Software Testing Methodologies

BITS Pilani

Prashant Joshi

Module 1: Agenda

innovate achieve lead

Module 1: Introduction to Software Testing & Techniques

- **Topic 1.1** — Introduction to Software Testing
- **Topic 1.2** — Overview of the Course
- **Topic 1.3** — Software Testing Techniques
- **Topic 1.4** — Software Testing – Quality Attributes, Types & Levels

innovate achieve lead

BITS Pilani

Pilani Campus

Topic 1.1: Introduction to Software Testing

Software Testing Methodologies

Software Testing & Methodologies

- What is your view?
- What do you think it is all about?
- What do you wish to learn?
- Why do you want to learn?
- Is it important? How important is it?
- Why do you do it?
- What does it tell you?
- Is there a career in it?

A definition

Testing is a process of executing a program with the intent of finding errors

Why do we test?

- Make a judgement about quality or acceptability
- To discover Problems

What world are we talking of?

Embedded Systems

Messaging

Multimedia

Software

Space

Hardware

Networking

You Name it!

Security

Travel

Database

Learning

Telecommunications

Automobile

Analysis

Home

Medicine

Infotainment

What world are we talking of?

Messaging

Multimedia

Embedded Systems

Software is (nearly) everywhere. Thus we are talking about that "everything".

it!

