Software Testing and Software Maintenance

Software Testing Overview

- Testing is a part of the software processes
- Testing is to detect faults in programs or specifications
  - Faults: incorrect material in the specification can lead to incorrect material in the source code which can lead to failures during program execution
  - Failures: failure in software execution (e.g. system crash, loss of function or data)
- Manual testing or Automate testing

Key test activities in SD process

- Unit testing:
  - An individual unit of software is tested to ensure that it works correctly.
- Integration testing:
  - Two or more units are tested to ensure that they interoperate correctly.
  - It can be Top-Down, Bottom-up, or take an ‘end-to-end user functionality’ approach.

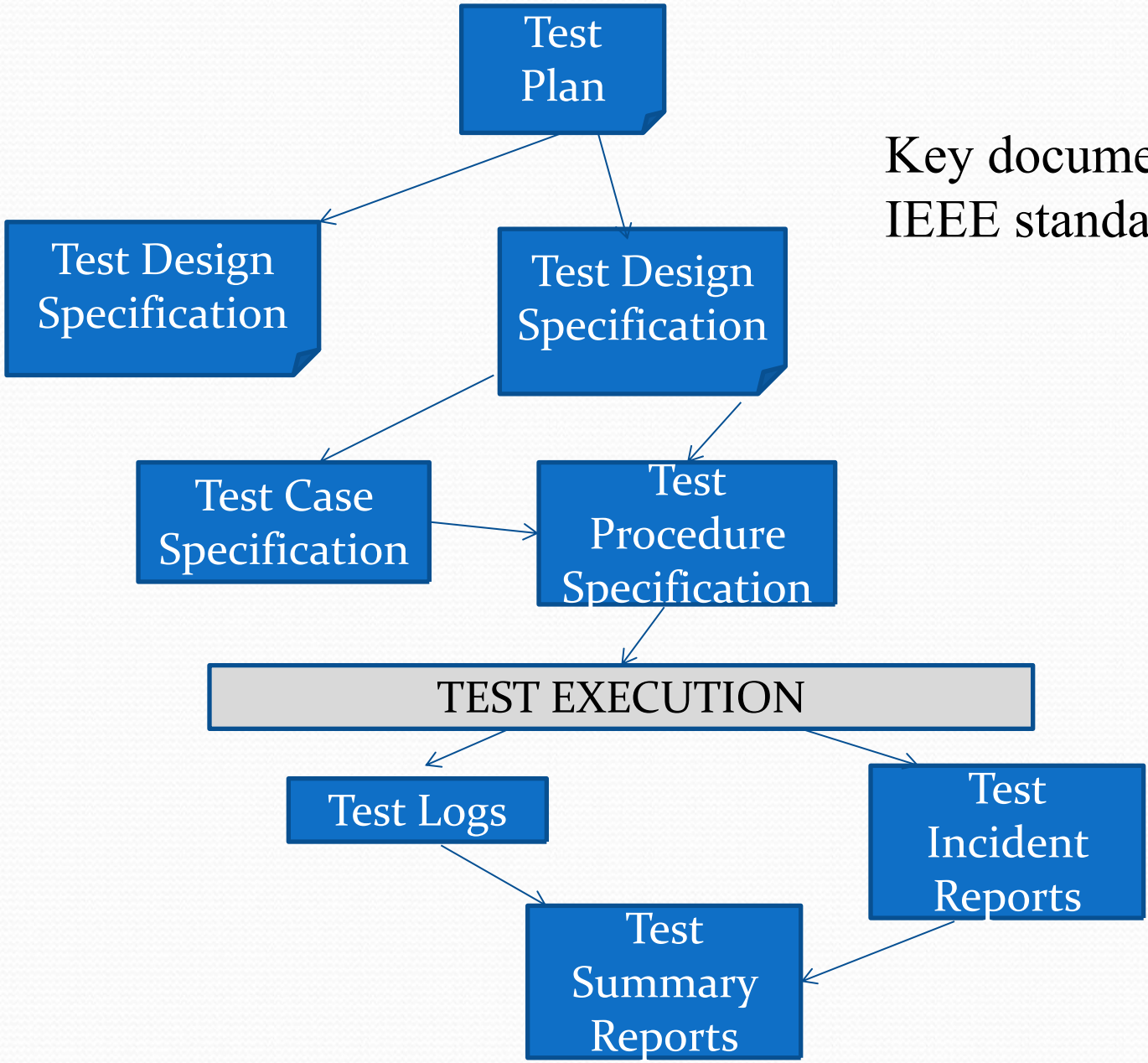
Key test activities

- System testing
  - The entire software system is tested to make sure that it works correctly and that it meets/solves the user's needs/problem.
- Acceptance testing
  - The entire software system is tested at the customer environment (field testing or alpha and beta testing) to make sure that it meets the user's needs.

