Sara Shahzad

Analyzing the Extreme Programming Practices and Knowledge Management in Software Engineering Education - Shrinking the Gap between Industry and Academia

---

Dissertation
vorgelegt an der
Technischen Universität Graz

zur Erlangung des akademischen Grades
Doktor der Technischen Wissenschaften
(Dr.techn.)

durchgeführt am Institut für Softwaretechnologie
Technische Universität Graz

Betreuer: Univ.-Prof.Dr.techn.Dipl.-Ing. Wolfgang Slany
Begutachter 1: Univ.-Prof.Dr.techn.Dipl.-Ing. Wolfgang Slany
Begutachter 2: FH.-Prof. Univ.-Doz. Dr. Elke Hochmüller

März 2010

"To my mother whose passion for education became my strength and motivation to accomplish this goal of my life"

Acknowledgements

Praise be to Almighty ALLAH, the Beneficent, the Merciful, Who has awarded me with this great achievement of my life.

I dedicate this achievement to my parents who have always inspired me and helped me in my studies. I also dedicate and share my joy with my brothers, sister and in-laws especially my father in-law though he is not among us anymore but due to his support I was able to avail this opportunity to study in Austria.

I find no words of thanks for my family, especially for my husband Syed Tanveer Shahzad and for my little stars Salleh and Daniyal. Their share in my studies is beyond words which cannot be explained, only my achievement is more than everything for them. I owe my deepest gratitude to my supervisor Professor Dr. Wolfgnag Slany for his constant encouragement, help, and invaluable supervision of my research. Lots of sincere wishes and thanks for my external supervisor Professor Dr. Elke Hochmüller for her guidance, support, and encouragement in the final and critical stages of my research. Many thanks to my colleagues at the Institute for Software Technology whose help and guidance also has a share in the success of my PhD studies. My heartiest tribute to my country Pakistan, to the Higher Education Commission of Pakistan (HEC), and to the Austrian Agency for International Cooperation in Education and Research (ÖAD-GmbH). Special thanks to my employer University of Peshawar (Pakistan), and to the Institute for Software Technology, Technical University of Graz (Austria), for their positive role and for providing all possible support required for my studies.

Sara Shahzad

Abstract

Software engineering education has emerged as a combination and unification of theories from the diverse fields of social sciences and psychology, learning and education, and from the field of management sciences, along with the required computer related technical skills and knowledge which are necessary for the software engineering profession. The merger of all these fields into software engineering education and the concept of university-industry collaboration has changed the face of the conventional teaching process. It has taken the traditional student-teacher classroom interaction to a broader spectrum of practical-oriented training which covers not only the coaching of the classical concepts of software engineering but also provides training into how to cope with the challenges of real-life software development industry. Furthermore, the easy availability of experience based knowledge from professional software engineers and from the people related to software development industry, directly from the person or through computer-based knowledge sharing and collaboration mechanism, further assists in enhancing and broadening the learning experience of the students. On top of this knowledge management also plays an important part into this spectrum.

This thesis provides an insight into the findings of the research conducted about agile software development methodologies, especially Extreme Programming (XP) methodology, through a research and development project conducted in an academic environment and the application of the results and lessons learned from the project in designing a software development methodology course taken as a case study to develop a structure and organization framework for teaching XP.

This research contribution includes a framework based on the structure and organization a of course for teaching XP in a university environment to a large group of students incorporating many unconventional pedagogical aids and concepts to simulate an industry like situation within a university setup. The framework is agile and can be adopted for teaching other classical and modern software design methodologies and also other subjects based on theory and practice. The framework also provides a research strategy, well blended into the routine structure of the course, for gathering empirical data which allows a self test mechanism. This research strategy

can also be adapted according to requirements of the course for which the framework is implemented.

The in-practice experience of teaching and the analysis of the framework in the case study demonstrates that there is a need for more research and improvement in teaching software development methodologies to fulfill not only the technical demands of the software development industry but also to provide the new software engineers a training in the fields related to the profession of software engineering so that they have better chances to survive in the industry.

Keywords: Agile Software Development Processes, Extreme Programming, Software Engineering Education, Knowledge Management,

Zusammenfassung

Die Ausbildung zum Softwareingenieur stellt sich immer mehr aus einer Kombination aus und Verbindung von mehreren Disziplinen heraus: Sozialwissenschaft, Psychologie, Bildungswissenschaften und Management ebenso wie technische Kenntnisse und Wissen aus dem Bereich der Informatik, die alle für die Beherrschung des Berufsbildes des Software Engineers notwendig sind. Die Kombination dieser Themen in Verbindung mit einem verstärkten Austausch zwischen Hochschulen und Industrie hat sich auf die Art und Weise ausgewirkt, wie die Lehre dieser Disziplin stattfindet. Das klassische Lehrer-Schüler-Bild wurde dabei von einer wesentlich breiteren und praxisnäheren Ausbildung abgelöst, die nicht nur die klassischen Themen der Informatik umfasst, sondern vermehrt auch auf die Anforderungen der kommerziellen Software Industrie Rücksicht nimmt. Zudem beeinflusst die einfache Verfügbarkeit von Expertenwissen aus der Industrie — sei es aus erster Hand oder über ein Computer-gestütztes Wissensmanagementsystem in Kombination mit kollaborativen Werkzeugen — ein vertieftes und erweitertes Lernerlebnis der Schüler und Studierenden

