# On the influence of Test-Driven Development on Software Design

Conference on Software Engineering Education and Training 2006

David Janzen (djanzen@ku.edu)

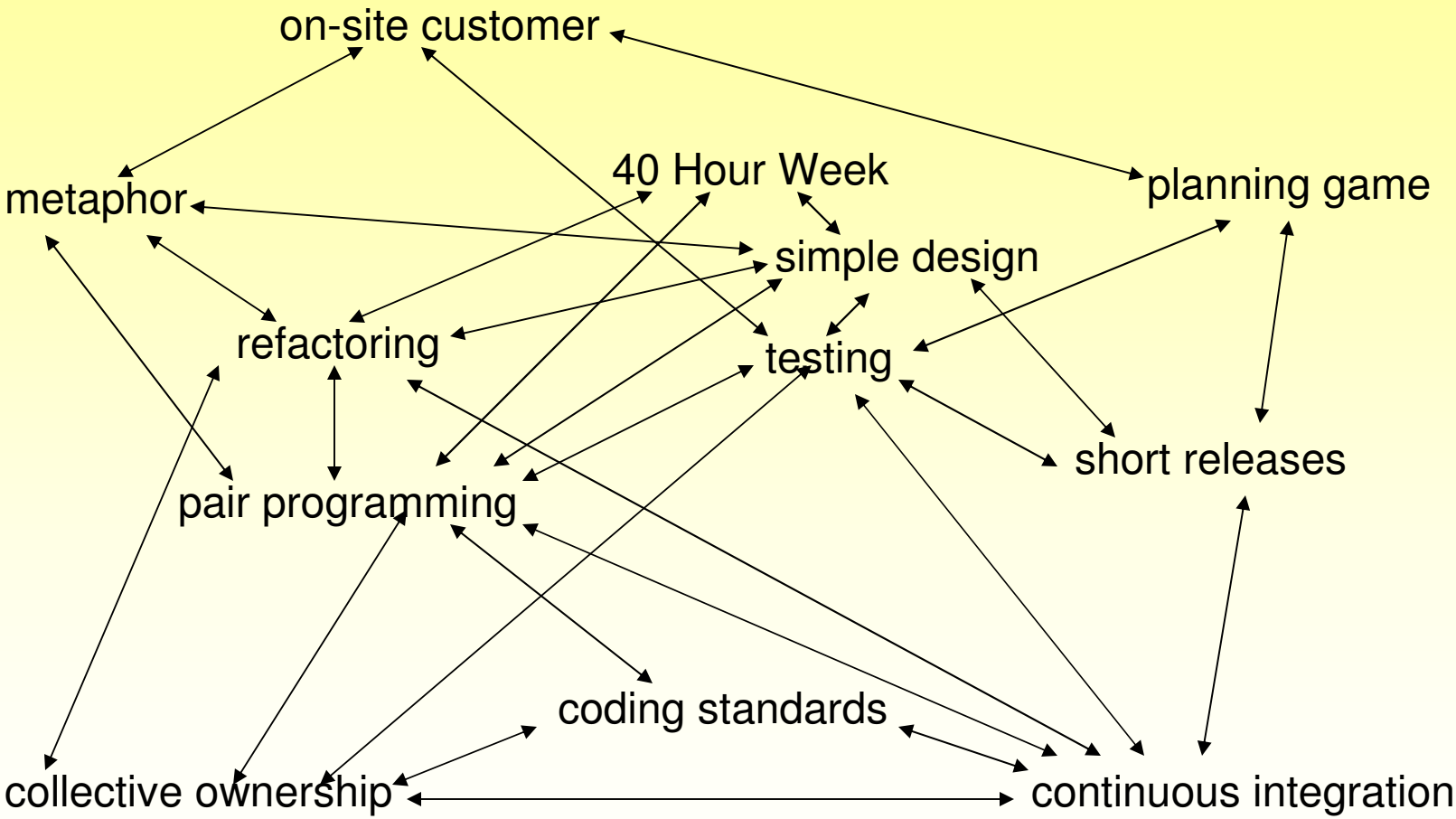
PhD Candidate in Computer Science
University of Kansas

Owner/Principal Consultant and Trainer
Simex LLC

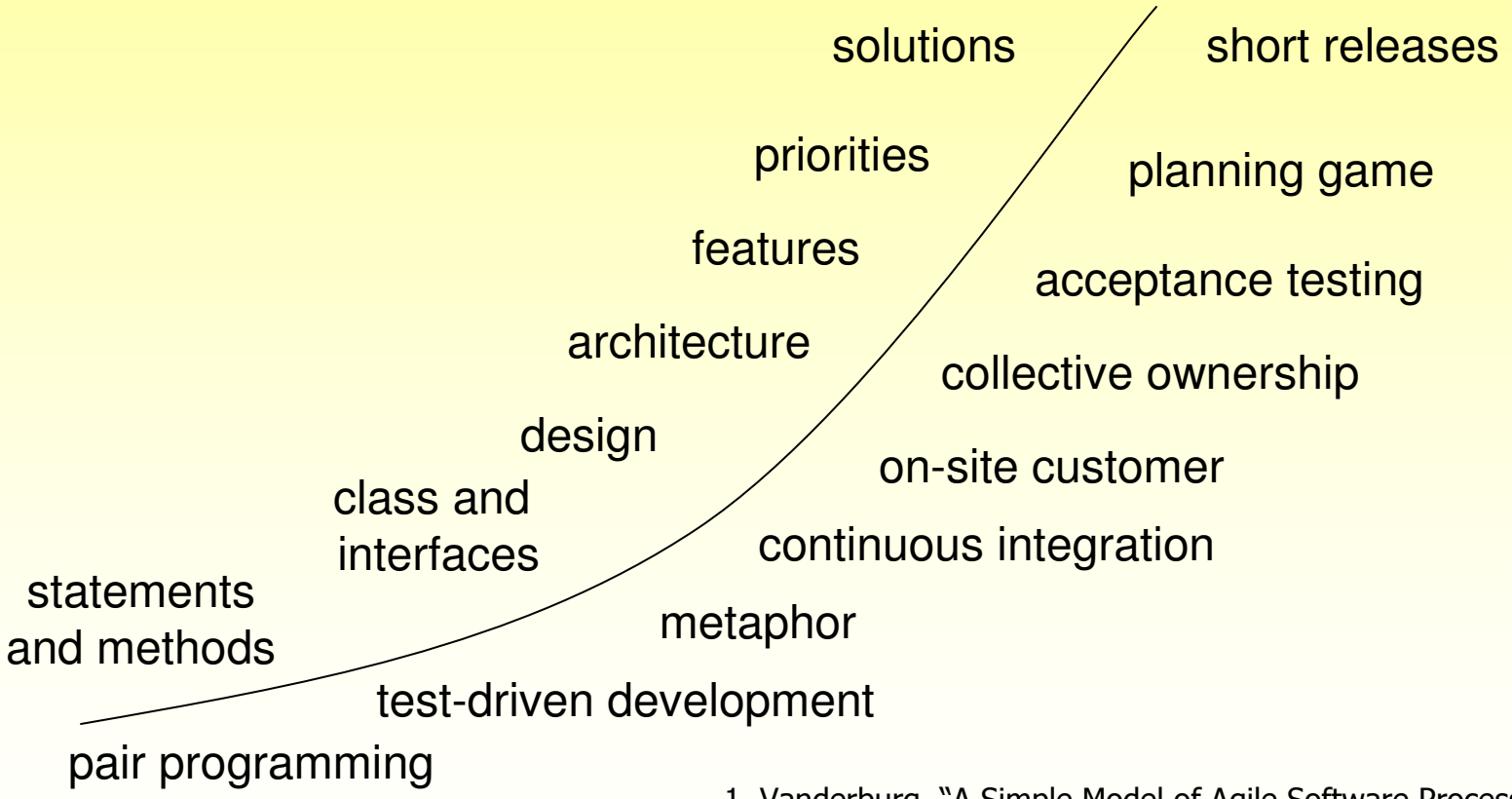

# Schedule

- Focusing on TDD
- Previous Work: External Quality
- TDD and Internal Design Quality
- Empirical Studies
- Results

