An Exploration into the Current State of Test-First vs. Test-Last Testing

Christopher Morris Thomas

Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA

2013 Fall Senior Seminar

Outline

Outline

Why Do we Care about testing methods?

In 2007, a study it was found that businesses spent at least 50% of their production costs on product verification and testing.

Because of this, software businesses are constantly looking to optimize their testing practices in an attempt to reduce costs.

Software Testing Defined

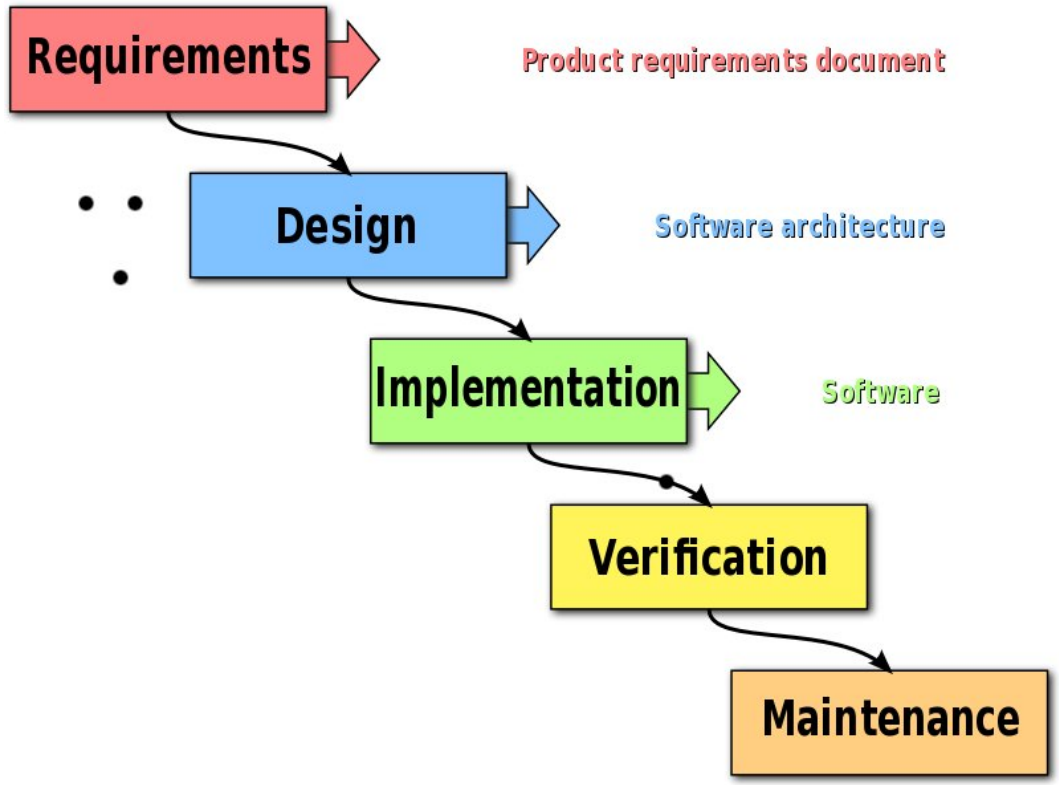
Software testing is a branch of software engineering that uses various practices to:

- identify potential malfunctions in code
- demonstrate functionality of software

Testing is usually either done manually or automatically using test code.

Waterfall Development

Wikipedia Commons
commons.wikimedia.org/wiki/File:Waterfall_near_Brienzersee.jpg



Wikipedia
en.wikipedia.org/wiki/File:Waterfalldevelopment_model.svg

Test-Last Testing

Test-last testing is:

- Writing tests after the code
- Used in waterfall development
- Considered the “conventional testing method"

Agile Software Development

Agile development was developed in response to criticisms of waterfall methods.

Some common traits of agile development:

- Time boxed iterations
- working sub-products
- Customer communication
- Test driven development

Test-First Testing

Test-first testing is:

- writing automated tests before the code is written
- often used to define code functionality
- usually used as part of a development model

Test Driven Development

Test driven development (TDD) is the most well known and used test-first development model. Currently, TDD refers to a wide variety of different test-first methods that are similar to the original TDD method as defined by Kent Beck.

The original TDD method consists of three steps that are repeated until the program is complete:

- Write a new failing test
- Write code to make tests pass
- Optimize your new code

Outline

Measuring Test Methods

When it comes to comparing two test methods there is no one attribute that proves one method is superior to another.

While there are many attributes that can be used to compare testing methods, I focused on three attributes:

- code coverage
- total development time
- code correctness

Article review by Kollanus

In 2010, Kollanus reviewed 40 different articles that provided empirical evidence comparing TDD to test-last development methods.

In the review Kollanus found that TDD methods:

- potentially had better code correctness
- potentially had longer development times
- suggested by one study increase code coverage

Lemos et al Study Setup

In 2012, Lemos et al preformed an experiment comparing test-first and test-last testing with auxiliary functions (10-200 lines of code).