Diese Dissertation vermittelt einen Einblick in die Forschungsergebnisse, die im Bereich der agilen Softwareentwicklung — insbesondere Extreme Programming (XP) — während eines Forschungsprojektes im wissenschaftlichen Umfeld ermittelt wurden. Weiters wird die Anwendung der Ergebnisse und die im Rahmen einer Lehrveranstaltung gewonnenen Erfahrungen vorgestellt, die sich mit der Erstellung eines Frameworks zum Lehren von Extreme Programming beschäftigte.

Der wissenschaftliche Beitrag der vorliegenden Dissertation versteht sich dabei als Framework zur strukturellen und organisatorischen Gestaltung einer Lehrveranstaltung für Extreme Programming für eine große Gruppe von Studierenden im universitären Umfeld. Dabei werden unkonventionelle pädagogische Ansätze und Konzepte eingesetzt, um möglichst praxisnahe Bedingungen zu simulieren.

Dieses Framework selbst ist ebenfalls agil und leicht an das Vermitteln anderer moderner Softwareentwicklungsmethoden oder allgemeiner Themenbereiche aus Theorie und Praxis anpassbar. Mittels einer Forschungsstrategie, welche in die Struktur der Lehrveranstaltung eingebettet ist, wurden empirische Daten für die

Analyse der Methodik erhoben. Diese Strategie kann einfach auf die Anforderungen von Lehrveranstaltungen angepasst werden, bei denen das Framework verwendet werden soll. Die praxisnahe Lehre und die Analyse des Frameworks in einer Case Study zeigen, dass dieser Forschungsbereich noch weitere Forschungsmöglichkeiten bietet, um die Lehre nicht nur auf die technischen Aspekte der Anforderungen aus der Industrie anzupassen, sondern den Studierenden auch eine sehr gute ganzheitliche Ausbildung zum Softwareentwickler mit guten Erfolgschancen in der Industrie anbieten zu können.

Stichwörter: Agile Softwareentwicklungsprozesse, Extreme Programming, Didaktik und Lehre des Software Engineerings, Wissensmanagement

Contents

List of Publications

The following publications have been produced in the context of this research.

1. User Interface Design for a Content-aware Mobile Multimedia Application: An Iterative Approach (2007) [46]
2. Inside view of an Extreme Process (2008) [98]
3. User Interface Design for a Mobile Multimedia Application: An Iterative Approach (2008) [48]
4. Optimizing Extreme Programming (2008) [45]
5. Probing an agile Usability Process (2008) [112]
6. Integrating Extreme Programming and User-Centered Design (2008) [50]
7. Agile User-Centered Design Applied to a Mobile Multimedia Streaming Application (2008) [49]
8. Concept and Design of a Contextual Mobile Multimedia Content Usability Study (2009) [51]
9. Learning from Experience: The analysis of an Extreme Programming Process (2009) [97]
10. Integration of Extreme Programming and User-Centered Design: Lessons Learned (2009) [52]
11. Knowledge Management Issues in Teaching Extreme Programming (2009) [96]

Chapter 1

Introduction and Motivation

1.1 Introduction

Software engineering is passing through a revolution. A demand for new tools and techniques for software development comes almost on daily basis. The technical advancement is being made with an overwhelming pace. The software development industry demands for new developers with the skills matching that of experienced software engineers. Educational institutions are expected to play their role in this regard but the situation in this paradigm has not been very ideal. In 1994, Gibbs [29] suggested that the prerequisite for the development and improvement in software engineering was software engineering education [40]. Along with many other reasons it was argued by the academicians that the education system lacked in proper training and education of the developers [29]. After more than a decade the situation is still not under control to a large extent. The flagship conference on Software engineering education and training (CSEE&T) held in 2009 [105] asked for contribution in the field of software engineering education especially to focus on the fact that the education and training process in the field of software engineering needed drastic change for improvement as the fresh graduates were lacking skills considered important for the profession of software engineering.