Test Planning

- A typical test plan would include the following information:
  - Items to be tested: method, class, function, module, component
  - Tasks to be performed: design test cases, test data, test script, etc.
  - Responsibilities: who
  - Schedules: when
  - Required resources

Test Artefacts



Key documents in IEEE standard

Testing approaches

- Static verification
  - A form of software verification where the software isn't actually used.
  - Generally not detailed testing, but checks mainly for the sanity of the code, algorithm, or documentation.
- Dynamic testing
  - The software must actually be compiled and run to test the physical response from the system.
  - Working with the software, giving input values and checking if the output is as expected by executing specific test cases which can be done manually or with the use of an automated process.

Types of Static verification

- Design review:
  - A design review provides the opportunity to verify that the design for software is correct before it is implemented.
  - The purpose of the review is to ensure that the design allows the software to fulfill its goals
- Static Code Analysis:
  - Walk-through: walk through the code, not formal
  - Code inspection: formal verification to check the trail of execution

Types of Dynamic testing

- Black-box testing
  - examines the functionality of an application/software under the test without peering into its internal structures or workings
- White-box testing
  - tests internal structures or workings of an application, as opposed to its functionality
- Fault Insertion
  - Faults are inserted into the source code (or called mutation), and the code is checked to see if the mutant produces a different output.

Test activities

- Analysis of the source code and/or specification to design the test cases according to a testing technique (e.g. techniques in whitebox /blackbox)
- Test case design
  - A set of conditions or variables under which a tester will determine whether an application, software system or one of its features/unit is working as it was originally established for it to do.
  - E.g.: A test case is if age is an invalid input value (i.e. <0)

Test activities

- Test data design, a test may cover several test cases
  - Test data includes input data and expected output data
- Test code or test procedure implementation
- Test execution: running the test

Unit testing example
Path coverage technique (white-box)

