Chapter 7 Software Testing

Lesson Objectives

- Software testing fundamentals
- Testing techniques and strategies
- Test case design and planning
- Testing guidelines/ principles

Software Testing Fundamentals

Introduction

- The quality of a piece of software depends on its design, development, testing, and implementation
- One aspect of software quality is its reliability. A software is reliable if, when used in a reasonable manner, it does not produce failures that are dangerous or costly
- Testing can only demonstrate the presence of errors. It cannot show that there are no errors in a program.
- Testing should be scheduled as part of the project planning process. Adequate resources must be made available for testing.

Introduction

Main Goals of Testing

- To demonstrate to the developer and the customer that the software meets the requirements

  → validation testing

- To discover faults or defects in the software where the behaviour of the software is incorrect, undesirable or does not conform to its specification

  → defect testing

Introduction – Stages in Testing Process

- Generally the stages in testing process are:
  - Unit Testing
  - Module Testing
  - Sub-system Testing
  - System Testing
  - Acceptance Testing (Alpha Testing)

Introduction – Stages in Testing Process
1. Unit Testing

- Individual components are tested to ensure that they operate correctly. E.g. Student Registration component
- The test cases needed for unit testing should exercise each condition and option

Example of Test Case

Program Name: Test Date:			Tester:		
No	**Objective/Test Cases**	**Test Data**	**Expected Results**	**Actual Results**	**Remarks/ Comments**
	To generate a report to list selected month's sales	June to September Sales data	Monthly sales report		

Introduction – Stages in Testing Process
2. Module Testing

- A module is a collection of dependent components such as an object class, an abstract data type or some looser collection of procedures and functions

Introduction – Stages in Testing Process
3. Sub-System Testing

- This phase involves testing collections of modules, which have been integrated into sub-systems

Introduction – Stages in Testing Process
4. System Testing

- System testing tests the integration of each sub-system in the entire system
- It also test to find discrepancies between the system and its original objective, current specifications, and systems documentation
- The primary concern is the compatibility of individual modules.

Introduction – Stages in Testing Process
5. Acceptance Testing

- Final stage in the testing process before the system is accepted for operational use
- Also called **user acceptance test (UAT)** or **alpha testing**
- Conducted by users

Introduction – Beta Testing

- When a system is to be marketed as a software product, a testing process called beta testing is often used.
- Beta testing involves delivering a system to a number of potential customers who agree to use that system.
- This expose the product to the real use & detects errors that may not be anticipated by the developers

Introduction – Stages in Testing Process

- Generally the stages in testing process are:
  - Unit Testing
  - Module Testing
  - Sub-system Testing
  - System Testing
  - Acceptance Testing (Alpha Testing)

Introduction – Test Data

- 2 different sources of test data: -
  - live data
  - artificial data

Introduction
Test Data – Live Data

- Live test data are those that are actually extracted from organization files.
- After a system is partially constructed, programmers or analysts often ask users to key in a set of data from their normal activities
- However, it is difficult to obtain live data in sufficient amounts to conduct extensive testing

Introduction
Test Data – Artificial Data

- Artificial test data are created solely for test purposes, since they can be generated to test all combinations of formats and values
- These data can be quickly prepared by a data-generating utility program
- The most effective test program use artificial test data generated by persons other than those who wrote the programs

Introduction – Test Library

- To assure that all systems are properly tested, many organizations establish test libraries.
- A testing library is a set of data developed to thoroughly test a system of programs.