The seriousness of this matter is felt by researchers from the fields of academia as well as from the software development indus-

try. A special attention is placed on teaching Software Development Methodologies (SDMs) and many researchers have investigated in this field of education to provide the students with the skills required by the industry. Agile SDMs are in center of focus as these are gaining more and more popularity in the industry. The demand for professional expertise in agile SDMs has prompted many universities around the world to offer graduate level courses teaching different agile SDMs. There are many universities where Extreme Programing (XP) methodology ([8],[9]) is being taught as an agile SDM. As XP takes a humanistic approach to software development ([34], [7]) it is desirable that this property of XP practices is highlighted in teaching. Many studies covering different aspects of XP teaching have been presented at different international conferences and research forums. For example, Williams and Upchurch [109] and Assassa et al. [28] examine the use of XP in teaching software engineering skills. Hazzan and Dubinsky [32] have formulated principles for teaching SDMs based on the analysis of XP practices. LeJeune [65] also examines the use of XP as a model for software development from students' perspective. Some researcher and academicians, like Schneider and Johnston [95] have also shown their reservations about the use of XP to teach large scale software development process.

1.2 Research Focus and Motivation

While many of the studies have covered different aspects of XP as a whole and the use of individual XP practices in teaching software development practices (for example, [73]) there is still a lot to be explored in using XP as an overall process for teaching SDM. The motivation behind exploring ways to teach XP as an SDM comes from the continuing demands of software development industry to produce better trained developers in agile processes facilitated by the room for experimentation due to the agile nature of the XP process and developing interest of the research community and people from industry to collaborate in software engineering education to provide the students with a better SDM learning experience in order to expose the students to real-life software industry situation. Experts from academia and industry have done formal as well as informal university-industry collaboration and have evaluated the

effects to see how it meets the critical needs of software engineering education and professional development activities [11]. Carrington [14] has also proposed an industry/university collaboration to upgrade software engineering knowledge and skills in industry. Furthermore, the inherent property of knowledge management activities in XP emphasized by the practice of teamwork offers more in the direction of learning, education and training for real-life software development activities. Bjørnson and Dingsøyr [12] emphasize the greater satisfaction of agile development organizations regarding knowledge management as compared to companies following other non-agile methodologies. The explicit ideas of role playing in XP further enhances the education related aspects of SDMs. Lastly and most importantly the guidelines provided for software engineering education ([41],[5]) provide a good motivation for further research into the direction of agile software engineering education.

1.3 Research Problem

The thesis is focused on defining an educational and teaching solution for young developers to learn XP as an agile software development methodology in a way that they are able to be integrated, readily and easily, into the software development industry using XP and other agile techniques as software development methodology. The research question can be formulated as:

How conventional and non-conventional pedagogical techniques can be combined with the concepts of XP practices to create a better learning, understanding, and knowledge management solution for the students of software development methodologies?

To answer this question it is eminent that first a thorough knowledge and research into the field of agile software development methodologies is required. This research is based on the findings of a software development project which used XP as software development methodology. The practical experience with the XP methodology has provided an insight into the logic and rationale behind XP practices. It has also provided a pattern for learning and applying the methodology in development teams. It has also highlighted the

problems which may be encountered while applying the methodology in software development teams. These conceived concepts of "practical" XP are combined with inherent knowledge management capabilities of XP following the general guidelines for software engineering education ([41],[5]) which has paved the way to define a structure and organization framework to teach XP to a large group of students in a university environment. The study presented here can be treated as a general guideline for structuring and organizing an XP course for students having little or no knowledge about software development methodologies. The framework revolves around the a humanistic approach to teaching using knowledge management concepts; assisted by the concepts from the theory of learning and development such as role playing; exposes to a real-life experience in the form of simulation of information handling and management model of corporate industry; and provides a close to real XP style setup. The framework is agile and can be adopted for teaching other classical and modern software design methodologies and also other subjects based on theory and practice. The framework also provides a research strategy, well blended into the routine structure of the course, for gathering empirical data which allows a self test mechanism. This research strategy can also be adapted according to requirements of the course for which the framework is implemented. It can also be scaled to be implemented as a coaching course for XP and other agile methodologies to train developers new to agile software development methodologies.

1.4 Organization of the Thesis

The study presented here is adapted and formalized according to the concepts of conducting and presenting research in software engineering provided by Runeson and Höst in [90]. The author has adapted the study to the guidelines provided by Runeson and Höst for conducting a case study research and software engineering research. In this regard the case study is selected keeping in view the goals set for the research, the details of the study conducted are presented, data is collected to analyze the working of the concepts implemented during the study. The data is analyzed and reviewed with respect to the goals of the research.

Following is the general organization of the thesis and a short

description of each chapter.

Chapter 2 provides research foundations and literature review taken to formalize the research and to identify the problems, incompletenesses, and shortcomings in the field of software engineering education.

Chapter 3 gives an account of the research background based on a project in which the author participated for a test ground for investigating the field of agile software development methodologies. This chapter summarizes the work done by the author for the project from May 2007 to June 2009 which has provided the opportunity to investigate the practical aspects of agile software development methodologies and set the basis for the proposed study.

Chapter 4 reports on what is concluded from this project and is then used as the motivation of the actual study conducted. It gives details of the lessons learned from working with XP methodology.

