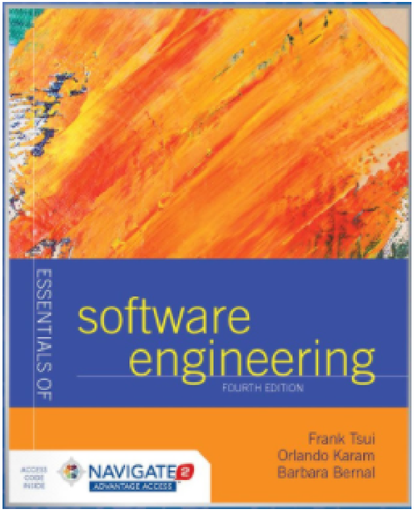
Testing and Quality Assurance

CSC 355, Software Engineering II

Chapter 10: Testing and Quality Assurance



Objectives

- Understand the basic techniques for software verification and validation
- Analyze basics of software testing and testing techniques
- Discuss the concept of “inspection” process

Levels of Testing

- Order test cases are written
  1. System
  2. Integration/Component
  3. Functional
  4. Unit
- Order test cases are executed
  1. Unit
  2. Functional
  3. Integration/Component
  4. System

Quality Product

- How do you know if something is a quality product?
- What products do you think about when you hear “Quality”?
- Why?

Quality Product

- Is quality verified throughout the process?
- Is there a process for testing?
- Is the team proud of the product?
- Does it meet specifications?
- Is it “fit for use”?

What is Quality?

- Quality Assurance
    - measure quality
    - improve quality
    - team training
- Quality Control
    - verify quality
    - detect errors
    - fix errors (prior to release)

What is Quality?

- Verification
  - checking software conforms to its requirements
  - is the system correct, according to the specifications?
- Validation
  - checking software meets user requirements
  - are we building the correct system?

Looking for Errors

- Testing
  - test cases verify output is correct
- Inspection and reviews
  - review of models, specifications, documents, and code
  - design reviews
  - code walkthroughs

Looking for Errors

- Formal Methods
  - mathematically “proven” correct
- Static analysis
  - automated “checker”
  - looking for error-prone conditions

Types of Errors

- Error
  - a mistake made by a developer
- Fault (defect, bug)
  - result of an error; condition that may cause a failure in the system
- Failure (problem)
  - inability of system to perform according to its specification due to some fault

Prioritizing Errors

- Fault or failure severity
  - based on consequences
  - blocker / critical / major / minor / trivial
- Fault or failure priority
  - based on importance of developing a fix
  - critical / high / medium / low / won't fix

Levels of Testing

- What is tested?
  - Unit code testing
  - Functional code testing
  - System testing (regression testing)
  - User interface testing
  - Acceptance testing

Testing

- Activity performed for:
  - Evaluating product quality
  - Improving products by identifying defects and having them fixed prior to software release
- Dynamic (running program) verification of program's behavior on a finite set of test cases selected from execution domain
- Testing can not prove product works 100%

Testing

- Why test?
    - Acceptance (customer)
    - Conformance (std, laws, etc.)
    - Configuration (user vs. dev.)
    - Performance, stress, security, etc.

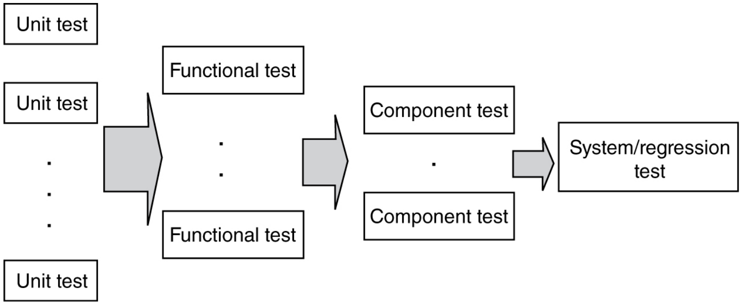
Testing

- How (test cases designed)?
  - Intuition
  - Specification based (black box)
  - Code based (white box)
  - Existing cases (regression)

Testing Responsibilities

- Who tests?
    - Programmers
    - Testers / Requirements Analyst
    - Users (beta group)

Levels of Testing



When to Stop Testing?

- Simple answer: stop when
  - all planned test cases are executed
  - all problems that are found are fixed
- Not when you ran out of time (poor planning)
  - Build time for all levels of testing in the program plan

System Testing

- Performed from the end user's perspective
- Run these tests after unit, functional, and component tests are successful
- Based on Software Requirements Specification

Unit Testing

- Test each individual unit (method, class, file, ...)
- Usually done by the programmer
- Test each unit as it is developed (small chunks)
- Keep test cases / results around
  - allows for regression testing
  - facilitates refactoring
  - tests become documentation