */
    if (NumberOfScores > 0) {
            Mean = SumOfScores / NumberOfScores;
            printf(" The mean score is %f\n",
            Mean);
    } else
            printf ("No scores found in file\n");
}
```

White Box Testing

Construct a ***Logic Flow*** Diagram

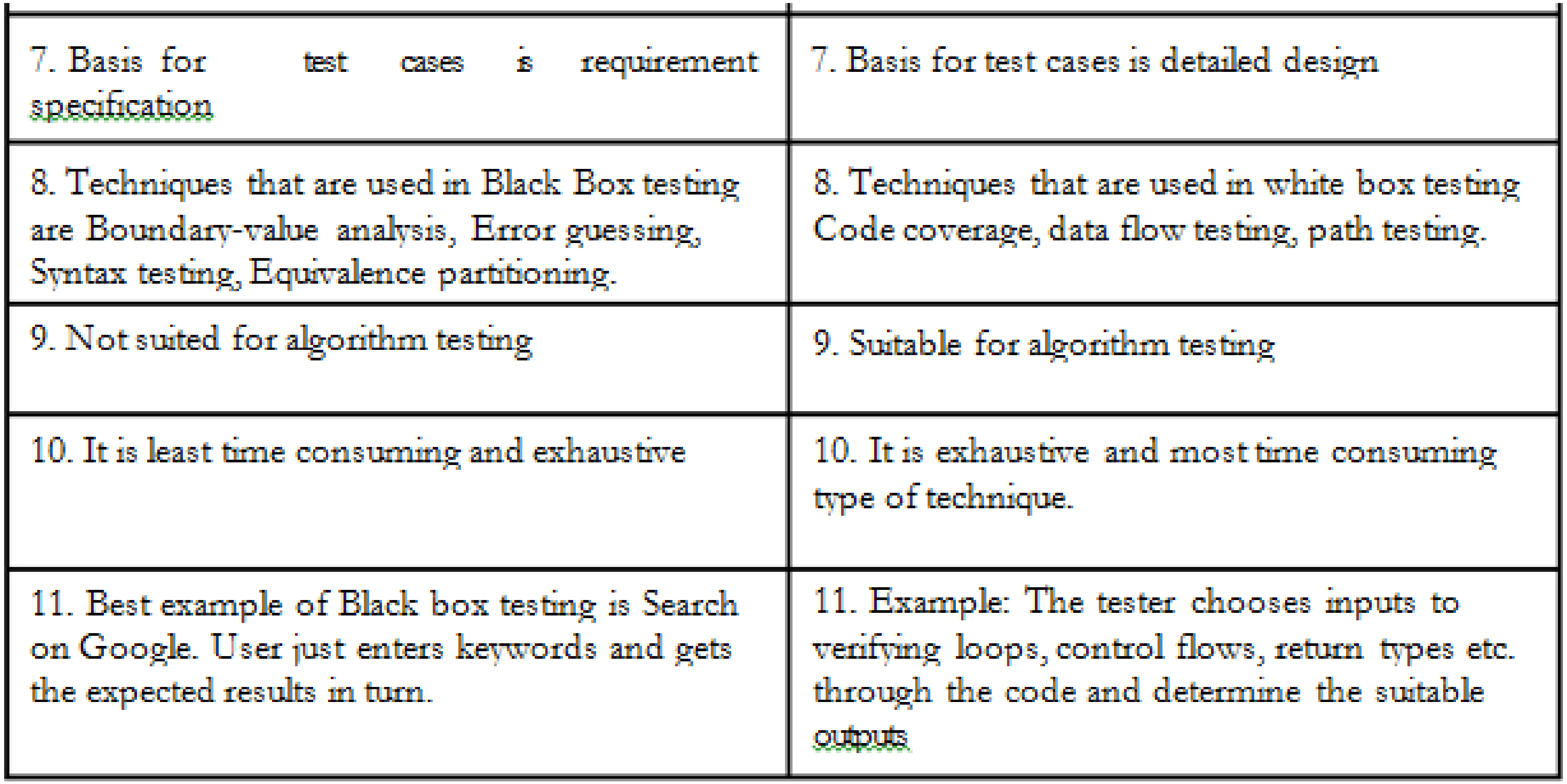

Prompt for Exchange Rate: "Enter the currency exchange rate from www.dailyfx.com:"
Input Exchange Rate: Assign value to the appropriate variable.
Error Check: if exch. rate input Not a number
F
T

http://www.ciese.org/pathways/rwlo/rwlos/2868/Cprogramming%20RWLO/How_to_Create_a_Logic_Flow_Diagram.doc



Black Box Testing	White Box Testing
1. Black box testing techniques are also called functional testing techniques.	1. White box testing techniques are also called structural testing techniques.
2Black Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is NOT known to the tester	2. White Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester.
3. It is mainly applicable to higher levels of testing such as Acceptance Testing and System Testing	3. Mainly applicable to lower levels of testing such as Unit Testing and Integration Testing
4. Black box testing is generally done by Software Testers	4. White box testing is generally done by Software Developers
5. Programming knowledge is not required	5. Programming knowledge is required
6. Implementation knowledge is not required.	6. Implementation knowledge is required

(Ramnath, n.d.)

