Information Technology

FIT2001 – Systems Development
Seminar 10: Security & Testing

Chris Gonsalvez

Our road map

- Security
- Testing

- What are Information Systems?
- How do we develop them? Systems Development (SDLC) – key phases
- Traditional vs. Agile approaches to developing systems
- Some System Development roles and skills
- Understand the requirements gathering process
- Managing stakeholders
- Requirements gathering and documentation techniques
- Prototyping & Interface Design
- Detailed Design – Design Class Diagrams & Sequence Diagrams

At the end of this seminar you will be able to:

- Understand the importance of ensuring your information system is secure
- Understand the need for testing
- Describe the various types of tests, and explain how and why each is used

Lecture Outline - Security

1. Security – Why?

2. Security model - How?

System Design – Security: Why?

- Information systems need to be secure to be reliable
- Information systems are at risk from:
    - Human error (e.g. wrong data, accidental data deletion)
    - Technical errors (e.g. h/w failure, s/w crash)
    - Accidents and disasters (flood, earthquake, fire)
    - Fraud (e.g. deliberate attempts to corrupt or amend previously legitimate data and information)
    - Commercial espionage (e.g. unauthorised access of sensitive data)
    - Malicious damage (e.g. by internal employees, external hackers)

Information Security Model – How?

**The CIA security model** is the benchmark for evaluating information systems security**:**

- **Confidentiality** - *Preventing intentional or unintentional unauthorized disclosure*

- **Integrity** - *Prevent unauthorized data modifications*

- **Availability** - *Ensures reliable and timely access to data*

Security – How?

- Information systems can be made more secure by addressing the following factors through a range of controls:

  - *Prevention* of errors and breaches
  - *Detection* to spot security breaches
  - *Deterrence* to discourage breaches
  - *Data recovery* to recover lost data or information

Cuts down on losses and legal liabilities

important to every organisation

Security controls
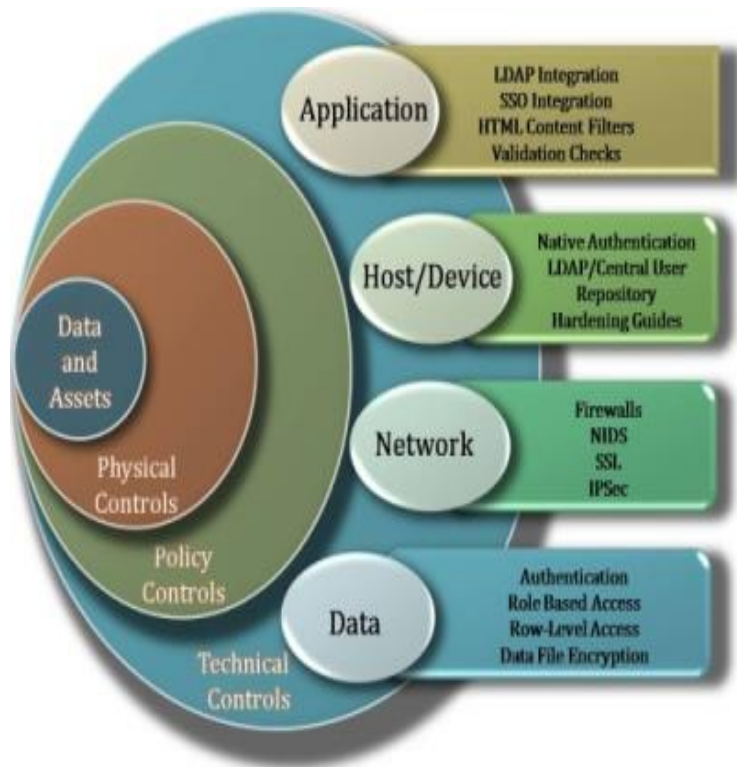
- ***Physical controls:*** *walls, locked doors, guards*
- ***Procedural controls:*** *managerial oversight, staff training, defined emergency response processes, fraud controls*
- ***Regulatory controls:*** *legislation, policy, rules of conduct*
- ***Integrity controls:*** *input controls, access controls, transaction logging, update controls, output controls, fraud controls*
- ***Redundancy, backup, and recovery controls***
- ***Technical controls:*** *cryptographic software, authentication and authorisation systems, secure protocols*

Lecture Outline - Testing

1. Testing – What?

2. Testing – Why?

3. Testing – How?

4. Testing in different methodologies

5. Test methods

6. Testing types

7. Test preparation principles

8. Test planning

9. Test cases

10. Automated testing tools

What is Testing.1?

- Testing represents the process of examining a component, subsystem, or system to:
    - determine if it contains any defects
    - uncovers design flaws and limitations
    - verify that it is 'fit for purpose' and meets the needs of the Business/End User
    - reduce the risk of failure

What is Testing.2?
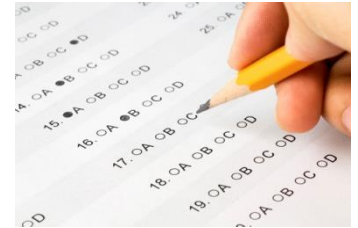
- The primary aim of testing is always:
  - to get the <u>system to fail</u> (i.e. to find errors)
  - rather than to confirm that the system is correct

- Various types of testing exist, and they can be conducted in several ways

- All systems must be tested for:
  - functional
  - non-functional aspects

Examples of Software Failures

- MARS Climate Orbiter
  - Simple conversion check was not performed
  - $125m spacecraft crashed in Martian atmosphere