Da

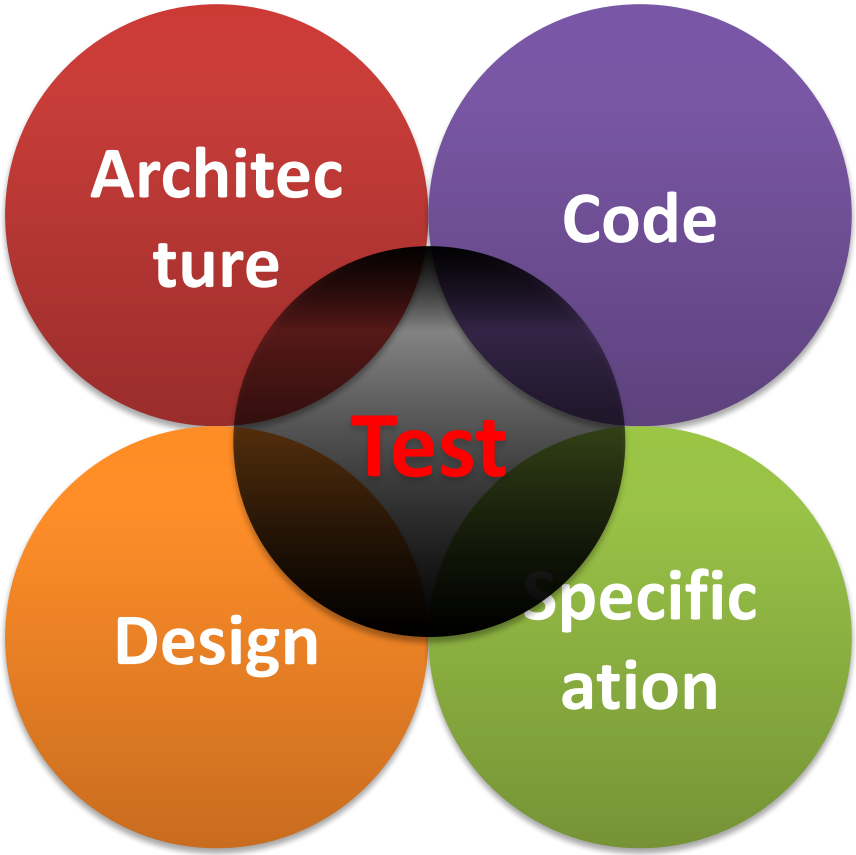
Home

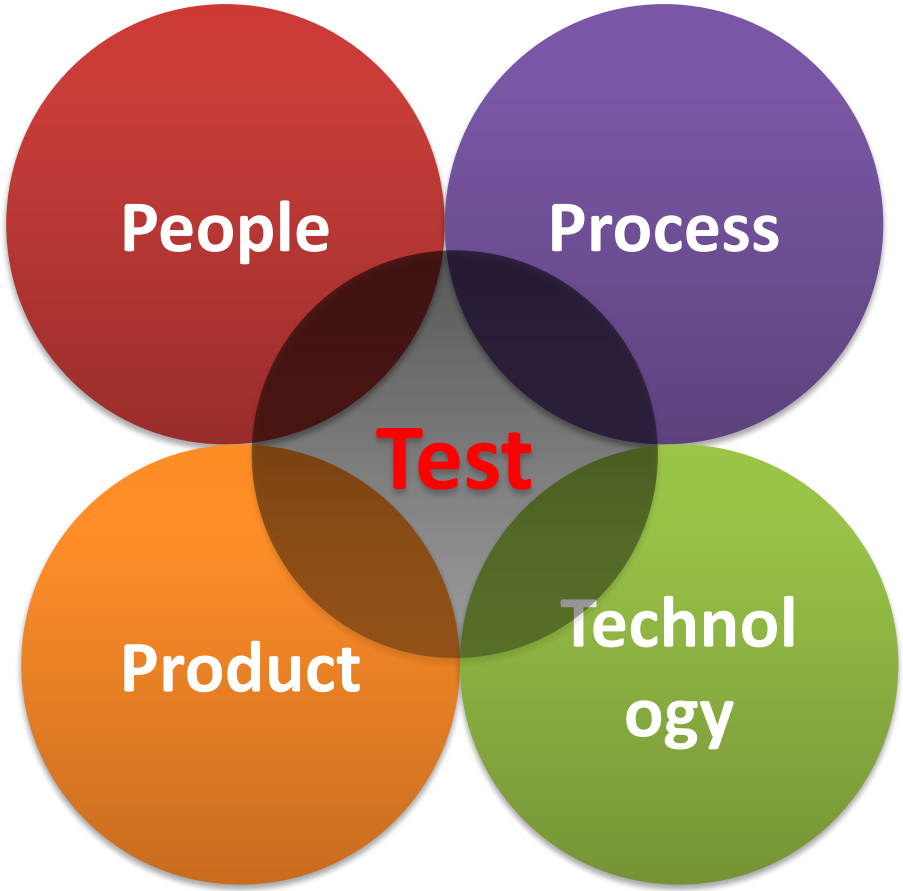
Infotainment

Medicine

Analysis

Building Blocks





Psychology of Test

innovate achieve lead

Psychology of test

- Attitude to break the system
- Constructively destructive
- How come it works!

The Medical Lab

- Reports normal Cannot detect the problem
- A failure would trigger the way to treat

We do not hope for failure we work for failures not to occur

Product

innovate achieve lead

Product Under Test

- Software is not alone
- Works with various systems
- Various systems work with each other

Most vital

- System must work right, always!
- Available, always!

Process

innovate achieve lead

Steps to success

- Increase probability of success
- Bring generality and consistency
- Measure: What you cannot measure you cannot control!
- Continuously improve

Even making tea is a process; To make good tea requires good process

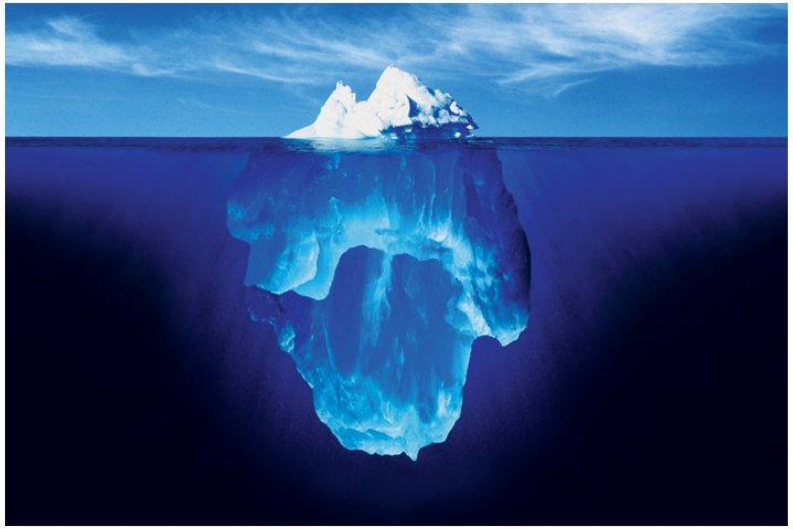
Technology

Something all of us look for and wish to excel

Software Testing is a *highly technical* activity

Depth of STM

- Lets look at the depth of the subject
- Lets explore
- Lets look at the facets



Views of STM

Software Engineering

IEEE Definition

- Software Engineering: (1) The application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of Software; that is the application of engineering to software (2) The study of approaches as in (1).

