Software Quality

SOFTWARE TESTING

By: MSMZ

Introduction

- Testing was the 1st software quality assurance tool applied to control software product quality.

Software Test - Definition

- Software testing is a **formal** process carried out by a **specialized testing team** in which a software unit, several integrated software units or an entire software package are examined by **running the programs on a computer**. All the associated tests are performed according to **approved test procedures** on **approved test cases**.

Software Testing Strategies

- *Direct objectives*

a. **To identify and reveal as many errors as possible in the tested software**
b. **To bring the tested software, after correction of the identified errors and retesting, to an acceptable level of quality.**
c. **To perform the required tests efficiently and effectively, within the limits budgetary and scheduling limitation.**

Software Testing Strategies

- *Indirect objectives*

  a. **To compile a record of software errors for use in error prevention (by corrective and preventive actions)**

Software Testing Strategies

- ***Incremental testing strategies:***
  - Bottom-up testing
  - Top-down testing
- ***Big bang testing***

Software Testing Strategies

- Big Bang Testing
  - To test the software as an entirely, meaning is, once the completed package is available.

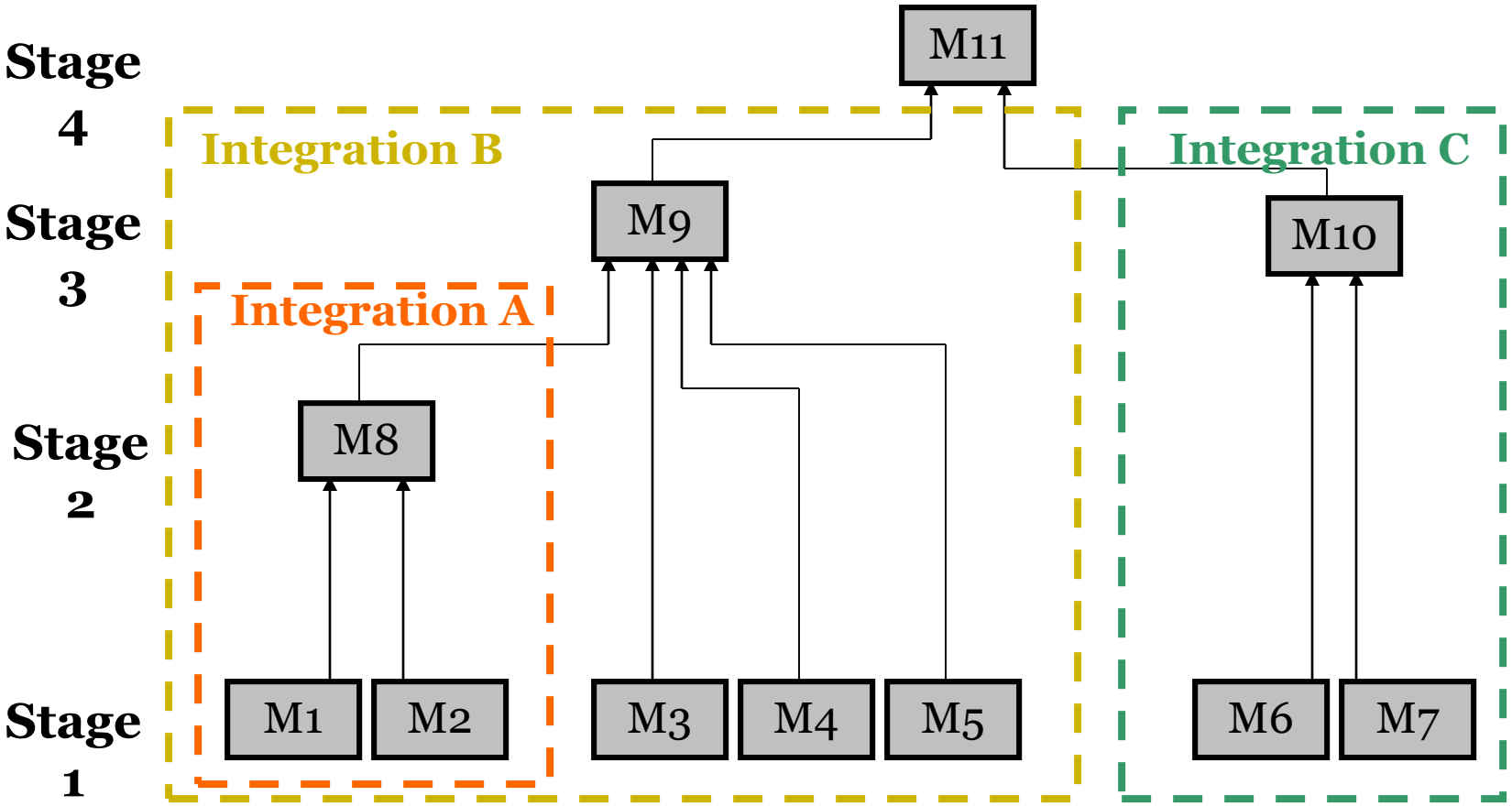
Software Testing Strategies

- Incremental Testing
  - Test the software:-
    - piecemeal in modules, as they are completed (unit test)
    - Test group of tested modules integrated with newly completed modules (integration test)
    - Entire package is test as a whole (system test)
  - Performed according to 2 basic strategies:-
    - Bottom-up
    - Top-down

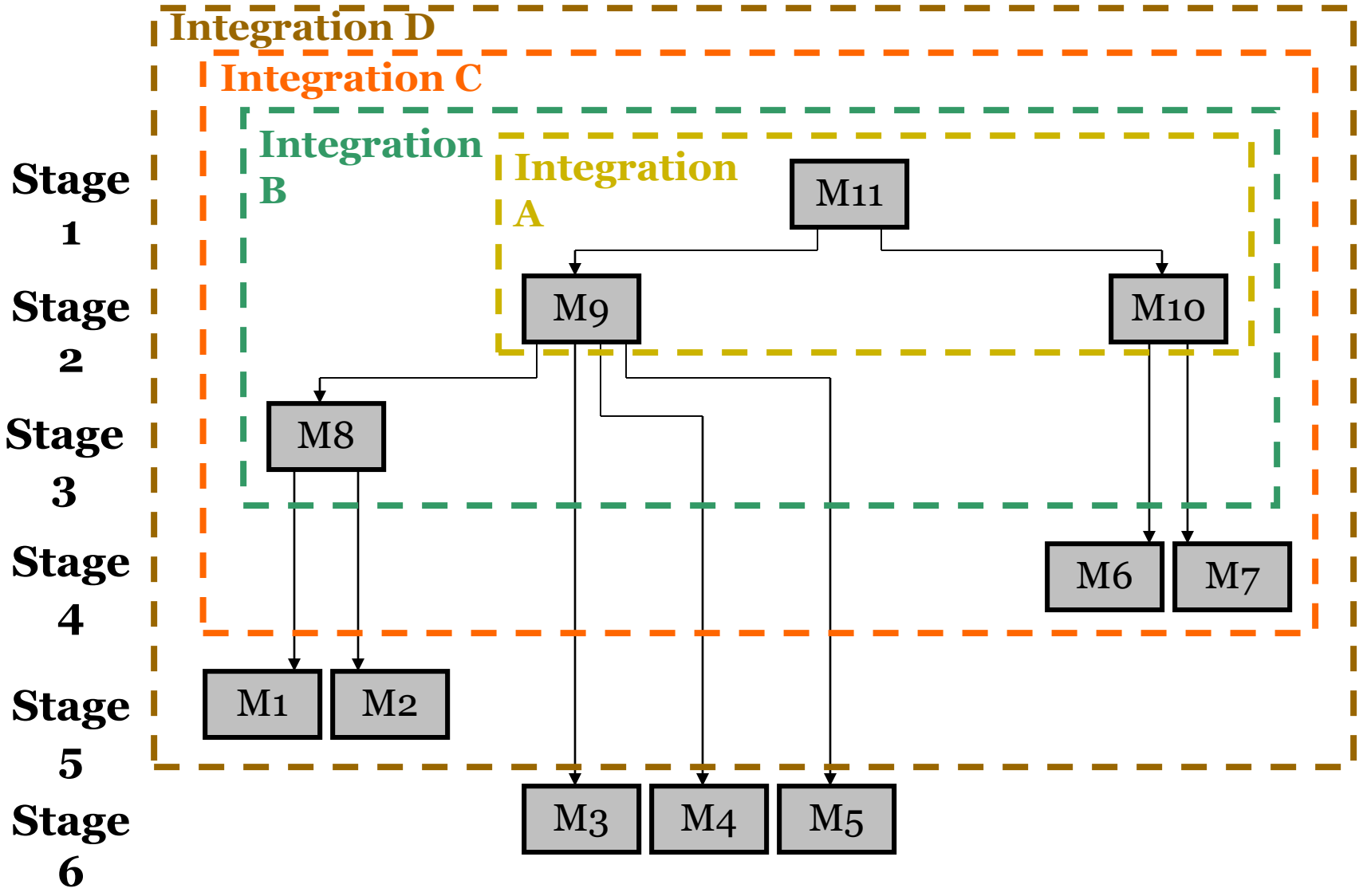
Software Testing Strategies

- Top down
  - The 1st module tested is the main module which is the highest level module in the software structure and the last module to be tested are the lowest level modules.
- Bottom up
  - The order of testing is reversed: the lowest level module are tested first with the main module will be tested last.