Chapter 5 gives XP teaching course case study design, planning and protocol and the outlines the actual study conducted in order to collect the data for the empirical and descriptive analysis of the research.

Chapter 6 presents the details of the analysis and provides the results.

Chapter 7 discusses the results, specifies limitations of the study and gives future directions for following and enhancing the study.

Appendices provide a view of the questionnaires which are used for data collection for the analysis of the study.

Chapter 2

Research Foundations and Literature Review

Software engineering education has become of interest since the advancement in software development techniques. Not only the people from academia but also computer professionals and people from industry and business community have taken part and played their role in enhancing the field of software engineering education. The reason for this multi faceted acceptance of the field of software engineering education is that it is effecting the whole software development industry. Technological advancement has not been utilized to its full extent because of the inability of the software development force to tackle with and work with it optimally. For this reason measures have been demanded on the development of the workforce so that the industry can get trained professionals who can readily be integrated into the running software development activities, and with reduced effort. To attain this there is a need to establish an education system which can provide the students with the required technical knowledge and skills, and an exposure to the kind of environment and situation in which they have to survive ultimately in the industry.

The software engineering education system still lacks in the required infrastructure which fulfills these requirements most optimally. There is an ongoing process which has been started to improve the education system and to introduce the improvements whatever is needed according to the requirement of the industry. The professional skills have been tried to improve the situation by

introducing knowledge about the latest tools. A lot of research has been conducted in this regard. The output of the research is given in the form of guidelines, frameworks, and information to tackle with different problems occurring in the education as well as in software engineering industry.

In the context of this research the author concentrated on the issues concerning with the software engineering education in general and specifically with teaching agile software development methodologies, especially extreme programming. The objective of this chapter is to provide a brief review of the concepts on which this research is based. These include introduction to software engineering education, agile software development methodologies, and knowledge management, and about some other issues are considered to bridge the gap between the education and training provided by academia and the requirements of the software development industry. Although the spectrum set in this review study is wide but the focus is on the specific issue of software engineering education.

The following sections give a review about the topics related to the context of this thesis described in the previous paragraph. The selection is based on the issues presented, the experience of the researcher with software engineering education and the experience of the author with the agile software development methodology. The studies mentioned here present only a subset of the studies that the author reviewed for research and are mentioned here for the purpose of setting a base for the background of the research and for setting a ground for refining the concepts regarding the research problem.

2.1 Issues in Software Engineering Education

The guidelines presented for software engineering education in 1999 by Bagert et al. [5] acknowledged that the academic programs were not devoting enough time for areas of software engineering necessary to be taught for "effective commercial software development". To tackle that issue a set of guidelines was provided to assist the faculty in the design, organization and implementation of software engineering and other related courses. In a previous report named "Software Chronic Crisis" [29] Gibbs gave reasons for the failure of many large scale software projects and suggested that software engineering education needed to be improved. Some other researchers

and practitioners had also emphasized the need to equip software developers not only with the technical knowledge but also to prepare them to practice software engineering. Hilburn [40] also presented a conceptual model for improving software engineering education by tackling issues related with people, processes, and technology. Later on Hilburn et al. ([41],[42]) proposed guidelines for software engineering education. Shaw [99] identified four crucial challenges of providing education based on roles related to software development, providing education for promoting good software development skills, keeping education and training up to current state of the art technology and standard, and the establishment of essentials that accurately represent the ability of software developers. She also proposed a set of aspirations for the educators to meet these challenges in future. Basili et al. [6] talked about experimentation as a tool for the advancement in software engineering education, Žagar et al. [106] discussed knowledge management and social issues related with software engineering education and presented an approach to teaching in [107]. Thompson [102] presented a software engineering education framework in 2008.

2.2 Teaching Agile Software Development Methodologies

Many software development methodologies (SDMs) have been designed to provide an organization and structure to software design process. Agile SDMs have their roots in "Agile Manifesto" [39] presented in 2001 and have become of greatest interest as they provide a flexible way of tackling with the problems in software development while maintaining the basic requirements of completeness and timely delivery of software.

Agile SDMs have been applied by many organizations in software development industry and have shown success [53]. Of all agile methodologies XP ([8],[9],[10]) has been practiced and researched a lot throughout the world and practitioners and educators both have taken interest in using and promoting the XP methodology. XP defines a humanistic approach to software development and follows a set of values, principles and practices. With the increasing use of XP and other agile SDMs in the software development industry

educational institutes have been facing the demand to educate new developers in this field so that they are ready to work with agile methodologies when they enter the software development industry. To provide an understanding of the XP methodology many studies have been presented to review the basic concepts behind the practices of XP as well as to see how these actually work and to be implemented. The research in the field of software engineering education and that of teaching XP have both now taken a common ground. In 2006, Hazzan and Dubinsky presented an educational framework ([33]) which is based on a collection of 14 principles to teach Software development methodologies. Hazzan and Dubinsky have used the concepts of XP ([8],[9],[10]) to exemplify these principles. This framework is a follow-up of their experience with teaching a course of XP at a university [32] and of their framework presented in [21]. Many studies have been presented in this regard. Some of these are mentioned in the table 2.1.

