A Comparison of Inquiry-Based Conceptual Feedback vs. Traditional Detailed Feedback Mechanisms in Software Testing Education: An Empirical Investigation

Lucas Cordova

(Western Oregon University)

cordoval@wou.edu

Jeffrey Carver

(University of Alabama)

carver@cs.ua.edu

Gursimran Walia

(Georgia Southern University)

gwalia@georgiasouthern.edu

Noah Gershmel

(University of Alabama)

nggershmel@crimson.ua.edu

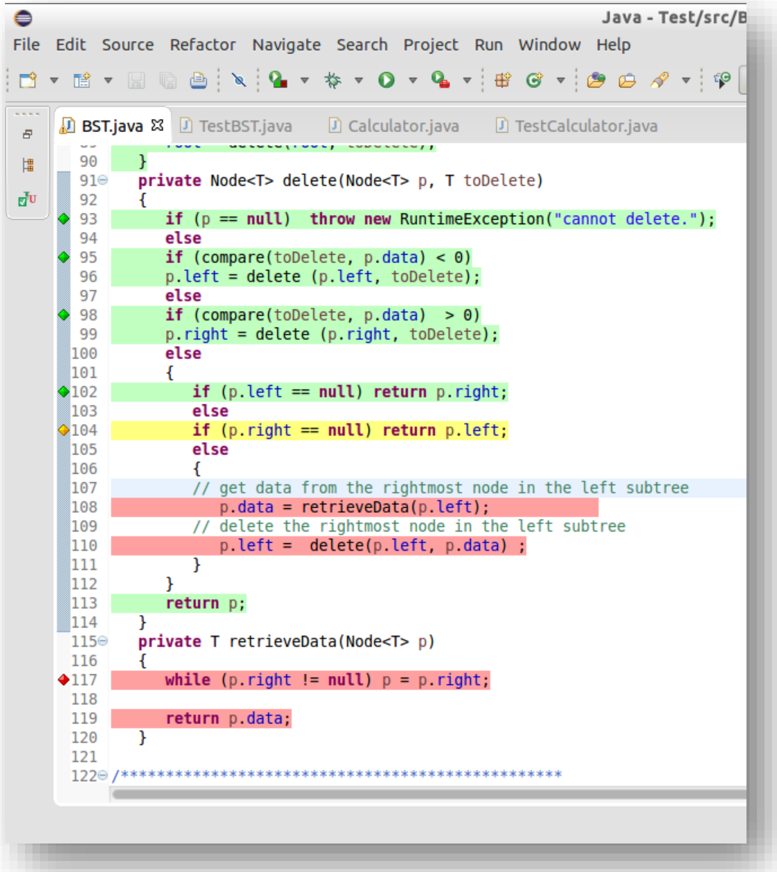
Problem Statement

Many students today are graduating with a knowledge gap about software testing

Many students resort to trial-and-error while programming and debugging

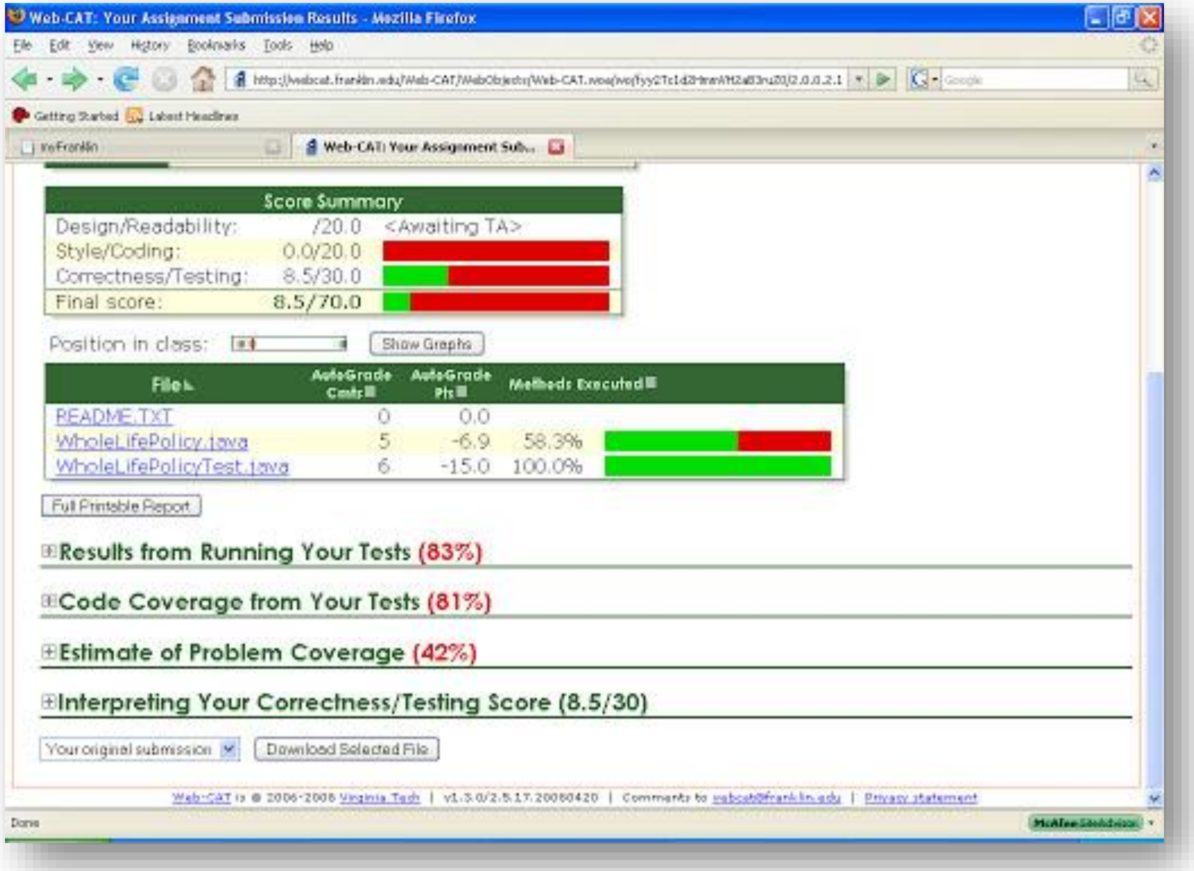
Current software testing pedagogical tools (e.g. WebCAT) encourage this behavior

