

# A Leveled Examination of Test-Driven Development Acceptance

**David Janzen**

California Polytechnic State University  
San Luis Obispo, CA

**Hossein Saiedian**

University of Kansas  
Lawrence, KS

# Outline

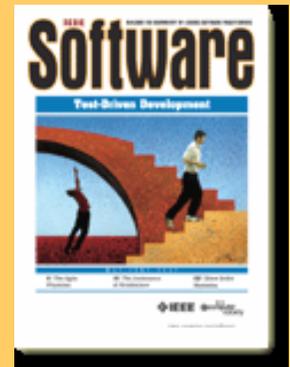
- Study Context
- Survey Questions
- Survey Results
- Confounding Factors
- Conclusions
- Future Work

# Interest in Test-Driven Development

- TDD is a design (and testing) approach involving short, rapid iterations of



- TDD is a key practice in XP
  - Emerging as a stand-alone practice for use with many processes
- May/June 2007 IEEE Software is a special issue on TDD



# TDD Questions

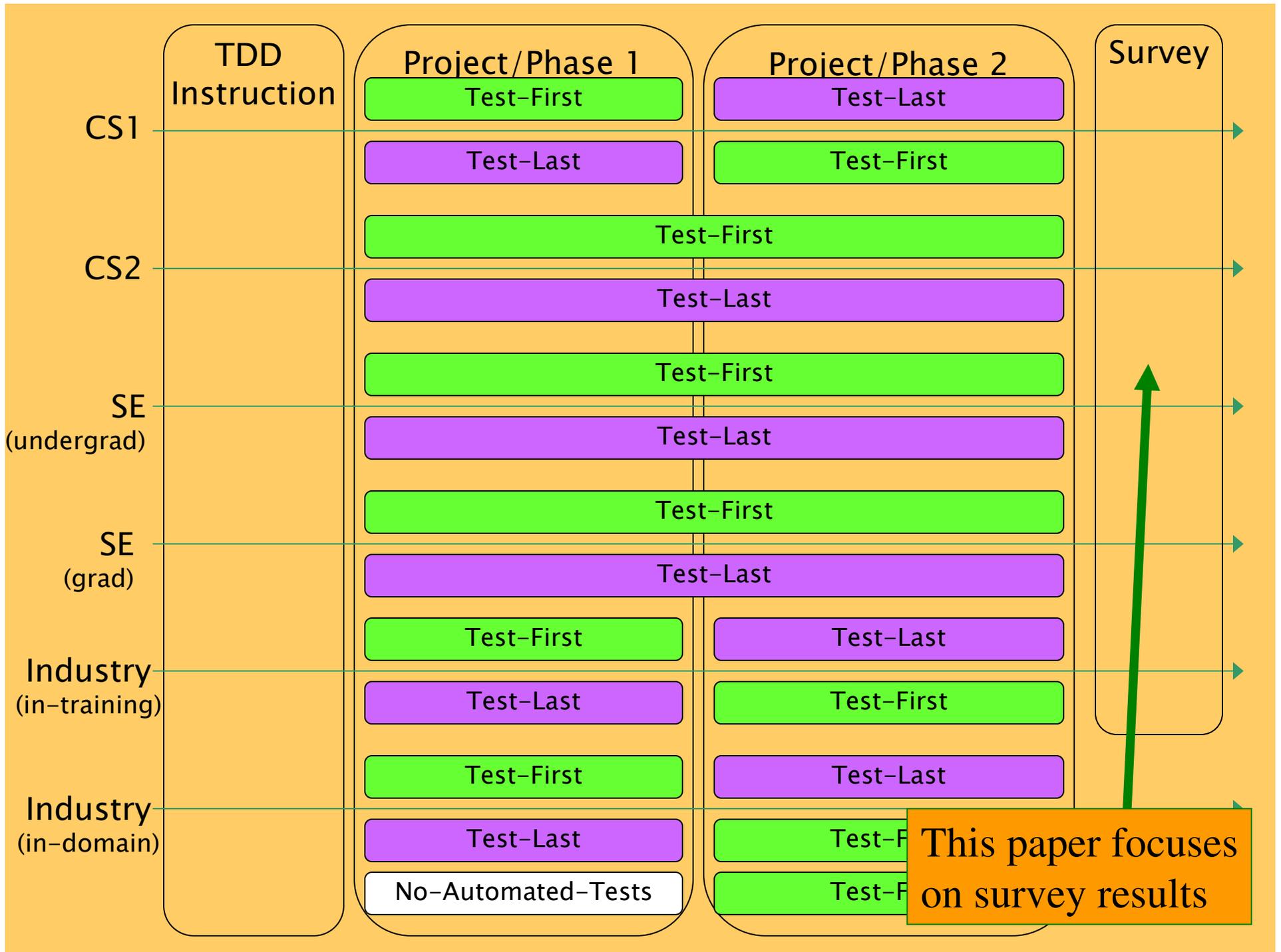
Original research focus

- Should we adopt TDD?
  - Does TDD reduce defects? If so, at what cost?
  - Does TDD improve internal software quality?
- How do we teach TDD?
- When do we introduce TDD?
  - CS1, CS2, SE, MSE, Industry Training
- What do students think about TDD?



# Original Study

- Series of leveled studies to compare test-first (TDD) and test-last approaches
- Hypothesis:
  - TDD improves internal software quality
    - Complexity, size, coupling, cohesion, testability
- Leveled:
  - CS1, CS2, Undergrad SE, Grad SE, Industry



# Study Profiles

Course	# Participants	Language	Collaboration	Project Length
CS1 Fall'05	80	C++	solo	2 weeks
CS2 Fall'05	36	C++	solo	2 weeks
CS2 Spring'06	38	C++	solo	2 weeks
Undergrad SE Summer'05	10	Java	teams of 3 or 4	16 weeks
Grad SE Fall'05	9	Java	teams of 3	16 weeks
Industry '04-'06	12	Java	mostly solo, some pairing	3 hours in training, 4-6 months in study

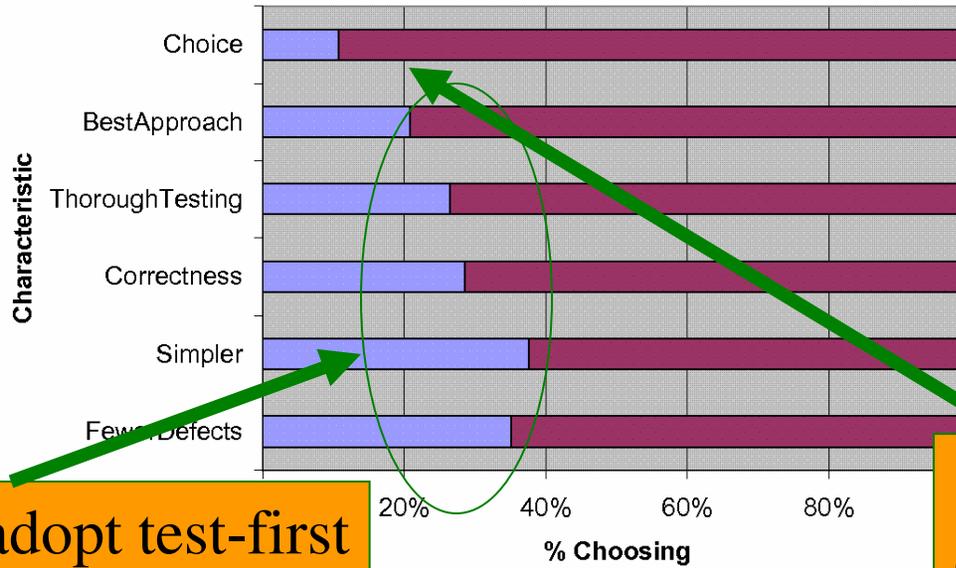
# Post-Study Survey Questions

- **Choice:** Which approach they would choose in the future?
- **BestApproach:** Which approach was the best for the project(s) they completed?
- **ThoroughTesting:** Which approach would cause them to more thoroughly test a program?
- **Correct:** Which approach produces a correct solution in less time?
- **Simpler:** Which approach produces code that is simpler, more reusable, and more maintainable?
- **FewerDefects:** Which approach produces code with fewer defects?

