Computer Science

Christian H. Becker

Using eXtreme Programming in a Student Environment: A Case Study

---

Master's Project

E2010:05

To Steffi and Emil

Using eXtreme Programming in a Student Environment: A Case Study

Christian H. Becker

This thesis is submitted in partial fulfilment of the requirements for the Masters degree in Computer Science. All material in this thesis which is not my own work has been identified and no material is included for which a degree has previously been conferred.

Christian H. Becker

Approved, 10th of June 2010

Advisor: Donald F. Ross

Examiner: Thijs Holleboom

Abstract

With the advent of shorter time to market of software products there an increasing requirement for techniques and methods to improve the productivity levels in software development together with a requirement for increased flexibility and the introduction of late changes. This in turn has lead to the introduction of a set of techniques known as "Agile methods" which include one methodology known as "eXtreme Programming". This is a collection of values, principles, and practices. Since these methods are becoming more common in industry, is has become more important to introduce these ideas in the undergraduate curriculum.

This case study analysed whether or not it is possible to teach eXtreme Programming at a university by means of a course that presents a mixture of theory and practice within eXtreme programming. In this context, a case study was carried out to determine which of the practices of eXtreme Programming are more appropriate to university projects. The case study indicates that it is worth investing the effort to teach eXtreme Programming to students to enable them to apply eXtreme Programming or at least some of its practices in future business and university projects.

Acknowledgements

I would like to thank my supervisor Donald F. Ross for his patience, support and continuous interest in improvement. I would like to thank my parents for giving me the opportunity of spending an unforgettable time at Karlstad University in Sweden.

Contents

List of figures

List of tables

1 Introduction

This project is the result of a case study as to whether or not the ideas of eXtreme Programming may be taught in the undergraduate curriculum. The increasing adoption of agile methods [80] and eXtreme Programming [20] in industry means that these methods should be introduced to students as part of their education. The question is how best to achieve this.

It is a point of discussion as to how eXtreme Programming may be taught to students in a University. This thesis addresses this question by analyzing a student lab exercise for the Software Engineering course in the Department of Computer Science at Karlstad University by means of a case study. The main goal of this case study was to measure certain (15) eXtreme Programming practices realizable in a student environment [appendix G] by means of surveys, analysis of archival information from programming tools and direct observations of the students. However, this thesis aims further by analyzing the success or failure in teaching the 15 eXtreme Programming practices and in giving ideas to improve this process.

This thesis consists of 4 chapters. Chapter 2 presents the background information such as the external circumstances of the course, basics about evaluative research and case studies as well as eXtreme Programming. Thereupon, chapter 3 defines the analysis model used in the empirical study in chapter 4 and introduces the items set up for the study and allocates them to the different data sources. Chapter 4 shows the students' responses to the surveys, evaluates all data sources and combines these results for an interpretation of this learning process. Finally, chapter 5 compares these results with the 15 eXtreme Programming practices and to related work in order to give recommendations for future courses in this area.

2 Background

2.1 Introduction

This chapter introduces topics and methods that are required to read and understand this thesis.

In the first section the students' background on programming skills is introduced. This is followed by an introduction on evaluative research. The tools from the toolbox of evaluative research which have been applied in this project are introduced. The procedure model used is then presented. To complete the procedural overview the next section is about case studies. The case study methods used in this dissertation are briefly introduced. Furthermore it is explained how the evaluation tools fit into the case study research methodology. The last section of this chapter introduces eXtreme Programming to the reader. The reader is required to know some key facts about eXtreme Programming to be able to understand the analysis of the course performed in this dissertation. Finally as a complement to the first section of his chapter follows an introduction to how eXtreme Programming is taught at Karlstad University and what the level of the students' knowledge about eXtreme Programming was before they attended the course.

2.2 Students Programming Background

The programming background of those students who attended the course which was evaluated in this dissertation is introduced. This information is given in the form of the programming courses that the students had already taken. The curriculum of the students' first two years is presented in Figure 2.1. This curriculum is of interest since the course evaluated in this project is a second year course.

Årskurs 1

Hösttermin		Vårtermin	
DAV A07 Programmeringsteknik 5 p	DAV A02 Prog.-utv.-metodik 5 p	DAV A14 Datorsystemteknik 5 p	DAV B01 Operativsystem 5 p
MAA 110 Analys EA1 5 p	MAA 107 Algebra A 5 p	MAA B05 Diskret matematik 5 p	TEL 143 Digitalteknik 5 p

Årskurs 2

Hösttermin			Vårtermin		
TEL 242	**Ingenjörsvetenskapens grunder**	**15 p**	**CIT B01**	**Tillämpad systemkonstruktion**	**15 p**
	Elektronik	4 p	DAV B02	Datakommunikation	3 p
	Mekanik	2,5 p	DAV B04	Databasteknik	3 p
	Optik	2,5 p		Programkonstruktion	2 p
	Fysik Lab (Optik)	1 p		Statistik	3 p
	Matematik (Linjär algebra + Analys EA2, B1,B2)	5 p		Företagsekonomi	3 p
				Interkultur	1 p
TEL 243 Projekt i ingenjörsvetenskapens grunder 5 p			CIT B02 Projekt i tillämpad systemkonstruktion 5 p		

Figure 2.1 The 2004 computer engineering curriculum at the University of Karlstad

All courses on programming are indentified by an orange rectangle. As Figure 2.1 shows the students had 4 courses on programming before they attended the course on eXtreme Programming (CIT B02 red rectangle on the bottom right of Figure 2.1). From this, the students may be classified as intermediate programmers In addition to their current knowledge, the students were required to learn an entire new infrastructure (tools the students have not used before) as well as the ideas behind eXtreme Programming.

2.3 Evaluative Research

The main methods used in this dissertation are taken from the area of evaluative research. According to Bortz & Döring [24], Rossi & Freeman [34], Weiss [35], Wittmann [36] and Thierau & Wottawa [37] evaluative research is a variation on empirical research methods, applied to a particular group of questions. Modern evaluative research has its roots in the 1930s in the USA where it was applied to analyze programs, interventions and arrangements in the health care and the education system[1].

Since evaluative research is a very wide topic, this section is only about those parts, used in the research for this thesis. During this research, some tools out of the toolbox of evaluative research have been used. These tools were:

a) Survey; two surveys that have been filled in by the student group.
b) Observation; an observation on the student group.
c) Analysis of log files; The log files of the tools the students used in their project

Due to the fact, that these tools have the purpose to gather data they can be understood – following Yin [23] - as data sources or, as Yin [23] calls them sources of evidence. The strengths and weaknesses of the different data sources, according to Yin [23], can be found in Figure 2.2 In the following sections all applied data sources are explained in detail.

[1] For more information regarding the evolution of the evaluation in the USA see Mertens [38]. For information regarding the evaluative research in Europe see Leeuw [39] and for information regarding evaluative research in Germany see Stockmann [40]

Source of Evidence	Strengths	Weaknesses
Documentation	• stable—can be reviewed repeatedly • unobtrusive—not created as a result of the case study • exact—contains exact names, references, and details of an event • broad coverage—long span of time, many events, and many settings	• retrievability—can be low • biased selectivity, if collection is incomplete • reporting bias—reflects (unknown) bias of author • access—may be deliberately blocked
Archival Records	• *[Same as above for documentation]* • precise and quantitative	• *[Same as above for documentation]* • accessibility due to privacy reasons
Interviews	• targeted—focuses directly on case study topic • insightful—provides perceived causal inferences	• bias due to poorly constructed questions • response bias • inaccuracies due to poor recall • reflexivity—interviewee gives what interviewer wants to hear
Direct Observations	• reality—covers events in real time • contextual—covers context of event	• time-consuming • selectivity—unless broad coverage • reflexivity—event may proceed differently because it is being observed • cost—hours needed by human observers
Participant-Observation	• *[Same as above for direct observations]* • insightful into interpersonal behavior and motives	• *[Same as above for direct observations]* • bias due to investigator's manipulation of events
Physical Artifacts	• insightful into cultural features • insightful into technical operations	• selectivity • availability

Figure 2.2 Six sources of evidence: Strengths and Weaknesses taken from Yin [24]

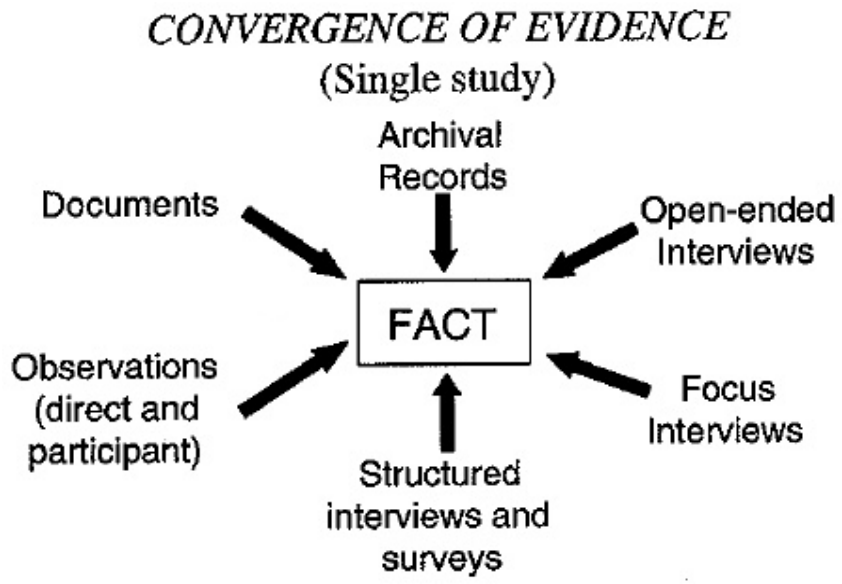

Figure 2.3 Convergence of Evidence taken from Yin [24]

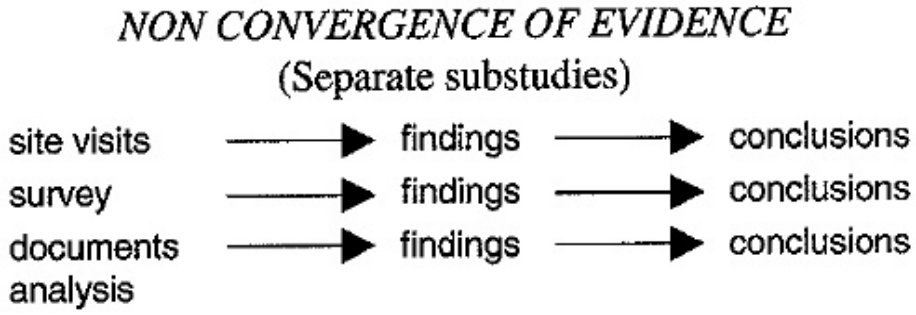

Figure 2.4 Non Convergence of Evidence taken from Yin [24]

Figure 2.3 & Figure 2.4 show that, according to Yin [23], there exist 2 different ways of interpreting data sources that have been analysed during research. The first one – in this case convergence of evidence – builds the evidence based on different data sources whereas the second one derives a conclusion from every single data source. In this thesis, the way described in Figure 2.3 has been applied. All sources of evidence – described in the next sub sections – are used to find one fact about an item. The detailed procedure can be found in chapter 3 where the analysis model is introduced in detail.

2.3.1 Surveys and Interviews

The Survey in this dissertation is regarded as a special kind of interview. An advantage of the survey is that surveys are performed without interviewer control. This can be understood as an advantage since any influence by the interviewer can be minimised. Another relevant

advantage, in the context of this thesis, is that it is an economical way to perform an interview. All disadvantages mentioned in Bortz & Döring [24] may be neglected. These disadvantages are:

- Influence by the appearance of the interviewer (who)
- Influence due to different point of time of the interview of each respondents (when)
- Influence by different surroundings of the interview of each and every respondent (where)

These disadvantages may appear if the respondents do not answer the survey under the same circumstances. This means the respondents have to answer the survey at the same time in the same room to avoid influence from other respondents and other outside influences.

The survey as a method has been chosen since surveys are meant to gather data about the students' state of mind regarding their knowledge about eXtreme Programming, its practices and their interest in applying eXtreme programming during the students' practical project. Details about how the surveys have been set up can be found later in this document in Chapter 3 where the whole analysis model is introduced.

2.3.2 Observation

In this research the observation method was chosen as instrument with the goal of evaluating whether the students did apply a certain practice or not. This means - according to Bortz & Döring [24] the qualitative approach has been chosen. This corresponds to the following criteria presented in Adler & Adler [41]

- Observation in the natural surrounding (in this case the classroom)
- Active interaction of the observer
- Focusing on larger units, systems and behaviour patterns instead of focusing on small variables
- Open for new input and not focused on an observation schema

The last point might sometimes be hard to follow since the observers in this project had to fill in an observation template which focused on the 15 practices used in the project. More about these practices can be found in the section 2.5. Regarding the last point it needs to be mentioned that not only the fixed observation schema is evaluated since the observers wrote a small diary for every day and analysed this diary with focus on the questions, defined in chapter 3 where the whole analysis model is introduced.

2.3.3 Tools

Tools are source of evidence for every case study. According to Yin [23] the tools themselves are not to be seen as source of evidence. The tools deliver artefacts – in this case called archival records – that can be seen as evidence for the evaluation. According to Yin [23] examples for archival records include:

- *"Service records, such as those showing the number of clients served over a given period of time*
- *Organizational records, such as organizational charts and budgets over a period of time*
- *Map sand charts of the geographical characteristics or layouts of a place*
- *Lists of names and other relevant items"*

Yin [23] states that archival records do show one criterion that is relevant for case studies, namely that the archival records will never be similar if you run a project twice.

2.4 Case Study

Numerous definitions of a case study may be found but for this research the definition according to Yin [23] has been applied. According to Yin [23] a case study is one of several methods developed in social science research. Another definition according to Merriam [6] is that a case study is an "investigation of a specific occurrence, e.g. a program, an event, a person, an institution or a social group." This section is giving an overview over the history of case studies, what a case study can be, what different characteristics a case study can have.

2.4.1 History of Case Studies

Case study research is historically marked by periods of intense use and periods of disuse. According to Tellis [7] the earliest use of this form of research can be traced to Europe, predominantly to France. The fields of sociology and anthropology are credited with the primary shaping of the case study concept as we know it today (see Barnes, Conrad, Demont-Heinrich, Graziano, Kowalski, Neufeld, Zamora, and Palmquist [9]). However; case study research has drawn from a number of other areas as well, such as:

- the clinical methods of doctors
- the casework technique being developed by social workers
- the methods of historians and anthropologists
- the qualitative descriptions provided by quantitative researchers

- the techniques of newspaper reporters and novelists.

2.4.2 Description of Case Studies

Apart from the definition at the beginning of this chapter, a case study can also be described in terms of its special properties. The properties that are relevant to this research project are, taken from Merriam [6]:

- Particularistic: Means that a case study "focuses on a certain situation, event, occurrence or person." This reflects the definition of case study mentioned earlier.
- Heuristic: Means that it "can improve the readers understanding of the occurrence that is being studied"

In contrast to other designs of experiments, surveys and history research, case studies do not have any special methods for collecting or analyzing information (data). All types of methods for collecting scientific data, from observation to interview and survey, can be used in a case study.

2.4.3 Qualitative versus Quantitative Methods

According to Barnes et al. [9] qualitative and quantitative methods are usually used in conjunction with each other. Typically qualitative data uses words and pictures while quantitative data uses numbers to describe what the researcher has extracted from what the researcher studied. Another major difference between the 2 is that qualitative research is inductive, i.e. it evolves abstractions, concepts, hypotheses and theories, while quantitative research is deductive, i.e. reaches to test already existing theories. In qualitative research, a hypothesis is not needed to begin research. However (according to Barnes et al., [9]), "all quantitative research requires a hypothesis before research can begin."

2.4.4 The use of Case Study in the Experiment

It was decided to perform this experiment as a case study since this research did not require control of behavioural events but focused on contemporary events. These characteristics are, according to Yin [23], typical for a case study as research method. As can be seen in section 2.4.3, the boundaries between qualitative and quantitative case studies are clear, and in this experiment both will be used.

The following data gathering methods have been used:

- Observation is used here as a qualitative method. Due to the typical characteristics of observation this makes the case study particularistic and heuristic. Particularistic due to the investigation of the students group and heuristic due to the analysis of the students' understanding and acceptance of the eXtreme Programming practices. More details about the definition of particularistic and heuristic can be found in section 2.4.2
- Surveys: the survey is used here as a quantitative research method. It asks the same questions of the different people in a group and delivers a variety of impressions under the same circumstance. For instance if you ask a group of people using a 5 category response scale (++/+/0/-/--) the same question, the answers are still liable to be different based on individual impressions.
- Archival records are used here as a qualitative research method. This is a analysis of historical data that has been collected during the experiment. It is, according to Bortz & Döring [24], very important to exclude interpretations as much as possible, to ensure that only hard facts have been taken into consideration.

2.5 eXtreme Programming in a nutshell

The purpose of this section is to present a quick introduction of eXtreme Programming to the reader. This starts with the history of the origin of eXtreme Programming, followed by an introduction on the basic idea behind eXtreme Programming. There then follows a description which shows the relevance of eXtreme Programming to student programming and industry applications

The section is completed by a short walk through the points of interest of eXtreme Programming to this research.

2.5.1 An overview of eXtreme Programming

eXtreme Programming is a lightweight programming methodology that, according to [75], supports a team by writing software that contains fewer bugs and therefore takes shorter time. This methodology consists of several values, principles and practices to be considered. Some of these must be applied by the whole company, some by the team and most by each and every member of the team. The 2 essential aspects of eXtreme Programming are

1. short term and very specific programming goals
2. an awareness of the social aspects of programming

The first edition of Beck [1] was published in October 1999. In effect, eXtreme Programming originated in this first edition. Over the years eXtreme Programming has been applied by a number of programming teams and several books have been written on the subject. Many companies (see [20]) applied the process given by Beck and realized that in their experience eXtreme Programming is a very good and efficient way to create software. eXtreme Programming is a process and has a number of practices that are used to reach a project goal. The eXtreme Programming process also includes a philosophy of software development which is based on 5 values, namely; communication, feedback, simplicity, courage and respect. To integrate these values into eXtreme Programming, Beck received help from a psychologist, Cynthia Andres. This was in order to show that eXtreme Programming is not only a new style of programming or a new idea, but also a process that deals with the social behaviour of the whole team and more or less of the whole company. Beside the 5 values, eXtreme Programming uses 24 practices and 13 principles. A detailed explanation of these values, principles and practices can be found in Beck [2]. In the second edition of eXtreme Programming Beck [2] published in 2004 some minor and some major modifications in the process. One major modification was that the developing team should select the practices they really need for the project and exclude the remaining practices. This approach was new, since in the first edition of eXtreme Programming Beck [1] Beck's opinion was that the team has to apply all practices. To offer the practices in a different way Beck divided the practices into primary and corollary practices where the primary practices are independent of the corollary practices. The corollary practices are, according to Beck, difficult to use without first mastering the primary practices. This might be caused by the enormous change of behaviour the corollary practices require from the whole team and in some cases even from the whole company. Beck is not trying to give one defined way to apply eXtreme Programming into a team; rather he wants to give some directives to the team from which the developer is able to take what he requires to become a better team member and developer.

2.5.2 Applying eXtreme Programming in a Student Project Environment

This research shall bring up whether eXtreme Programming can be taught in a project like the applied system construction project or not. Therefore it needs to be analysed if the

teaching that took place in advance to the project was sufficient so that the students felt prepared enough to apply certain practices of eXtreme Programming during the project.

The teaching took place in a course called “Tillampad systemkonstruktion” (Eng. Applied System Construction) where the students were taught the basic ideas of eXtreme Programming. As can be found in Figure 2.1 this course took place exactly before the applied system construction project was started. For further information about the applied system construction project see Figure 2.1

Table 2-1 and Table 2-2 all eXtreme Programming practices are listed. All these practices where taught at the projects preparation course. The students have been asked to apply only the practices that show a “yes” in the column named applied. That seems not to be a problem since Beck says in [2] that there is no need to apply all given practices. The decision what practice the students were asked to apply and what practices the students were not asked to apply was taken by the teachers of the course. The teachers justified their decision by the circumstances given, by a simulated project.

#	Practice Name	Applied	Aspect
1	Sit Together	Yes	2
2	Whole Team	No	2
3	Informative Workspace	Yes	2
4	Energized Work	Yes	2
5	Pair Programming	Yes	1
6	Stories	Yes	2
7	Weekly Cycle	Yes	1
8	Quarterly Cycle	No	1
9	Slack	Yes	1 & 2
10	Ten-Minute Build	Yes	2
11	Continuous Integration	Yes	2
12	Test-First Programming	Yes	2
13	Incremental Design	Yes	2

Table 2-1 Primary practices, the aspect of the practice and whether applied or not

The numbers in the column called “Aspect” are to be understood as follows:

1. short term and very specific programming goals
2. an awareness of the social aspects of programming

#	Practice Name	Applied	Aspect
1	Real Customer Involvement	No	2
2	Incremental Deployment	No	1
3	Team Continuity	No	2
4	Shrinking Teams	No	2
5	Root-Cause Analysis	No	2
6	Shared Code	Yes	1 & 2
7	Code & Test	Yes	2
8	Single Code Base	Yes	2
9	Daily Deployment	No	1 & 2
10	Negotiated Scope Contract	Yes	2
11	Pay-Per-Use	No	Non

Table 2-2 Corollary practices, the aspect of the practice and whether applied or not

The last question to think about is why might eXtreme Programming be interesting to students? The authors opinion regarding this question is that there are 2 answers. The first is to improve the quality of the students' programming. Secondly, there is another reason why attending such courses like the one about eXtreme Programming is interesting to students. It is the need within industry for employees that are skilled with new methodologies and philosophies. Usually it takes some years until a new software development philosophy enters companies. This is caused by some kind of approval process so that the companies are able to filter the useful innovations from non-useful methods. Whenever a company decides to start working following new principles the whole company spirit needs to be adapted and with this all business processes need to be changed. This is very expensive and it becomes even more expensive when it comes to the employee training. All this is much cheaper and easier when a company is able to speed this process up by bringing in new employees who already know the new methodology and have some relevant experience. So the new employees become experienced employees and are able to help the company not to make mistakes which may occur while a methodology like eXtreme Programming is introduced.

2.6 Review

The purpose of chapter 2 was to let the reader aware of the necessary background to the methods and principles of the research and of what has been researched. Transferring this declaration to the concrete thesis the reader should now know the relevant ideas of:

- The students background on programming
- Evaluative research
- The tools (survey, observation, archival records) that have been applied in the research
- Case Studies
- Qualitative and quantitative methods
- eXtreme Programming
- The students background on eXtreme Programming

3 The analysis model

3.1 Introduction

The student project course was as follows. A lecture on eXtreme Programming was held in the first place. The author attended this lecture to receive an impression about the content of the student project course. Following this lecture, Survey 1 was filled in by the student group. This first survey represents the first data source gathered for evaluating the student project course. After the survey was filled in and in correlation to the lectures, the student project started. The student project lasted 5 weeks and the students' only task was to apply eXtreme Programming. During these 5 weeks a permanent observation was carried out and recorded as a protocol. This protocol represents the second data source analyzed for the evaluation of the course.

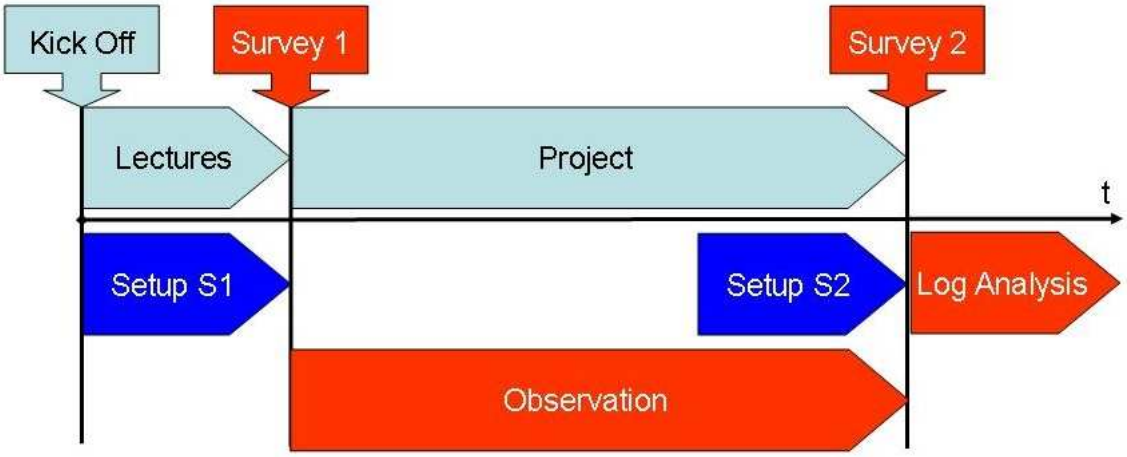

Figure 3.1 Project timeline and action events

At the end of the student project, the students had to fill in the second survey. For the evaluation, the analysis phase started. The second survey represents the third data source that exposed the students' view on the observation performed by the author. The last phase in this dissertation study was the analysis of the log files generated by the tools used by the students during the student project. These log files serve as last data source. Figure 3.1 illustrates the

data sources (orange items: surveys 1 and 2, the observations and the log analysis) and other relevant parts on the timeline.

In Chapter 3 the data sources are explained in detail with special attention to their content and the significance. Furthermore, chapter 3 highlights how the data sources are evaluated and how the results may be interpreted. In addition, chapter 3 introduces the hypothesis that supports the evaluation of the course. The data analysis is presented in chapter 4.

3.2 Hypotheses

The goal of the thesis is too great to be proven by one data source. As a result of this aspect small milestones have been developed to prove small parts and at the end put together to create a high level view. According to that assumption, for each main data source, a hypothesis has been derived. These hypotheses are proven by not proving the corresponding null hypotheses. The following hypotheses and their null hypotheses have been set up:

(1) Survey 1: Hypothesis H_{S1}

Hypothesis H_{S11}: A student group that has attended the theoretical sessions on eXtreme Programming understands the practices well.

Null hypothesis H_{S10}: A student group that has attended the theoretical sessions in eXtreme Programming does not understand the practices well.

This hypothesis reflects one of the key issues in Survey 1 and is tested in the evaluation of Survey 1.

(2) Observation: Hypothesis H_{O1}

Hypothesis H_{O1}: The attitude a student has towards an eXtreme Programming practice has an influence on how eXtreme Programming will be applied during the practical project.

Null hypothesis H_{O0}: The attitude a student has towards an eXtreme Programming practice has no influence on how it will be applied during the practical project.

This hypothesis – derived from the content of the project – is tested by evaluating the observation as it is possible to see how a student applied the practices.

(3) Survey 2: Hypothesis H_{S21}

Hypothesis H_{S21}: The student groups' self evaluation about the level of application of an eXtreme Programming practice will match the groups' real behaviour.