Main Issues with Current Approaches



```
    root = delete(root, toDelete);
    }
    private Node<T> delete(Node<T> p, T toDelete)
    {
       if (p == null)  throw new RuntimeException("cannot delete.");
       else
       if (compare(toDelete, p.data) < 0)
       p.left = delete (p.left, toDelete);
       else
       if (compare(toDelete, p.data)  > 0)
       p.right = delete (p.right, toDelete);
       else
       {
          if (p.left == null) return p.right;
          else
          if (p.right == null) return p.left;
          else
          {
          // get data from the rightmost node in the left subtree
             p.data = retrieveData(p.left);
          // delete the rightmost node in the left subtree
             p.left =  delete(p.left, p.data) ;
          }
       }
       return p;
    }
    private T retrieveData(Node<T> p)
    {
       while (p.right != null) p = p.right;

       return p.data;
    }
 /*************************************************
```



Web-CAT: Your Assignment Submission Results - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://webcat.franklin.edu/Web-CAT/WebObjects/Web-CAT.woa/wo/fyy2Tc1d2HmnWH2a83ru20/2.0.0.2.1
Google
Getting Started
Latest Headlines
myFranklin
Web-CAT: Your Assignment Sub...
Show Graphs
Full Printable Report
Your original submission
Download Selected File
Done
McAfee SiteAdvisor

Score Summary		
Design/Readability:	/20.0	<Awaiting TA>
Style/Coding:	0.0/20.0	🟥
Correctness/Testing:	8.5/30.0	🟩🟥
Final score:	8.5/70.0	🟩🟥

Position in class:

File	AutoGrade Cmts	AutoGrade Pts	Methods Executed	
README.TXT	0	0.0		
WholeLifePolicy.java	5	-6.9	58.3%	🟩🟥
WholeLifePolicyTest.java	6	-15.0	100.0%	🟩

⊞Results from Running Your Tests (83%)

⊞Code Coverage from Your Tests (81%)

⊞Estimate of Problem Coverage (42%)

⊞Interpreting Your Correctness/Testing Score (8.5/30)

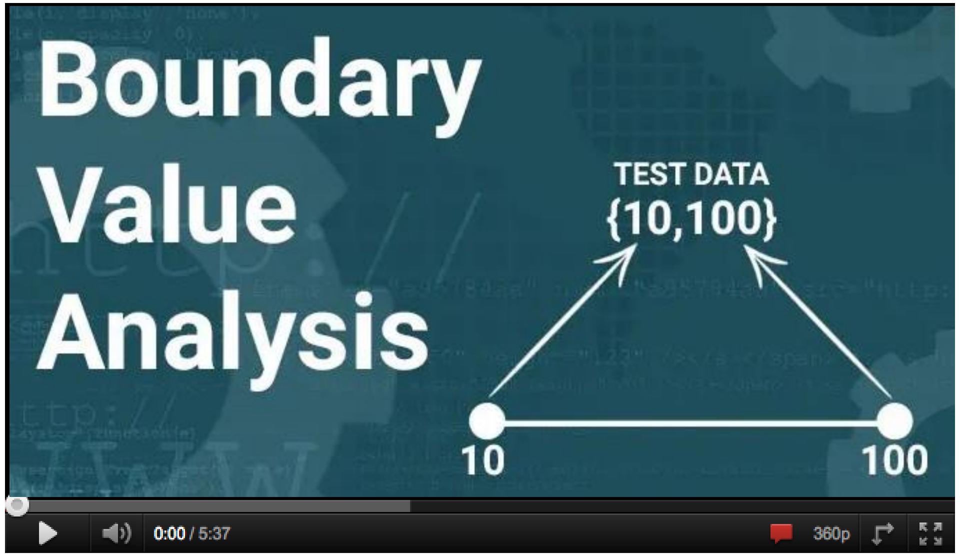
Web-CAT is © 2006-2008 Virginia Tech | v1.3.0/2.5.17.20080420 | Comments to webcat@franklin.edu | Privacy statement

