Scoring Regression Tests
for Establishing Release Quality

David Roland

SSQA January 9, 2007

http://www.geocities.com/the522roland/2ndFieldSW/scoring.pdf

Agenda

Answer these key quality assurance questions:

- How are we doing?
  - Using test scoring
  - Reporting quality quotients
- How do we know?
  - Automated regression testing
  - Using Matched versus Pass/Fail

How are we doing?

- When will it be done?
  - "all" tests are passed
- Where should we be now?
  - tests for current release are passed
- What's QA doing?
  - Percentage of needed tests available
- How good is the product now?
  - How many running tests pass

Quality Quotients

Assuming that Product Completion requires passing a minimum set of tests:

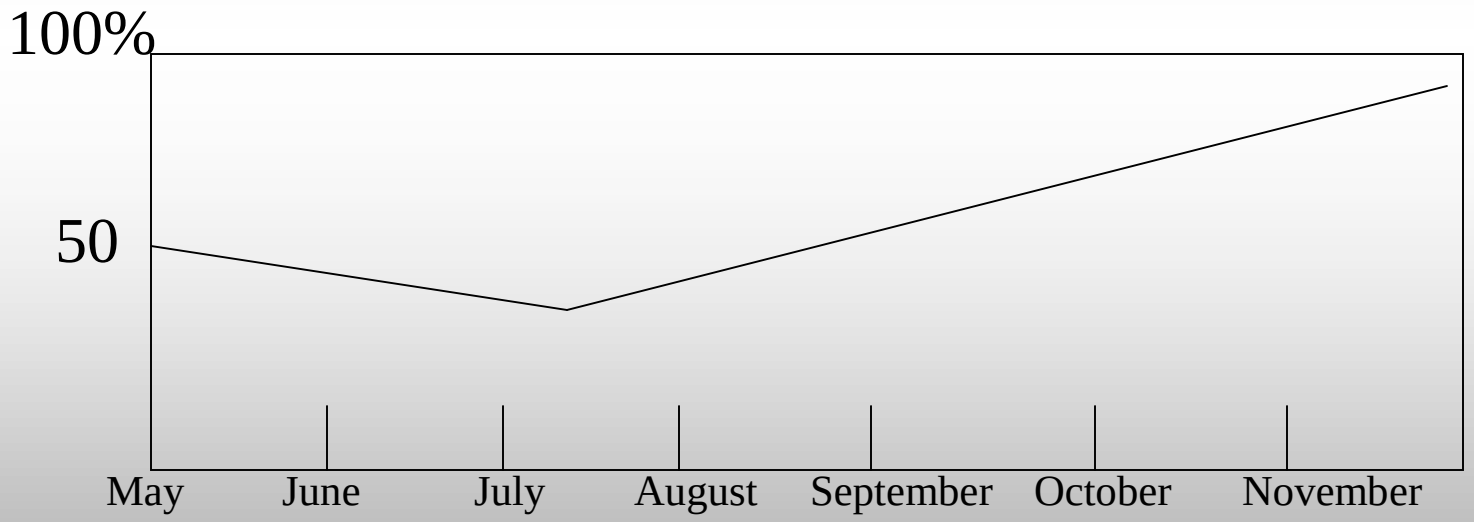
$$\frac{\text{Tests Passed}}{\text{Tests Available}} \Rightarrow \text{Current Product Quality}$$

$$\frac{\text{Tests Available}}{\text{Tests for Release}} \Rightarrow \text{QA Readiness}$$

$$\frac{\text{Tests for Release}}{\text{Tests for Product}} \Rightarrow \text{Product maturity}$$