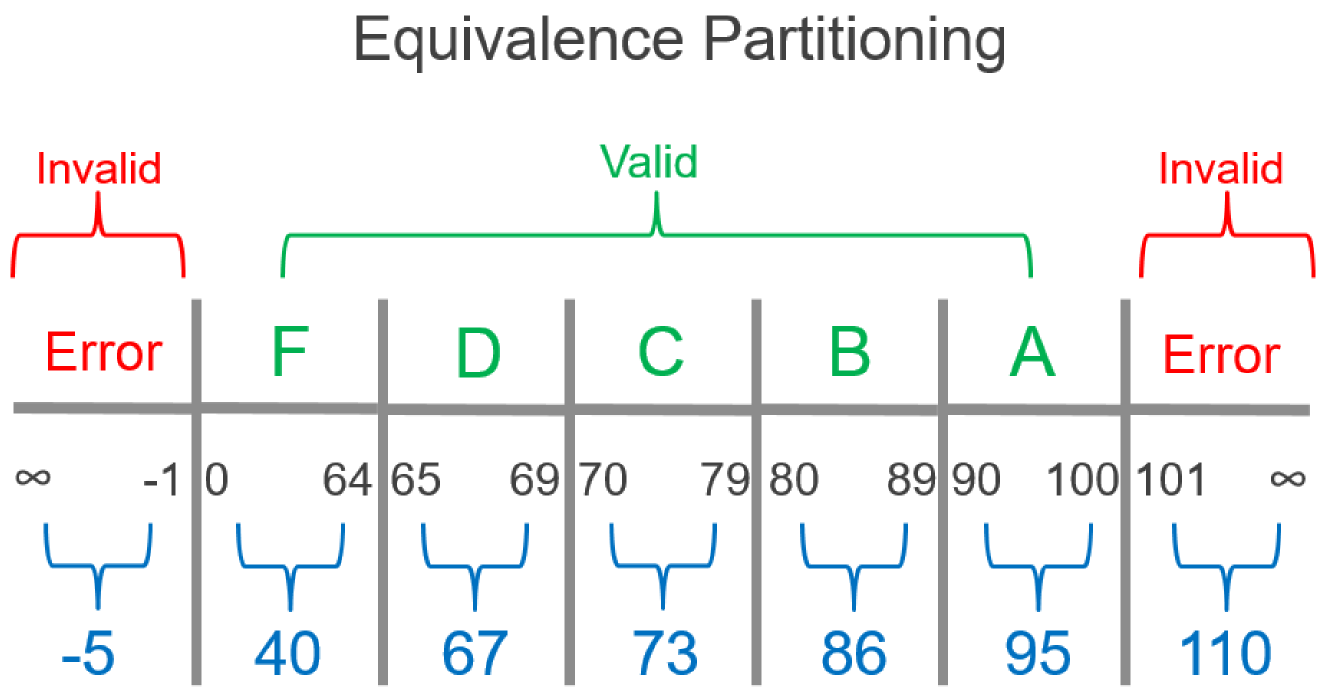


7. Basis for test cases is requirement specification	7. Basis for test cases is detailed design
8. Techniques that are used in Black Box testing are Boundary-value analysis, Error guessing, Syntax testing, Equivalence partitioning.	8. Techniques that are used in white box testing Code coverage, data flow testing, path testing.
9. Not suited for algorithm testing	9. Suitable for algorithm testing
10. It is least time consuming and exhaustive	10. It is exhaustive and most time consuming type of technique.
11. Best example of Black box testing is Search on Google. User just enters keywords and gets the expected results in turn.	11. Example: The tester chooses inputs to verifying loops, control flows, return types etc. through the code and determine the suitable outputs

(Ramnath, n.d.)

Common Techniques

- Partition Testing
- Boundary Value Analysis
- Intuition and Experience



(Derisk QA, 2017)

Partition Testing (blackbox)

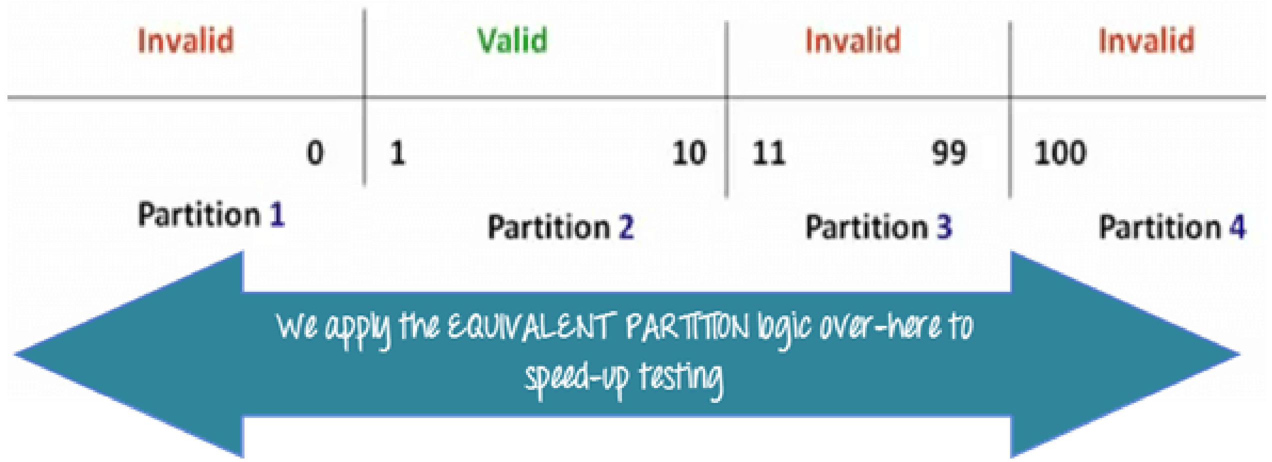
Idea is to *partition* the program's *input space* every element of a given partition would be "handled" (i.e., mapped to an output) in the same manner.

