



Testing

Why Testing

- Researchers investigate many approaches to improving software quality
- But the world tests
- > 50% of the cost of software development is testing
- Testing is consistently a hot research topic



Testing Practice

Outline

- Manual testing
- Automated testing
- Regression testing
- Code coverage
- Bug trends

Manual Testing

- Test cases are lists of instructions
 - "test scripts"
- Someone manually executes the script
 - Do each action, step-by-step
 - Click on "login"
 - Enter username and password
 - Click "OK"
 - ...
 - And manually records results
- Low-tech, simple to implement

Manual Testing

- Manual testing is very widespread
 - Probably not dominant, but very, very common
- Why? Because
 - Some tests can't be automated
 - Usability testing
 - Some tests shouldn't be automated
 - Not worth the cost

Manual Testing

- Those are the best reasons
- There are also not-so-good reasons
 - Not-so-good because innovation could remove them
 - Testers aren't skilled enough to handle automation
 - Automation tools are too hard to use
 - The cost of automating a test is 10X doing a manual test

Topics

- Manual testing
- Automated testing
- Regression testing
- Code coverage
- Bug trends

Automated Testing

- Idea:
 - Record manual test
 - Play back on demand
- This doesn't work as well as expected
 - E.g., Some tests can't/shouldn't be automated

Fragility

- Test recording is usually very fragile
 - Breaks if environment changes
 - E.g., location of a textbox
 - Code changes - the name of dialog changes.
- More generally, automation tools cannot generalize a test
 - They literally record exactly what happened
 - If anything changes, the test breaks
- Maintaining tests is a lot of work
 - Broken tests must be fixed; this is expensive
 - Cost is proportional to the number of tests
- A hidden strength of manual testing
 - Because people are doing the tests, ability to adapt tests to slightly modified situations is built-in

Improved Automated Testing

- Recorded tests are too low level
 - E.g., every test contains the name of the dialog box
- Need to abstract tests
 - Replace dialog box string by variable name X
 - Variable name X is maintained in one place
 - So that when the dialog box name changes, only X needs to be updated and all the tests work again

Data Driven Testing (for Web Applications)

- Build a database of event tuples
 - < Document, Component, Action, Input, Result >
- E.g.,
 - < LoginPage, Password, InputText, \$password, "OK">
- A test is a series of such events chained together
- Complete system will have many relations
 - As complicated as any large database

Discussion

- Testers have two jobs
 - Clarify the specification
 - Find (important) bugs
- Only the latter is subject to automation
 - The oracle problem.
- Helps explain why there is so much manual testing

Topics

- Manual testing
- Automated testing
- Regression testing
- Code coverage
- Bug trends

Regression Testing

- Idea
 - When you find a bug,
 - Write a test that exhibits the bug,
 - And always run that test when the code changes,
 - So that the bug doesn't reappear
- Without regression testing, it is surprising how often old bugs reoccur

Regression Testing (Cont.)

- Regression testing ensures forward progress
 - We never go back to old bugs
- Regression testing can be manual or automatic
 - Ideally, run regressions after every change
 - To detect problems as quickly as possible
- But, regression testing is expensive
 - Limits how often it can be run in practice
 - Reducing cost is a long-standing research problem

Research in Regression Testing

- Test selection
 - A change is made at line x , should I rerun the whole regression set?