Table 2.1: Agile/XP in Software Engineering Education

Publications	Focus	Year
Kivi et al. [60]	about team designing	2000
Williams [109]	about software engineering skills and XP	2001
Jovanovic et al.[57]	Introducing programming	2002
Hazzan/Dubinsky [32]	Teaching SDMs	2003
Schneider/Johnston [94]	SE Education	2003
Hedin et al. [35]	Introducing SE	2003
Williams et al. [111]	Teaching Agile	2005
Schneider/Johnston [95]	Criticizing XP for SE Teaching	2005
Hedin et al. [35]	XP teaching	2005
Assassa [28]	XP teaching case study	2006
Williams et al. [110]	Pair programming	2008

In general, many approaches have been adopted for strengthening software engineering education. One of these which is widely used and researched is university-industry collaboration. Macias [70], while presenting an empirical assessment of XP discussed that most software engineering education systems stress on providing coaching in the theoretical aspects of software development and assume that a small project will provide enough training to understand the practical concepts of the process. But it is not true. A more true representation of real-world software development is needed by the

students to visualize the software engineering paradigm. One option is university-industry collaboration to provide training of theoretical concepts in universities and using industry platform for practical training. [62] proposed a model to close the gap between industry and software engineering education. Along with informal industry-university collaboration a formal collaboration platform also exits between universities and software development industry. Beckman et al. [11] defined such a case of collaboration.

Team learning is another concept used for enhancing learning. van Solingen et al. suggest that the quality of developers is measured by the extent of their knowledge [104] and include team learning in one of the nine enablers for learning in software development projects - hence depicting the importance of practice in learning.

Knowledge management in software development teams has been another issue of interest. Bjørnson and Dingsøyr ([12]) conclude that agile companies are more satisfied with their knowledge management activities as compared to traditional software development organizations. This eventually leads to the concepts of agile knowledge management (AKM) which removes barriers of long term organizational goals and the corporate hierarchy of roles, which exist in business organizations to "promote collaboration and sharing" of meaningful knowledge [72]. Project management also applies to this situation as business organizations need a prompt reaction to change propagated by timely collaboration and sharing of knowledge among stakeholders, hence integrating AKM and project management concepts. All this knowledge management structure revolves around each and every person associated with the development process including developers, managers, customers, other stakeholders, and also usability experts associated with the team and the project.

Role playing is another concept which is widely used and researched in the field of education and has also been explored for software engineering education. Henry et al. [37] define that as software engineering is not only a technical activity but also involves human effort in which the issues of "communication, collaboration, motivation, work environment, team harmony, sense of purpose, engagement, training, education, etc" take a central place so letting students play different roles which exist in software development industry also pays off the effort. [3] also favours the same concept.

2.3 Summary

All the concepts mentioned in the previous sections and along with many other provide a foundation for further study and research in the field of more fruitful agile SDMs education and training. One such attempt is made by this study which presents a teaching framework as set of guidelines which may be adopted to organize and structure a graduate course for providing training in XP methodology. The setup of this framework is greatly inspired by the educational framework suggested by Hazzan and Dubinsky [33]. The benefits of this framework is the simplicity in organization and structure of the framework itself. The basic concept that is adopted is that the emphasis should be placed on teaching the methodology and not on making the framework more complicated. The framework is designed keeping in mind the current education and knowledge level of the course participants and therefore should be adapted according to the level of education and knowledge of the population sample for which it is used.

Chapter 3

Research Context and Motivation

The following sections provide an overview of the research conducted from May 2007 to August 2009 in the context of m3 project. The project is about developing a mobile multimedia streaming application for large archives of audio-video (AV) content, with a research interest to examine the use of agile software development methodologies, especially extreme programming (XP) and user-centered design (UCD) for mobile applications [51]. The research has been conducted at Graz University of Technology, within the competence network Softnet Austria (www.soft-net.at) and funded by the Austrian Federal Ministry of Economics (bm:wa), the province of Styria, the Steirische Wirtschaftsförderungsgesellschaft mbH.(SFG), and the city of Vienna in terms of the center for innovation and technology (ZIT).

3.1 Introduction to m3 Project

The project has its name "m3" from the words mobile multimedia. The development team consists of five developers, a project manager and a business person who has also performed the duties of the on-site customer and has contributed to the research [45]. The author has participated in the project as a developer in the team. The results of the research have been presented, in the form of research papers, at different international academic forums (in conferences and workshops) and have been reviewed by fellow researchers and

academicians. The research papers are grouped into two categories. One category relates to the team's experience with the adoption of XP, the lessons learned, the retrospective review of the XP process applied. The other category is the application of UCD in the context of m3 project, which emphasizes the usability aspects of the application by giving the details of the usability process employed, the human-computer interaction (HCI) instruments used, and the integration of the UCD into the XP process [49].