# XP Practice Coupling

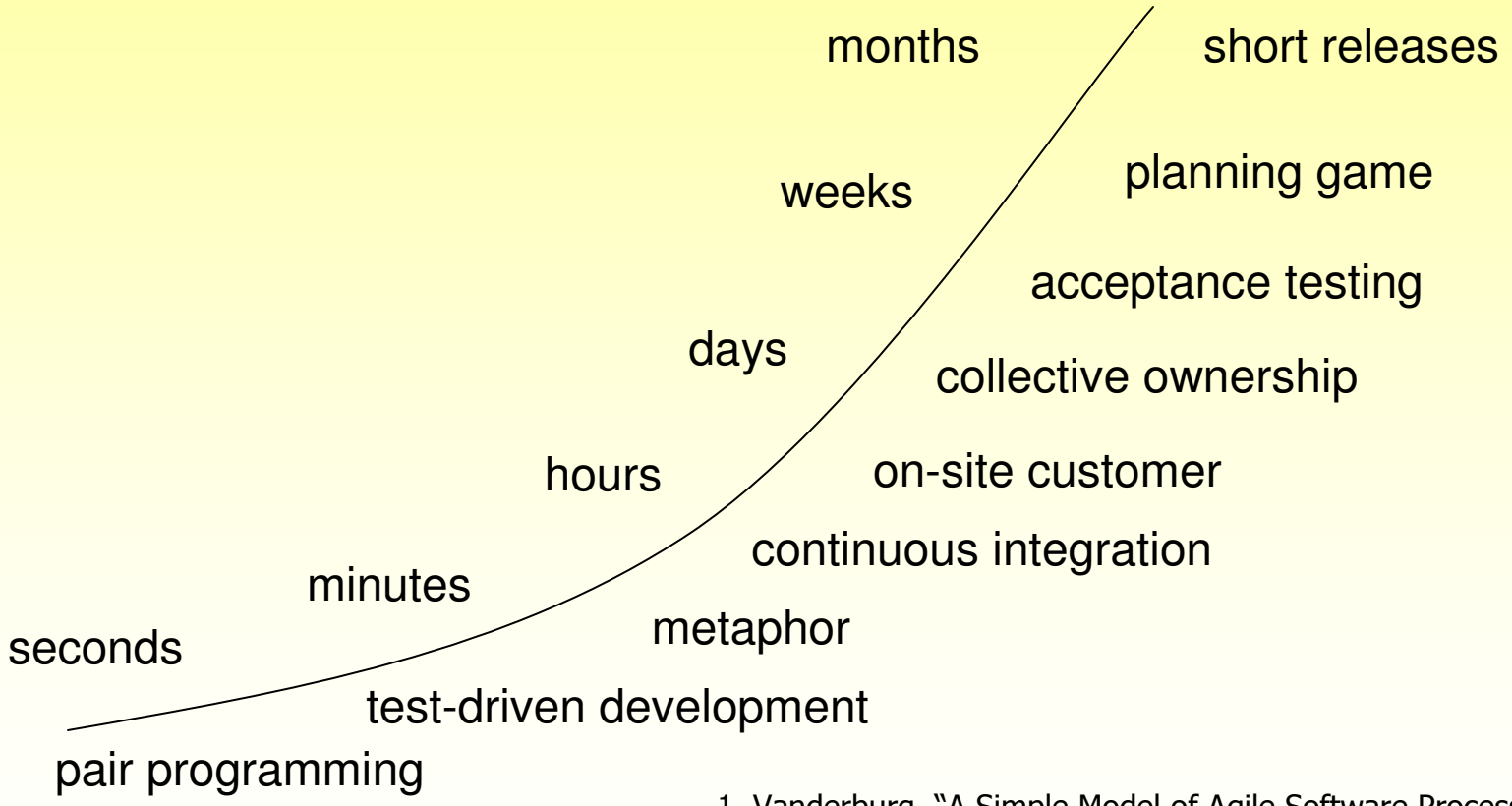


# XP Scale-Defined Practices[1]



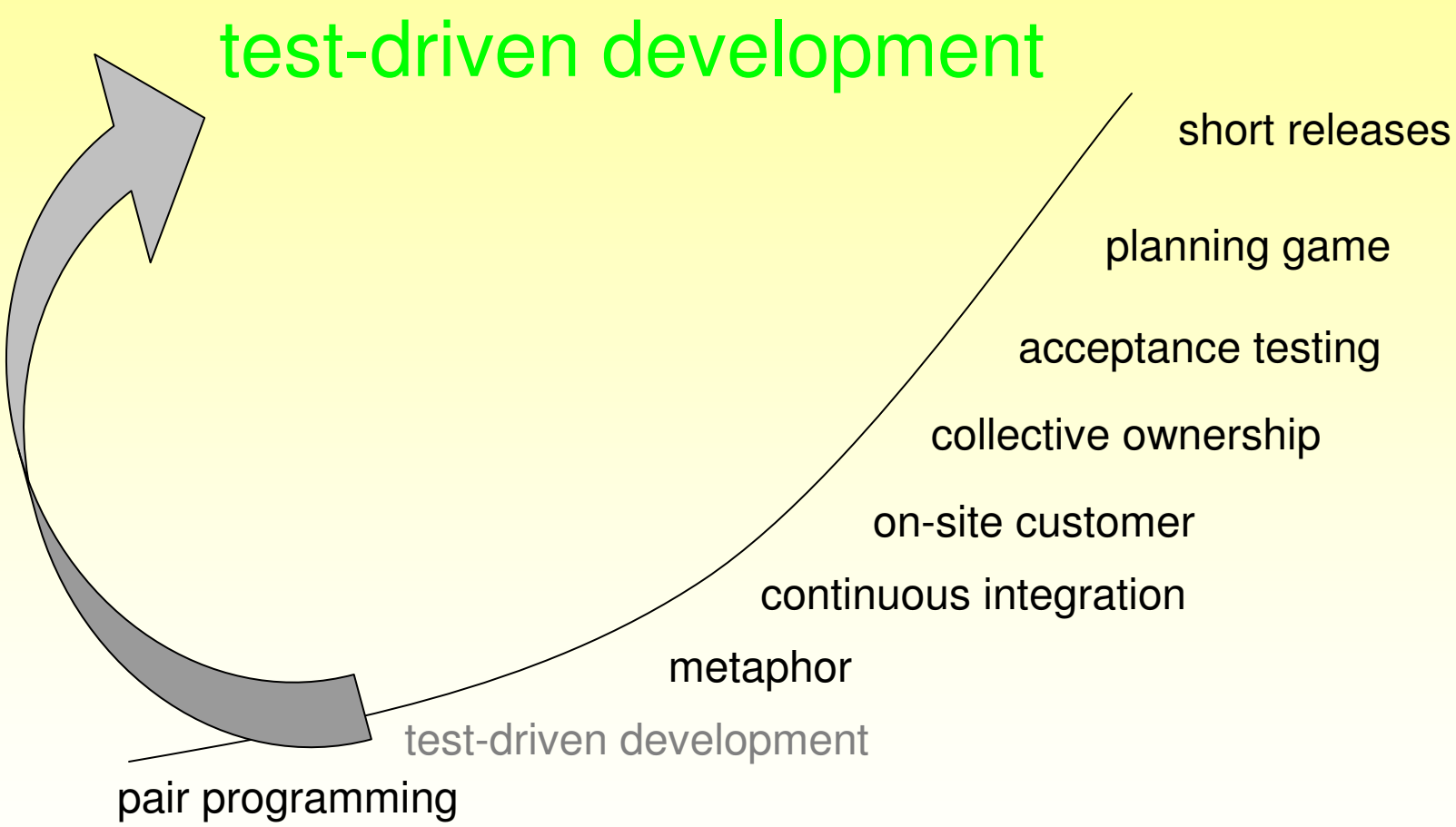

1. Vanderburg, "A Simple Model of Agile Software Processes", OOPSLA'05

# XP Practices and Time Scales[1]




1. Vanderburg, "A Simple Model of Agile Software Processes", OOPSLA'05

# Extracting TDD from XP



# Schedule

- Focusing on TDD
- Previous Work: External Quality
- TDD and Internal Design Quality
- Empirical Studies
- Results

# Related TDD Studies in Industry

| Study[a] | Type | # companies | # programmers | Quality effects | Productivity effects |
|---|---|---|---|---|---|
| George[1] (NCSU 2004) | CE | 3 | 24 | TDD passed 18% more tests | TDD took 16% longer[b] |
| Maximillien[2] (NCSU 2003) | CS | 1 | 9 | 50% reduction in defect density | Minimal impact |
| Williams[3] (NCSU 2003) | CS | 1 | 9 | 40% reduction in defect density | No change |

[a] Studies reported less time spent debugging with TDD

[b] TDD group wrote many more tests than control group


1. George and Williams, "A Structured Experiment of Test-Driven Development", Info & Sw Tech, 2004
2. Maximilien and Williams, "Assessing Test-Driven Development at IBM", ICSE, 2003
3. Williams et. al., "Test-driven development as a defect-reduction practice", Sw Rel. Eng, 2003

# Related TDD Studies in Academia

| Study | Type | # programmers | Quality effects | Productivity effects |
|---|---|---|---|---|
| Edwards[1] (Virginia Tech 2003) | CE | 59 | 54% fewer defects | n/a |
| Kaufmann[2] (Bethel 2003) | CE | 8 | improved information flow | 50% improvement |
| Müller[3] (Karlsruhe 2002) | CE | 19 | no change, but better reuse | no change |
| Pančur[4] (Ljubljana 2003) | CE | 38 | no change | no change |
| Erdogmus[5] (Torino 2005) | CE | 35 | no change | 28% improvement |


1. Edwards, "Rethinking Computer Science Education from a Test-first Perspective", OOPSLA, 2003
2. Kaufmann and Janzen, "Implications of test-driven development: a pilot study", OOPSLA, 2003
3. Muller and Hagner, "Experiment About Test-First Programming", IEEE Software, 2002
4. Pancur et. al., "Towards Empirical Evaluation of Test-Driven Development in a University Environment" Eurocon, 2003
5. Erdogmus, "On the Effectiveness of Test-first Approach to Programming", IEEE Trans on SE, 2005