Null hypothesis H_{S20}: The student groups' self evaluation about the level of application of an eXtreme Programming practice will not match the groups' real behaviour.

This hypothesis proves whether the students' self evaluation matches their observed behaviour or not. Survey 2 serves in testing this last hypothesis.

3.3 Survey 1

3.3.1 Introduction

Survey 1 collects information about the students' standard of knowledge regarding the 15 eXtreme Programming practices after the students had attended the theoretical sessions. It asks the students to self verify their knowledge about eXtreme Programming. This verification has to be done in the 5 categories explained later. Survey 1 also asks the students how well they think the certain practice is applicable in the student project and if the students are willing to embrace eXtreme Programming in this project. These questions aim to evaluate the students' attitude towards eXtreme Programming and its practices. Moreover, Survey 1 takes a snapshot of the students' attitude towards the structure of the course. The question regarding the structure of the course is repeated in Survey 2 to see whether the students' attitude towards the structure of the course changed after performing the student project.

3.3.2 Terms of reference for Survey 1

Since eXtreme Programming is a process that needs to be experienced by the whole project team. The focus has been moved from the individual to the group. Keeping that in mind, Survey 1 has been planned and executed on the assumption that the group will be analyzed as a whole. For that reason, the group response to each of the questions is taken to be the weighted mean [81] of the individual student responses.

In the survey, there are 2 types of response scales. The first one has 5 respective labels: Very Well (1); Well (2); Neutral (3); Not Well (4); Not Well At All (5). The scale is based on the Likert-Scale [31], [82] were the most equidistant style according to Rohrmann [30] and Wyatt & Meyers [29] is chosen.

For the first scale, the 5 point scale, the weights for each category are given in parentheses - Very Well (1); Well (2); Neutral (3); Not Well (4); Not Well At All (5). For question 1 of survey 1 this leads to a weighted mean value of 2.3. I.e. ((1*0) + (2*7) + (3*3) + (4*0) + (5*0))/10 where 10 is the number of responses. This value is interpreted as being a single point value for the student group response. These values for the group response may then be

interpreted within the 5 categories by defining a range for each category as shown in Table 3-1

Category	Very Well	Well	Neutral	Not Well	Not well at all
Value of category	1	2	3	4	5
Range of category	1 – 1.5	1.6 – 2.5	2.6 - 3.5	3.6 – 4.5	4.6 -5
Example value		2.3			

Table 3-1 Weighted Average Example

The group response to the question "How well did you understand all practices" was thus "Well". Note that the lower the value of the weighted average, the better the response.

The second response scale consists of the options Yes (1) and No (2). This is according to Grosse & Wright [32] objectively analyzable if the answer is in the categories Yes or No. This approach has been chosen according to Bortz & Döring [24], to force the students to make an either or decision.

Survey 1 was written in English together with a native English speaker and a native Swedish speaker in order to minimize the possibility of misunderstandings caused by differences in culture and language. Therefore, no language or cultural support like Haeberlin [28] mentioned in Bortz & Döring [24] was required.

3.3.3 Items of Survey 1

The items listed in this section and their purpose is presented in order to be able to explain the relationship of the items to the experiment as a whole. The word item is chosen since some items consist of several questions. The aim of the survey was to get a snapshot of the of the student's self-evaluation about their awareness of eXtreme Programming at the start of the student project. The students' attitude towards this project is very important to its success. The survey contains questions to prove or not to prove the hypothesis. Question 33 in Survey 1 is repeated in Survey 2 (as Question 25) to see whether or not the students' point of view changed during the course of the project. The survey consisted of 32 questions which have been consolidated into the following 5 items.

Item 1 How well did you understand all practices?

The formula to calculate the mean ($\overline{X}$) is $\overline{X} = \frac{1}{n} * \sum_{i=1}^{n} x_i$. As shown above, the value is 2.3 indicating that the group thought that it understood the practice well. This value may be used as an indicator together with the mean of the weighted means in item 3 i.e. the mean value of the group understanding for all the practices.

Item 2 How well do you think the practice "XY" is applicable during the project?

This question has been asked each time for each of the 15 practices. For the sake of brevity, it is just listed once. This item supports the process of proving or not proving hypothesis (**H_{O1}**) as explained in section 3.5. This item directly asks the students how well they think a certain practice is applicable in the practical project. It has been asked since it can be checked against the observation that shows if the students applied the practice or not. This item shows whether some students thought that a certain practice was not applicable and then changed their point of view because the group as a whole applied it.

Three categories have been created, namely “Applicable”, “Undecided” and “Not Applicable” to be able to evaluate this item properly. The ranges for the categories are defined as follows:

A) The group of students who answered this question by “Well” or better (response ≤ 2.5) thinks that practice XY is applicable in the practical project.

B) The group of students who answered this question by “Not Well” or worse (response ≥ 3.5) thinks that practice XY is not applicable in the practical project.

C) The group of students who answered this question as neither category A nor B is considered to be “undecided” (2.5 < response < 3.5).

Item 3 How well did you understand the practice "XY"?

This question has been asked for each of the 15 practices, and may be used to show the group response for each practice. The group responses may then be averaged and used as an indicator of how well the group understood all of the 15 practices. Item 1 also gives such an indicator.

It is interesting to see whether the mean of the mean of all questions ($\overline{X_{item3}}$) in item 3 matches[1] the mean value ($\overline{X}$) of item 1. The formula to calculate the mean ($\overline{X_{item3}}$) is as follows: $\overline{X_{item3}} = \frac{1}{n} * \sum_{i=1}^{n} (\frac{1}{m} * \sum_{j=1}^{m} x_m)_n$ where *m* = answers within one question (team) and *n* = the 15 practices;

This item is relevant for 3 reasons:

1. The first proof is the internal check of Survey 1 described in item 1.
2. The second proof is the proof of hypothesis **H_{S11}** against the response of this item by disproving the null hypothesis. The null hypothesis **H_{S10}** can be seen as unproved if no responses are in categories that are defined as "Not Well". This is numerically expressed as response ≥ 3.5.
3. The third and last proof is to check this response together with the response of item 2 to see whether a relation exists or not.

A scale according to the scale mentioned in item 2 has been created by combining the previous categories given in Table 3-1 to be able to measure this item properly. The scale categories are named as "Understand", "Undecided" and "Not Understand". The category ranges are defined as follows:

Category	Very Well	Well	Neutral	Not Well	Not well at all
Value of category	1	2	3	4	5
Range of category	1 – 1.5	1.6 – 2.5	2.6 - 3.5	3.6 – 4.5	4.6 -5
Scale	Understand		Undecided	Not Understand	
Range of scale	1-2.5		2.6 - 3.5	3.6-5	

Table 3-2 Three Category Scale

A) A Student group that answers this question by "Well" or better (response ≤ 2.5) understood practice XY.

B) A Student group that answers this question by "Not Well" or worth (response ≥ 3.6) not understood practice XY.

[1] A response can be understood as matched when the values are in the same category. Since the categories are from 1 to 5 it must be 1 – 1.5; 1.6 – 2.5; 2.6 - 3.5; 3.6 – 4.5; 4.6 -5.

C) A Student group that answers that question neither as mentioned neither in A nor in B is undecided (2.5 < response <= 3.5).

Item 4 Have you decided to embrace XP in the project?

This item is the only item in the first survey that has a response scale with 2 labels: Yes (1) and No (2). This approach has been chosen since the students should be forced to decide if they embrace eXtreme Programming or not without the possibility of a neutral category. The response to this item will show - when calculating the mean value - if more than 75 percent of the student group are willing to embrace eXtreme Programming in the project.

Item 5 How do you like the idea of having theoretical sessions on XP followed by performing the theory into practical use?

This item is asked in Survey 1 and Survey 2. This item allows the evaluation if the students' opinion towards the structure of the course has changed while they performed the course or not. The evaluation model can be found in section 3.6.

3.3.4 Summary

Finally, Survey 1 can be summarized as:

- Survey 1 indicates if the students understood all practices.
- Survey 1 indicates if the students think a practice is applicable well or not applicable.
- Survey 1 indicates if the students are willing to embrace eXtreme Programming in the project or not.
- Survey 1 indicates – together with Survey 2 - proof if the students think that the structure of the course is good or not.

3.4 The tools

This section explains how the log files are evaluated. The evaluation of the log files supports the evaluation done for the protocol, i.e. the record of the observations performed during the 5 weeks of the experiment. According to Yin [23] a log file is an Archival Record which is a very important source of evidence for a case study. The log files are recorded by the tools used by the students while the students performed the practical project. These files

are a reliable data source since there is no way of manipulating them. The only log files that are analyzed are the files recorded from the tool CVS[1] which is a maintenance tool for source code and other files. As mentioned in Table 3-3 the practices with need to be checked by the log file of CVS[1] are:

(1) **Continuous Integration**: this practice requires that when a code section has been completed the code should be immediately added to the code base so that the automatic tests suites and builds are applied. Every time the code base has been changed it is recommended to immediately run the code and test it against the given test suites. Special Continuous Integration servers exist that run tests every time the code base has been changed. However, such a server was not available for the students and would have probably been beyond the scope of this project. Nevertheless, the testing can be performed manually as well so that the log files provide the information about whether the code has been checked into the base. Only observation can show whether the tests have been performed or not. This practice is measured by the growth of the file count and average file size over the duration of the project. This part of the practice is counted as valid if the values of the file count and average file size exhibit continuous increase during the project.

(2) **Shared Code**: this practice is about whether every member of the project group feels responsible for the quality of the whole code base or not. This can be shown with the log by checking whether different students worked on the same files or not and can be measured by 2 indicators. The first one is if every student used[2] CVS[1]. The second one is if all students actually added and modified transactions.

(3) **Code & Test**: This practice means that the project team has no other documents than those documents which may be generated from code and the tests. Since the students were forced by the specification of the project to maintain a web page that informs the lecturers about the status of the project they were not able to apply this practice to its full extend. In order to measure this practice, StatCVS[3] offers the possibility of checking whether other files than code and test files existed or not.

[1] Concurrent Version System - CVS is an open source tool to manage source code. For the file history CVS uses the same structure as Revision Control System - RCS which is an successor of Source Code Control System – SCCS. For more details about these tools see: [5], [15], [16] & [17]

[2] Used in this context can be understood in the sense that the student has transactions on the log file.

[3] StatCVS is a tool that is used for statistical analysis of CVS repositories. For more details see [18]

(4) **Single Code Base**: this practice is not really applicable in a project of this extent with only one product and one customer but CVS offers the possibility to evaluate whether the students had different repositories or not.

Finally the purpose of the Tools can be summarized as:

- The Tools an indicator as to whether certain eXtreme Programming practices have been applied in the practical project or not.
- The Tools represents a very resilient data source that delivers information which helps to give an overall impression of the project.

3.5 The observation protocol

3.5.1 Introduction

The observation is one of the most relevant and interesting data sources available in this evaluation. The reader should note that there was a possibility that the students observed felt influenced by the observers. While setting up the observation plan, the definition according to Laatz [25] has been taken into account. This definition of Laatz [25] clarifies the difference between an observation in the common understanding and an observation with scientific background. Laatz [25] says that an observation from a scientific point of view is much more goal focused, and controlled and this is reached via tools that guarantee the self reflection and systematic and helps to widen the borders of our perception.

Caused by timeline – five forty hour weeks - and in lack of observers – 2 observers for the whole period – it was not possible to apply all protocol rules described in Faßnacht [27]. The 11 rules for the content of the protocol, listed by Faßnacht [27] are:

1. Observe the behaviour in relation to the situation
2. Describe all circumstances in a situation
3. Avoid interpretations
4. Report how the subject has done anything
5. Report who is interacting with the subject
6. Report the complete order of events
7. Write always from a positive perspective
8. Report the surrounding to its full extent
9. Never sum up more than one action of the subject in one sentence

10. Never sum up more than one action of persons interacting with the subject in one sentence
11. Never report while using timestamps

These rules have been defined for the observation of smaller events. In this case the protocol would have exceeded an analysable size and the project would not have been realizable. Therefore, the following rules found in Ingenkamp, Parey & Tent [26] have been applied:

- **Selection**: in this case means that some items are defined that the observers take care of.
- **Abstraction**: this rule means that the observer has to focus on the event and not to interpret side events e.g. while asking question A the face might give indication about the expected answer.
- **Classification**: this means that the selected and abstracted events need to be classified.
- **Systematization**: this rule says that the classified events need to be taken into one big high level protocol that supports the analysis and makes the observation measurable.
- **Relativization**: this point says that the observers need to be able to separate events and classify them to their relative background. This means for instance unexpected situations that influence the observed situation.

An explanation is given on the following pages as to how the observation protocol was set up and which criteria the focus was on.

3.5.2 Terms of the observation

Since this evaluation measures how many of and whether the practices have been applied, the observation has been set up to take this into consideration. According to this structure, the protocol represents a list of all evaluated practices. Table 3-3 is an enriched copy of Table 2-1 and Table 2-2 and shows which practices have been applied, how they have been evaluated and how the impact on the influence of the observers is assumed to be. This impact

assumption has 3 categories: low[1], medium[2] and high[3]. In some rows "Tool" is listed as evaluation technique. More about these practices is mentioned in section 3.4.

#	Practice Name	Used	Automated	Possible Techniques	Influence
1	Sit Together	Yes	No	Observation	Low
2	Informative Workspace	Yes	No	Observation	Low
3	Energized Work	Yes	No	Observation	High
4	Pair Programming	Yes	No	Observation	Medium
5	Stories	Yes	No	Observation	High
6	Weekly Cycle	Yes	Partial	Observation	Low
7	Slack	Yes	No	Observation	Low
8	Ten-Minute Build	Yes	No	Observation	Low
9	Continuous Integration	Yes	Yes	Tool & Observation	Low
10	Test-First Programming	Yes	No	Observation	High
11	Incremental Design	Yes	No	Observation	Low
12	Shared Code	Yes	Yes	Tool	Low
13	Code & Test	Yes	Partial	Tool & Observation	Low
14	Single Code Base	Yes	Partial	Tool & Observation	Low
15	Negotiated Scope Contract	Yes	No	Observation	None

Table 3-3 Evaluated practices and their evaluation method

To be able to measure the practices and to finally prove the hypothesis **H_{01}** the observation results are mapped to numerical values. The 3 possible results for the observation and their numerical equivalent (in brackets) are Yes (1), Partial (0.9 to (-0.9)) and No (-1). The category "Partial" is spread from + 0.9 to – 0.9 since it can have a tendency to Yes or to No.

The events that caused the decision to partial are measured, categorized and expressed in numerical values to be able to calculate an appropriate value for partially applied practices. These values are added or removed from the Yes (1) or No (-1) value.

[1] Low – Nearly no impact. The students notice that the observers are writing a protocol but their behaviour will not change.

[2] Medium – No great impact. The students might think about applying the practice since the observers' presence might remind the students about the task to apply this practice.

[3] High – Great influence on the students behaviour. The students categorize the observers as not belonging to the group and by this as a control instance. Therefore, the risk of a changed behaviour is high.

3.5.3 The measurement of each observed practice

(1) Sit Together

An attendance check list showing students' attendance or absence has been used to measure this practice. This check list enabled the observers to determine whether the students did work / sit together or not / separately. The observers had to trust that the students calling in sick or absent actually had a legitimate reason for their absence and were not working on their own at home or at another place. The observers also recorded if there were features in the room the students were sitting in that made communication less efficient, for instance plants blocking the view so that the students could not see each other or other people that were not involved in the project disturbing the students working on the project. Since this information was needed on a daily basis, it has been measured several times each day.

(2) Informative Workspace

Since a scale is needed to measure how informative the student workspace was, 4 points according to Beck's definition (see Beck [2]) of Informative Workspace were chosen. These points are:

- Do the students have story cards?
- Do the students have a story wall?
- Do the students have access to water and snacks?
- Do the students have a workspace that is clean and organized?

Since these 4 points could change from day to day they have been measured on a daily basis.

(3) Energized Work

To measure this practice a table was used, storing the average working hours of the students per week and the average working hours of the whole student group per week. This table was recorded for the whole 5 week period and then was evaluated at the end of the student project.

(4) Pair Programming

By using a drawing of the room and the workplaces the observers could easily mark where the students were working from day to day. While the students were working, the observers were not supposed to check what they were working on. The observers just observed whether they were working in pairs or not.

(5) Stories

By attending the students` design sessions, the observers could check whether if the students wrote story cards or not and whether they stuck them on the story wall. While checking these points, the observers did not examine which stories the students wrote, the observers were just interested in seeing if the students wrote the story cards at all.

(6) Weekly Cycle

Since the project duration was 5 weeks, the students had the chance to perform a weekly cycle 5 times. While the observers observed the students, the observers were checking on whether the students followed a weekly cycle or not. Here, the observers checked whether the group had a design meeting every Monday where the students made story and task cards which would later be accepted a meeting with the boss. On Fridays, the observers checked whether the students built the system and updated the document pages. The other days of the week, the observers checked whether the team worked in coding sessions followed by system builds on a half day basis, i.e. twice a day.

(7) Slack

On the weekly meetings, the observers asked the boss if any tasks had been dropped due to the fact that the task was not needed to make a feature of the program to work. Therefore, the boss asked the students which tasks they had performed and which they dropped.

(8) Ten-Minute Build

Here, the observers had to ask and trust the students how long the build took.

(9) Continuous Integration

The observers used CVS to check if the students integrated changes in the system. Unfortunately, there is no logging option in Eclipse[1] that the author was aware of to see how often tests had been executed. So, the observers had to note whether the students tested every 2 hours

[1] For details about Eclipse, see [10] & [11]

(10) Test-First Programming

The students were using JUnit[1] to test their classes and methods. The observers were not fully able to see whether or not the students wrote the test cases before or after they wrote the actual code that was tested. So, the observers would have had to compare the time stamps on the test files with the time stamps on the corresponding code files to be able to see if the students made the test cases first. However, this was particularly difficult since it was not obvious which test file corresponded to which code file. So instead, the observers had to ask and trust the students if they wrote tests before they wrote the corresponding code.

(11) Incremental Design

By observing the students on a daily basis the observers were able to see how much time the students invested on the design each day.

(12) Shared Code

With the CVS tool the observers were able to see if the students regularly uploaded the code to the CVS-server, thus sharing the code with the whole group.

(13) Code & Tests

The observers were not able to check if the students were uploading any files not related to Code & Tests by just viewing the file-names. The students could possibly keep documents and artefacts outside CVS. So, the observers needed to check the content of every file, document and artefact of students to see if they had documents and artefacts not concerning Code & Tests. Furthermore, the students were forced to keep statistics over the estimated time each task took and the actual time it took to finish a task. As mentioned before in section 3.4, the students were not able to apply this practice to its full extent but this was a result of the definition of the project.

(14) Single Code Base

Since the students had only one customer and one product, there was no need for more than one code base. Taking this into consideration, it made the evaluation of this practice redundant. Nevertheless, the observers checked the CVS repository to assure that only one code base existed.

(15) Negotiated Scope Contract

To measure this practice the observers used the simple approach of asking the boss of the project if the boss negotiated scope contracts with the project group or not.

3.5.4 Summary

- The observation protocol gives proof if the eXtreme Programming practices have been applied in the practical project or not.
- The observation protocol gives proof of the relation between the attitude towards a practice and its real execution.
- The observation protocol represents a very resilient data source that delivers views which help to form the big picture.

3.6 Survey 2

3.6.1 Introduction

Survey 2 was handed out to the students after the practical project had been finished. As before in Survey 1, the main requirement of Survey 2 is to measure the self evaluation responses of the students. As in Survey 1, the questions in Survey 2 have been grouped into items. This section is thus structured by these items and explains which questions belong to which item, the purpose of each item and how it is evaluated.[2]

3.6.2 The terms of reference of Survey 2

Survey 2 is compared to the different data sources that have been evaluated before survey 2 has been carried out namely Survey 1, the Tools and the Observation. The difference in response scales between the surveys (five point scale) and the observation protocol (three point scale) needs to be addressed to be able to compare the 2 data sources. Therefore, a formula - according to Aiken [33] – has been derived and is applied.

---

[1] JUnit is a framework to continuously run automated test suites against source code. It is represented by a GUI that shows the result of the test run which can either be pass or fail. For more information see: [4], [12], [13] & [14]

[2] It has to be mentioned that some questions were not analyzed in the context of this thesis. The questions might be used for other purposes, studies or evaluations in the future. These questions are not introduced in this thesis.

The terms of Survey 2 are similar to the terms of Survey 1 in section 3.3.2. Therefore, there are no further aspects that need to be mentioned here.

3.6.3 Items of Survey 2

The second survey has been set up to gather data regarding 8 items. These items are listed below. The items asked in Survey 2 are:

Item 1 This item is about how the project worked out

S2Q1 How did the project work out in your opinion?

S2Q18 How well did the project work at all?

The questions are both asking for the same kind of information. According to that assumption, the mean weighted values ($\overline{X}$) of the answers of the question are compared. The mean value ($\overline{X}$) is calculated according to the following formula:

$$\overline{X} = \frac{1}{n} * \sum_{i=1}^{n} x_i$$

The evaluation shows whether the mean values are in the same category[1] or not. If the 2 mean values are in the same category, this can be interpreted as a match. When it could be seen as a match it shows that the students have a similar level of understanding for the 2 questions that are asked in different places but query the same content.

One major difference between these 2 questions is that S2Q1 asks about the students' own opinion and S2Q18 just asks the status. This effect can be neglected since a survey always represents the respondents' own opinion.

Item 2 How well do you think your team used the 15 practices?

This item refers to S2Q2 and has been asked to compare the students' impressions about all 15 practices with the students' impression about every single practice (see item 4). Therefore, the results of item 2 are evaluated together with item 4 (see below).

Item 3 This item is about applying eXtreme Programming in a student environment.

S2Q3 Do you think eXtreme Programming is a good process to use in a student project?

[1] A response can be understood as matched when the values are in the same category. Since the categories are from 1 to 5 it must be 1 – 1.5; 1.6 – 2.5; 2.6 - 3.5; 3.6 – 4.5; 4.6 -5.

S2Q28 How well do you think eXtreme Programming works in a student environment?

These 2 questions are aiming for the students' attitude towards eXtreme Programming and how the students think about applying it in a student environment. Therefore, the mean weighted values of the answers are presented together. The mean weighted value is calculated as described in item 1. The item can be seen as valid if the answers are within the same category range[1]. This reveals whether the students think that a student environment is a good environment to apply eXtreme Programming or not.

Item 4 How well could you perform the practice "XY"?

This item is represented by S2Q4, Q5, Q6, Q8, Q9, Q10, Q11, Q12, Q13, Q14, Q15, Q16, Q17, Q23 &Q27 and it asks the students about their opinion of how well they were able to apply practice XY. XY in this case stands for the 15 practices, listed in Table 3-3 the students were asked to apply. This item is used for 2 different evaluations. The first one is a cross check with item 2. There are 2 differences between item 2 and 4. First, item 2 asks what each student thinks how well the team worked, whereas item 4 asks how well each student thinks he worked. To be able to compare the 2 items some steps are needed. Hence, the mean value of item 4 - the group response – is calculated. The second difference between item 2 and item 4 is that item 4 is asked for every single practice whereas item 2 is asked for all 15 practices. This difference is solved by calculating the mean value of all 15 questions asked in item 4. That makes it possible to talk about all 15 practices. Based on these 2 assumptions the 2 mean values are calculated and compared. The mean value for item 2 is calculated as described in item 1 and the mean value for item 4 ($\overline{X_{item4}}$) can be calculated as:

$$\overline{X_{item4}} = \frac{1}{n} * \sum_{i=1}^{n} (\frac{1}{m} * \sum_{j=1}^{m} x_m)_n$$

m = answers within one question (team);

n = the 15 practices;

The second use of item 4 is that it can give proof to the hypothesis H_{S21} together with the response from item 1.

[1] See foot note 1 for information about the category definition.

Item 5 This item is about how eXtreme Programming worked out in the students' project.

S2Q7 How well do you think XP worked in your project?

S2Q19 Do you think the project would have worked out better without eXtreme Programming?

These 2 questions were asked to evaluate the student groups' attitude towards eXtreme Programming in their project. The selectivity between item 5 and item 3 is just that item 3 asks about the students opinion regarding eXtreme Programming in a student environment whereas item 5 asks whether the students felt eXtreme Programming in the concrete project that had been performed, supportive or obstructive.

A very important point to mention in this item is that the 2 answers have different response scales. While Q1 has the typical Not Well At All to Very Well scale (1 – 5) Q2 has the Yes / ? / No (1; 0; (-1)) scale. To be able to compare the answers the following transformation formula according to Aiken [33] has been used.

Applying formula $Y = L_y + 0.5w_y + \frac{c_y * w_y(X - L_x + 0.5w_x)}{c_x * w_x}$ to the given scales

$X = \{1; 2; 3; 4; 5\}$ and $Y = (-1); 0; 1\}$ means[1]:

$Y = (-1) -0 .5 + (X-1+0.5) \frac{3}{5}$

$Y = (-1.5) + X * \frac{3}{5} -0 .5 * \frac{3}{5}$

$Y = (-1.8) + X * \frac{3}{5}$

$Y (X) = \{(-1.2), (-0.6), 0; 0.6, 1.2)$

In the end, the mean values of the 2 questions are compared to check whether the answers are in the same category[2] or not. The mean value is calculated as mentioned in item 1 and item 3.

[1] More information about the formula and how it is derived can be found in Aiken [33].

[2] A response can be understood as matched when the values are in the same category. Since the categories are from (-1) to 1 it must be. (-1) – (-0.5); (-0.4) - 0.4; 0.5 – 1.

Item 6 Do you think you would have worked in a different way if you had not been observed by us?

This question was asked in S2Q20 to be able to evaluate the influence that has been felt by students while they were observed during the practical project. It helps to interpret the gathered data. This response needs to be shown very detailed since it is very important how the different students answered. The median, the modal value and the mean value are calculated to analyze whether some students felt influenced or not. The item has the response scales Yes - 1 / ? - 0 / No – (-1) where every answer that is not No can be counted as Yes since even if a student is not sure that the observers' presence changed his behaviour it might have caused a change.

Item 7 Would you apply eXtreme Programming practices in the future?

S2Q21 If you would do another project, do you think you would apply some of the eXtreme Programming practices?

This question is followed by the following request: "If yes would you please list them on the other side of this sheet."

In contrast to the other items this item can be understood as a control indicating the success for the whole course. This question gives an indicator of the success of the goal of the lecture to show the students the software development process known as eXtreme Programming and enable them to apply it in real world projects. The fact that the students are willing to apply a practice is the first step in this direction. The responses of this item are analyzed in 2 ways. The first one is that every response that is not Yes for S2Q21 which has a 3 item response scale. The 3 items are Yes - 1 / ? - 0 / No – where (-1) counts as failure. Due to the open response possibility that followed S2Q21 its responses are mapped to the practice – student – list for evaluation. This list contains the information which student will apply what practices in future projects. All additional information will be neglected.

Item 8 How do you like the idea of having theoretical sessions on XP followed by performing the theory into practical use?

