Testing Background

- History and Background
    - Problems of past still exist today
        - High cost
            - Productivity = Lines of code / hour (avg 1/hr)
        - Difficult to manage
        - poor reliability
        - lack of user acceptance
        - difficult to maintain
- MTBF = Mean Time Between Failure

- Reliability and Testers
    - Software defects are around 1 defect / LOC
    - Testers vs (Scientific) Pollsters
        - Both paid to make predictions
        - Pollsters have constraints [Money, and time]
        - "exhaustive" ["representative samples"]
- Define common testing terminology (Derived mainly from IEEE and ACM)
    - Reliability: The probability that a software program operates for some given time period without software error
    - Validation: "Are we building the right product?"
    - Verification "Are we building the product right?"
    - Testing: The examination of behavior of the program by executing it on sample data sets
    - Error: Mistake made by a human (ex: forgot about requirement)
    - Defect/Fault: Result of error manifested in the code
    - Failure: Software doesn't do what it is supposed to do (triggered by defects/faults)
- Testing Background Practice Quiz
    - True or False? Verification focuses on if the requirements are meeting customer needs and are what the customer wants.
        - False: Verification looks at building the product right given the assumption that the requirements are correct. Validation looks at building the right product to bring value and meet the customer's requirements.
    - A team of testers are testing software against a given set of requirements. The requirements have already approved by the customer. Is the testing team performing verification or validation of the software, or neither?
        - Verification: Verification looks at building the product right given the assumption that the requirements are correct. Validation looks at building the right product to bring value and meet the customer's requirements.

- Classic Testing Mistakes:
    - The role of testing
        - Thinking that the purpose of testing is to find bugs
        - Starting testing too late (bug detection, not bug reduction)
    - Planning the complete testing effort
        - A testing effort biased toward functional testing
        - An overreliance on beta testing (customer testing)
        - Sticking stubbornly to the test plan
    - Personnel issues
        - Using testing as a transitional job for new programmers
        - A physical separation between developer and testers
        - Believing that programmers can't test their own code
        - Programmers are neither trained nor motivated to test
    - Test automation
        - Attempting to automate all tests
    - Code Coverage
        - Embracing code coverage with the devotion that only simple numbers can inspire
        - Using coverage as a performance goal for testers

Testing Throughout Life Cycle

- Describe how testing is integrated into software development phases
- Waterfall
    - Requirements > Design > Code > Test
    - Do not begin next step until the previous step is done (rigid)
    - Testing occurred at the end, until code is complete
    - Testers typically involved in requirements and each step, review the requirements to ensure that they were indeed testable. Thinking about system level requirements.
- Agile
    - Emphasizes high degree of customer interaction with development process
    - Continuous Integration (at least daily)
        - Static Code Analysis (automated)
        - Compile
        - Unit Test
        - Deploy into Test Environment
        - Integration / Regression Test
- Test Driven Development [TDD] (Red, Green, Refactor Cycle)
    - Emphasis on writing test cases first, use test cases to define behavior that software developer needs to implement
    - Red Phase: Write a minimal test on the behavior needed
    - Green Phase: Write only enough code to make the failing test pass
    - Refactor Phase: Improve code while keeping tests green

- Software Development Process vs Test Development Process
  - Requirement ~ Test objectives (what do we want testing to accomplish? Stress, usability, etc.)
  - Design ~ Test Design (sampling strategy)
  - Code ~ Write Test Cases / Test Scripts
  - Test ~ Executing Tests
  - Maintenance ~ Update tests
- Define the objectives of the different levels of testing
- Testing Levels
  - Unit / Component Testing
    - Individual developer, make sure their unit does what its supposed to, To verify that an individual developer's unit is tested properly. Maybe TDD. High degree of automation with tools like JUnit. Objective is to make sure unit / component does what its supposed to do. Verification.
  - Integration Testing
    - May happen daily or more. Objective is to make sure that new units are working together with existing systems.
  - System Testing
    - Software has been integrated together, but hardware / software integration is a concern. Objective: To test both functional and nonfunctional requirements. Still worried about functional capabilities, but also security, performance, stress, usability, safety, etc. Functional and nonfunctional testing as well.
  - Acceptance Testing
    - Customer testing. Customer needs to be able to install or set up. Might have independence being done or by customer.
  - Beta Testing
    - Some sort of preferred testing. Separate sort of customer and let them do some form of testing. Risks: There might be problems damaging to the company.
- Test Types
  - Functional
    - Verification, validation, making sure the software does what it's supposed to be doing. Pretty much applicable to each level of testing.
  - Non-functional
    - "ilities". secure, performance, usable. Higher levels of testing.
  - Structural
    - White box, making sure code logic is good. Dataflow coverage. Primarily at the unit level.
  - Regression
    - Applicable at all levels. Making sure that new changes don't break the code.

- Testing Throughout Life Cycle Practice Quiz
    - How often does testing happen in agile development?
        - At least daily Agile testing follows the theme of continuous integration where code is integrated and tested at least daily.
    - What is the correct phase order in test driven development?
        - Red, Green, Refactor. The red phase is where a small test is written that defines the behavior needed from the code. Then, in the green phase, code is written to ensure the failing test from the previous phase passes. In the refactor phase, the quality of code is improved while all tests are still passing.
    - What is the objective of system testing?
        - To test both functional and nonfunctional requirements. System testing is when other components, such as hardware and software integration testing (functional requirements) and aspects such as stress, usability, and performance testing (nonfunctional requirements) are tested.
    - What is the objective of unit level testing?
        - To verify that an individual developer's unit is tested properly. Unit level testing is where each developer tests their unit (section of code) to ensure it is working properly

