Software Architecture Improvement through Test-Driven Development

1. Problem 1: Does TDD improve Software Quality?
- Does TDD produce code with better internal quality (i.e., better designs)?
- Is code more modular, easier to reuse, easier to maintain?
- Does code have fewer defects?
- Does TDD take longer to develop the same functionality?

2. Problem 2: How do we teach TDD?
- Can undergraders learn TDD?
- Where is the optimal level in the curriculum to teach TDD?
- How should we train students in TDD?

3. Related Work: Focus on defects and external quality

4. Solution: Conduct empirical studies that focus on design and internal quality
- Conduct formal experiments in academic and professional settings
- Isolate TDD as an independent variable
- Experiment group develops software with Test-First approach
- Control group develops software with Test-Last approach

5. Initial Results: Experiment in Undergraduate Software Engineering Course

6. Conclusions
- Test First team was more productive
- Test First team produced more Object- Oriented Design
- Test Last team wrote more tests
- Tested code was more complex and more decoupled
- Test First team produced more decoupled code
- Test First perceptions improved with use while Test Last perceptions worsened
- Additional industry and academic studies are warranted

7. You can contribute
- Industry Professionals
  - Contribute project data
  - Email: djanzen@kansas.edu
- Academic Scholars
  - Adopt TDD
  - Duplicate studies

8. Acknowledgements
- Dr. Hassem Sawiyan, advisor