Testing Techniques and Strategies

Introduction – Stages in Testing Process

- Generally the stages in testing process are:
  - Unit Testing
  - Module Testing
  - Sub-system Testing
  - System Testing
  - Acceptance Testing (Alpha Testing)

Testing Techniques

- As discussed earlier, two fundamental testing activities are:
- i) Component testing/Unit testing
  - based on understanding on how the components should operate and testing will perform by software developer
- ii) System testing
  - Basically involve 2 phases i.e. integration testing & release testing.
  - Based on a written system specification and testing will perform by independent testing team

Testing Techniques

System testing - Integration testing phase:

1. **Top-down testing**
   - ✓ As the name implies, begins with the upper-level modules and works downwards
2. **Bottom-up testing**
   - ✓ Where testing starts with the fundamental components and works upwards
3. **Regression testing**
   - ✓ Rerun the tests for previous increments when a new increment is integrated
   - ✓ If problems, should check whether these problems in previous increment that the new increment has exposed or whether these are due to the added increment of functionality
   - ✓ Expensive

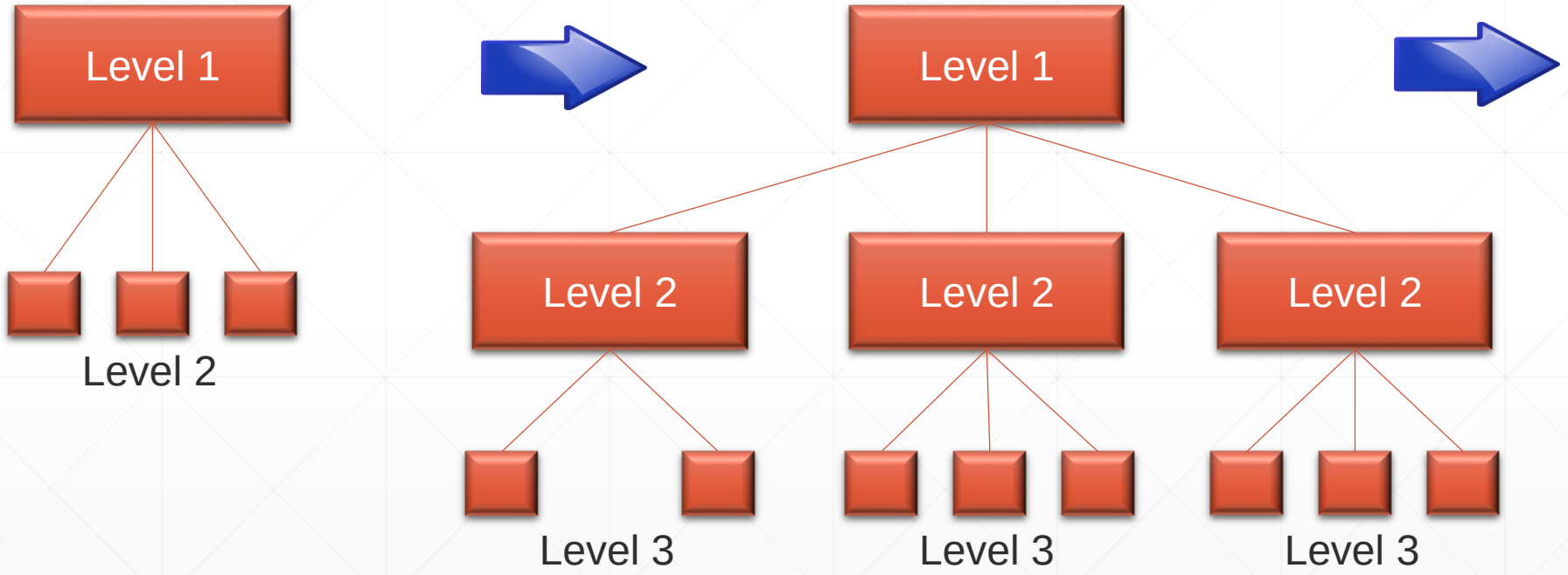
Testing Techniques – Integration Testing
1. Top Down Testing