In the study, Lemos had third year computer science students with basic test-first and test-last knowledge solve coding problems using test-first and test-last development methodologies.

Lemos et al Results

After the study was completed Lemos found that test-first testing:

- produced 40% more code coverage than test-last testing
- took 12% longer to implement than test-last testing
- did not increase code correctness

Contradictory Studies

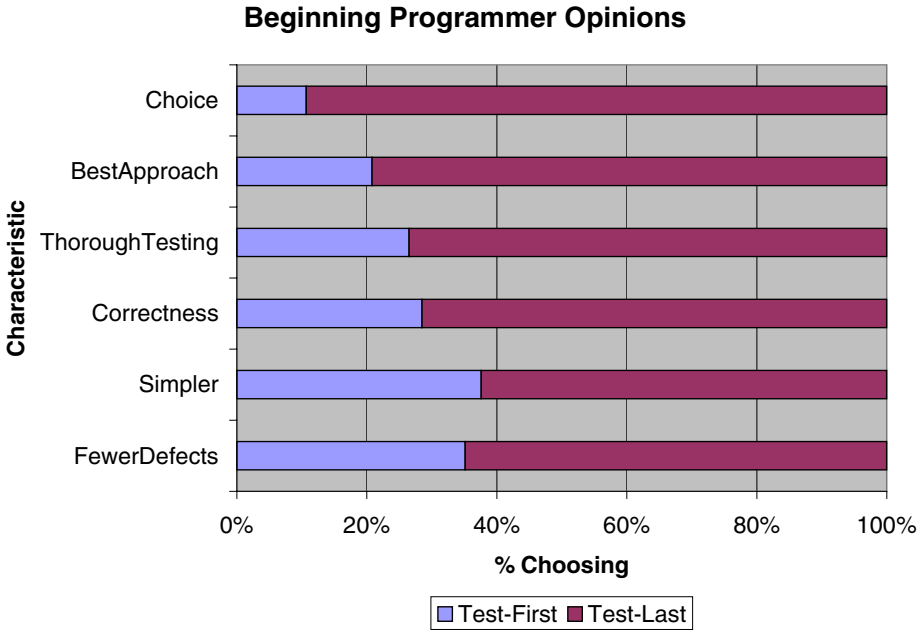
One large problem currently in the test-first vs test-last debate is that many studies contradict each other.

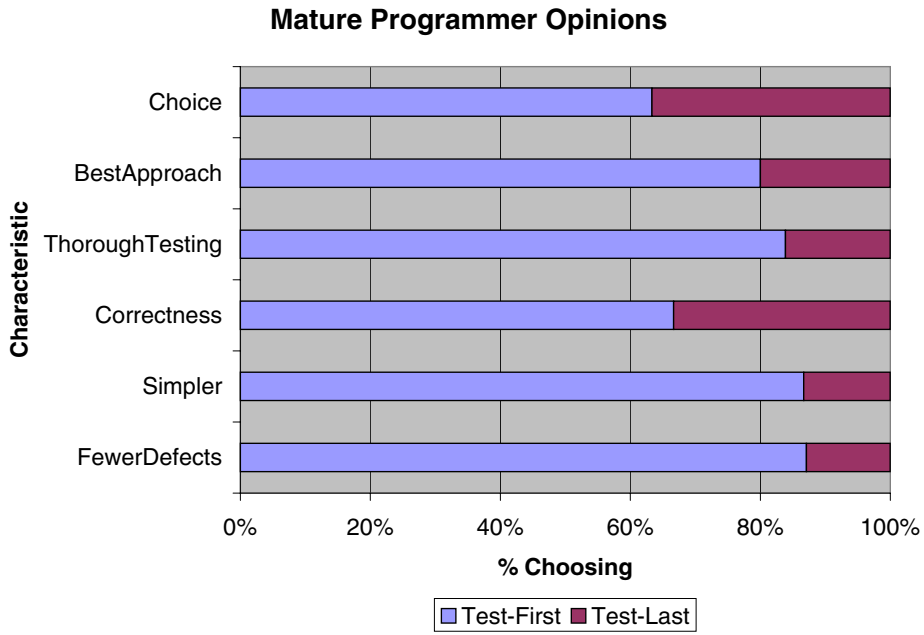
Although no one currently knows exactly why these contradictions occur, a few theories exist. Three potential problems in current TDD research that could produce contradictions are:

- method difficulty
- lack of TDD method documentation
- TDD conformance issue

Method Difficulty

Results from an opinion study by Janzen et al:





Conclusions

Due to current issues in contradictory data it is hard to make solid conclusions.

That being said some strong trends were noticed:

- test-first testing takes at least as much time as test-last testing
- test-first testing tends to increase code coverage
- test-first testing has at least the same code correctness as test-last testing
- test-first testing is potentially more difficult then test-last testing

Outline

Difficulty of TDD

Most the data for test-first testing comes from studies using TDD. This means that the results have a potential to reflect properties of TDD and not test-first testing.

