Final Feedback - Evaluation Results

Summary:

The Extreme Programming Lab 2007b was perceived as being a very successful one. The participant in general enjoyed being part of the team and the character of the lab challenge (developing an adaptive mobile game) helped to raise the students' motivation. Stories, which mainly focused on research questions and whose contribution to the game development was not directly visible tend to be disliked.

For every aspect of the practical course, at least on of the students rated it "excellent" respectively "appropriate". The most space for improvements might be in the usage of the wiki as a common knowledge base. Here, openness of the approach and the need for a general structure were sometimes counteracting. The same was criticized for the product architecture, as some students felt lost in a rapidly and parallel developing code structure.

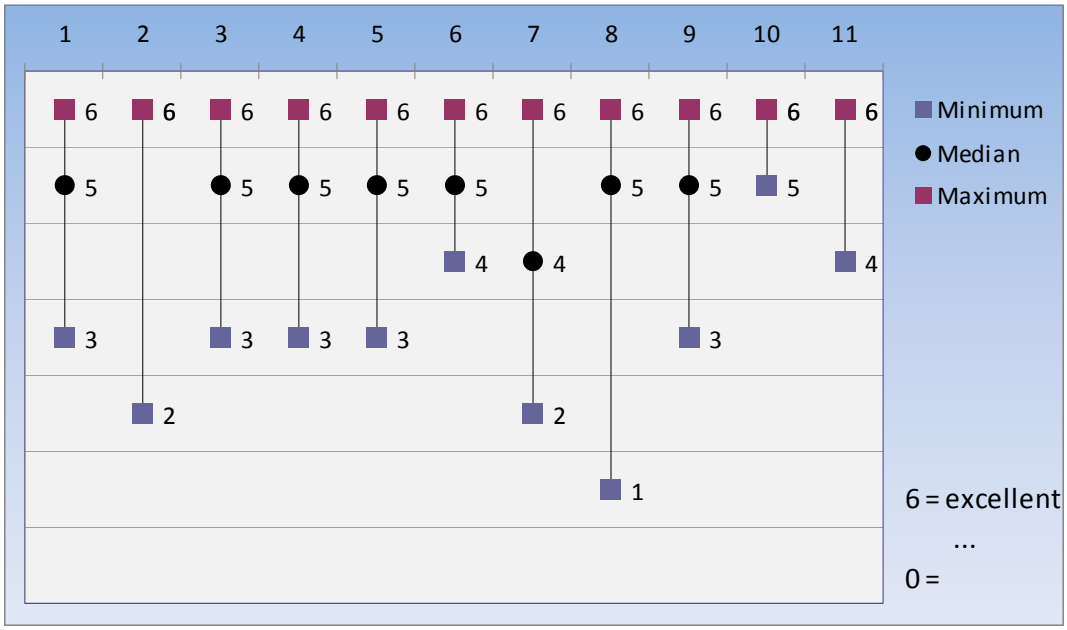
XP-skills like testing, task-estimation, or pair-programming were developed by the participants, communication and programming skills have been supported by the lab as well. The use of English as a lab language didn't seem to be a problem for most of the participants.

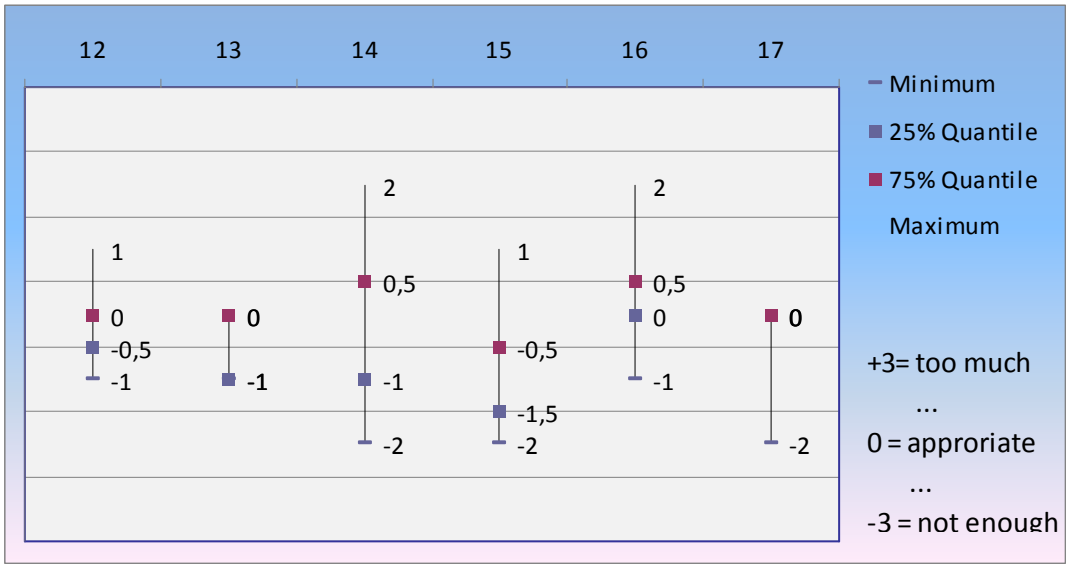
The participants agreed that the preparation time upfront the lab was important and they felt prepared for the lab challenges. Nevertheless, no request to further extend the preparation phase could be identified and the introduction of more school-like tutoring before and during the lab was discouraged.

The challenge for future labs is the improvement of using results from social feedback loops like the final iteration reflection. Here, students criticized that findings did not lead as much to changes in the following iterations as it might have been possible. Nevertheless, the preliminary preparation and the total execution of the lab were heavily lauded by all students and a continuation of the lab series was recommended.