The rest of this chapter is arranged as follows. Each section presents the summary of a publication selected from the group of research papers which have been published in the context of m3 project (refer to List of Publications). A total of six publications are selected to be included in this chapter. The publications in category "XP Process" are directly related to the main research idea of the author presented here in this thesis and describe the motivation for this research. Whereas, the publications selected under category "Integrating Extreme Programming and User-Centered Design" are indirectly related to the main topic of the research presented as the m3 project has provided a context for the research conducted. These papers are summarized here to provide an introduction to the project as well as to show how the learning process in the context of XP and UCD has worked. The papers, presenting the ongoing work on the m3 project, are published between the years 2007 to 2009 and outline the details of the progress of the project during a given period of time around which these papers were produced. As the overall concept is of iterative improvement, therefore, in some cases, the implementation of the same idea seems conflicting from one publication to another. What still holds of the process is defined in publication [52] summarized at the end of this chapter.

3.2 XP Process: Learning from Experience

3.2.1 "Inside View of an Extreme Process" (Item 2 in List of Publications)

Summary:

This paper has been presented at the 9th International Conference on "Agile Processes in Software Engineering and Extreme Pro-

gramming". The paper presents the m3 project, team setup, and the XP environment in which the project is being developed. The paper has been published as a short paper and is presented as a poster at the conference. It attempts to provide a precise outline of the goals of the research, the m3 project, and most importantly the details of the process which was being followed at that stage during the project development. In the end this paper provided the motivation to the author to look into the process aspects of XP in detail and research into the learning process being followed by the m3 team.

The paper is divided into sections defining the motivation behind using XP as the development, the detail introduction of the m3 project, and the process of evaluating the XP methodology which is one of the main goals of this research. The paper is short, concise, self explanatory, and provides an overall view of the m3 project.

XP Motivation: The m3 project relates to the development of a multimedia streaming application for mobile devices with an emphasis to utilize huge archives of TV and radio programs and other documentary and entertainment content. XP was selected as design methodology with an intention to use it in a progressive manner: applying each practice, and looking for the improvement and optimization of the whole XP process at the same time. The m3 team was not a usual XP or development team but was rather a team of researchers who aimed at participating in the software development as a test-project for their PhD research. The agile nature of the XP process facilitated the team to schedule and investigate their research ideas during the tight schedule of the project.

The Project The project team consists of six PhD students, five developers (a mix professional programmers and members from academia) and one business person. The members of the team belonged to different social and cultural backgrounds, as two of the team members were from Asia and the rest of the team was European. The customer of the team was a qualified business person, who also performed the business related functions associated with the project. He also performed team mediation during meetings and discussion to keep the team focused on the topic of discussion. His professional experience in team

mediation helped in this regard.

The team's XP-room facilitated team discussions as it is surrounded by six white boards, to place the story cards as well as for drawing graphs and diagrams. The space also provided to having three pairing stations and six private sitting places for the team members.

Evaluating the Process Process evaluation was given utmost importance along with the routine developmental tasks. As the team members were researchers so they had enough interest to perform these tasks. The data required for analyzing the process performance was collected from the implemention of the practices. Different tools for planning (e.g., "xplanner" [23]) and for empirical data collection (e.g., "Shodan 2.0 Input Metric Survey" [63]) were used in the project. In order to develop the base for research and development side-by-side the team made use of routine weekly retrospective meetings to discuss and reflect on the process.

3.2.2 "Optimizing Extreme Programming" (Item 4 in the List of Publications)

Summary:

This paper presents the initial experience of the m3 team with XP methodology during the first year of the project. The main objective of the paper is to present the team's approach towards XP, and what worked and not worked for the team and the project. Initially, XP was applied as a whole, using all applicable practices defined by Kent Beck ([8] [9]). Due to the unique setting of the project and the team setting it soon became obvious that some changes were required in the XP process. To find an optimized process for m3 project, the applied process was evaluated continuously to see how it was working for the project. It was noticed that some practices could be adopted directly but for some practices structural modifications were needed to make them work for the team and the project [101]. It was concluded that even though XP was a simple and light weight process, it had to be tailored to the nature of each team and project.

The paper starts by introducing the on-going research in the field of agile software development methodologies especially XP. The pa-

per also briefly provides the results of the evaluation of the XP process for which the data was collected through code analysis, process evaluation tools, and notes of process review meetings.

The important points of the main sections of the paper are summarized below.