- Guided-missile Cruiser
  - Use of '0' Zero as a data value in a formula
  - Shut down the ship's propulsion system
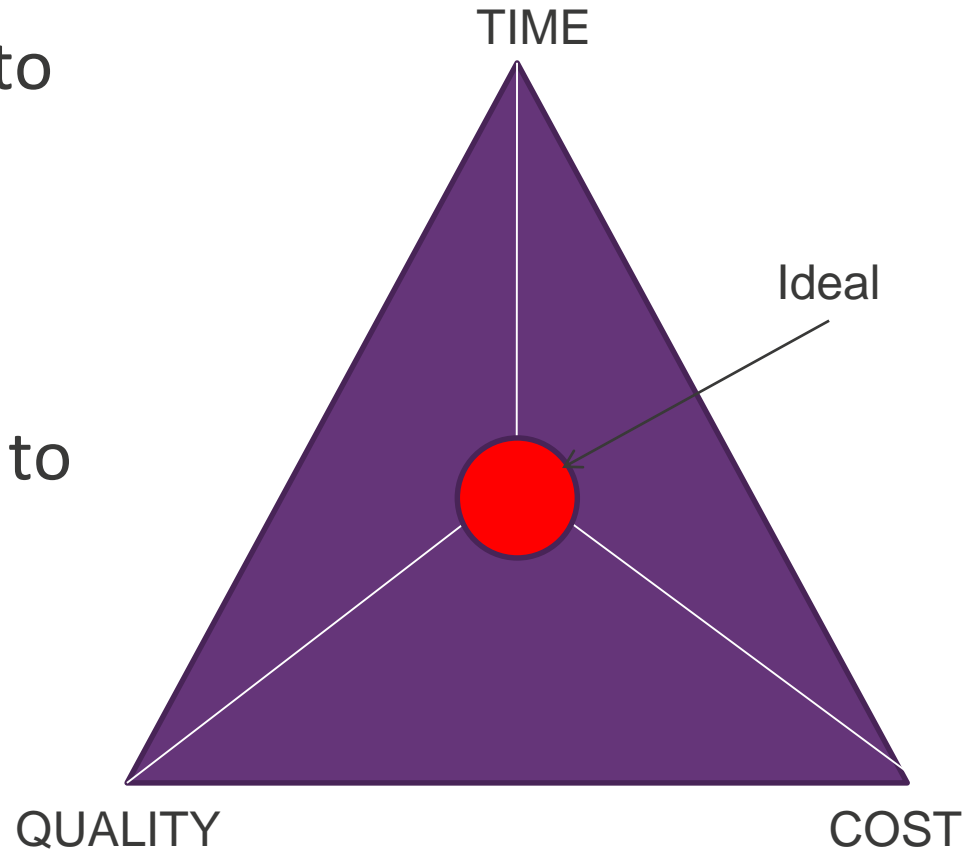 - Ship was dead in the water for several hours
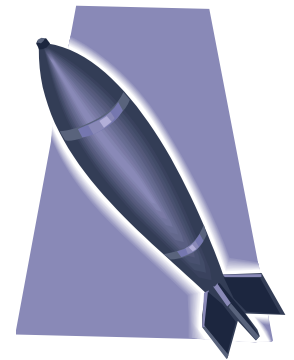
Cost of Errors
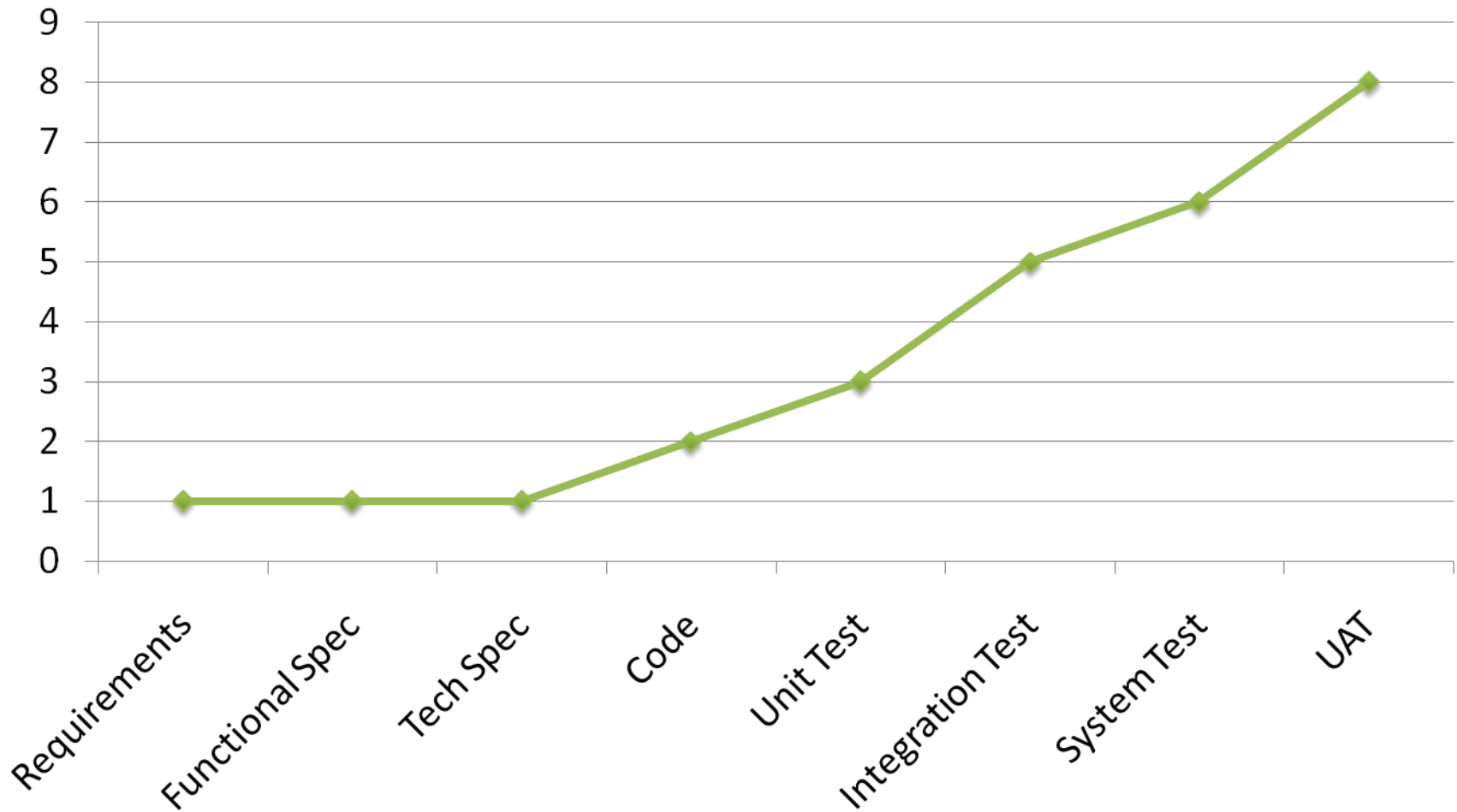
We can't find ALL errors ????