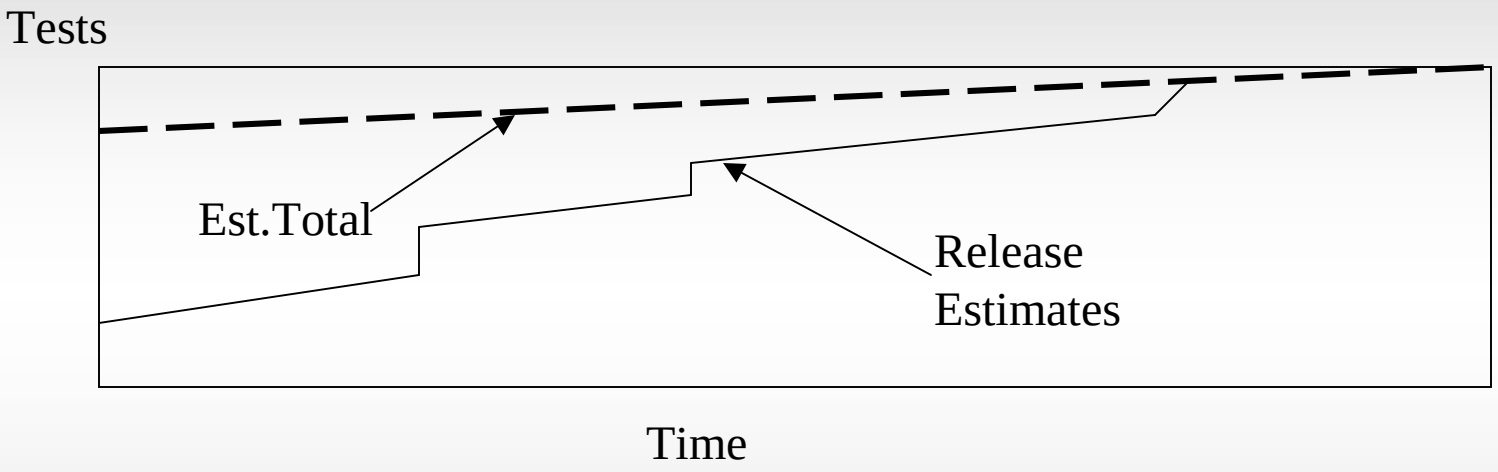
Changing Ratios

	Date:	5/4	7/11	11/23
$\frac{\text{Tests Passed}}{\text{Tests Available}}$	=	$\frac{50}{100}$	$\frac{100}{300}$	$\frac{950}{1000}$