Bottom-up testing



Top-down testing

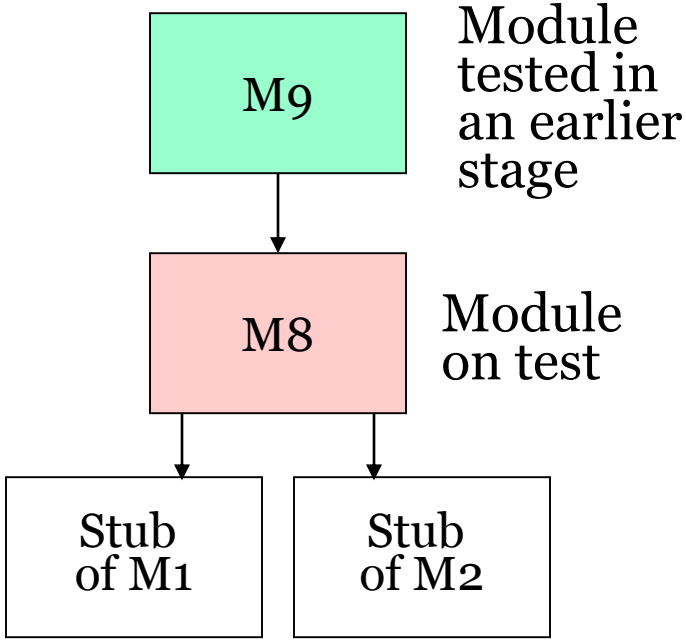


Stubs and Drivers

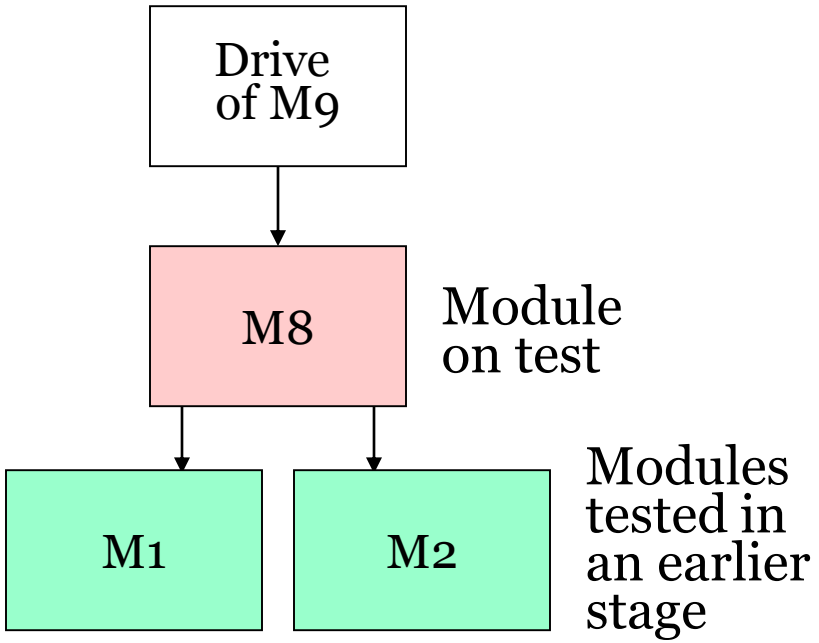
- Stubs and drivers are software replacement simulators required for modules that not available when performing a unit or an integration test.
- Stub (dummy module) replaces unavailable lower level module. It is required for top down testing of incomplete system.
- Driver is a substitute module of the upper level module that activates the module test. It is required in bottom up testing.

Use of Stubs and Drivers for Incremental Testing

Top-down testing of module M8



Bottom-up testing of module M8



Big Bang vs. Incremental Testing

Big Bang

- Identification of errors becomes quite difficult
- Error correction is often a very difficult task
- Requires only single testing operation

Incremental

- Need to do several testing operations for the same program
- Need to prepare stubs and drivers.

Advantages of Incremental Testing

- Usually performed on relatively small software modules, which makes it easier to identify higher percentages of errors compared to testing the entire software package.
- Identification and correction of errors is much simpler and requires fewer resources because it is performed on a limited volume of software.