For example, interfaces.

Two types of cases need to be identified:

1. ***valid*** *cases*(corresponding to valid inputs)
2. ***invalid*** *cases*(corresponding to invalid inputs)

If interested, have a look at:
Title: Boundary Value Analysis & Equivalence Partitioning with Examples
Link: https://www.guru99.com/equivalence-partitioning-boundary-value-analysis.html



(Guru99, n.d.)

Boundary Value Analysis (blackbox)

A technique based on identifying, and generating test cases to explore boundary conditions.

Natural language based specifications of boundaries are often ambiguous, as in "for scores between 0 and 40"

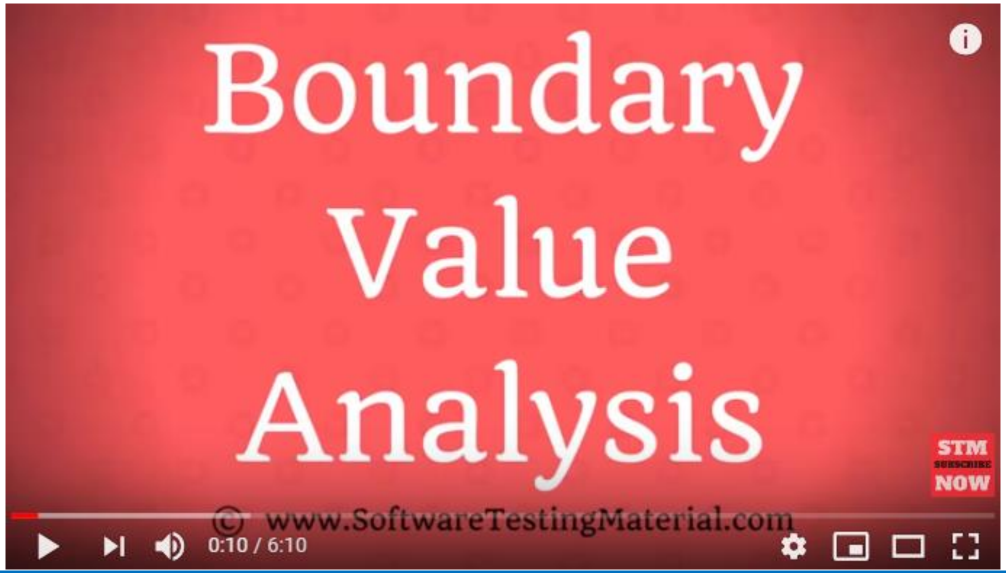
- Ranges: Identify values at the endpoints of the range, just below and just beyond.
- Number of values: a *file will contain 1-25 records*
    - Boundary values: empty file (IV), file with 1 (V), 25 (V), and 26 (IV) records

Please watch this video:

Title: Boundary Value Analysis In Software Testing - Test Design Technique

Duration: 6:10 minutes

Link: https://www.youtube.com/watch?v=21wOCNHsSU4 (Software Testing Material, 2017)



Why Equivalence & Boundary Analysis Testing

- This testing is used to reduce very large number of test cases to manageable chunks.
- Very clear guidelines on determining test cases without compromising on the effectiveness of testing.
- Appropriate for calculation-intensive applications with large number of variables/inputs

Summary:

- Boundary Analysis testing is used when practically it is impossible to test large pool of test cases individually
- Two techniques - Equivalence Partitioning & Boundary Value Analysis testing techniques is used
- In Equivalence Partitioning, first you divide a set of test condition into a partition that can be considered.
- In Boundary Value Analysis you then test boundaries between equivalence partitions
- Appropriate for calculation-intensive applications with variables that represent physical quantities

Intuition and Experience

Also known as *Error Guessing, Ad Hoc Testing,* etc. Testers use intuition and experience to identify potential errors and design test cases to reveal them.

Design tests for reasonable but ***incorrect assumptions*** that may have been made by developers.