- Software Engineering is the establishment and use of a sound engineering principle in order to obtain economically software that is reliable and works efficiently on real machine. **[Fritz Bauer]**

- **Where did Software Engineering come from?**
- **Rather why was there a thought of Software Engineering?**

Software Engineering

Software Engineering came in the 60s; with attempts to develop large software systems various problems occurred

- Cost Overruns
- Late delivery
- Lack of reliability
- Inefficient Systems
- Performance problems
- Lack of usability

The aim was to overcome the above issues and much more…

Goals of Software Engineering

1. To improve quality of software
2. To improve the productivity of developers and software teams

And many more…

Our focus is on 1 as a part of our course

Software Quality Attributes

- **Reliability**
- **Efficiency**
  - Speed
  - Resource management
- **Usability**
  - User friendliness
  - Intuitive
- **Maintainability**: Should be easy to maintain
- **Scalability**: Should scale as per the requirements of the user
- **Portability**: Should work on various platforms/systems/hardware
- **Security**: Should be secure from attacks
- **Testability**: Should be testable

Software Quality Attributes

- What is important to User?
- What is important to the Engineers?
- What is important for Mission Critical System and a Word Processor?
  - Compare say Space Shuttle software and a Text Editor

Software Testing Methodologies

BITS Pilani

Prashant Joshi

innovate achieve lead

BITS Pilani

Pilani Campus

Topic 1.2: Overview of the Course

Learning Principles

Make everything as simple as possible, but not simpler

- Albert Einstein

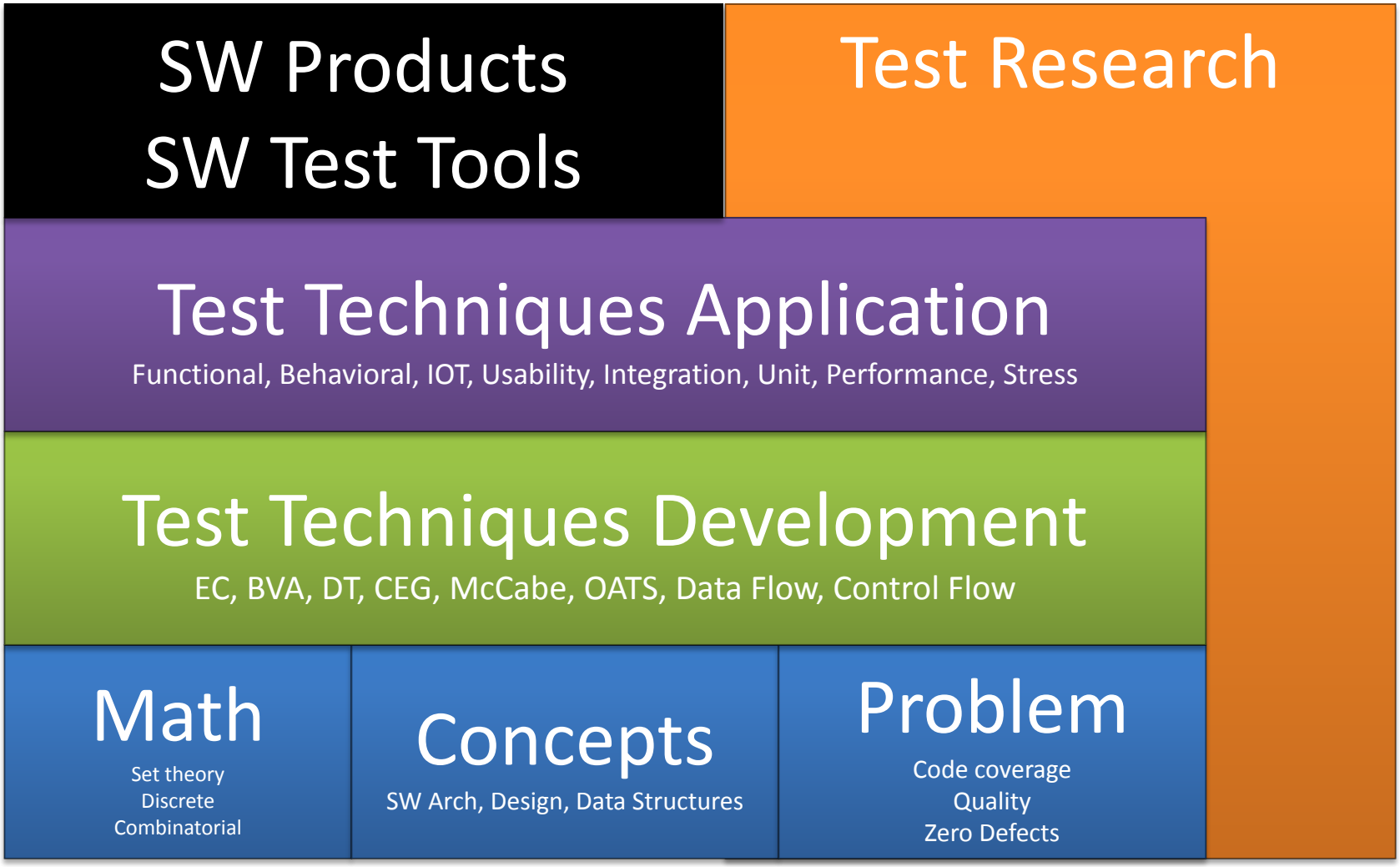
I hear and I forget, I see and I remember, I do and I understand

