CH8-Software Testing

Keywords	Definition
Test cases	Inputs to the test and the expected output from the system (the test results)
Test data	**inputs** that have been devised to test a system.
Stress testing	form of performance testing where the system is deliberately overloaded to test its failure behaviour
Equivalence partition	Dividing everything based on "equivalence" **For example**, all data less than 1 gb memory

* Program testing goals

- The first goal leads to validation testing
  - To demonstrate to the developer and the customer that the software meets its requirements
- The second goal leads to defect testing
  - To discover situations **faults** or **defects** in which the behavior of the software is incorrect

* Advantages of inspections

- During testing, errors can mask (hide) other errors
- Incomplete versions of a system can be inspected without additional costs
- As well as searching for program defects
- an inspection can also consider broader quality attributes of a program

* Stages of testing:

1. **Development testing:** where the system is tested during development to discover bugs and defects.
2. **Release testing**: where a separate testing team test a complete version of the system before it is released to users.
3. **User testing:** where users or potential users of a system test the system in their own environment

* What tests should be included in object class testing?

- Tests for all objects in isolation.
- Tests that set and access all object attributes.
- Tests that force the object into all possible states

* Automated test components:

- **A setup part**: initialize the system with the test case
- **A call part**: call the object or method to be tested.
- **An assertion part**: compare the result of the call with the expected result

* General testing guidelines:

1. Force all error messages.
2. Force invalid outputs.
3. Try to cause overflow.
4. Repeat the same input numerous times.
5. Force computation results to be too large or too small.

* Interface testing types:

- Parameter interfaces
- Shared memory interfaces
- Procedural interfaces
- Message passing interfaces

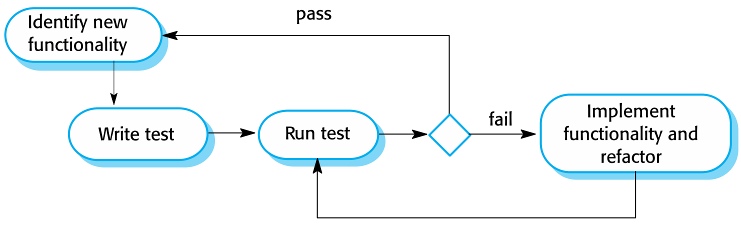
* Classes of interface errors:

1. interface Misuse
2. interface Misunderstanding
3. Timing Errors.

* Principal concerns of system testing

- Testing the interactions between the components and objects that make up the system.
- Testing reusable components and systems to check that they work as expected when integrated into the system.

* Test-driven development (TDD) process activities:


Identify new functionality
pass
Write test
Run test
fail
Implement functionality and refactor

a) Identify increment of functionality required
b) Design tests for this functionality and implement as executable programs.
c) Run test along with other implemented tests. The test will fail.
d) Implement the functionality and re-run the test. Iterate until the test works.
e) Move on to implement the next chunk of functionality

* Benefits of test-driven development:

- **Code coverage**
  - Every code segment that you write has at least one associated test so all code written has at least one test.
- **Regression testing**
  - A regression test suite is developed incrementally as a program is developed.
- **Simplified debugging**
  - When a test fails, it should be obvious where the problem lies. The newly written code needs to be checked and modified.
- **System documentation**
  - The tests themselves are a form of documentation that describe what the code should be doing

* Why stress testing is useful?

To stress the system and bring defects to light that might not normally be discovered.

* Types of user testing:

- **Alpha testing:** where users work with the development team to test the software as it is being developed.
- **Beta testing:** where the software is released to selected users for testing before the formal system release
- **Acceptance testing**: where customers test a system to check that it is ready for deployment

Verification	VS	Validation
Are we building the **product right**		Are we building **the right product?**
The software should conform to its specification		The software should do what the user really requires

Software inspections	VS	Software testing
analysis of the **static system** representation to discover problems		exercising and observing product behaviour
static verification		dynamic verification

Unit testing	Component testing	System testing
focus on testing the functionality of objects or methods.	focus on testing component interfaces.	focus on testing component interactions