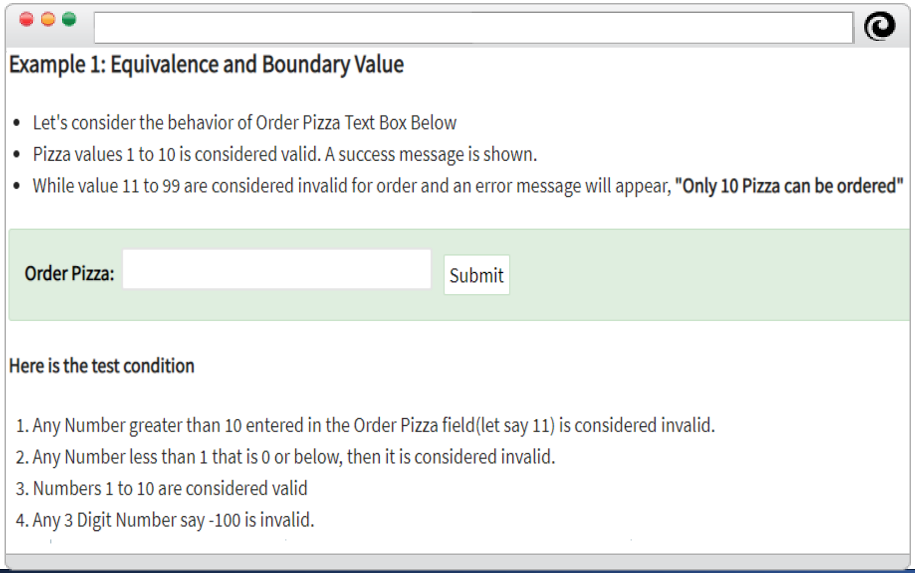
Alternative Approach: *Conceptual Feedback*

“The test suite has not fully tested all boundary conditions.”

“The test suite misses part of a compound Boolean expression.”

+ Resources





Example 1: Equivalence and Boundary Value

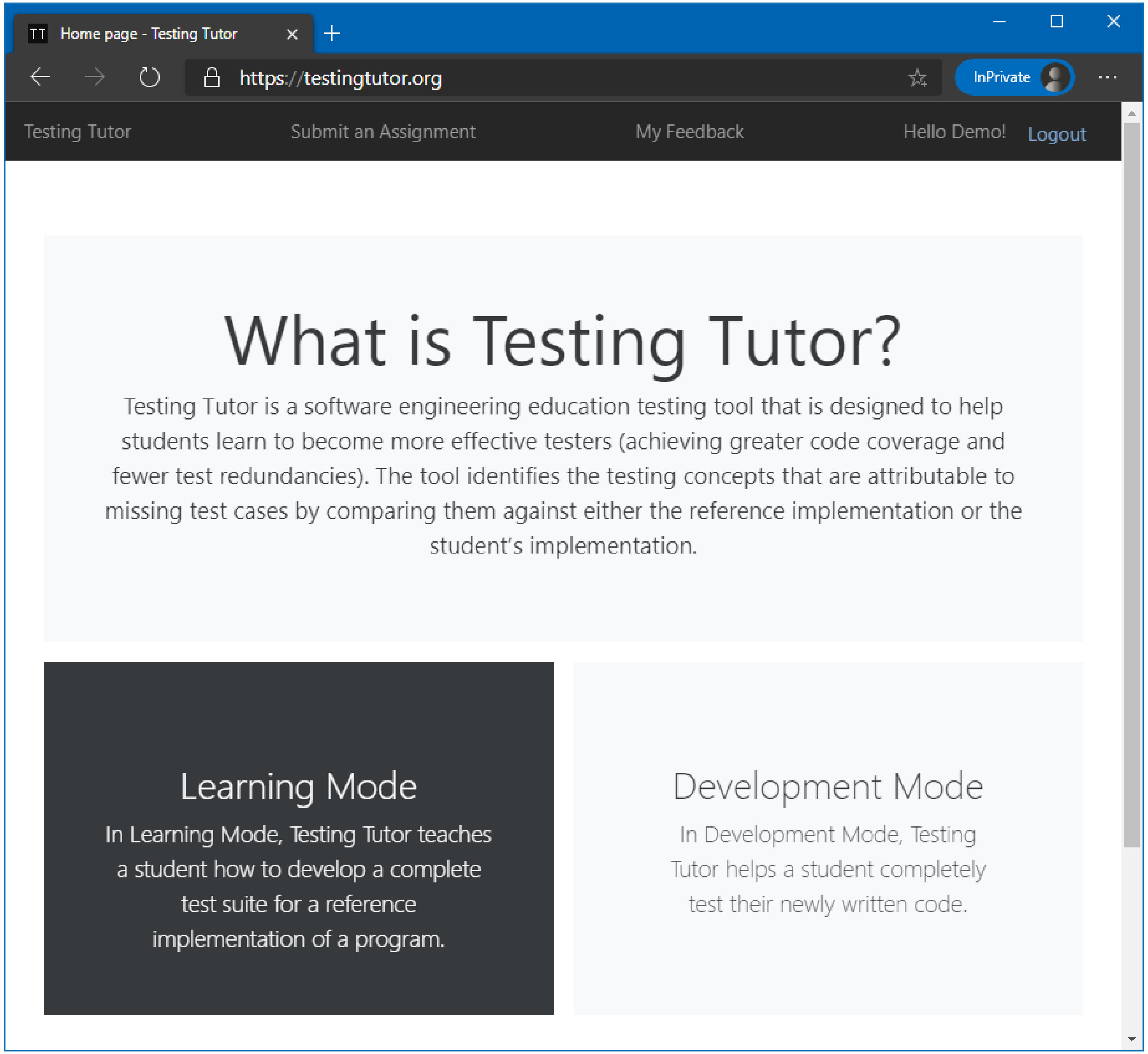
- Let's consider the behavior of Order Pizza Text Box Below
- Pizza values 1 to 10 is considered valid. A success message is shown.
- While value 11 to 99 are considered invalid for order and an error message will appear, **"Only 10 Pizza can be ordered"**