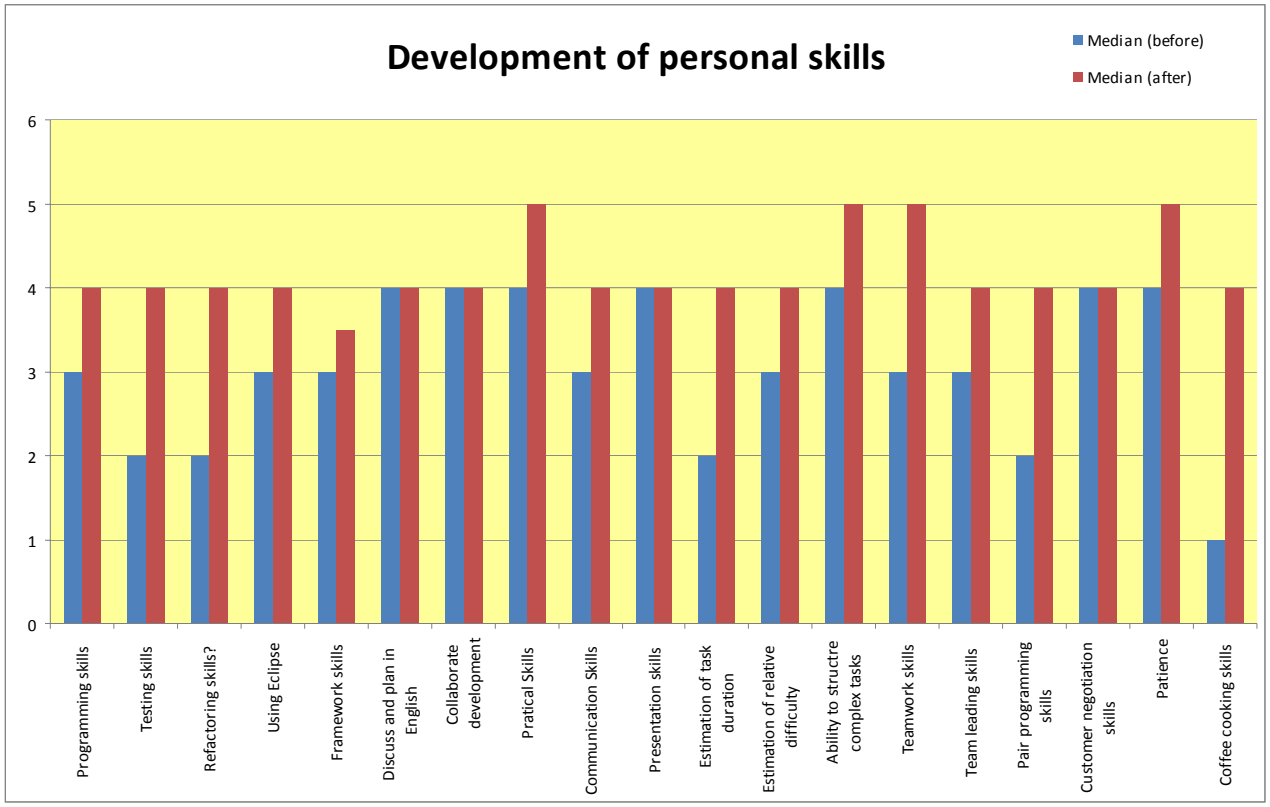
Analytical Images:

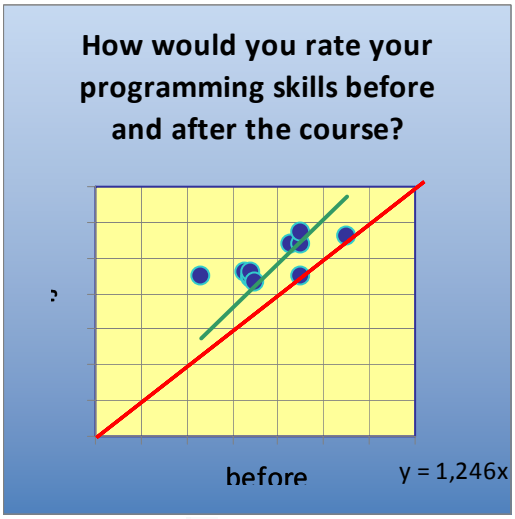
Questions about the practical course (Q 1-17)

