

Regression Testing

SE401: Software Quality Assurance and Testing

Outline

- Software Evolution
- Regression Testing
- Basic Problems of Regression Testing
- Selecting and Prioritizing Regression Test Cases
- Summary

Software Evolution

- Change happens throughout the software development life cycle.
 - Before and after delivery
- Change can happen to every aspect of software
- Changes can affect unchanged areas
 - break code, introduce new bugs
 - uncover previous unknown bugs
 - reintroduce old bugs

Regression Test

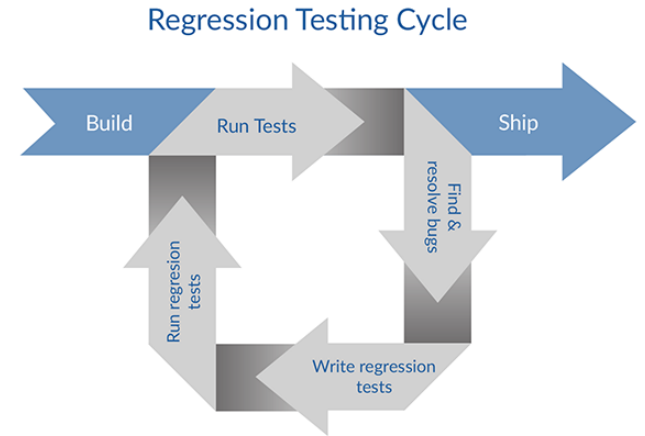
- Testing of a previously tested program following modification to ensure that new defects have not been introduced or uncovered in unchanged areas of the software, as a result of the changes made.
- It should be performed whenever the software or its environment is changed.
- It applies to testing at all levels.

Regression Testing

- It is a type of software testing that intends to ensure that changes (enhancements or defect fixes) to the software have not adversely affected it.
- Rerunning of tests can be on both functional and non-functional tests.
- It is a type of change-related testing to detect whether defects have been introduced or uncovered in unchanged areas of the software.

Regression Test

- Keep a test suite
- Use the test suite after every change
- Compare output with previous tests
 - Understand all changes
- If new tests are needed, add to the test suite.



Regression Testing?

	System	Acceptance	Regression
Test for ...	Correctness, completion	Usefulness, satisfaction	Accidental changes
Test by ...	Development test group	Test group with users	Development test group
	Verification	<i>Validation</i>	Verification

Regression Test

- Yesterday it worked, today it doesn't
 - I was fixing X, and accidentally broke Y
 - That bug was fixed, but now it's back
- Tests must be re-run after any change
 - Adding new features
 - Changing, adapting software to new conditions
 - Fixing other bugs
- Regression testing can be a major cost of software maintenance

Reasons

- Regression testing is conducted to make sure that fixing one thing has not broken another thing.
- The need for Regression Testing could arise due to any of the changes below:
 - Defect fix
 - New feature
 - Change in an existing feature
 - Code refactoring
 - Change in technical design / architecture
 - Change in configuration / environment (hardware, software, network)

Basic Problems of Regression Test

- Maintaining test suite
 - If I change feature X, how many test cases must be revised because they use feature X?
 - Which test cases should be removed or replaced? Which test cases should be added?
- Cost of re-testing
 - Often proportional to product size, not change size ** cost*
 - Big problem if testing requires manual effort

Test Case Maintenance

- Some maintenance is inevitable
 - If feature X has changed, test cases for feature X will require updating
- Some maintenance should be avoided
 - Example: Trivial changes to user interface or file format should not invalidate large numbers of test cases
- Test suites should be modular!
 - Avoid unnecessary dependence
 - Generating concrete test cases from test case specifications can help

Obsolete and Redundant

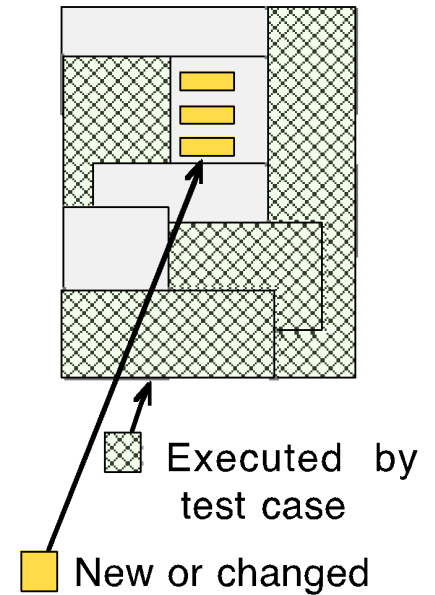
- **Obsolete:** A test case that is not longer valid
 - Tests features that have been modified, substituted, or removed
 - Should be removed from the test suite
- **Redundant:** A test case that does not differ significantly from others
 - Unlikely to find a fault missed by similar test cases
 - Has some cost in re-execution
 - Has some (maybe more) cost in human effort to maintain
 - May or may not be removed, depending on costs

Selecting and Prioritizing Regression Test Cases

- Should we re-run the whole regression test suite? If so, in what order?
 - Maybe you don't care. If you can re-rerun everything automatically over lunch break, do it.
 - Sometimes you do care ...
- Selection matters when
 - Test cases are expensive to execute
- Prioritization matters when
 - A very large test suite cannot be executed every day

Code-Based Regression Test Selection

- Observation: A test case can't find a fault in code it doesn't execute
 - In a large system, many parts of the code are untouched by many test cases
- So: Only execute test cases that execute changed or new code



Specification-Based Regression Test Selection

- Like code-based regression test case selection
 - Pick test cases that test new and changed functionality
- Difference: No guarantee of independence
 - A test case that isn't "for" changed or added feature X might find a bug in feature X anyway
- Typical approach: Specification-based prioritization
 - Execute all test cases, but start with those that related to changed and added features

Prioritized Rotating Selection

- Basic idea:
 - Execute all test cases, eventually
 - Execute some sooner than others
- Possible priority schemes:
 - Round Robin:
 - Priority to least-recently-run test cases
 - Track record:
 - Priority to test cases that have detected faults before
 - Structural:
 - Priority for executing elements that have not been recently executed

Summary

- Regression testing repeated after each change
 - After initial delivery, as software evolves
- Reasons for regression testing
- Basic Problems of Regression Testing
- Selecting and Prioritizing Regression Test Cases