System testing

Start from upper modules & works downwards

Integration Testing:
Top Down Testing

Testing Techniques – Integration Testing
1. Top Down Testing



Integration Testing: **Top Down Testing**

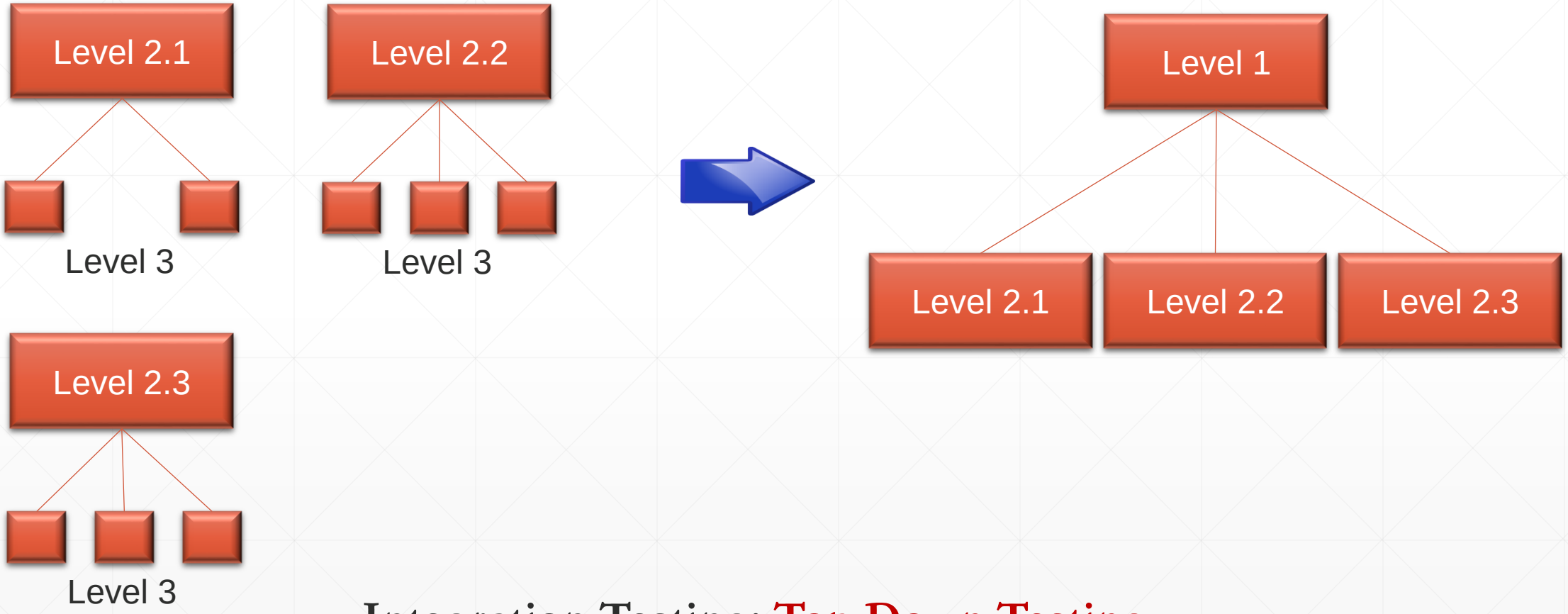
Testing Techniques – Integration Testing
2. Bottom Up Testing