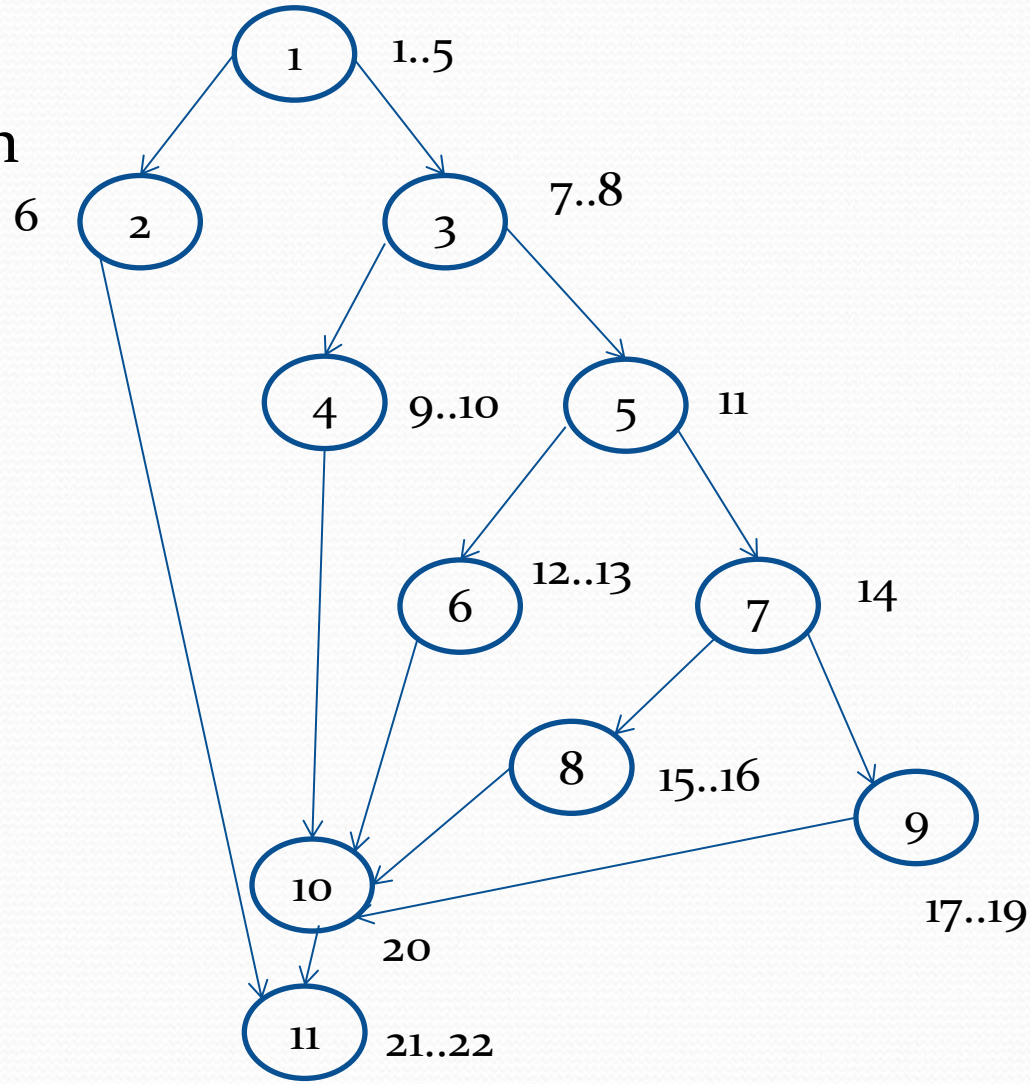
1	`public static String Grade (int exam, int course) {`
2	`String result="null";`
3	`long average;`
4	`average = Math.round((exam+course)/2);`
5	`if ( (exam<0) || (exam>100) || (course<0) || (course>100) )`
6	`result="Marks out of range";`
7	`else {`
8	`if ( (exam<50) || (course<50)) {`
9	`result="Fail";`
10	`}`
11	`else if (exam <= 60) {`

Program Code – Grade (contd.)

```
12  result="Pass,C";
13  }
14  else if ( average >= 70) {
15  result="Pass,A";
16  }
17  else {
18  result="Pass,B";
19  }
20  }
21  return result;
22  }
```

Test case design

- Aim to cover 100% path in the program
- Control Flow Graph:



Example- Grade

- Test case

Case	Nodes
1	1.2.11
2	1.3.4.10.11
3	1.3.5.6.10.11
4	1.3.5.7.8.10.11
5	1.3.5.7.9.10.11