Software Test Classification: Concept

- Black box (functionality) testing
  - Identifies bugs only according to software malfunctioning as they are revealed in its erroneous outputs.
  - In cases that the outputs are found to be correct, black box testing disregards the internal path of calculations and processing performed.
- White box (structural) testing
  - Examines the internal calculation paths in order to identify bugs.
  - It's other name, glass box testing, better expresses its basic characteristic of investigating the correctness of code structure

IEEE Definitions

Black box (functionality) testing

1. Testing that ignores the internal mechanism of the system or component and focuses solely on the outputs in response to selected inputs and execution conditions
2. Testing conducted to evaluate the compliance of a system or component with specified functional requirements

White box (structural) testing

Testing that takes into account the internal mechanism of a system or component

Software Test Classification: Requirements

Factor category	Quality requirement factor	Quality requirement sub-factor	Test classification according to requirements
Operation	1. Correctness	1.1 Accuracy and completeness of outputs, accuracy and completeness of data	1.1 Output correctness tests
		1.2 Accuracy and completeness of documentation	1.2 Documentation tests
		1.3 Availability (reaction time)	1.3 Availability (reaction time) tests
		1.4 Data processing and calculations correctness	1.4 Data processing and calculations correctness tests
		1.5 Coding and documentation standards	1.5 Software qualification tests
	2. Reliability		2. Reliability tests
	3. Efficiency		3. Stress tests (load tests, durability tests)
	4. Integrity		4. Software system security tests
	5. Usability	5.1 Training usability 5.2 Operational usability	5.1 Training usability tests 5.2 Operational usability tests
Revision	6. Maintainability 7. Flexibility 8. Testability		6. Maintainability tests 7. Flexibility tests 8. Testability tests
Transition	9. Portability 10. Reusability 11. Interoperability	11.1 Interoperability with other software 11.2 Interoperability with other equipment	9. Portability tests 10. Reusability tests 11.1 Software interoperability tests 11.2 Equipment interoperability tests

Software Test Classification: Requirements

Test classification according to requirements	White box testing	Black box testing
1.1 Output correctness tests		+
1.2 Documentation test		+
1.3 Availability (reaction time) tests		+
1.4 Data processing and calculation correctness tests	+	
1.5 Software qualification tests	+	
2. Reliability tests		+
3. Stress tests (load tests and durability tests)		+
4. Software system security tests		+
5.1 Training usability tests		+
5.2 Operational usability tests		+
6. Maintainability tests	+	+
7. Flexibility tests		+
8. Testability tests		+
9. Portability tests		+
10. Reusability tests	+	
11.1 Software interoperability tests		+
11.2 Equipment interoperability tests		+

White box testing
"Path" vs "line" coverage

Applying concept white box, which is based on checking the data processing for each test case. 2 alternative approaches are introduced:-

Path coverage

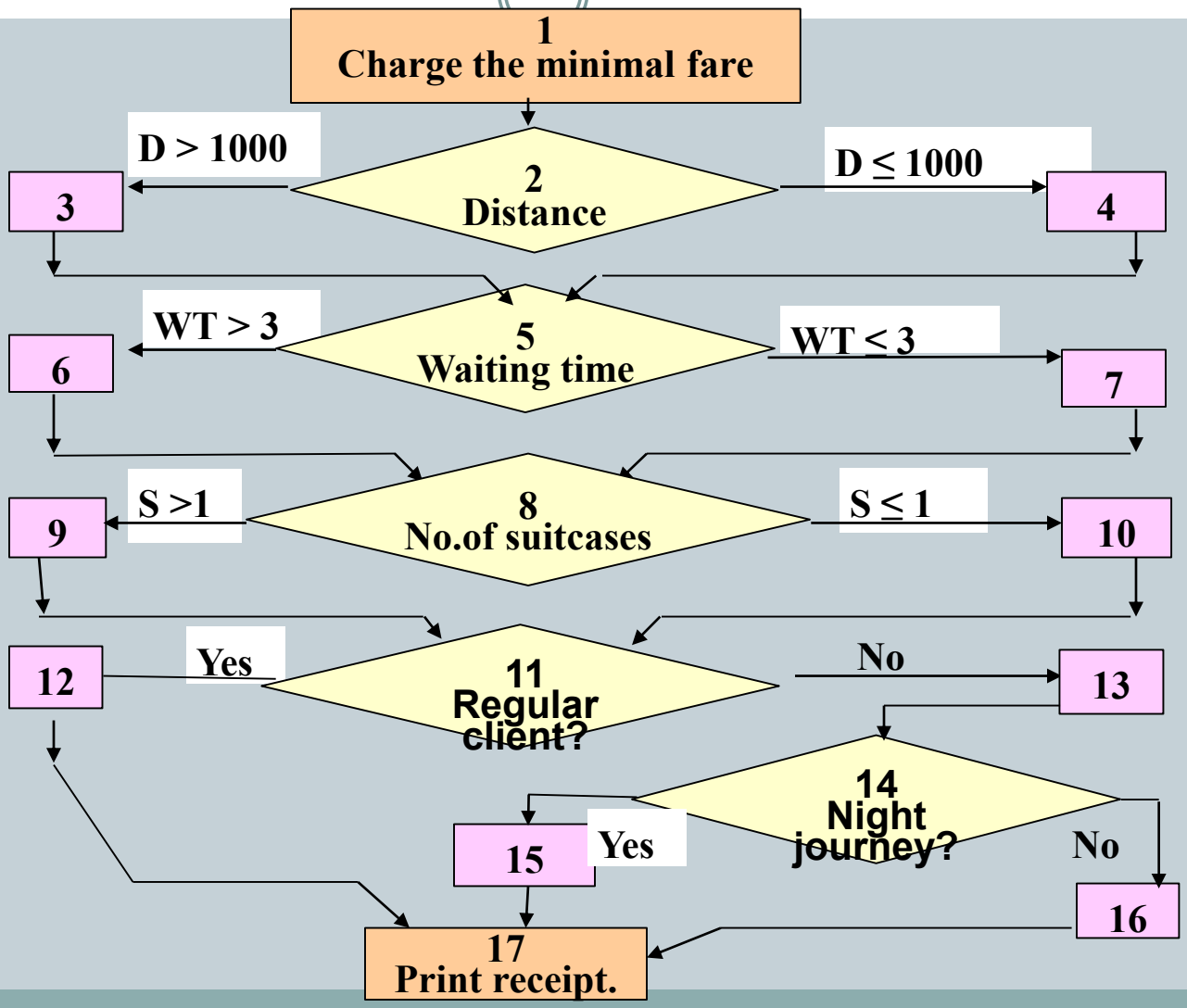
- To plan our test to cover all the possible path, where coverage is measured by the percentage of paths covered.