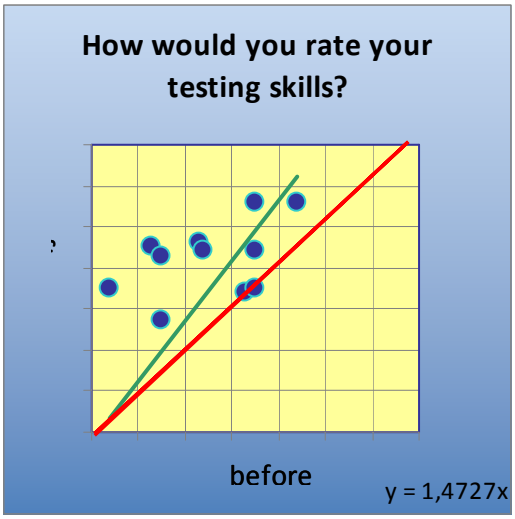


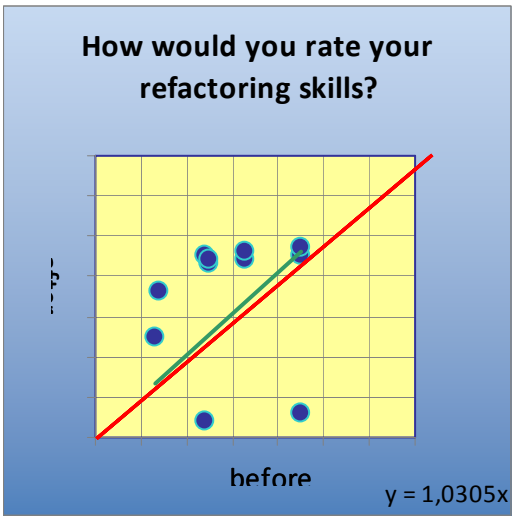


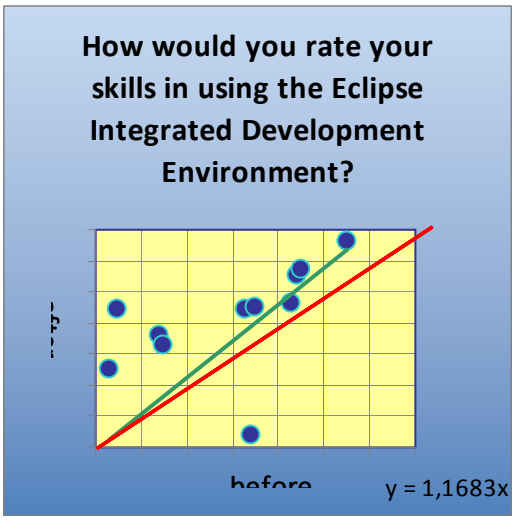
Self assessment

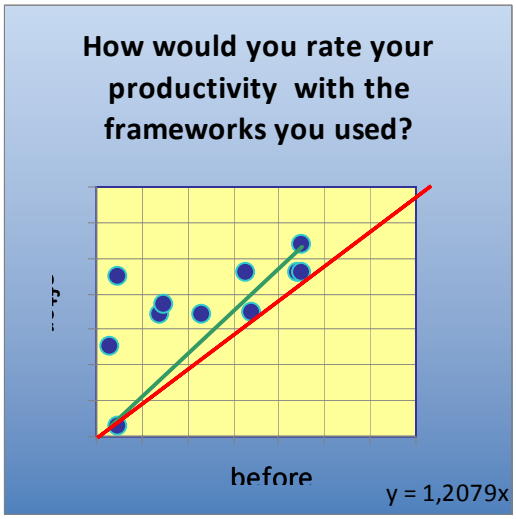


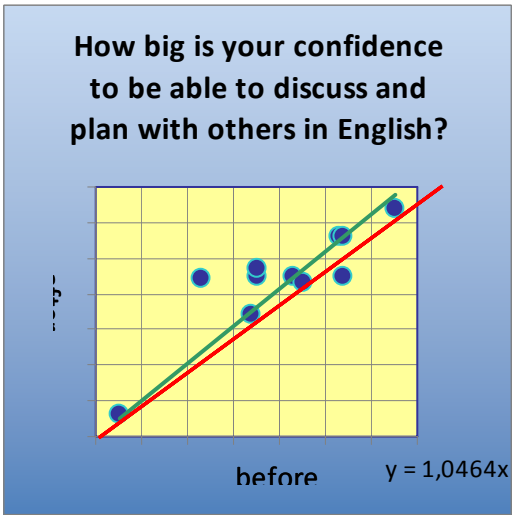


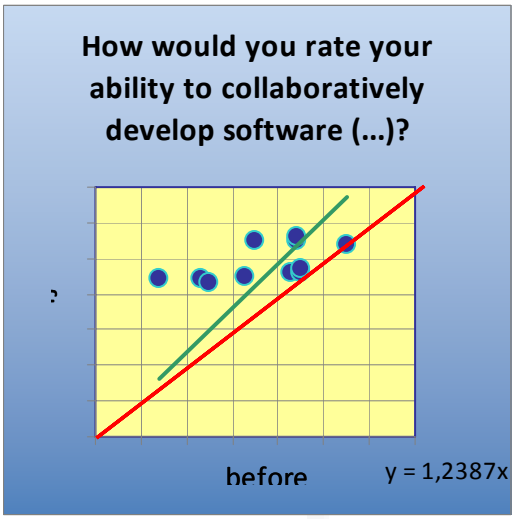


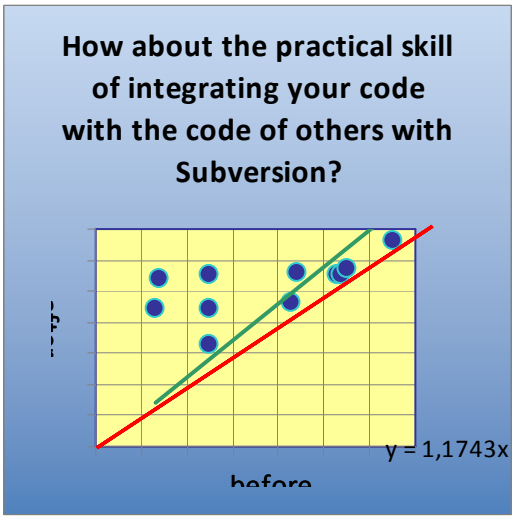

How would you rate your refactoring skills?
before
y = 1,0305x

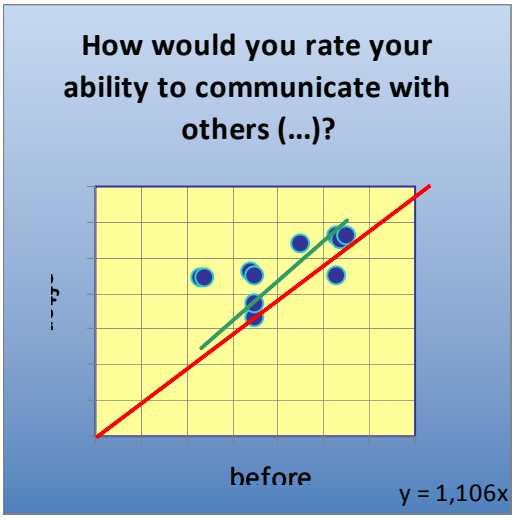

How would you rate your skills in using the Eclipse Integrated Development Environment?
before
y = 1,1683x