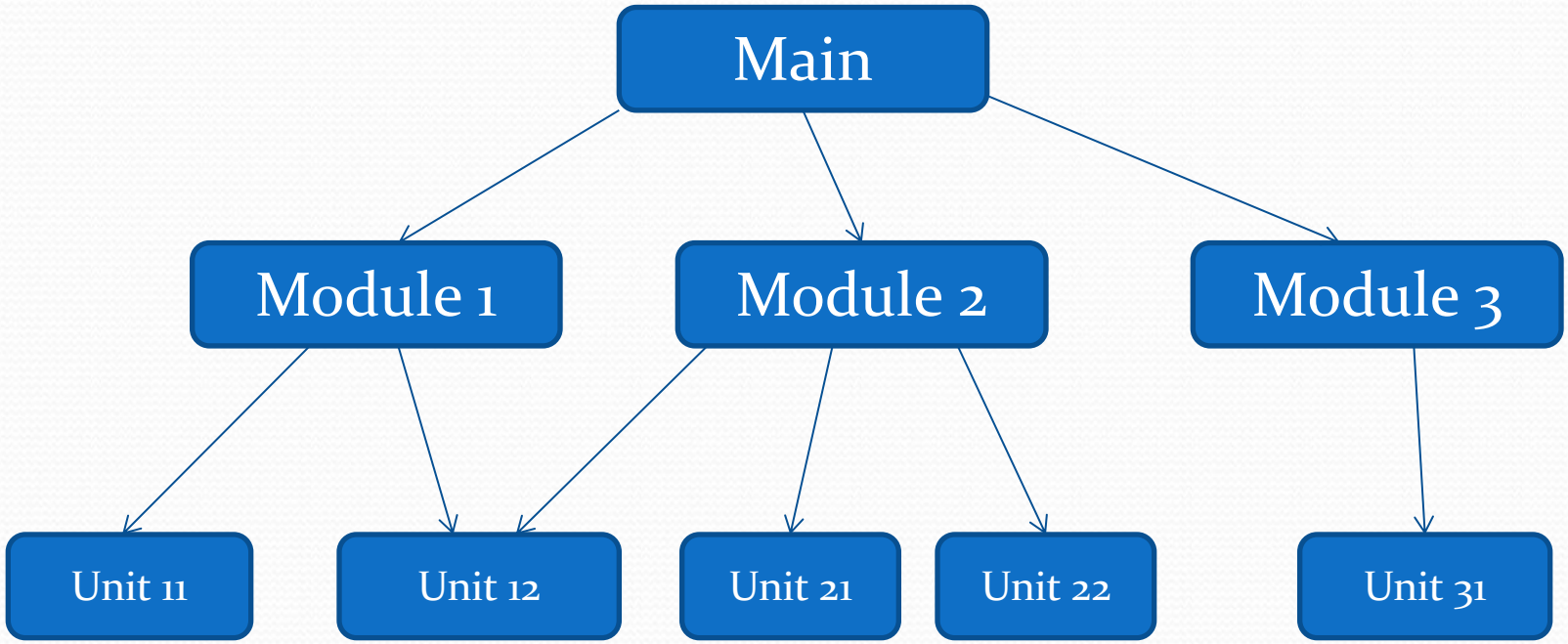
Grade- Test data design

Test No.	Test Cases	Inputs		Expected Outputs
		exam	*course*	*Result*
1	1	-1	1	Marks out of Range
2	2	40	50	Fail
3	3	55	50	Pass, C
4	4	90	50	Pass, A,
5	5	80	50	Pass, B

Integration testing

- Two or more units are tested to make sure they interoperate correctly.
- Test cases explore the interaction between components
- Usually use black box testing
- Approaches
  - Top down integration testing: Testing moves from the top layer of the program to the bottom layers
  - Bottom up integration testing: Testing moves from the bottom layer of the program to the top layer.
  - End to end user functionality: Gradually testing features/functions of the system (branches of the hierarchy of functions)

Example of system components



System testing

- Testing the system as a whole and usually done using Black box tests.
- System testing can be broken down into a number of categories, for instance:
  - Functional testing: verify if the system behaves as specified, using the system interface for tests
  - Regression testing: verify that any changes to the software have not introduced new faults to previously working software

System testing

- Interoperability testing: verify that the system can exchange and share information with other required software products. Tests are run on all pairs of products to see that sharing or exchanging information is correct
- Stress testing: used to verify failure behaviour and expose defects when the load exceeds the specified limits.
- Performance testing: verify that the performance targets for the software have been met. E.g. response latency, maximum numbers of simultaneous users, etc.

Field testing and acceptance testing

- Verify that a system works correctly in a real operational environment.
- Acceptance testing is used prior to payment of a system , the tests are performed by customers to ensure that the system that they ordered works properly.

Software Maintenance

- The process of changing a system after it has been delivered
- Three different types of software maintenance
  - Fault repair: Fixing coding errors or design errors
  - Environmental adaptation: when there is change in system's environment such as the platform operating system, or other support software
  - Functionality addition: The system requirements change in response to organizational or business change

Observations

- Software maintenance takes more budget than development
- More of maintenance budget is spent on implementing new requirements than on fixing bugs

Key points

- Software testing is a part of software development process
- Testing is to detect errors/faults of the programs/specifications
- Testing in SD process may include static verification and dynamic testing.
- Static verification doesnot actually run the source code/software, dynamic testing does need.
- Testing in SD may include unit testing, component testing (integration testing), system testing and acceptance testing

Key points

- Software maintenance is performed after the software has been delivered
- Software maintenance can be bug fixing, modifying the software to adapt new environment, and implementing new or changed requirements
- The cost of software maintenance usually exceed the cost of software development