Quality Ratio - % Tests Passed

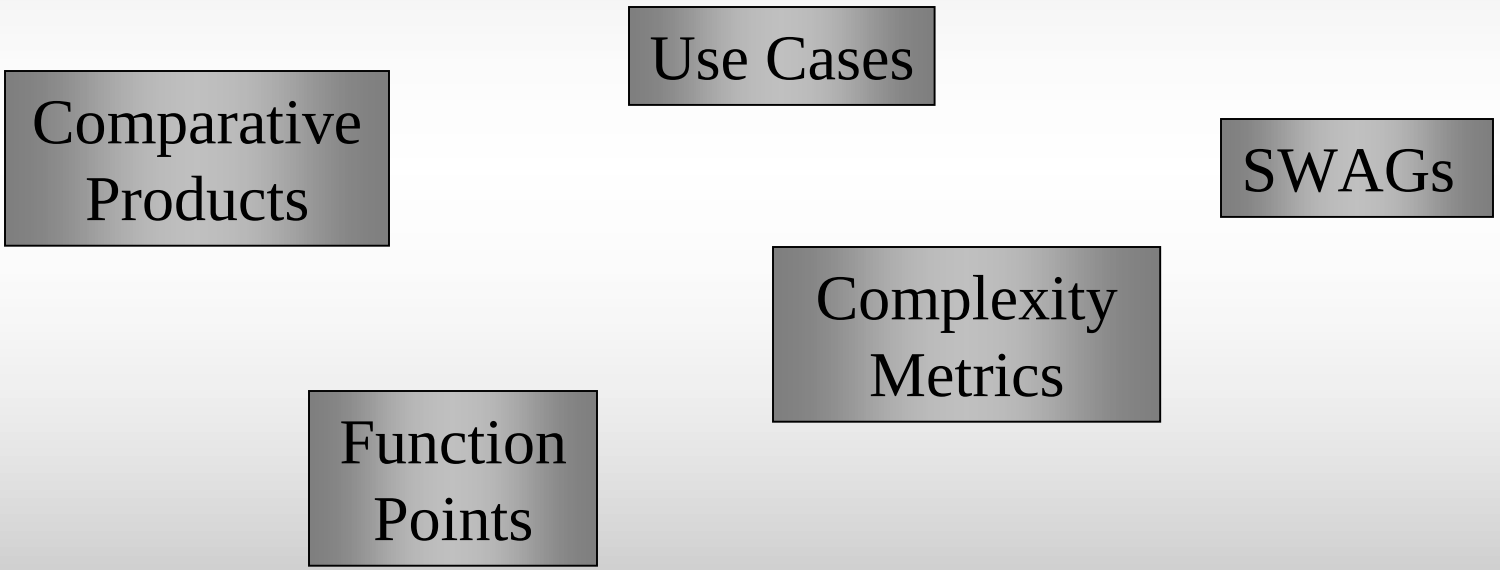
All the values change over time



Estimated tests by release during a project

Getting the Test Estimates

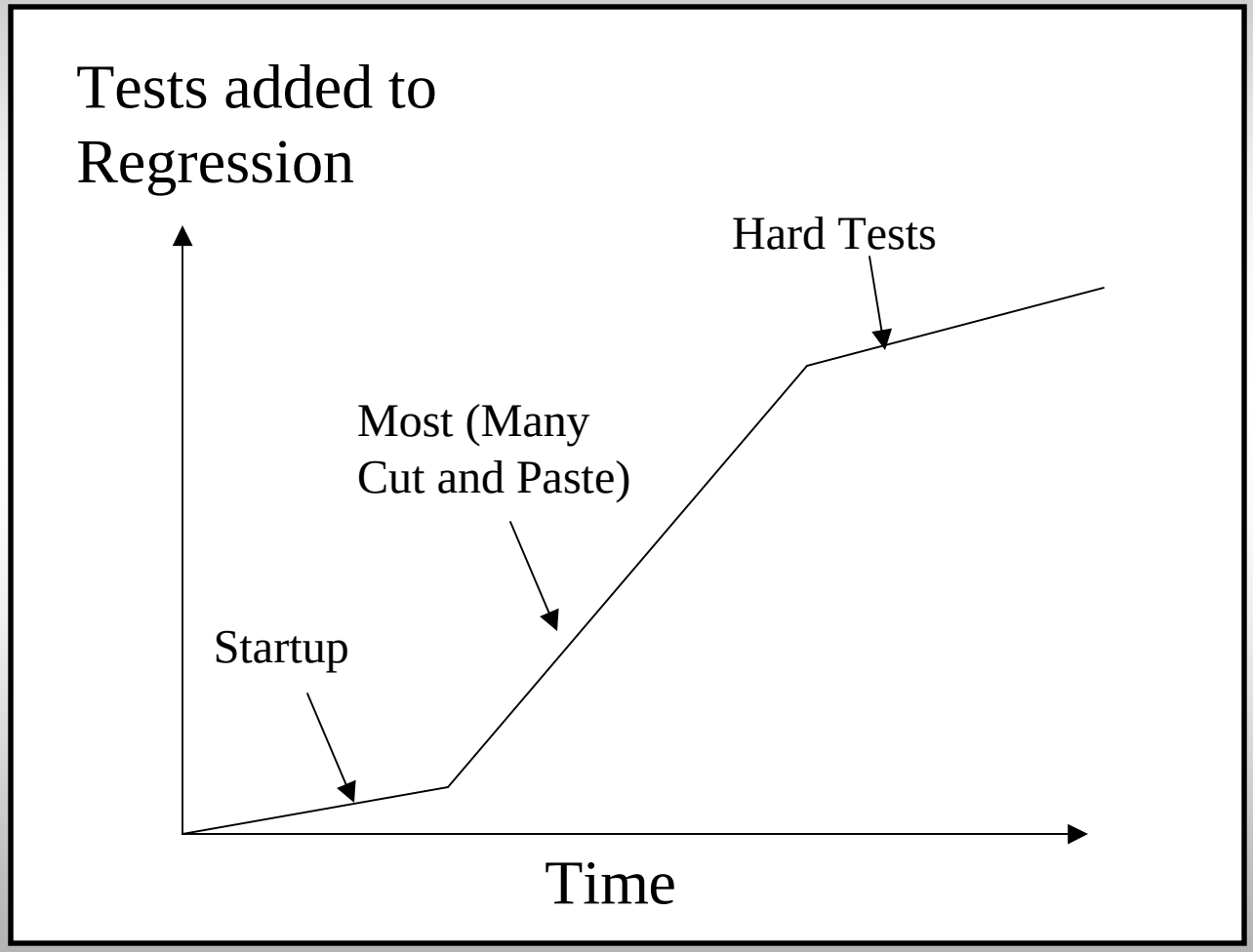
Estimating the total tests for the product:



Example Test Count Estimates

- Use Cases
  - 40 Use Cases with average 10 Scenarios
  - Average 10 test cases per scenario
  - Estimated Total Tests = 40 X 10 X 10 = 4000
- Complexity Metrics
  - McCabe Cyclomatic => Minimum Unit Tests
  - Add average additional count, e.g., 10 for multithreads, etc.

Planned Test Case Development



Reporting Progress

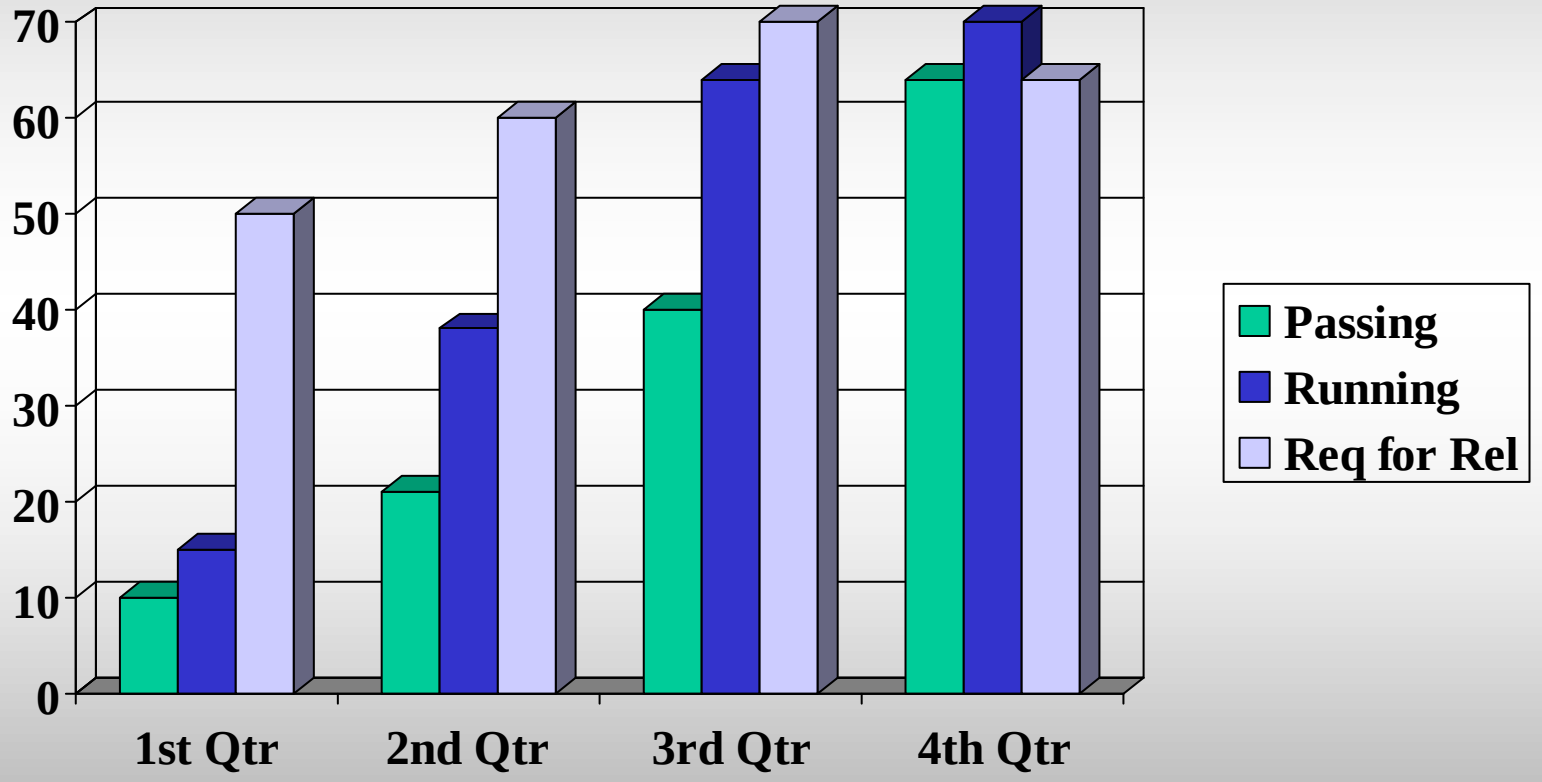
- Tracking score sheet
  - Score sheet for tests planned, running, passing
  - Can be spreadsheet or database
  - Correlate to requirements (e.g. Use Cases)
- Summary Graphs
  - Rolls up test status
  - Can be weekly, monthly, or quarterly

