Why Johnny Can't Test and Can Test-Driven Development Help: A People-Centered Analysis of Testing

Orit Hazzan
Department of Education in Technology and Science
Technion
Email: oritha@techunix.technion.ac.il

- Who likes testing?
- Who hates testing?

- Why?

Outline

- Working assumption: Software is not tested enough
  - Why people don't like testing?
  - How may Test Driven Development (TDD) help?
  - TDD within Extreme Programming

- Thanks: Uri Leron and Yael Dubinsky

Outline

- Why people don't like testing?
  - Cognitive
  - Social
  - Affective
  - Managerial

    Think about TDD…
- How may TDD help?
- TDD within Extreme Programming

Why people don't like testing?

- **Observation 1:** In traditional development environments, testing appears as one of the last stages and is usually done under pressure.

Software Engineering/Hans Van Vliet

- P. 386-397: … the testing activity often does not get the attention it deserves. By the time the software has been written, we are often pressed for time, which does not encourage thorough testing.
- p. 397: Postponing test activities for too long is one of the most sever mistakes often made in software development projects. This postponement makes testing a rather costly affair.

  Yet, this book presents the traditional life cycle of software development.

Why people don't test

- **Observation 1:** In traditional development environments, testing appears as one of the last stages and is usually done under pressure.
  - Time pressure is a naive explanation: it is based on an external/technical factor
  - The truth should be looked after in people's behavior, emotions and cognition

Some Observations

- **Observation 2:** Testing may give a negative feedback (and who likes it?)

Hamlet and Maybee

Testing a program ought to be more fun than designing and coding it, certainly more fun than negotiating with its users about the requirements. For the first time in development, you get to see it work! There's no denying that watching all that information processing horsepower in action is interesting, even exciting. After so much work, running the program should be a reward.

Maybe the reason testing is not always thought of as fun is that there's a flip side: the program may not work. In the earlier parts of development, things can go wrong, but failures are not as absolute and graphic as they are in testing. A developer can even (unconsciously or on purpose) sweep problems under the rug during requirements, specification, design, and coding—but when those problems show up as failed tests, it's no longer possible to kid yourself.

Or maybe the fun of testing diminishes if it isn't your own code being tried. But testing is still detective work that would be a serious challenge for Sherlock Holmes. The program and its specification and design documentation contain the clues to the crime (the lurking crashes), and the tester must find them as a triumph of deduction. Then again, not everyone likes detective work, and many people have no talent for it. Dr. Watson had none.

Some Observations

- **Observation 2:** Testing may give a negative feedback (and who likes it?)
  - The game ends with a failure
  - Popper's perspective

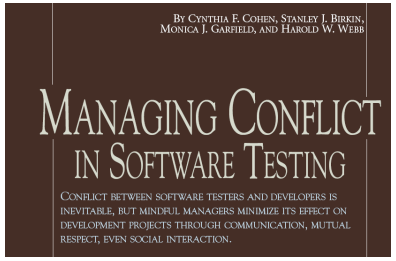
Some Observations

- **Observation 3:** Testing in traditional environments is done by someone else
  - The responsibility is transferred
    - Especially, if it gives a negative feedback!

Some Observations

- **Observation 4:** Testing in traditional software development environments is carried out at the end of the production line, and, inspired by tradition working class jobs, gets low status (which further intensifies the conflict just mentioned).
  - There is a tension between different groups

CACM, 47(1): Managing Conflict In Software Testing, Cohen *et al.*



- Though most organizations recognize the need for high-quality testers and their specialized skill set, testers still struggle to win the respect they deserve. One manager told us, "If you had a diagram with God at the top, the engineers [developers] would put themselves above that." Many testers feel they struggle to maintain their place relative to that of developers.
- P. 80: The lack of status and support makes the tester's job more difficult and time consuming, as the struggle for recognition becomes part of the job itself.

