Lec 4 Validation and Verification

Objectives

- [ ] Recall the fundamentals of Software validation and verification
- [ ] List items of the testing toolbox
- [ ] Introduce ISO/IEC/IEEE 29119
- [ ] Study different testing approaches and associated coverage criteria

---

IEEE Standard 1012 → standard for System, Software, and Hardware **Verification and Validation (V&V)**
Verification & Validation → determine whether the development products of a given activity conform to the requirements of that activity and whether the product satisfies its intended use and user needs

- They are separated but interrelated, complementary technical processes
- Purpose → **build quality into the system**

Verification → Are we building the product correctly?

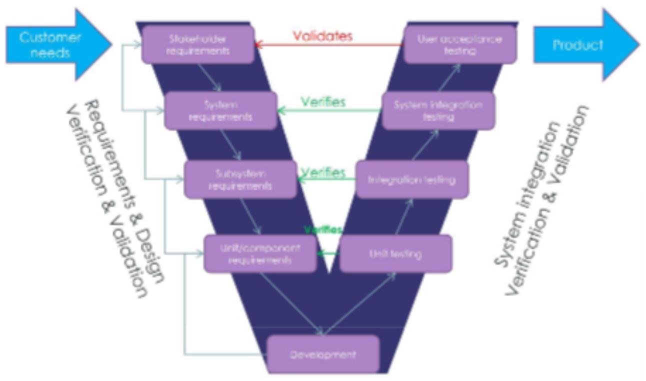
- During the life cycle activity
- Evaluation of work products (not final product) during a work phase, to make sure they meet the requirements for that phase.
- **Evaluate** → plan, requirements specs, design specs, code, test cases
- **Using** → tests, reviews, walkthroughs

Validation → Are we building the correct product?

- At the end of the product life cycle activity
- Evaluating software (product) during or at the end of development to see if it meets specified business requirements
    - Does it match the user's needs?
    - Does it fulfil its intended use?
- **Early validations**
    - **Evaluations** → specifications
    - **Using**: Reviews, scenarios, questionnaires, MVP with user feedback
- **Final/work product validation**
    - **Evaluations** → software product/ version
    - **Using** → Testing (e.g. acceptance testing)

V&V in the V-Model

- SDLC model where the process executes sequentially in a V-shape. Based on the association of a testing phase for each corresponding development stage.



- Displays the verification process being associated with the phases during the development phase, while the validation process being at the end of the phase

Testing

- Systematic steps taken to:
  - Push and prod the system to see if it behaves as expected
  - Check if the system actually does what it is meant to do
  - Understand/stress the system (e.g. what are the limits)
  - Experience how a user experiences the system (and compare it to our expectations)

  Purpose
  - **Quality Control** → Find issues/ defects/ bugs

  Expected Results
  - PASS/FAIL
  - However results can be TP, TN, FP, FN

Proving the absence of defects through testing → exhaustive testing is impractical

- Cannot fully trust test results
- Have to accept we are coding bugs
- Define AC (including coverage criteria)
- We have to deal with both (bugs and criteria) in a smart way

Bug Report:

- Check if a similar bug has been reported
  - Yes, check if the bug is the same
    - Yes, just rack and ask about the status
    - No, may be useful to reference similar bug reports you found in the “context” part of your bug report
- Make it simple as possible to re-produce
- Explain how to reproduce it
- Provide (minimalistic) source code and data → as simple as possible
- Provide meaningful details about your environment

As a **developer** → provide fix, a **set of tests** to show the bug is fixed

Hypothesis Testing

1. **Model + Conjecture**
   - Logging in is a difficult thing to achieve well
   - I have a feeling there is a flaw with logging in – I don’t think I even need to register to be able to log in
2. **Hypothesis**
   - That the logging in utility is compromised (insecure logins are possible)
3. **Systematic testing of hypothesis:**
   - Using the “back button” after logging out
   - Refreshing page
   - Seeing if passwords are sent in plain text
   - Logging in as admin, password, 1234
   - Running an SQL injection query

Testing and Automation → “Automated testing” is not testing

- Testing → human process of thinking about how to verify/falsify a hypothesis
- Automation → technique used to help with making the testing process easier

Software Under Test (SUT) → A set of automated tests (software), aimed at finding flaws in another software.
Activities of test software development:

- Requirement Gathering
- Requirement Analysis
- Test Planning
- Test Design
- Test Execution
- Defect Tracking
- Defect Resolution
- Test Closure

Testing ToolBox
Types of Testing techniques:

- **Manual** → performed by a human being
- **Automated** → performed by a software
- **Static testing techniques**
    - Look at the (static) code or documentation
    - Static code analysis, cross-document traceability analysis, reviews
- **Dynamic testing techniques**
    - Forcing failures in executable items

Types of Dynamic & Manual Testing Techniques

	Advantages	**Disadvantages**
Scripted Testing	Testing is repeatable and can simply be run again Scripted test cases can be traced back to requirements Test cases can be retained as reusable artefacts for the current and future project	Time-consuming and costly than unscripted test execution Test cases defined prior to test execution are less able to adapt to the system as it presents itself Can be less stimulating for the test executors as most of the analysis work has been completed
Unscripted testing	Testers are not constrained by a script and can follow ideas generated by test execution in real time Can tailor “Test Design and Implementation” and “Test Execution” to behaviour of the system in real time Quickly explore test item	Tests are not generally repeatable Ability to apply a wide variety of test design techniques as required Provide little or no record of what test execution was completed

Testing Toolbox
Black box Testing → aka Specification-based testing

- Testing without looking internally
- E.g **System testing** (functional) → Testing the system
    Regression, performance, sanity, smoke, installation, GUI,...

White box Testing → aka Structure-based testing

- Internal testing
- E.g **Unit testing** (functional) → individual units are tested

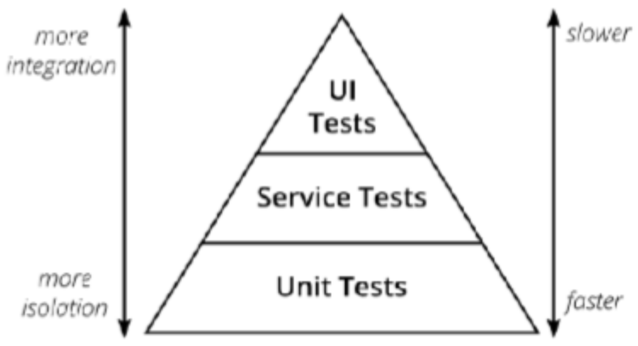
Grey box testing
- Some combination of black and white techniques
- E.g **Integration Testing** (functional) → Testing interface between two modules (when they are integrated), API testing

Regression Testing
- Selective testing of a system or component that has previously been tested to verify that modifications have not caused unintended side-effects.

End-to-end testing
- Scenario testing
- Actual user interaction for the whole transaction
- Uses actual data and simulated “real” settings

Acceptance Testing
- Formal tests
- Customer decides if the requirement is accepted
- Includes End-user Acceptance Testing (UAT), Business Acceptance Testing (BAT), Regulations/Standards Acceptance Testing (RAT), Alpha/Beta Testing

Test Principles



The test pyramid

Testing methods

Test Driven Development (TDD) → Write tests first, then features

Behaviour Driven Development (BDD) → focused on expected behaviour, therefore on requirements/specifically
- Write automated tests from the specifications
- Semi-structured languages for specifications
    - Given, When, Then