Although all the trends have this potential issue. The most worrisome data point is method difficulty. This is because many articles noted that TDD can be a hard to implement for reasons other then test-first testing.

Difficulty of TDD Cont.

In a survey given to participants of a series of TDD experiments it was noted that:

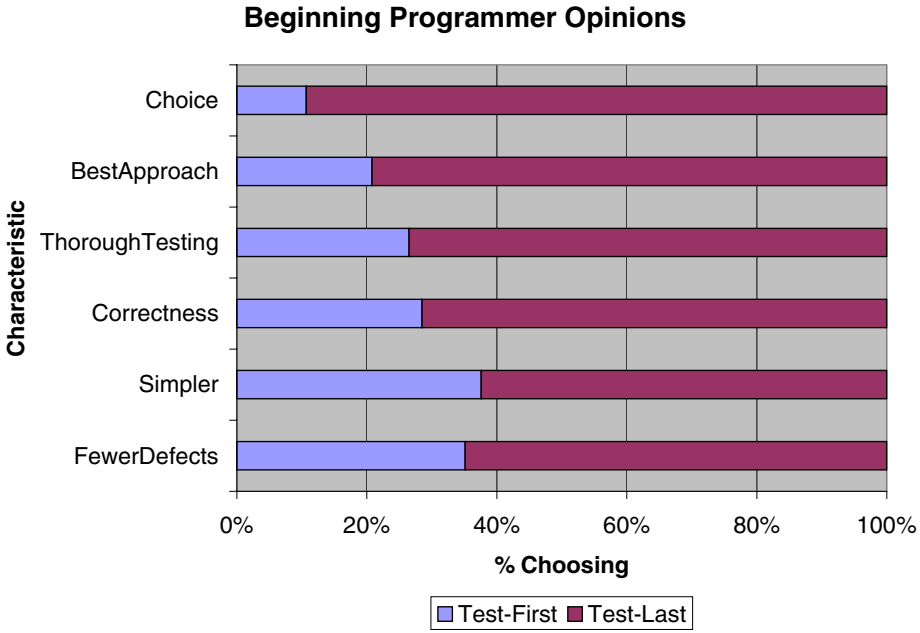
- 56% of participants said they had difficulty adapting to the TDD mindset
- 23% of participants said that lack of upfront design was found to be a hinderence

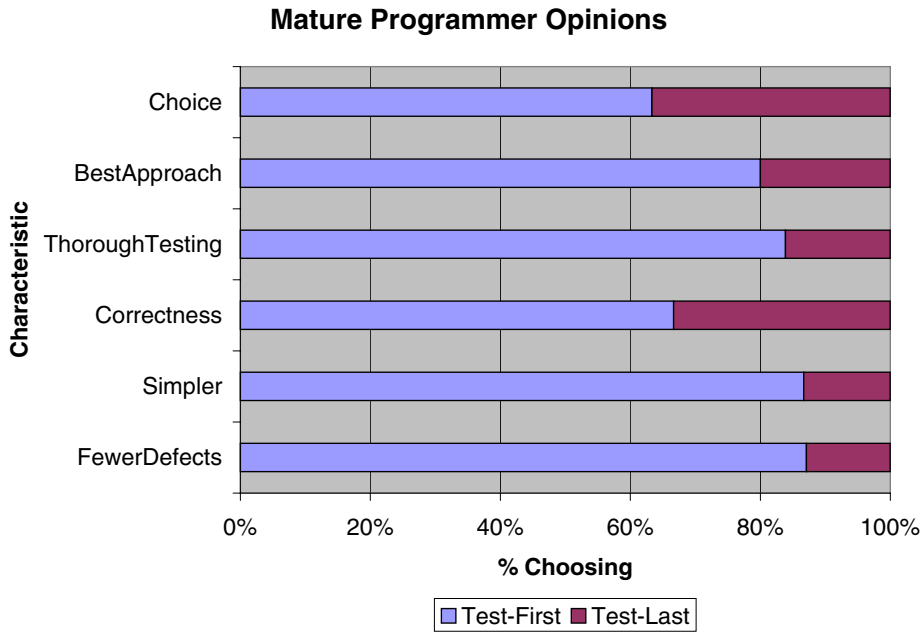
Another survey, given online, found that **25% of TDD programmers admitted to frequently or always making mistakes in following the steps of TDD**. Some of these mistakes include:

- writing tests that are too complex for effective TDD
- forgetting to optimize their new code

TDD Difficulty Cont. 2

Results from the opinion study by Janzen et al:





Things to Take Away From this Presentation

- currently testing can be divided into test-first and test-last testing
- test-first testing tends to be favored in Agile Development
- test-last testing tends to be favored in Waterfall Development
- Current data on comparing test-first to test-last testing tends to produce contradictory results over multiple studies, though a few main trends in the data exist
- TDD may not be the optimal way to preform test-first testing.