Addition of the Kanban Method
http://youtu.be/kcEX31U4MFk

Daniel Speicher, Pascal Bihler, Paul Imhoff, Günter Kniesel, Holger Mügge, Jan Nonnen, Tobias Rho, Mark von Zeschau, Armin B. Cremers
Computer Science III, University of Bonn, Römerstraße 164, 53117 Bonn - {dsp, bihler, gk, muegge, nonnen, rho, mvz, abc}@cs.uni-bonn.de

10 Years of Agile Lab Courses for International Students

The Institute of Computer Science III of the University Bonn regularly offers Agile Lab Courses to students from Germany and all over the world as part of the International Program of Excellence at the Bonn-Aachen International Center of Information Technology. In the recent ten years we offered about 10 courses with a duration of four to six weeks. Typically around twelve students are introduced into Agile Software Development by one and a half to three colleagues. During this time the teams develop software of realistic complexity that is of real value for a research project or an external customer.

Experiencing the Real Thing
In the article that is commonly seen as one of the founding documents of the waterfall approach to software development [Roy70] Winston W. Royer wrote: “The testing phase which occurs at the end of the development cycle is the first event in which timing, storage, input/output transfers, etc., are experienced as distinguished from analyzed. These phenomena are not precisely analyzable.” Although he recommends thorough analysis and careful design he was only aware of the dangers of late insight into unforeseeable problems. In a way most of our students are in the same situation with respect to their software technology knowledge. They get an overview of traditional and Agile approaches and are able to reproduce their knowledge on an exam. Yet, they do not know the experience whether their understanding is strong enough to contribute to a real software project. This is what our courses are out to offer.

Early Substantial Feedback
Besides technical challenges, long development cycles lead to the problem of outdated business needs and customer expectations. This is why Extreme Programming and Scrum focuses so much on the early feedback. In one of the first days we ask the students to implement some functionality that cuts through all technological layers so that they get some first feedback about the required technologies. The functionality of the system does not need to be extremely simple. The usage of tests is strongly encouraged and the students at the right light that makes the most of the recent science of the continuous integration serve as visible. Introducing continuous integration was one of many improvement suggestions that we got over the years from an external industry consultant, Michael Mathies. We make sure that the students understand that software development is not only about taking some tricky technical tasks but about providing value to someone who would taut the actual need of the company and the project stakeholders. This is why we offer a lot of feedback together with the feedback from the customer [MSK04] evaluating the value of different functionalities while leaving the technical decision to the students (the other colleagues).

We introduce this perspective using the XPframe and encourage to present the results of an iteration truthfully from the perspective of the customer.

Reflective Improvement for Responsible Students
Software of realistic complexity requires a broad range of knowledge. We investigate on the number of students by asking everyone to become an expert in one area. This requires a very deep knowledge base and consequent pair programming allows the knowledge to extend into the team. The students are offered the responsibility to estimate the effort a certain functionality requires. With our support we train them that they can accept this responsibility. Planning poker during planning is very helpful in this respect and also progressive saying gives feedback about the quality of their estimates. Daily stand-up meetings make sure that problems become obvious early and everyone shares the latest insights that are of value to the team. Each iteration ends with the presentation to the customer and the other colleagues. In a respectful atmosphere and with the help of the colleagues they can discuss the challenges freely and actions addressing them can be found.

What did we learn?
The students are typically very satisfied with the course, praise the friendly and productive atmosphere and are surprised by the improvement of their skills. Nevertheless we are not always successful from the start, so that we can offer as well some lessons learned. There need to be some breadth within the project so that two or more somewhat independent functional areas can be addressed by the students in parallel. Setting up a testing framework for a specific technology is beneficial and can be supported but not completed by a single course of four to six weeks. There is a limit to what can be estimated appropriately. If the team is in doubt that something takes more than two days it is probably too big to be estimated. Authorizing that finishing functionality should be tested over integrating it into a new one is not enough as the final integration seems too tedious and challenging. Visualizing the work and placing limits on the work in progress finally solved this issue for us [MSK15].