

Test-Driven Development and Software Maintenance

Scott Tilley

Department of Computer Sciences
Florida Institute of Technology
stilley@cs.fit.edu

Abstract

Test-driven development is a relatively new approach to software engineering, involving the iterative construction of test cases first, and then the application code that passes the test cases second. This panel session will discuss the impacts of test-driven development on long-term software maintenance costs. The panelists represent different research disciplines related to this topic, including software maintenance, software testing, program redocumentation, program understanding, and empirical studies.

Keywords: test-driven development (TDD), software maintenance, empirical studies, economics, quality

1. Motivation

Test-driven development (TDD) [1] techniques, such as Extreme Programming (XP) [2], emphasize the role of unit testing in improving the quality of software products. In the TDD model, test cases are written first, before any application code is developed. Once the test cases are ready, the “real” program is constructed. This is the reverse order of a traditional development process.

Advocates of TDD typically describe the benefits that accrue from adopting this approach to software engineering in terms of initial application development. For example, delivery schedules can be tightened by reducing the amount of time spent in the latter part of the project, because the intensive testing effort normally associated with the last phase of the software lifecycle has been going in parallel with other activities during the entire development process.

History may show that TDD is indeed a superior approach to software development. However, it is unclear if TDD is the best approach when it comes to the long-term maintenance costs of the resultant application. For example, does the availability of a large number of unit tests support code refactoring in the future? What role can unit tests play in supporting program

understanding [3] (in lieu of traditional documentation aids)? How maintainable are the tests themselves? These are important questions that underscore the current lack of objective evidence to support (or refute) many of the claims attributed to the benefits of TDD over a product’s entire life cycle.

Since the vast majority of cost and effort for a product is expended during post-development maintenance, it is critical to know if TDD really produces better results than other software engineering practices when all associated costs are amortized over the product’s lifetime, and not just for the first release.

2. Structure

The panel will begin with a short introduction to the topic by the moderator (see below). Following the presentation of this motivational material, each panelist will present a short position statement of approximately ten minutes. The remainder of the session will be a structured discussion, involving active participation from the rest of the audience.

This panel is part of a broader research initiative focused on determining the impact of TDD on long-term software maintenance. The overall goal is to better support the engineering of maintainable software products. Providing objective evidence as to the efficacy of TDD in this context is one way of achieving this goal.

One desirable outcome from this inaugural panel session would be an agreement to develop a community of interested researchers that are willing to collaborate on this interdisciplinary problem. The Center for Software Testing Education and Research (CSTER) [4] at the Florida Institute of Technology might be used to coordinate such an effort.

3. Panelists

The panel moderator is **Scott Tilley**. He is an Associate Professor in the Department of Computer Sciences at the Florida Institute of Technology. His research interests include software evolution, program redocumentation, technology adoption, end-user programming, and automotive software systems. He has a Ph.D. from the University of Victoria. He is Chair of the Steering Committee for the IEEE Web Site Evolution (WSE) series of event, the current President of the ACM Special Interest Group on Design of Communication (SIGDOC), and General Chair of SIGDOC 2004.



Arie van Deursen is a Professor of Software Engineering at the Delft University of Technology, The Netherlands. He also works at CWI as part of the Software Renovation Research Group. His research interests include reverse engineering, program comprehension, aspect-oriented programming, embedded systems, and software architecture. He holds a Ph.D. from the University of Amsterdam.



Cam Kaner is a Professor of Software Engineering in the Department of Computer Sciences at the Florida Institute of Technology. His research interests include software testing, software metrics, and computer law & ethics. He received a J.D. from Golden Gate University, and a Ph.D. from McMaster University. He is the Director of the Center for Software Testing Education & Research (CSTER), an NSF-funded initiative focused on improving the education of software testers.



Panos Linos is a Professor of Computer Science and Software Engineering at Butler University. His research interests include program understanding, software maintenance, and curriculum development. He received a Ph.D. from Wayne State University. He directs the Center for Applied Software Engineering Research (CeASER), and coordinates the EPICS community service-learning program. He is the General Chair of ICSM 2004.



References

- [1] Astels, D. *Test Driven Development: A Practical Guide*. Prentice Hall, 2003.
- [2] Beck, K. *Extreme Programming Explained: Embrace Change*. Addison-Wesley, 1999.
- [3] Chikofsky, E.; and Cross, J. "Reverse Engineering and Design Recovery: A Taxonomy." *IEEE Software* 7(1):13-17, January 1990.
- [4] The Center for Software Testing Education and Research (CSTER). Florida Institute of Technology. Online at <http://www.testingeducation.org/>.