- Aim is to reduce the Risk to the business when the solution is implemented

  - Find as many critical problems as possible to reduce impact to clients and their customers
  - Metrics 80/20 rule

TIME

Ideal

QUALITY

COST

Test Process

- Plan - "What" to test
- Design - "How" you are going to test
- Schedule - "When" you are going to tes
- Execute - "What" was the result?

- When defects are found and fixed, must then retest

Test + Fix + Re-Test = Accepted

Waterfall

- Generally all testing and quality control points come late in the project (if time permits)

- Testing is usually scheduled late because people think that testing can only be done on code – NOT TRUE
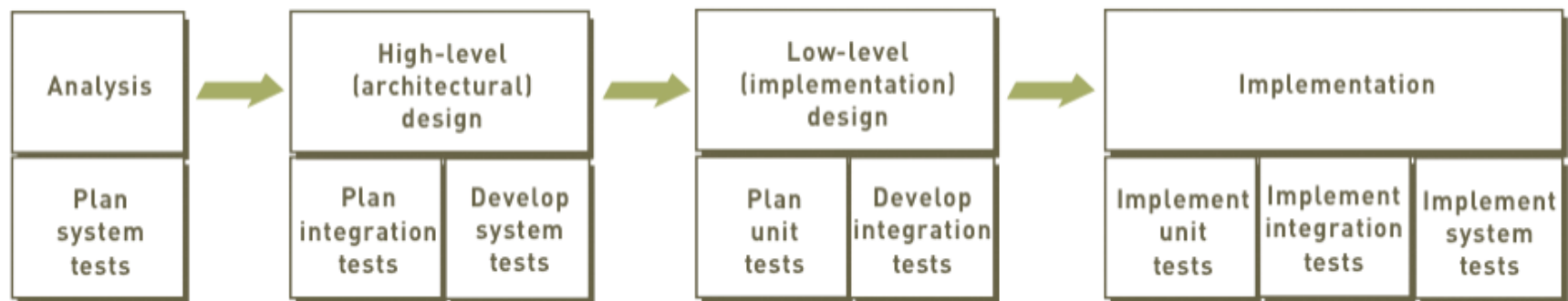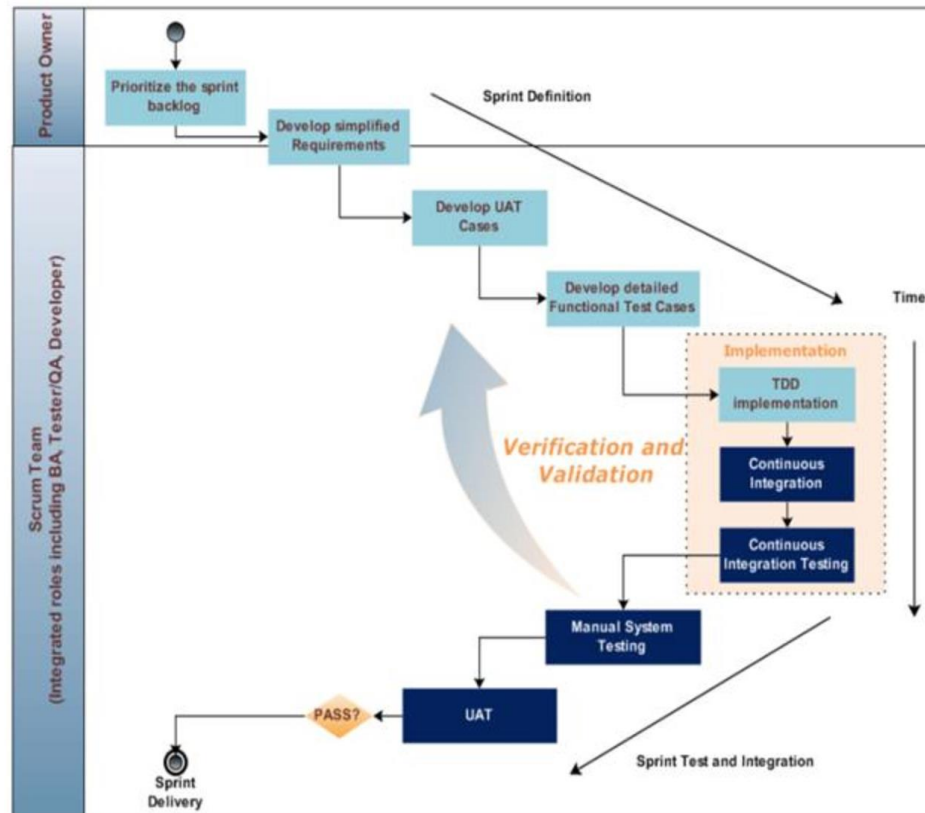
Agile

In **agile development**, *testing is integrated throughout the lifecycle*; testing the software continuously throughout its development.