Additional Observations

- Managerial difficulties:
  - Testing slows down development
  - It's hard to manage testing (when)
- Cognitive difficulties:
  - It's hard to know what to test
  - It's hard to know how much testing should be done

Summary of Part A

- Why people don't like testing?
- Testing is hard
  - Time pressure
  - Testing gives negative feedback
  - The responsibility is transferred
  - Low status of testing
  - More…
  - Conflicts within the person and between teams
- Plausible explanation: Software intangibility

Outline

- Why people don't like testing?
- How may Test Driven Development (TDD) help?
- TDD within Extreme Programming

Kent Beck (2001).
Extreme Programming Explained, p. 116

Remember the principle "Work with human nature, not against it." That is the fundamental mistake in the testing books I've read. They start with the premise that testing is at the center of development. You must do this test and that test and oh yes this other one, too. If we want programmers and customers to write tests, we had better make the process as painless as possible, realizing that the tests are there as instrumentation, and it is the behavior of the system being instrumented that everyone cares about, not the tests themselves. If it was possible to develop without tests, we would dump all the tests in a minute.

How TDD copes with the challenge

- Time pressure – TDD inspires a **controlled process**: it is done all the time; it is not postponed to the end of the process; it is done is small steps.
- Testing gives negative feedback – in TDD the rules of the game are reversed; **TDD ends with a success.**

Practitioners' reflection

- Code Unit Test First http://c2.com/cgi/wiki?CodeUnitTestFirst
  Why don't people like testing? Well, the traditional way of testing is tough to take. You write what seems to be perfectly sensible code, then you write a test and the test tells you that you failed. No one wants to hear that.
  Let's turn it around. Write the test first; run it. Of course it fails.. You haven't written the code under test yet. Start writing code.. keep testing. Soon, the test will tell you that you've succeeded! MichaelFeathers
- When it says your code failed (because it wasn't written yet), the UnitTest has itself succeeded. -- EricScouten

How TDD copes with the challenge

- Time pressure – **TDD is done all the time**; it is not postponed to the end of the process
- Testing give negative feedback – **TDD ends with a success**
- The responsibility is transferred – TDD is done **by the developers who write the code**
- Low status of testing – **All developers** are testers

Additional Observations

- Managerial difficulties:
  - Testing slows down development
  - It's hard to manage testing (when)
  - TDD fosters development processes
  - TDD turns development (and testing) to be a **controlled** process
  - **Automatic** (not manually) process

Practitioner's reflection

- A key aspect of this process: don't try to implement two things at a time, don't try to fix two things at a time. Just do one. When you get this right, development turns into a very pleasant cycle of testing, seeing a simple thing to fix, fixing it, testing, getting positive feedback all the way. Guaranteed flow. And you go so fast! Try it, you'll like it.
  RonJeffries

Additional Observations

- Managerial difficulties: TDD turns development (and testing) to be a controlled process
- Cognitive difficulties: TDD improves **understanding** of what you develop
  - It's hard to You know what to test (what)
  - It's hard to You know how much testing should be done

Outline

- Why people don't like testing?
- How may Test Driven Development (TDD) help?
- TDD within Extreme Programming