- Confucius

"… You can know the name of a bird in all the languages of the world, but when you're finished, you'll know absolutely nothing whatever about the bird. You'll only know about humans in different places, and what they call the bird, So let's look at the bird and see what it's doing – that's what counts"

– Richard P Feynman (Book: What Do You Care What Other People Think)

Building Blocks – A View



Ask the Questions!

innovate achieve lead

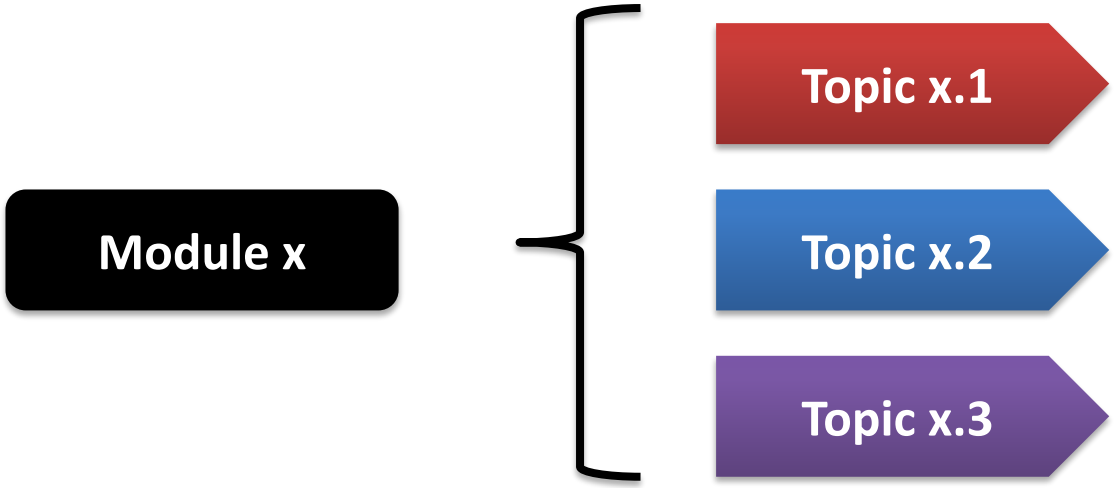
- What
- When

How

- Why

Our Course Structure

- Modules
  - Course is divided into Modules
- Topics
  - Each module is divided into Topics



- Self Study & Quick Review
  - Every Module has Self Study & a Quick Review

Modules

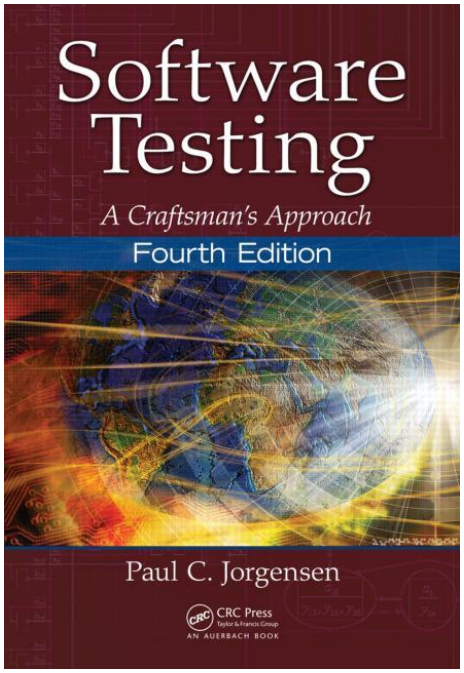
Module No	Module Title	Objectives
1	Introduction to Software Testing & Techniques	Introduce the course and course handout. Bring a perspective of need and motivation for this course. Provide an overview of the course, quality attributes, levels and types of Testing
2	Mathematics and Formal Methods	Provide a base to the software testing techniques in form of mathematics and formal methods. Review topics of permutation/combination, discrete mathematics and graph theory. Focus is on the relevance to software testing.
3	Specification Based Testing	Bring an approach to look at the system from specification perspective. Learn the relevant techniques for testing specifications – Equivalence Class, Boundary Value Analysis, Combinatorial, Decision Tables and Domain Testing
4	Code Based Testing	Take a code level approach to testing and assuring quality. Learn the relevant techniques for testing code – Path Based Testing and Data Flow Testing
5	Model Based Testing	Introduce Model Based Testing. Various Model for Software testing, their choice and techniques. Learn Finite State Machine, Petri Nets and State Charts. Learn to use these to derive testing cases