Order Pizza: Submit

Here is the test condition

1. Any Number greater than 10 entered in the Order Pizza field(let say 11) is considered invalid.
2. Any Number less than 1 that is 0 or below, then it is considered invalid.
3. Numbers 1 to 10 are considered valid
4. Any 3 Digit Number say -100 is invalid.

Proposed Solution: Testing Tutor



What is Testing Tutor?

Testing Tutor is a software engineering education testing tool that is designed to help students learn to become more effective testers (achieving greater code coverage and fewer test redundancies). The tool identifies the testing concepts that are attributable to missing test cases by comparing them against either the reference implementation or the student's implementation.

Learning Mode

In Learning Mode, Testing Tutor teaches a student how to develop a complete test suite for a reference implementation of a program.

Development Mode

In Development Mode, Testing Tutor helps a student completely test their newly written code.

Paper Goal

Assess whether conceptual feedback helps students produce better, more concise and comprehensive test suites

Experiment RQs

RQ1: How do different types of feedback (conceptual, detailed, none) affect the quality of student test suites?

RQ2: What is the students’ perception of the usefulness of Testing Tutor in terms of its usability and the feedback provided?

Studies Conducted

Study	Group A (Traditional Detailed Feedback)	Group B (Conceptual Feedback)
Spring 2019	15	16
Summer 2019	13	15