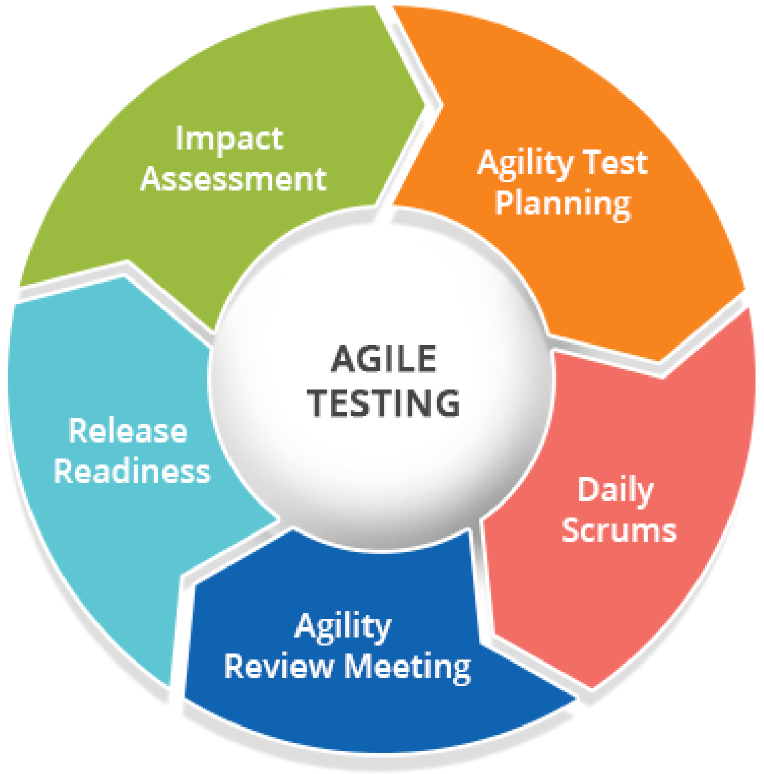
- Blanks or null characters in strings
- Non-numeric values in numeric fields
- Inputs that are too long or too short

Test for adherence to ***Business Rule constraints***

Agile Testing Lifecyle

Impact Assessment
Gather inputs from stake holders and users, this will act as feedback for next deployment cycle.

Approval Meetings for Deployment
At this stage we review the features that have been developed/ implemented are ready to go live or not.



Start- Test Plan Meet Up
All stakeholders come together to plan the schedule of testing process, meeting frequency and deliverables.

Daily Standup Meetings
This includes the everyday standup morning meeting to catch up on the status of testing and set the goals for whole day.

Agility Review Meeting
Weekly review meetings with stakeholders meet to review and assess the progress against milestones.

(ReQtest, 2018)

Agile Testing Methods

Testing Methods:

- Behaviour Driven Development (BDD)
- Acceptance Test Driven Development (ATDD)
- Exploratory Testing

(ReQtest, 2018)

Behaviour Driven Development (BDD)

- BDD improves communication amongst project stakeholders so that all members correctly understand each feature before the development process starts.

- There is continuous example-based communication between developers, testers, and business analysts. The examples are called Scenarios which are written in a syntax called Gherkin Given/When/Then.

- The **Given-When-Then** formula is a template intended to guide the writing of acceptance tests for a User Story:
  - (Given) some context
  - (When) some action is carried out
  - (Then) a particular set of observable consequences should obtain

An example:

  - Given my bank account is in credit, and I made no withdrawals recently,
  - When I attempt to withdraw an amount less than my card's limit,
  - Then the withdrawal should complete without errors or warnings

(ReQtest, 2018)

- https://www.agilealliance.org/glossary/gwt/

Acceptance Test Driven Development (ATDD)

- ATDD focuses on involving team members with different perspectives such as the customer, developer, and tester.
- **Three Amigos** meetings are held to formulate acceptance tests incorporating perspectives of the customer, development, and testing.
- The customer is focused on the problem that is to be solved, the development is focused on how the problem will be solved whereas the testing is focused on what could go wrong.
- The acceptance tests are a representation of the user's point of view and it describes how the system will function.
- It also helps to verify that the system functions as it is supposed to. In some instances acceptance tests are automated.

