**Week 9 software testing**

**What is software testing?**

An investigation conducted to provide stakeholders with information about the quality of the product or service under test.

* Make software fail
* Find bugs and fix failures
* Try to generate more reliable software
* Divide a large program into smaller ones and conquer them

Software testing can be executes as soon as executable software is completed (even partially complete)

Waterfall model testing: In most functional models of project development, testing occurs after system requirements have been defines and then implemented in testable programs. (beta stage)

Agile model testing: Another practice is to begin software testing at the same moment as development begins, creating a continuous process of testing until project completion.

Software testing involves the execution of a software component opr system component to evaluate properties of the software.

These components test:

* Program meets requirements as laid out in design
* Responds correctly to all inputs
* Performs functions within an acceptable time
* Is useable
* Can be run in intended environments
* Achieves the general result desired

**Types of defects/ failures**

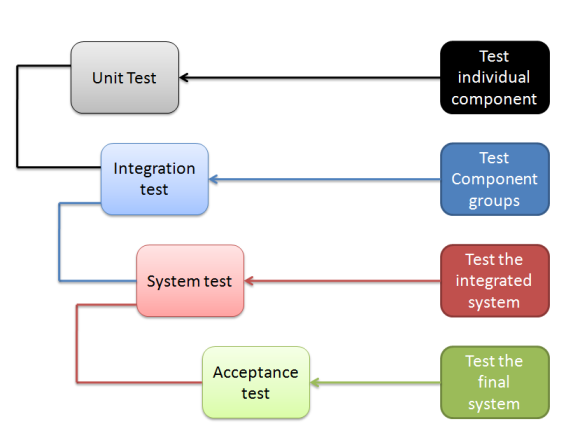
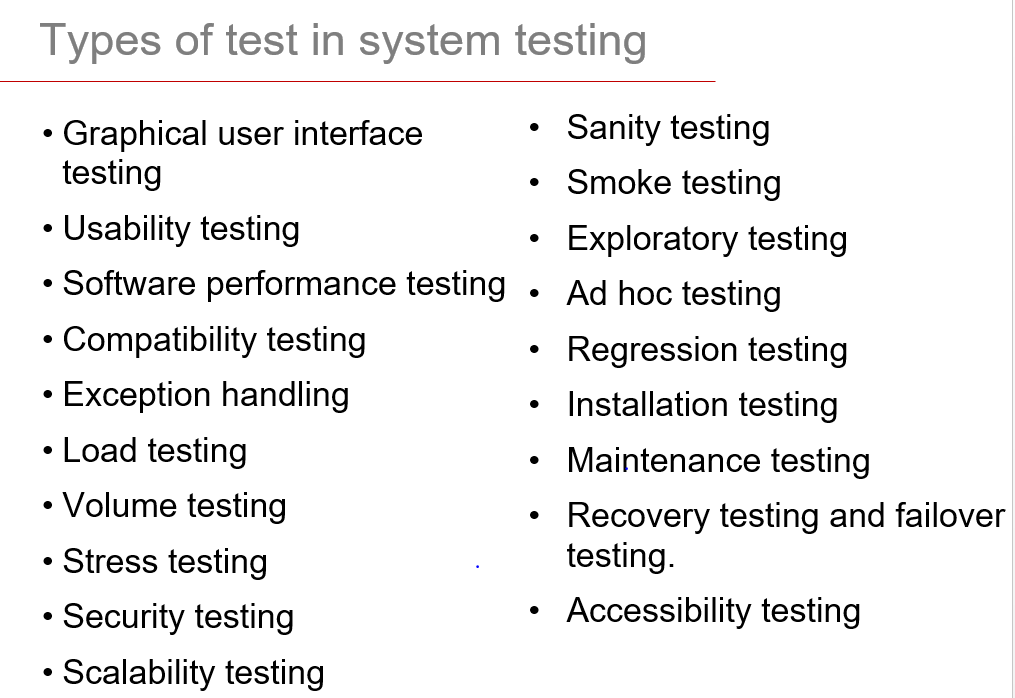
Coding errors

Requirement gaps

Perception gaps in development team/ development stages

**Testing methods**

* Static: reviews and inspection of code
* Dynamic: executing code with test cases
* White-box: test internal structure or frunctions
* Black-box: examining functionality without any knowledge of interal implementation, without seeing source code.
* Gray-box: combining both white-box and black-box to paint an entire testing picture of the program

**Testing levels Types of tests**

**Creating testable software**

* Clean and neat code
* Refactor code- change internal structure but not functionality
* Clear understanding of the function of a module and how it interacts with other modules
* Avoid large amounts of global variables
* Avoid large amounts of references and pointers
* Support unit testing
* Support fault injection
* Large amount of assertions

**Coverage testing**

Testing is based on samples

* Cannot run all possible tests
* Need to know that all areas of design are tested
* A test suite is not measured by its size

Solution: coverage analysis

Code coverage is a measure used to describe the degree to which the source code of a program is executed when a particular test suite runs. The main ideas behind coverage are to systematically create a list of tasks (testing requirements) and check that each task is covered during the testing.

Goals:

* Measure the quality of test sets
* Supplement test specifications by pointing to untested areas
* Help create regression suites
* Provide a stopping criteria for unit testing

Values:

* Quantitative measurement of testing efforts
* Directs future efforts
* Demonstrates redundancy in test cases
* Can be used as entry/exit criteria between test phases

Pros:

* Easily estimating software reliability from test outcomes. Test inputs are randomly generated according to an operational profile, and failure times are recorded.
* Do not have bias: unlike manual testing, it does not overlook bugs because there is misplaced trust in some code.
* Use of random test inputs may save some of the time and effort

Cons

* Semantically redundant inputs
* Can only find basic bugs
* Effective oracle is seldom available