TDD = Too Dumb Developers?
Implications of Test–Driven Development on maintainability and comprehension of software

Marco Torchiano: marco.torchiano@polito.it
Alberto Sillitti: alberto.sillitti@unibz.it

Outline

- Introducing TDD & Agile
  - Just in case...

- Working
Software Development

Building a ship? Growing a garden?

Traditional development

- ..or heavyweight
  - Emphasis on documentation, process
    - Waterfall, prototyping, iterative,
    - Iso 9000, Vision, CMM
  - Still, some projects fail (do they?)
Agile

- Or lightweight
  - Individuals and interaction over process and tools
  - Working software over comprehensive documentation
  - Customer collaboration over contract negotiation
  - Responding to change over following a plan

- Actually, nothing really new, but mix is innovative

Extreme programming

- Kent Beck: Extreme Programming Explained
  Addison–Wesley, 2000
12 practices

- Customer satisfaction
  - On-site customer
  - Small releases

- Software quality
  - Metaphor
  - Simple design
  - Refactoring
  - Pair programming
  - Testing

- Project management
  - Planning game
  - Sustainable development
  - Collective code ownership
  - Continuous integration
  - Coding standards

How everything fits together

From Beck: XP, Page 70
Test–Driven Development


TDD in XS

- TDD in eXtreme Synthesis:
  - Pick a piece of story/requirement/feature
  - Write a test code for it
  - See it fail
  - Write the relative production code
  - Run the test and fix until a green bar
  - Add further tests until enough
  - Possibly refactor
TDD in context

80–20 rule...

- Pareto
  - You know it, do you?
- Beck
  - If you apply even 80% of the XP practices you won’t get more that 20% of the potential benefits
- Focusing on single technique (TDD) makes no sense
  - According to Kent Beck....
According K.B.

- We should stop here and have a walk in the Stanley park….

Empirical findings

- In the last 5–6 years some empirical work has been done
  - In the Agile area
  - About TDD
- No conclusive findings
TDD and Maintainability

- 20 empirical studies
  - From 2001 to 2008
  - Different types
  - Most performed with students

- Meta-analysis
  - Effect on code quality
  - Effect on understandability

![Graph showing the effect magnitude of TDD on comprehension and quality](attachment:chart.png)
Today

- Brainstorming
  - Main problems
  - Research questions
- Empirical design
  - Definition of experiments
- Presentations
  - And commitments
- Homework: execution
Open questions

1. What is the right way?
2. How can you judge conformance?
3. What are the cues of TDD?
4. Is test quality important?
5. Tests are code
6. Do you need agile process?
7. Which are the right developers?
8. How is TDD essential to XP?
9. Overall effort
10. Cost of quality
11. Economic assessment

Empirical design
Experiment

- **GOAL**: evaluate usefulness of TDD test cases as design documentation for the purpose of perfective and corrective maintenance.

- **Type of design**
  - Comparative design:
    - 2 groups (experimental, control)
    - experimental group vs. baseline

Objects

- **Exp objects**
  - real world projects
    - Hard to find 2 comparable systems
  - 2 small toy (1 person) projects
    - Can be developed by researchers following either a TDD or a Traditional approach
  - 2 student groups
    - Developing TDD or traditional
  - 2 classes (n + m groups) screened
    - Same as above but select best pair
Objects

- Threats:
  - professional vs. students developers
  - process conformance
  - what is traditional? w/fall, iterative, RUP
    - define list of artifacts
    - same set of defects?
    - defect seeding (may be incompatible with process)
    - perfective maintenance
    - evolution: adding a new feature

Measures

- dependent variable:
  - # fixed bugs
  - # added features
    - acceptance test suite

- independent variables:
  - group (control, experimental)
  - bug category
Population and design

- Subjects:
  - students <
  - Professionals

- Design:
  - 2 groups 2 treatments 1 object
  - factorial design (2x2x2)

Hypotheses

- Hypotheses:
  - mean(# fixed defects | TDD) !≠ mean(# fixed defects | traditional)
  - mean(# fixed defects | TDD) > mean(# fixed defects | traditional)

- Practical relevance
Presentation

Actually we worked all together so...

Follow up

- Publish documents
  - Research questions
  - Experiment design
- Open comments
- Possible cooperation
  - Benchmarks
  - Detailed design
- Contact us for further work