Testing Best Practices and Standards

- Testing Principles
    - Principle 1
        - Testing only shows the presence of defects - proof of correctness
    - Principle 2
        - Exhaustive testing is impossible
    - Principle 3
        - Start testing early
    - Principle 4
        - Defects cluster. Individual developer gets sloppy on one section of code or is a novice, some portions of code are more complex than other. Not realistically 1 error per 1000 LOC.
    - Principle 5
        - Testing is context dependent. Test cases may be based on system being in a particular case, maybe what has happened prior to running the test case? Context could be location, time of day.
    - Principle 6
        - Absence-of-errors fallacy.
- Testing Attitude
    - Independence
        - Realize bias. In a perfect world, all testing would be done independently.
    - Customer Perspective

        - Include testing from customer's perspective. Try what customer would try. Sometimes we don't have the big picture of how the software will be used. Validation.
    - Demonstrate that the system works (test intended functionality)
        - Tester's go in and try to break the system.
    - Demonstrate that the system is bullet proof (test unintended functionality)
    - Professionalism
        - Need certifications, ensure that testers are educated and qualified.
- Classic Testing Mistakes
    - Believing the primary objective of system testing is to find bugs
        - Test must concentrate on finding important problems
        - Test must provide an estimate of system quality
    - Not focusing on usability issues
        - Testers not concerned about the success about the product, just what's their job
    - Starting too late
        - Test must help development to avoid problems
    - Delaying stress and performance testing until the end
    - Not testing the documentation [Customer]
    - Not staffing the test team with domain experts
        - Need experts who understand the customer and the customer's perspective
    - Not communicating well with the developers
    - Failing to adequately document and review test designs
- General Testing Standards (ISO/IEC/IEEE 29119 Software Testing)
    - ISE/IEC: Concepts and Definitions
    - ISE/IEC: Test Processes
    - ISE/IEC: Test Documentation
    - ISE/IEC: Test Techniques
    - ISE/IEC: Keyword Driven Testing

- DO-178C Software Considerations in Airborne Systems and Equipment Certification (example)
    - Level A: Catastrophic
    - Level B: Hazardous/Sever
    - Level C: Major
    - Level D: Minor
    - Level E: No Effect
    - Each level would have a test coverage/specific kinds of tests required.
- When to Stop Testing
    - Out of time / money (probably bad answer)
    - No more defects found (probably bad, maybe testing not good anymore)
    - Demonstrated all requirements are met

    - Demonstrated Code coverage
    - Meets reliability objectives
    - Customer is satisfied
- ISTQB Code of Ethics
    - Public:
        - Certified software testers shall act consistently with the public interest
    - Client and Employer
        - Certified software testers shall act in a manner that is in the best intersects of their client and employer, consistent with the public interes
    - Product
        - Certified software testers shall ensure that deleverables they provide meet highest professional standards possible
    - Judgement
        - Certified software testers shall maintain integrity and independence in their professional judgement
    - Management
    - Certified software test managers and leaders shall subscribe to and promote an
    - Profession
    - Colleagues
    - Self
- Testing Principles and Best Practices
- Which group of people should perform system testing
    - An independent test team. A testing attitude is to gain independence to counter bias during testing.
- What is(are) the primary objective(s) of system testing?
    - To find important problems and predict reliability. Finding important problems that affect the customer greatly is more important than finding many minor problems in system testing. Testing must also provide an estimate of reliability.

- When is a reasonable time to stop testing?
    - Once we have met our test objectives. Test objectives typically include testing all of the functional and nonfunctional requirements.
- Why do defects cluster?
    - Because of the complexity of code, programmer skill, etc. Defect clustering happens for many reasons. Some parts of the code are more complex, causing more defects.

- Explain best practices for software testing
- Best Testing Practices
  - Assess software reliability via statistical testing
  - Develop an agile test design
    - Accommodate late changes
    - Emphasis on regression testing (new changes don't break current code)
  - Utilize model-based testing techniques
    - State diagrams
    - Develop a model, and use that model to generate code
  - Develop cross-functional development and test teams
  - Automate Test generation where possible
  - Emphasize usability testing

- Software testing best practices: Review of reading
  - Reviews and inspections.
    - Also review and inspect test cases and test strategies
  - Formal Entry and Exit Criteria
    - What point do start testing like beta testing?
    - When can we stop testing?
  - Multi-platform testing
    - Test code on multiple configurations such as different phones, different OS, etc
  - Automated Test Execution
    - Do automated test where we can
  - In-Process ODC (orthogonal defect classification) feedback loops
    - Measurement method that uses the defect stream to provide precise measurability into product and process.
    - Go in and look at defects, to be able to classify them, their priority, why were they made. Look at that info and have a feedback mechanism in place so we can learn from our mistakes.
  - Requirements for Test Planning
  - Usability Testing
    - Critical
  - Teaming testers with developers, make them work together
  - Code Coverage
    - Use tools to ensure we achieve adequate code coverage, but not excess
  - Automated environment generator
    - Build code based on test cases
  - Testing to help ship on demand
  - State Task Diagram
    - Model based testing as known today
  - Statistical testing
  - Semi-Formal Methods

        - Program verification techniques
    - Bug Bounties
        - Incentive testers to find bugs in the system. Known as crowd testing today. Individuals can be hired and rewarded for finding bugs.
- Automations Role in the Fall of Software Testing
    - Sometimes we put too much focus on trying to automate
    - But it plays a key role in regression testing
    - Sometimes automation leads to erroneous conclusions that we don't need as many testers, or we don't need to spend as much time on testing.
    - Automation tests usually imply shallow testing, easy to create scripts, validate boundary values. But deeper kinds of testing need more involvement usually.
    - Summary: automation has a role, but software development needs testers who have a deep cognitive knowledge to perform different kinds of testing.