- Does not have a separate test phase as such.
- Developers heavily engaged in testing, writing **automated repeatable unit tests** to validate their code.
- Supports the principle of <u>small, iterative, incremental releases</u>.
- Testing is done as part of the build, ensuring that all features are working correctly each time the build is produced.
- **Integration is done as you go**.
- The purpose of these principles is to keep the software in **releasable condition** throughout the development, so it can be shipped whenever it's appropriate.
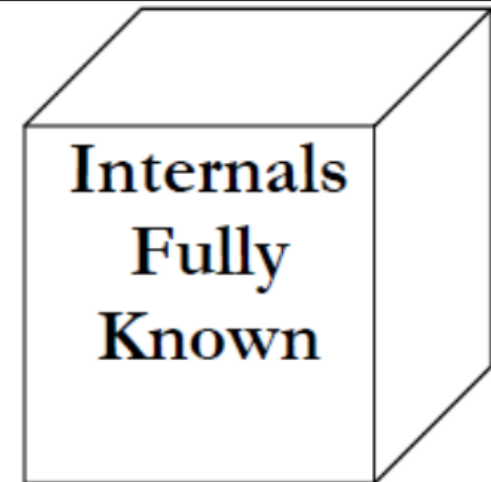
Agile: Testing process

Ref: http://www.codeproject.com/Articles/551161/Agile-Testing-Scrum-and-eXtreme-Programming#_Toc309143137

Test Methods

- The common methods include:

  - Black Box Testing
  - White Box Testing
  - Grey Box Testing
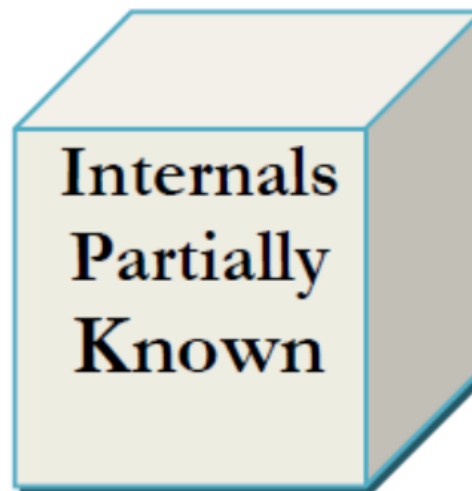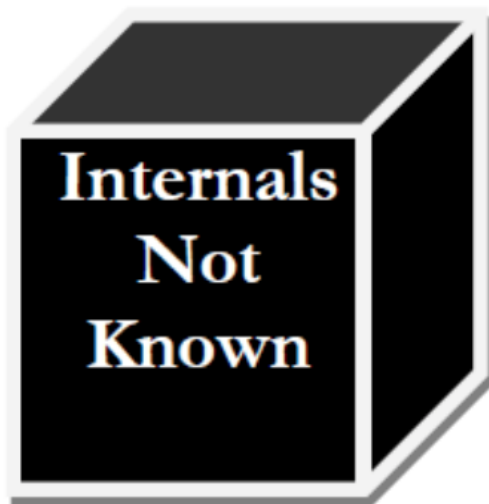
Black Box Testing
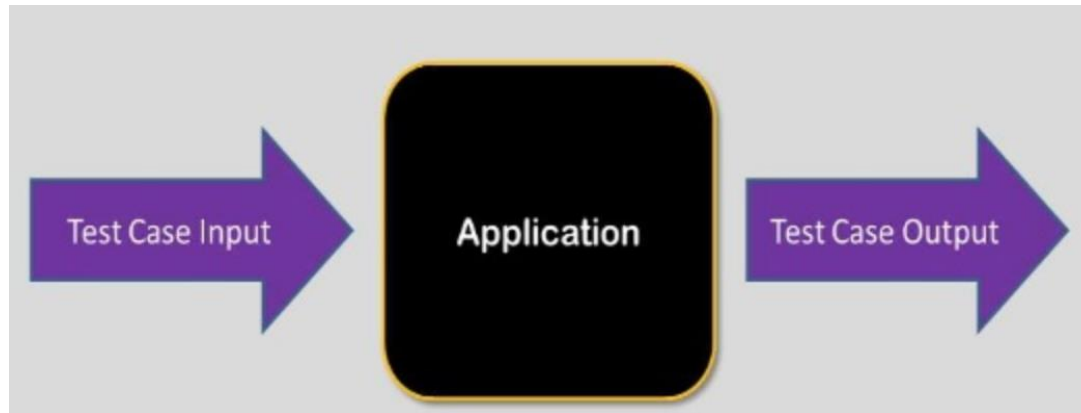
- – Testing, either functional or non-functional, without reference to the internal structure of the component or system (i.e. code not visible)
- – Typically executed in functional test phases i.e. Unit, Integration, System or Acceptance test phases
- – Can be independently tested

White Box Testing
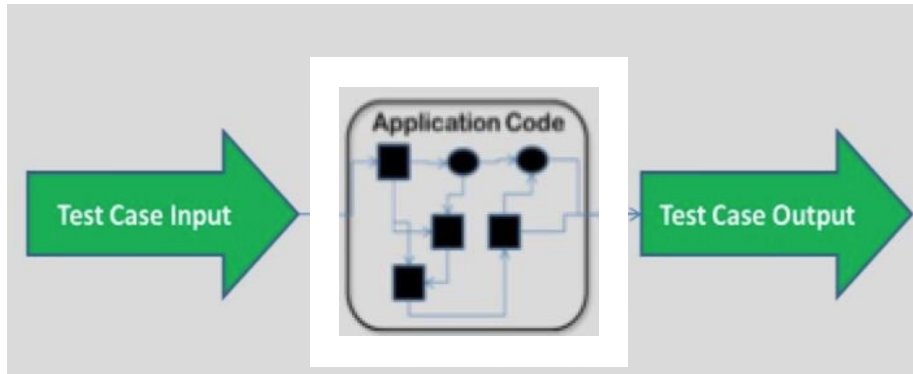
– Uses an internal perspective of the system to design test cases based on internal structure.
– It requires programming skills to identify all paths through the software
– Due to internal perspective, maximum coverage of a scenario is possible