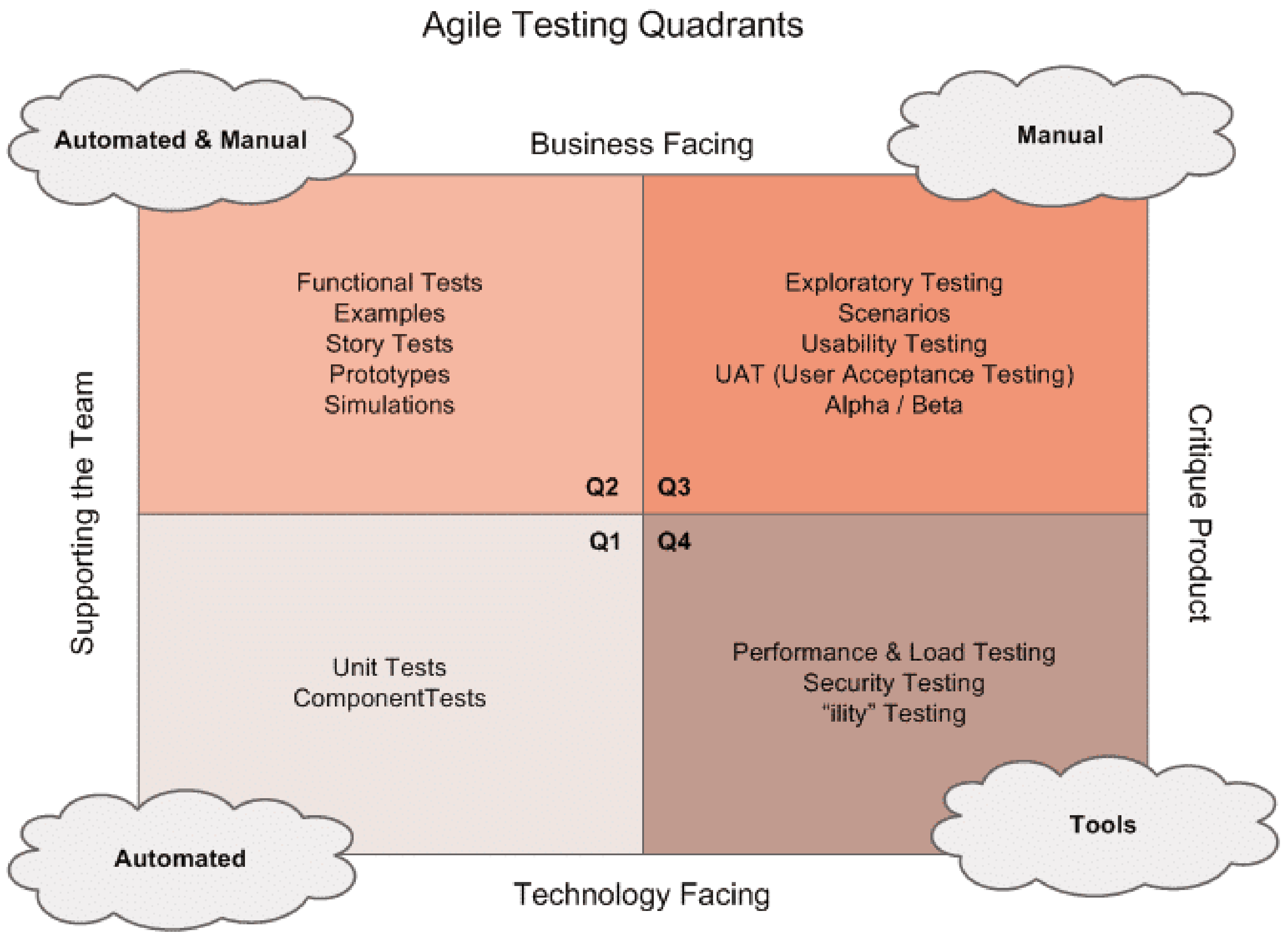
(ReQtest, 2018)

Exploratory Testing

- In this type of testing, the test design and test execution phase go hand in hand.
- Exploratory testing emphasizes working software over comprehensive documentation. The individuals and interactions are more important than the process and tools.
- Customer collaboration holds greater value than contract negotiation.
- Exploratory testing is more adaptable to changes. In this testers identify the functionality of an application by exploring the application.
- The testers try to learn the application, and design & execute the test plans according to their findings.

(ReQtest, 2018)

Agile Test Strategy



(Ghahrai, 2018)

Agile Testing Advantages

Advantages:

- Cost and time efficiencies
- Reduces documentation
- Flexible and highly adaptable to changes
- Improved quality and
- Improved accuracy
- More powerful
- It provides a way for receiving regular feedback from the end user
- Better collaboration
- Better determination of issues through daily meetings

(Ishwaran, 2016; ReQtest, 2018; Gordan, 2017)

Regression Testing

Please watch this video

Title: What is Regression Testing? A Software Testing FAQ - Why? How? When?

Duration: 11:14 minutes

Link: https://www.youtube.com/watch?v=xmQuLTarGI4

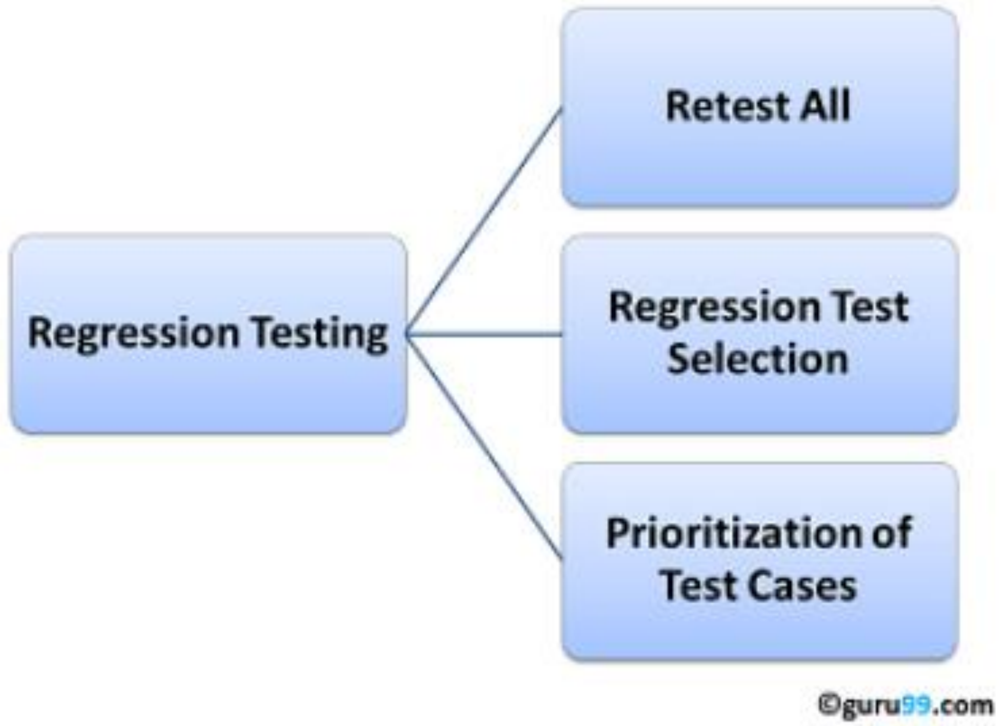
(EvilTester - Software Testing, 2017)

Regression Testing

Regression Testing: is a type of software testing to confirm that a recent program or code change has not adversely affected existing features.

