Topics for this Lecture

- What is regression testing?
- When to do it?
- What is the strategy?

Regression testing

- **Regression testing** is any type of software testing that seeks to uncover new errors in existing functionality after changes have been made to the software, such as functional enhancements or patches. Therefore, regression testing tests both the modified code and other parts of the program may be affected by the program change.
- A program **P** has been modified into program **P`** (maintenance)
- **P`** needs specific testing attention to ensure that
  1. newly added or modified code behaves correctly
  2. code carried over unchanged, continues to behave correctly

When to do it

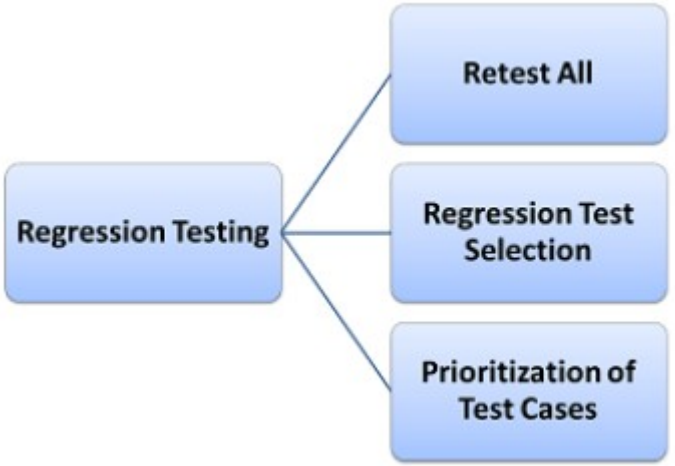
- Regression Testing is required when there is
  - Change in requirements and code is modified according to the requirement
  - New feature is added to the software
  - Defect fixing
  - Performance issue fix
- Anytime new capability is added all **previous**, **validated** tests are run, and the results are compared with the standard results already stored on file.
- Anything that goes wrong with the **old** tests can be traced to something done between the last time the regression test was run, and the time the latest one has run.

Regression Testing vs. Testing

- **Availability of test plan**
  - Testing starts with a specification, an implementation of the specification and a test plan (black-box and/or white-box test cases, test scenarios, coverage, tools). Everything is new
  - Regression testing starts with a (possibly modified) specification, a modified program, and an old test plan (which requires updating)
- **Scope of test**
  - Testing aims to check the correctness of the whole program
  - Regression testing aims to check (modified) parts of the program
- **Development information**
  - During testing, knowledge about the development process is available (it is part of it)
  - During regression testing, the testers may not be the ones who developed/tested the previous versions
- **Completion time**
  - Completion time for regression testing should normally be much less than that for exhaustive testing (only parts are tested)
- **Frequency**
  - Testing occurs frequently during code production
  - Regression testing occurs many times throughout the life of a product, possibly once after every modification is made to it

What's the strategy

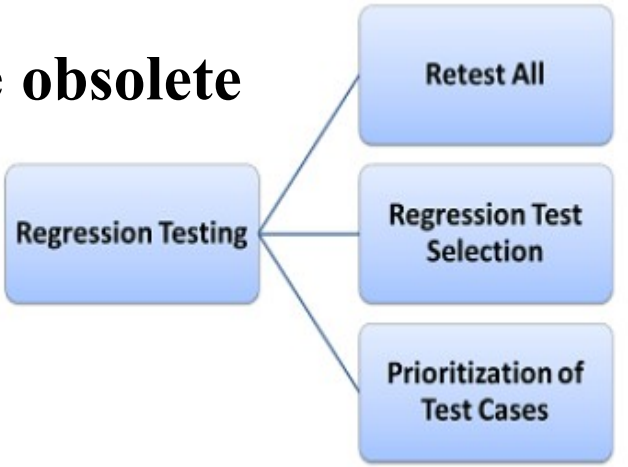
- **Retest All**
  - This is one of the methods for regression testing in which all the tests in the existing test suite should be re-executed. This is very expensive as it requires huge time and resources.



- **Regression Test Selection**
  - Instead of re-executing the entire test suite, it is better to select part of test suite to be run
  - Test cases selected can be categorized
    1) **Reusable Test cases** exercise **only** unmodified code and unmodified specification, and do not be to run, but may be saved and used in succeeding regression cycles.
    2) **Retestable Test Cases** exercise modified code or test modified specification and should be run.
    3) **Obsolete Test Cases** can't be used in succeeding cycles.

What's the strategy

There are three ways that a test case may become **obsolete**



1. t∈T is defect specific and can't be used in succeeding cycles. Smart way is to use t∈T is when relative defect occurs.
2. The modifications from P to P' change some equivalence classes. t∈T used to exercise a boundary, which is no longer a boundary. t∈T is obsolete as it does no longer test a construct boundary) of interest.
3. Program modification leads to (now) incorrect input-output relations due to specification modifications.

- **Prioritization of Test Cases**
  - Prioritize the test cases depending on business impact, critical & frequently used functionalities. Selection of test cases based on priority will greatly reduce the regression test suite.

Regression Test Selection Techniques

- **Random Selection**
  - Select randomly tests from test suite *T*
  - The tester decides how many tests to select
  - May turn out to be better than no regression testing at all
  - But may not even select tests that exercise modified parts of the code!
- **Selecting Modification Traversing Tests**
  - Selecting a subset of tests such that only tests that guarantee the execution of modified code and code that might be impacted by the modified code in **P`** are selected.
  - A technique that does not discard any test that will traverse a modified or impacted statement is known as a "**safe**" regression test selection technique.

Test Prioritization

- When very ***high quality software*** is desired, it might not be wise to discard test cases as in test selection.
- In such cases use **Test prioritization**:
  1. Ranking tests (1st, 2nd, …)
  2. Deciding to stop execution of tests after the n^{th} ranked test
- Test prioritization requires a **criterion**, or criteria for ranking
- Single criterion prioritization
  - **Criteria 1**: cost (e.g., execution time)
    - Tests with lower costs are ranked first while test with higher costs are ranked last.
  - **Criteria 2**: risk (expected risk of not executing a test)
    - Tests with higher risks are ranked first while test with lower risks are ranked last.
- One goal of prioritization is to increase the likelihood of **revealing faults earlier** in the testing process.

References:

http://www.guru.com/regression-testing.html
SOFTWARE ENGINEERING BY K.K.AGGARWAL, YOGESH SINGH