# Gaps in TDD studies

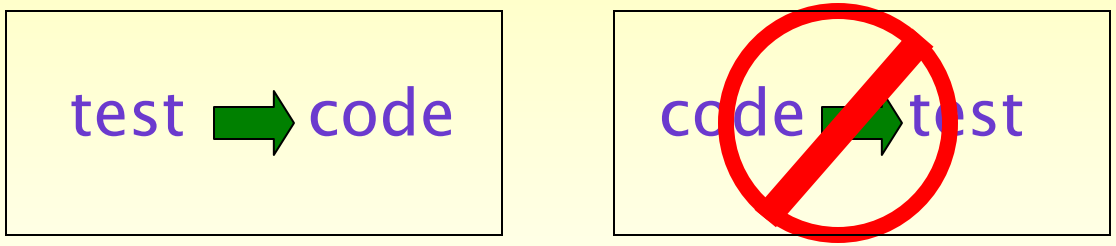
- Defect density was only quality consideration
  - No consideration of design quality
  - No quantification of reuse potential
- Inconclusive results
  - Few, small studies
  - Inconsistent results from inconsistent studies
    - Iterative test-first vs. iterative test-last
    - Iterative test-first vs. traditional test-last (non-emergent design)
- Ignores Pedagogy
  - No consideration of how or where to teach TDD
  - No examination of incidental benefits

# Schedule

- Focusing on TDD
- Previous Work: External Quality
- TDD and Internal Design Quality
- Empirical Studies
- Results

# Test-Driven Development (TDD)

- Disciplined development approach
- Emerged from agile methods (XP)
- Reverses traditional micro workflow

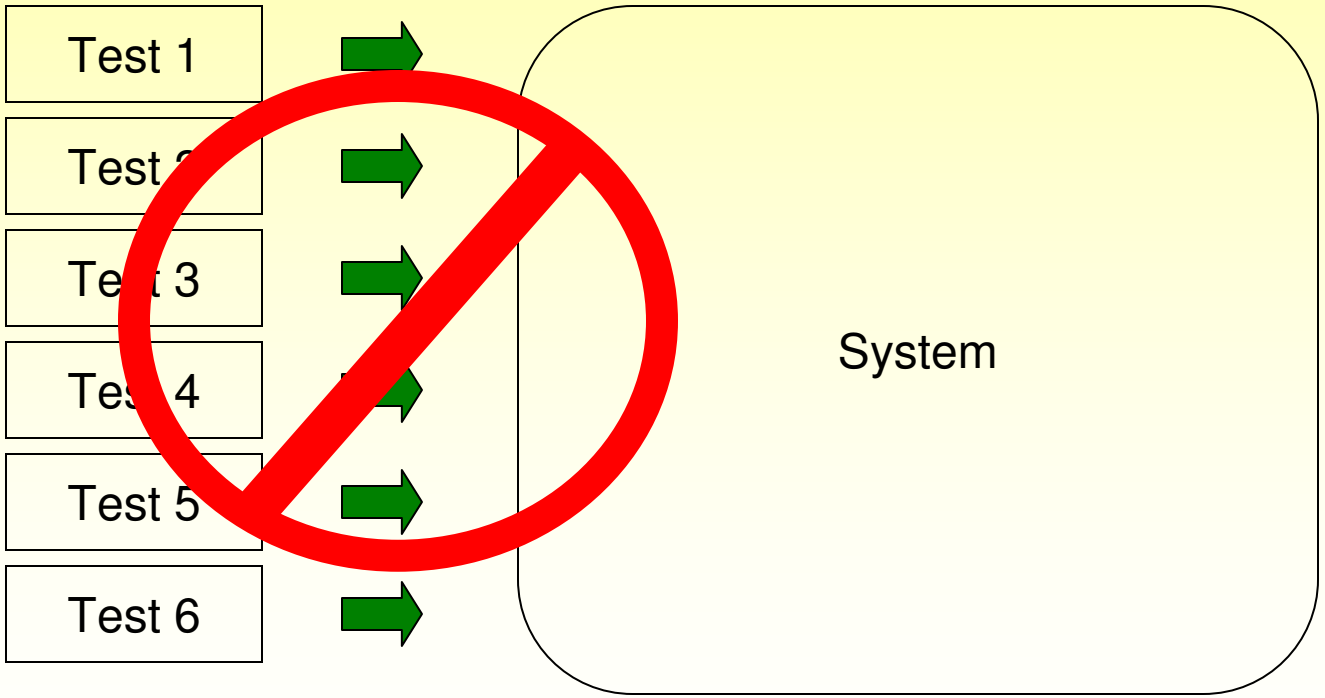



- More about design than testing[1]
- Supported by automated testing frameworks such as JUnit

1. Beck, "Aim, Fire", IEEE Software 2001

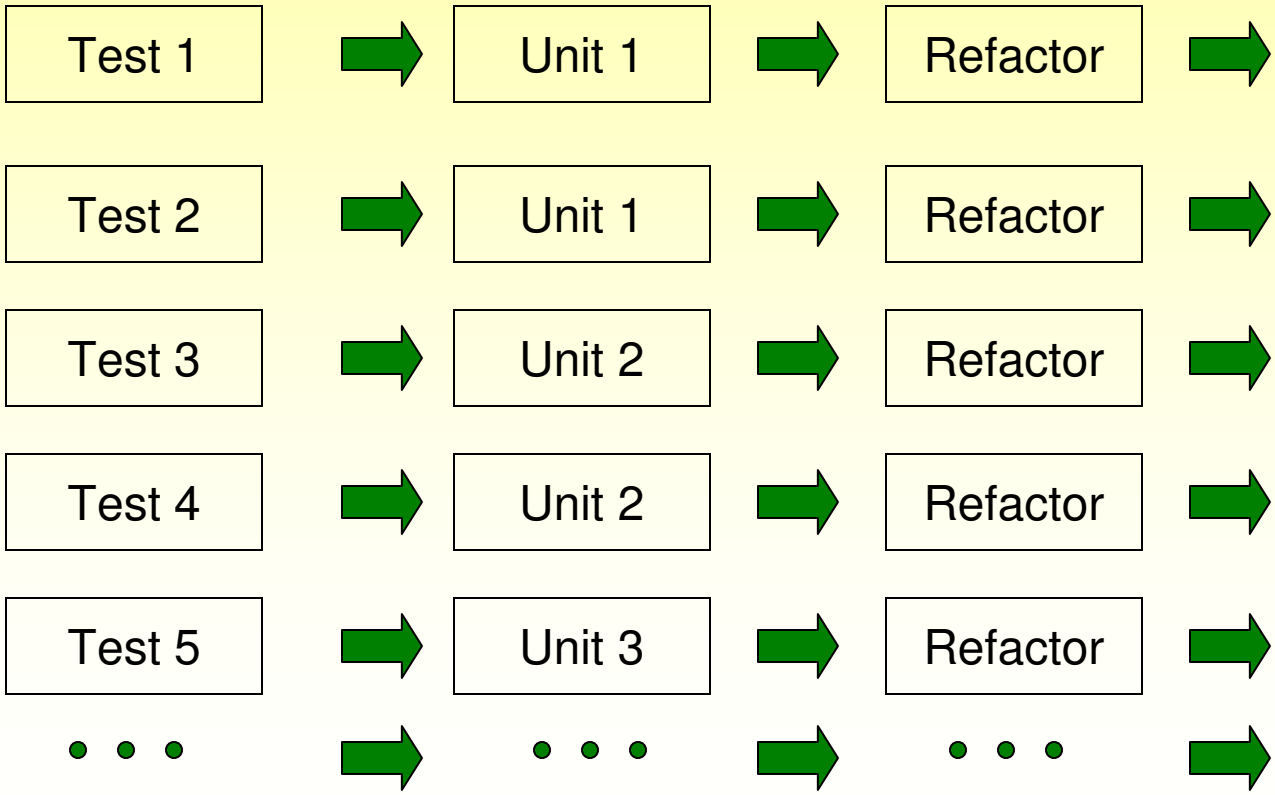
# TDD Misconception

- TDD does not mean “write all the tests, then build a system that passes the tests”



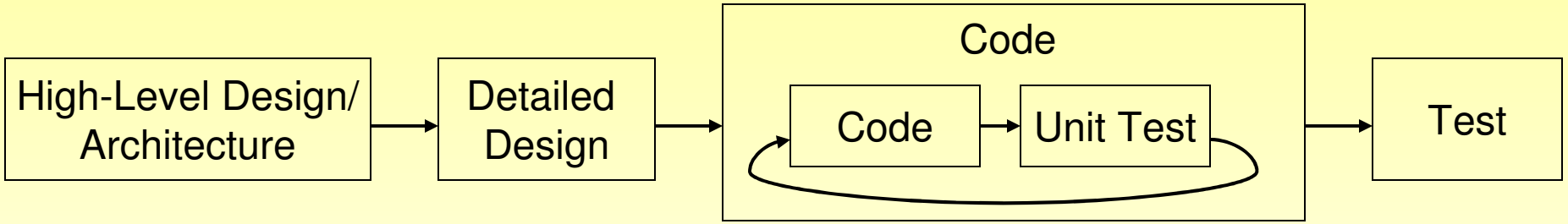
# TDD Clarified

- TDD means “write one test, write code to pass that test, refactor, and repeat”