Grey Box Testing

- – Is a combination of black box and white box testing
- – We look into the "box" being tested just long enough to understand how it has been implemented. Then we close up the box and use our knowledge to choose more effective black box tests.
- – Increase testing coverage

Comparison between the Three Testing Types

	Black Box Testing	Grey Box Testing	White Box Testing
1.	The Internal Workings of an application are not required to be known	Somewhat knowledge of the internal workings are known	Tester has full knowledge of the Internal workings of the application
2.	Also known as closed box testing, data driven testing and functional testing	Another term for grey box testing is translucent testing as the tester has limited knowledge of the insides of the application	Also known as clear box testing, structural testing or code based testing
3.	Performed by end users and also by testers and developers	Performed by end users and also by testers and developers	Normally done by testers and developers
4.	-Testing is based on external expectations -Internal behavior of the application is unknown	Testing is done on the basis of high level database diagrams and data flow diagrams	Internal workings are fully known and the tester can design test data accordingly
5.	This is the least time consuming and exhaustive	Partly time consuming and exhaustive	The most exhaustive and time consuming type of testing
6.	Not suited to algorithm testing	Not suited to algorithm testing	Suited for algorithm testing
7.	This can only be done by trial and error method	Data domains and Internal boundaries can be tested, if known	Data domains and Internal boundaries can be better tested

Testing Types

- The common testing types include:

    – Functional Testing
    - Unit, Integration, System, Acceptance
    – Regression Testing
    – Static & Dynamic Testing
    – Performance, Load & Stress testing
    – Usability testing
    – Accessibility testing
    – Security testing
    – Backup & Recovery testing

Functional Testing

- Testing whereby the system is tested against the requirements specification
- Typically, functional testing involves the following steps:
  - Identify functions that the software is expected to perform.
  - Create input data based on the function's specifications.
  - Determine the output (expected results) based on the function's specifications.
  - Execute the test case.
  - Compare the actual and expected outputs.

Unit Testing

- The testing of individual software components at a coding level

- Unit testing is usually done by the developers themselves, before the module is handed over for integration with other modules of the same solution

- Also known as String, Component or Module testing

Integration Testing

- Integration testing is an interim level of testing applied between unit and system test

- Tests the interaction and consistency of integrated components

- Allows partial system level test without waiting for all the components to be available

System Testing

- System testing represents testing of the complete system or functionality to be delivered

- System testing is typically performed after all unit and integration testing has been completed

- Includes performance / stress testing

  - Response time, Throughput

User Acceptance Testing (UAT)
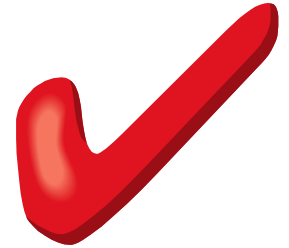
- Formal testing with respect to user needs, requirements and business processes

- Conducted to determine whether or not a system satisfies the acceptance criteria – written for user stories

- Enables the user/client to determine whether or not to accept the system

- Acceptance tests often be run by the client themselves with support from the project team

- This is the last stage of testing prior to deployment

MONASH University

Regression Testing

– Testing of a previously tested program following modification to ensure that defects have not been introduced or uncovered in unchanged areas of the software, as a result of the changes made.

– Can be performed in all test phases

Performance, Load & Stress

- Performance: The process of testing to determine the performance and efficiency of a software product or infrastructure.
- Load: Testing to evaluate whether a system can handle large quantities of data simultaneously
- Stress: Testing conducted to evaluate a system or component at or beyond the limits of its specified requirements

Static Testing

- – is a form of software testing where the software isn't actually used.
- – It is generally not detailed testing, but checks mainly for the sanity of the code, algorithm, or document.
- – It is primarily syntax checking of the code or manually reading of the code or document to find errors.
- – This type of testing would primarily be used by the developer who wrote the code, in isolation.

Accessibility Testing

– Testing to determine the ease by which users with disabilities can use a component or system.
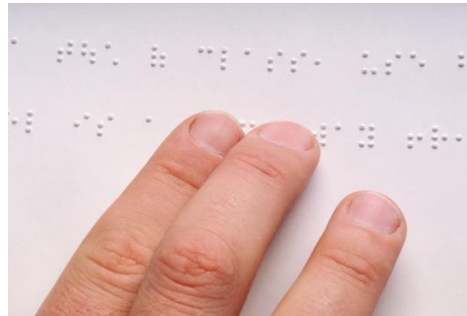
Usability Testing

- Testing to determine the extent to which the software product is understood, easy to learn, easy to operate and attractive to the users under specified conditions. [After ISO 9126]
- Typically executed in the later phases of testing when system is more stable i.e. System Test

Security Testing

– The process to determine
   that a System protects data
   and maintains functionality as intended.   The six
   basic security concepts that need to be covered
   by security testing are: confidentiality, integrity,
   authentication, authorisation, availability and
   non-repudiation.