This item refers to S2Q26 and is the same as item 5 of the first survey. In the end, the 2 items are compared with each other to see whether the students' opinions changed during the course of the practical project or not. To compare the 2 items the mean values ($\overline{X}$), the modal values ($\overline{X}_{MO}$) and the median values ($\overline{X}_{MD}$) are compared with each other. Comparing all these values will give a detailed comparison between the 2 items.

The median value ($\overline{X}_{MD}$) returns the value where 50% of the answers are below and 50% of the answers are above the median value. Its formula is:

$$\overline{X}_{MD} = X_{\frac{n+1}{2}} \qquad \text{if } n \text{ is an even number of responses.}$$

$$\overline{X}_{MD} = \frac{\left(X_{\frac{n}{2}} + X_{\frac{n}{2}+1} \right)}{2} \qquad \text{if } n \text{ is an odd number of responses.}$$

Whereas the modal value ($\overline{X}_{MO}$) stands for the most frequent answer. The modal value offers the most common answer and the frequency of the most common answer.

3.6.4 Summary

- Survey 2 indicates whether the students felt like having applied the practices of eXtreme Programming.
- Survey 2 gives indicates whether he student group felt influenced by the observers.
- Survey 2 gives indicates whether the students are satisfied with the project.
- Survey 2 gives indicates whether the students felt eXtreme Programming as hindering.
- Survey 2 gives indicates whether the students disliked applying the practices in a student environment.
- Survey 2 indicates – together with Survey 1 – if the students' attitude towards the structure of the course, i.e. lectures followed by a practical project, changed after they had performed the practical project.

3.7 The sample

The sample base consisted of ten students. These students were all Swedish males. Two students were twins with each other and one student was wheelchair bound. The group's precise age distribution can be found in Figure 3.2

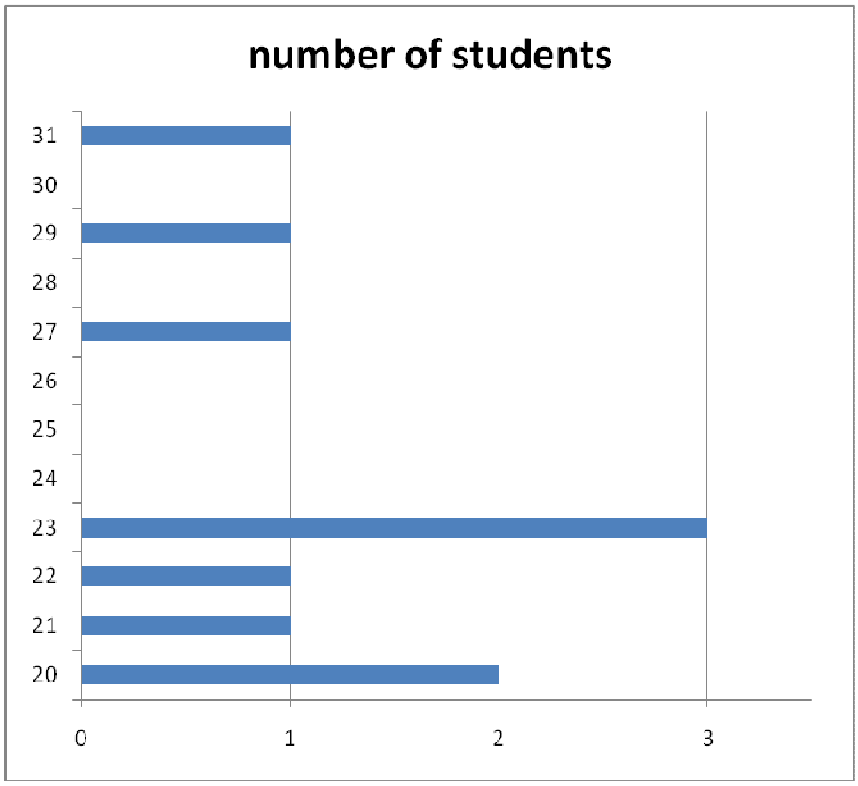

Figure 3.2 Distribution of the students' age

The test whether the sample is representative or not was needless since the sample represented the whole semester of the computer science department at Karlstad University. As can be found in Bortz & Döring [24] the test whether a sample is representative or not is only needed to prove that the sample is representative for the base it represents. Since in this case the sample represents the whole base a proof is not needed.

From the second relevant point of view - the eXtreme Programming point of view - the group can be seen as representative since the group consisted of 10 people according to Gladwell [8] twelve is the maximum number of team members to interact comfortably with each other.

3.8 Summary

In this chapter, the analysis model, the hypotheses, all data sources and methods to prove or not to prove the theories have been shown. All significant metrics have been shown.

The data sources have been introduced as Survey 1, Survey 2, the Tools and the Observation. First, 2 interviews, realized as surveys (Survey 1, Survey 2), were set up that captured the students self evaluation. Survey 1 asks about the students' level of understanding after finishing the theoretical lectures and the students' attitude towards the eXtreme Programming practices and the applicability of the practices during the practical project. Survey 2 is a snapshot taken of the students' self evaluation on how well they felt that they had applied eXtreme Programming and its practices during the practical project, how the students liked the project, how the students liked the project in a student environment and whether the students will apply any eXtreme Programming practice in future projects. Besides, it was shown how the 15 practices were measured while the students applied them in the practical project with help of the Observation Protocol and the Tools.

According to Yin [23] and Bortz & Döring [24] the given variety of data sources is sufficient to validate the hypotheses and other fact building theories. Moreover, the sample can be considered as significant and representative in the given context.

4 Empirical Study

4.1 Introduction

In this chapter, all the evaluated data is presented, analyzed and interpreted according to the analysis model introduced in chapter 3. For each and every result the raw data, for example the responses of the surveys, the protocol can be found in the appendix.

4.2 Survey 1[1]

4.2.1 Introduction

This section aims in proving or disproving the hypothesis H_{s11}. Therefore, the items are connected with each other as required by the model in chapter 3.

Two response scales exist in Survey 1. The first response scale has 5 labels:

1-Very Well

2-Well

3-Neutral

4-Not Well

5-Not Well At All

The other response scale is a Yes / No scale. In this section these verbal scales are converted to numerical scales since numerical scales can be calculated more detailed.

4.2.2 Evaluation

The 5 items described in section 3.3 are evaluated below. The items have been evaluated independently and then analysed in conjunction with other items of survey 1.

Item 1 How well did you understand all practices?

The mean value of item 1 is 2.3. The calculation of this value comes as follows: ((1*0) + (2*7) + (3*3) + (4*0) + (5*0))/10; 1 for answers in the category Very Well, 2 for answers in the category Well, 3 for answers in the category Neutral, 4 for answers in the category Not

Well and5 for answers in the category Not Well At All. All of this is divided by 10 which is the number of responses. Items 1 and 3 give an indication of the degree to which the students and the group as a whole understood the practices.

Item 2 How well do you think the practice "XY" is applicable during the project?

The aim of item 2 is to evaluate whether the students have been willing to apply the practice XY of eXtreme Programming or not. Therefore, the 3 categories “Applicable” (response≤2.5), “Not Applicable” (response≥3.5) and “Undecided” (2.5<response<3.5) have been set up. Figure 4.1 shows the “Undecided” category as dashed transparent area between the “Applicable” and the “Not Applicable” areas.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1.6	2.9	2.5	1.6	2.2	2.4	3.0	2.3	2.2	2.9	2.5	2.1	1.8	2.2	2.7

Table 4-1Weighted mean values for survey 1 item 2

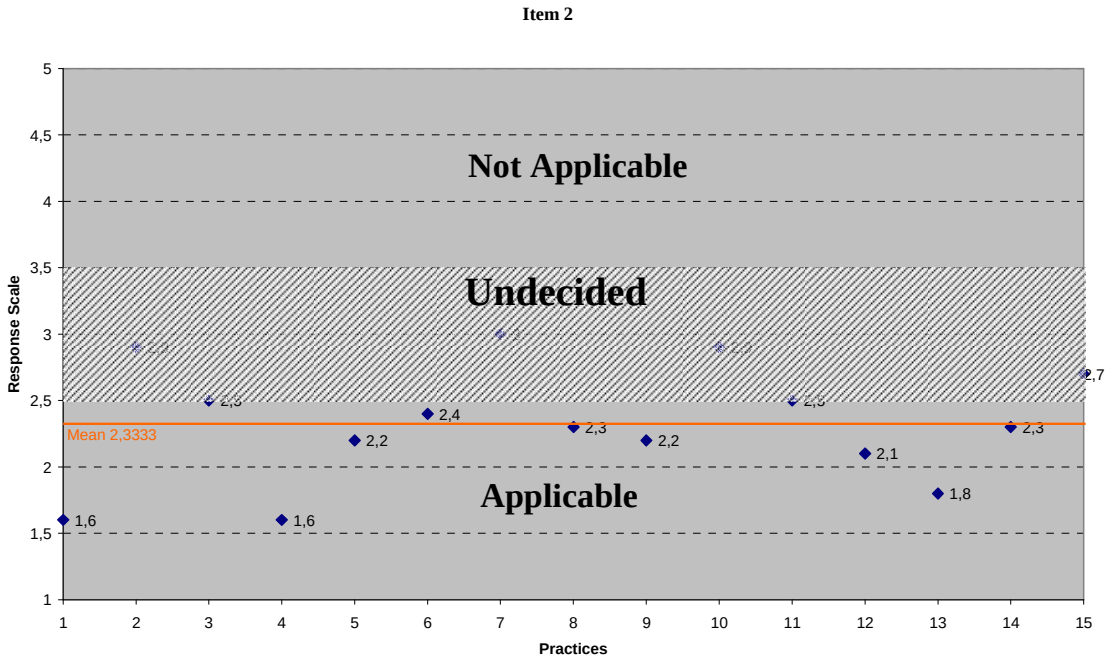

Figure 4.1 Responses to survey 1 item 2

[1] The responses of Survey 1 can be found in appendix C.

The y-axis values of Figure 4.1 represent the response scale 1 – 5 whereas the x-axis values of Figure 4.1 represent the following 15 practices:

1. Sit Together
2. Informative Workspace
3. Energized Work
4. Pair Programming
5. Stories
6. Weekly Cycle
7. Slack
8. Ten Minute Build
9. Continuous Integration
10. Test-First Programming
11. Incremental Design
12. Shared Code
13. Code & Test
14. Single Code Base
15. Negotiated Scope Contract

The values shown in Figure 4.1 represent the weighted mean values for every single practice over all students asked in Survey 1. The orange line across the whole figure represents the mean over all 15 practices. This has been shown to enable the reader to see the relation between the mean of one practice and the mean of all practices.

As can be seen in the Figure 4.1, there are 4 mean values in the "Undecided" category which represent the practices Informative Workspace, Slack, Test-First Programming and Negotiated Scope Contract. This result is related to the results of item 3 below. Furthermore, the result will be used to prove the hypothesis H_{o1} in section 4.4 where the observation is evaluated.

Item 3 How well did you understand the practice "XY"?

The responses to item 3 are illustrated in Figure 4.2. Again, the x-axis represents the 15 practices.[1] The answers are divided into the 3 categories "Understand" (response $\leq$ 2.5),

[1] To map the numbers 1-15 to the 15 practices the list of item two can be applied.

“Undecided” (2.5 < response < 3.5) and “”Not Understand” (response ≥ 3.5) shown in Figure 4.2.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1.7	1.8	2.4	1.5	2.1	1.9	2.7	2.4	2.1	2.3	2.2	2.1	2.2	2.6	3

Table 4-2 Weighted mean values for item 3

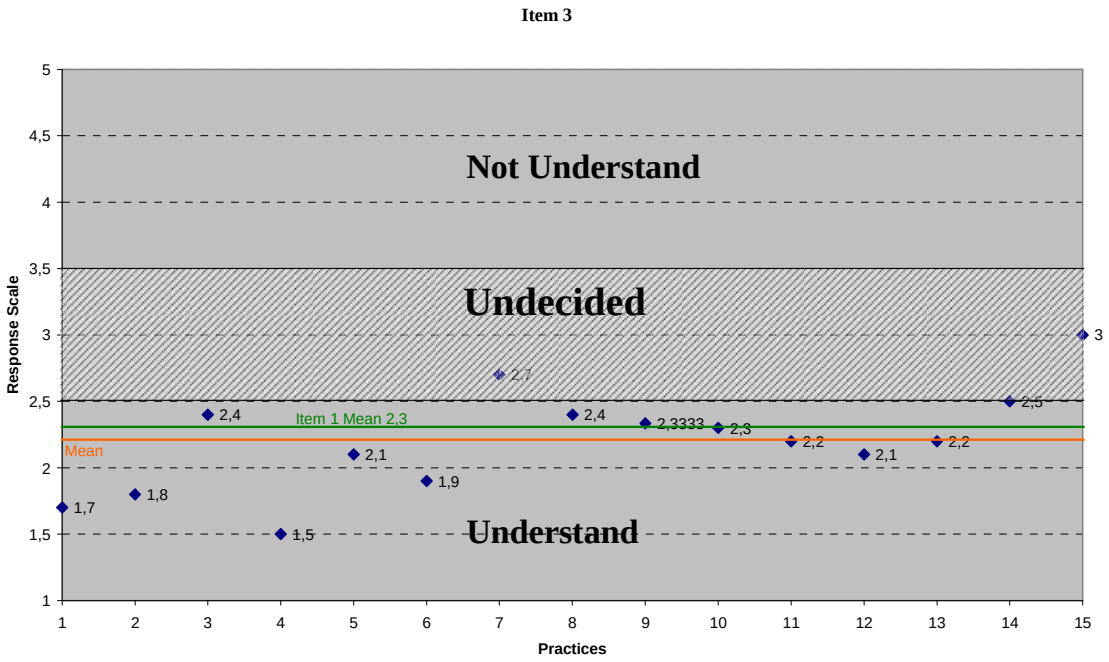

Figure 4.2 Responses to item 3

As can be seen in Figure 4.2, all practices have been well understood except the practices Slack (7) and Negotiated Scope Contract (15) in the “Undecided” category.

This result can be related to the following results of item 1 and 2 in order to be evaluated.

The mean value of item 1 (2.3) and the mean value of item 3 (2.2) are both in the category “Understand” indicating that the students have in general understood the practices well. The students’ understanding of all the practices matches their understanding of each practice in total.

Furthermore, item 3 can be linked to the results of item 2 since a relation between the level of understanding and the estimation of how well practice XY would be applicable seems to be

a reasonable assumption. Therefore, Table 4-3 compares the 4 values (2, 7, 10, 15) in the "Undecided" category of item 2 with the respective answers (2, 7, 10, 15) of item 3.

#	Practice	Item 2 (Applicable)	Item 3 (Understand)
2	Informative Workspace	2,9	1,8
7	Slack	3	2,7
10	Test-First Programming	2,9	2,3
15	Negotiated Scope Contract	2,7	3

Table 4-3 Comparing the understand & the applicable mean values

As Table 4-3 shows on the one hand, the students' level of understanding is in the "Neutral" category (range 2.6 to 3.5) for the practices Slack and Negotiated Scope Contract. For these practices a reasonable interpretation is that the students have difficulty understanding them since they have had little experience in these practices. On the other hand, Table 4-3 shows that the students understood the practices Informative Workspace and Test-First Programming but thought they were not applicable. So here, it cannot be considered that the students had the same problems they had with the other 2 practices. The students rather dissociated themselves from applying this practice since they had understood it. It will be interesting to see further in this evaluation if the students' minds changed while performing these practices.

Finally, the results of item 3 serve in examining the hypothesis H_{S11}. The purpose of hypothesis H_{S11} is to prove whether the students understood the practices or not. The precise wording of the hypothesis is: "A student group that has attended the theoretical sessions on eXtreme Programming understands the practices well." The wording of the counter hypothesis is: "A student group that has attended the theoretical sessions in eXtreme Programming does not understand the practices well." H_{S11} is proved when H_{S10} is not proved. If the response for a student group is in the category "not well", H_{S10} would be proved. In item 3, "Not Understand" is defined as: response $\geq$ 3.5. As can be seen in Figure 4.2, there is no response value > 3. Thus, there is no answer in the category "Not Understand" so that H_{S10} is

unproved. This leads to the conclusion that the hypothesis H_{S11} is proven. Hence, a student group that has attended the theoretical sessions understands the practice well.

Item 4 Have you decided to embrace XP in the project?

This question was significant since it is important to see whether the students wanted to apply eXtreme Programming or not while evaluating the data. The response to this Yes / No item was 100 percent Yes. This significant result can be seen as a very positive starting point for the practical project where the students had to fulfil only one task: applying eXtreme Programming for 5 weeks as much as possible. One could say that the students' task was to apply XP and therefore they needed to answer like this but from the authors point of view the students had no reason to connect the answers of the survey to the attitude of the teacher since the teacher did not receive the responses immediately.

Item 5 How do you like the idea of having theoretical sessions on XP followed by performing the theory into practical use?

Item 5 will be evaluated together with Survey 2 item 8 in section 4.5.2. The responses from this item can be found in Table 4-4. For now can be said that the students liked the structure of the course. This can be said since all results are in the well category.

Question	Modal value	Median	Mean value
How do you like the idea of having theoretical sessions on XP followed by performing the theory into practical use?	2	2	2,3

Table 4-4 Responses on item 5

4.2.3 Conclusion

This section about Survey 1 confirmed a number of expectations.

The first and very important step was to see whether the students understood the practices they had to apply during the practical project or not. As the evaluation of the students' responses shows, the students understood, except for 2 practices, all practices well. This is supported by the verification of the hypothesis. The 2 practices the students did not understand well are Slack and Negotiated Scope Contract. The practical project will show whether the students finally understood the practices while performing the practical project or not.

Another goal to be reached by Survey 1 was to evaluate whether the students thought that the practices would be applicable in their project or not. Regarding this item, the survey revealed that the students thought that 4 practices (Informative Workspace, Slack, Test-First Programming and Negotiated Scope Contract) are not well applicable in the practical project. In addition to the 2 practices Slack and Negotiated Scope Contract which the students did not understand well, the students did not believe that the practices Informative Workspace and Test-First Programming were applicable during the practical project. For the 2 practices the students did not understand (Slack and Negotiated Scope Contract) it seems to be consistent that the students were not sure if these practices are applicable or not. For the other 2 practices (Informative Workspace and Test-First Programming) the students seemed to have problems in understanding how these practices have to be performed in this kind of practical project i.e. in a teaching environment and not an industrial environment.

The last result from Survey 1 is that the whole student group was happy to embrace eXtreme Programming in the practical project.

All entry criteria for the practical project have been fulfilled and the fact that the students were suspicious about applying 4 of these, namely Informative Workspace, Slack, Test-First Programming and Negotiated Scope Contract) of the 15 practices, can be seen rather as an opportunity than a problem since a critical point of view is worth much more than an indifferent attitude.

4.3 The Tools

4.3.1 Introduction

This section verifies if the 4 practices that can be evaluated by analyzing the log files have been applied or not. These were Continuous Integration, Shared Code, Code & Test and Single Code Base, and are evaluated in this order.

4.3.2 Evaluation

4.3.2.1 Continuous Integration

Continuous Integration is evaluated by checking whether the file count and the average file size increased while the project was performed. Therefore, Figure 4.3 and Figure 4.4 that are

generated by StatCVS provide information about the application of the practice by the student group.

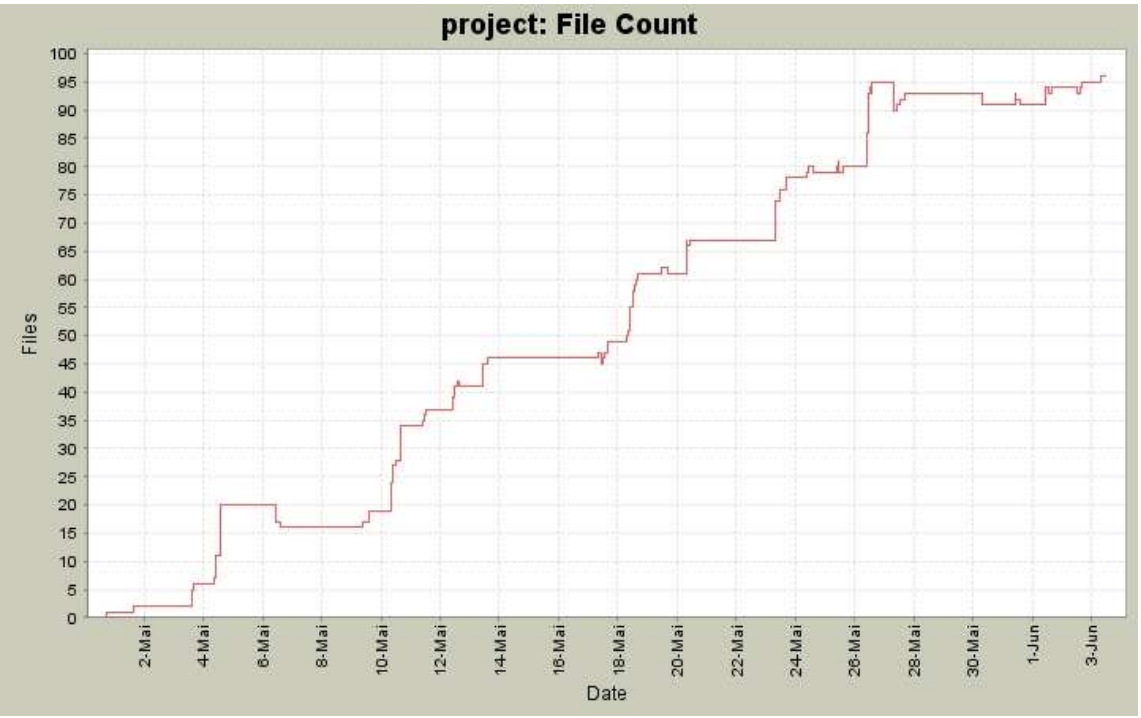

Figure 4.3 File count during the project phase

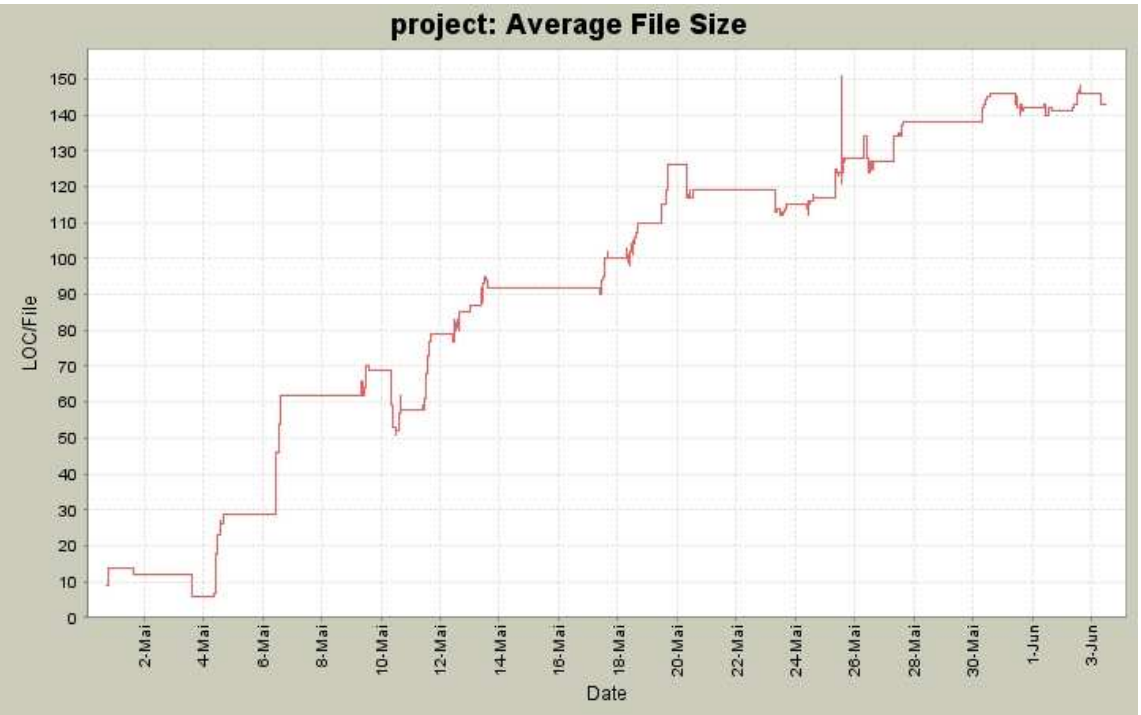

Figure 4.4 Average file sizes during the project phase

As can be seen in Figure 4.3 and Figure 4.4 the file count as well as the average file size nearly grew linearly during the 5 weeks. Based on this information this practice can be counted - from the point of view of the tools - as applied fully.

4.3.2.2 Shared Code

Shared code means that all team members have access to the code and all team members are working on all files of the code. The measuring point to evaluate this practice is whether all students used CVS or not. Use in this case is defined as doing add and modify transactions.

The data - presented in Table 4-5 - has been extracted from CVS to verify that the students used CVS.

Committer	Total transactions	LOC[1]
Student 1	418	8173
Student 2	504	7004
Student 3	5	3174
Student 4	91	1510
Student 5	54	1205
Student 6	29	1179
Student 7	95	1155
Student 8	30	1105
Student 9	39	776
Student 10	9	439
Total	**1274**	**25720**

Table 4-5 CVS transactions

As Table 4-5 shows, all students used CVS since all students performed transactions. However, from Table 4-5 it is not shown whether the students performed both add and modify transactions. Therefore, Figure 4.5 divides the total number of transactions per student into add transactions and modify transactions.

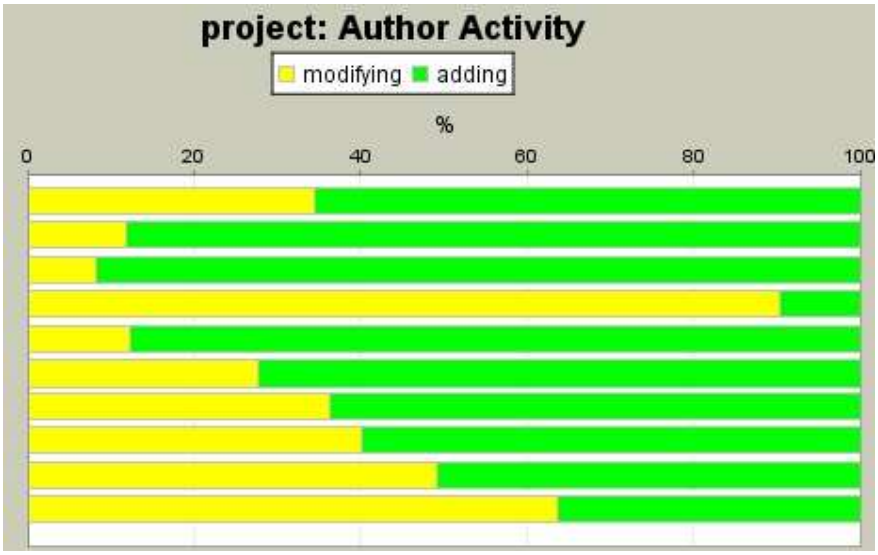

Figure 4.5 Anonymous quotas of the students' add and modify transactions

[1] LOC = Lines Of Code

As Figure 4.5 shows, each student performed both types of transactions. The results from Table 4-5 and Figure 4.5 provide enough evidence to prove this practice as technically performed.