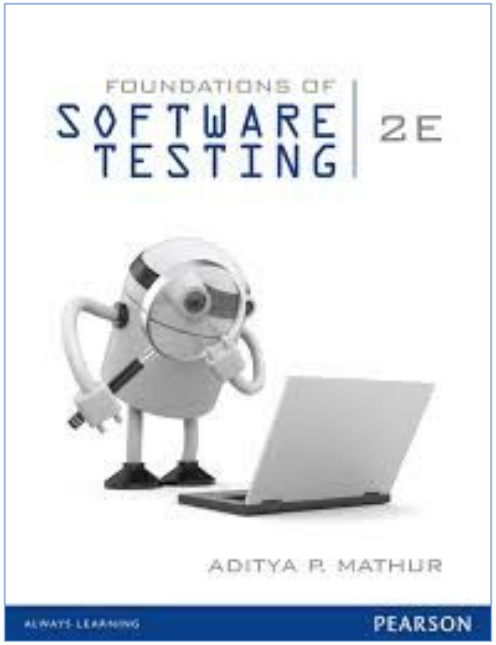
Modules

Module No	Module Title	Objectives
6	Object Oriented Testing	Understand the issues in OO Software Testing. Learn techniques and sublets of Unit, Integration and Systems Testing of OO Software. GUI Testing for OO Software
7	Integration & System Testing	Overview and need for Integration and Systems Testing of Software. Learn the techniques of Integration and Systems Testing
8	Life-Cycle Based Testing	Provide an overview from a life-cycle perspective of Software and Software Products. Agile Testing and Agile Model-Driven Development. Role of Test engineers in life-cycle-based testing
9	Test Adequacy & Enhancement	Learn the need for test adequacy and need for enhancement of test cases. Various techniques and criteria for measuring of test adequacy (data and control flow). Using the criteria to enhance test cases.
10	Test Case Minimization, Prioritization and Optimization	Explore and understand the need for minimization and prioritization. Review the regression test problem. Selection of test cases for regression.

Text Books

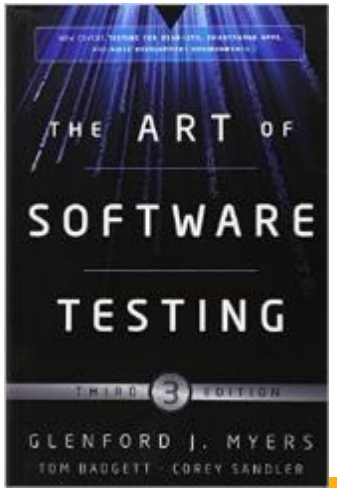
T1	Software Testing – A Craftsman's Approach, Fourth Edition, Paul C Jorgenson, CRC Press
T2	Foundations of Software Testing, Second Edition, Aditya P Mathur, Pearson

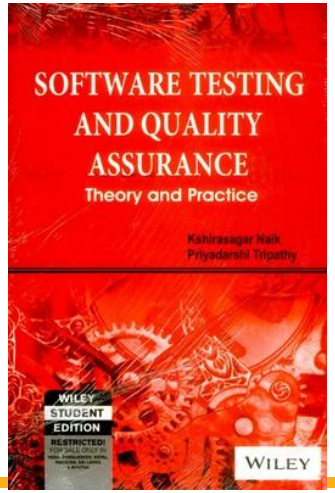


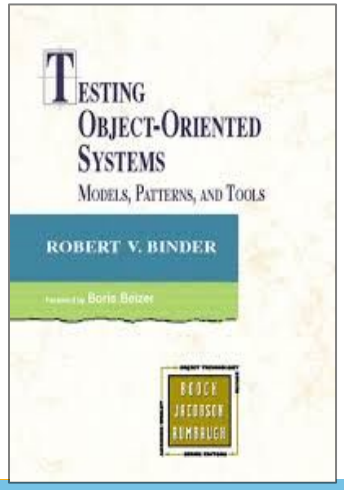


References

R1	The Art of Software Testing, Third Edition, Glenford J. Myers, Tom Badget, Corey Sandler,
R2	Software Testing and Quality Assurance – Theory and Practice, Kshirasagar Naik, Priyadarshi Tripathy, Wiley, 2013
R3	Testing Object Oriented Systems: Models, Patterns and Tools, Robert V Binder, Addison Wesley
R4	Guide to Software Engineering Body of Knowledge, Version 3, IEEE