System testing

Start from fundamental modules & work upwards

Integration Testing:
Bottom Up Testing

Testing Techniques – Integration Testing
2. Bottom Up Testing



Integration Testing: **Top Down Testing**

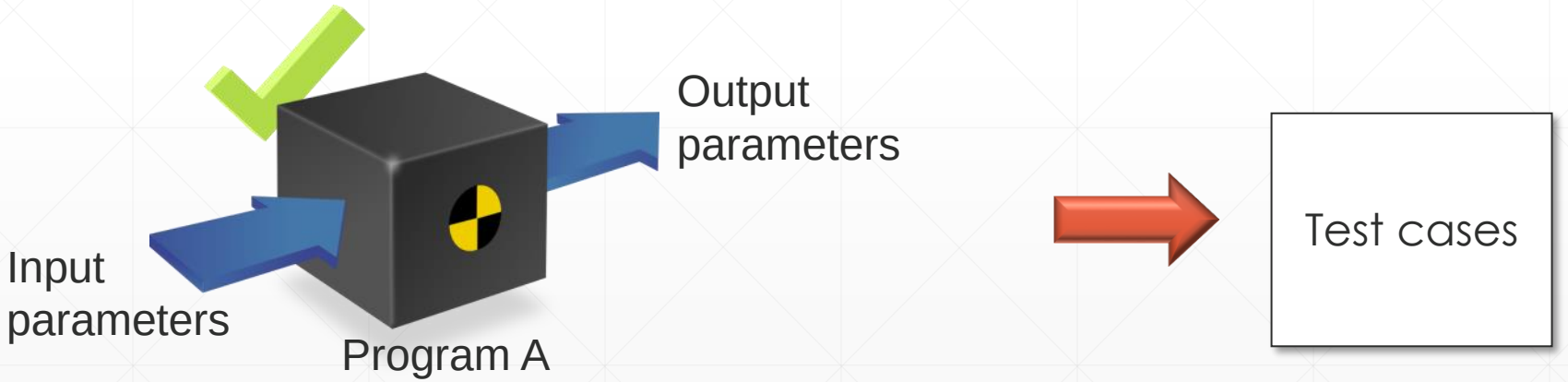
Testing Techniques – Integration Testing
3. Regression Testing

Integration Testing:
Regression Testing

- System testing
- Rerun test for previous increments when new increment is integrated
- Expensive

Testing Techniques – Release Testing
1. Functional/Black Box Testing

- **Functional or black-box testing**
  - relies on the specification of the system or component, which is being tested to derive test cases.
  - the system behaviour can only be determined by studying its inputs and the related outputs.



Testing Techniques – Release/Integration Testing
2. White Box Testing

Structural or white-box testing

- analyze the code and use knowledge about the structure of a component to derive test data.
- the analysis of the code can be used to find many test cases to guarantee a given level of test coverage.

```
Void ....
{
 ..............;
 ..............;
}
```

Test cases

Testing Techniques – Performance/Stress Test

- **Performance/Stress testing:**
  - Once a system has been completely integrated, test the system to ensure it can process the intended load
  - stressing the system by going beyond its specified limits and hence testing how well the system can cope with over-load situation.
  - E.g. in a banking system, analysts want to know what will happen if all tellers sign on at their terminals at the same time before the start of the business day, will the system able to handle this situation?

Testing Techniques

- In component testing/unit testing, different types of components can be tested:

  i. Individual functions/methods in an object

  ii. Object classes that have several attributes/methods

  iii. Composite components that made up of several different objects/functions

Testing Techniques – Component/Unit Testing

i. To test Individual functions/methods in an object

- Simplest type of component
- Your tests are a set of calls to these routines with different inputs parameters
- Can use the approaches to test case design, in next section, to design the tests

Testing Techniques – Component/Unit Testing

ii. To test Object classes that have several attributes/methods

- When testing object classes, should design tests to cover all features in an object
- The testing in isolation of all operations associated with the object
- The setting and interrogation of all attributes with the object
- The exercise of the object in all possible states

Testing Techniques – Component/Unit Testing

iii. To test Composite components that made up of several different objects/functions

- When testing composite components, the primary concern is to test that the component interface behaves according to its specification
- Interface errors cannot be detected in individual objects/components testing.
- Errors may arise due to interactions between its parts

Testing Techniques

To test Composite components that made up of several different objects/functions

- **Interface Testing** - There are different types of interface between program components and, consequently, different types of interface error that can occur:-
  - parameter interface
  - shared memory interfaces
  - procedural interfaces
  - message passing interfaces

Testing Techniques – Interface Testing

Data passed from one method to another

Parameter interface

Unit Testing

Composite Component: Interface Testing

Testing Techniques – Interface Testing

Block of memory is shared between procedures/ functions

Parameter interface

Shared memory interface

Unit Testing

Composite Component: Interface Testing

Testing Techniques – Interface Testing

Parameter interface
Shared memory interface
Procedural interface

Sub-system encapsulates a set of procedures to be called by other sub-systems

Unit Testing

Composite Component: Interface Testing

Testing Techniques – Interface Testing

Parameter interface
Shared memory interface
Procedural interface
Message passing interface