Project Environment This section outlines the research and development setup of the project which is about developing a multimedia streaming application for mobile phones. The objectives of the project are to develop a software product according to needs of the users and the optimization of the XP process. The application is being developed using XP in a progressive manner, that is, applying each practice that can be applied and analyzing the process for possible improvement, in the context of the project. So, the developers, also being researchers, are developing the application and at the same time consciously looking for improvement in the process.

The team consists of five developers and a product manager. The product manager plays the role of an "on-site customer" who also communicates with the other project partners. The developers also communicate with the usability engineer who is working in a close collaboration with the team.

Process This section is divided into two subsections namely "fully implemented practices" and "partially implemented practices". As the name implies the section fully implemented practices provides details about the XP practices which the team applied following the guidelines provided by Kent Beck ([8] [9]) from the very beginning of the project. These were the practices with which the team was satisfied and which were working for the project. These practices included small releases, planning game, pair programming, co-location, collective code ownership, and 40-hours week. Whereas, due to the particular nature of the project setting and the fact that it was the first experience of all team members to work with XP methodology, it was possible to implement some of the practices only partially. The team knew about the application of those practices but in its opinion these were not working all the time. The practices of on-site customer, metaphor, simple design, refactoring, and test first design are mentioned in this section.

Reflection The team was holding a retrospective review meeting on the process after every iteration and release, to reflect over the XP process and application development. The empirical data was collected from various sources, describing the performance of each applied XP practice. For example, the "Shodan 2.0 survey" [63] was filled in by the whole team at the end of each release. The XP tracker tool "xplanner" [23], and different code analysis tools (for example, [1] [2]) were used for collecting quantitative data. The analysis of all that data provided an insight into the performance of the team and the XP process.

Conclusion This section concludes the paper by specifying what was achieved in the first year of practicing XP. It was observed that most of the practices were helpful for the m3 project which had multiple objectives to achieve. Those objectives included academic research, application development and business. The practices of pair programming, co-location, face-to-face communication, stand-up and planning meetings, and retrospective review meetings were given higher ranking, as compared to the rest of the XP practices. These practices contributed in the improvement of the process and had also increased the overall morale of the team.

The following figures represent a pictorial view of the some notable facts presented in the process. Figure 3.1 shows the segmentation of the main workload aspects after the first one-month release.

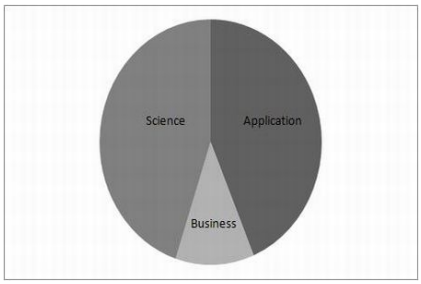


Figure 3.1: Contribution of Application, Research and Business aspects in a Release

The figures 3.2 and 3.3 show the planning boards of a release and iteration, respectively, presenting the story cards planned for

the development cycle.

Figure 3.2: Selected story cards on the Release-Board (Release Planning)

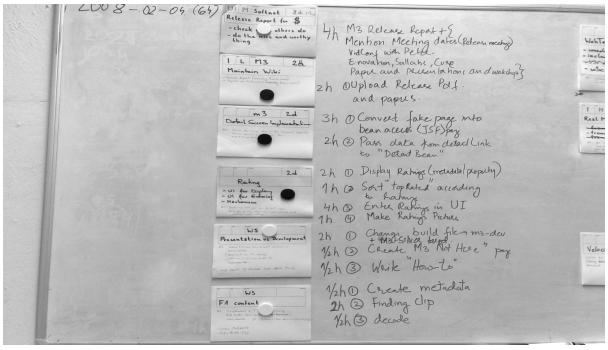


Figure 3.3: Selected story cards on the Iteration-Board (Iteration Planning)

Figure 3.4 shows a graph comparing lines of executable code, lines of test code and test coverage. The data was collected using Emma, a Java code coverage tool [1] and LinesOfCodeWichtel [2], and was based on the work performed during the second release of the project.

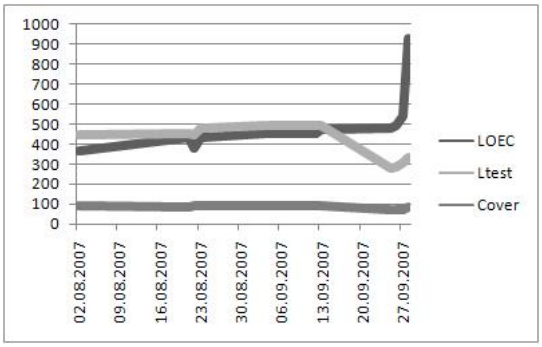


Figure 3.4: Executable code versus test code and coverage

3.3 Integrating Extreme Programming and User-Centered Design

3.3.1 "User Interface Design for a Mobile Multimedia Application: An Iterative Approach" (Item 3 in List of Publications)

Summary:

This paper introduces main features of the m3 application, the process that was followed as the design methodology, the important details of the development process, the development team, and the goals of the project. The main objective of the paper is to present the application, different stages of the user interface (UI) design, application usage scenarios and the approach to application development.