Dependent Variables

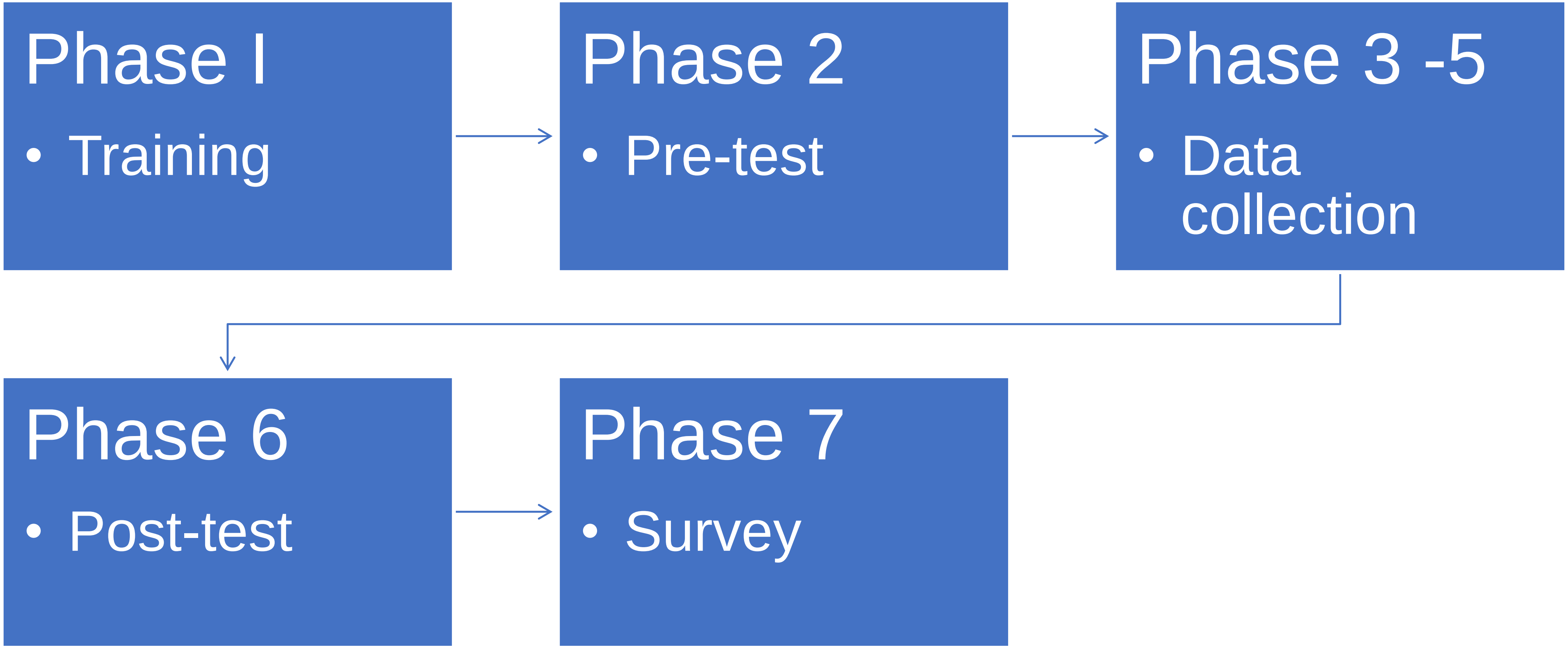
Testing Tutor collected

- Line coverage
- Branch coverage
- Conditional coverage
- # Redundant tests

Additional data gathered

- Assignment grade
- Perception of student understanding of the feedback

Study Design



Phase I

- Training

Phase 2

- Pre-test

Phase 3 -5

- Data collection

Phase 6

- Post-test

Phase 7

- Survey

Pre-test Results

Dependent Variable	Group A	Group B
Line Coverage	35%	35.7%
Branch Coverage	35.3%	34.9%
Conditional Coverage	35.1%	36.6%
Redundant Tests	4.86	4.9
Assignment Grade	57.95%	58.42%

Main Results

Dependent Variable	Treatment A (Traditional Detailed Feedback)	Treatment B (Conceptual Feedback)
Line Coverage	43.4%	55.1%
Branch Coverage	43.1%	52.7%
Conditional Coverage	45.4%	57.5%
Redundant Tests	4.86	3.33
Assignment Grade	60.37%	68.27%

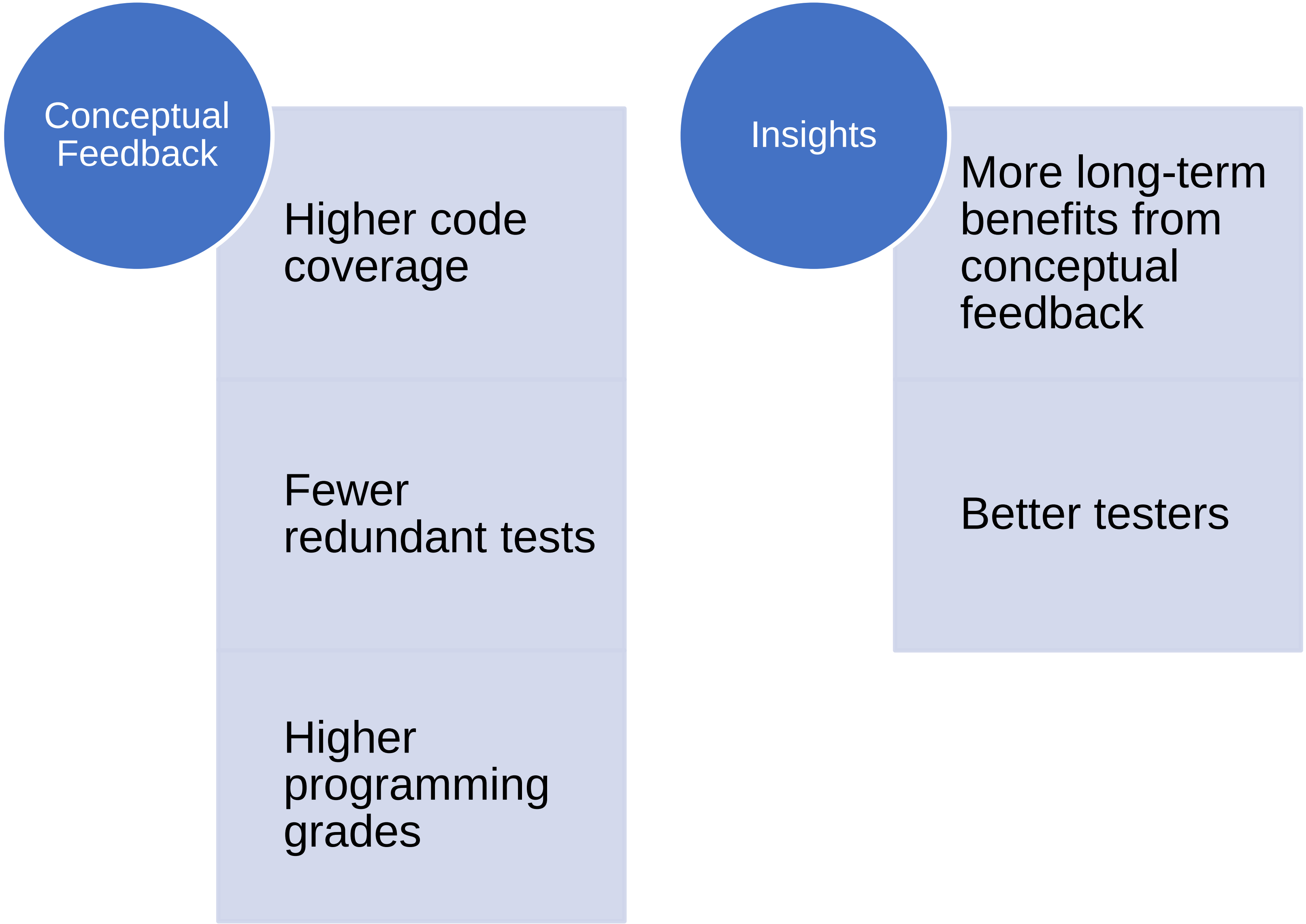
*[all differences significant p < .05]