Sub-systems request services from other sub-systems

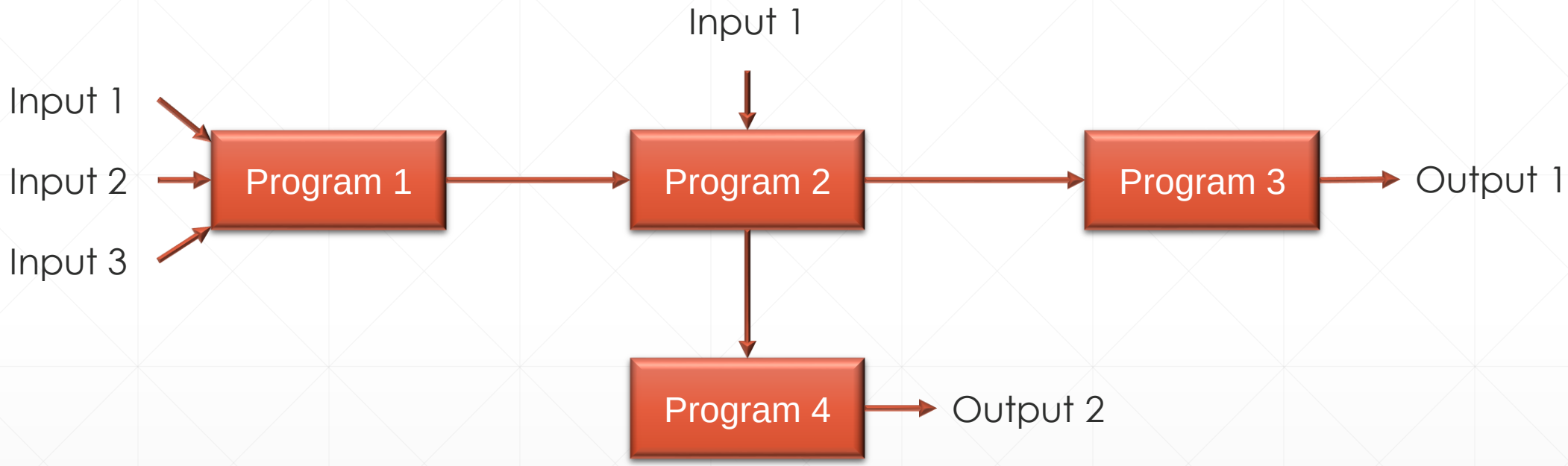
Unit Testing

Composite Component: Interface Testing

Testing Techniques - Others

- **Thread testing / "Transaction flow" testing**
  - Which is used for systems with multiple processes where the processing of a transaction threads its way through these processes
  - Event-based approach i.e. tests are based on the events trigger system actions.
  - Involves identifying and executing each possible processing threads

