Chapter 6: Testing

- "I can only test my code"
    - Not true
    - What can be tested
        - Code
        - Design Artifacts
        - Pretty much everything we make
        - The testing itself

- Testing
    - Two basic types
        - Execution-based testing
        - Non-execution-based testing

    - V & V
        - Verification: Is the software doing things right?
            - Determine if the workflow was completed correctly
            - Ensures the software is designed to deliver all functionality according to the requirements

        - Validation: Is the software doing the right things?
            - Determine if the product as a whole satisfies the requirements
            - Ensures the functionality, as defined by the requirements, is the intended behavior of the software

    - Software Quality
        - Not "excellence"
        - The extend to which the software satisfies its specifications
        - Every software professional is responsible for ensuring that his or her work is correct
        - Quality must be built in from the beginning

    - Software Quality Assurance
        - The members of the SQA group must ensure that the developers are doing high-quality work
            - At the end of each workflow
            - When the product is complete

        - In addition, quality assurance must be applied to the process itself
            - Ex. Standards

        - Managerial Independence

        - There must be managerial independence between the:
            - development group
            - SQA group
        - Neither group should have power over the other
        - More senior management must decide whether to
            - Deliver the product on time but with faults, or
            - Test further and deliver the product late
        - The decision must take into account the interests of the client and the development organization
- Non-Execution-Based Testing
    - Underlying principles
        - We should not review our own work
        - Group synergy (developers and SQA)
    - Testing software without running test cases: Carefully read through it
        - Review software: Carefully read through it
        - Verify all artifacts produced from each workflow for each increment
    - Walkthroughs
        - A walkthrough team consists of four to six members
        - It includes representatives from
            - The current workflow team
            - The next workflow team
            - The SQA team
        - The walkthrough team is chaired by the SQA team
        - In a walkthrough, we *detect* faults, not correct them
            - A correction produced by a committee is likely to be of low quality
            - the cost of a committee correction is too high
            - Not all items flagged are actually incorrect
            - A walkthrough should not last longer than two hours
            - There is no time to correct faults as well
    - Inspections
        - Inspection Team
            - Moderator
            - Member of current workflow team
            - Member of next workflow team
            - Member of SQA team
        - An inspection has five formal steps

            - Overview of artifact to review
            - Preparation of list of types found
                - Statistics of fault types
                - Concentrate on areas where most faults have occurred
            - Inspection
                - Meticulous walkthrough of artifact
            - Rework
                - Responsible for artifact to resolve faults
            - Follow-up
        - Fault Statistics
            - Faults are recorded by severity
                - Not a Fault
                - Enhancement
                - Trivial
                - Minor
                - Major
                - Critical
                - Blocking
            - Assign *priorities*
            - Faults are recorded by fault type
            - For a given workflow, we compare current fault rates with those of previous products
            - We take action if there are disproportionate number of faults in an artifact
                - Redesigning from scratch is a good alternative
            - We carry forward fault statistics to the next workflow
                - We may not detect all faults of particular type in the current inspection
    - Comparison of Inspections and Walkthroughs
        - Walkthrough
            - Two Step, informal process
                - Preparation
                - Analysis
        - Inspection
            - Five-step, formal process
                - Overview
                - Preparation
                - Inspection

            - Rework
            - Follow-up

        - Strengths and Weaknesses of Reviews
            - Review can be effective
                - Faults are detected early in the process

            - Reviews are less effective if the process is inadequate
                - Large-Scale software should consist of smaller, largely independent pieces
                - The documentation of the previous workflows has to be complete and available online

            - Code reviews lead to rapid and thorough fault detection
                - Up to 95% reduction in maintenance costs

- Non-execution VS Execution Based Testing
    - Non-execution based testing used when testing artifacts of the requirements, analysis, and design workflows
    - Execution based testing applied to only the source doe of implementation workflow
    - Non execution based testing of code (code review) has been shown to be as effective as execution-based testing (running test cases)

- Execution-Based Testing
    - Organizations spend up to 50% of their software budget on testing
        - But delivered software is frequently unreliable

    - Testing is the process of *finding differences between the expected behavior and the observed behavior*
    - fault detection technique that tries to create failures or erroneous states in a planned way
    - A test is successful if it identifies faults or proves that no fault are present
    - What should be tested?
        - Correctness
            - Requirement satisfaction

        - Utility
            - Extent to which the product meets the user's needs
                - Ease of use, useful functionalities, cost effectiveness

        - Reliability
            - Measure of the frequency and criticality of failure

        - Robustness

            - The range of operating conditions
            - The possibility of unacceptable results with valid input
            - The effect of invalid input
        - Performance
            - Extent to which space and time constraints are met
                - Real and Hard time constraints, data loss because software is slow
    - Who Should Perform Execution Based Testing?
        - Programming is constructive
        - Testing is destructive
            - A successful test finds a fault
            - So, programmers should not test their own code
        - Solution
            - The programmer does informal testing
            - The SQA group then does systematic testing
            - The programmer debugs the module
        - All test cases must be
            - Planned beforehand, including the expected output
            - Retained afterwards
- When Testing Stops
    - Only when the product has been irrevocably discarded