Regression Testing

I Didn’t Change Anything!
Regression Testing - Definition

- The process of testing changes to software to make sure that unchanged code still works with the new changes
- Intent is to check that code has not regressed
- Typically done in the context of:
  - Iterative software development process
    - Agile methods
  - Component reuse and integration
    - The change of use contexts
  - Component maintenance
Strategies after component changes

- Repeat all the testing activities
  - Develop testing plan
  - Develop test cases
  - Run all the test cases

- Anything wrong?

- Reuse the test cases available for the components
  - Fundamental question: What test cases to re-run?

- Develop new test cases for the functionalities that have been added or modified
Terminologies

- **Baseline**
  - Point of reference for a regression test suite

- **Delta version**
  - Small change from a baseline, or one of a sequence of such changes

- **Delta build**
  - An executable configuration of the SUT that contains all the baseline and delta components

- **Regression test case**
  - Passed on the baseline version
  - Expected to pass for the delta version

- **Regression fault**
  - Fault introduced when components change
  - Revealed by a regression test case that fails on a delta
Motivation

- Regression testing is effective for revealing regression faults due to:
  - Delta side effects
  - Delta/baseline incompatibilities
  - Undesirable interactions between baseline and a delta
  - **Bad fixes**
    - High tech companies like IBM have fix failure rates from 2% to 20%
    - 1/3 of fixes either don’t fix the problem or break something else
    - Up to 80% of fixes not working has been reported in some large systems
When

- Regression testing a class
  - A bug fixed in the class
  - When a new subclass has been developed
  - When a superclass is changed
  - When a server class is changed
- Regression testing a system/subsystem
  - When a new build is available
  - …
Regression Testing Procedure

- Basic regression testing procedure
  1. Remove broken test cases from the original test suite
  2. Choose a full or a reduced regression test suite
  3. Set up test configuration
  4. Run the regression test suite
  5. Take appropriate action for any failed tests
Regression Faults

- Occurs when both a stable baseline B and a delta component D pass individually test suites but fail when used together.

Examples:
- D is a client of B
- B is a client of D
  - D’s postconditions have changed
  - D is a new subclass of a polymorphic hierarchy
- D has a side effect on B
Management of Regression Test Suites

- Test Suites Decay:
  - Broken test cases
    - Cannot be run any more
  - Obsolete test cases
    - No clear purpose for the test cases
  - Uncontrollable test cases
    - Depends on states or inputs out of control
  - Redundant test cases
Management of Regression Test Suites - 2

- Example of aerospace manufacturer if test suites are not pruned
  - 165,000 test cases in suite, unacceptable test run time
- Analysis
  - ran on instrumented build
    - 90% of test cases were redundant
    - many segments of application not reached
- Solution
  - after cleanup, 18,000 test cases remained
  - about 3000 new test cases
- In a study by Jones
  - 30% of regression test cases were duplicates
  - 12% contained errors
Safe Regression Test Selection

- Time and cost constraints may require to reduce regression test suite
- Given an IUT, a set of test cases T for IUT, and a modified version IUT’, a selection technique is safe if,
  - For any test case $t$ in T that is not select, IUT and IUT’ will yield identical outputs on $t$
- A trivial safe selection technique
  - Select all tests cases in T
Test Selection Strategies

- **Safe strategies**
  - Retest all
  - Retest changed code
  - Retest within firewall

- **Unsafe strategies**
  - Retest risky use cases
  - Retest by profile
Retest All

- **Intent**
  - Rerun the entire baseline test suite on a delta build

- **Context**
  - Can be applied at any scope

- **Fault Model**
  - Catch any kind of regression fault

- **Strategy**
  - Rerun baseline test suite after removing broken test cases
Entry and Exit Criteria

- **Entry criteria**
  - The delta components pass component scope testing
  - A suitable baseline test suite exists

- **Exit criteria**
  - All no pass test cases reveal bugs whose presence and severity are deemed acceptable
  - All remaining test cases pass

- **Consequences**
  - Is Safe
  - Has the lowest risk of missing a regression fault
  - But highest test cost
  - General
Retest Risky Use Cases

- **Intent**
  - Use risk-based heuristics to select a partial test suite

- **Context**
  - Full regression run has a high cost
  - How to select a subset of the baseline test suite?

- **Entry and Exit Criteria**
  - As in *Retest All*
Retest Risky Use Cases - 2

- **Strategy**
  - Apply a risk criteria to select the subset
    - Suspicious use cases
      - Depend on components that are unstable or unproven, have a complex implementation, where subject to a lot of modifications or have not been shown to work together before
    - Critical use cases
  - Consequences
    - Unsafe: Moderate risk of missing a regression fault
    - Low cost of analysis and setup
Retest by Profile

- **Intent**
  - Use a budget-constrained operational profile to select a partial regression test suite

- **Context**
  - Given a deadline or budget, how to select a subset of the baseline test suite?

- **Strategy**
  - Select test cases proportional to relative frequency of each use case
  - And consider total budget for regression time
Retest by Profile - 2

- Entry and Exit Criteria
  - Same as Retest All

- Consequences
  - Requires that baseline test suite was developed using Allocate Test by Profile
  - Unsafe
    - Moderate risk of missing a regression fault
Retest Changed Code

- **Intent**
  - Use code change analysis to select partial regression test suite

- **Context**
  - Full regression run has a high cost
  - How to select a subset of the baseline test suite?

- **Entry and Exit Criteria**
  - Same as *Retest All*
Test Model

- Primary goal for regression test selection
  - Find baseline test that will reveal regression faults
- Regression faults are related to new, modified or deleted code
- Need to compare each segment in baseline and delta components
Test Procedure

- Use a coverage analyzer to list the codes segments exercised per test case
- Use a control version-based tool to generate the differences between baseline and delta
  - Mark each segment as new, changed, same and modified
- Regression test suite should include all baseline test cases that exercise segments marked as changed or modified
Consequences

- It is safe
  - All baselines test cases that can produce a different result are selected
- Cost (in time) can be high due to the dependency analysis involved