Testing Techniques - Others

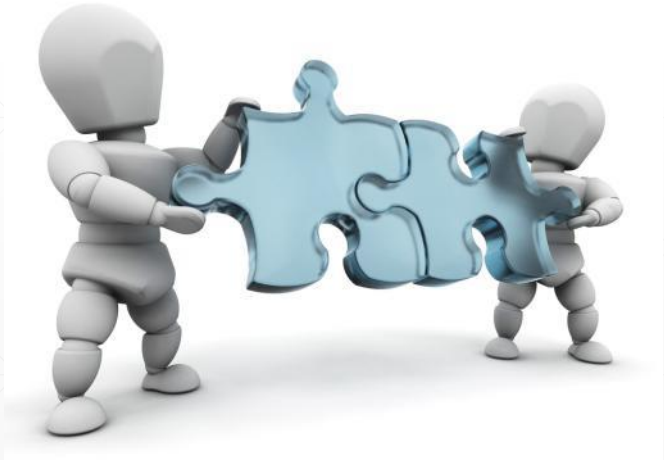


Multiple Thread Testing

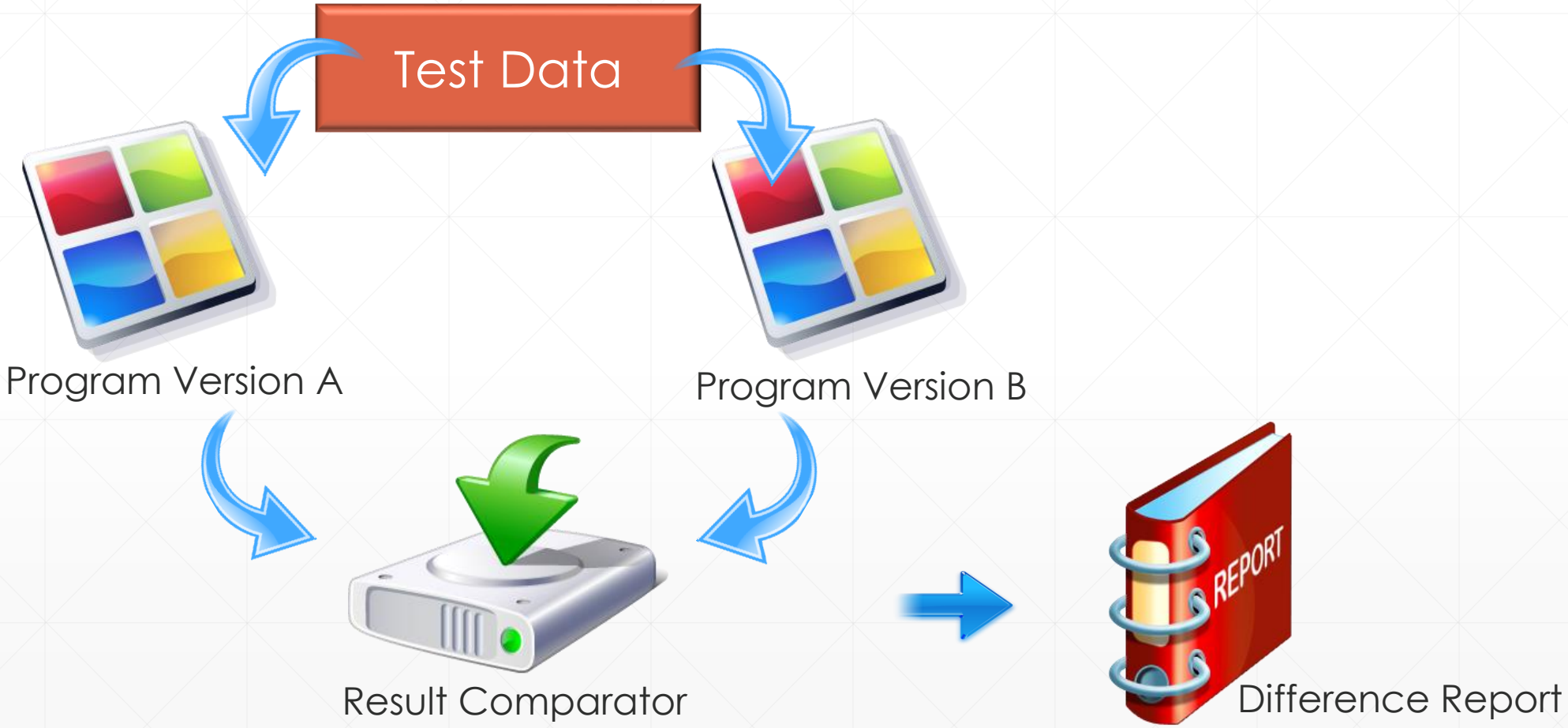
Testing Techniques - Others

Back-to-back testing

- Which is used when versions (e.g. version 1 and version 2) of a system are available. The systems are tested together and their outputs are compared.
- Differences between the test results highlight potential problems



Testing Techniques - Others



Testing Techniques - Others

- There are other tests that are in a special category, as listed below:
  - Peak load testing
  - Storage testing
  - Procedure testing
  - Recovery testing
  - Human Factors testing
  - etc

Testing Strategy

- A testing strategy is a general approach to the testing process, rather than a method of devising (work out of) particular system or component tests.

