A Course on Traditional and Agile Testing

Dr. Christoph Steindl
Senior IT Architect and Method Exponent
Certified ScrumMaster
Christoph_Steindl@at.ibm.com

Agenda

- A Tale of Two Tracks
- Agile Engineering Practices
- Course Structure
- Pedagogical Ideas
- Grading
- Results and Feedback
- Future Directions
- Online Demo

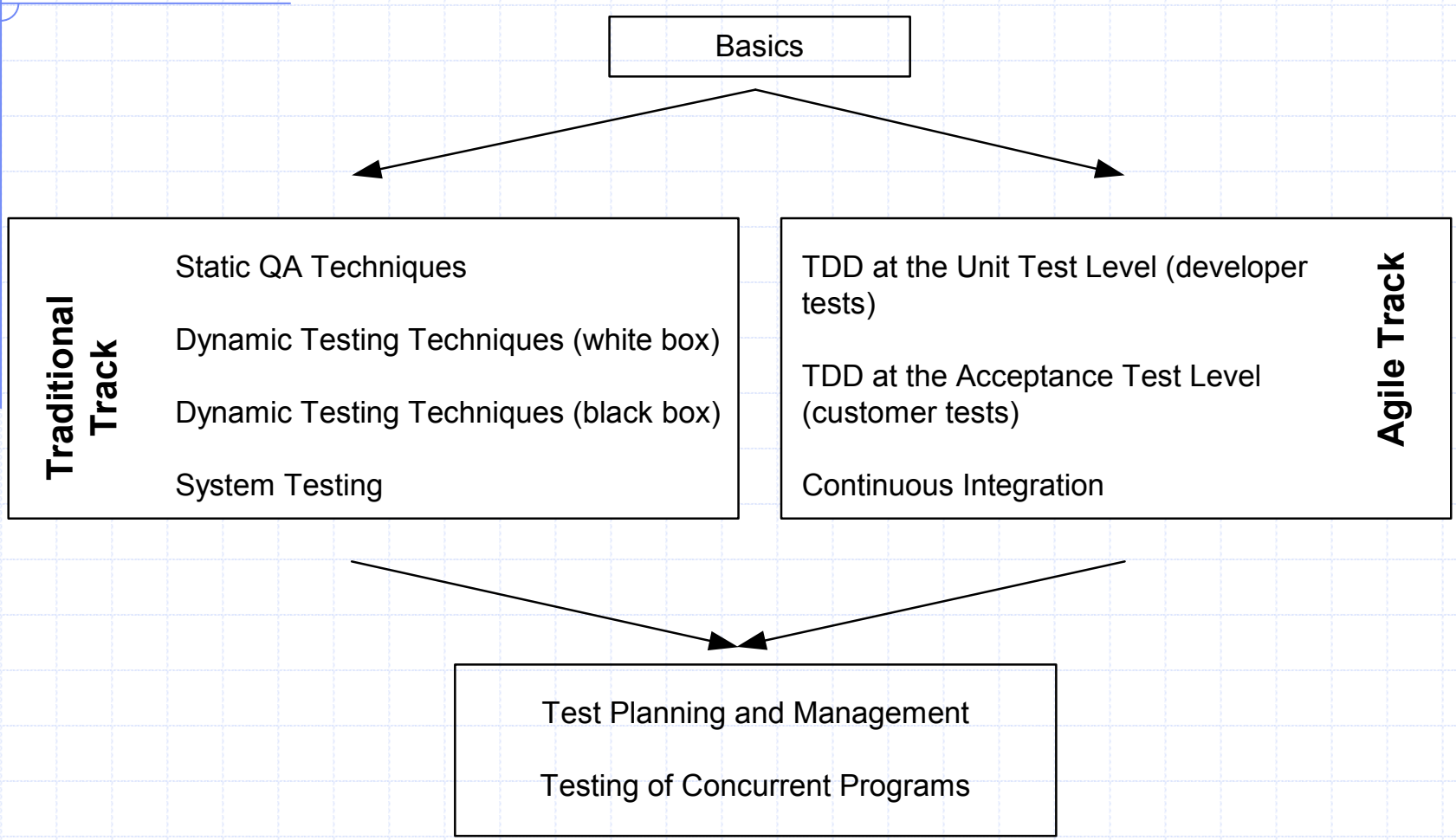
A Tale of Two Tracks

- On a traditionally-managed project
  - Distinct phases delineated by milestones (waterfall-style)
  - The tester is responsible for the test plan, sets up the test environment, prepares the tests, performs them, and reports on the results.
  - The developers hand over the code/system to the tester, the tester tests it and hands back error reports.
  - The tester employs mainly black-box techniques and focuses on functional tests, integration and system tests.
- On an agile project
  - All project work is done within the iteration cycles
  - The tester is part of the development team.
  - The team reaches a common understanding about what features (e.g. functions, content, infrastructure and environment) to test, at what project stage the tests should happen, and what quality aspects should be considered (e.g. functionality, reliability, usability, efficiency, performance, etc.).
  - The tester helps the customer with writing customer tests, advocates test-driven development, teaches the developers testing techniques (mainly white-box, but black-box as well).
  - The tester helps to automate tests and focuses more on customer tests than on developer tests, since the developers are responsible for the latter.

Agile Engineering Practices

1. Having a source code control system (e.g. CVS)
2. Doing code reviews before checking in code (e.g. pair programming or code reviews)
3. Checking in at least daily (even if it leads to build failures)
4. Having daily builds
5. Doing unit tests
6. Having an automated test harness for unit tests (e.g. Junit)
7. Using test driven development at the automated test harness level
8. Doing continuous builds (perform a build every time unit tested code has been checked in and not just once a day, e.g. CruiseControl)
9. Doing acceptance test driven development (apply the acceptance tests once code is checked in, e.g. FIT, Fitnesse)
10. Refactoring

Course Structure - Overview



Course Structure - Details

Day 1

- Ariane 5 (10')
- Basics (60')
- TDD Unit Level (100')
- Static QA (20')
- Dynamic (WB) (20')
- Dynamic (BB) (90')
- TDD Accept. Level (60')
- Retrospective and Homework (15')

Day 2

- System Testing (2,5h)