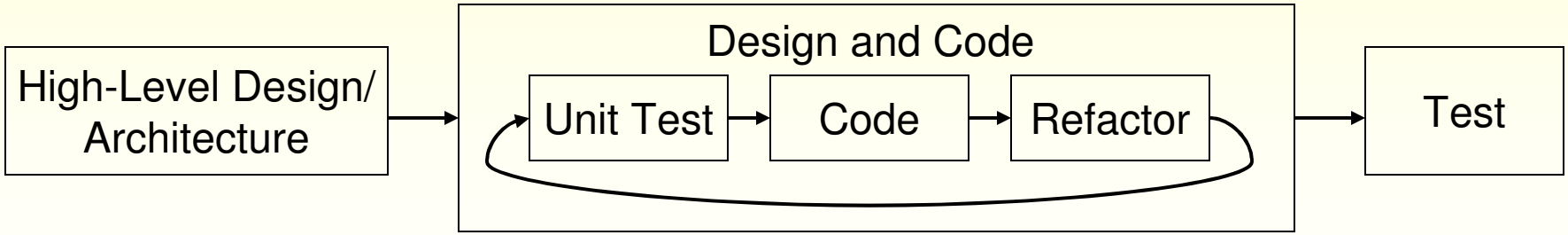


# TDD is about Design

- Traditional test-last process

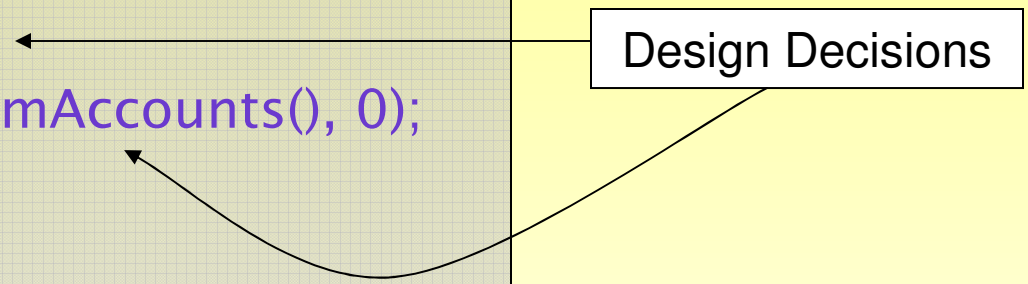



- TDD process



# TDD is about Design

```
public class TestBank extends TestCase {
    public void testCreateBankEmpty() {
      Bank b = new Bank();
      assertEquals(b.getNumAccounts(), 0);
    }
}
```

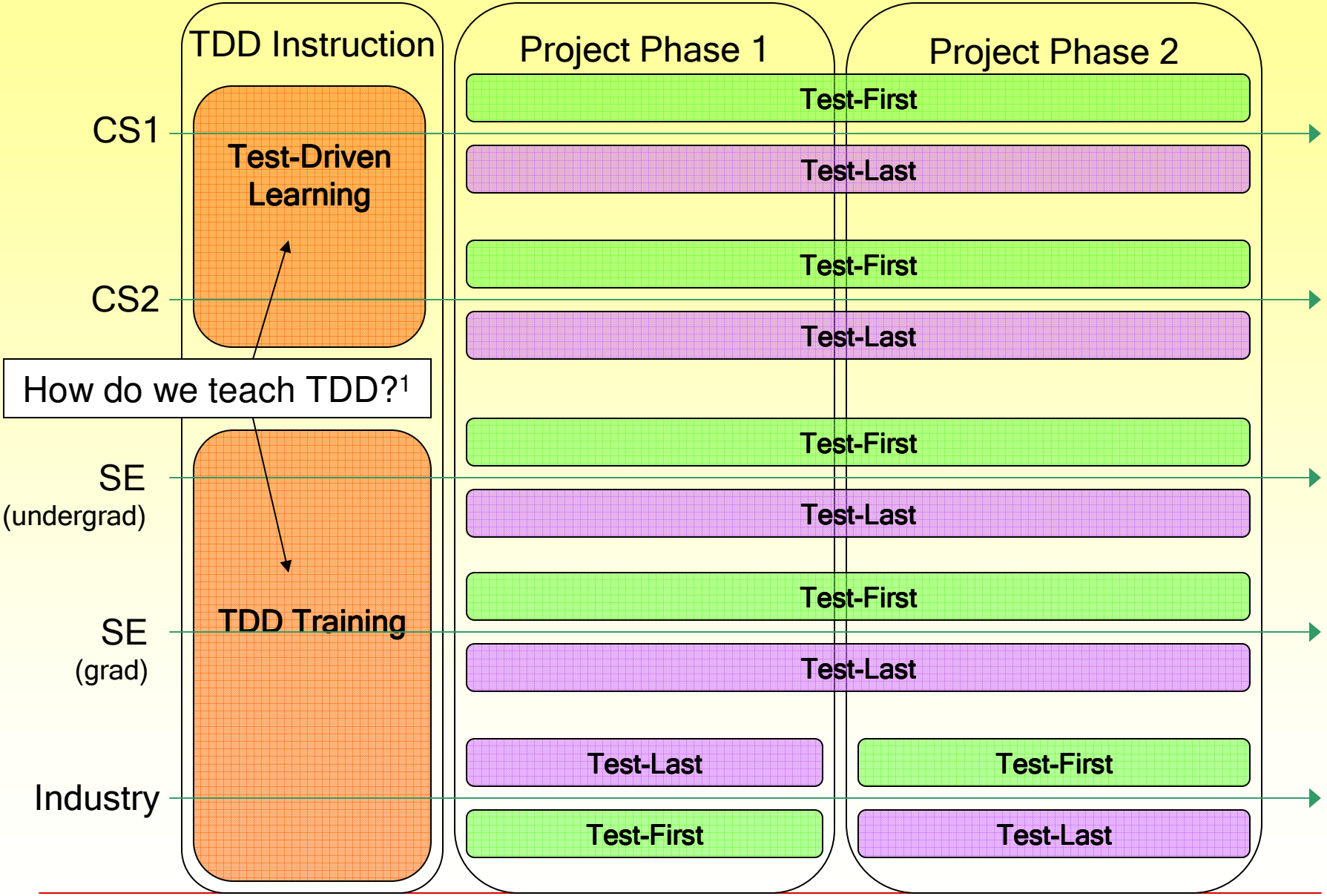



- TDD causes the developer to give early focus to a unit's:
  - Interface: How will I use it?
  - Behavior: What does it do?
  - Reuse: Multiple clients (test and source)
  - Coupling: Units need to be tested in isolation
  - Cohesion: Testable units have one purpose

# Schedule

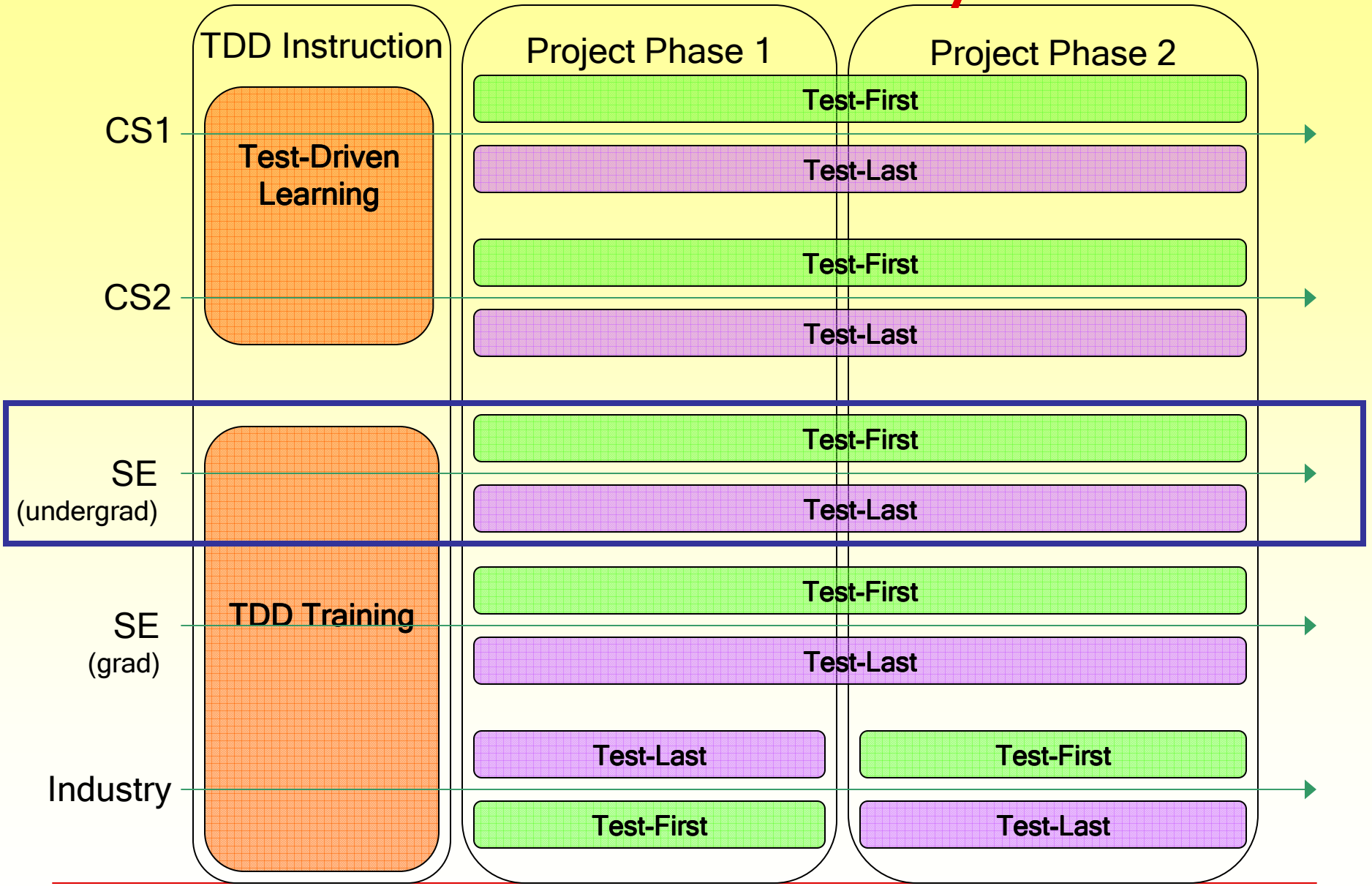
- Focusing on TDD
- Previous Work: External Quality
- TDD and Internal Design Quality
- Empirical Studies
- Results