# Survey Results

- Noted differences in survey results between early programmers (CS1/CS2) and more mature programmers (SE courses, Industry)
- What do you expect the differences to be?
  - Early programmers are open to TF/TL equally because they don't have experience with either?
  - Mature programmers are more open to TL because that is what they are familiar with?

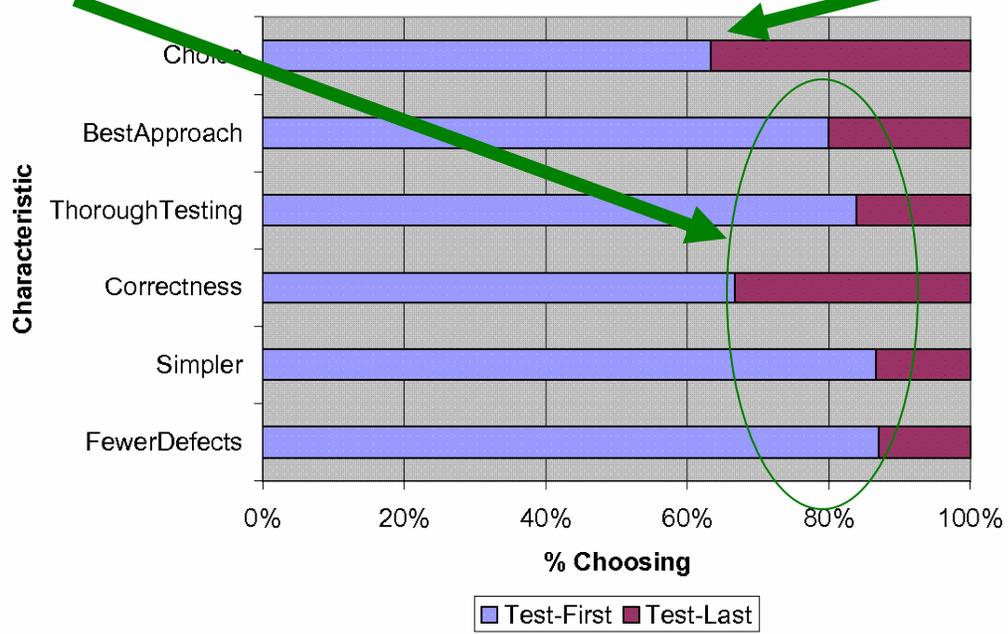
### Beginning Programmer Opinions



Reluctance to adopt test-first despite perceived benefits

11% vs 63% would choose test-first

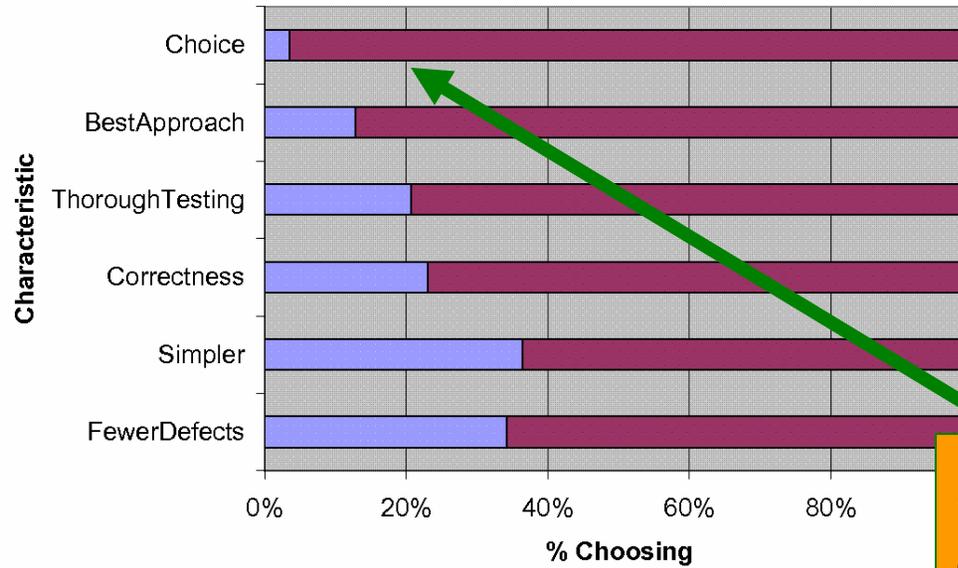
### Experienced Programmer Opinions



# Influence of TDD Experience

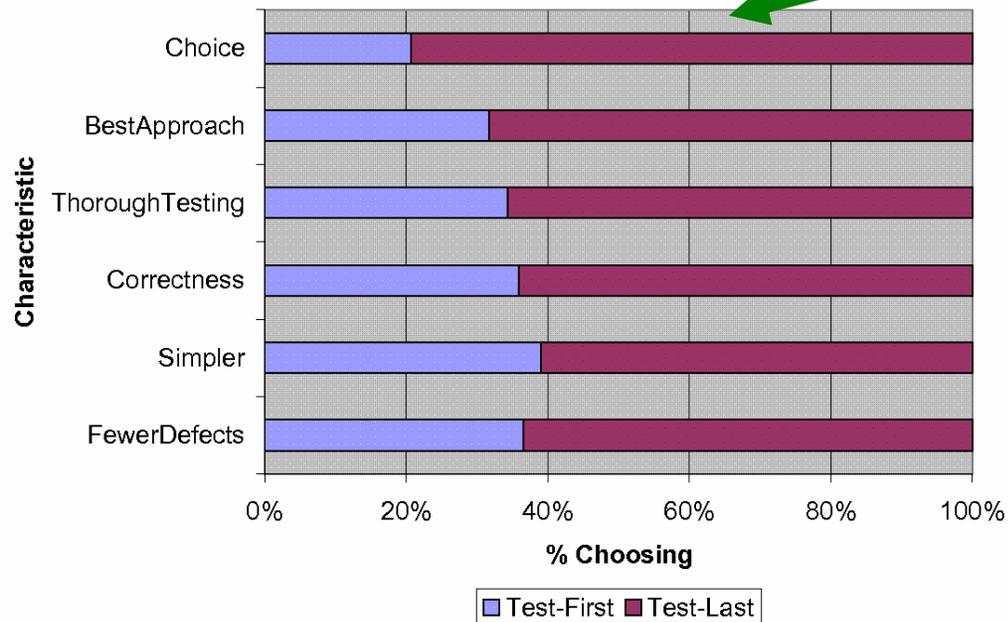
- Did using TDD influence programmer opinions regarding TDD perceptions?