4.3.2.3 Code & Test

The practice Code & Test is defined as not keeping other than project related files. A report - based on the CVS repository - has been created to evaluate this practice. The following analysis only considers files stored in the CVS. All files that are stored outside of the CVS were not evaluated by this verification. Table 4-6 shows which file types have been stored in the CVS file structure.

Type	Files	Files %
Totals	96	100.00%
*.java	73	76.00%
*.txt	1	1.00%
*.xml	6	6.30%
Others	1	1.00%
Non-Code Files	15	15.60%

Table 4-6 File types in the CVS

As shown in Table 4-6 more than 15 percent of the repository is used for non code files. It is to be determined which files these are and why they are stored in the CVS. The repository showed that the non-code files consisted of 3 files of database scripts, 1 RTF about how to configure Eclipse, 8 JAR archives, 2 help files and 1 picture. The database scripts are needed. The RTF was necessary for the students since the students are not well trained developers who bring the knowledge about development environments. Thus, it is no problem for the evaluation of this practice. The picture was needed for the GUI. The JAR archives do, according to best practice approaches, not belong into a CVS repository. However, since JAR archives contain just code, they do not break with the practice. Contrary to all these files, the 2 help files, used to explain the developed application, break the practice. According to Beck's definition (see Beck [2]) of Code & Test, the users' manual does not belong into the repository.

This leads to the conclusion that the practice has not been applied fully.

4.3.2.4 Single Code Base

This practice was tested by checking the existence of just one repository. The administrator of the CVS server was a person from the staff of the university. Therefore, it was impossible for the students to create another repository. This practice would have been redundant in the given context and should have been taken out of the student environment project. Considering these circumstances, the practice has been applied fully.

4.3.3 Conclusion

Except Code & Test, all other practices, namely Continuous Integration, Shared Code and Single Code Base, have been applied.

4.4 The observation protocol[1]

4.4.1 Introduction

This section presents a summary of the observation protocol and shows how the items are verified using this summarized data. It lends proof to the hypothesis H_{O1} together with the data gathered in Survey 1 and evaluated in section 4.2 as well as the results from section 4.3 for the practices Continuous Integration, Shared Code, Code & Test and Single Code Base.

4.4.2 Evaluation

Table 4-7 shows a summary of the detailed observation protocol. The table illustrates which practices have been applied or not during the 5 weeks of practical project. The practice Weekly Cycle was not evaluated in the first week.

[1] The whole observation protocol can be found in appendix D.

#	Practice Name	Week 1		Week 2		Week 3		Week 4		Week 5		Mean
1	Sit Together	Part²	0,75	Part²	0,75	Part²	0,35	Part²	0,95	Part²	0,95	0,75
2	Informative Workspace	Yes	1	Yes	1	Yes	1	Yes	1	Yes	1	1,0
3	Energized Work	Yes	1	Yes	1	Yes	1	Yes	1	Yes	1	1,0
4	Pair Programming	No	-1	No	-1	No	-1	No	-1	No	-1	-1,0
5	Stories	Yes	1	Yes	1	Yes	1	Yes	1	Yes	1	1,0
6	Weekly Cycle	n/a	n/a	No	-1	No	-1	No	-1	Yes	1	-0,5
7	Slack	Yes	1	Yes	1	Yes	1	Yes	1	Yes	1	1,0
8	Ten-Minute Build	Yes	1	Yes	1	Yes	1	Yes	1	Yes	1	1,0
9	Continuous Integration	Yes	1	Yes	1	Yes	1	Yes	1	Yes	1	1,0
10	Test-First Programming	Yes	1	Part²	0,3	Part²	0,3	Part²	0,3	Part²	0,3	0,44
11	Incremental Design	Yes	1	Yes	1	Yes	1	Yes	1	Yes	1	1,0
12	Shared Code	Yes	1	Yes	1	Yes	1	Yes	1	Yes	1	1,0
13	Code & Test	No	-1	No	-1	No	-1	No	-1	No	-1	-1,0
14	Single Code Base	Yes	1	Yes	1	Yes	1	Yes	1	Yes	1	1,0
15	Negotiated Scope Contract	No	-1	Yes	1	No	-1	Yes	1	Yes	1	0,20

Table 4-7 Practices applied per week in words and numerical values

The numerical values are calculated according to the rules in section 3.5. The values for the partial fields have been classified and calculated. The Sit Together values have been calculated as follows: If the group did not go to lunch together it cost the practice 0.05 points whereas the whole absence of one student cost 0.20 points. For the values in the Test-First columns the following approach has been chosen: The 0.3 was determined since it represents the borderline between the “Applied” and the “Undecided” category. 0.3 is counted as “Applied” and this was due to 2 factors. The first one is that even the students who did not apply Test-First Programming thought about the practice but did not see a benefit in applying it. The second point is that some students applied Test-First Programming but since this particular group represented less than fifty percent of the total number of students it is not placed in the middle of the “Applied” category.

By comparing the data from Table 4-8 with the data from Survey 1, the hypothesis H_{O1} can be validated. Therefore, the response given in Survey 1 item 2 is listed in Table 4-8 and transformed to the scale of (-1; 0; 1) to be able to compare it to the results of the observation and tools. Both results as well as their difference in absolute value are shown in Table 4-8.

#	Practice Name	Survey 1	T & O[1]	Difference
1	Sit Together	0,84	0,75	0,09
2	Informative Workspace	0,06	1,00	0,94
3	Energized Work	0,30	1,00	0,70
4	Pair Programming	0,84	-1,00	1,84
5	Stories	0,48	1,00	0,52
6	Weekly Cycle	0,36	-0,50	0,86
7	Slack	0,00	1,00	1,00
8	Ten-Minute Build	0,42	1,00	0,58
9	Continuous Integration[3]	0,48	1,00	0,52
10	Test-First Programming	0,06	0,44	0,38
11	Incremental Design	0,30	1,00	0,70
12	Shared Code[3]	0,54	1,00	0,46
13	Code & Test[3]	0,72	-1,00	1,72
14	Single Code Base[3]	0,42	1,00	0,58
15	Negotiated Scope Contract	0,18	0,20	0,02

Table 4-8 Survey 1 item 2 compared to the protocol of the observation and the tools

The hypothesis H_{O1} said that a student's expectation about how he will apply practice XY is in relation to how the student will apply it in the practical project. Meanwhile, the corresponding null hypothesis H_{O0} says that the attitude a student has towards an eXtreme Programming practice has no influence on how it will be applied during the practical project. The hypothesis is proved if the null hypothesis is not proven. It can be assumed that there is no relation between the expectation before the practical project (value of Survey 1) and the real application (value of the tools and observation) if the difference between the 2 values is at least 0.6. In the (1-5) scale one category has a width of 1. Transferring this value into the ((-1) – 1) scale it will become 0.6, so that one category has a width of 0.6. Thus, if the difference between the 2 values is at most 0.6 the answers are in the same category. Whereas a difference that is more than 0.6 it is too great to be understood as close.

As can be seen in Table 4-8, there are 7 "Difference" values higher and 8 values lower than 0.6. This result makes it possible to prove the null hypothesis H_{O0} for 7 practices and by this the hypothesis H_{O1} can not be proven for those practices. Nevertheless, the null hypothesis

[1] Column T & O (Tools and Observation) contains the results of section 4.3 and 4.4
[2] Part = Partially applied
[3] This practices has been measured via Tools

H_{O0} can be disproved for 8 practices so that for these practices the hypothesis H_{O1} could be proved. Eight out of 15 is more than 50 percent but far away from a significant result that could have verified the hypothesis H_{O1} for all practices.

However, it is possible to conclude that a relationship between the expectation and the appliance cannot be excluded for the following practices:

- Sit Together
- Stories
- Ten-Minute Build
- Continuous Integration
- Test-First Programming
- Shared Code
- Single Code Base
- Negotiated Scope Contract

4.4.3 Conclusion

The main purpose of the observation was to evaluate whether the students applied the eXtreme Programming practices or not. So, the observation has been performed and recorded. As shown in this section, the majority of practices have been applied. Only the 3 practices Pair Programming, Weekly Cycle and Code & Test have not been used. For the practice Weekly Cycle this is to bee seen with the exception, that it has been applied in the last week. However, the 3 practices Sit Together, Test First Programming and Negotiated Scope Contract have been partially applied.

Furthermore, the unproved hypothesis H_{O1} shows that there is no evidence for the relation of the students' attitude before starting a project and the students' way of behaviour during the project for at least 7 of 15 practices. This prepares the ground for the assumption that the teachers do not need to convince the students about the practices. They rather need to explain the practice very detailed to enable the students to apply the practice.

4.5 Survey 2[1]

4.5.1 Introduction

Survey 2 is very important since it supports the conclusion of the other data sources. From a high level perspective there are 4 main issues to be verified by Survey 2:

1 the students' attitude towards the project they had just performed,
2 their view on the structure of the course,
3 the influence the observers had on the students and a
4 learning success measurement by checking if and which practices the students will apply in future business and university projects.

4.5.2 Evaluation

Survey 2 consists of 8 items which are evaluated below in detail.

Item 1 How did the project work?

The purpose of this item is to measure the students' attitude towards the project. Therefore, the following 2 questions have been asked:

S2Q1 How did the project work out in your opinion?
S2Q18 How well did the project work at all?

There are 2 ways of evaluating this item: either by evaluating each question on its own or by using the mean weighted value of the 2 questions. Figure 4.6 shows every single answer as well as the mean values per student.

[1] The original responses of Survey 2 can be found in appendix E.

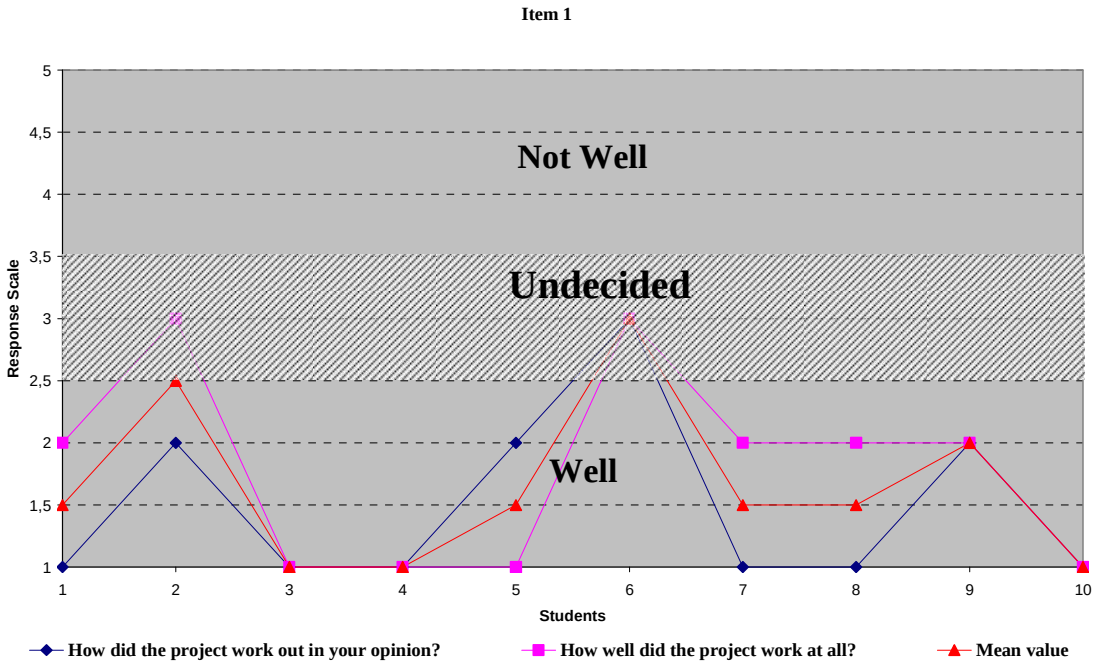

Figure 4.6 Responses on item 1

There are 2 points that can be drawn from Figure 4.6. As can be seen in Figure 4.6, 5 students or 50 percent of the student group answered equally for all questions whereas the other 50 percent are just separated by one scale unit. That shows that all students answered both questions very closely. Except for one response with a mean value in the "Undecided" category, all mean values are in the "Well" category. However, this result just gives an answer to each individual student's opinion. In order to evaluate the group opinion, the mean value is needed. Therefore, this value is listed in Table 4-9.

Question	Mean value
How did the project work out in your opinion?	1,5
How well did the project work at all?	1,8
Mean value	1,65

Table 4-9 Mean value of item 1

As can be seen in Table 4-9 the lowest value is 1.8 standing for the "Well" category (1-2.5). This data suggests the assumption that the student group thought that the project went well.

Item 2 How well do you think your team used the 15 practices?

This item is evaluated to have a reference value for item 4. Therefore, the mean value is calculated and compared with the responses of item 4. The calculation as well as the comparison can be found under Item 4.

Item 3 About applying eXtreme Programming in a student environment

Item 3 asks the students what they think about applying eXtreme Programming in a student environment respectively the current practical project. It seemed to be interesting to see if the students thought that the student environment caused problems while the students wanted to apply the practices.

Figure 4.7 shows the students' responses on item 3.

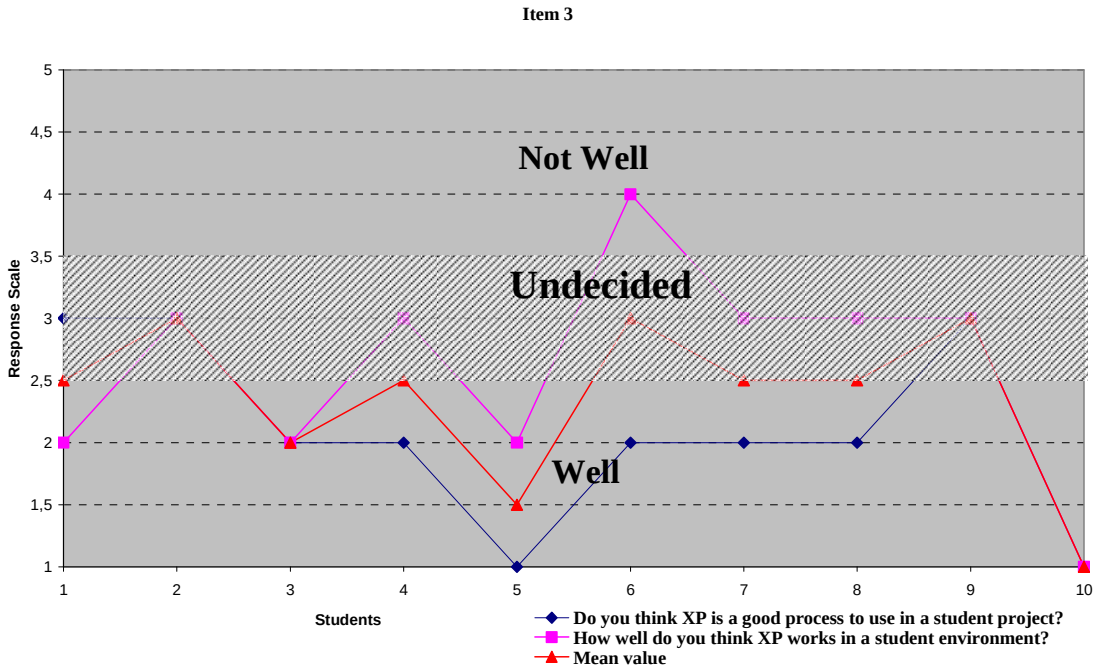

Figure 4.7 Responses on item 3

As shown in Figure 4.7, 40 percent of the students (students 2, 3, 9, 10) answered using the same value; 50 percent (students 1, 4, 5, 7, 8) answered with a difference of 1 in the value, and one student (student 6) answered with a difference of 2 in the value. Figure 4.7 just shows the answers per student and not the opinion of the student group. It is not possible to draw a conclusion just by having a look at the single values so that the mean value needs to be calculated (see Table 4-10).

Question	Mean value
Do you think XP is a good process to use in a student project?	2.1
How well do you think XP works in a student environment?	2.6
Mean value of item 3	2.35

Table 4-10 Mean value of item 3

As can be seen in Table 4-11, the mean value of the whole group and item 3 is in the "Well" category (1-2.5).

This leads to the conclusion that even though the responses of some students were in the categories "neutral", "not well" and "not well at all", the student group did not see a problem in the fact that the project had to be performed as a student project in a student environment. Of course, the students' replies could have been better (< 2.35). Since the value 2.35 is close to the "neutral" category, this will be discussed in more detail in the conclusion.

Item 4 How well could you perform the practice "XY"?

Item 4 was set up for 2 reasons. First, it was meant to compare the student group's overall impression with each individual impression. Therefore, the mean value of item 2 is compared to the mean value of all students of item 4. These mean values as well as the answers to item 4 of each student are illustrated in Figure 4.8.

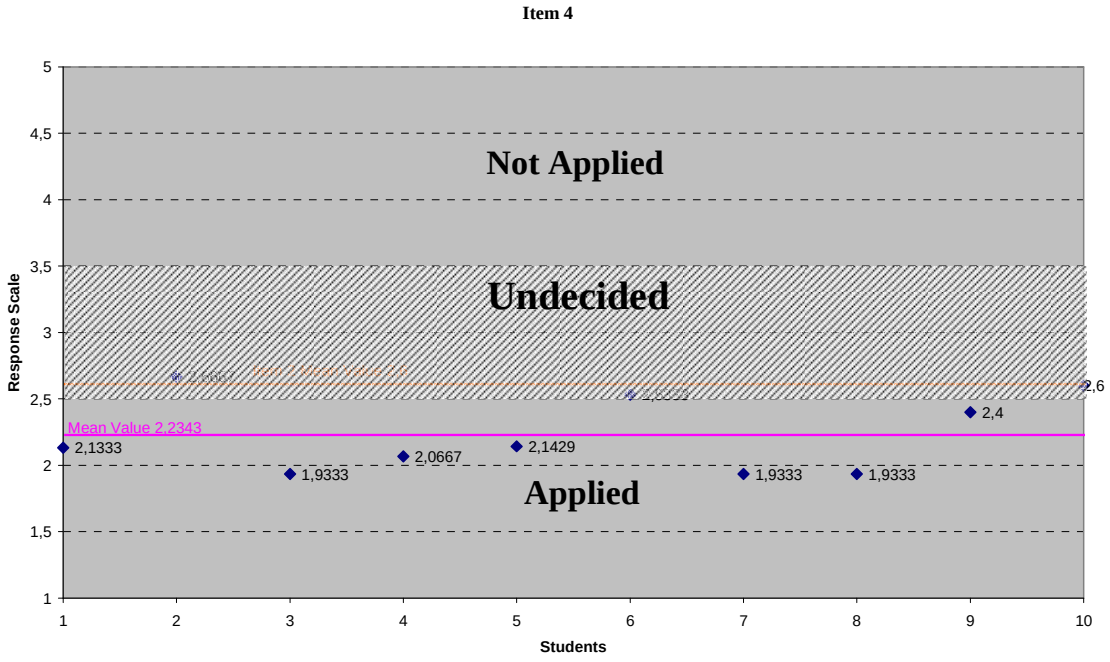

Figure 4.8 Responses on item 4

As can be seen in Figure 4.8, the mean values of item 2 (illustrated in orange) and item 4 (illustrated in pink) do not match. The mean value of item 2 (2.6) lies in the "Undecided" category whereas the mean value of item 4 (2.2) lies in the "Applied" category. One possible reason could be that each student, except for the 3 outliers in the "Undecided" category, ranks himself better than his perceived behaviour of the group as a whole. Another possibility could be that the students have a negative opinion about the practices in total – visible in the higher mean value of item 2 – however, when the students think about each practice in detail they come to the conclusion that each practice in itself has been applied better than the overall impression for all the practices.

The second reason why item 4 was set up has been to verify hypothesis H_{S21}. This hypothesis said that a student groups' self evaluation about the level of application of an eXtreme Programming practice will match the groups' real behaviour. Therefore, the mean values per practice of Survey 2 are compared to the mean values of the observation protocol for all weeks (see Table 4-11). The difference between the 2 data sources gives information about the hypothesis H_{S21}. The difference is calculated as $|A-B|$ where A stands for the mean value of Survey 2 and B for the mean value of the observation.

#	Practice Name	Survey 2[1]	O & T[2]	Difference[1]
1	Sit Together	0,84	0,75	0,09
2	Informative Workspace	0,06	1,00	0,94
3	Energized Work	0,48	1,00	0,52
4	Pair Programming	0,12	-1,00	1,12
5	Stories	0,24	1,00	0,76
6	Weekly Cycle	0,24	-0,50	0,74
7	Slack	0,54	1,00	0,46
8	Ten-Minute Build	0,00	1,00	1,00
9	Continuous Integration	0,48	1,00	0,52
10	Test-First Programming	0,90	0,44	0,46
11	Incremental Design	0,48	1,00	0,52
12	Shared Code	0,84	1,00	0,16
13	Code & Test	0,18	-1,00	1,18
14	Single Code Base	0,80	1,00	0,20
15	Negotiated Scope Contract	0,72	0,20	0,52

Table 4-11 Comparison of Survey 2 Item 4 and, the observation and tools

[1] The values of Survey 2 are transferred to the (-1; 0; 1) scale.
[2] Column T & O (Tools and Observation) contains the results of section 4.3 and 4.4.

The following assumption is needed to numerically prove the hypothesis H_{S21}:

The hypothesis H_{S21} is proven if the corresponding null hypothesis H_{S20} is unproven. The null hypothesis H_{S20} said that a student groups' self evaluation about the level of application of an eXtreme Programming practice will not match the groups' real behaviour. That means numerically if $0.6 \leq |A\text{-}B|$ the null hypothesis is unproven. In the (1-5) scale one category has a width of 1. Transferring this value into the (-1; 0; 1) scale it will become 0.6, so that one category has a width of 0.6. Thus, if the difference between the 2 values is at most 0.6 the answers are in the same category, meaning the group's self evaluation matches the group's real behaviour. Whereas in case the difference is more than 0.6, the 2 answers are too far apart from each other to be counted as close.

If any field in the right-most column of Table 4-11 is shaded orange it means that the difference between the response of the survey and the evaluation of the observation is less than 0.6. As can be seen in Table 4-11, 9 out of 15 practices - that means nearly two thirds of the answers to Survey 2 - are close to the value of the observed behaviour as the corresponding value is at most 0.6. This is not sufficient to disprove the null hypothesis. As a result, the hypothesis H_{S21} cannot be proven and thus, cannot be verified. As consequence, a student groups' self evaluation about the level of application of an eXtreme Programming practice does not match the groups' real behaviour.

The fact that the hypothesis cannot be proven does not support the assumption that the students are fully aware that they have applied the eXtreme Programming practices. This leaves space for the assumption that the students were not aware of the practices they applied.

Item 5 About how eXtreme Programming worked out in the students' project.

The purpose of this item is to evaluate if the students felt eXtreme Programming as supportive or obstructive while the students performed the practical project. Therefore, the following 2 questions have been asked:

S2Q19 Do you think the project would have worked out better without XP?

S2Q7 How well do you think XP worked in your project?

The response scale of the second question needed to be transferred into the response scale of the first question to be able to evaluate these 2 questions. Figure 4.9 shows the responses in the same response scale.

[1] The difference is represented as absolute value |difference|.

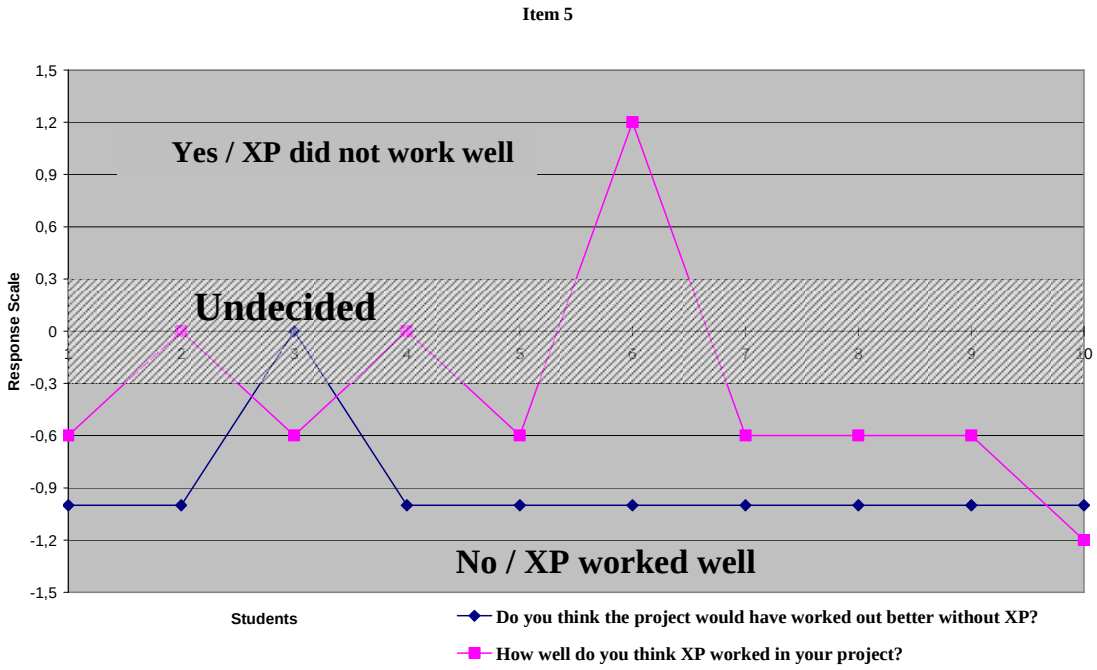

Figure 4.9 Responses on item 5

As Figure 4.9 shows, except for one response all students responded in the "No" category when it was about if the students felt handicapped by eXtreme Programming. When it comes to the question whether the students felt eXtreme Programming supportive, it can be said that 7 out of 10 students answered in the "XP Worked Well" category, one student answered in the "XP Worked Not Well" category and 2 students were undecided. Table 4-12 describes the group tendencies of item 5 by showing the mean values of the whole group.

Question	Mean value
Do you think the project would have worked out better without XP?	-0,9
How well do you think XP worked in your project?	-0,36

Table 4-12 The mean value of item 5

As can be seen in Table 4-12 the mean values are respectively in the category "No / XP Worked Well". That can be interpreted as a reasonable success since the response on the first question leads to the conclusion that the students did not feel eXtreme Programming as a constraint. Furthermore - since the mean value of the second question is in the "XP Worked Well" category as well – the second question can be interpreted that the students felt eXtreme Programming as supportive.