Efficient Regression Testing

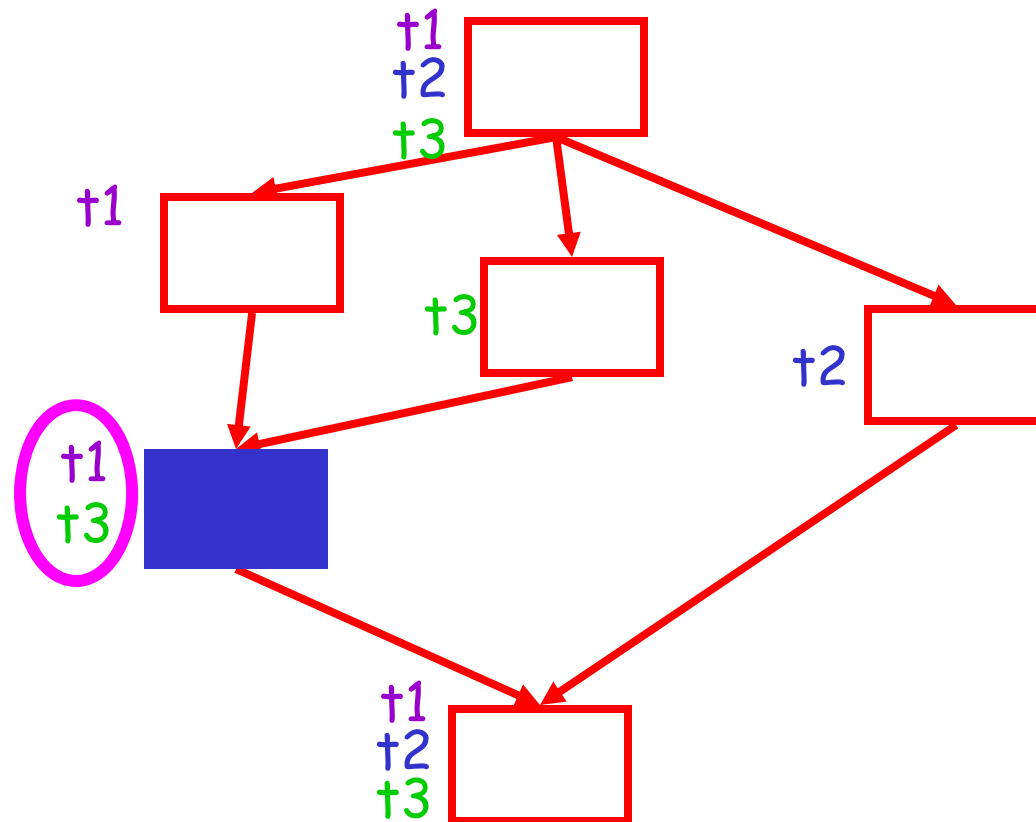
- Problem: Regression testing is expensive
- Observation: Changes don't affect every test
 - And tests that couldn't change need not be run
- Idea: Use a conservative static analysis to prune test suite

The Algorithm

Two pieces:

1. Run the tests and record for each basic block which tests reach that block
2. After modifications, do a DFS of the new control flow graph. Wherever it differs from the original control flow graph, run all tests that reach that point

Example



Label each node of the control flow graph with the set of tests that reach it.

More

- Test minimization
 - A test suite is often redundant, select a minimal set that satisfies a certain criterion
- Test prioritization
 - Assign test cases different priorities given certain constraints.
 - Greedy algorithms

A Problem

- How do we know when we are done?
 - Could keep going forever
- But, testing can only find bugs, not prove their absence
 - We need a proxy for the absence of bugs

Topics

- Manual testing
- Automated testing
- Regression testing
- Code coverage
- Bug trends

Code Coverage

- Idea
 - Code that has never been executed likely has bugs
- This leads to the notion of *code coverage*
 - Divide a program into units (e.g., statements)
 - Define the coverage of a test suite to be

$$\frac{\text{\# of statements executed by suite}}{\text{\# of statements}}$$

Code Coverage (Cont.)

- Code coverage has proven value
 - It's a real metric, though far from perfect
- But 100% coverage does not mean no bugs
 - E.g., a bug visible after loop executes 1,025 times
- And 100% coverage is almost never achieved
 - Infeasible paths
 - Ships happen with < 60% coverage
 - High coverage may not even be desirable
 - May be better to devote more time to tricky parts with good coverage

Using Code Coverage

- Code coverage helps identify weak test suites
- Code coverage can't complain about missing code
 - But coverage can hint at missing cases
 - Areas of poor coverage indicate areas where not enough thought has been given to specification

More on Coverage

- Statement coverage
- Edge coverage
- Path coverage
- Def-use coverage

Mutation Coverage

- Create mutations of the subject program by performing simple transformations
 - $x < y$ transforms to $x \leq y$, $x < y + c$, ...
- Kill set = all mutations P' s.t. there exists a test case t $P(t) \neq P'(t)$
 - Adequacy = $|\text{kill set}| / \# \text{ of mutations}$

Topics

- Manual testing
- Automated testing
- Regression testing
- Code coverage
- Bug trends

Bug Trends

- Idea: Measure rate at which new bugs are found
- Rational: When this flattens out it means
 1. The cost/bug found is increasing dramatically
 2. There aren't many bugs left to find

The Big Picture

- Standard practice
 - *Measure progress often*
 - *Make forward progress* (regression testing)
 - *Stopping condition* (coverage, bug trends)

Test Generation

- Combinatorial testing
- Concolic testing.