What is Extreme Programming

What is Extreme Programming

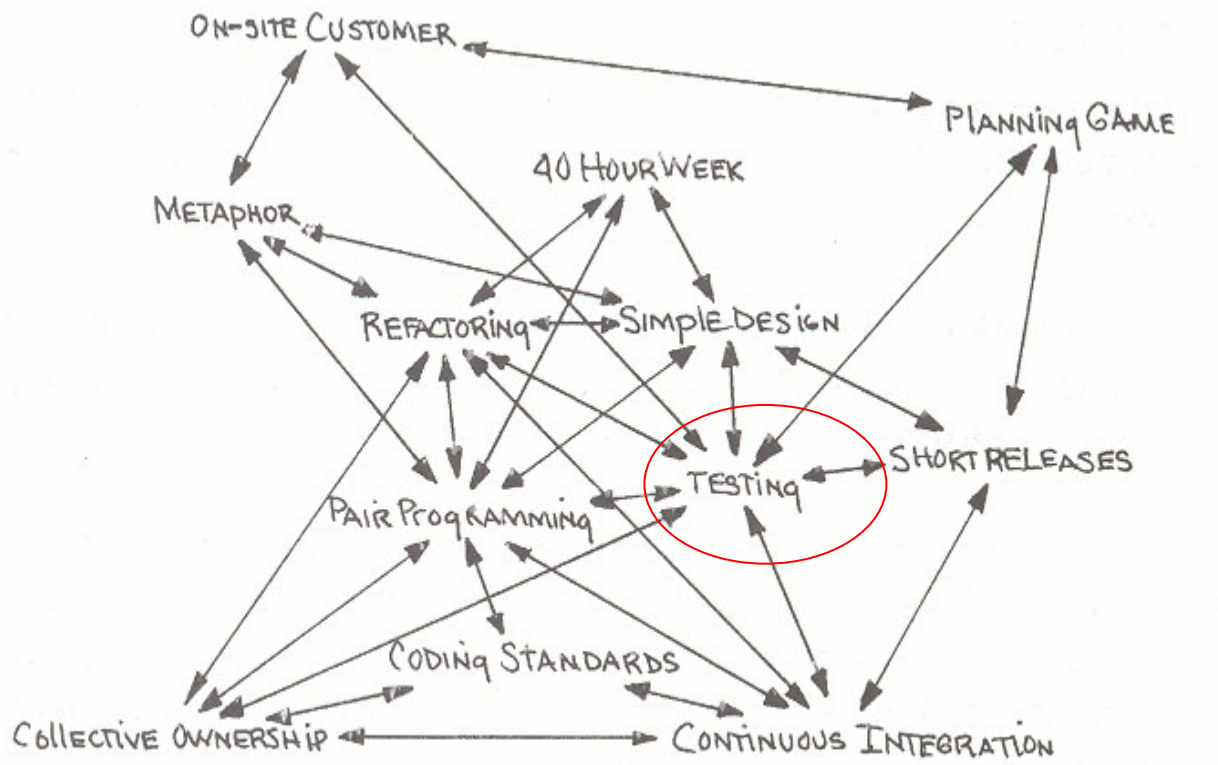


FIGURE 4. The practices support each other

Note:

nothing is new; gathering the practices together is XP uniqueness

Source: Beck, K. (2000). *eXtreme Programming explained*, Addison Wesley.

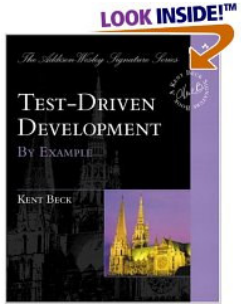
TDD within Extreme Programming

- How does TDD fit in Extreme Programming?
  - Specific rules:
    - Tightness
    - Clear development process

TDD within Extreme Programming

- How does TDD fit in Extreme Programming?
  - Specific rules:
    - Tightness
    - Clear development process
  - Nothing is new

About the book: by Kent Beck
Test Driven Development: By Example



- **From Google:** Clean code that works - now. This is the seeming contradiction that lies behind much of the pain of programming. Test-driven development replies to this contradiction with a paradox - test the program before you write it. A new idea? Not at all. Since the dawn of computing, programmers have been specifying the inputs and outputs before programming precisely. Test-driven development takes this age-old idea, mixes it with modern languages and programming environments, and cooks up a tasty stew guaranteed to satisfy your appetite for clean code that works - now.

TDD within Extreme Programming

- How does TDD fit in Extreme Programming?
  - **Specific rules:**
    - Tightness
    - Clear development process
  - **Nothing is new**
  - **Fosters feedback**

Test-Driven Development in XP

Ron Jeffries, 11/08/01,
http://www.xprogramming.com/xpmag/whatisxp.htm#test

- Extreme Programming is obsessed with feedback, and in software development, good feedback requires good testing. … Almost effortlessly, teams produce code with nearly 100 percent test coverage, which is a great step forward in most shops. …

Test-Driven Development in XP

Ron Jeffries, 11/08/01,
http://www.xprogramming.com/xpmag/whatisxp.htm#test

- It isn't enough to write tests: you have to run them. Here, too, Extreme Programming is extreme. These "programmer tests", or "unit tests" are all collected together, and every time any programmer releases any code to the repository (and pairs typically release twice a day or more), every single one of the programmer tests must run correctly. One hundred percent, all the time! This means that programmers get immediate feedback on how they're doing.

TDD within Extreme Programming

- Specific rules:
  - Tightness
  - Clear development process
- Nothing is new
- Fosters feedback

Conclusion

- TDD may help in coping with
  - Cognitive
  - Affective
  - Social
  - Managerial

  challenges of testing.

Practitioner's reflection

- http://xp.c2.com/UnitTest.html

  It's a wonderful … experience … .

  I don't write code any other way anymore.

  My code has less problems, I have more confidence and management has more confidence. – *sg*

Questions?

Orit Hazzan
Email: oritha@techunix.technion.ac.il