Software Testing Methodologies

BITS Pilani

Prashant Joshi

BITS Pilani
Pilani Campus

Topic 1.3: Software Test Techniques

Test Techniques

Based on Engineers experience and intuition

- Exploratory
- Ad-hoc

Specification Based Techniques

- Equivalence Partitioning
- Boundary Value Analysis
- Decision Tables
- …

Code Based Techniques

- Control Flow
- Data Flow
- …

Test Techniques

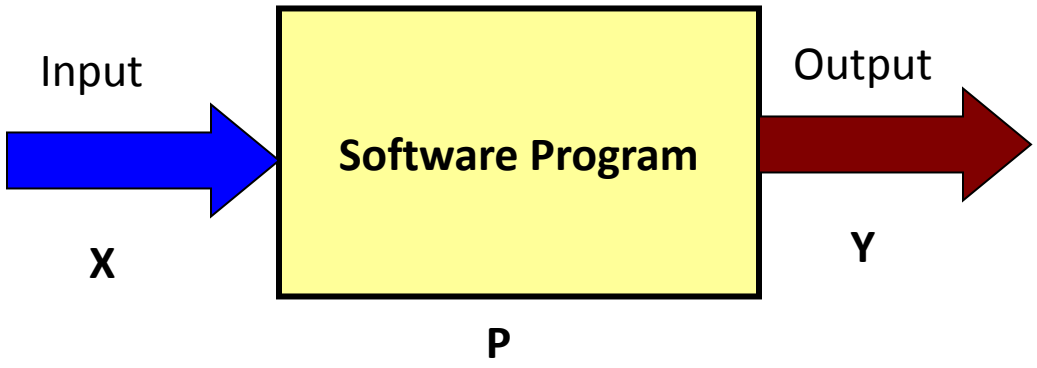
Techniques based on nature of application

- Object Oriented
- Component-based
- Web-based
- GUI
- Protocol Conformance
- Real Time Systems

Usage based testing

- Operational Profile
- Reliability Engineered Testing

Testing Methods



Testing methods

Based on the source of information used for testing

- No information is used
- Specification (Example: Requirements specification)
- Design or LLD or Source Code (Internals of a program)

A Simple Example

```
#include <iostream> int main() { int a, b, c; // initialize the
three number to zero a=b=c=0; int max=0; // Indicate the program
purpose std::cout << "Program computes max of three integers" <<
std::endl; // Ask for input of numbers std::cout << "Enter the
values for a, b, & c one on each line" << std::endl; std::cin >>
a; std::cin >> b; std::cin >> c; // Start with a as max max=a;
//Logic to find the max. Compare with other two. if(a<b) { max =
b; } if(max<c) { max=c; } // Output the result of the
computation std::cout << "Max of three is " << max << std::endl;
} // End of program
```