- Test Planning and Management (1,5h)
- Testing in Practice (1,5h)
- Continuous Integration (50')
- Retrospective and Homework (15')

Day 3

- Student presentations (7 x 20' for presentation and 7 x 10' for discussion)
- Testing Concurrent Programs (45')
- Retrospective and Homework (15')
- Preparation for the exam (30')

Pedagogical Ideas

- Shu-Ha-Ri (http://c2.com/cgi/wiki?ShuHaRi)
- In-class do-it-yourself examples (Berkeley tips #89, #92, #97 [Gr+], and Self-Test Pattern [Ste00])
- In-class demonstrations
- Homework exercises
- Presentation or report
- Invited talks
- Pointers to additional literature
- Real-life stories
- Index cards (Berkeley tip #96)
- Catalogue of questions (Berkeley tip #40)
- Divide your course into parts (Berkeley tip #20)
- Devote the last day of class to an overall review (Berkeley tip #166)

Shu-Ha-Ri (from June Kim)

- In the phase of Shu, the person tries to abide by the rules. She tries to learn all the principles and information by heart. But she can't abide by all the rules while she is doing the practice. Her body (including her brain) starts to remember them bit by bit through repetitious practices. When the time comes she can internalize and abide by all the rules -- when Shu is achieved, Shu phase is finished and she enters into Ha phase.
- In the phase of Ha, she tries to break the (old) rules. She tries to self-reflect on herself and her knowledge, and come up with anti-theses such as exceptions of the rules in the real world. But she can't break all the old rules while she is doing the practice. Her rules start to get more complete (or become more like "case-by-case") as the rules encompass exceptions bit by bit. When the time comes she can break all the rules and see both sides of every rule (maybe substituting with a set of her own rules) -- when Ha is achieved, Ha phase is finished and she enters into Ri phase.
- In the phase of Ri, she tries to leave the rules. She tries to get free from all the rules, and get into the state of no distinction, or into a new dimension. But she can't leave all the rules while she is doing the practice. Her body starts to forget them bit by bit through following natural laws and flows (or Tao). When the time comes she can leave all the rules -- when Ri is achieved, Ri phase is finished and she enters into a new dimension of Shu.
- At the end of Shu, what she sees is nothing but the rules -- everything looks like the rules. At the end of Ha, what she sees is nothing like the rules. At the end of Ri, she doesn't see but work with her mind.