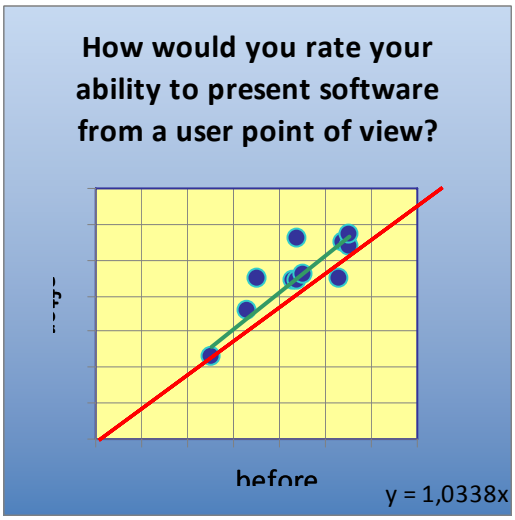

How would you rate your productivity with the frameworks you used?
before
y = 1,2079x

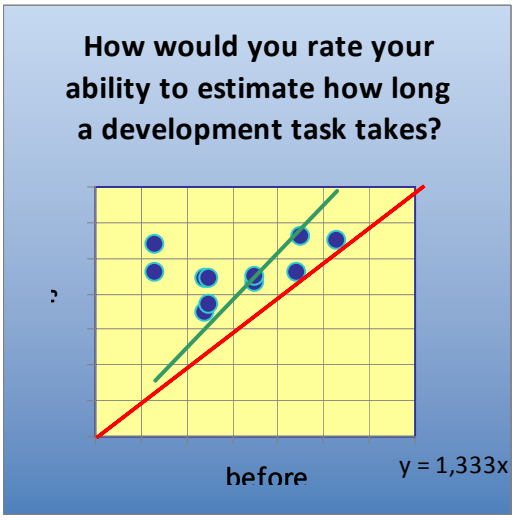

How big is your confidence to be able to discuss and plan with others in English?
before
y = 1,0464x

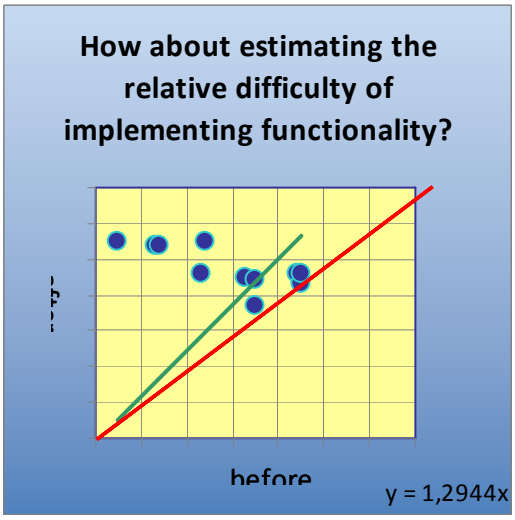

How would you rate your ability to collaboratively develop software (...)?
before
y = 1,2387x

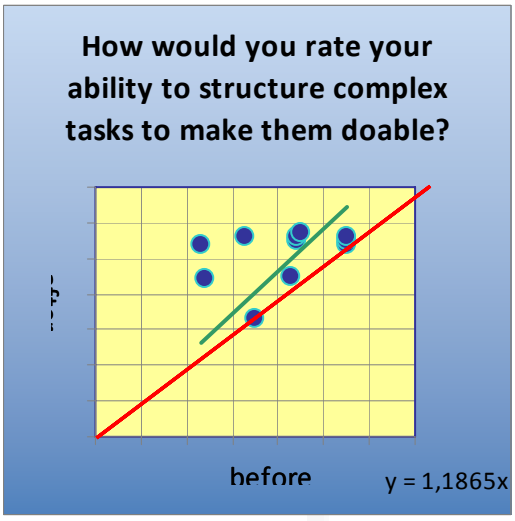

How about the practical skill of integrating your code with the code of others with Subversion?
before
y = 1,1743x

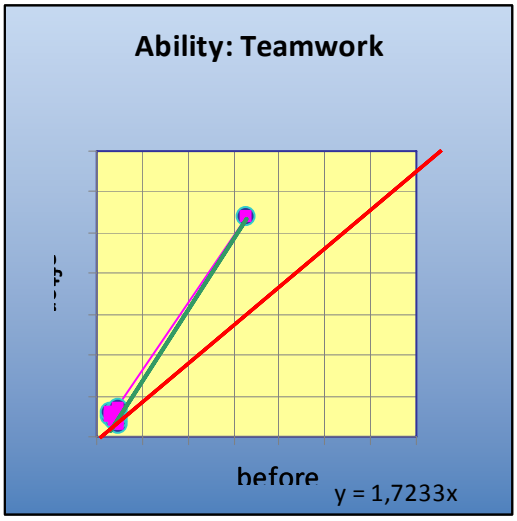

How would you rate your ability to communicate with others (...)?
before
y = 1,106x

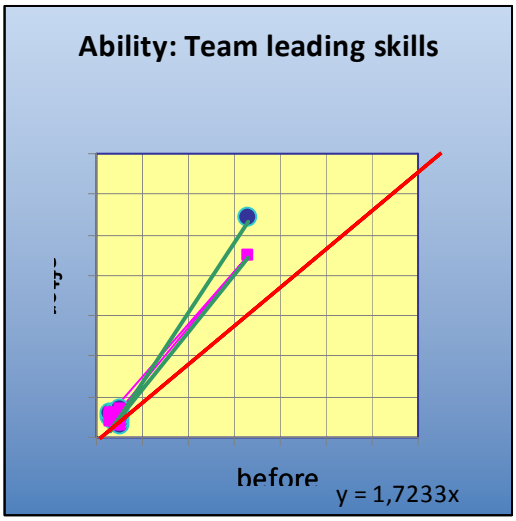

How would you rate your ability to present software from a user point of view?
before
y = 1,0338x