Regression Testing is required when there is a need for:

- Change in requirements and code is modified according to the requirement
- A new feature is added to the software
- Defect fixing
- Performance issue fix



(Guru99, n.d.)

Why Regression Testing Is Important

- It is an essential part of the quality process and ensures that code changes do not hurt the existing functionality.
- Testing teams should follow '**Test after change and Test often**' approach which will provide the results that inform whether new code is compatible with the existing code base.
- Includes executing an increasing set of tests along with covering existing functionality until the product is done.
- Continuous regression testing help teams build software that behaves as intended and remains stable.

(Kumar, 2016)

Integration Testing

Integration test

- Evaluates the ***behaviour*** of a ***group*** of methods or classes
- Identifies interface compatibility, unexpected parameter values or state interaction, and run-time exceptions

System test

- Integration test of the ***behaviour*** of an **entire system** or an independent subsystem

Build and Smoke test

- A *System Test* performed daily or several times a week

2 UNIT TESTS, 0 INTEGRATION TESTS
via reddit.com/r/programmerhumor

(Software Testing Fundamentals, n.d.)

Usability Testing

Determines whether a method, class, subsystem, or system meets **user requirements**

Performance test is an example

- Determines whether a system or subsystem can meet time-based performance criteria
  - ***Response time*** specifies the desired or maximum allowable time limit for software responses to queries and updates
  - ***Throughput*** specifies the desired or minimum number of queries and transactions that must be processed per minute or hour

User Acceptance Testing (UAT)

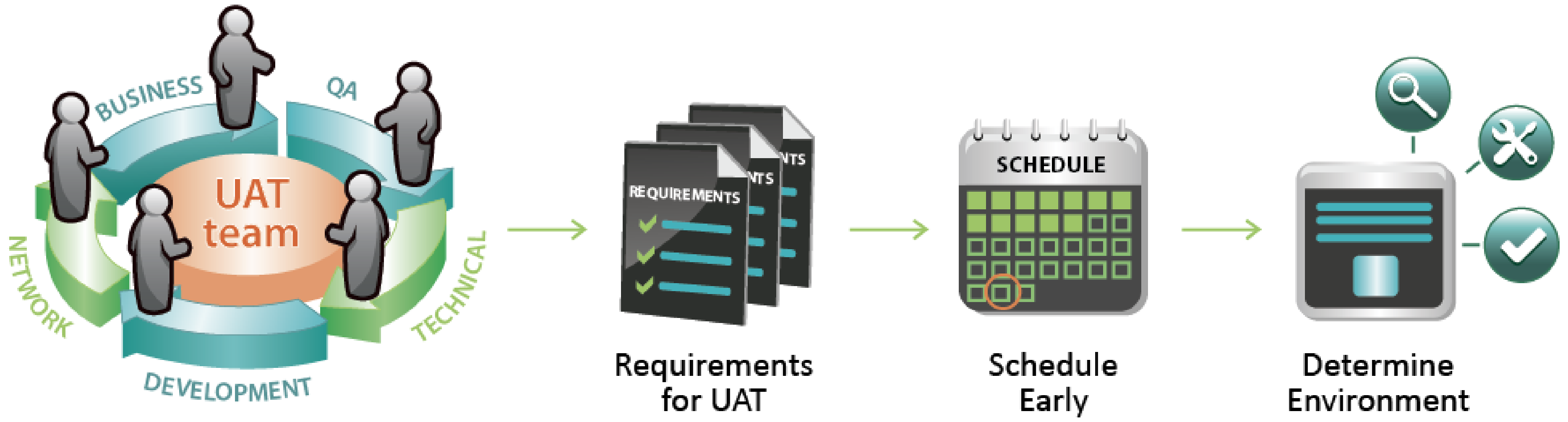
Determines whether the system fulfills the user's requirements

- Involves the ***end users***

System Test Acceptance:

- testing conducted to ensure that a system is "ready" for the system-level test phase

Acceptance testing is a very ***formal*** activity in most development projects



(Zephyr, n.d.)