A Simple Example

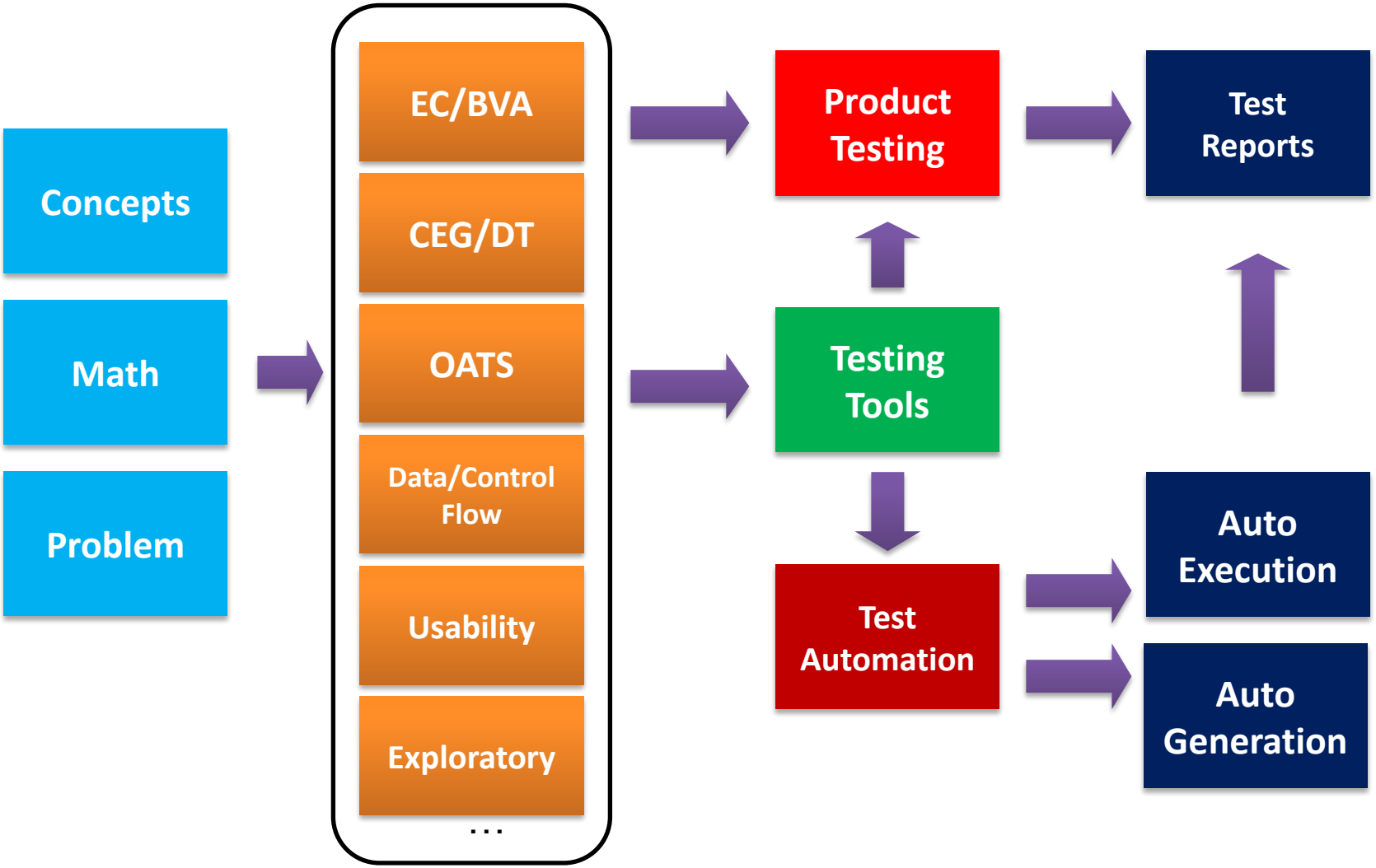
innovate achieve lead

```
#include <iostream>

int
main()
{
  int a, b, c;
  // initialize the three number to zero
  a=b=c=0;
  int max=0;

  // Indicate the program purpose
  std::cout << "Program computes max of three integers" << std::endl;

  // Ask for input of numbers
  std::cout << "Enter the values for a, b, & c one on each line" << std::endl;
  std::cin >> a;
  std::cin >> b;
  std::cin >> c;

  // Start with a as max
  max=a;

  //Logic to find the max. Compare with other two.
  if(a<b) {
    max = b;
  }
  if(max<c) {
    max=c;
  }

  // Output the result of the computation
  std::cout << "Max of three is " << max << std::endl;

} // End of program
```

A Simple Example – Output

```
~/dev/stm $ ./a.out
Program computes max of three integers
Enter the values for a, b, & c one on each line
3
4
5
Max of three is 5
~/dev/stm $ ./a.out
Program computes max of three integers
Enter the values for a, b, & c one on each line
34
54
89
Max of three is 89
~/dev/stm $ ./a.out
Program computes max of three integers
Enter the values for a, b, & c one on each line
56
57
99
Max of three is 99
~/dev/stm$
```

Progression



Software Testing Methodologies

BITS Pilani

Prashant Joshi

innovate achieve lead

BITS Pilani

Pilani Campus

Topic 1.4: Software Testing – Quality Attributes, Types & Levels

High Level Design

Modular

- Structure chart based
- Broken down into various modules and has call relationships i.e. (set of modules + call relationships)
- Executes in a sequence
- Uses a modular design pattern

Object Oriented Design

- Class diagram = Set of Classes + relationships
- Inheritance
- Association
- Aggregation

Low Level Design

innovate achieve lead

Low Level Design

- Major Algorithms
- Data structures

Implementation

- Choice of programming language
- Coding
- Low level algorithms
- Low level data structures
- Specific code constructs

Basic Definitions

innovate achieve lead

Error

- An error is a mistake. Errors propagate. A requirements error may (will) get magnified as design and still amplified in later phases

Fault

- A fault is a result of an error. Fault aka. Defect, is an expression of error, where representation is a mode of expression ex: narrative text, dataflow diagrams etc
  - Faults of omission: Occurs when something is missed out
  - Faults of commission: Occurs when some representation is incorrect

Failure

- A failures occurs when a fault executes.
  - How do we relate this to faults of commission and omission?