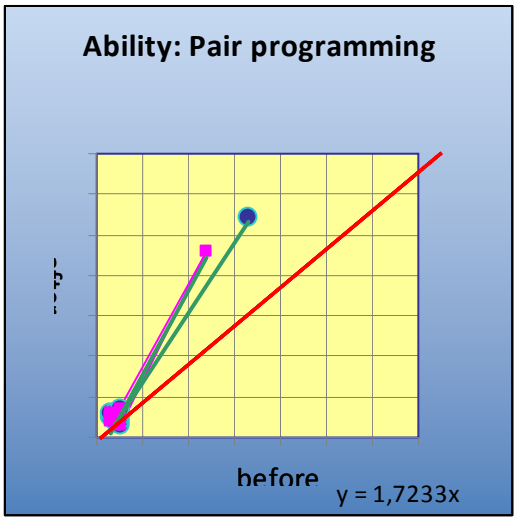

How would you rate your ability to estimate how long a development task takes?
before
y = 1,333x

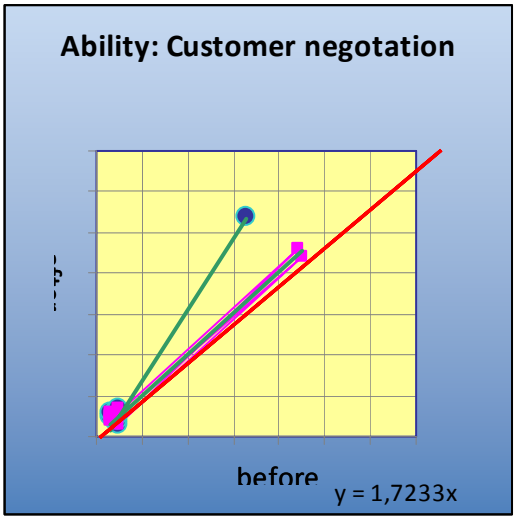

How about estimating the relative difficulty of implementing functionality?
before
y = 1,2944x

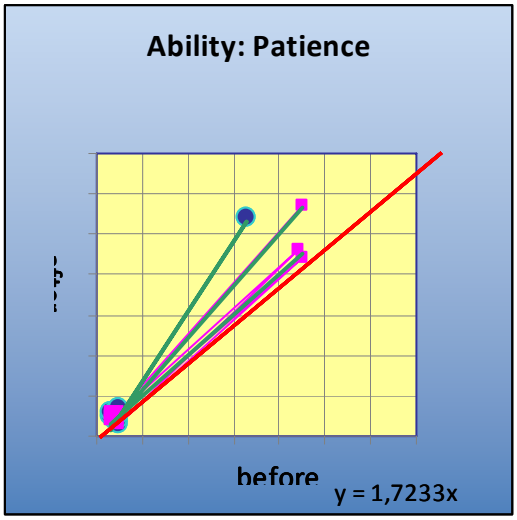

How would you rate your ability to structure complex tasks to make them doable?
before
y = 1,1865x