Backup & Recovery Testing
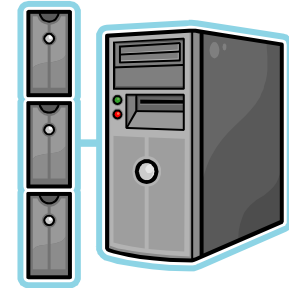
- Taking a backup is the process of taking a copy of an application, its data and environment, such that it may need to be recovered (or restored) in the event of a system failure
- Recovery is the process of reverting to the backup copy of an application, its data and environment such that normal business can be resumed.
- Backup & recovery testing is disruptive so is therefore executed during the quiet times of the later phases of testing

Test Preparation Principles

- Starts early in the project lifecycle

- Produce a Test Strategy Plan first

- Review all documentation

  - Business Requirements
  - Functional Requirements

- Design test requirements/conditions

- Design test cases (manual test) or scripts (automated test) to meet the test requirements

- Prepare the test environment(s) and required data

Test plan & test cases

▪ Testing needs to be planned in detail

▪ A **test plan** is prepared that identifies:

  – how and when testing will be done, how long it will take

  – types of tests, things to be tested/not tested

  – resources, staff, facilities, tools, training needed

  – schedule

  – risks

▪ **Test cases and comprehensive test data** need to be prepared

  – time required for this is often underestimated .. this results in incomplete/inaccurate tests, inadequate test data

test documents

Test Plan — Strategic: Created for the entire Project

Test Design 1 — Created per Package

Test Case 1 — Tactical: Created per Element (like a Use case or a Class)

Test Case 2

Test Case 3

example documents
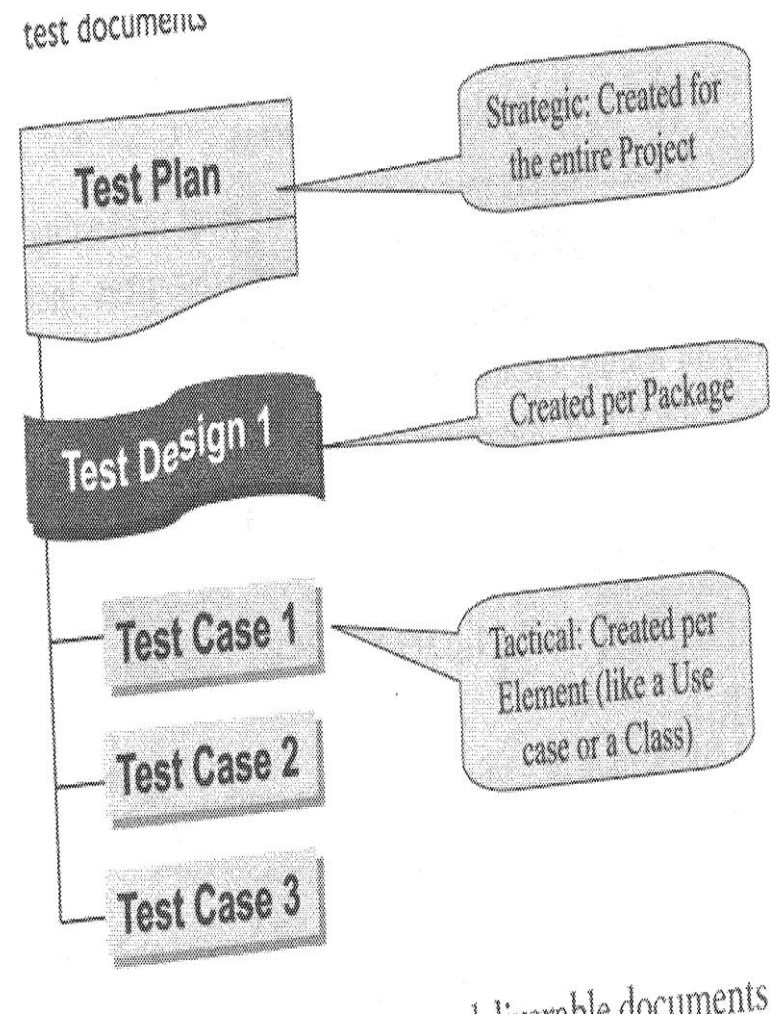
Test cases

- A test case is a formal description of
  - Starting state
  - Events to which software responds
  - Expected response or ending state

- Important part of testing is specifying test cases and test data

- Analysis phase documentation is useful in preparing test cases (use-case driven)

- Test data is defined to be used with a test case

Developing test cases

- Test cases are developed for each use case
  - Requirements traceability

- Test cases represent usage scenarios
  - Test for normal operations
  - Test for significantly different operations

- Test cases comprise the actions to be taken in the steps of the scenario, and the expected system responses

Developing test cases

Test cases are developed for each use case

Test cases can be prepared once the detail of the use case is defined.

The following tasks are involved in developing test cases from use cases:

- **Task 1**: Identify relevant scenarios (event flows)
    - basic flow and alternative flows
- **Task 2:** Identify the variables for each use case step
- **Task 3:** Identify the significantly different options for each INPUT variable
- **Task 4:** Combine options to be tested to form test cases
- **Task 5:** Assign values to variables

Illustrated with an examples in the next few slides

Use case 'Recording traffic ticket'

Use Case Name:	Record traffic ticket	
Scenario:	Record traffic ticket	
Triggering Event:	Officer sends in new ticket	
Brief Description:	The officer gives the traffic ticket to the clerk. Using the information on the traffic ticket, the clerk first verifies the officer by entering the badge number. Then, the clerk verifies the driver information by entering the driver's license number. Finally, the clerk enters the ticket information.	
Actors:	Clerk	
Stakeholders:	Manager Officer	
Preconditions:	The officer must exist. The driver must exist.	
Postconditions:	The ticket must exist and be associated with the driver, the officer, and a court.	
Flow of Events:	Actor	System
	1. Clerk enters officer badge number.	1.1 System reads officer information and displays the name.
	2. Clerk enters driver's license number.	2.1 System reads the driver information and displays the driver name and address.
	3. Clerk enters ticket information.	3.1 System displays ticket and driver information.
Exception Conditions:	1.1 Officer is not found. 2.1 Driver is not found.	