### Beginning Programmer Opinions - TL Only

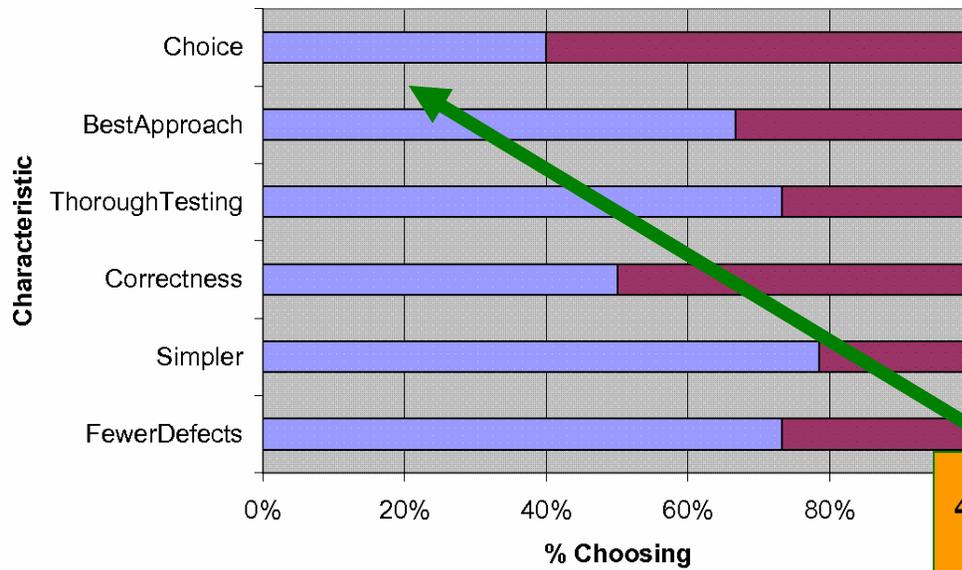


3% vs 21% would choose test-first

### Beginning Programmer Opinions - Tried TF

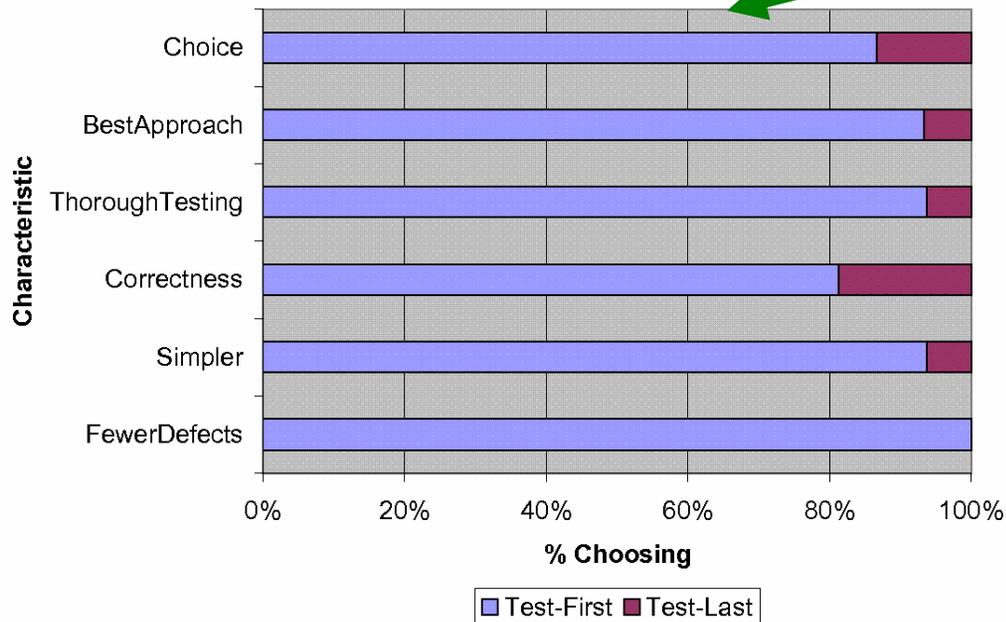


### Mature Programmer Opinions - TL Only



40% vs 87% would choose test-first

### Mature Programmer Opinions - Tried TF



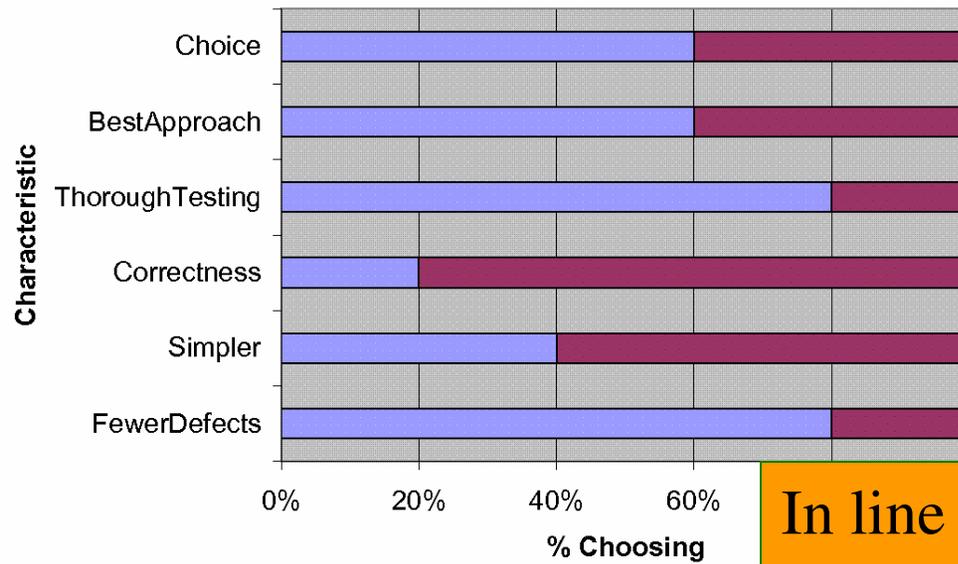
# Confounding Factors

- How much did the following affect programmer perceptions of TDD?
  - Programming language (C++/Java)
  - Automated testing framework (assert vs. JUnit)
  - Solo vs pair/team
  - Short (2 week) vs long (16 week) projects