Test Matrix

Add new column here

Scenario/Testcase	Name	Status	Current	Rel 0.9	Rel 0.8	Rel 0.7
Read Operations	read.in	4/5/6	2/4	3/4	1/2	1/1
Read Null	read01	R	F	F		
Read Legal Value	read02	R	P	P	P	P
Read Value Too Big	read03	R	F	P	F	
Read Value Too Small	read04	R	P	P		
Read Ill-formed Value	read05	U				
Write Operations	write.in	6/8/9	5/6			

Management Reporting



How Do We Know?

Regression Testing:

"… the practice of comparing the results of a program … with a fixed set of expected results, in order to verify that it continues to behave as *expected* from one version to the next."

John Lakos, Large-Scale C++ Software Design pg 156.

Elements of Regression Testing

- Repeated use of test cases
  - Same setup and data as last version
  - Execute test, capture results and evaluate
- Comparison to *expected* results
  - Manual tests: humans understand equivalents
  - Automation: Requires consistent output
- Changes are flag for human intervention...

Human Intervention

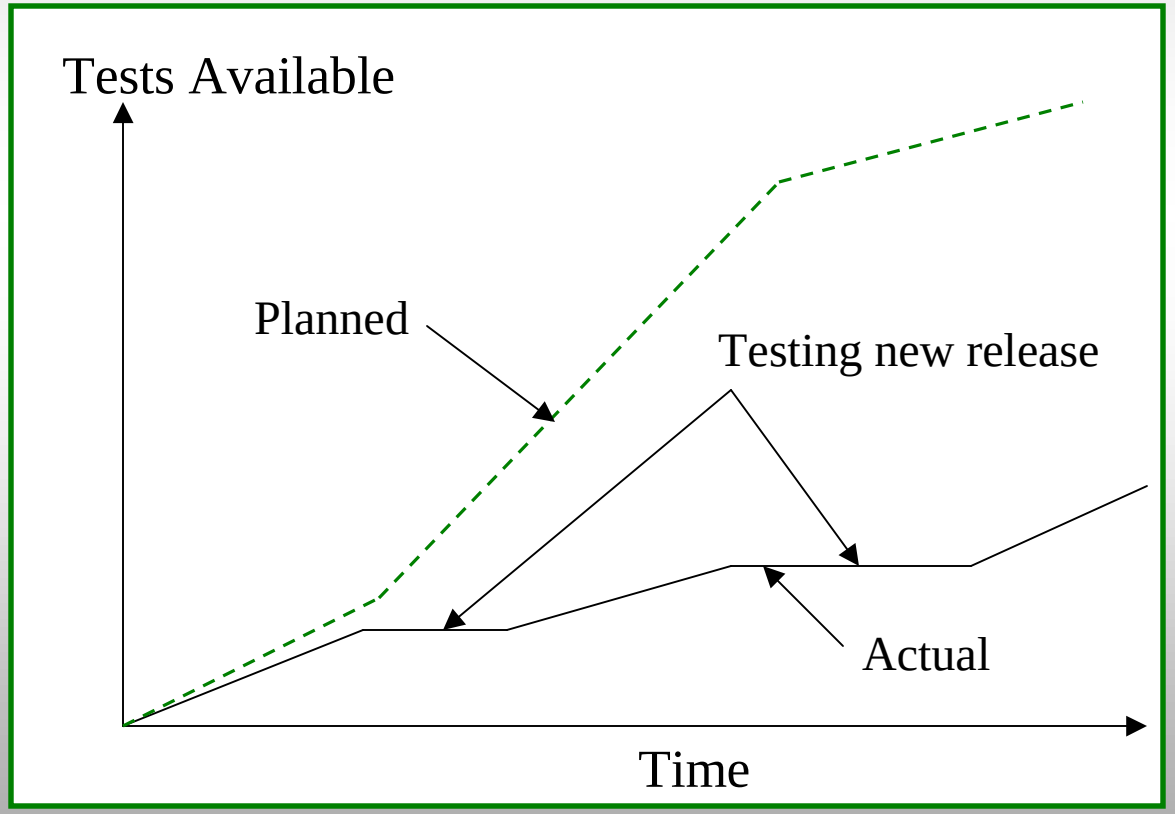
- When results are not as *expected;* but ...
- What result is *EXPECTED?*
  - Truth-based (assertion) testing expects correct
  - Change-based expects previous (maybe wrong)
- Change-based limits investigations:
  - 1000 tests
  - 100 failing
  - 10 changed (different than last baseline)