Post-test Results

Dependent Variable	Group A	Group B
Line Coverage	37.9%	68.8%
Branch Coverage	38.6%	69.4%
Conditional Coverage	44.8%	72.6%
Redundant Tests	4.29	2.29
Assignment Grade	60.31%	78.95%

*[all differences significant p < .05]

Answers to RQ1 – Effects of Feedback



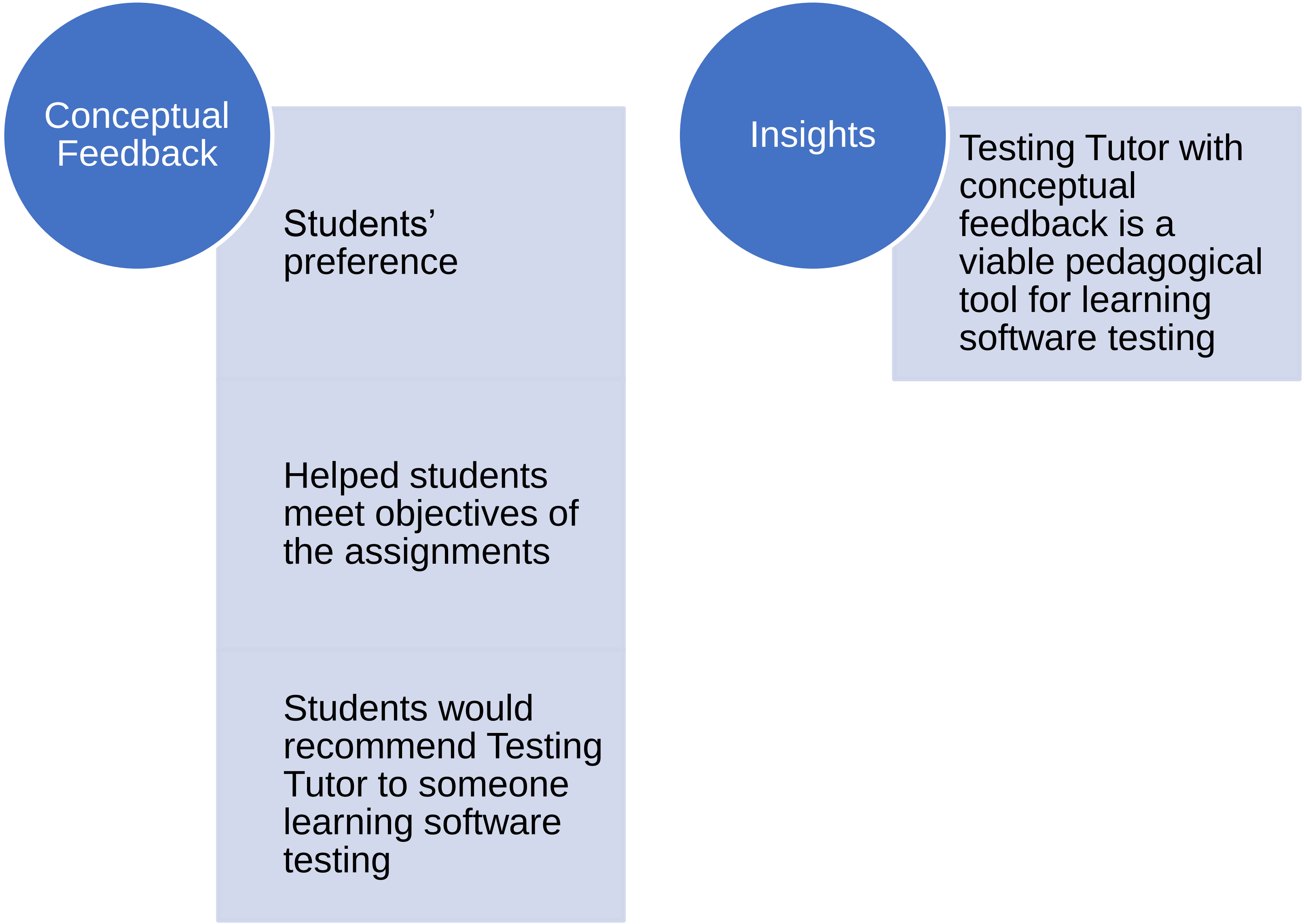
Conceptual Feedback

- Higher code coverage
- Fewer redundant tests
- Higher programming grades

Insights

- More long-term benefits from conceptual feedback
- Better testers

Answers to RQ 2 – Student Perceptions



Conceptual Feedback

Students' preference

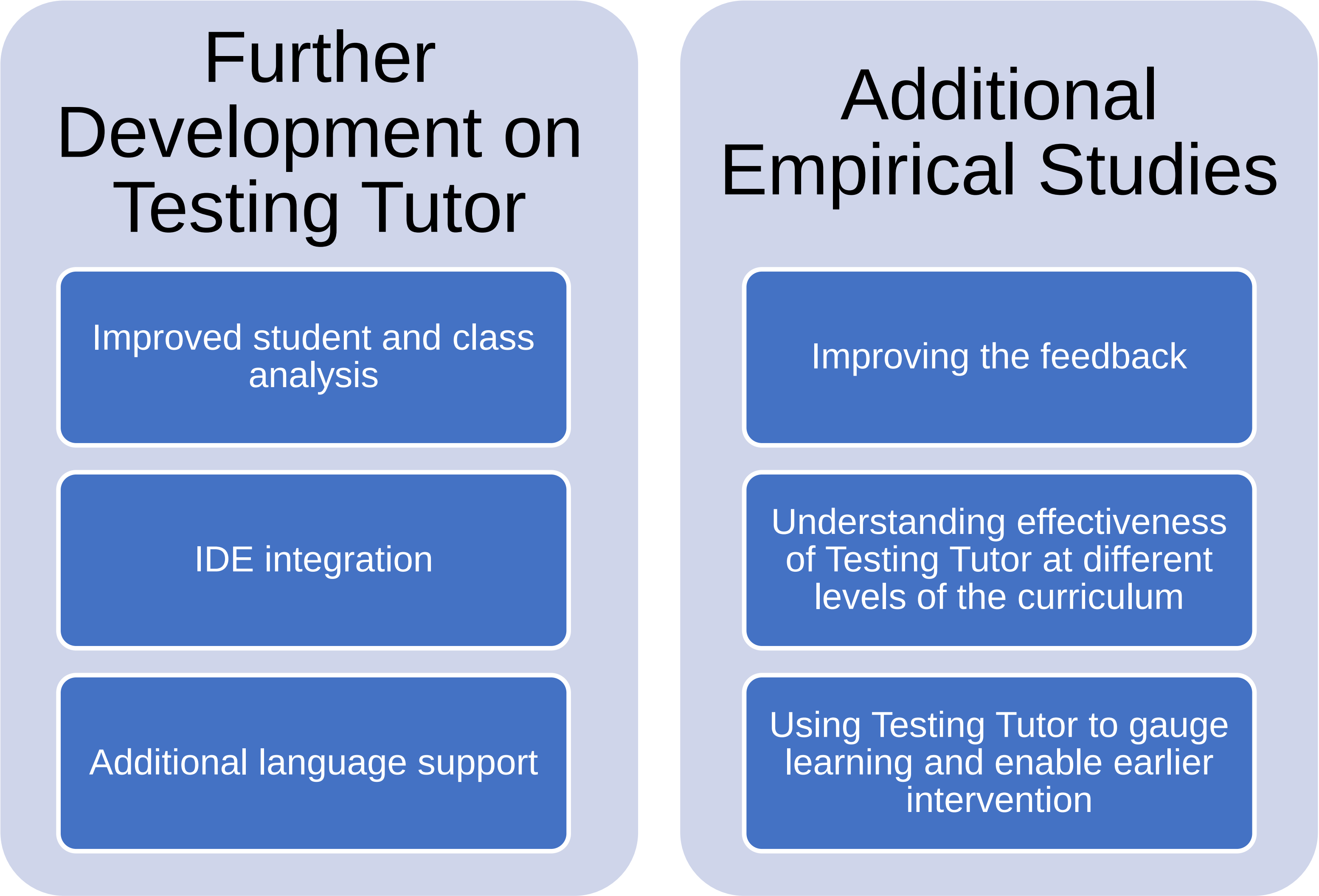
Helped students meet objectives of the assignments