# Where does TDD fit in Curriculum?




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

# The First Controlled Study



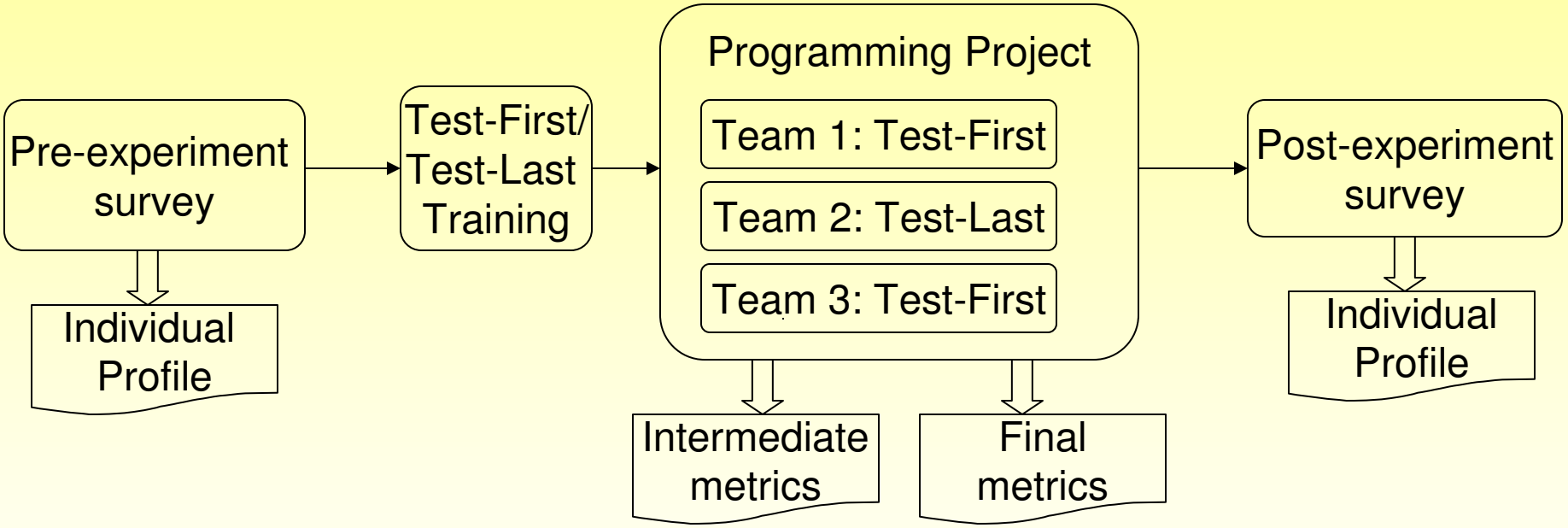
# Formalized Hypotheses: Productivity and Internal Quality

| Name | Null Hypothesis | Alternative Hypothesis |
|---|---|---|
| P1 | $Prod_{TF} = Prod_{TL}$ | $Prod_{TF} > Prod_{TL}$<br>Test-First Programmers are more productive |
| Q1 | $IntQlty_{TF} = IntQlty_{TL}$ | $IntQlty_{TF} > IntQlty_{TL}$<br>Test-First code has higher internal quality |
| Q2 | $IntQlty\|Tested_{TF} = IntQlty\|Not\text{-}Tested_{TF}$ | $IntQlty\|Tested_{TF} > IntQlty\|Not\text{-}Tested_{TF}$ |

# Formalized Hypotheses: Testing and Opinions

| Name | Null Hypothesis | Alternative Hypothesis |
|---|---|---|
| T1 | $\#Tests_{TF} = \#Tests_{TL}$ | $\#Tests_{TF} > \#Tests_{TL}$<br>Test-First Programmers write more tests |
| T2 | $TestCov_{TF} = TestCov_{TL}$ | $TestCov_{TF} > TestCov_{TL}$<br>Test-First Programmers write tests with better code coverage |
| O1 | $Op_{TF} = Op_{TL}$ | $Op_{TF} > Op_{TL}$<br>Programmers perceive Test-First as better approach |
| O2 | $Op\|TF_{TF} = Op\|TF_{TL}$ | $Op\|TF_{TF} > Op\|TF_{TL}$<br>Programmers who have attempted Test-First prefer Test-First |