Testing Strategy

Two Techniques:

- ✓ Proactive
- ✓ Reactive

Exercise

You have successfully developed an online e-commerce website which sells Malaysia handicrafts to all over the world. Your customer (owner of the website) requested to test the website before it is published. He also requested to ensure the reliability of the website in view of thousands of potential users from all over the world.

Suggest **TWO (2)** testing techniques to fulfill the customer's requirements.

Test Case Design and Planning

Test Case Design & Planning

- Test planning is concerned with setting out standards for the testing process rather than describing product tests
- In order to yield a useful testing result, a test plan should be produced. The contents of the test plan may include: -
  - **The testing process**
  - **Requirement traceability**
  - **Tested items**
  - **Testing schedule**
  - **Testing recording procedures(example)**
  - **Hardware & software requirements**
  - **Constraints**

Test Case Design & Planning
Test Plan Content

Test Plan

1. The testing process

- ✓Unit Testing
- ✓Module Testing
- ✓Sub-system Testing
- ✓System Testing
- ✓Acceptance Testing

Test Case Design & Planning
Test Plan Content

Test Plan

1. The testing process
2. Requirement Traceability

✓Unit Testing - SRS item x
✓Module Testing - SRS item y

Test Case Design & Planning
Test Plan Content

Test Plan

1. The testing process
2. Requirement traceability
3. Tested items

- ✓Program Modules
- ✓User Procedures
- ✓Operator Procedures

Test Case Design & Planning
Test Plan Content

Test Plan

1. The testing process
2. Requirement traceability
3. Tested items
4. Testing Schedule

✓Gantt Chart on the schedule to test each item

Test Case Design & Planning
Test Plan Content

Test Plan

1. The testing process
2. Requirement traceability
3. Tested items
4. Testing Schedule
5. Testing recording procedures(example)

Test Case Design & Planning
Test Plan Content

Program Name:			Tester:		
Test Date:					
No	**Objective/Test Cases**	**Test Data**	**Expected Results**	**Actual Results**	**Remarks/ Comments**
	To generate a report to list selected month's sales	June to September Sales data	Monthly sales report		

Test Case Design & Planning
Test Plan Content

Test Plan

1. The testing process
2. Requirement traceability
3. Tested items
4. Testing Schedule
5. Testing recording procedures(example)
6. Hardware & software requirements

✓E.g. Samsung Galaxy Alpha, OS Android, 2G GSM, 3G UMT

Test Case Design & Planning
Test Plan Content

Test Plan

1. The testing process
2. Requirement traceability
3. Tested items
4. Testing Schedule
5. Testing recording procedures(example)
6. Hardware & software requirements
7. Constraints

- ✓test item availability
- ✓test resource availability
- ✓time constraints

Test Case Design & Planning

- Three approaches for test case design:
  1. Requirements-based testing
     - Test cases are designed to test the system requirements are being met
  2. Partition testing
     - Identify input and output partitions and design tests so that the system executes inputs from all partitions and generates outputs in all partitions
  3. Structural testing
     - Use knowledge of the program's structure to design tests that exercise all parts of the program
     - Should try to execute each program statement at least once

Testing Principles/Guidelines

What are the guidelines to improve software testing?

Testing Principles/Guidelines

- **All tests should be traceable to customer requirements** – Most severe defects →If the program fail to meet its requirements.
- **Tests should be planned long before testing begins** – Test plan can begin as soon as the req. model is complete. Test cases can begin as soon as design model has been confirmed.
- **The Pareto principles applies to software testing** – 80% of all errors found during testing will likely traceable to 20% of all program. So, isolate suspect components & thoroughly test them.
- **Testing should begin "in small" and progress toward testing "in the large"** – Firstly, the testing focuses on individual component, then, focus shifts to find errors in integrated components and finally in the entire system.

Testing Principles/Guidelines

- **Exhaustive testing is not possible** – impossible to test every combination of paths during testing because the numbers of possible paths could be very large.
- **To be most effective, testing should be conducted by an independent third party** – developers may selectively test the parts that they think have defects.