Test-Driven Development

SC12 Educator’s Session
November 13, 2012
Outline

- Software Quality
- Overview of Testing
- Automated Testing Tools
- Test-Driven Development
SOFTWARE QUALITY
Software Quality

- Multiple Definitions

- Developer’s View vs. User’s View
  - Developers =
  - Users =
Definitions

- Software must **do the right things**
  - Perform the right functions
  - Often referred to as

- Software must **do things right**
  - Perform intended functions without problems
  - Often referred to as
Quality Definitions:

Defects

- Failure
- Fault
- Error
Quality Definitions:

Defects

SC12
Educator's Session
Quality Focus

- Customers/Users
- Developers
QA Activities: Types

- Defect Prevention
- Defect Reduction
- Defect Containment
QA Activities: Timing

- **Pre-release**
  - Defect Prevention & Reduction
  - Remove as many defects as possible before release

- **Post-release**
  - Repair failures
    - Reduces defects
    - Significantly more expensive than pre-release
    - Bad PR
  - Contain defects
    - Minimize impact
    - Expensive, so limited in application
Defect Prevention: Introduction

- Reduce the chance of fault injection
- Approach depends on source

- There may be specific tools or technologies that can also help
- Important to establish the correct root-cause
Defect Prevention: 
Education and Training

- People are most important factor in quality and success

- Education and Training can improve the quality of the work done by practitioners

- Elimination of misconceptions will reduce the probability of defect injection
Defect Prevention: Education and Training

- Product and Domain Specific Knowledge
- Software Development Expertise
- Knowledge about tools, methods, techniques
- Development Process Knowledge
Help eliminate error sources and verify the absence of faults.
Formal Methods: Axiomatic Approach

Biggest obstacle to use of formal methods:
Defect Prevention: Other Techniques

- Use of additional software development methodologies (besides Formal Methods)
  - Prevent extra functionality
  - Reduce complexity

- Better management
  - Concrete process definition
  - Enforcement of standards

- Use of specific tools
  - Enforce coding standards
QA Activities:
Defect Prevention
Defect Reduction: Introduction

- Unrealistic to expect **Defect Prevention** step to stop all defects
- Different approaches
Defect Reduction: Testing

- Execution of software and checking results
  - Locates failures
  - Isolate and fix the fault(s) that led to the failure

- When to test
  - Need some executable
  - Unit tests of components through acceptance test of entire system
  - Can also use prototypes
Defect Reduction:
What to Test

- Functional (black box)

- Structural (white box)
Defect Reduction: When to Stop Testing

- Can use coverage criteria
  - Assumption:

- Reliability goals
Defect Reduction: Observations

- Many other techniques available

- In-field measurement and repair not normally considered part of QA

- Important to determine risky components
QA Activities:
Defect Reduction
Defect Containment: Introduction

- Important for systems where the impact of failures is substantial
- Not all faults can be eliminated (cost, time)
Defect Containment: Fault Tolerance

- Different from manufacturing
- Approaches

- Does not focus on identifying and removing the faults that cause the failures
Defect Containment:
Safety Assurance

- Safety-critical systems:
- Address even low probability failures
- Safety Assurance techniques
QA Activities:
Defect Containment
INTRODUCTION TO TESTING
Types of Testing
Unit Testing

- What?

- Who?

- What is the focus?

- What type of testing techniques do we use?
Component Testing

- What?
- Who?
- What type of testing techniques do we use?
- COTS and CBSD
Integration Testing

- What?

- Who?

- What is the focus?

- What type of testing techniques do we use?

- Merged with System Testing?
System Testing

- What?

- Who?

- What is the focus?

- What type of testing techniques do we use?

- Embedded systems?
Acceptance Testing

- What?
- Who?
- What is the focus?
- What type of testing techniques do we use?
Functional vs. Structural
Functional vs. Structural

- Individual elements
- Interactions of elements
- Inputs and outputs – functional
Black Box (Functional) Testing: Overview
Black Box (Functional) Testing: Process

- Planning
- Execution
- Analysis
White Box (Structural) Testing:

Overview
Stopping Criteria: Coverage-Based

- Ensures some item has been covered
- Assumes that
- Approaches
Coverage Based Testing: Process

- Define the model
- Check the model elements
- Define the coverage criteria
- Derive the test cases
Test Activities

- Test Planning
- Test Execution
- Analysis and Follow-up
Test Planning

- High level goal:

- Make the following decisions

- Have to account for personnel
Test Planning:
Test Case Creation

- What is needed?
- How are they generated?
Test Planning:
Test Suite Preparation

- What is a test suite?
- How are they created?
- Expensive → should be maintained for future use
Test Planning:  
Preparation of Procedure

- Ordering of test cases

- One test case should leave the system ready to execute the next

- Assignment of personnel
Test Execution: Overview

- Major steps:

- Prevent failed test cases from halting execution

- Environment
Specific Approaches to Testing

- Control Flow Testing
- Partition-based Testing
- Usage-based Testing
- Data-flow Testing
Systematic Testing