# Late Breaking News

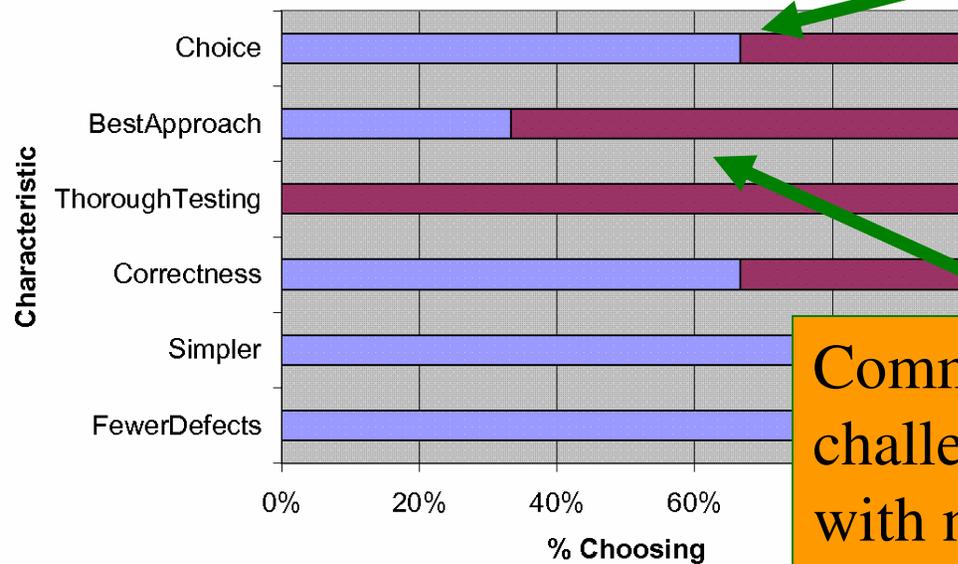
Course	# Students	Language	Collaboration	Project Length
Undergrad SE Capstone Fall'06–Spring'07	12	Java w/ GWT,JBoss, Spring, Hibernate	Teams of 5 or 6 with industry customer	30 weeks

### SE Capstone Only Test-Last



In line with 63% of other mature programmers

### SE Capstone Tried Test-First



Comments indicated challenges using test-first with new technologies

Test-First Test-Last

# Conclusions

- Mature programmers may have higher opinions of TDD and be more likely to adopt TDD than early programmers

# Speculation

- *Ubiquitous* TDD instruction may indoctrinate early programmers into thinking that TDD is the normal way to program

# Test-Driven Learning

- Teach testing (and TDD) by example
  - Introduce and explore new concepts through automated unit tests in a test-first manner
- Inspired by
  - Testing Patterns In Beck’s “TDD: by Example”
    - Explanation Test
      - ask for and provide explanations in terms of tests
    - Learning Test
      - if you want to use a new method, class, or API, first write tests to learn how it works

# Test-Driven Learning<sup>1</sup> in CS1/CS2

## Traditional Approach

```
int sum(int min, int max) {
    int sum = 0;
    for(int i=min;i<=max;i++)
        sum += i;
    return sum;
}
int main() {
    cout << sum(3,7); //should print 25
    cout << sum(-2,2); //should print 0
    cout << sum(-4,-2); //should print -9
}
```

## TDL Approach

```
int sum(int min, int max) {
    int sum = 0;
    for(int i=min;i<=max;i++)
        sum += i;
    return sum;
}
void runTests() {
    assert(sum(3,7)==25);
    assert(sum(-2,2)==0);
    assert(sum(-4,-2)==-9);
}
int main() {
    runTests();
}
```

1. Janzen and Saiedian, "Test-Driven Learning: Intrinsic Integration of Testing into the CS/SE Curriculum," *Technical Symposium on Computer Science Education (SIGCSE'06)*, 2006

# Future Work

- Conduct same study in early programming course with Java and JUnit
- Apply TDL throughout early programming course
- Compare TDD with pairs and TDD with solo programmers

Questions?