Students would recommend Testing Tutor to someone learning software testing

Insights

Testing Tutor with conceptual feedback is a viable pedagogical tool for learning software testing

Future Work



Further Development on Testing Tutor

- Improved student and class analysis
- IDE integration
- Additional language support

Additional Empirical Studies

- Improving the feedback
- Understanding effectiveness of Testing Tutor at different levels of the curriculum
- Using Testing Tutor to gauge learning and enable earlier intervention

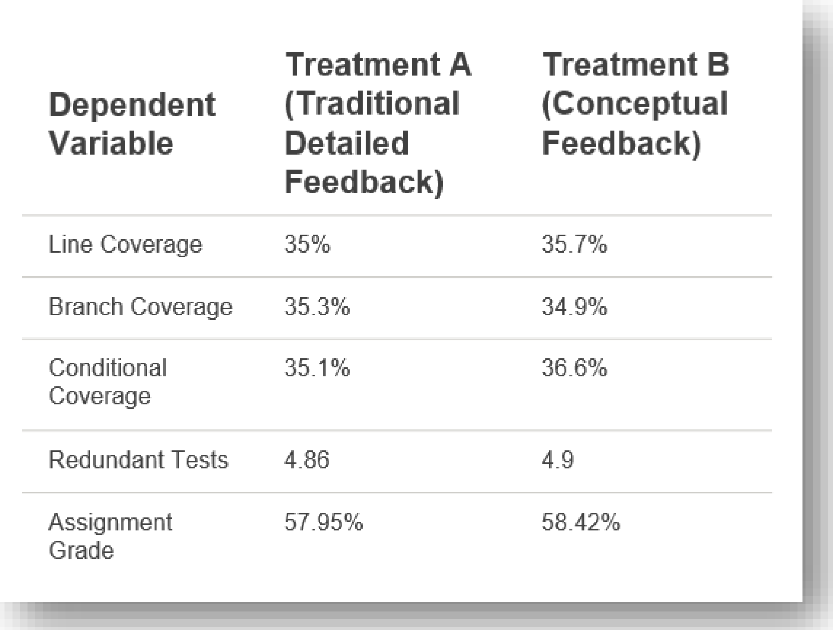
Beta Test

Interested in using Testing Tutor in early classes?