Item 6 Do you think you would have worked in a different way if you had not been observed by us?

This item aims in appraising the influence the observers had on the student group. Therefore, the students were asked if they felt any influence or if they think they have behaved in a different way. The response of this item is that 100 percent of the students did not feel influenced in their behaviour. This response leads to the conclusion that the risk of influence can be neglected.

Item 7 Would you apply eXtreme Programming practices in the future?

The purpose of item 7 is to check the learning success of the students. Therefore, it consists of one question followed by a request. The question checks if the students will apply eXtreme Programming practices in future projects. The instruction offered the possibility to the students to respond in free text which practices the students think they will apply in future projects.

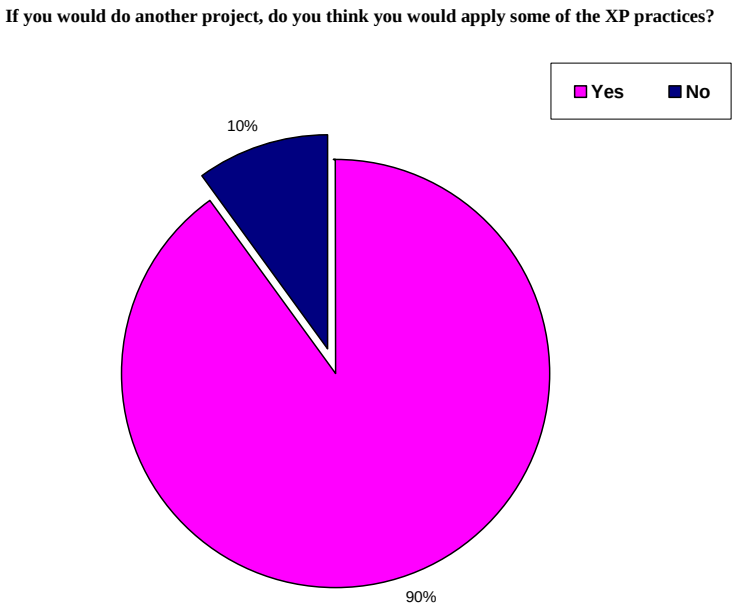

Figure 4.10 Responses on item 7

As can be seen in Figure 4.10 90 percent of the students responded that they would use eXtreme Programming practices in future projects. One response was in the “Neutral" category but according to the framework of this item this response must be counted as failure.

Based on this data and the data from the practices listed by the students, Table 4-13 has been derived.

#[2]	Practices	Summary[3]	Result[1]
1	Sit Together	56%	0,40
2	Informative Workspace	44%	0,30
3	Energized Work	56%	0,40
4	Pair Programming	78%	0,60
5	Stories	33%	0,20
6	Weekly Cycle	33%	0,20
7	Slack	33%	0,20
8	Ten-Minute Build	44%	0,30
9	Continuous Integration	33%	0,20
10	Test-First Programming	44%	0,30
11	Incremental Design	33%	0,20
12	Shared Code	44%	0,30
13	Code & Test	33%	0,20
14	Single Code Base	44%	0,30
15	Negotiated Scope Contract	33%	0,20
16	Whole Team	33%	
17	Quarterly Cycle	33%	
18	Real Customer Involvement	33%	
19	Incremental Deployment	33%	
20	Team Continuity	33%	
21	Shrinking Teams	33%	
22	Root-Cause Analysis	33%	
23	Daily Deployment	33%	
24	Pay-Per-Use	33%	

Table 4-13 Responses on item 7

As can be seen in Table 4-13, the responses provided by the students in the list in this item are not as enthusiastic as on the first question of this item. The only practice that reached a good result is Pair Programming. It is interesting to see that even practices - not applied during the practical project - were listed in the response. The outcome of this item is that nearly all students want to apply eXtreme Programming practices in future projects but most of the students are not sure – except of Pair Programming - what practices they are going to

[1] The values in the Result column are calculated as follows: for every student that has answered with Yes a (+1) is listed for every student that has not responded a 0 is added and for every student who has responded with No a (-1) is added. At the end the mean value has been calculated and is presented.

[2] The fields that are coloured orange represent the practices that were used in the student project and the fields that are coloured in yellow represent the practices that were not used in the student project.

[3]The percent values are based on 9 students that have agreed on applying eXtreme Programming in future projects.

apply. This leads to the conclusion that the students were not convinced of the advantages of most practices except for Pair Programming.

Item 8 How do you like the idea of having theoretical sessions on XP followed by performing the theory into practical use?

This item consisted of the one question asked in both surveys. Therefore, the response of this item is compared with item 5 of Survey 1. Since the surveys were anonymous, it is not possible to compare the answers for each individual student and how the student's attitude towards the structure of the course changed. Therefore, the median, the modal value and the mean value are compared. The mean value shows whether the student group's average opinion changed, the median shows where exactly the responses are divided since the mean does include the outliers in a not transparent way. The modal value shows which answers have mostly been given. The value inside the brackets shows how often this response has been occurred.

Item	Modal value	Median	Mean value
Survey 1	2[5]	2	2,3
Survey 2	2[3], 3[3]	2,5	2,5

Table 4-14 Comparing Survey 1 with Survey 2

As can be seen in Table 4-14, from the first survey to the second survey all values moved from the well side (1 – 2.5) towards the not well side (3.5 – 5) but stayed in the well category. The median and mean values are on the borderline to neutral but still on the well side. However, the modal value is 2 and 3. The fact that the responses are worse in all 3 results of Survey 2 than in Survey 1 can be seen as evidence that the students' enthusiasm was less after the practical project had been performed. This can be understood as a sign to improve the practical project since the students were expected to like the structure even more after they have performed the project.

4.5.3 Conclusion

The purpose of this section was to clarify many issues regarding the students' attitude towards the project, eXtreme Programming in the project and eXtreme Programming in a student environment. It was meant to see whether the students' attitude towards the structure of the course changed. It was also meant to evaluate the level of influence the students felt in being observed during the practical project. The last issue to be verified by Survey 2 (Item 7) was an indicator of how well the goal of the course succeeded i.e. the use of eXtreme programming as a tool.

About the exit criteria, it can be said that the project went well from the students' point of view. The response of the appropriate items was better than well. That leads to the conclusion that the students were relatively positive regarding the progress of the project. This could be influenced by the fact that the student group finished the given task. This positive enthusiasm continued in the next item number 2 regarding the students' opinion towards eXtreme Programming in the project. In this step, the students responded not only that they did not feel limited while they had to apply the eXtreme Programming practices. In fact, the students indicated that eXtreme Programming worked well in their project. This can be understood as success since the students had a positive attitude towards eXtreme Programming. The last issue in this context was the issue regarding applying eXtreme Programming in a student environment. This was asked in item 3 since the students might dislike some practices because the students thought that these practices were not applicable in a student environment. The students' response about this item number 3 was that they still believed eXtreme Programming was applicable in a student environment. The fact that the response was on the borderline to the "Neutral" category should be understood as an opportunity for the lecturers to implement some improvements and on that way increase the efficiency of the course. Finally, it can be said that the student group finished the practical project with a positive attitude the relevant issues project history, the support of eXtreme Programming in the project and the appliance of eXtreme Programming in a student environment.

Another issue to be verified by this section is if the students changed their minds regarding the structure of the course while performing the practical project. An item has been set up that was asked in Survey 1 (item 5) and again in Survey 2 (item 8) to evaluate this question The issue is clarified by checking whether the attitudes became better, worse or remained the same compared with Survey 1. Before the students performed the practical project the students' response regarding this point was that they liked the structure of the course well. After the students had performed the practical project the students responded that they still like the structure of the course with respect to the theoretical sessions. This indicates that the students did not strongly change their mind. Not strongly since the category did not change but the actual values were higher (i.e. worse). This must not be seen as a bad sign but might be an indicator that there is still potential left to improve the structure of the course.

It is positive to see that the students did not feel influenced by the observers while they performed the practical project. Since 100 percent of the students excluded the influence there is no need to consider an influence on the evaluation.

In addition, Survey 2 serves to prove the hypothesis H_{S21} in order to see if the students were aware of the practices they applied or not. However with regard to this point, some of the tool based support indicated that the students did not do what they said they did. Unfortunately, this hypothesis could not been proven for all 15 practices which means that the students applied some practices but were not fully aware that they applied them. In some cases, the student group did not apply practices but thought that they applied them. This is a very tricky situation and needs to be analyzed later, when the overall result is analysed. Many misunderstandings could cause such an effect. Just to mention some: the students could have had trouble in deeply understanding the practice or the students interpreted the practice different than explained.

The last issue to be tracked by Survey 2 was whether or not the course achieved its goals. One of the purposes of the course was to open the students' mind to eXtreme Programming and to enable the students to apply eXtreme Programming and its practices whenever it might be useful. The students have been asked if they could imagine applying eXtreme Programming practices in future projects or not to be able to verify this issue. Ninety percent of the students responded that they would apply eXtreme Programming practices in future projects. The student group has also been asked which practices they would apply in future projects to be able to evaluate where improvements might be useful. The most favoured practice was Pair Programming followed by Sit Together and Energized work. These were followed by Informative Workspace, Ten Minute Build, Test First Programming, Shared Code and Single Code Base. Thus, there could be a chance to improve the education for the other practices so that the level of acceptance increases.

Finally, the result of Survey 2 can be understood as a reasonable success. It delivers a lot of evidence that supports the process of drawing the large picture.

4.6 Summary

During this chapter, all the data sources have been evaluated. The data collected has been presented, commented and interpreted. The data has become evidence to prove certain items that are important to draw the big picture in chapter 5.

Before drawing any conclusions from the data, the conclusions of Survey 1 and Survey 2 need to be summarised.

Survey 1 shows that the students who attended the lectures understood most of the practices well but that the students did not think that all practices were applicable in the practical project. The reason behind this attitude was not the fact that the students did not understand the practices, since they had understood some of them well.

The observation and the tools showed that it is not possible to substantiate a connection between the students' attitude about how well a practice is applicable before the project started.

Survey 2 showed that the students had a positive attitude towards eXtreme Programming and its performance in a student environment. The students left the practical project with a positive impression about how the project worked out. Survey 2 strengthens the observation since the students dispelled all doubts about the influence the observers had on them. It showed that the student groups were not fully aware about which practices they applied and which practices they did not apply. That has been interpreted as evidence that the group was lacking a deeper understanding of certain practices. The majority of the group wants to apply eXtreme Programming practices in future projects although most of the students were not able to mention which practices. The answer to S1Q33 (2.3) and S2Q28 (2.5) (the same question) showed that the students were less enthusiastic to the idea after the practical project but nonetheless the answer in Survey 2 was still in the “well” category but borderline “neutral”.

5 Result by practices

5.1 Introduction

Though, the students left the course with a positive attitude towards eXtreme Programming and the eXtreme Programming practices and a good understanding of the majority of the practices, there are some improvements that can be introduced in future courses. So, this chapter draws a short conclusion for each practice combined with a comparison to other scientific work done regarding the appropriate practice by different researchers. Furthermore, an outlook that could be interesting to be researched in future projects is given.

5.2 Sit Together

Sit Together can be seen as an important pillar of the eXtreme Programming portfolio. However, on the other hand according to Trampel [42] offshore IT projects are twenty to thirty percent cheaper than near shore projects and this is according to Deloitte & Touche [43] a contributing factor for 82 percent of the companies that apply offshore development. This is one reason why research about distributed eXtreme Programming is done as described by Kircher, Corsaro, Levine [44] and Braithwaite, Joyce [45]. It appears to be clear that Sit Together would be heavily affected in a distributed environment. This leads to the conclusion that Sit Together needs to be adapted to be able to be applied in other contexts. Therefore, more research will be performed in future projects. The experience found in this project can be summarized that Sit Together has been transferred to the students successfully. The students know in detail what Sit Together is about and how they have to apply this practice. Thus, this practice has been taught successfully.

The outcome of the practical project shows that the benefit of Sit Together is easy to understand and to apply in a project environment. For future research it would be interesting to apply the changes of eXtreme Programming to become distributed eXtreme Programming described, developed and tested by Kircher, Corsaro, Levine [44] and Braithwaite, Joyce [45]. Finding good applicable solutions for offshore projects is a very interesting field since

according to Deloitte & Touche [43] offshore development will increase within the next decades and should to be taught to computer science students as early as possible.

5.3 Informative Workspace

This practice, Informative Workspace, is a good example of a lack of information. The students thought that they understood this practice well but considered it as not applicable. Contrary to the observation protocol that shows that the student group applied this practice to the full extent possible. However, after the practical project, the students were still not aware of this as the response in Survey 2 was exactly the same level as before. This might account for a lack of awareness about this practice on the part of the students. Further emphasis should be placed on describing this practice in order to increase the students' awareness.

For the same reasons - mentioned in the section about Sit Together – attempts have been made to automate this practice in real world projects. One solution is therefore the tool XPSwiki [47] that has been applied in eXtreme Programming research projects Gianini & Sillitti [46] to analyze if it is possible to apply Informative Workspace in distributed project environments. The approach of automating Informative Workspace can be motivated by several reasons. One reason is that an Informative Workspace that is virtual is totally independent of a physical location and therefore a member of a team can work from anywhere in the world. It was somewhat ironic that the students' project [19] was to implement a virtual story wall which could be a part of a tool to support distributed Informative Workspace. It would be interesting to see whether future courses would be able to work with virtual Workspaces.

5.4 Energized Work

Energized Work is not only an issue in software development projects it is also a social problem. As analyses in Canada [48] and Germany [49] show, it has been shown that increasing working hours have negative influence on the employee's health and by this on the quality of the employee's working results.

The case study showed that the students understood this practice well, applied it during the project and consider it as applicable in future projects. For future research, it would be interesting to evaluate if a student group that applied Energized Work feels less stressed and

healthier after a project has been finished in comparison to a student group that performed the same project with the same goal but without intentionally applying Energized Work.

5.5 Pair Programming

Pair Programming showed the highest score regarding the level of understanding. Based on this, the students categorized it as highly applicable. However, this was followed by a nearly total absence of application which the students have not been aware of. In the end, Pair Programming is the leading practice when it comes to the question which practice to apply in future projects. What could be a reason for this result? One possible answer is that – according to Ramachandran & Shukla [71] - Pair Programming is one of the 3 most popular practices of all eXtreme Programming practices. This could play its part as well as the lack of knowledge about the details regarding the appliance of this practice. Perhaps the students thought that sitting together while coding is already Pair Programming. This issue raises more questions than can be answered with help of the data gathered in this project. However, answers to these questions may be given in future projects that place the emphasis on such distinctions.

It should be noted about Pair Programming, that much research exists on whether Pair Programming improves quality of code enough to legitimise the cost intensive practice or not. As described by Williams, Kessler, Cunningham & Jeffries [72], Pair Programming helps to produce a better software quality in less time than with the common one programmer coding method. This is supported by Cockburn & Williams [76] who showed 3 projects with fewer lines of code but with the same functionality. The fact that all 3 projects needed fewer lines of code leads to the conclusion that Pair Programming results in a much more sophisticated design. Furthermore Williams & Upchurch [73], Nagappan, Williams, Ferzli, Wiebe, Yang, Miller & Balik [78] and Williams [75] describe that Pair Programming can be supportive in a students' course as well. They mentioned that the students had greater learning experience in a shorter time. Nagappan, Williams, Ferzli, Wiebe, Yang, Miller & Balik [78] summarize it as follows:

- "Pair programming helps in the retention of more students in the introductory computer science stream.
- Students in paired labs have a more positive attitude toward working in collaborative environments; this should ultimately help the student in his/her professional life.

- Pair programming in an academic environment reduces the burden on the LI[1] because the pairs helped each other, enabling the LI to perform more efficiently.
- From the results we have obtained regarding the tests and the projects, we can conclude significantly that pair programming among students is in no way a deterrent to student performance."

Another interesting observation has been made by Williams & Kessler [74], Williams & Kessler [77], who noted the fact that programmers are usually used to working alone. This needs to be mentioned while talking about Pair Programming since for some programmers it might be difficult to work in pairs. Williams & Kessler come to the conclusion that most programmers enjoy working in pairs after they committed themselves to doing so. Pair Programming offers possibilities for all circumstances e. g. partner picking principles (see [79]). Finally, advantages of pair programming can be summarized as follows (see [76]):

- "many mistakes get caught as they are being typed in rather than in QA test or in the field (continuous code reviews);
- the end defect content is statistically lower (continuous code reviews);
- the designs are better and code length shorter (ongoing brainstorming and pair relaying);
- the team solves problems faster (pair relaying);
- the people learn significantly more, about the system and about software development (line of-sight learning);
- the project ends up with multiple people understanding each piece of the system;
- the people learn to work together and talk more often together, giving better information flow and team dynamics;
- people enjoy their work more;"

5.6 Stories

More than all other practices Stories had to be internalized by the student group as the goal of the student project (appendix B) was that the students had to build a virtual story wall with virtual story cards on it. This practice has been understood well and has been well performed. However, after the project, the students thought that they applied this practice less than well.

[1] LI stand for Lab Instructors. See [78] for more details.

It is not obvious why the students changed their minds regarding this practice. One possible reason could be that the students developed a deeper sense for this practice since the goal of the students' project was to develop a tool that represented a virtual story board.

The practice Stories is – according to Beck and Fowler [51] - an interesting and completely different way of gathering Use Cases. This comparison has been also drawn by Paulk [50] who compared or mapped eXtreme Programming to the Capability Maturity Model (CMM) and the practice Stories to the CMM Level 2, Requirements Management. For future research it could be very interesting if the different style a story has from a Use Case – according to the UML [52] definition of Use Case – makes it easier to fully gather requirements. Therefore, metrics for good requirements need to be defined and 2 project teams could gather requirements in different ways.

5.7 Weekly Cycle

Based on the level of understanding the students had towards this practice, they should have been able to apply it. However, during the practical project, the practice Weekly Cycle had just been performed in one week. After the practical project was finished, the students' feeling about their level of appliance was still higher than the value that has been measured in the observation. One possible conclusion drawn on the given result is that the students thought applying this practice was performing the project in 5 weeks.

Weekly Cycle can also be seen as responsible for one very interesting effect eXtreme Programming has on the project group. It helps the team members – who consist in this particular case of students - in a project to move the emphasis away from the deliverables and by doing this, aim for the goal of the project, see Noll & Atkinson [53]. Another very interesting result was obtained by Abrahamsson & Koskela [54] who adapted the development cycles[1] to the project flow. During the first 3 weeks the cycle was on a 2 weeks basis and for the last 2 weeks it has been applied on a weekly basis. In the last part of the project the developed LOC decreased whereas the post release defect rate increased. This is a very interesting approach to be able to fix bugs found in the post release stage. An interesting future task could be to compare the 2 different project workflows. One of the projects with

[1] According to the Rational Unified Process [83] a development cycle consists of the four phases: Requirements, Analysis & Design, Implementation, Test and Deployment. In iterative software development this cycle needs to be repeated as many times as needed until the deployments meets the final requirements of the customer.

fixed iteration cycles of 1 or 2 weeks and the other one with adapting intervals like in Abrahamsson & Koskela [54].

5.8 Slack

There has been a special attention on this practice as it is one of the 2 practices the students considered as not understood. Maybe the majority of the students did really not understand this practice and this caused the low expectation regarding the application in the practical project. Surprisingly, as arose during the practical project, this practice has been applied to its full extent. As can be seen in appendix D, the students were, at the beginning of the project, not fully aware of that fact. This raises the possibility of assuming that some students understood this practice, performed it and by this taught it to the rest of the group. This is the way the information should flow and why the practical project was set up in the first place. The group explains the practices to itself. A detailed lecture on Slack should be given in future lessons since it is important that enough students have a deeper sense of understanding when the whole group leaves the course, knowing the advantages of the practice Slack.

Slack can become very important in the release planning process as described by McDaid, Greer, Keenan, Prior, Taylor & Coleman [55]. It is useful to add slack but it is also important to not plan too much slack per release so as to be able to guarantee a commitment from the developers. McDaid, Greer, Keenan, Prior, Taylor & Coleman [55] say that up to 70% of the scope, planned for one release is usually realized. This leads to the conclusion that Slack supports the productivity since it puts more pressure on the operative project team. Anderson [56] goes a bit further and constructs the following scenario: A Design-Test-Unit is able to test ten units per day. According to bottlenecks in the previous process steps for one week only thirty testable units exist. One possibility could be to reduce the group to 6 Test-Units per day to perform the units in one week. The second approach, which is recommended by Anderson, is to stay in the ten Test-Unit mode and give the team 2 days off. He justifies this approach by the explanation that the team might develop problems in the future when it gets used to the 6 Test-Units per day.

McDaid, Greer, Keenan, Prior, Taylor & Coleman [55] as well as Anderson [56] offer very interesting approaches but should be analyzed much more detailed. An interesting point to be analyzed would be if a group becomes more efficient when the pressure is increased and if yes when the maximum level of pressure is reached. The maximum load in this case can be seen as the point where a resignation arises in the project team.

5.9 Ten-Minute Build

This practice has been understood well by the students. This might have contributed the good attitude towards applying Ten-Minute Build during the practical project, justified by its full appliance caused by the auto build function provided by Eclipse. The fact that the students had nothing to do on their own to apply this practice, made applying this practice passive, and thus easier, and by this it moved out of the students' minds. From the author's point of view, it could be discussed whether applying this practice makes sense or not. A larger project would be needed to exceed the given time period of ten minutes but that could not be managed within 5 weeks.

5.10 Continuous Integration

As the section about Survey 1 has shown, the students understood this eXtreme Programming practice well and performed it to its full extent during the practical project. However, the acceptance level is quite low. There are different reasons to explain this outcome. Firstly, it can be seen as a problem in understanding the benefit of this practice. Secondly, there could be a lack of knowledge as this practice could not be performed 100% due to the student environment. Excluding this eXtreme Programming practice from the practical project or enlarging the project so that it makes more sense to apply it could be possible solutions to solve that problem.

According to Fowler & Foemmel [57], every time source code is committed to the code base a complete build needs to be performed. In the paper a complete build is defined as:

- All the latest sources are checked out of the configuration management system.
- Every file is compiled from scratch.
- The resulting object files (Java classes in our case) are linked and deployed for execution (put into jars).
- The system is started and suite of tests (in our case, around 150 test classes) is run against the system.

All these tasks can be performed by an automated build server like Apache ANT[1], Apache Maven[1] [58] or a continuous integration server as mentioned in Fowler [21] or a sanitized build machine as mentioned in Appleton, Konieczka & Berczuk [22]. However in the

[1] See http://ant.apache.org/ for more information about Apache ANT

particular student project used in this study, that would have exceeded the scope of the project. The students had to learn a number of new methods and technologies and if the students would have had to learn how to set up ANT or Maven to do automate build-jobs – including system tests – it would have been too many tasks for the given time.

5.11 Test-First Programming

Even though the students were not sure if they will be able to apply this practice during the practical project they applied it well at the end and consider it as well applicable in future projects. Therefore, the lecture on Test-First Programming can be seen as an absolute success. It managed to convey the benefit of writing test cases before writing the code.

According to Kaufmann & Janzen [59], Test-First Programming and Test-Driven-Development (TDD) can be seen as comparable or familiar. Unfortunately, they do not come to a clear conclusion if a project applying TDD or Test-First Programming is better than a project without applying TDD or Test-First Programming. A completely different conclusion was drawn by George & Williams [60] who found out that applying TDD needs more time but delivers better quality. There is still potential for more research in this context, since clarifying if Test-First Programming delivers better software quality while investing the same resources or not would be very interesting.

5.12 Incremental Design

The study shows that this practice can be seen as success. The only response – given by the students – that clouds the expectations is that only very few students are willing to apply Incremental Design in future projects.

In the first version of eXtreme Programming, this practice was part of the practice called Planning Game. According to Williams & Upchurch [61], Planning Game is - especially in the educational context - very useful for the students to reflect their last delivery and to receive feedback. Incremental Design is an essential part of the Planning Game since it is responsible for the implementation made during the following iteration. Another point regarding incremental design is mentioned by Müller & Tichy [62]. They state that it is hard to follow this practice in huge project groups which occur in real world software projects. As

[1] See http://maven.apache.org/ for more information about Apache Maven

mentioned in Beck [2], twelve team members shall be in one eXtreme Programming project team which might be a bit less for major software projects. Müller & Tichy [62] split the team up so that some worked on the design whereas others worked on other parts of the incremental process. This is not the basic idea of eXtreme Programming but is applied as solution for the problems mentioned by them Müller & Tichy [62]. It could be interesting in this field to establish a model that enables large project groups to apply eXtreme Programming efficiently.

5.13 Shared Code

The results of the study indicate that the students understood this practice, considered it as applicable, applied it and were absolutely aware of the fact that they had applied it. Some of the students (4 out of 9) considered it as applicable in future projects. This leads to the conclusion that the students have not been fully convinced about the advantages offered by Shared Code.

As mentioned before, distributed eXtreme Programming becomes more and more interesting according to the growing popularity of offshore development, projects with greater complexity and increasing energy costs. Schümmer & Schümmer [63] mention that Shared Code does not appear to be a problem regarding the distribution of a project team. Tools like CVS that have been applied in the current project are developed to enable project teams to work on the same source code from all over the world. Watkins [64] gives a good and detailed overview about the state of the art systems for version control systems.

5.14 Code & Test

This practice is kind of interesting. The students understood this practice well and considered it as applicable. However, this practice could not been performed by the group since the project specification forced the students to break this practice so that it was impossible for the students to apply Code & Test during the practical project. Strangely, the students did not apply Code & Test but were not sure about how well they applied it. Several points regarding this evaluation show that the students did not develop a deeper understanding of this practice. Firstly, they thought that it would be applicable in the practical project. One reason could be that the students did not really study the requirements of the project. Secondly, the students have not been aware of the situation that they did not apply it. This is represented by the movement from a good value in the well section to the neutral section. The

last point to mention is that the level of future appliance decreased to a value in the neutral section which appears to be absolutely appropriate.

According to Card, McGarry & Page [65], Lientz & Swanson [66] and Rombach & Basili [67], software documentation is a key component of software quality. Software documentation in this case means technical documentation and not end user documentation. The studies show that documentation that is poor, out of date or completely missing is a major cause for problems in software maintenance and development. Cook & Visconti [68] goes somewhat further by showing that working on a higher documentation level decreases the amount of defects contained in the software. These studies just take up partially a contrary position to the practice Code & Test. The studies reveal that a number of defects are already included while the requirements are captured and the design is defined. eXtreme Programming prevents this by offering the practices Real Customer Involvement and Incremental Design. It would be interesting to see a direct comparison between eXtreme Programming and the maturity process mentioned by Cook & Visconti [68]. This comparison would probably exceed the possibilities of a university project but could be realized with the support of a business partner.

5.15 Single Code Base

Single Code Base reflected a major aim of the course, since there was no possibility for multiple code bases. When the students finished the lecture, they were not sure how well they understood Single Code Base and how well they think Single Code Base would be applicable. During the practical project, the students seemed to realize that there was no way for them to break this practice since they were not able to start another code bases. This was a result of the fact that they only had one project and no administrative access to the CVS server. After the project was finished, this learning procedure was reflected in their responses as they considered the application of Single Code Base as very well. This revealed itself when the students responded; that Single Code Base is considered to be applicable in future projects.