Line coverage

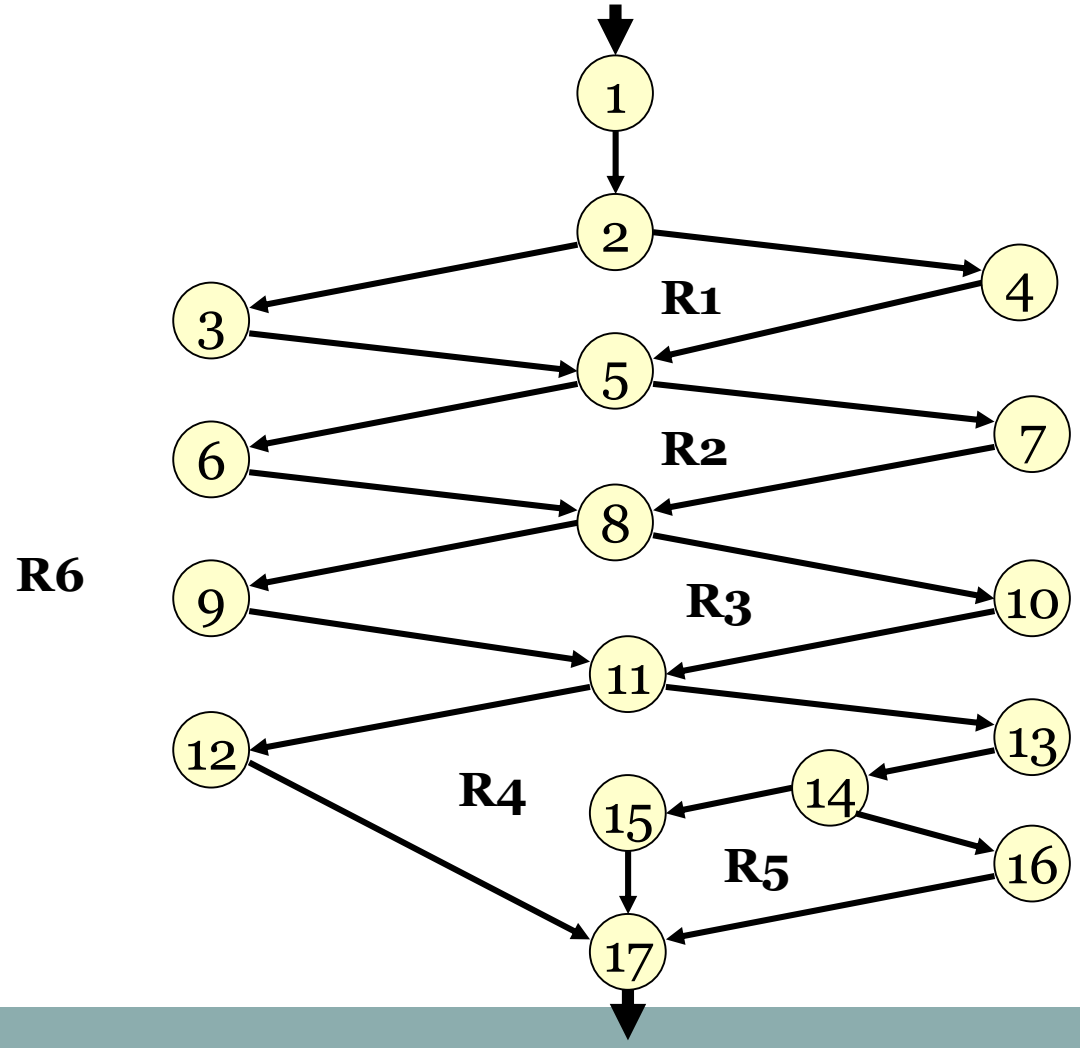
- To plan our test to cover all the program code line where coverage is measured by the percentage of lines covered.