Grading

1. **Final examination** (45 minutes) with a questions-and-answers part and a couple of examples (about unit testing and function testing)
   ➔ weight of 40%, must be positive in order to complete the entire lecture positively
2. **Homework exercises** with detailed instructions. We expect the students to return the exercise sheets with their name and to indicate how long it took to complete the various exercises
   ➔ weight of 30%
3. **Presentation** about a topic (literature of approximately 100 pages) **or paper** about a topic (approximately 20 pages) in groups of four
   ➔ weight of 30%

Results and Feedback

- The course is not compulsory. In the recent years, 20-30 students enrolled for the course. This year (winter semester 2004), approximately 90 students enrolled and 71 finished the course. The Austrian grading scale has 5 values: 1 (very good) to 5 (failed). The distribution for this course was: 42 x 1, 24 x 2, 5 x 3.
- The reports were mostly of good quality. The presentations seem to have been more difficult: to select the right messages and to manage the available time.
- The students spent too much time on installing the tools. On the one hand, it is a good and necessary exercise, on the other hand this time would be better invested in practicing the techniques.
- The students provided (among others) the following feedback:
  - +) Demos of up-to-date tools, breadth of tools, practice-orientedness
  - +) That one example (Triangle) was used for several techniques and technologies
  - +) The homework examples were relatively free or unspecified (after installation of the tools), which helped to get a good understanding.
  - -) Whole days of classroom education are exhausting.
  - -) Homework should be checked, so that students cannot cheat.
  - -) Focus on Java for the tools (would have favoured C#/C++/.NET)
  - -) Having to do group presentation or report
  - +) Detailed instructions on how to install and use the tools that were handed out

Future Directions

Slot	Level	Intended Changes
TDD – Unit	Shu ➔ **Ha**	Add material about unit testing, enhance material on stubs and mock objects, enhance material on in-container-testing; practice mock objects and in-container testing in the homework assignment.
Dynamic – BB	Shu ➔ **Ha**	Enhance material for functional testing of GUIs, add in-class example; practice the technique in the homework assignment.
TDD – Acceptance	Shu ➔ **Ha**	Add in-class example for writing a Fixture; practice the technique in the homework assignment with non-trivial example.
Homework 1 + 2	Shu + Ha	Reduce effort for installation of tools by providing a VMware image, have the students pair for the homework; configure a Yahoo-group for discussion of issues and problems.
System Testing	Shu	Divide in smaller pieces, add in-class examples.
Test planning and management	Shu	Divide in smaller pieces, add in-class examples.
Student Presentations or Reports	Ha ➔ **none**	Probably remove and save time for other slots.
Concurrency	Shu	Spend more time on bug patterns, add in-class demo; practice the technique in the homework assignment with non-trivial example.

References

All material for the course is available at (some parts in German, some parts in English): http://www.ssw.uni-linz.ac.at/Teaching/Lectures/Testen/2004/

[Gr+] Barbara Gross Davis, Lynn Wood, Robert C. Wilson: *A Berkeley Compendium of Suggestions for Teaching with Excellence*, http://teaching.berkeley.edu/compendium/

[Ste00] Christoph Steindl: *SelfTest Pattern*, in Martine Devos and Andreas Rüping: Proceedings of the 5th European Conference on Pattern Languages of Programs, Universitätsverlag Konstanz, 2001.