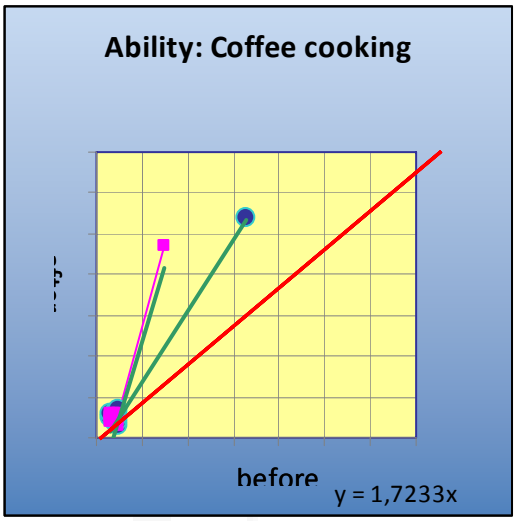

Ability: Teamwork
before
y = 1,7233x


Ability: Team leading skills
before
y = 1,7233x

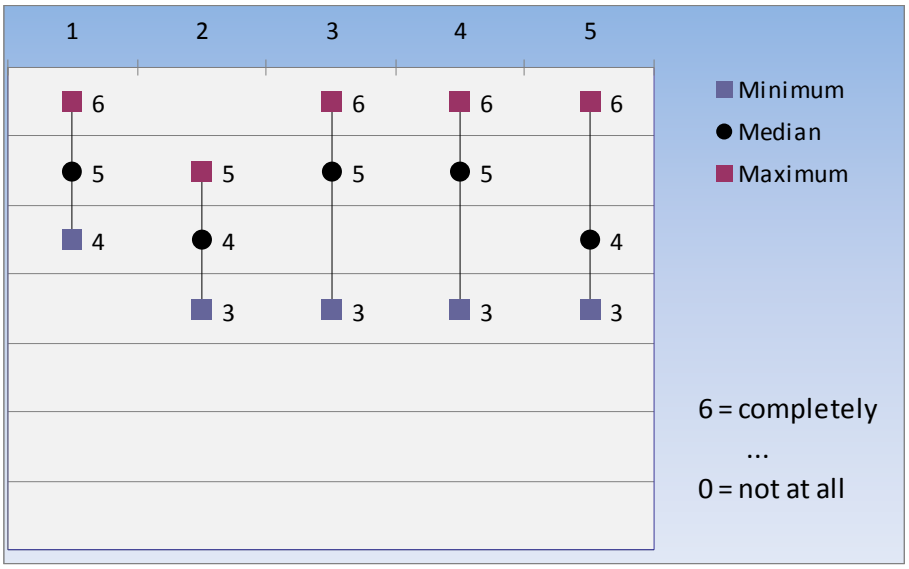

Ability: Pair programming
before
y = 1,7233x

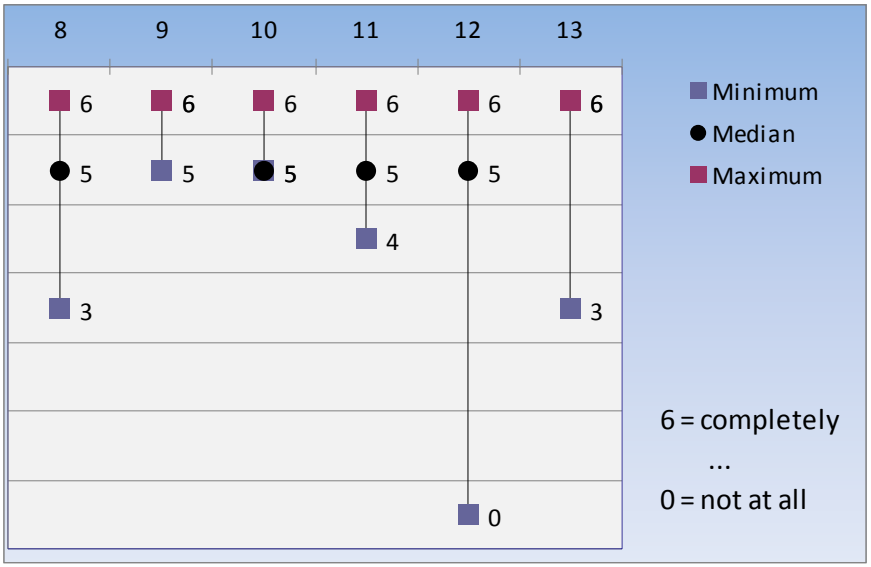

Ability: Customer negotation
before
y = 1,7233x

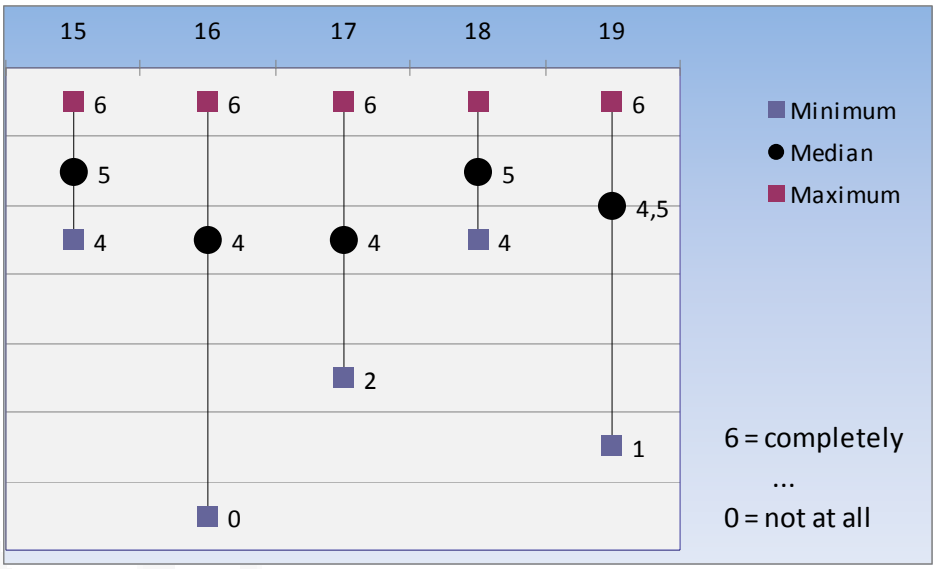

Ability: Patience
before
y = 1,7233x

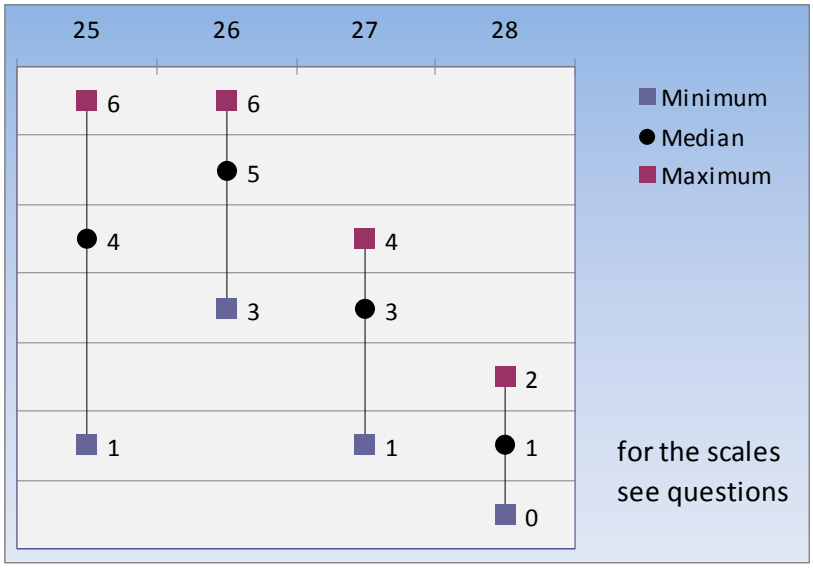

Ability: Coffee cooking
before
y = 1,7233x

Overall organization, style, atmosphere

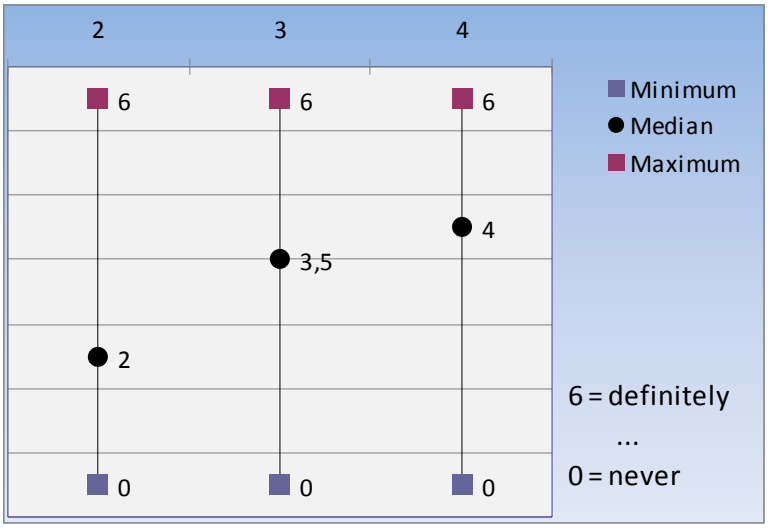








Feedback to some ideas about improving the course



Detailed Responses:

Following the statistic evaluation of the feedback forms. Textual comments as given by the participants.

Please give us your rating of the following elements of the practical course:

		Median
1)	XP-Game to explain the real Planning Game.	**Very good**
2)	Shared Breakfast to start relaxed in the day.	**Excellent**
3)	Stand up meeting to avoid getting stuck, to share knowledge, to build team spirit.	**Very good**
4)	Planning Poker to quickly find a shared estimate of the required effort.	**Very good**
5)	Restricting the possible values to 1, 2, 3, 5, 8, and infinite bricks.	**Very good**
6)	Reflecting after each iteration about our practices.	**Very good**
7)	Wiki as a collective knowledge base.	**Good**
8)	The "First Day Challenge" to get a reliable impression about techniques	**Very good**
9)	Reflecting from time to time about our practices.	**Very good**
10)	Subversion as a shared repository.	**Excellent**
11)	CruiseControl and the Rapid Feedback Device (Traffic Light)	**Excellent**
12)	Defining functionalities as stories written on index cards.	**Appropriate**
13)	Defining implementations steps as tasks written on index cards.	**Appropriate**
14)	Discuss about design when defining the tasks that are necessary to implement a story.	**Appropriate**
15)	How was the level of detail of the information on the task cards? (Implementation details, Task dependencies)	**Improvable**
16)	Burn-Down Charts and its constant visualization to get a clear picture about the progress.	**Appropriate**
17)	One hour lunch break at 1pm, to start fresh in the second part of the day?	**Appropriate**

Comments:

Task-Break-Down lead to very long discussions on how to actually implement the stuff and what tasks were really needed, so this enlarged the break-down too much.

I believe there should be an extra step of interaction with customer between stories and tasks for story evaluation because some stories contradict or add to each other or just must be reworked before implementation.

Self assessment:	*Median progress*	*# Resp.*
1) How would you rate your programming skills before and after the course?	**+1**	*11*
2) How would you rate your testing / test-first skills before and after the course?	**+2**	*11*
3) How would you rate your refactoring skills to achieve simple design / code quality?	**+2**	*9*