Example Case: Imperial Taxi Services (ITS)

- Imperial Taxi Services (ITS) serves one-time passengers and regular clients (identified by a taxi card). The ITS taxi fares for one-time passengers are calculated as follows.
  1. Minimal fare: $2. This fare covers the distance travelled up to 1000 metres and waiting time (stopping for traffic lights or traffic jams, etc.) of up to 3 minutes.
  2. For every additional 250 metres or part of it: 25 cents
  3. For every additional 2 minutes of stopping or waiting or part thereof: 20 cents
  4. One suitcase: no charge; each additional suitcase: $1
  5. Night supplement: 25%, effective for journeys between 9.00pm and 6.00am
- Regular clients are entitled to a 10% discount and are not charged the night supplement

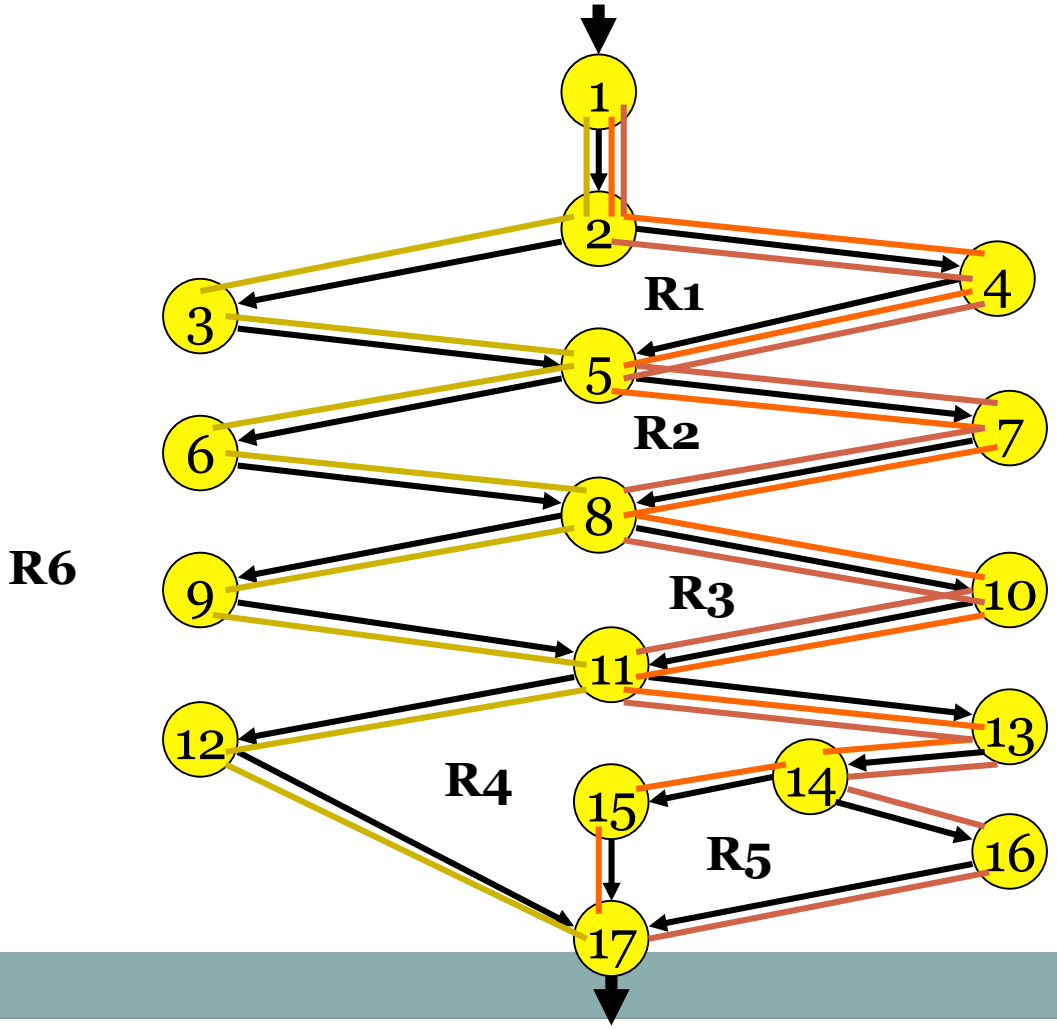