Task 1: Identify basic and alternative event flows

- <u style="color:red">Basic flow</u> in the recording traffic ticket use case

  1. Clerk enters officer badge number. (B1)
     - system reads officer information and displays name
  2. Clerk enters driver's licence number. (B2)
     - system reads driver information and displays driver name and address.
  3. Clerk enters ticket information. (B3)
     - system displays ticket and driver information.

- Alternative flows
     - 1.1 Officer is not found (A1)
     - 2.1 Driver is not found. (A2)

Task 1: Identify basic and alternative event flows
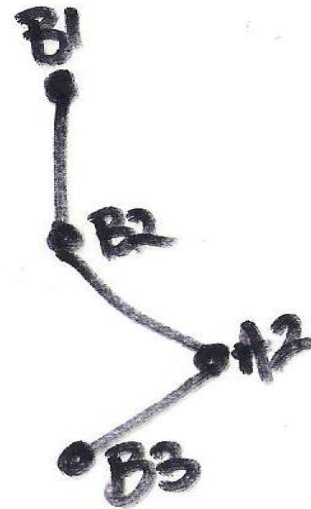
- Basic flow is a <u>straight line</u> down – sunny day scenario
- While alternative flows are usually <u>deviations from straight line</u>.
- Four possible scenarios are indicated. There are more scenarios than the no. of alternative flows
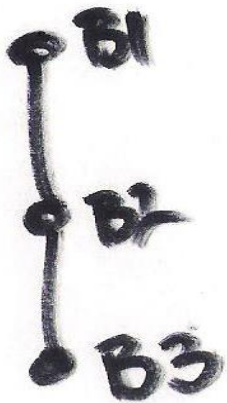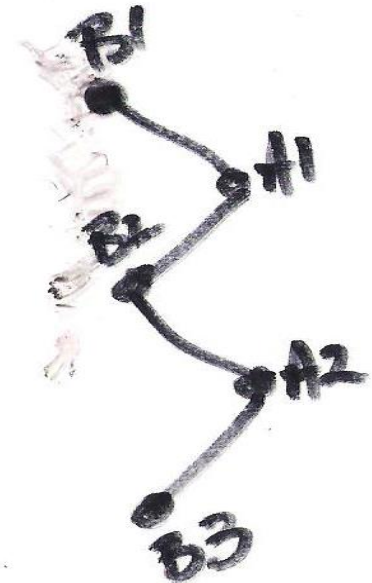
Scenario 1:
Basic flow
(straight line)
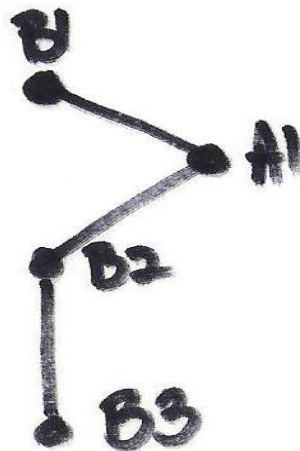
Scenario 2:
Alternative
flow A1

Scenario 3:
Alternative
flow A2

Scenario 4:
Alternative
flow A1,A2

Exhaustive Testing

- This means to test every possible path through the system with every possible data combination.

- This type of testing is not possible.

- Simple Example

  - 20 different inputs, all independent
  - Each value possible 4 variables
  - Exhaustive testing would mean 4^{20} tests
  - This equates to 1,099,511,627,776 test cases

- Other options are available

  - Equivalence Partitioning, Boundary Value Analysis, Sampling and Risk Based Testing

Task 2: Identify the variables at each step

Steps	Variables	Input variable
B1	badge number (input), officer name ….	badge number (input)
B2	licence number (input), driver name, driver address ….	licence number (input)
B3	ticket no, date of ticket, time of ticket (input)	ticket no, date of ticket, time of ticket (input)
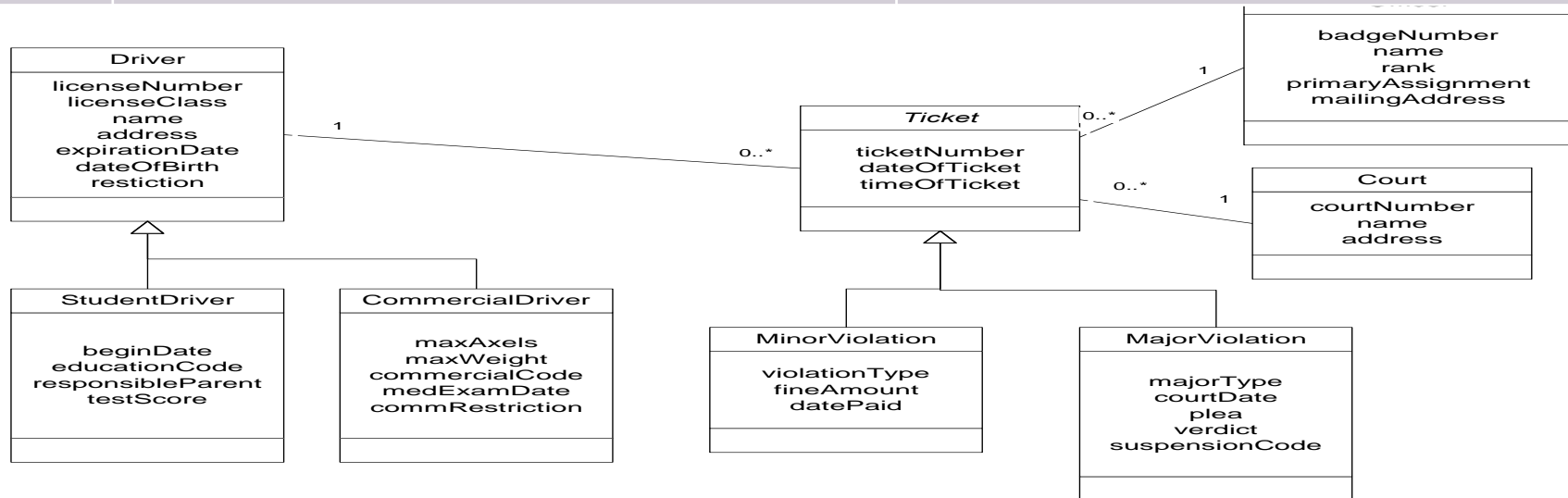
- **Use the class model to identify variables**