Basic Definitions

innovate achieve lead

Incident

- An incident occurs when a failure occurs. An incident is the symptom associated with a failure that alerts the user the occurrence of a failure

Test

- Testing is concerned with errors, faults, failures and incidents. A test is an act of exercising testing cases with two goals,
  - to find failures
  - to demonstrate correct execution

Test Case

- A test case has an identity and is associated with a program behaviour. A test case also has a set of inputs and a list of expected outputs

Good Test Case

- High probability of finding a defect which is yet to be discovered
- It is not redundant
- “Best of the Breed”.
- Neither too simple nor too complex

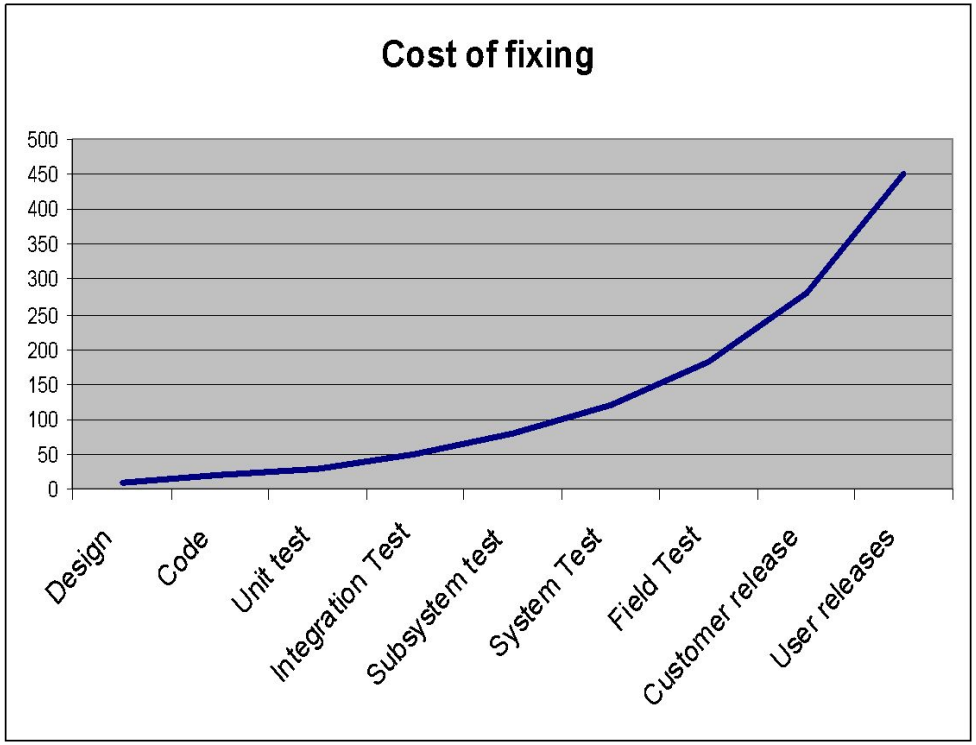
A Test Case

- **A unique ID**
- **An input**
  - Precondition: Circumstances that hold prior to execution of the test case
  - Actual inputs: The actual inputs for a particular test case/method
- **Expected Output**
  - Postconditions: Circumstances that hold after the execution of the test
  - Actual outputs: The actual outputs
- **Verdict**: A final PASS/FAIL/INDETERMINATE statement for the test activity

Goals of SW Testing

- To show that the software system satisfies the requirements and performs as expected. The requirements may be explicit or implicit.
  - Explicit: User Interface, Specified Output
  - Implicit: Error handling, performance, reliability, security
- To have “confidence” in the software system. To assure that the software works. To demonstrate that the Software works.
- To find defects
- To prevent defects
- Ensure software quality

Cost of fixing



- Software Testing (exhaustive) is a very time consuming activity
- Software testing can be most expensive activity in Software development and maintenance

Testing Types & Levels

Test Levels based on Target of Test

- Unit testing
- Integration testing
- Sub-system testing
- System testing
- Acceptance testing
- Alpha/Beta Testing
- Field testing

Types based on execution

- Dynamic Testing (Execution based testing)
- Static Testing (No execution is involved)

Testing Types & Levels

innovate achieve lead

Test Levels based on Objective of Testing

- Acceptance
- Installation testing
- Alpha/Beta Testing
- Conformance/IOT
- Reliability
- Regression
- Performance
- Stress
- Usability

Test Suites

- Test Suite is a set of test cases for a particular software system or a product
- A *typical* Test Suite contains
  - Random tests
  - Specification based tests
  - Code Based tests

Software Testing Methodologies

BITS Pilani

Prashant Joshi