1
Charge the minimal fare
2
Distance
D > 1000
D ≤ 1000
3
4
5
Waiting time
WT > 3
WT < 3
6
7
8
No.of suitcases
S >1
S ≤ 1
9
10
11
Regular client?
Yes
No
12
13
14
Night journey?
15
Yes
No
16
17
Print receipt.

ITS - Program flow graph



ITS - The minimum number of paths

for full line coverage



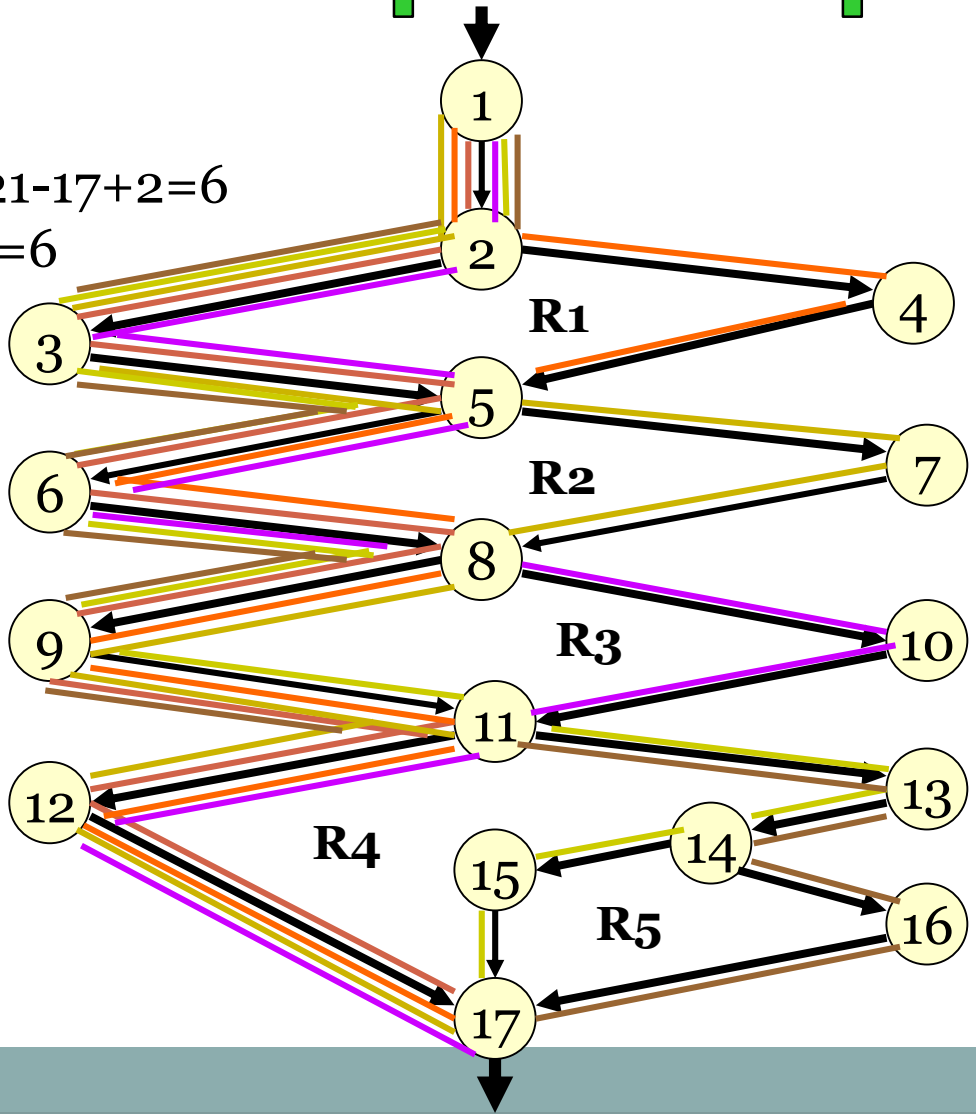
McCabe's Cyclomatic Complexity Metrics

- Measures the complexity of a program or module at the same time as it determines the maximum number of independent paths needed to achieve full line coverage of the program
- And independent path is Any path on the program flow graph that includes at least one edge that is not included in any of the former independent graphs.
- Cyclomatic complexity calculation:
  - V(G)=R
  - V(G)=E-N+2
  - V(G)=P+1