Change Causes/Actions

- Bad to Good - change test status in matrix
- Good to Bad - generate Incident Report
  - Bad product changes product's status in matrix; generates bug report
  - Bad test changes test's status in matrix; generates test change request
- Good to Good, i.e., different but OK
- Bad to Bad - i.e., different but still broken

Goal of Automation

Do each test only once manually, providing time to create more tests



Hurdles to Automation

- Time spent testing
- Tool to drive testing
- Time spent testing
- Debugging when product fails
- Tools to filter variable results
- Time spent testing
- Maintaining test cases
- Testing prevents building infrastructure

Automated Regression Flow

--- make - causes a regression to run after build
    --- test_driver -r runId -f list_of_tests
        --- for each test case in the input list*
            --- enter into test run log
            --- (if exists basename.presh) start it
            --- execute the test script; output to result file
            --- (if exists basename.postsh) start it
            --- (if exists basename.bas) diffs against result -
                report MATCH/NOT MATCH and
                put differences into the run log
        --- next test case
    --- Summarize the tests run in log
--- Investigate and act on tests NOT MATCHED

Automation Infrastructure

- Tools to drive tests and report results...
- Tools to setup testing environment...
- Tools to filter variability from results...
- Tools to update score sheet
- Staff to automate developers' tests
- Staff to maintain tests

Automation Tool…

- Runs sets of test cases
  - Provides way to set initial conditions
  - Executes a test, capturing results
  - Prepares results and compares to expectation
- Many sources
  - Commercial
  - Public Domain
  - Roll your own

Testing Environments…

- Comprehensive
  - Functional (i.e., “toy”) test cases
  - Operational (i.e., “real”) test cases
- Complex initial conditions
  - Database in known state
  - Event logs cleared
- Current challenges
  - Ported applications, e.g, UNIX, NT, Browsers
  - Client/Server - testing each side independently

Preparing/Comparing Results…

- Dates and other uncontrollable variables
- Cross platform variations
- Reordering results
  - Multithreading
  - Networking
- Multiple sources, e.g., databases, event logs
  - Tools to extract and format “hidden” results
  - Tools to combine for baselining

Preparing a Testcase

- Prepare testcase and its setup testcase.presh
- Run testcase to obtain results
- Determine whether product/test works
- If necessary - fix test or report defect
- Update score sheet - Test Status, Pass/Fail
- Prepare testcase.postsh to neutralize results
- Create baseline (i.e., neutralized results)
- Add testcase to regression suite

Non-Automated Tests

- Manual setup required
- Manual intervention (e. g., pull power cord)
- Dangerous to people or equipment
- Random data testing (pseudo may be OK).
- Long running (e.g., 24x7x365)
- Performance (usually needs custom config)

Scoring Release Readiness

- Concrete measures of progress
  - Parameters and ratios that provide metrics
  - Easy score sheet and reporting
  - Quantifies QA contribution and status
- Automated regression testing important
  - Baseline matching minimizes human review
  - Frees staff time to develop more tests