For the future labs it can be said that the teachers should place the emphasis on showing the advantages of Single Code Base so that the students are aware of these advantages. This is motivated by the result that shows that the majority (5 out of 9) of the students are not going to apply Single Code Base in future projects. One reason for this result could be that the students did not develop a deeper understanding for the practice since they had no choice.

5.16 Negotiated Scope Contract

Negotiated Scope Contract can be seen as a total failure. After the lecture, none of the practices had such a low level of understanding as Negotiated Scope Contract. This led to the result that the students considered this practice not as well applicable. During the project, the students applied it only during some weeks. After finishing, the students did not even see their failure and voted Negotiated Scope Contract as well applied. The only gleam of hope is that the students did not really feel interested in applying this practice in future projects.

In the case of Negotiated Scope Contract the lecturers should rethink the methods and put the emphasis more on the benefit of Negotiated Scope Contract to enable the students – technically and mentally - to apply this practice in future projects.

Negotiated Scope Contract is a very interesting practice since Beck[3] says that the major difference in applying Negotiated Scope Contract is, that the customer gets what he wants at the end of a project instead of getting it at the beginning of a project. According to Coldewey [69], the practice Negotiated Scope Contract delivers the kind of standard contracts that capture the essence of how an agile process runs. This can be understood in the way that this practice is absolutely matching the idea behind agile software development. From a completely different point of view Negotiated Scope Contract is discussed by Favaro & Robertson [70] who say that "…the purpose of the requirements process should not be to "cover all eventualities," or to "limit the damage," or to "minimize risk," or even to "satisfy the customer." The purpose of the requirements process is to add business value." It delivers Negotiated Scope Contract as a solution for that problem. It includes the customer into the development process instead of capturing all relevant data before the design is started. In that way the customer is able to change items without great impact.

It would be interesting to compare 2 software projects and the level of satisfaction the customer has at the end. One project should be performed in the common way by defining all requirements in the beginning and then build the artefacts. The other project should be performed following the eXtreme Programming practices. The requestor should be the same and the projects should have about the same complexity to be able to compare them. At the end, the timeline, costs, usability, quality and customer satisfaction should be measured.

5.17 Final Comments

In a world where requirements become more and more complex while lifecycles become shorter and shorter software development processes are needed that are able to adapt to

changing requirements quickly and efficiently while guaranteeing good quality and maintainability. According to Kent Beck, eXtreme Programming is one answer. Even if eXtreme Programming as a whole might be difficult to apply for some companies it delivers a tool box to meet the challenges of our decade. However, this tool box needs to be under permanent evolution in order to deliver tools and answers to new and frequently upcoming questions. Therefore, it is challenged by researchers all around the world. As has been showed in this thesis, researchers are even working on the issue of offshore development which is clearly in opposition to some practices of eXtreme Programming.

It is up to the universities to enable software developers, requirement engineers, test managers, designers and project leaders to be able to apply eXtreme Programming in their future projects. Motivated by this task this thesis traces if eXtreme Programming can already be taught to students at the university and how to arrange it as efficiently as possible. As a result, this thesis delivers the proof that it is possible and worth to open students' minds to different approaches of software development and to give them the possibility to choose the fitting approach for every single challenge that might come up during their professional careers.

Nevertheless, this case study revealed that not all practices of eXtreme Programming are suitable for university projects. Whereas the practices: *Whole Team, Quarterly Cycle, Ten-Minute Build, Continuous Integration, Real Customer Involvement, Incremental Deployment, Team Continuity, Shrinking Teams, Root-Cause Analysis, Code & Test, Single Code Base, Daily Deployment* and *Negotiated Scope Contract* are not ideal for practical university projects, the practices *Sit Together, Energized Work, Pair Programming, Informative Workspace, Stories, Weekly Cycle, Slack, Test-First Programming, Incremental Design* and *Shared Code* are very well applicable in practical university projects. Therefore, the practices not qualified for practical projects should be emphasised in the theoretical lectures.

Finally can be said that teaching eXtreme Programming in a practical project is a very interesting approach and it will show whether it supports the way of eXtreme Programming into software development projects or not.

References

[1] *Kent Beck. eXtreme Programming Explained. Addison-Wesley, First edition, 1999*

[2] *Kent Beck. eXtreme Programming Explained: Embrace Change. Addison-Wesley, Second edition, 2004*

[3] *Kent Beck First Class Software Dave Cleal. Optional Scope Contracts 2003*

[4] *Stefan Edlech and Martin Backschat. J2EE Entwicklung mit Open Source Tools. Spektrum Akademischer Verlag, First edition, 2004*

[5] *David MacKenzie, Paul Eggert, and Richard Stallman. Comparing and Merging Files. 2.8.1 edition, April 2002*

[6] *Sharan B Merriam. Fallstudien som forskningsmetod. Studentlitteratur, 1994*

[7] *Winston Tellis. Introduction to case study. The Qualitative Report, vol. 3, 1997*

[8] *Malcolm Gladwell, The Tipping Point: How Little Things Can Make a Big Difference. Back Bay Books, 2002*

[9] Jeffrey Barnes, Kerri Conrad, Christof Demont-Heinrich, Mary Graziano, Dawn Kowalski, Jamie Neufeld, Jen Zamora, and Mike Palmquist. (2005). Generalizability and Transferability. Writing@CSU. Colorado State University Department of English. Retrieved 03/2010 from http://writing.colostate.edu/guides/research/gentrans/

[10] eclipse.org The Eclipse project's webside http://www.eclipse.org (link worked 02.2008)

[11] Die Eclipse-Architektur http://www.fh-wedel.de/~si/seminare/ws02/Ausarbeitung/ 7.eclipse/ eclipse2.htm (link worked 02.2008)

[12] The JUnit Framework and all it's components can be found under: http://www.junit.org (link worked 02.2008)

[13] junit.2.png http://www.adictosaltrabajo.com/tutoriales/junit/junit.2.jpg (link worked 02.2008)

[14] junit.png http://tejasconsulting.com/images/junit.png (link worked 02.2008)

[15] CVS - Concurrent Versions System http://www.gnu.org/software/cvs/ (link worked 02.2008)

[16] Online Catalog: CVS Pocket Reference http://www.oreilly.com/catalog/cvspr/ (link worked 02.2008)

[17] GNU http://cssc.sourceforge.net/ (link worked 02.2008)

[18] statcvs http://statcvs.sourceforge.net/ (link worked 02.2008)

[19] CIT B01 - Projektinformation http://www.cs.kau.se/cs/education/courses/citb01/vt2005 /index.php?projectinfo=true (link worked 04.2005) Added to the appendix B

[20] List of Extreme Programming projects around the world http://c2.com/cgi/wiki?ExtremeProgrammingProjects (link worked 02.2008)

[21] Continuous Integration by Martin Fowler http://www.martinfowler.com/articles/continuousIntegration.html (link worked 04.2008)

[22] Brad Appleton, Steve Konieczka & Steve Berczuk Continuous Integration – Just another buzz word? http://www.cmcrossroads.com/articles/agilesep03.pdf (link worked 04.2008)

[23] Robert K. Yin Case Study Research: Sage Publications, Inc; Third Edition (18 Feb 2003)

[24] Jürgen Bortz & Nicola Döring; Forschungsmethoden und Evaluation; Springer; 4. Auflage 2006

[25] Wilfried Laatz; Empirische Methoden: Ein Lehrbuch für Sozialwissenschaftler; Harri Deutsch; 1973

[26] Karlheinz Ingenkamp, Evelore Parey & Lothar Tent; Schätzen und Messen in der Unterrichtsforschung; Beltz; 1984

[27] Gerhard Faßnacht; Systematische Verhaltensbeobachtung; UTB für Wissenschaft; 1979

[28] Urs Haeberlin; Wortschatz und Sozialstruktur; Benziger Verlag GmbH; 1982

[29] Randall C. Wyatt & Lawrence S. Meyers; Psychometric Properties of Four 5-Point Likert Type Response Scales ; SAGE Publications; 1987

[30] Bernd Rohrmann; Empirische Studien zur Entwicklung von Antwortskalen für die sozialwissenschaftliche Forschung ; Zeitschrift für Sozialpsychologie 9, 222-45 ; 1978

[31] Rensis Likert; A technique for the measurement of attitudes; Archives of Psychology. 1932 Vol 22 No. 140 55; 1932

[32] Martin E. Grosse & Benjamin D. Wright; Validity and Reliability of True-False Tests ; SAGE Publications; 1985

[33] Lewis R. Aiken; Formulas for Equating Ratings on Different Scales; Educational and Psychological Measurement; 1987

[34] Peter H. Rossi & Howard E. Freeman; Evaluation: A Systematic Approach; SAGE Publications Inc; Edition 5th (14. March 1993)

[35] Carol H. Weiss; Evaluierungsforschung. Methoden zur Einschätzung von sozialen Reformprogrammen; VS Verlag für Sozialwissenschaften; 1974

[36] Werner W. Wittmann; Evaluationsforschung. Aufgaben, Probleme und Anwendungen; Springer-Verlag GmbH; 1985

[37] Heike Thierau & Heinrich Wottawa; Lehrbuch Evaluation; Huber, Bern; 3. Edition (October 2003)

[38] Donna M. Mertens in Reinhard Stockmann (Herausgeber); Evaluationsforschung: Grundlagen und ausgewählte Forschungsfelder (Institutionalizing Evaluation in the United States of America); Waxmann; Auflage: 3. (Dezember 2006)

[39] Frans L. Leeuw in Reinhard Stockmann (Herausgeber); Evaluationsforschung: Grundlagen und ausgewählte Forschungsfelder (Evaluation in Europe); Waxmann; Auflage: 3. A. (Dezember 2006)

[40] Reinhard Stockmann (Herausgeber & Autor); Evaluationsforschung: Grundlagen und ausgewählte Forschungsfelder (Evaluation in Deutschland); Waxmann; Auflage: 3. A. (Dezember 2006)

[41] Adler P. & Adler P. in Norman K. Denzin & Yvonna S. Lincoln (publisher) ; The Sage Handbook of Qualitative Research (Observational Techniques) ; Sage Pubn Inc; Third.Edition (27. April 2005)

[42] Julia Trampel; Offshoring oder Nearshoring von IT-Dienstleistungen? - Eine transaktionskostentheoretische Analyze; ARBEITSPAPIERE des Instituts für Genossenschaftswesen der Westfälischen Wilhelms-Universität Münster; Nr 39 März 2004

[43] Deloitte & Touche UK LLP; Outsourcing und Offshoring mit indischen IT-Unternehmen - Die IT-Welt im Wandel; Deloitte & Touche UK LLP; 2003

[44] Michael Kircher, Prashant Jain Angelo Corsaro, David Levine; Distributed eXtreme Programming; Second international conference on eXtreme Programming and Agile Processes in Software Engineering; 2001

[45] Keith Braithwaite and Tim Joyce; XP Expanded: Distributed Extreme Programming; In XP 2005: Proceedings of the 6th International Conference on Extreme Programming and Agile Processes in Software Engineering

[46] Gabriele Gianini, Alberto Sillitti; Sharing experiences on agile methodologies in open source software developmen; Polimetrica s.a.s.; 2006

[47] Sandro Pinna, Simone Mauri, Paolo Lorrai, Michele Marchesi and Nicola Serra; XPSwiki: An Agile Tool Supporting the Planning Game; Springer Berlin / Heidelberg; Volume 2675/2003

[48] Margot Shields; Long working hours and health; Canada's National Statistical Agency; March 8, 2000

[49] Alfred Oppolzer; 42 Stunden sind längst üblich – NRW-Umfrage belegt längere und flexible Arbeitszeiten; AiB-Verlag GmbH; Arbeit & Ökologie-Briefe im Jahr 2004 Heft 5/2004

[50] Mark C. Paulk; Extreme Programming from a CMM Perspective; Paper for XP Universe, Raleigh, NC, 23-25 July 2001

[51] Kent Beck, Martin Fowler; Planning Extreme Programming; Addison-Wesley Longman, Amsterdam; Oktober 2000

[52] Unified Modelling Language (UML) http://www.uml.org/ is administrated by the Object Management Group. http://www.omg.org/ Links worked 05.2010

[53] John Noll and Darren C. Atkinson; Comparing Extreme Programming to Traditional Development for Student Projects: A Case Study; Springer Berlin / Heidelberg; Volume 2675/2003

[54] Pekka Abrahamsson and Juha Koskela; Extreme Programming: A Survey of Empirical Data from a Controlled Case Study;ACM-.IEEE International Symposium on Empirical Software Engineering (ISESE 2004), Redondo Beach CA, USA

[55] K. McDaid, D. Greer, F. Keenan, P. Prior, P. Taylor, G. Coleman; Managing Uncertainty in Agile Release Planning; ; Agility in the Software Process 04.2008

[56] David J. Anderson; The Four Roles of Agile Management; Cutter IT Journal July 2004

[57] Martin Fowler, Mathew Foemmel; Continuous Integration; Though Works; 2005

[58] Erik Drolshammer; Improved Backward Compatibility and API Stability with Advanced Continuous Integration; Norwegian University of Science and Technology Department of Computer and Information Science June 2007

[59] Reid Kaufmann, David Janzen; Implications of Test-Driven Development A Pilot Study; OOPSLA 2003

[60] Boby George, Laurie Williams; A Structured Experiment of Test-Driven Development; Information and Software Technology; Volume 46, Issue 5, 15 April 2004, Pages 337-342; Special Issue on Software Engineering, Applications, Practices and Tools from the ACM Symposium on Applied Computing 2003

[61] Laurie Williams and Richard Upchurch; EXTREME PROGRAMMING FOR SOFTWARE ENGINEERING EDUCATION?; 2001 IEEE; October 10 - 13, 2001 Reno, NV

[62] Matthias M. Müller, Walter F. Tichy; Case Study: Extreme Programming in a University Environment; Computer Science Department Universität Karlsruhe; IEEE Computer Society; International Conference on Software Engineering; Proceedings of the 23rd International Conference on Software Engineering; Toronto, Ontario, Canada; Pages: 537 - 544; Year of Publication: 2001

[63] Till Schümmer, Jan Schümmer; Support for Distributed Teams in eXtreme Programming; Addison-Wesley Longman Publishing Co., Inc in Extreme programming examined; Pages: 355 - 377; Year of Publication: 2001

[64] Ellis Rowland Watkins; Trusted Collaboration in Distributed Software Development; UNIVERSITY OF SOUTHAMPTON Faculty of Engineering, Science and Mathematics School of Electronics and Computer Science; June 2007

[65] Card,., McGarry, Frank E. and Page, Gerald T.; Evaluating software engineering technologies; IEEE Transactions on Software Engineering; Vol. Se-13, No. 7, 1987

[66] Bennet P. Lientz, and E. Burton, Swanson; Problems in applications software maintenance; Communications of the ACM; November 1981

[67] Rombach, H.D. and Basili, V.R.; Quantitative assessment of maintenance: an industrial case study; IEEE Proceedings of Conference on Software Maintenance; City Press, 1987

[68] Curtis Cook and Marcello Visconti; DOCUMENTATION IS IMPORTANT; CrossTalk, 1994

[69] Jens Coldewey, Senior Consultant, Cutter Consortium; Contracting Agile Projects; Cutter Consortium; Agile Project Management Advisory Service Executive Update Vol. 7, No. 17 ;2006

[70] John Favaro and Suzanne Robertson; Managing Requirements for Business Value; The Atlantic Systems Guild; 2002

[71] Vinay Ramachandran and Anuja Shukla; Circle of Life, Spiral of Death: Are XP Teams Following the Essential Practices? ; Verlag Springer Berlin / Heidelberg ; Buch Extreme Programming and Agile Methods — XP/Agile Universe 2002 Volume Volume 2418/2002 Pages 173-246

[72] Laurie Williams, Robert R. Kessler, Ward Cunningham and Ron Jeffries; Strengthening the Case for Pair-Programming; Publisher Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA; Year of Publication: 2001; Pages: 223 – 243; The costs and benefits of pair programming

[73] Laurie Williams, Richard L. Upchurch; In support of Student Pair-Programming; Technical Symposium on Computer Science Education; Proceedings of

the thirty-second SIGCSE technical symposium on Computer Science Education; 2001 , Charlotte, North Carolina, United States

[74] Laurie A. Williams, Robert R. Kessler; Experimenting with Industry's "Pair-Programming" Model in the Computer Science Classroom; Computer Science Education, Vol.. 11, No. 1, pp. 7-20, March 2001

[75] L. A. Williams; Pair programming and pair trading: effects on learning and motivation in a CS2 course; Consortium for Computing in Small Colleges; May 2003

[76] Alistair Cockburn and Laurie Williams; The Costs and Benefits of Pair Programming; Addison Wesley, 2001

[77] Laurie A. Williams and Robert R. Kessler; All I Really Need to Know about Pair Programming I Learned In Kindergarten; Communications of the ACM; 1999

[78] Nachiappan Nagappan, Laurie Williams, Miriam Ferzli, Eric Wiebe, Kai Yang, Carol Miller, Suzanne Balik; Improving the CS1 Experience with Pair Programming; SIGCSE'03, February 19-23, 2003

[79] Laurie Williams and Robert Kessler; Pair Programming Illuminated; Addison-Wesley Longman, Amsterdam; (Juli 2002)

[80] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta; Agile software development methods: Review and Analysis; Espoo, Finland: Technical Research Centre of Finland, VTT Publications 478; 2002

[81] http://en.wikipedia.org/wiki/Weighted_mean (Link worked 10.05.2010)

[82] http://en.wikipedia.org/wiki/Likert_scale (Link worked 10.05.2010)

[83] Stefan BergStröm, Lotta Raoberg; Adopting the Rational Unified Process; Addison-Wesley; December 2004P

A Appendix List of abbreviations

- AWT - Abstract Windowing Toolkit
- CMM - Capability Maturity Model®
- CVS - Concurrent Version System
- DXP – Distributed eXtreme Programming
- GUI - Graphical User Interface
- HTML - Hyper Text Markup Language
- IP - Internet Protocol
- IT - Information Technology
- JAR – Java™ Archive
- LOC – Lines Of Code
- RCS - Revision Control System
- RTF – Rich Text Format
- SCCS - Source Code Control System
- SDK - Software Development Kit
- SQL - Structured Query Language
- SWT - Standard Widgeting Toolkit
- TDD - Test-Driven Development
- UML – Unified Modelling Language
- VCM - Version and Configuration Management
- XML - eXtensible Markup Language
- XP - eXtreme Programming

B Appendix The project specification

During this chapter we copied the project specification specified under [19]

Project information

In the project you should (1) develop a software system and (2) continously document the project. The idea of the course is that the students of CITB02 are team of XP programmers at a company DevForYou. The company DevForYou accepted the proposition of another company EduForYou to develop a software system.

Apart from the team of XP programmers, the following people are involved in the project:

Customer (EduForYou) The customer is responsible for the requirements and for the deployment on the customer's equipment (e.g. database system). The customer is represented by Mari Göransson. Please contact her by following this link.

Chef (DevForYou) The chef is responsible for the supervision (e.g. time protocols, steering committee meetings, story card acceptance) and for your equipment (e.g. cvs). The chef is Mari Göransson. Please contact her by following this link.

Web consultant (DevForYou) The web consultant is hired to help the team of XP programmers with their project documentation pages. The consultant is Katarina Asplund. Please contact her by following this link.

Experts Various experts from other parts of DevForYou are available for specific parts of the software system and the documentation respectively. Please contact them by following this link.

Further down on this page you find more information on what and how to develop as well as on what and how to document.

Software development

Process

The management and the programmers of DevForYou have not been happy with previous software development approach and jointly decided to try a recent approach, namely eXtreme Programming (XP). In particular, everybody agreed to follow the following practices: sit together, informative workspace, energized work, pair programming, stories, weekly cycle, slack, ten-minute build, continuous integration, test-first programming, incremental design, shared code, Code & Tests, single code base, and negotiated scope contract.

- Monday (i.e., first day of week):
  Monday is planning day. The team should write story and task cards, distribute them and estimate them. The cards have to be accepted by the customer and the chef. The team visits Mari Göransson on Monday afternoon between 15 and 16 o'clock and gets the cards accepted. The chef provides the team with cards and magnets and the team should put their ”stories on a wall”.
  In addition, on Mondays the team should follow up last week.
- Friday afternoon (i.e., last day of week):
  Friday afternoon is release day. The system is build and the project documentation pages are updated.
- Other working days of the week:
  Each half day the team has a coding session followed by a build of the system.

The management and the programmers of DevForYou discussed the equipment which should be used and arrived to the following agreement. The management provides the eclipse integrated development environment (under MS-Windows). Moreover, the management provides a cvsserver for the code repository. The programmers agree to use eclipse with JUnit and cvs. In addition the programmers agreed to use a common style guide, e.g. Sun's code conventions for Java.

Program

The customer EduForYou requested a client-server-database (see the figure below) software system for the maintenance of XP ”story cards”.

Obviously the system should provide functionality to handle story cards (an example of a story card can be found on page 45 in Beck’s book). The system should provide the following functionality:

- creation of story cards according to a template
- modification of story cards
- deletion of story cards

A story card template defines which fields are available on a story card, where they are located and how big are. Fields are for example: story name, priority, source (author), estimate, description, et cetera. The system should provide the following functionality:

- creation of story card templates
- modification of story card templates
- deletion of story card templates

Story cards should be organized according to the picture on page 40 in Beck’s book. The system should provide the following functionality:

- graphical presentation of the ”wall” (page 40)
- automatic mouse over story card zooming
- manual zooming in and out
- mouse controlled movement of story cards
- automatic update of the wall (e.g. if another user moves a story card)
- creation of story card areas
- modification of story card areas
- deletion of story card areas
- hierarchical story card areas
- overlapping story card areas

The software system should come as close as possible to the ”real thing”, i.e. real paper cards and a real wall.

Apart from the requirements above, the customer as a couple of non-functional requirements concerning the environment in which the software is supposed to operate:

- the software has to be written in Java. The IT support department at EduForYou is familiar with the Java programming language and thus able to maintain the software after delivery.
- the software has to store its data in a mySQL database. The customer EduForYou already has other software using a mySQL database and does not want to maintain several different database systems.

Documentation

The customer is very concerned about the project’s costs with respect to its progress. To be able to easily access the latest builds of the software and the current cost of the project, the project group agreed to provide several web pages that are updated every Friday.

The agreement between EduForYou and DevForYou proposes the following requirements for project documentation web pages and the underlying server. The should be one web page for each programmer and one main web page for the whole project. The main web page should contain 3 sections, one section with links to the page of each individual programmer, one section describing the project’s cost so far, and one section allowing the download of the latest build of the developed software.

Individual programmer’s page Each individual programmer’s page should contain a short description of the programmer, the accumulated spend time for that programmer, and a field to request contact information for that programmer. No contact information should be provided directly on the web page to prevent spammers to exploit that information. The contact information request should work as follows. The web page contains a field to enter an email address. After entering an email address and pressing a ”commit” button the contact information is mailed to the entered email address if and only if the computer which was used to enter the email address is within Karlstad university’s network. The IP address, the

hostname, and the number of requests for each IP address is recorded in a database on the web server side.

The chef of DevForYou sees the development of these kind of services as potential future enterprise and wants all programmers to have the knowledge and ability to develop these kind of service and requests therefore that each programmer develops and maintains such a page by herself.

Moreover, since the team is new to the XP paradigm, the chef wants to be able to follow the quality of the XP development. To do so, each individual programmer's page should show for each iteration the probability that the programmer fulfills the time estimates for her tasks. An expert, Catrin Bergkvist, is available for questions concerning the statistics involved.

Project's cost The main page should present the accumulated cost of the project for the completed iterations.

The chef of DevForYou delegated the tasks to identify and present relevant factors for the calculation of project's cost to the team itself. Two experts, Anette Hedbern and Margareta Bjurklo, are available for questions concerning the economics involved.

Download Links to all releases (1 jar file for the client and 1 jar file for the server) should be provided on the main web page. The download should be secured by a login. The login information has to be communicated to the customer and within DevForYou.

C Appendix Survey 1

The value in each field represents the number of responses in the given category.

This survey is part of our (Christian and Mathias) level D dissertation "Case Study on teaching XP". Since this survey is only for evaluation it has no influences on your grade and is strictly anonymous. After every question there is short space to comment your answers if you want. In the case that the space is not enough you can continue on the last page. You can answer in Swedish, English or German.

Question	**Not Well At All**	**Not Well**	**Neutral**	**Well**	**Very Well**
1. How well did you understand all practices?			3	7	
2. How well do you think the practice "sit together" is applicable during the project?			1	4	5
3. How well did you understand the practice "sit together"?			1	5	4
4. How well do you think the practice "informative workspace" is applicable during the project?		3	4	2	1
5. How well did you understand the practice "informative workspace"?			1	6	3
6. How well do you think the practice "energized work" is applicable during the project?			5	5	
7. How well did you understand the practice "energized work"?			5	4	1
8. How well do you think the practice "pair programming" is applicable during the project?			1	4	5
9. How well did you understand the practice "pair programming"?				5	5
10. How well do you think the practice "stories" is applicable during the project?			4	4	2
11. How well did you understand the practice "stories"?		1	3	2	4
12. How well do you think the practice "weekly cycle" is applicable during the project?		2	2	4	2
13. How well did you understand the practice "weekly cycle"?			2	5	3
14. How well do you think the practice "slack" is applicable during the project?		2	6	2	
15. How well did you understand the practice "slack"?		3	1	6	
16. How well do you think the practice "ten minute build" is applicable during the project?		1	3	4	2
17. Have you decided to embrace XP in the project?	**0**				**10**
18. How well did you understand the practice "ten minute build"?		1	4	3	2

19. How well do you think the practice "continuous integration" is applicable during the project?		1	3	3	3
20. How well did you understand the practice "continuous integration"?		1	3	3	2
21. How well do you think the practice "test-first programming" is applicable during the project?		2	5	3	
22. How well did you understand the practice "test-first programming"?		1	4	2	3
23. How well do you think the practice "incremental design" is applicable during the project?		1	4	4	1
24. How well did you understand the practice "incremental design"?			3	6	1
25. How well do you think the practice "shared code" is applicable during the project?			4	3	3
26. How well did you understand the practice "shared code"?		1	2	4	3
27. How well do you think the practice "code and test" is applicable during the project?			1	6	3
28. How well did you understand the practice "code and test"?			4	4	2
29. How well do you think the practice "single code base" is applicable during the project?			4	4	2
30. How well did you understand the practice "single code base"?	1	1	2	5	1
31. How well do you think the practice "negotiated scope contract" is applicable during the project?		1	6	2	1
32. How well did you understand the practice "negotiated scope contract"?	1	1	6	1	1
33. How do you like the idea of having theoretical sessions on XP followed by performing the theory into practical use?		2	1	5	2

Table 0-1 Responses to Survey 1

D Appendix The Observation Protocol

Week 1

In the first week the students familiarized themselves with the room, the equipment and the software. On the first day of the week, the students had a planning day were they made story cards which they put on the story wall; hence an informative workspace began to take shape. The students also talked about writing some extra story cards that they might drop later on, consequently integrating slack into their planning. The integration of slack was made without any of the students actually being aware of that they performed the practice “Slack”, mainly because none of the students even mentioned the practice “Slack”. This became even more obvious later the same day when they had a design meeting with the boss and the customer. The boss asked the students if they thought about implementing slack into their system and they all agreed that they should, so they put two stories “on hold” to make use of the practice “Slack”. During the project the students had all their documentation on a common homepage together with personal homepages. During the first day some of the students worked on the homepages, some tried to learn how Eclipse worked and some did activities not related to the project at all. The students decided that they should break for lunch in two groups so that some students were always present in the room. This made the practice “Sit Together” hard to follow since the communication path between the two groups was stretched and the whole team did not sit together. From the other point of view the students did separate into two groups so that they are able to take lunch and stay in groups.