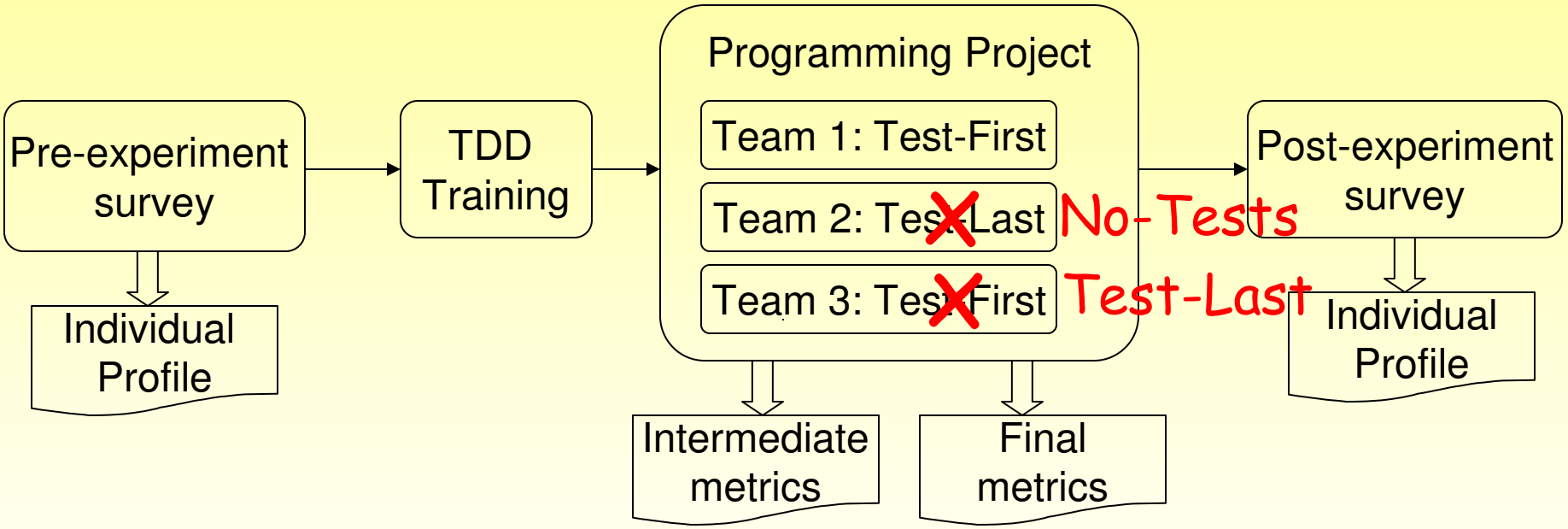
# Experiment Design



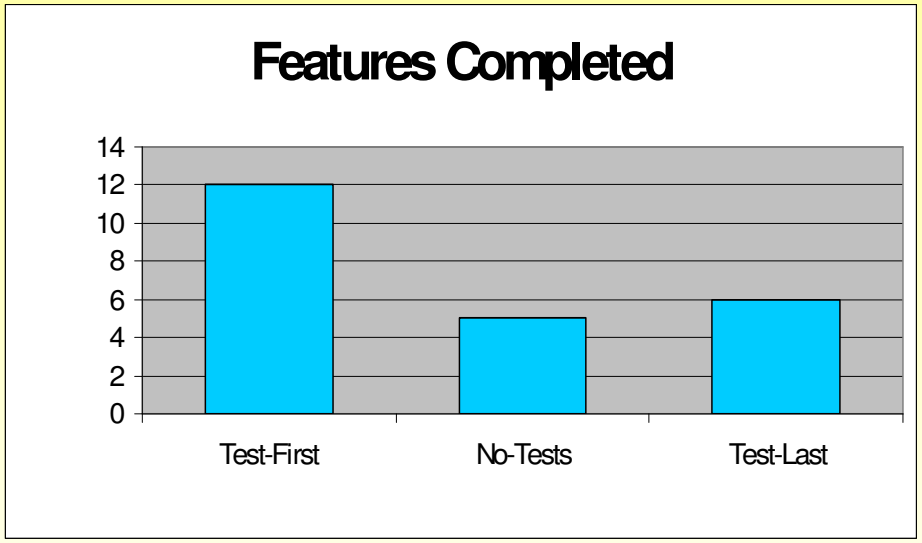
# Schedule

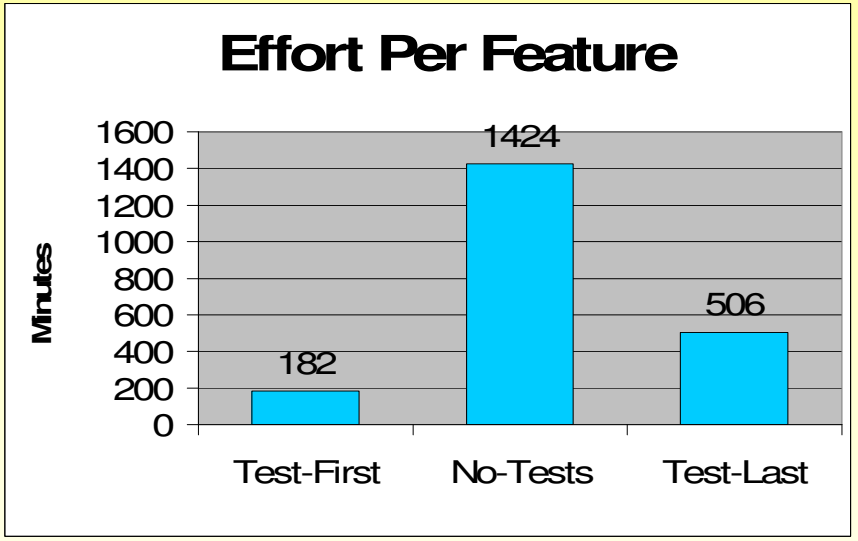
- Focusing on TDD
- Previous Work: External Quality
- TDD and Internal Design Quality
- Empirical Studies
- Results

# Experiment Design Reality



# Productivity Results




- Test-First spent 88% less effort/feature than No-Tests
- Test-First spent 57% less effort/feature than Test-Last
- Only Test-First completed both phases

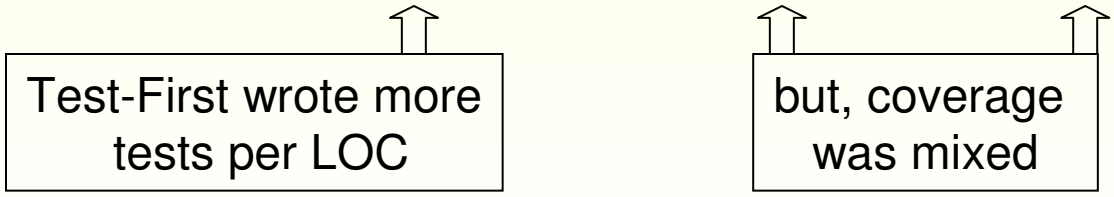
# Code Size and Test Density

- Code size (Source only)

| | # of classes | LOC | #methods | methods/class | LOC/class | LOC/method | LOC/feature |
|---|---|---|---|---|---|---|---|
| Test-First | 13 | 1053 | 87 | 6.69 | 81.00 | 12.10 | 87.75 |
| No-Tests | 7 | 995 | 36 | 5.14 | 142.14 | 27.64 | 199.00 |
| Test-Last | 4 | 259 | 35 | 8.75 | 64.75 | 7.40 | 43.17 |

- Code size (Test only) and Test Coverage

| | Test LOC | % Classes Tested | Assertions/SLOC | Test Coverage (lines) | Test Coverage (branches) |
|---|---|---|---|---|---|
| Test-First | 168 | 38.46% | 0.077 | 19.00% | 39.00% |
| No-Tests | 0 | 0.00% | 0.000 | 0.00% | 0.00% |
| Test-Last | 38 | 25.00% | 0.045 | 29.00% | 23.00% |



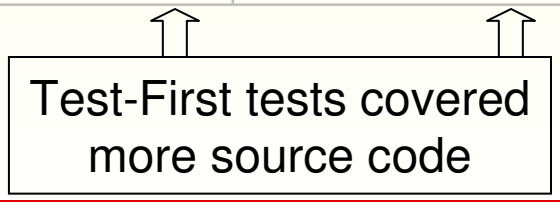
# Code Size and Test Density (No GUI)

- Test-first project included an extensive GUI
- GUI's are traditionally difficult to test
- Code size (Source only without GUI)

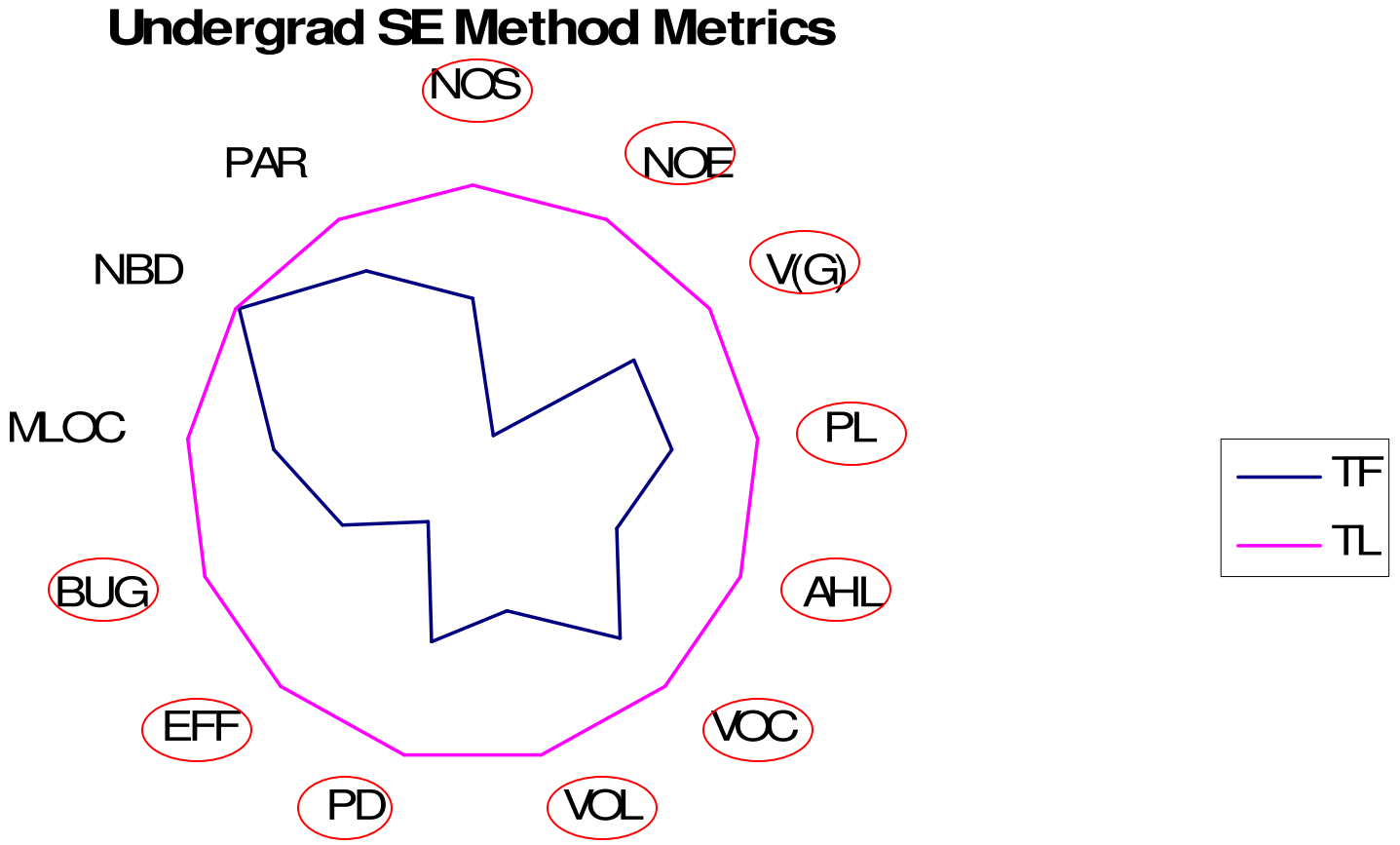
| | # of classes | LOC | #methods | methods/class | LOC/class | LOC/method | LOC/feature |
|---|---|---|---|---|---|---|---|
| Test-First | 11 | 670 | 57 | 5.18 | 60.91 | 11.75 | 55.83 |
| No-Tests | 7 | 995 | 36 | 5.14 | 142.14 | 27.64 | 199.00 |
| Test-Last | 4 | 259 | 35 | 8.75 | 64.75 | 7.40 | 43.17 |

- Code size (Test only) and Test Coverage

| | Test LOC | % Classes Tested | Assertions/SLOC | Test Coverage (lines) | Test Coverage (branches) |
|---|---|---|---|---|---|
| Test-First | 168 | 38.46% | 0.086 | 31.00% | 43.00% |
| No-Tests | 0 | 0.00% | 0.000 | 0.00% | 0.00% |
| Test-Last | 38 | 25.00% | 0.045 | 29.00% | 23.00% |