R=Regions
N=Nodes
E=Edges
P=Decisions

ITS - The maximum set of independent paths

V(G)=R=6
V(G)=E-N+2=21-17+2=6
V(G)=P+1=5+1=6



McCabe's cyclomatic complexity metrics

Advantages and disadvantages of white box testing

Advantages:

* Direct determination of software correctness as expressed in the processing paths, including algorithms.
* Allows performance of line coverage follow up.
* Ascertains quality of coding work and its adherence to coding standards.

Disadvantages :

* The vast resources utilized, much above those required for black box testing of the same software package.
* The inability to test software performance in terms of availability (response time), reliability, load durability, etc.

Advantages and disadvantages of black box testing

Advantages:

* **Allows us to carry out the majority of testing classes, most of which can be implemented solely by black box tests, i.e. load tests and availability tests.**
* **For testing classes that can be carried out by both white and black box tests, black box testing requires fewer resources.**

Disadvantages:

* **Possibility that coincidental aggregation of several errors will produce the correct response for a test case, and prevent error detection.**
* **Absence of control of line coverage. There is no easy way to specify the parameters of the test cases required to improve coverage.**
* **Impossibility of testing the quality of coding and its strict adherence to the coding standards.**

Software Quality

VERIFICATION & VALIDATION

By: MSMZ

Verification & validation

- The difference between verification and validation are succinctly expressed by Boehm (1979): *'Verification: are we building the product right?'* and *'Validation: are we building the right product?'*
- These definitions tell us that the role of verification involves checking that the software conforms to its specification. You should check that the system meets its specified functional and non-functional requirements. Validation, however, is a more general process. You should ensure that the software meets the expectations of the customer.

Requirement Validation

- Requirement validation is ensure that the requirements actually define the system which customer wants. It has much in common with analysis as it is concerned with finding problems with the requirements.
- Requirement validation is important because errors in a requirement document can lead to extensive rework costs when they are subsequently discovered during development or after the system is in service.

- During the requirements validation process, different types of checks should be carried out on the requirements in the requirement document. These checks include:
  - Validity checks
  - Consistency checks
  - Completeness checks
  - Realism checks
  - Verifiability

- Validity checks
  - A user may think that a system is needed to perform certain functions. However, futher thought and analysis may identify additional or different functions that are required. A System has diverse users with different needs and any set of requirements is inevitably a compromise across the user community.
- Consistency checks
  - Requirements in the document should not conflict. That is, there should not be contradictory constraints or different descriptions of the same system function

- Completeness checks
  - The requirements document should include requirement which define all functions and constraints intended by the system user.
- Realism checks
  - Using knowledge of the existing technology, the requirements should be checked to ensure that they can actually be implemented.
- Verifiability
  - To reduce the potential for dispute between customer and vendor, system requirements should always be written so that they are verifiable. This means that a set of checks can be designed which can demonstrate that the delivered system meets the requirement.

Verification & validation Planning

- Verification and validation are expensive processes. For large system, such as real-time system with complex functional constraints, half of the system development's budget may be spent on verification and validation. Planning is needed to get the most out of inspections and testing and to control the costs of the verification and validation process.
- The planning of validation and verification of a software system should start early in the development stage.

The Test Plan

- The test plan is a mandatory document. You cannot test without one. For simple, straight-forward projects the plan does not have to be elaborate but it must address certain items.

The Test Plan

Table 4.5: Items Covered by a Test Plan

Component	Description	Purpose
Responsibilities	Specify who the people are and their assignments	Assigns responsibilities and keeps everyone on track and focused
Assumptions	Code and systems status and availability	Avoids misunderstandings about schedules
Test	Testing scope, schedule, duration and prioritisation	Outlines the entire process and maps specific test
Communication	Communication plan – who, what, when and how	Everyone knows what they need to know when they need to know.
Risk Analysis	Citical items that will be tested	Provides focus by identifying areas that are critical for success
Defect Reporting	How defects will be logged and documented.	Tell how to document a defect so that it can be reproduced, fixed and retested.
Environment	The technical environment, data, work area and interfaces used in testing.	Reduces or eliminates misunderstandings and sources of potential delay.

THANK YOU