Other Types of Testing

Alpha:

- actual end-user testing performed within the development environment

Beta:

- end-user testing performed within the user environment prior to general release

Performance Test:

- Focus is on requirements related to through-put, response time, memory utilisation, input/output rates, etc.
- Requirements must be stated in quantifiable terms.

Stress Test:

- Focus is on system behaviour at, or near, overload conditions (i.e., ‘‘pushing the system to failure’’).
- Often undertaken with performance testing.
- In general, products should exhibit ‘‘graceful’’ failures and non-abrupt performance degradation.

Other Types of Testing

Security Test:

- Focus is on the vulnerability of resources to unauthorised access or manipulation.
    - Physical security of computer resources, media, etc.,
    - Login and password procedures/ policies,
    - Levels of authorization for data or procedural access, etc.

Recovery Test:

- Focus is on the ability to recover from exceptional conditions associated with hardware, software, or people.
    - detecting exceptional conditions,
    - switch-over to standby systems,
    - recovery of data,
    - maintaining audit trails, etc.
    - May also involve external procedures such as storing backups, etc.

Who Tests Software?

Programmers

- Unit testing
- Testing buddies can test other's programmer's code
- Tiger teams can try to "break" other programmers software

Users

- Usability and acceptance testing
- Volunteers are frequently used to test beta versions

Quality assurance personnel

- All testing types except unit and acceptance
- Develop test plans and identify needed changes

Test Policy Documents

Purpose:
to represent the testing philosophy of the organisation and provide direction that testers should follow. This will likely cover both new projects, and maintenance for existing systems.

Contents may incorporate elements such as:

- Definition of Testing
- Guidelines for staff
- Description or objectives of the test process
- Risk mitigation component
- Test Evaluation for manual or automatic testing (or both)
- Quality Level to be achieved
- Approach to Test Process Improvement

(Ghahrai, 2018)

Test Policy Documents

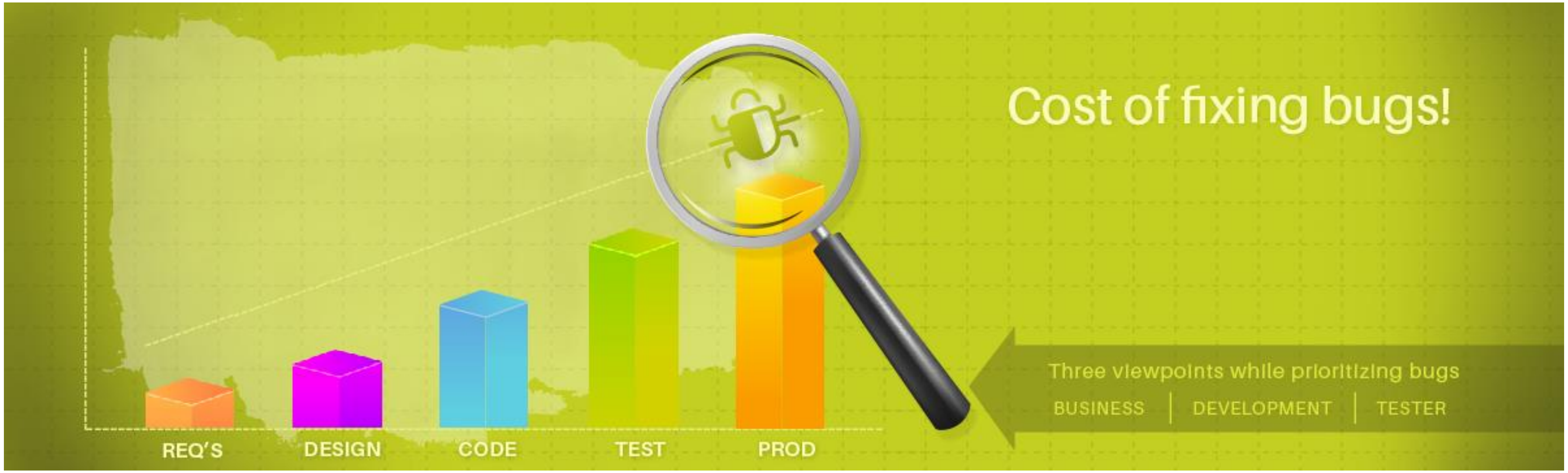
Please have a look at the following extensive example

- University of Liverpool:
  Computing Services Department Testing Policy and Guidelines
- Link:
  https://www.liverpool.ac.uk/media/livacuk/computingservices/regulations/testingpolicyandstrategy.pdf

- **Testing Policy**
  - A high level view of the values that underpin the approach to testing
- **Testing guidelines**
  - Provides guidelines and good practice around how testing should be undertaken, what needs to be considered, and the various test stages that can be adopted

(University of Liverpool, 2017)

Software Bugs



https://reqtest.com/testing-blog/how-to-reduce-the-cost-of-bugs/

Weekly Activity

Aside from the videos and articles used throughout the lecture slides., here are some additional readings for you to consider:

1. **Title:** TEST PLAN: What is, How to Create (with Example)
   Link: https://www.guru99.com/what-everybody-ought-to-know-about-test-planing.html
2. **Title:** How to Write Test Cases
   Link: https://www.guru99.com/test-case.html

   Video Link: https://www.youtube.com/watch?v=BBmA5Qp6Ghk

References

- 360Logica (n.d.) Effective Software Testing Tips – Testers on the Go! [image]. Retrieved 12 July 2024 from https://www.360logica.com/blog/effective-software-testing-tips-testers-go/
- AnimatedVideoAgency (2012, November 27) Software Testing Tutorial-Animated [video] Retrieved 12 July 2024 from https://www.youtube.com/watch?v=Y7Wg4508tHo
- Derisk QA (2017, December 22) Test Design Techniques: Boundary Value Analysis [image]. Retrieved 12 July 2024 from http://www.deriskqa.com/Article-Boundary-Value-Analysis.html
