

---

## Eclipse Summit Europe 2007

Symposia "Integrating Test-Driven Development in the Development Process"

---

### Summary



Christine Mitterbauer, George Mesesan  
MicroDoc Computersysteme GmbH  
Elektrastrasse 6  
D-81925 München  
[www.microdoc.com](http://www.microdoc.com)  
[microdoc@microdoc.com](mailto:microdoc@microdoc.com)

---

## **Marc Hoffmann (introducing TDD) - Mountainminds, Germany:**

- Reasons for TDD: time to market , system test overhead, costs
- It's all about the people
- Important factors for introducing:
  - Working setup for all developers -> instant test execution
  - Training/HowTo: assertions, write units, decoupling, dependency injection
  - Rules: never break the build, commit often, continuous integration
  - Tools: provide proper tools
- Experience: project has to fail first -> then TDD
- Developer resistance -> Failure
- education is the most important basis (Hr. Brede)
- [www.elemma.org](http://www.elemma.org)

---

## **Peter Sommerlad (overview of TDD) - HSR Rapperswil, Switzerland:**

### Recommended Books:

- Gerard Meszaros - XUnit Patterns (<http://xunitpatterns.com/>)
- Michael Feathers - Working effectively with legacy code
- Dave Astels - Test-Driven Development
- Kent Beck - Test-Driven Development by Example
- J.B. Rainsberger - JUnit Recipes
- Andy Hunt, Dave Thomas - Pragmatic Unit Testing using Junit
- Andy Hunt, Dave Thomas with Matt Hargett - Pragmatic Unit Testing in C# with NUnit 2nd Ed.
- Mike Clark - Pragmatic Project Automation

### Possible problems with TDD:

- Need for reasonable good programmer
- Need to refactor code and tests
- Ugly test -> shows bad design
- Bad: if tests depend on external resources
- Refactoring without tests is dangerous
- Refactoring automation tools are necessary (tools developed for Ruby, Python, JavaScript, Groovy...)
- Bug tracking system forcing the user to write automated test case for each bug in order to demonstrate the bug
- Continuous Integration is very important
- Having no bugs is kind of dangerous-> no contact to customers -> maybe bad marketing

---

## **Brett Hackleman (introducing TDD & field report) - Band XI, USA:**

- Fitness: Wiki interface for a FIT interface
- Implemented Fitness plug-in for eclipse
- Question: autogenerated Fitness code ? -> no

- GSM/GPS Tracker project: Hardware and Software developed with TDD approach, bugs are found earlier, quicker to get new features

---

## **George Mesesan (field report – TDD within the company) – MicroDoc, Germany:**

- Three test levels: acceptance, interface, inner module tests
- Fitness for acceptance tests, JUNIT for the others
- Customer feedback was good but no acceptance test input, no TDA from the customer
- Test through interfaces – behaviour testing
- Writing “good”, valuable tests is challenging -> developer has to change approach to writing/reading code
- Definition of good tests: non-overlapping, effective test data, not testing trivial code, maintainable, test modules/unit (not classes)
- TDD helps with module design
- TDD produces better code in less time:  
 $\text{time}(\text{test first}) + \text{time}(\text{code}) < \text{time}(\text{code first}) + \text{time}(\text{test}) + \text{time}(\text{debug/fix})$

---

## **Rene Kießig (field report – TDD within the company) - ALEA, Germany:**

- Client Server Application
- Problems: no experience, Cruise Control, UI Testing
- Iterations: 6 weeks
- Tests only for business logic -> less logic in the UI -> helps separating the model from the view
- Tests actually takes too long -> tests must run faster -> possible solution: Mocks, database simulation, only test the new code, do not test the whole framework with each test

---

## **Anthony Bennis (field report – TDD within the company) - Pilz, Ireland:**

- Small project teams (5-6 people), good communication within the team and with customers
- Bad software quality in big new project (35 developers), too many change requests
- TDD advantage: focus on the code which is needed
- TDD introduced by one developer using TDD approach – was test-infected and like an evangelist
- But it's not solving all the problems
- Need people to get test-infected (technical and management)
- No CruiseControl -> but it's in plan
- Problem: nightly build shows problems of 35 developers too late

---

## Uni Siegen (problems with TDD and tools)

- RCP application with tests written afterwards
- plan: use CruiseControl to get releases more often
- Problems: tests are more integration tests than unit tests, tests work in eclipse but not under CruiseControl
- CC is a nightmare to set it up and get plug-in-Tests run

Input from audience:

- Markus Barchfeld: PlugInBuilder on sourceforge.net -> helps setting up CruiseControl
- AntForEclipse – recommended tool

---

## Anna Dushistova (problems with TDD and tools) - MontaVista Software, Russia:

- org.eclipse.releng.builder (for Continuous Integration)
- UI testing with tools not possible
- Try to introduce TPTP -> not successful (tests were not portable between different OS)
- Actually: QA engineer doing UI testing manually

Input from audience:

- TPTP in a lot of projects successfully used – but only as profiler not in the area of Automated GUI Recording

---

## Hans-Joachim Brede (GUI automated testing) - BREDEX, Germany:

- TDD does not work for GUI tests when tests are programmed
- Experience with JUNIT:
  - not suited, bugs found manually not via JUNIT, effort for regression tests too high
- Bad programmers write bad test code, good programmers write bad GUI tests
- Better: someone else should write tests with different skills (usability, UI,...)
- Test paradigm for business code is not suitable for GUI testing
- Proposal: tests without coding
- Eclipse is a highly creative tool
- Acceptance test level
- Implementation:
  - AbstractComponent not coupled to Java implementation which knows: action, parameters (optional), for each GUI item
- GUIDancer not coupled to operating systems
- No change for code which has to be tested
- User interface has time constraints -> GUIDancer tests will take relatively long

---

## Short Summary

The symposia on test-driven development was very interesting and highly productive. We focused on practical aspects of real-life projects and the impact of the test driven approach on all involved parties.

Some of the attendees spoke about actual problems introducing TDD within their projects and the group developed possible solutions for them to move forward with. This was a great outcome and as a result we plan to set up a wiki for further discussions.

Key results of the symposia:

- Education (!), tools and a running tool-setup are basic requirements to be successful
  - TDD is challenging – our main discussion was: what is the difference between a “good” and a “bad” test ?
  - All participants reported consistently that TDD helped them to develop better software quality in less time especially for business logic
  - BUT on the other hand we figured out that TDD for UI is very difficult and there is a lack of adequate tools – fortunately we had Mr Brede in our symposia presenting his GUI testing solution GUIDancer
-