Contact us!

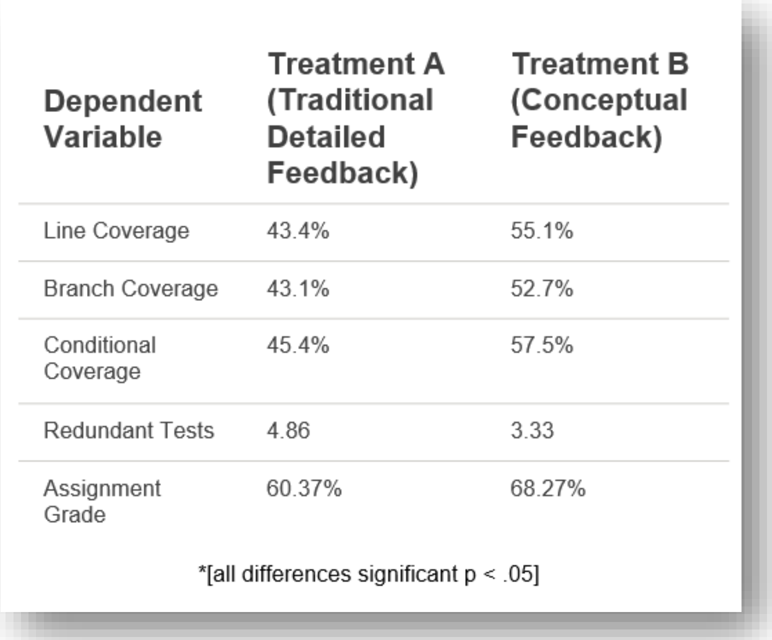
cordoval@wou.edu

Pre-test



Dependent Variable	Treatment A (Traditional Detailed Feedback)	Treatment B (Conceptual Feedback)
Line Coverage	35%	35.7%
Branch Coverage	35.3%	34.9%
Conditional Coverage	35.1%	36.6%
Redundant Tests	4.86	4.9
Assignment Grade	57.95%	58.42%

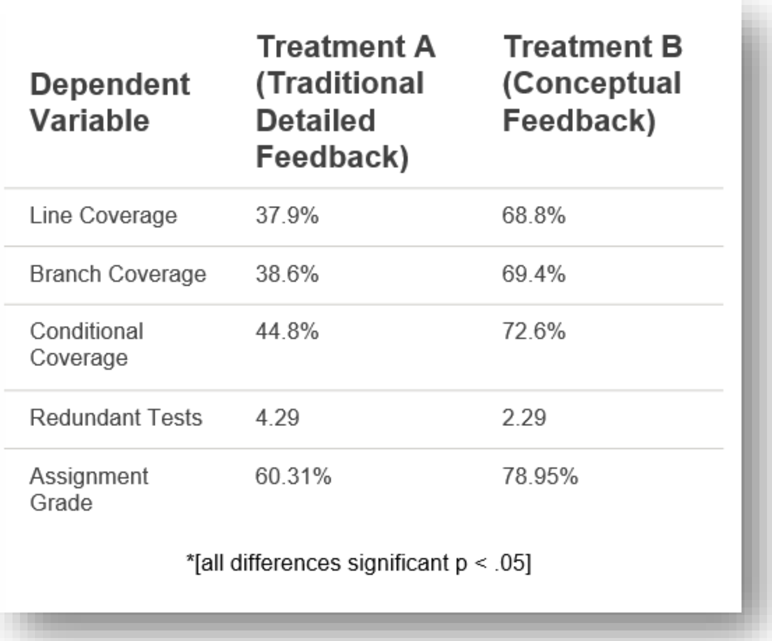
Main Results



Dependent Variable	Treatment A (Traditional Detailed Feedback)	Treatment B (Conceptual Feedback)
Line Coverage	43.4%	55.1%
Branch Coverage	43.1%	52.7%
Conditional Coverage	45.4%	57.5%
Redundant Tests	4.86	3.33
Assignment Grade	60.37%	68.27%

*[all differences significant $p < .05$]

Post-test



Dependent Variable	Treatment A (Traditional Detailed Feedback)	Treatment B (Conceptual Feedback)
Line Coverage	37.9%	68.8%
Branch Coverage	38.6%	69.4%
Conditional Coverage	44.8%	72.6%
Redundant Tests	4.29	2.29
Assignment Grade	60.31%	78.95%

*[all differences significant $p < .05$]

Lucas Cordova

cordoval@wou.edu

https://github.com/TestingTutor/Data/tree/master/SIGCSE21

Conclusion