

Position Statement ESE Test-Driven Development

I am test-infected since 1998. I use test-based development for almost all projects I work on and I motivated (forced) my employees, assistants and students to use automated testing for their projects. I still regret that I have to maintain pieces of code I've written without automated tests.

I try to improve the situation that except for java, almost all other languages supported by the Eclipse platform lack a similar support for test-automation and refactoring. My group of students and assistants is working hard to provide refactoring for C++, Ruby, Python and next semester also Javascript, PHP, and groovy. I myself created an easier to use C++ Unit Testing framework called CUTE, that we provide also an Eclipse CDT plug-in for. Having both unit test automation and refactoring for C++ is something I needed for my teaching of modern C++ and what is definitely needed for the huge C++ code bases.

The bad thing about TDD or any test-close-to-code approach is that if people are doing it badly and with little experience you might end up in a never-ending test-refactoring rollercoaster. As programming, writing good automated tests at the "right" level requires exercise and experience. Also self-reflection ability to see, what can be improved, is required. TDD is not something you can just run a 1 day training course and expect everybody to be able to do it well.

Just providing tools for doing so, help the initiated to be more efficient but might lead the inexperienced developer into the dead-end of over-specified or fragile test cases.

The understanding of who writes what level of automated tests and when is hard to get up front and IMHO requires experience in the concrete setting. Just saying "we do TDD" is too little.

In addition, if you are a development shop selling contracts based on time and material, using a test-driven approach might be bad economically, when your quality is too high. I've had customers cancelling maintenance contracts, because of the high quality of our solutions (still operational today!) in the past.

Nevertheless, those students and employees that I urged to try long enough writing automated tests for their code to "get it", stick with this behavior, even when their new environment is not always appreciating it.

As a summary: I am a long time practitioner of writing test close to code. Personally I often try to write test-first and I believe in its quality, since it can lead to simpler code. I also teach TDD and refactoring and try to improve the tool situation based on Eclipse for languages other than java. Maybe, we can come up for even better tools for java as well (i.e., we provided some quick-fixes for the FindBugs plug-in). The future might bring find-test-case-bugs and fixes as well. I hope this qualifies me as a viable ESE symposia participant.

Bio:

Peter Sommerlad is professor and head of Institute for Software at HSR Rapperswil. Peter is co-author of Pattern-oriented Software Architecture Vol.1 and Security Patterns. His long-term goal is to make software simpler by Incremental Development: Refactoring software down to 10% its size with better architecture, testability, quality and functionality.