- Drawbacks to *ad hoc* testing

- One way to structure is to build a checklist
Control Flow Testing: Overview

- Model of the software is a graph

- Use
Control Flow Testing:
Model Construction

- Can also be done with black box testing

- Elements
Control Flow Testing:
Path Selection
Control Flow Testing:

Path Selection
Control Flow Testing:
Model Construction

L1: input(a,b,c)
L2: d ← b*b - 4*a*c
L3: If (d>0) then
L4: r ← 2
L5: else_if (d=0)
L6: r ← 1
L7: else_if (d<0)
L8: r ← 0
L9: output (r)
Control Flow Testing: Creating Test Cases

- If each decision is based on an independent variable, then just choose appropriate values
  - Can use the idea of equivalence classes

- If decisions are not independent, some branches may be eliminated as infeasible

- Some decisions may be based on processing between decisions nodes – may be hard to develop test cases
Loops

- Loops complicate the CFT idea. Why?

- Parts of a loop:
Loops

While (C) do {B}

For (I; C; U) do {B}
Loop Testing: Difficulties

- When loops are nested, number of paths quickly grows unmanageable

- Complete path coverage not possible, have to be selective

- Where do most problems occur?

- What types of test cases do we need and why?
Loop Testing: Difficulties

- Concatenation/Nesting of loops

- How can we reduce number of test cases?
Specific Approaches to Testing

- Control Flow Testing
- **Partition-based Testing**
- Usage-based Testing
- Data-flow Testing
Partition Based Testing

- **Benefits**
  - Increased coverage
  - Reduced overlap

- **Examples:**
  - Solve for root of $ax^2 + bx + c = 0$
  - Thermostat
Partition Testing: Theory

- A set $S$ contains a list of unique elements

- Partition of $S$ creates subsets $G_1, G_2, \ldots G_n$ such that
  - Sets are mutually exclusive
  - Sets are collectively exhaustive

- $G_1..G_n$ are equivalence classes if created based on some definition of equality

- Properties
  - Symmetric
  - Transitive
  - Reflexive
AUTOMATED TESTING TOOLS
Automated Testing Frameworks

- Enable set of tests to be executed repeatedly

- Family of tools
  - jUnit
  - cUnit
  - … (xUnit)

- Demo
cUnit

- [http://cUnit.sourceforge.net](http://cUnit.sourceforge.net)

- Framework to create and execute tests

- Assertions
  - CU_ASSERT
  - CU_ASSERT_TRUE
  - CU_ASSERT_FALSE
  - CU_ASSERT_EQUAL
  - CU_ASSERT_NOT_EQUAL
  - …
cUnit: Test Registry

- Repository of test suites and tests

- Using the test registry
  - Create
  - Clean up

- Adding tests
  - Create a test suite
  - Add tests to the test suite
cUnit:
Running Tests

- Can run:
  - All tests
  - Individual suites
  - Individual tests

- Modes
  - Automated – non-automated / XML output
  - Basic – non-automated / stdout output
  - Console – interactive console under user control
TEST-DRIVEN DEVELOPMENT
Basic idea:

- Part of the *agile* software development approach
  - Pair programming
Test-Driven Development: Overview

- Focus on unit tests

- Often require

- Can be automated or manual

- Can be performed by developers or testers
Test-Driven Development: Motivation

- Programming practice that instructs developers to:
  - Test-Driven Development
Test-driven development (TDD) is the craft of producing automated tests for production code, and using that process to drive design and programming. For every tiny bit of functionality in the production code, you first develop a test that specifies and validates what the code will do. You then produce exactly as much code as will enable that test to pass. Then you refactor (simplify and clarify) both the production code and the test code.
Test-Driven Development: Additional Thoughts

- Refactoring

- Not a software development methodology

- Provides automated test
Test-Driven Development: Process
Test-Driven Development:
Example Story
Test-Driven Development: Automated Testing

- TDD assumes the presence of an automated testing framework

- Test Harnesses

- xUnit
  - Lets users write tests to initialize, execute, and make assertions about code being tested
  - Tests can serve as documentation
Test-Driven Development: Evaluation

- Performed in Industry and Academia

- Industrial studies
  - 4 studies in small companies
  - Measured defect density

- Results
  - Programmers using TDD produced code that passed 18% - 50% more tests
  - TDD programmers spent less time debugging
  - TDD decreased productivity - but they wrote more test cases
Test-Driven Development: Challenges to Adoption
Test-Driven Development: Example

- Design a system to perform financial transactions with money that may be in different currencies

- For example –
  - If the exchange rate from Swiss Francs to US Dollars is 2 to 1, then we can calculate
    - 5 USD + 10 CHF = 10 USD
    - 5 USD + 10 CHF = 20 CHF
References

- Slides adapted from materials in:


- Example taken from
  - Kenneth Anderson, Univ. of Colorado, Boulder
Updated slides and handouts available on the web

http://carver.cs.ua.edu/SC12_Tutorial/
HANDS-ON TIME