4) How would you rate your skills in using the Eclipse Integrated Development Environment?	**+1**	*10*
5) How would you rate your productivity while implementing with the framework(s) your team used?	**+0.5**	*8*
6) How big is your confidence to be able to discuss and plan with others in English?	**±0**	*10*
7) How would you rate your ability to collaboratively develop software (doing your part of a big whole, building on the results of others)?	**±0**	*11*
8) How about the practical skill of integrating your code with the code of others with Subversion?	**+1**	*11*
9) How would you rate your ability to communicate with others (explaining your ideas, asking for help, getting the ideas of your colleagues)?	**+1**	*11*
10) How would you rate your ability to present software from a user point of view?	**±0**	*11*
11) How would you rate your ability to estimate how long a development task takes?	**+2**	*11*
12) How about estimating the relative difficulty of implementing functionality? (Bricks)	**+1**	*11*
13) How would you rate your ability to structure complex tasks to make them doable?	**+1**	*11*
14) Ability: Teamwork	**+2**	*1*
14a) Ability: Team leading	**+1**	*1*
15) Ability: Pair programming	**+2**	*1*
15a) Ability: Customer negotiation	**±0**	*2*
16) Ability: Patience	**+1**	*1*
16a) Ability: Coffee cooking	**+3**	*1*

17) To be concrete: Are there things you definitely would have liked to learn in this course, but did not get the opportunity to?

Communication with a real customer without any computer knowledge
Finishing integration of Roots research projects and see them in action
More applied refactoring would be nice
How to refactor to patterns / Designing skills (Design patterns)
More about GPS and Bluetooth

18) Could you name the most important thing you learned in this course?

Teamwork, SOA, Test-First development, Developing in a larger group, XP Paradigm, Refactoring, Communication helps, OSGI (Ditrios), Web-Services
At least, I get to know the identity of TDD
I don't know. Planning and testing is quite important
Eclipse + coding/working with team
Good preparation of the lab is important.
This one was successful being so well prepared.

Overall organization, style, atmosphere:

		Median
1)	Are you satisfied with the product we developed?	**Mostly**
2)	Are you satisfied with your personal contribution to the product?	**In general, yes**
3)	Are you satisfied with the skills in new technologies you learned?	**Mostly**
4)	Are you satisfied with the agile software engineering skills you learned?	**Mostly**
5)	Software development is always risky as we mainly have to face unknown challenges and change is ubiquitous. The Planning Game, Stand Up Meetings, Burn Down Charts are meant to give the team a clear picture of the status and the ability to react as early as possible. This should give you the feeling that we are in control. Did you feel as being in control?	**In general, yes**

6) Comments on satisfaction?