The second day started with a design meeting; hence they made use of the practice “Incremental Design”. After the meeting it seemed that the students had a clear view on their roles in the project and on what had to be done. A group of students went for a meeting with an expert regarding their homepages and the database they were going to use in their project. The boss showed up before lunchtime and a small project meeting commenced where they talked about the database design, once again they made use of the “Incremental Design” practice. After the meeting new story and task cards were created. Some of the students began working on the database using the Eclipse development environment together with JUnit and CVS. The students working on the database wrote tests before they wrote the code. The rest

of the students worked on the homepages. None of the students made use of the practice “Pair Programming”, instead two students worked alone, some in a group of four students and the rest in a group of 3.

On the third day the students installed a meeting table in front of the story board, hence enhancing their “Informative Workspace”. After setting up the meeting table the students began the day with a design meeting (incremental design), followed by a coding session. On this occasion the students programmed in pairs with the exception of one group who consisted of 3 students because only 9 students were present. This was a sign that the students had thought about the practice “Pair Programming” but they did not follow the practice fully since they did not switch pairs during their coding session. Since the students who were working on the database created tests before they wrote the actual code, they used the practice “Test-First Programming” and since they used CVS they automatically used the practice “Continuous Integration”, “Shared Code” and “Single Code Base”. In Eclipse they used the “automatic build” function and when they timed the build together with the tests, it took about 3 seconds, therefore they did not break the ten minute limit for the practice “Ten-Minute Build”. Some of the students quit earlier that day because the felt that they had a lack of inspiration, thus they made use of the practice “Energized Work”.

The fourth day was a holiday so none of the students was present.

During the fifth day only 6 students were present since the rest of them had taken an extended holiday. By doing so the students broke the practice “Sit Together”. Two of the students worked together on the homepages and four on the code for the project. The two students who worked on the homepages were spending most of their time doing non-project-related work. On every Friday the students should have made a release, but since the students more or less had concentrated on getting acquainted on the project during the week, no release was made.

#	Practice Name	Used	Comments
1	SitTogether	Partial	Lunch at different hours, not all students present on the last day
2	Informative Workspace	Yes	
3	EnergizedWork	Yes	See Table 0-3
4	PairProgramming	No	This practice was only used on 1 day of the week and they did not switch pairs
5	Stories	Yes	

6	WeeklyCycle	n/a	Could not be measured during the first week
7	Slack	Yes	
8	Ten-MinuteBuild	Yes	
9	ContinuousIntegration	Yes	
10	Test-FirstProgramming	Yes	
11	IncrementalDesign	Yes	
12	SharedCode	Yes	
13	Code&Test	No	Documentation on statistics that did not concern code and tests were made on their personal homepages
14	SingleCodeBase	Yes	
15	NegotiatedScopeContract	No	The boss and customer did not discuss this practice with the students

Table 0-2 Recommended practices, used or not: Week 1

Student	1	2	3	4	5	6	7	8	9	10	Group
Average	7.7	7.4	4.7	4.7	7.0	0.8	7.1	7.0	7.2	5.8	6.0

Table 0-3 Average hrs/day: Week 1

As can be see in Table 0-3, the average working hours for each student were below the recommended average of 8 hours per day which is the standard Swedish work model.

Week 2

On the morning of the first day, the students had a design meeting where they talked about what had to be done and what they did the previous week. After the meeting a coding session commenced with some of the students finishing their personal homepages and the rest were coding on the project. The students did not program in pairs with one exception. Once again the students divided up into to groups when it was time to break for lunch, although they used the same room when they were working, the observers believe that the practice “Sit Together” was used to its full extent because it can be neglected that not all of the students were present at the same time, such as when they broke for lunch two groups took turn working in a two hour period. It is hard to decide whether the students followed the practice “Sit Together” or not. In the afternoon the students had a meeting with the boss and the customer. They talked with the customer about what they had finished and what to do for the second week. The students seemed to be on schedule even though they had had performance problems with the network connection at the university. They also talked with the boss about adding tasks that could be dropped or implemented if the students would have free time on their hands. The boss, customer and the students all orally agreed on the time, cost and quality to be spend on each of the tasks. This meant that they made use of the practice “Negotiated Scope Contract”.

After the meeting the students continued with a coding session, once again some of the students worked alone and some in pairs. During the coding sessions all students used Eclipse and its support for CVS. This meant that they made use of the practices “Continuous Integration” and “Shared Code”. These practices were used automatically throughout all the weeks the project took place. Since the students made use of the “build automatically” feature of Eclipse, the students continuously checked if their build took “a long time” or not. At this point the build took under 5 seconds and was hardly noticeable. The students who had little or no knowledge of Java-programming did not write test-cases before they wrote the actual code. The observers think that this was understandable since they wanted to concentrate on how Java worked before they could get into the more technical aspects of Java such as writing test-cases. The test-cases should also be made in Java. All students left a bit earlier due to fatigue. Since they stopped working when they ran out of energy, it meant that they made use of the practice “Energized Work”.

#	Practice Name	Used	Comments
1	SitTogether	Yes	Lunch at different hours, one student worked at home
2	InformativeWorkspace	Yes	
3	EnergizedWork	Yes	See Table 0-5
4	PairProgramming	No	Not switching pairs, some worked alone, some in groups
5	Stories	Yes	
6	WeeklyCycle	No	The activities on the second week did not correspond to the activities on the first week
7	Slack	Yes	
8	Ten-MinuteBuild	Yes	
9	ContinuousIntegration	Yes	
10	Test-FirstProgramming	Partial	Students working on the Graphical User Interface (GUI), found it impossible to write test before writing code
11	IncrementalDesign	Yes	
12	SharedCode	Yes	
13	Code&Test	No	Documentation on statistics
14	SingleCodeBase	Yes	
15	NegotiatedScopeContract	Yes	

Table 0-4 Recommended practices, used or not: Week 2

Day two started with a coding session followed by a design meeting. Here the observers think that it is clear that the students did not follow a certain pattern. We had thought that they would begin every day with a meeting, but instead they had meetings when the students

themselves thought it was necessary. After the meeting the coding session continued where they worked in a “3-2-2-2-1” programming fashion, this means that one student worked alone, 6 students worked in pairs and 3 students formed one group where they worked together. The coding session continued until lunch. After lunch the students worked together in a “3-3-2-2” programming fashion. All throughout their coding sessions this day, only two pairs used “Test-First Programming”. The rest of the students wrote code before they wrote the tests. Five of the students went home about an hour earlier, 3 students left about 30 minutes earlier and one student left at the end of the day.

On the third day the students once again began with a coding session followed by a design meeting. On this design meeting only 5 students attended while the rest continued coding in a “2-2-1” fashion. All throughout the day, the coding was made in a “3-2-2-2-1” fashion with no switching between the “pairs”. Again most of the students left earlier.

Student	1	2	3	4	5	6	7	8	9	10	Group
Average	6.0	6.2	6.1	6.1	6.0	5.2	6.8	6.1	6.5	6.3	6.1

Table 0-5 Average hrs/day: Week 2

Day four started with only two students programming, the rest of the students were conducting activities not connected to the project. Not until two hours into the day was every student working on the project. When all the students were programming, they worked in a “3-2-2-2-1” fashion. Five students left 3 hours early, while the remaining students had a small design meeting followed by a coding session. The remaining students worked in a “3-2” fashion. Half an hour before the day ended the remaining students left.

On the last day of the week, one student had taken the day off, one student was working from home (which clearly was in conflict with the practice “Sit Together”) and the rest worked in a “4-4” fashion. After the students had made their first release available for the customer to download from their project homepage, 3 of the students left four hours early. The rest of the students left for about three hours early. Since the students who worked on the Graphical User Interface (GUI) were using a graphical tool to build the GUI, they found it impossible to write tests before they wrote the code since the code was automatically generated. One way to conduct testing on the GUI would be to create classes that simulate buttons being pressed, menu options being chosen etc, but the students thought that the time for creating such classes would be greater than testing out the functionality of the GUI manually.

Week 3

On the first day only four people showed up due to confusion about whether it was a Swedish holiday or not. The students that were present decided to tidy the code and organize the story wall. No meeting with the customer and the boss was held during the day and the students all left at lunch time. The second, third and fourth day, the students programmed in a "3-2-2-2-1" fashion without switching "pairs". The students had design meetings when they felt it was necessary, thus making use of the practice "Incremental Design", and "as usual" they left early all of the days. The meeting with the boss and customer was held on the second day. New story and task cards were made and they also negotiated scope contracts. On the last day of the week, three of the students began the day with doing non project related work.

#	Practice Name	Used	Comments
1	SitTogether	Partial	Not all present on the first day, lunch at different hours
2	InformativeWorkspace	Yes	On the second day the workspace was a bit unorganized, but on th next day the workspace was organized again
3	EnergizedWork	Yes	See Table 0-7
4	PairProgramming	No	No pair switching, some worked alone and some in groups
5	Stories	Yes	
6	WeeklyCycle	No	Week three did not correspond to the previous week(s)
7	Slack	Yes	
8	Ten-MinuteBuild	Yes	
9	ContinuousIntegration	Yes	
10	Test-FirstProgramming	Partial	Students working on the Graphical User Interface (GUI), found it impossible to write tests before the code
11	IncrementalDesign	Yes	
12	SharedCode	Yes	
13	Code&Test	No	Documentastion on statistics
14	SingleCodeBase	Yes	
15	NegotiatedScopeContract	No	No meeting with the boss or the customer this week

Table 0-6 Recommended practices, used or not: Week 3

The rest of the students were working in a "2-2-2-1" fashion to make a release. After the release was made, two students were optimizing the code and test so that the build and tests would go as fast as possible. The total time for the tests and the build to run was about 13 seconds, well within the "Ten-Minute Build" practice. After lunch no work was done on the

project and two students even went home for the week. The rest of the students went a quarter earlier than the recommended 8 hour working day.

Student	1	2	3	4	5	6	7	8	9	10	Group
Average	5.7	5.9	4.9	4.9	5.1	6.0	6.2	5.0	6.0	2.8	5.3

Table 0-7 Average hrs/day: Week 3

Week 4

On the first day the students had a meeting with the boss and the customer where all parties agreed on new story and task cards. The students raised a voice of concern regarding the 200 working hours that they must have to complete the course. Due to the "Energized Work" practice, the students found it nearly impossible to work for 200 hours and still follow the practice "Energized Work".

#	Practice Name	Used	Comments
1	SitTogether	Partial	The students were usually working together in the room, but broke for lunch in different hours
2	InformativeWorkspace	Yes	
3	EnergizedWork	Yes	See Table 0-9
4	PairProgramming	No	Not switching pairs, some orked alone and some in groups
5	Stories	Yes	
6	WeeklyCycle	No	Week four did not correspond to the previous week(s)
7	Slack	Yes	
8	Ten-MinuteBuild	Yes	
9	ContinuousIntegration	Yes	
10	Test-FirstProgramming	Partial	Students working on the Graphical User Interface (GUI), found it impossible to write tests before the code
11	IncrementalDesign	Yes	
12	SharedCode	Yes	
13	Code&Test	No	Documentation on statistice
14	SingleCodeBase	Yes	
15	NegotiatedScopeContract	Yes	Decided on the meeting with the boss and customer

Table 0-8 Recommended practices, used or not: Week 4

The boss answered that the students could add time if they thought about project solutions when they were at home. This is in direct conflict with the practice "Sit Together" (i.e in the

same workspace). After the meeting the students had a coding session where they programmed in a “3-3-2-2” fashion. Once again the students took lunch in two groups at different hours.

Student	1	2	3	4	5	6	7	8	9	10	Group
Average	7.4	7.5	6.7	2.8	5.2	6.9	6.9	6.8	6.7	7.1	6.4

Table 0-9 Average hrs/day: Week 4

When all the students had come back from lunch, they programmed in a “3-3-1-1-1-1” fashion. On this day the students left after working for a full day (8 hours). The second, third and fourth day were more or less the same. One student was absent all three days. The students had coding and testing sessions with automatic builds. No real “Pair Programming” was done in the sense that they were not switching pairs and that some worked alone and some in groups of three or more. They took lunch at different hours and all of the days the students left a bit earlier. On the fifth day a release was made. After the release had been made available on the project homepage, the students spent their time on doing non project related work until they left for home an hour earlier than the recommended 8 hour work day.

Week 5

The first day of the last week began with a coding session. The coding session continued until lunch where the students again divided themselves into two groups who took lunch one hour apart. The coding session continued after lunch until the boss and customer showed up for a meeting. During the meeting a small demo of the product that the students had been working on, was shown to the customer. The customer suggested some minor changes and new story and task cards were written to meet the customer’s demands. During the meeting with the boss, they all talked about slack and performing tasks that had been “put on hold”. They also negotiated scope contracts orally for the new story and task cards. After the meeting the students continued coding and applying the changes that the customer had suggested. During the second, the third and the fourth day, the students had coding sessions followed by short design meetings. On the last day the students had a presentation and a demo of their product for the customer, boss and some of the experts.

#	Practice Name	Used	Comments
1	SitTogether	Partial	The students were usually working together in the room, but broke for lunch in different hours
2	InformativeWorkspace	Yes	

3	EnergizedWork	Yes	See Table 0-11
4	PairProgramming	No	Not switching pairs, some worked alone and some in groups
5	Stories	Yes	
6	WeeklyCycle	Yes	Compareable to week four
7	Slack	Yes	
8	Ten-MinuteBuild	Yes	
9	ContinuousIntegration	Yes	
10	Test-FirstProgramming	Partial	Students working on the Graphical User Interface (GUI), found it impossible to write tests before the code
11	IncrementalDesign	Yes	
12	SharedCode	Yes	
13	Code&Test	No	Documentation on statistics
14	SingleCodeBase	Yes	
15	NegotiatedScopeContract	Yes	

Table 0-10 Recommended practices, used or not: Week 5

All parties were pleased about the product and only a couple of minor changes were made before the students could put their final release up on their webpage for downloading purposes. After the presentation and the demo, the students answered questions about the project. All students thought it had gone well with no major problems. The students also said that they did not experience any major difficulties working with XP. One question about XP that we found particularly interesting was: "Do you think you would have followed XP differently if you had not made a product that used some of the XP practices?" And the students reply was: "Not likely, since we did not really think that much about following the XP practices during the project. Eclipse with CVS and JUnit helped us perform some of the practices automatically so that we did not even have to think about some of the practices" In Table 0-11 can be seen that the students worked very few hours..

Student	1	2	3	4	5	6	7	8	9	10	Group
Average	5.7	5.6	6.0	6.0	3.8	5.9	6.0	5.7	5.4	6.5	5.7

Table 0-11 Average hrs/day: Week 5

All 5 weeks

#	Practice Name	Used	Comments
1	SitTogether	Partial	The students were usually working together in the room, but broke for lunch in different hours and at one time one student worked at home

2	InformativeWorkspace	Yes	
3	EnergizedWork	Yes	See Table 0-13, Table 0-3, Table 0-5, Table 0-7, Table 0-9 & Table 0-11
4	PairProgramming	No	Not switching pairs, some worked alone and some in groups
5	Stories	Yes	
6	WeeklyCycle	No	Only the last two weeks were comparable
7	Slack	Yes	
8	Ten-MinuteBuild	Yes	
9	ContinuousIntegration	Yes	
10	Test-FirstProgramming	Partial	Students working on the Graphical User Interface (GUI), found it impossible to write tests before the code, others did the tests before the code
11	IncrementalDesign	Yes	Started out with designs meetings every morning, changed to having design meetings when the students felt it necessary
12	SharedCode	Yes	
13	Code&Test	No	Documentation on statistics was kept, hence the students had documentation not regarding code and tests
14	SingleCodeBase	Yes	
15	NegotiatedScopeContract	Yes	This practice was only not used during the first week

Table 0-12 Recommended practices, used or not: All Weeks

Student	1	2	3	4	5	6	7	8	9	10	Group
Average	6.5	6.5	5.7	4.9	5.4	5.0	6.6	6.1	6.4	5.9	5.8

Table 0-13 Average hrs/day: All Weeks

E Appendix Survey 2

In this appendix the real survey sheet can be found. The numeracy that can be found in the fields represents the amount of responses in the equivalent category.

This survey is part of our (Christian and Mathias) level D dissertation "Case Study on teaching XP". Since this survey is only for evaluation it has no influence on your grade and is strictly anonymous. After every question there is short space to comment your answers if you want. In the case that the space is not enough you can continue on the last page. You can answer in Swedish, English or German.

Question	**Not Well At All**	**Not Well**	**Neutral**	**Well**	**Very Well**
1. How did the project work out in your opinion?			1	3	6
2. How well do you think your team used the 15 practices?	1		3	6	
3. Do you think XP is a good process to use in a student project?			3	5	2
4. How well could you perform the practice "informative workspace"?			2	2	6
5. How well could you perform the practice "energized work"?	1	3	2	2	2
6. How well could you perform the practice "pair programming"?		1	2	5	2
7. How well do you think XP worked in your project?	1		2	6	1
8. How well could you perform the practice "weekly cycle"?			8	2	
9. How well could you perform the practice "slack"?		2	5		3
10. How well could you perform the practice "ten minute build"?	2	2	1		5
11. How well could you perform the practice "continuous integration"?			3	5	2
12. How well could you perform the practice "test first programming"?	1	1	5	3	
13. How well could you perform the practice "incremental design"?			4	4	2
14. How well could you perform the practice "shared code"?				5	5
15. How well could you perform the practice "code and test"?			4	4	2
16. How well could you perform the practice "single code base"?			1	4	5
17. How well could you perform the practice "negotiated scope contract"?			7	3	
18. How well did the project work at all?			2	5	3
19. Do you think the project would have worked out better without XP?	0		1		9

20. Do you think you would have worked in a different way if you had not been observed by us?	0		0		10
21. If you would do another project, do you think you would apply some of the XP practices?	9		1		0
22. If yes would you please list them on the other side of this sheet.					
23. How well could you perform the practice "sit together"?		1	1	1	6
24. How well do you think the work load was balanced between the members of your team?	3	4	2	1	
25. Are you happy with the way the project worked	9				1
26. How do you like the idea of having theoretical sessions on XP followed by performing the theory into practical use?		2	3	3	2
27. How well could you perform the practice "stories"?			1	6	3
28. How well do you think XP works in a student environment?		1	5	3	1

Table 0-14 Responses on Survey 2

Answers given on the request with the number 22 in Survey2:

- 10mbuild, pair programming, eg work, info workspace, single code base, shared code
- Pair programming, Energized Work, Sit Together
- It's no use to list them, but the style in which XP relies on with communication and ownership by all is the most valuable asset XP has.
- All of the practices, I like XP
- I believe that most parts will be of use. Many are natural while some such as test first, slack, 10-minute build are a little more difficult to absorb. However I would use most of the parts.
- Pair programming = good, testing = good
- As many as possible
- No answers
- Student did not respond that he will apply any practice
- Sit together, pair programming

F Appendix The CVS Log Files

Week 1

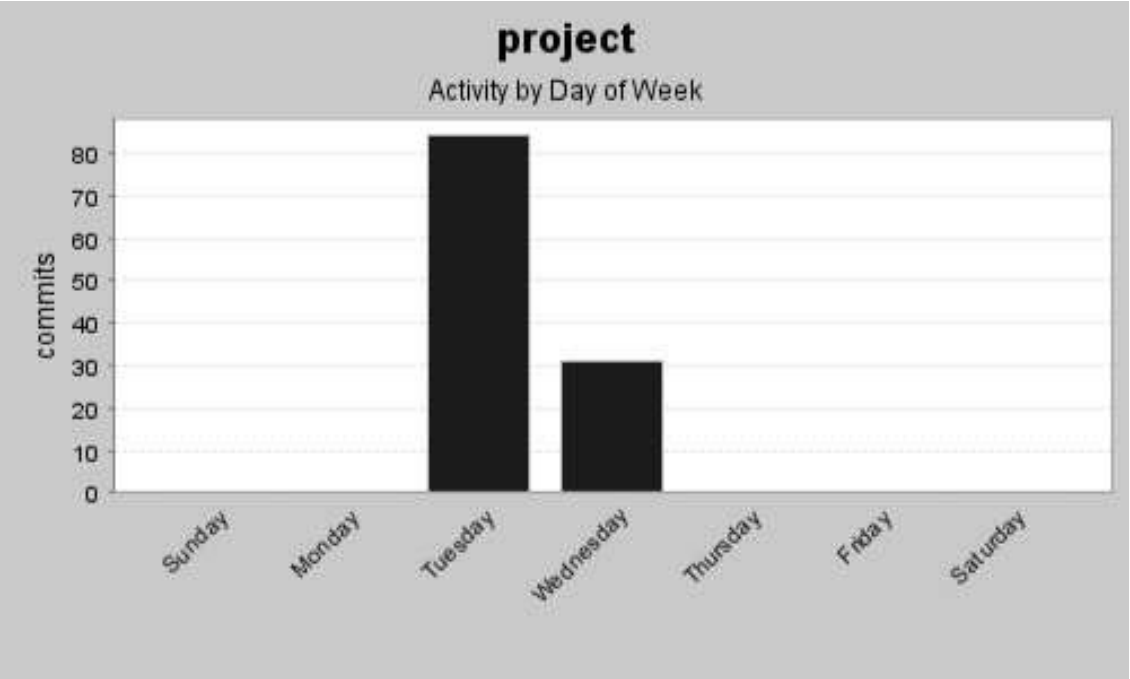

Figure 0.1 Day Commits: Week 1

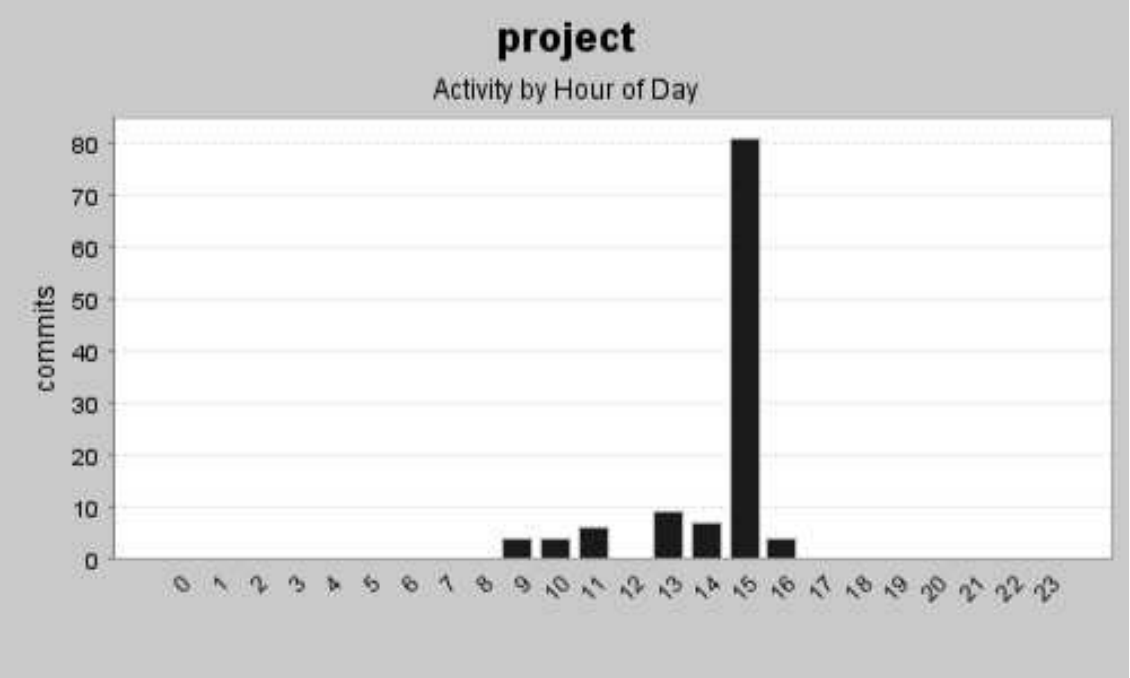

Figure 0.2 Time Commits: Week 1 average

The number of commits made to the CVS repository indicates the activity made by the students on writing tests and code. The figures showing the commits do not show other activities concerning the project such as design meetings, since no documentation regarding design was made and kept in the CVS repository. The figures are an indication of the project activity performed by the student group. In the first week, commits to the CSV repository were only made on Tuesday and Wednesday (see Figure 0.1 and the biggest average number of commits per hour on these two days were at 15:00 hours (Figure 0.2).