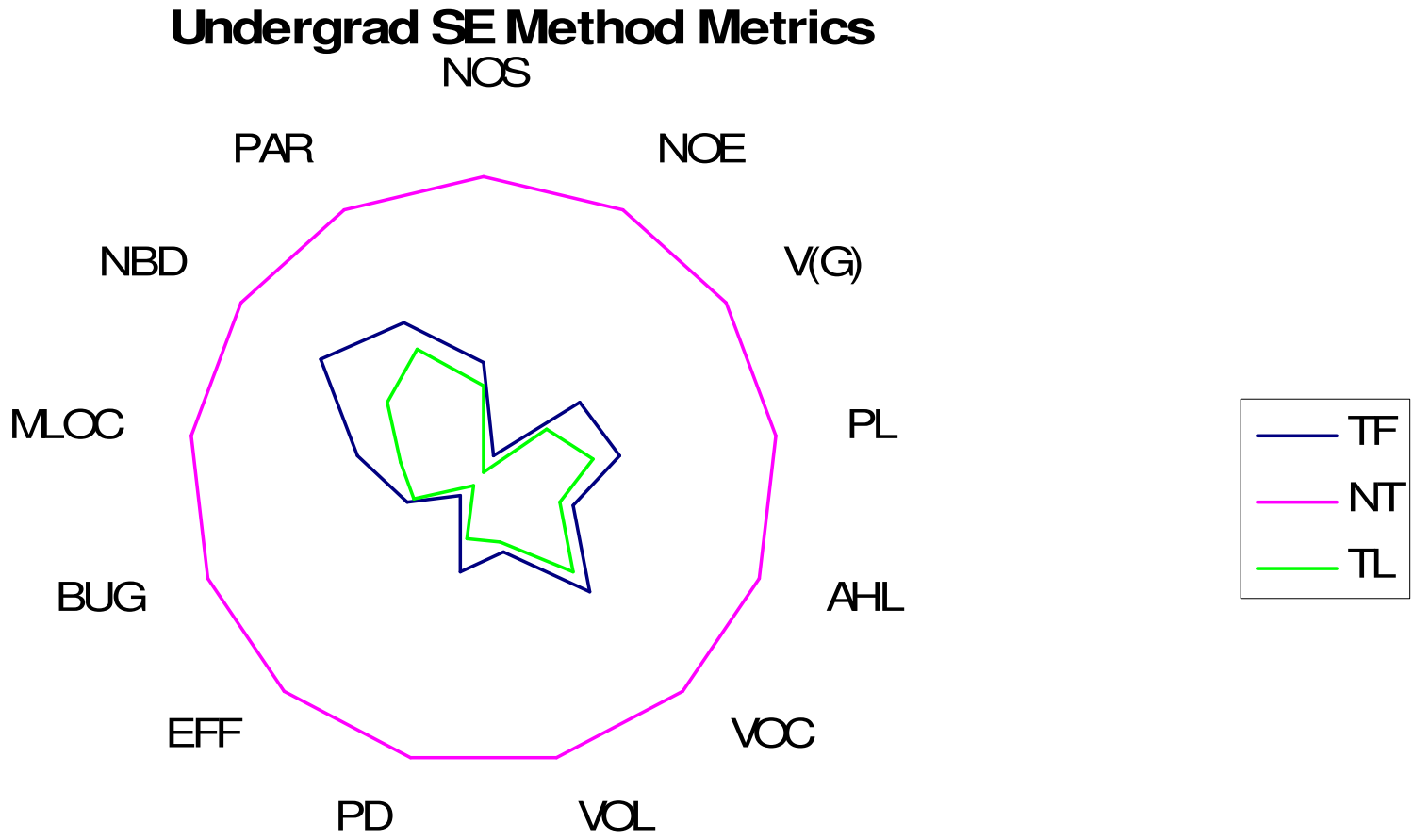


# Design Quality: Method-level Metrics



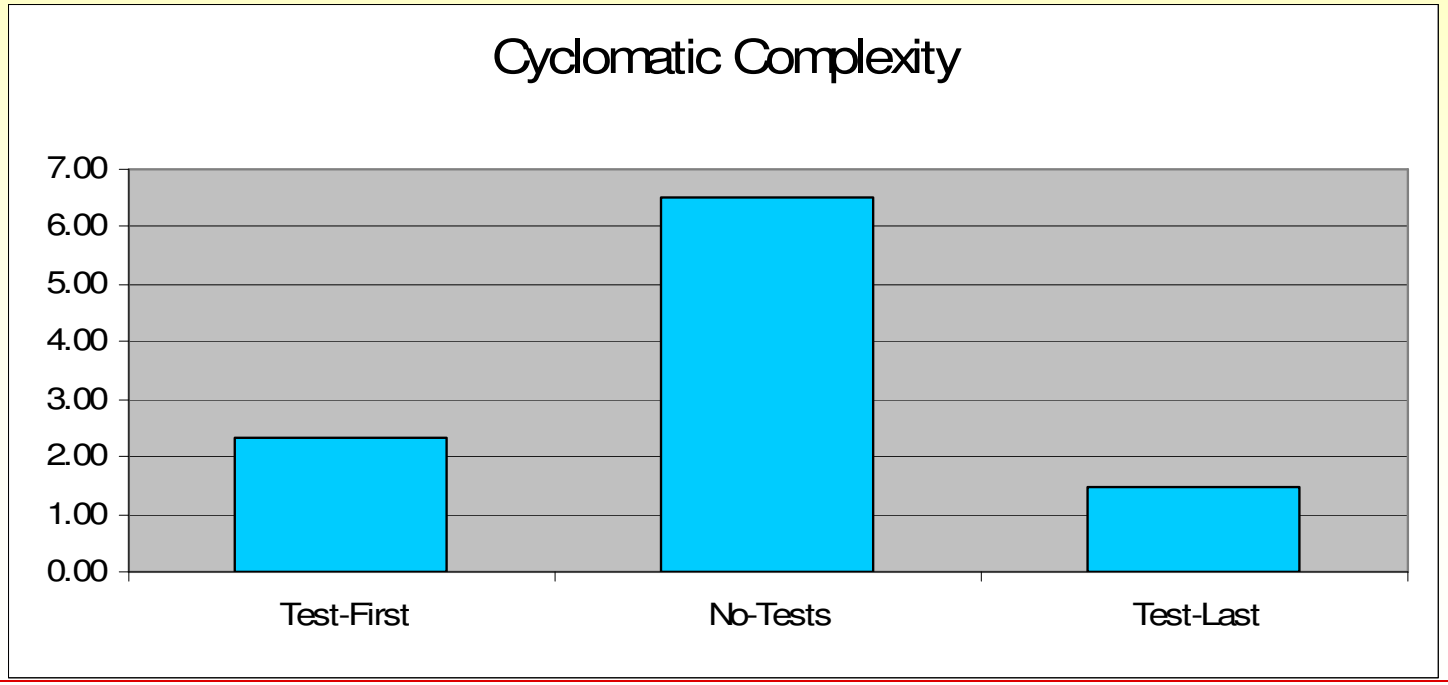

indicates statistically significant difference with $p<.05$

# Design Quality: Method-level Metrics

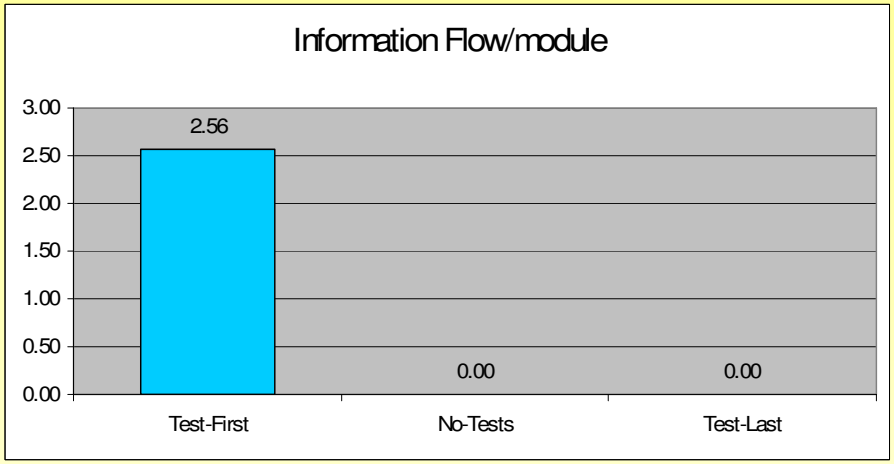


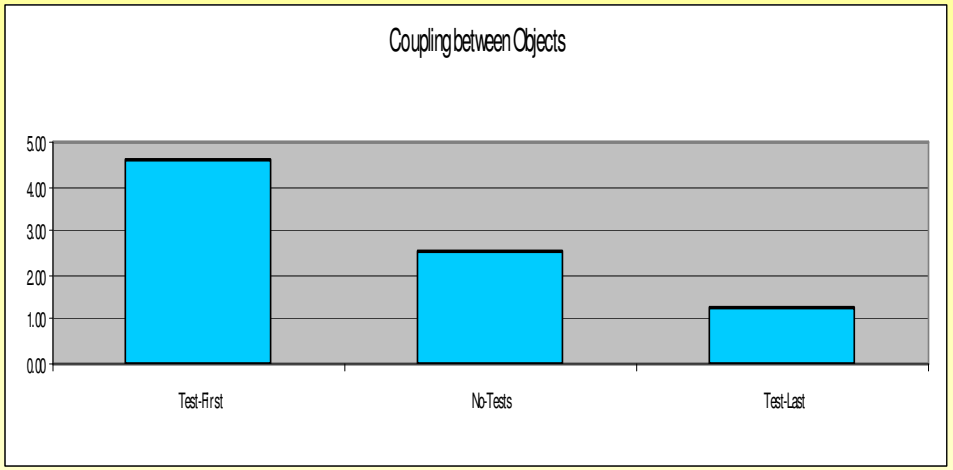
# Design Quality: Class-level Metrics

- Comparable/acceptable levels for most metrics: DIT, NOC, LCOM, …
- NII only metric with statistically significant diff
- Tested code was simpler