The product is more interesting than ones you are normally deal with this lab. It is fun to play with the application you have contributed to.
Personally, I'm very satisfied. But I think, I could have contributed more if I had known/worked on the technologies beforehand
More focus on playable game would be nice
Yes, good. Only one thing: I would liked to work more on game improvements than research-driven stories

7) Comments about steering & control?

Problems that were pointed out during reflection meetings weren't improved so much as they could have been.
I had the feeling that control was lost. Before presentation, the team was integrating and fixing until 10 minutes before. There should have been more control.
Burndown chart gives you an overview of the project progress, but was a little confusing
I was really impressed by the lead of maturity and understanding with respect to steering and control
It is wonderful how it worked
Maybe we should have added some self-reflection on the biggest miss-estimations. What did go wrong? How can we avoid this in future?

		Median
8)	Did the responsible persons collaborate in a way that was helpful for you?	**Mostly**
9)	Were you able to follow the *words* of the English speaking teachers?	**Completely**
10)	Were you able to follow the *ideas* of the English speaking teachers?	**Mostly**
11)	Were the English speaking teachers able to understand you?	**Mostly**
12)	Taking into account that communicating in English meant some extra effort that otherwise could have been invested into learning programming skills, was it worth to participate in a course held in English?	**Mostly**
13)	If the same course at the same agile/technical level would be offered by local teachers or Ph.D. students, but taught in German, would you still recommend it to other computer science students?	**Completely**

14) Do you have any tips about how to ease communication? What was helpful for you?

The English of non-native speakers would improve much more if there was an English native speaker
Talk more during the stand-up meeting
Since there was no centeral system architect (in the participants) the central design was not completely understood by everyone in the same way. Hence I felt everyone had their own view of Gameparameter/Playerparameter. Mabe we can identify clean responsibilities for each "story" so that there is some control and common understanding of the overall design
Just use easy words or try to describe more, then everyone understands you
Having a bigger room and smaller chairs would help to communicate more easily with other pairs

Preparation

	Median
15) Did you feel prepared enough for programming in general?	**Mostly**
16) Did you know enough about Design Pattern, Refactoring, and Testing (software engineering Topics)?	**In general, yes**
17) Did you feel prepared enough with respect to the challenges of the used frameworks?	**In general, yes**
18) Did the frameworks (Glassfish, GWT, Eclipse) contribute to the (successful) product?	**Mostly**
19) Compared with simpler technologies: Was it worth the time to learn our frameworks at the cost of other techniques?	**Mostly**

In advance to the lab, we had a Saturday of preparation.

	Median
20) Was the introduction to Subversion useful?	**In general, yes**
21) Was the introduction to Design Patterns useful?	**More or less**
22) Was the introduction to Refactoring useful?	**In general, yes**

23) What was the most important thing you learned in this preparation day?

Communication to others, preparing an English presentation
General concepts of WSDL, XP, Subversion
A very broad overview of each topic
Getting to know the others
The most topics were very interesting and some topics of the SWT-lecture I learned in the preparation meeting

24) What else would you have liked to learn?

Glassfish/OsGi/Ditrios, Web Services, GWT or Ajax, Service Retrieval
More concrete information about the principles and practices of XP itself. E.g. not only how but also when to apply refactoring

	Median
25) Do you think it was helpful to let participants in advance prepare for a technical topic necessary for the course?	**In general, yes**
26) Do you think it was helpful to let participants in advance prepare for a topic in agile software engineering?	**Mostly**
27) How many days have you have been willing to invest into the preparation of your topic?	**3 days**
28) How many days in addition would you have been willing to invest in meetings with the other participants to get talks or tutorials from them?	**1 day**

1) Comments on the expected benefits and effort of a more intense preparation?

You may get started faster, clarify more the "expert" in roles to have a real spreading of knowledge
I think it was enough
If possible, it is better to have two preperation meetings instead of a large one
I would say that there should be a repetition of the first day challange before the lab starts so that there is enough time given for "self practical preperation". I always felt I was behind in catching up the practicals of the theory I learned.
Maybe 4 weeks is too short for all of this to learn
It was ok to get in touch, but I got not much knowledge
For me it was alright. For less experienced programmers it could help to look out at some sample architectures and code in more detail

Give us feedback to some of our ideas about improving the course:

		Median
2)	If we wanted to integrate more focused training of programming, test, design, refactoring skills, we had to find the time for it. To set a clear barrier against the implementation pressure of the iteration we could decide upfront that every day at about 1 hour before the end implementation stops and training starts. Would you recommend this?	**Rather no**
3)	Would you recommend having this training hour rather in the morning?	**In general, yes**
4)	Another way to encourage simple design and good code quality could be to require all/some code to be reviewed by the tutors in the first or second iteration. Would you recommend this?	**In general, yes**

5) Concrete suggestion how to organize this?

Was good organized. In my opinion, it is better to let this stuff to the students
Make a list of topics related with the project. Give some sample codes in this training hour
I liked the focus on self-responsibility and do not recommend such a school-like situation
Clear responsibilities for each story. So that we have complete overview Yes, it would have been nice to review the code at least once, but this could be done during the iteration - or at least, it should be suggested.
Perhaps make sure that test first is applied
After finishing each task, get in touch with a tutor and explain the solution to him

		Median
6)	What do you think about the number of participants?	**Quite okay**
7)	What do you think about the number of semesters you studied before participating in this course?	**Quite okay**

Is there something we should have made different? Do you have any tips for future teachers and students? Do you have any other final remark?

Was a super lab! The only remark is the room we programmed in, the wall in the middle was a little disturbing for communication between programmers. Maybe this was the reason why repairing was difficult.
Collaborative game make the development process more fun. XP is really good
I really enjoyed being pat of this team.
Make a bit more planning of the project in advance
Great lab
It is an excellent lab, especially be it is so well prepared. You got to keep it going!