Task 3: Identify the significantly different options for each INPUT variable

Equivalence Partitioning:

- Based on the premise that by testing one value within a class is representative of all values within that field

- Only one data value from a class needs to be tested as it is considered as representative of all values within that partition
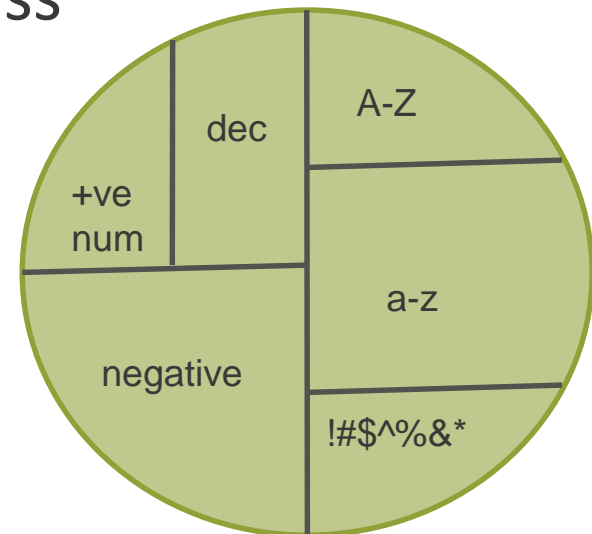
Task 3: Identify the significantly different options for each INPUT variable

Badge no. (integer, 8 digits)	Licence number (integer, 10 digits)	Ticket No. (integer, 6 digits)	Ticket date	Ticket time
• must be greater than 0 • Leading 0s must be entered • Maximum value 99999999 • Value should be in the database (Badge number is a key) • No spaces • Only numeric	• must be greater than 0 • Leading 0s must be entered • Maximum value 99999999 • Value should be in the database (License number is a key) • No spaces • Only numeric	• must be greater than 0 • Leading 0s must be entered • Maximum value 99999999 • No spaces • Only numeric	• Format must be DDMMYY • No spaces • Only numeric	• Format must be HHMM • No spaces • Only numeric

Task 3: Identify the significantly different options for each INPUT variable

Boundary Value Analysis:

- Extension of Equivalence, concentrates on maximum and minimum values

- To validate the boundary, tests must be carried out on either side of the boundary with the boundary values increased and decreased

Significantly different option is one when an input variable/action …

- Triggers a different (alternative) flow
- Triggers an error message
- Changes the user interface
- Causes different options to show in a drop-down list
- Is an input to a business rule
- Border condition
- Changes a default
- Entry format is not clearly defined
- International format differences (dates, currencies)

Options identified: Badge number

- Blank entry, zero entered, negative number, leading 0s not entered, correct number in database, correct number not in database, number too large, alpha only entry, alpha-numeric entry

Task 4: Combine options to form test cases

Step	Input variable	Test Case 1 (TC1)	Test Case 2 (TC2)	Test Case 3 (TC3)	Test Case 4 (TC4)
B1	Badge no	Valid badge no	Valid badge no, but not stored in database	Invalid badge no	Valid badge no
B2	Licence No.	Valid licence no.	Valid licence no.	Valid licence no.	Valid licence no. but not stored in db
B3	Ticket info.	Valid ticket info.	Valid ticket info.	Valid ticket info.	Valid ticket info.

Task 4: Combine options to form test cases (cont.)

Step	Input variable	Test Case 5 (TC5)	Test Case 6 (TC6)	Test Case 7 (TC7)	Test Case 8 (TC8)
B1	Badge no	Valid badge no	Valid badge no,	Valid badge no	Valid badge no
B2	Licence No.	Invalid licence no (< 10 digit).	Invalid licence no. (1 digit)	Valid licence no.	Valid licence no.
B3	Ticket info.	Valid ticket info.	Valid ticket info.	Invalid ticket info.(wrong ticket no)	Invalid ticket info. (invalid date)

Task 5: Assign values to variables to TC1

Step	Action	Value	Expected result	Actual result	Pass/ Fail	Comments
B1	Enter a valid badge number	98239289	Officer name "Jack Smith" is displayed	Officer name "Jack Smith" is displayed	Pass	
B2	Enter a valid driver's licence number	8112823	Driver name "Jane Doe" is displayed	Driver name "Wendy Smith" is displayed	Fail	
B3	Enter valid ticket information	12345, 24052013, 1505	Error message displayed – date wrong format	Ticket information is displayed	Fail	

Automated testing tools

- Document and create test cases

- Automated running of test cases
  - Including GUI scripts
  - Load testing (might run on hundreds of client machines to test typical load)

- Recording of results
  - Comparison of actual to expected results
  - Comparison of different test trials

Workshop Preparation

We will cover this material in Week 12

The Workshop for Week 11 will be focussed on Exam Revision

Thanks for watching

Resources:

Satzinger, J. W., Jackson, R.B., Burd, S.D. and R. Johnson (2016) Systems Analysis and Design in a Changing World,7th Edition, Thomsen Course Technology

- Security: Chapter 6, pp. 168 - 179

- Testing: Chapter 14, 446 - 453