- Evil-Tester, Software Testing (2017, October 20) What is Regression Testing? A Software Testing FAQ - Why? How? When? [video] Retrieved 12 July 2024 from https://www.youtube.com/watch?v=xmQuLTarGI4
- Ghahrai, Amir (2017, July 2) What is a Test Policy Document? Retrieved 6 June 2019 from https://www.testingexcellence.com/test-policy-document/
- Gordan, Alycia (2023, January 1) 6 Benefits of Agile Testing Automation | Software Testing Material. Retrieved 12 July 2024 from https://www.softwaretestingmaterial.com/6-benefits-agile-testing-automation/
- Guru99 (n.d.) What is V Model? Learn with a Case Study using SDLC & STLC. [image] Retrieved 12 July 2024 from https://www.guru99.com/software-testing-lifecycle.html

References

- Guru99 (n.d.) Boundary Value Analysis & Equivalence Partitioning with Examples [image] Retrieved 12 July 2024 from https://www.guru99.com/equivalence-partitioning-boundary-value-analysis.html
- Guru99 (n.d.) What is Regression Testing? Test Cases, Tools & Examples. Retrieved 12 July 2024 from https://www.guru99.com/regression-testing.html
- Hamilton. (2023 May 13). Levels of Testing in Software Testing. Retrieved 12 July 2024 from https://www.guru99.com/levels-of-testing.html
- Ishwaran, Anuradha (2016, June 21) 5 Benefits of Software Testing in an Agile Environment. Retrieved 12 July 2024 from http://www.tothenew.com/blog/5-benefits-of-software-testing-in-an-agile-environment/
- Kumar, Pavan (2016, January 20) The importance of Regression Testing. Retrieved 10 November 2018 from https://www.utest.com/articles/the-importance-of-regression-testing
- Metal Deploye Resistor (n.d) Contact Us [image] Retrieved 8 September 2018 from http://www.mdresistor.com/en/harmonic-filter-resistor/contact-us-email-meta-deploye-resistor-4/

References

- Pexels.com (n.d.) analysis-blackboard-board-bubble [image] Retrieved 12 July 2024 from https://www.pexels.com/photo/analysis-blackboard-board-bubble-355952/
- Ramnath (n.d.) Question: Describe the difference between black box and white box testing with the help of an example. Retrieved 12 July 2024 from http://www.ques10.com/p/2150/describe-the-difference-between-black-box-and-wh-1/
- ReQtest (2018, July 18) Agile Testing – Principles, methods & advantages. Retrieved 12 July 2024 from https://reqtest.com/testing-blog/agile-testing-principles-methods-advantages/
- Satzinger, Jackson and Burd. (2016) *Systems Analysis and Design in a Changing World*, Cengage Learning.
- Software Testing Fundamentals (n.d.) Integration Testing. [image] Retrieved 7 July 2023 from http://softwaretestingfundamentals.com/integration-testing/
- Software Testing Help (2023, June 18) Static Testing and Dynamic Testing – Difference Between These Two Important Testing Techniques. Retrieved 12 July 2024 from https://www.softwaretestinghelp.com/static-testing-and-dynamic-testing-difference/

References

- Soni, Sujit (2015, 27 March) The Logical Engineer – STLC Lifecycle. [image] Retrieved 12 July 2023 from http://automatetesterzone.blogspot.com/2015/
- Sterkowitz, Andy (2018, November 4) Retrieved 12 July 2024 from What is Unit Testing? Why YOU Should Learn It + Easy to Understand Examples [video]. Retrieved 10 November 2018 from https://www.youtube.com/watch?v=3kzHmaeozDI
- University of Liverpool (2017, March 29) Computing Services Department: Testing Policy and Guidelines. Retrieved 10 November 2018 from https://www.liverpool.ac.uk/media/livacuk/computingservices/regulations/testingpolicyandstrategy.pdf
- Zephyr (n.d.) The Science of Running Effective User Acceptance Testing Cycles [image] Retrieved 10 November 2018 from https://www.getzephyr.com/resources/whitepapers/science-running-effective-user-acceptance-testing-cycles

Questions ?

(pexels.com, n.d.)

Thank you