Week 2

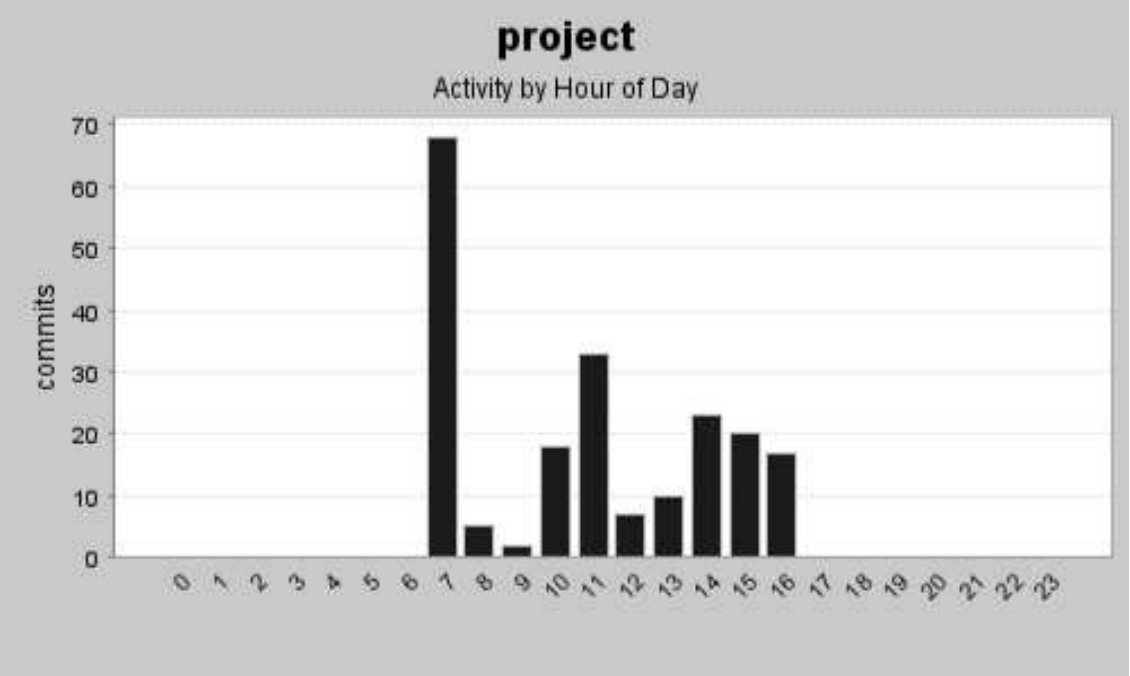

Figure 0.3 Day Commits: Week 2

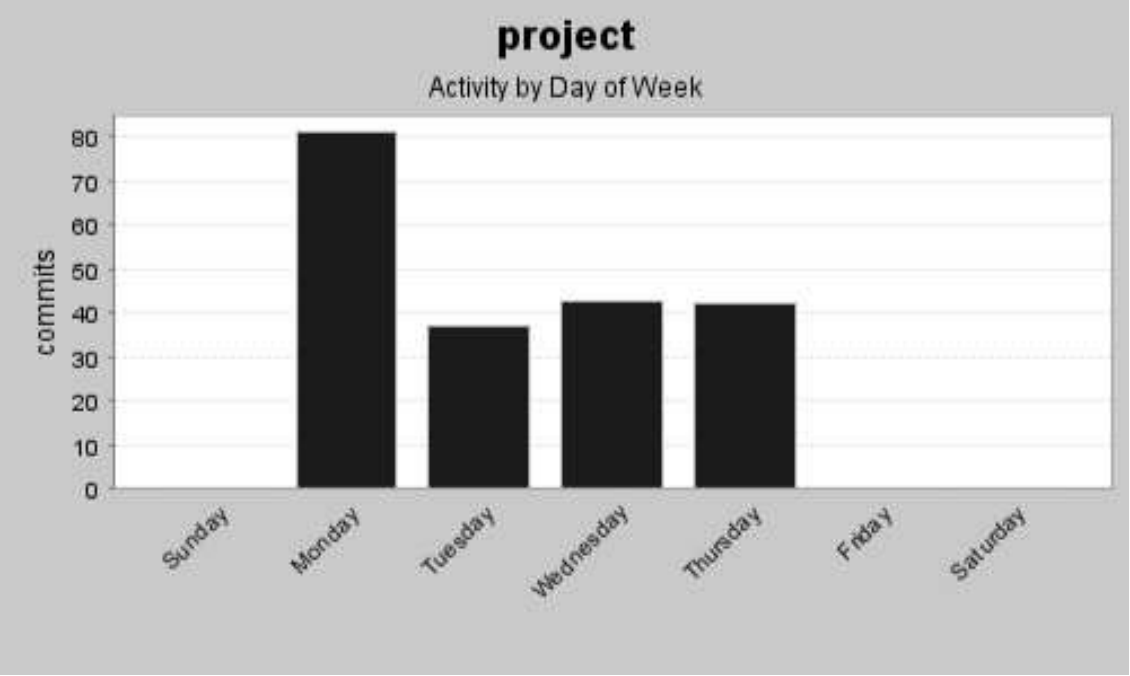

Figure 0.4 Time Commits: Week 2 average

On the second week, commits to the CVS repository were made from Monday until Thursday (Figure 0.3). As we can see in Figure 0.4, the average number of commits varies

from the different hours of the week. The low number of commits between 8:00 and 9:00 hours is due to design meetings and the low number at 12:00 and 13:00 hours is due to lunch breaks.

Week 3

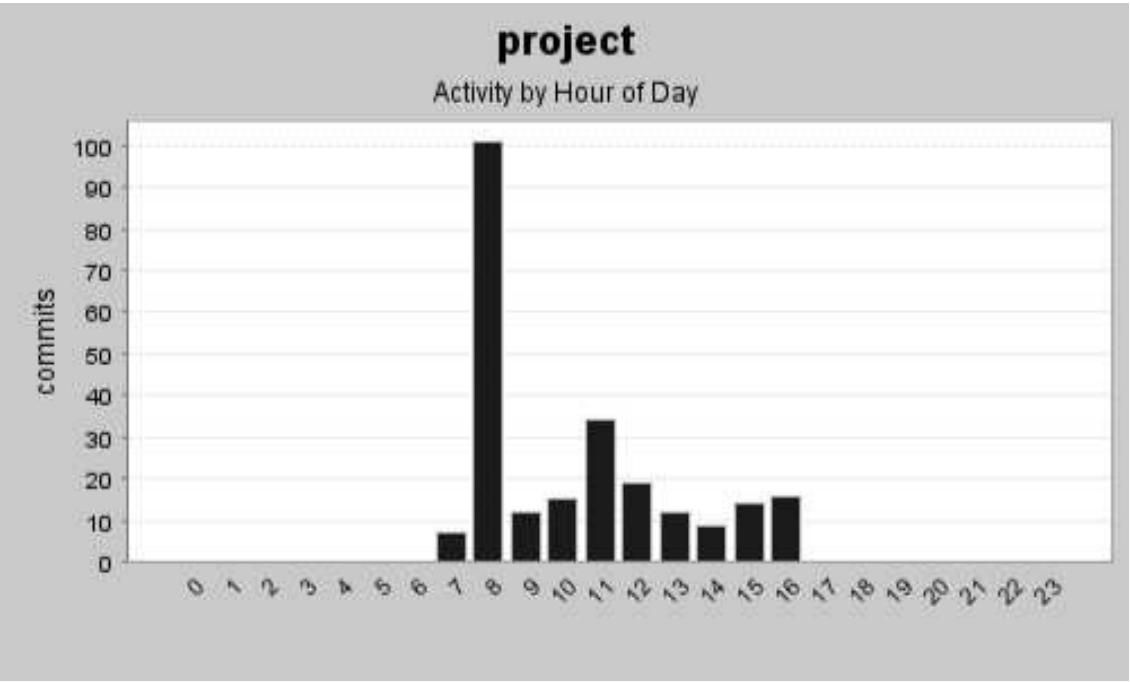

Figure 0.5 Day Commits: Week 3

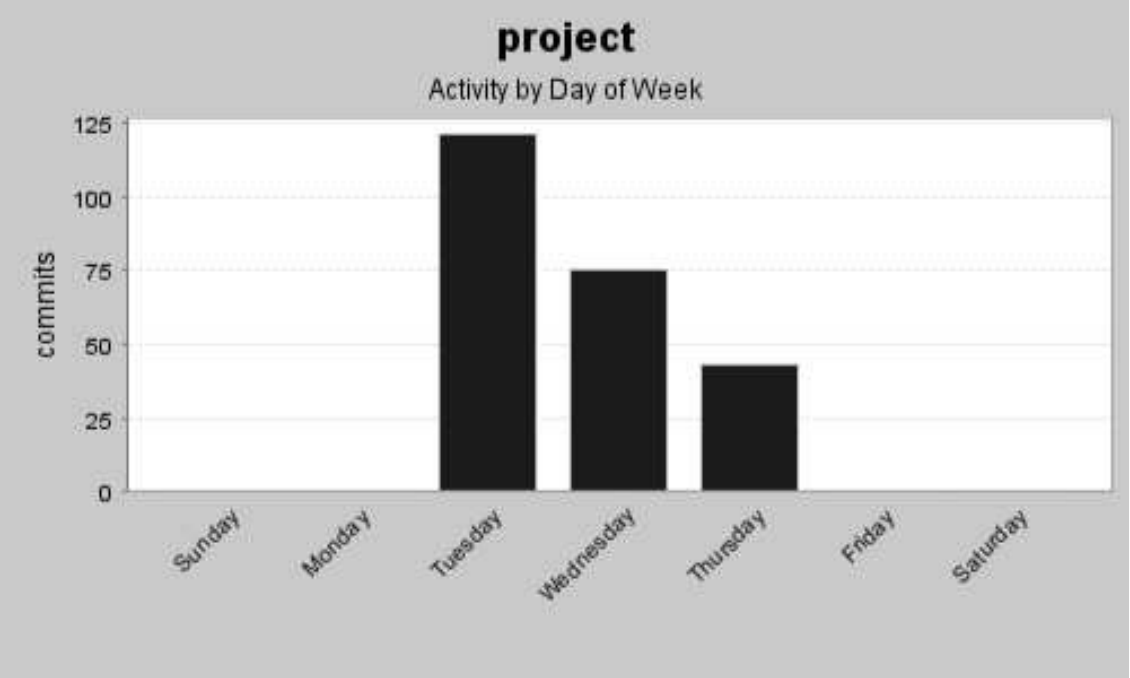

Figure 0.6: Time Commits: Week 3 average

During week three, commits were made from Tuesday until Thursday. In Figure 0.5 can be seen that the number of commits descend and in Figure 0.6 that the highest average number of commits per hour, was between 8:00 and 9:00 hours.

Week 4

.

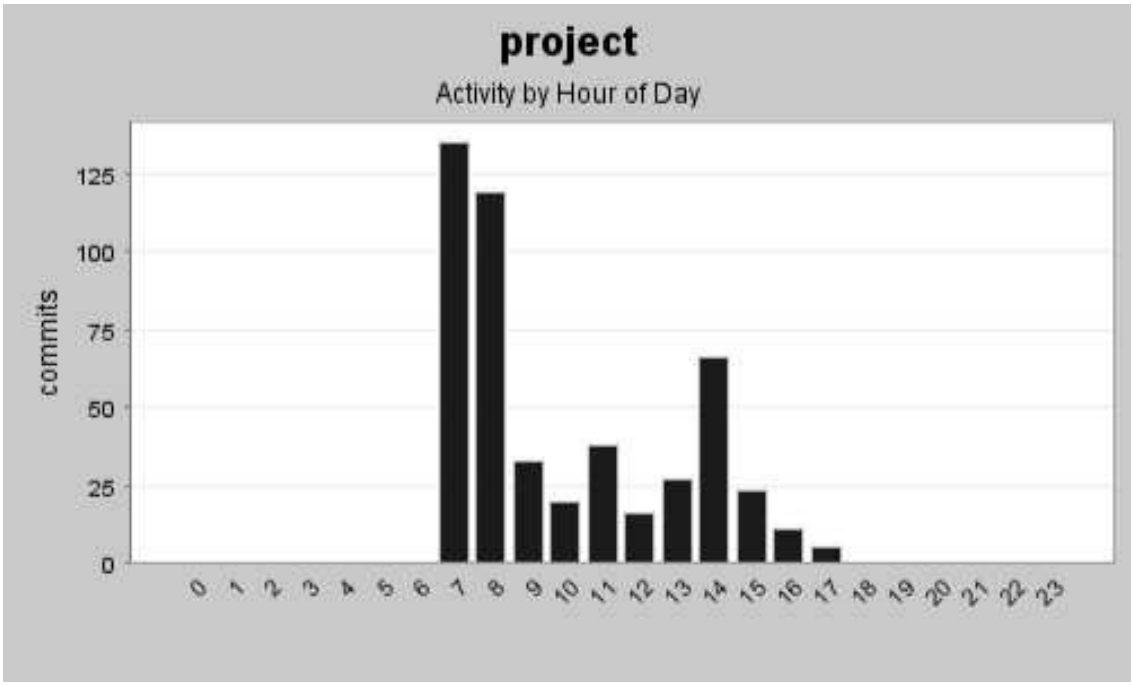

Figure 0.7 Day Commits: Week 4

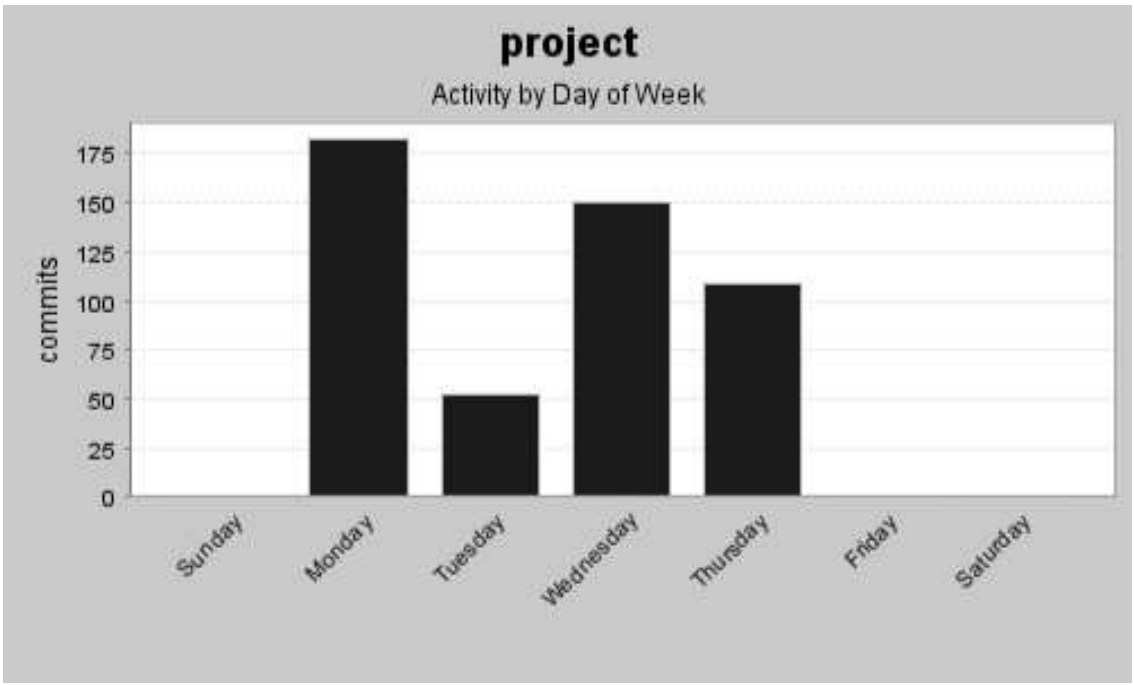

Figure 0.8 Time Commits: Week 4 average

In Figure 0.7 we see that the number of commits is irregularly distributed over the week and the average number of commits distributed over the hours is also irregularly distributed (Figure 0.8). One possible explanation for the irregular distribution over the hours may be due to the fact that the students had design meetings when they felt it to be necessary instead of having an hour at the beginning of each day.

Week 5

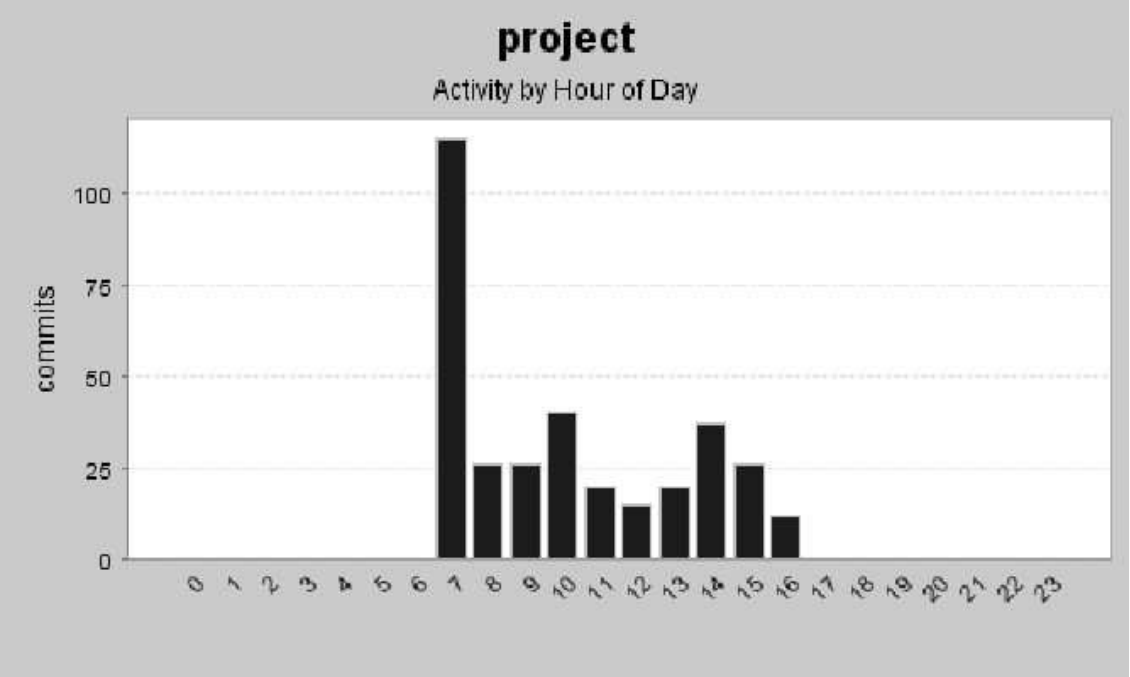

Figure 0.9 Day Commits: Week 5

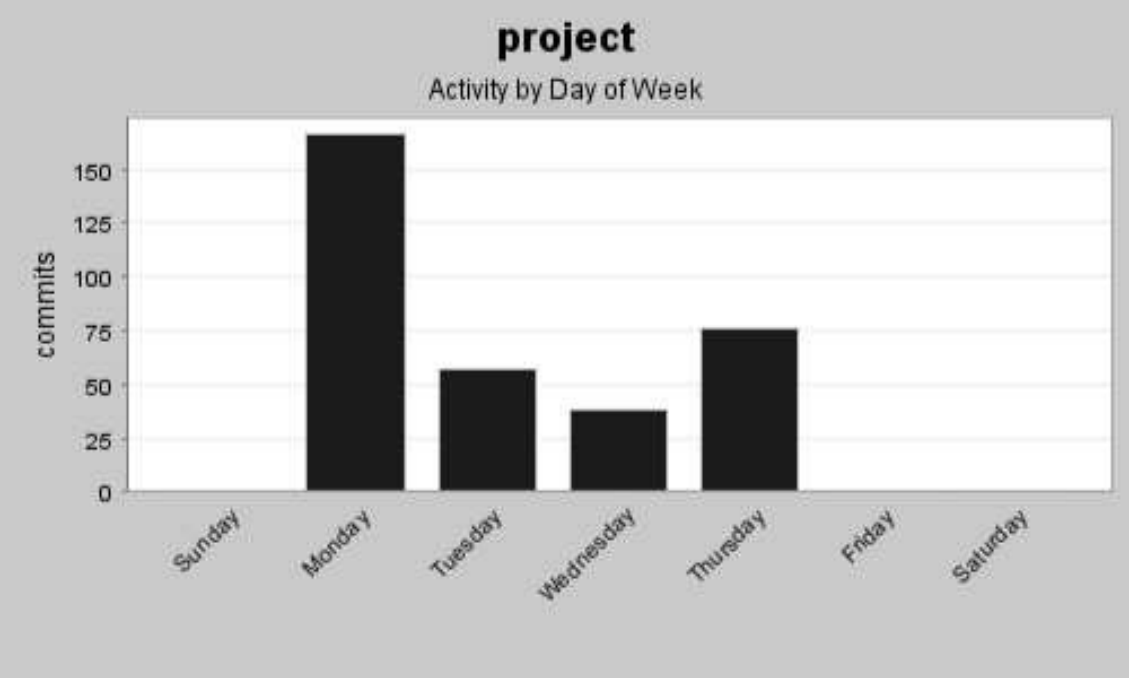

Figure 0.10: Time Commits: Week 5 average

In Figure 0.9 can be seen that the most commits were made on the first day of the week to later drop for the other days. In Figure 0.10 can be seen that most commits were made in the morning and then to even out for the rest of the hours.

All 5 weeks

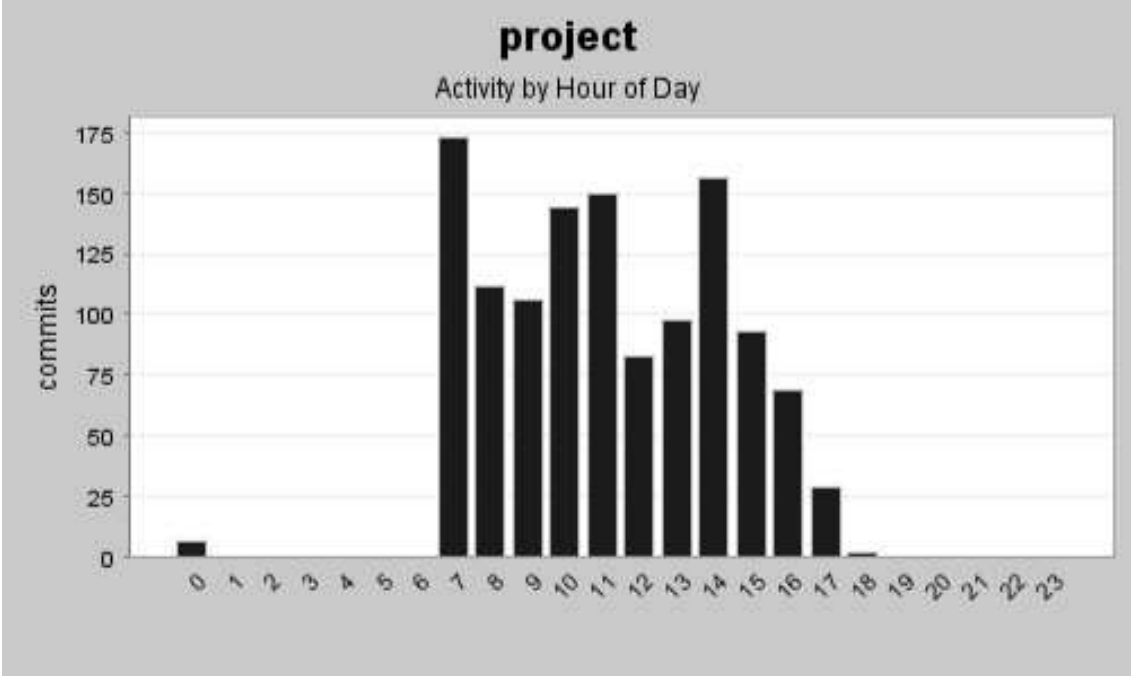

Figure 0.11 Day Commits: Average for all weeks

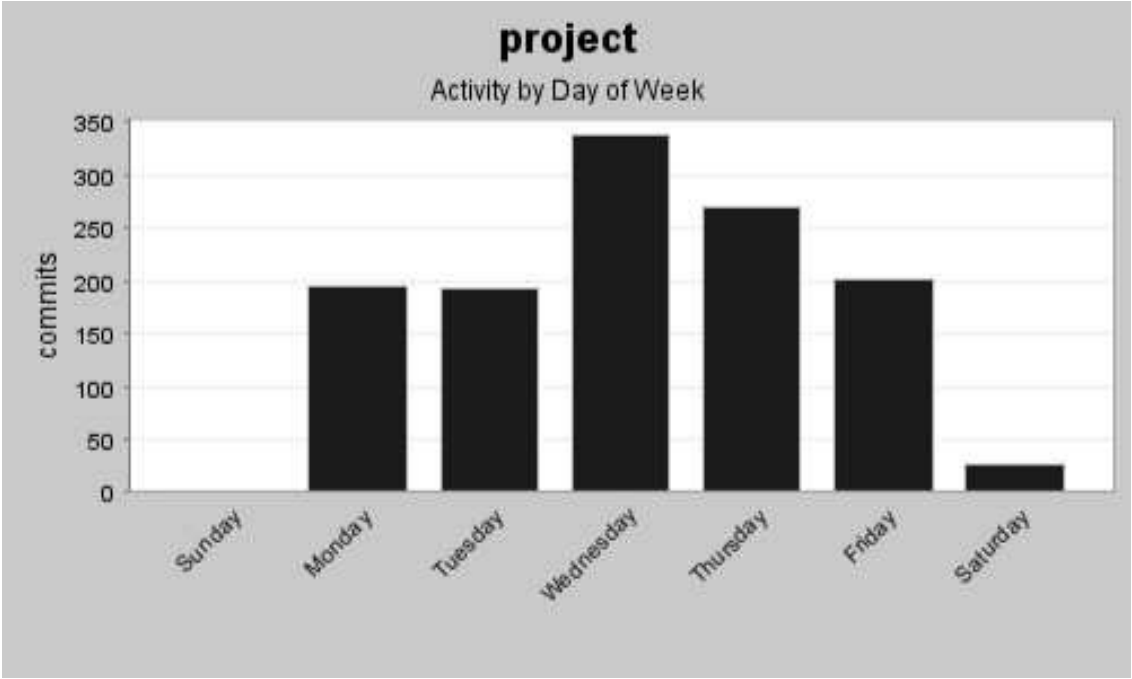

Figure 0.12 Time Commits: Average for all weeks

In Figure 0.11 we see that someone had worked on a Saturday, this was the boss testing out the CVS repository so it has nothing to do with the students. The same goes for Figure 0.12, where the boss made commits at midnight and after 18:00 hours to test the CVS repository.

G Appendix Summary of the meeting 050309 16:00-17:45

Present:	Christian Becker, Mathias Wagnsson	("Consultants")
	Tim Heyer, Mari Göransson	("Customers")
	Donald F. Ross	(Supervisor)

Project: Measurement of certain (15) XP practices in student lab exercises for the Software Engineering course in the Department of Computer Science at Karlstad University.

1. The project will be carried out as a case study (specification requirement).
2. The current version of the specification needs to be added to the project web page – see http://www.cs.kau.se/cs/education/courses/davd22/VT05/CBMW/
3. The project consists of 3 phases
    - a pre-survey of the students to ascertain their "beliefs and conception" of the 15 XP practices to be studied
    - measurements (in the lab/using s/w tools) of what the students actually do
    - conclusions from the survey and a comparison of the measurements with the "beliefs" (expected behaviour) from the pre-study
4. The 15 practices (of 24) which form the basis for the measurements are listed on the course home page – see ???.
5. Measurement may be automatic (using s/w tools or manual) – the **consultants** have to decide which are feasible and how the measurements are to be made. Several alternatives should be presented. The **customers** will make the final decision.
6. Lectures start week 17 and the students start work week 18 – the **consultants** should present their proposals sufficiently ahead of time that the measurement processes are agreed upon before then.
7. Points in the case study to note are
    - The pros and cons of case studies – these aspects should be clearly documented in the report.
    - The level of "intrusion" caused by the measurements
    - The risks of affecting the results involved in measuring
    - There is no compulsion on the students to use the recommended XP practices.
8. Mari G. should be present at each meeting between the consultants and the supervisor
9. For the next meeting (050323) the consultants are required to produce
    - A table describing the 15 practices in detail (as well as a textual description of each practice)
    - A new time plan showing expected milestones.

Donald F. Ross