# Design Quality: Class-level Metrics



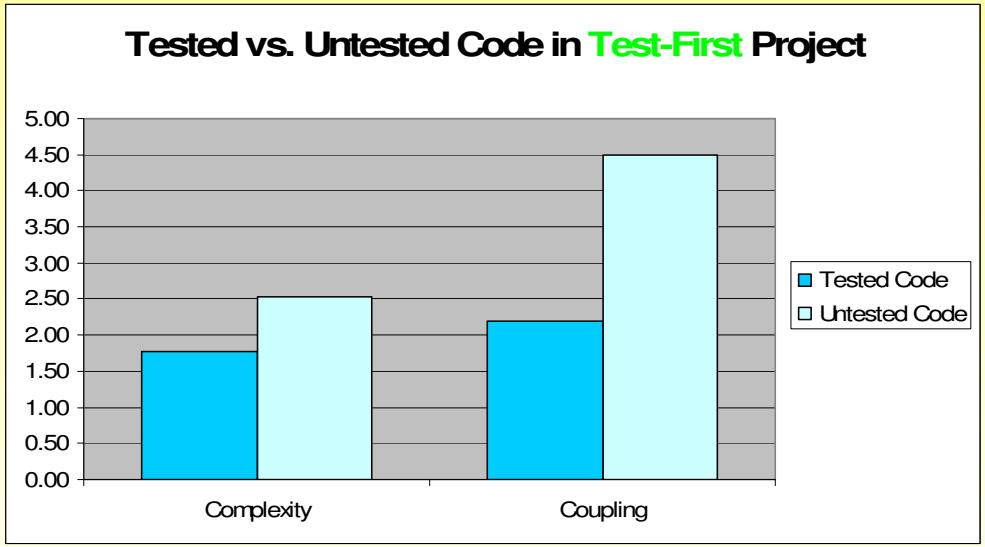




0 Information Flow indicates procedural/flat design in No-Tests and Test-Last teams

Higher coupling in Test-First

# Test-First Team Micro-evaluation

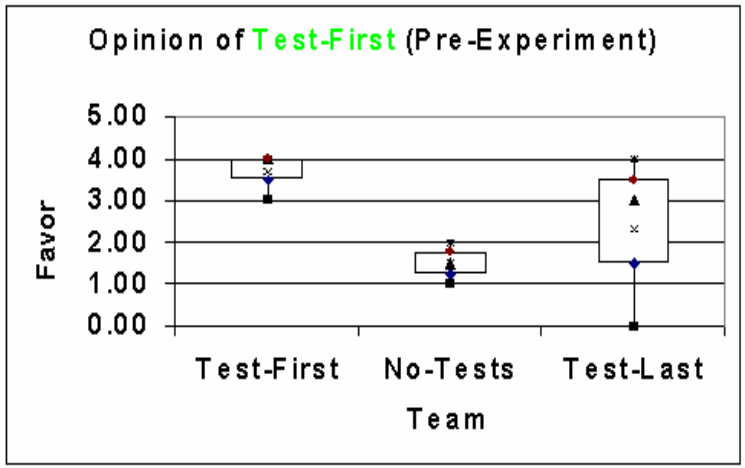
- Evaluated differences in methods tested versus those without tests

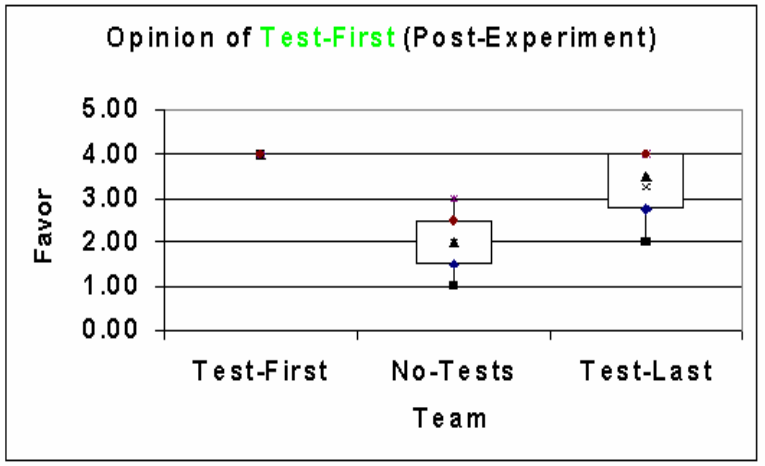



- About 28% of the methods were tested directly
  - These methods had ~43% lower complexity average
  - Not statistically significant at p=.08
- Classes that had some methods tested directly had an average coupling that was ~104% lower

# Student Perceptions[1]

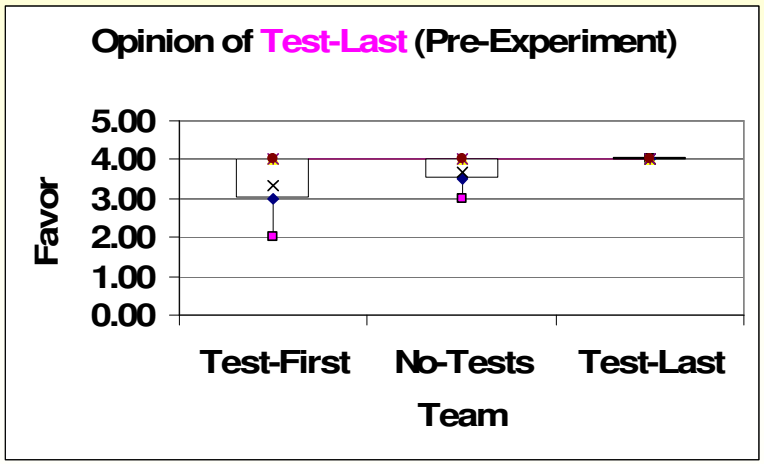
Test-First

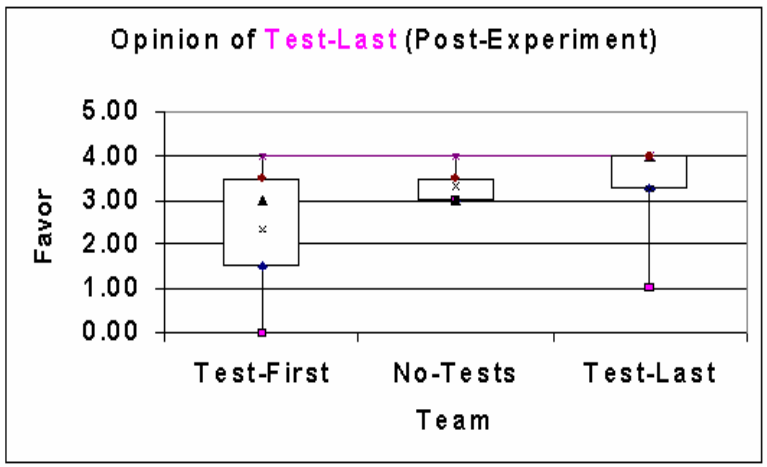



Opinions of TF improved

Test-Last




Opinions of TL declined

1. D. Janzen, "Software Architecture Improvement through Test-Driven Development," *Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'05)*, October, 2005, San Diego, CA

# Results of Undergrad SE study: Programmer Perceptions

- 89% of programmers thought Test-First produced *simpler designs*
- 70% thought Test-First produced code with *fewer defects*
- 75% thought Test-First was the *best approach* for this project

# Summary

- TDD can be used without XP
- Empirical studies can be conducted in undergrad SE courses
- TDD adoption must be motivated
- TDD shows promise of possibly improving productivity and test coverage
- TDD may lower complexity, but may increase coupling
- Results are suspect until we get a larger sample

# References


- D. Janzen and H. Saiedian, "Test-Driven Learning: Intrinsic Integration of Testing into the CS/SE Curriculum," *Technical Symposium on Computer Science Education (SIGCSE'06)*, March, 2006, Houston, TX
- D. Janzen and H. Saiedian, "Test-Driven Development: Concepts, Taxonomy and Future Directions," *IEEE Computer*, 38(9), 2005
- D. Janzen, "Software Architecture Improvement through Test-Driven Development," *Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'05) Student Research Competition*, October, 2005, San Diego, CA


# Acknowledgements


- Karen Janzen, Hossein Saiedian
- SIGCSE Special Projects Grant