The following subsections give a brief summary of the paper.

Application This section provides an overview of different multimedia streaming applications, in order to have a comparison with the m3 application. The comparison has been made on the basis of the range of facilities provided by different applications which includes different options for streaming AV content, type of multimedia archives (for example, broadcast data or user-provided content), and a range of client side application with facilities like content search and Web 2.0 ([83]) features. It was established that none of these applications, at that time,

provided streaming of large archives of data on mobile phones with an advanced search facility.

Usage Scenarios This section presents different usage scenarios of the application from which the idea of the application was conceived. These usage scenarios place emphasis on the availability of the application anywhere and all the times as the application was being developed for mobile phones. The important usage scenarios which are defined are concerned with providing video(TV) archives for subway riders, and radio archives for car drivers who can search for their favorite program and watch or listen to it while traveling. The users may also perform general search in the content or watch any multimedia content recommended by other users of the application. It was suggested that the application should provide standard play options like stop, pause, and resume letting the users consume the content according to their own timetables.

Usability The paper also emphasizes the usability issues of the UI of a mobile application. It introduces the steps which were carried out to design the user interface in an iterative manner. The graphic representation of the UI design workflow (shown in Figure 3.5 in the original paper) shows the cycle of UI design process from writing UI related user stories by the customer to the design of mock-ups by the developers, and refining the mock-ups iteratively according to the wishes of the customer. Once a mock-up was designed in accordance to the specifications given by the customer, it was presented to the usability engineer who then gave usability related feedback on the mock-up which was then incorporated in the design during the next iterative development. Hence employing the "quick feedback-and-change cycle". The usability feedback was incorporated into automated usability tests to ensure that everything was working according to what was required.

User-Centered Application Design This section introduces the idea of UCD and its integration into the development process. It emphasizes that the design of UI cannot be separated from the design of the underlying application. It is also stated that to focus on the usability issues inherent with mobile application design, user-centered application design and iterative UI

design were integrated. It is also noted that not only the customer's opinion but also end user input is important for the usable design of the application [56]. It is concluded that iterative feature development and UI design process helps to evolve system according to the needs of the user.

User-Based Recommendations This section gives details of the user-based recommendations system which is suggested as another feature of m3 application, implementable by keeping track of user-id to help maintain a user profile by the system. The user data collection can be done by means of two information acquiring models, the interactive model and the behavior-based model. The interactive model is based on user-provided ratings of the multimedia content, and behavior based model is based on the usage data collected by the system. A combined interactive-behavior based system is also suggested.

Conclusion It is concluded that the critical factor for the success of this kind of application is the user acceptance, which depends upon the fact that how much the application fulfills the users' needs. For this purpose, the usability engineering plays its role. In case of m3 application this usability engineering is facilitated by the iterative development process of XP.

Figures 3.5, 3.6, 3.7 are also presented in this paper. The workflow presented in Figure 3.5 illustrates the iterative design approach. Figure 3.6 and shows the process of designing the UI from a paper mock-up.

Figure 3.7 depicts a refined process of UI design, showing the progress from paper mock-up (a refined version of mock-up given in 3.6) to HTML mock-up and then to the actual interface design.

3.3.2 "Agile User-Centered Design Applied to a Mobile Multimedia Streaming Application" (Item 7 in List of Publications)

Summary:

The goal of this paper is to provide a detailed overview of the integrated XP and UCD process adopted for the m3 project, and

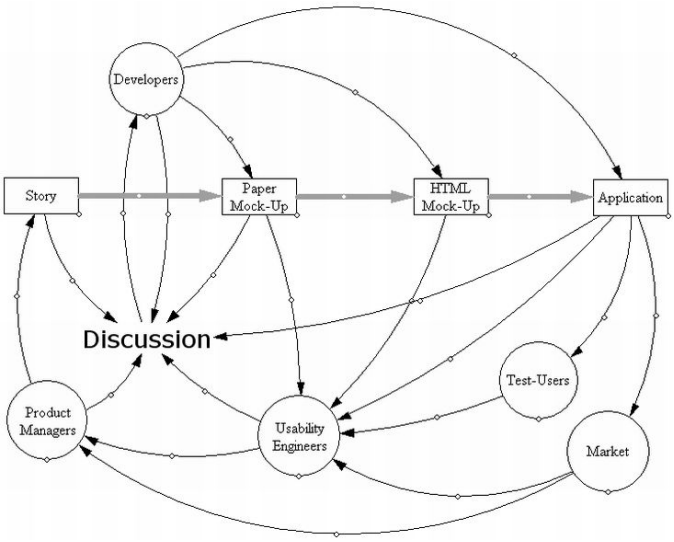
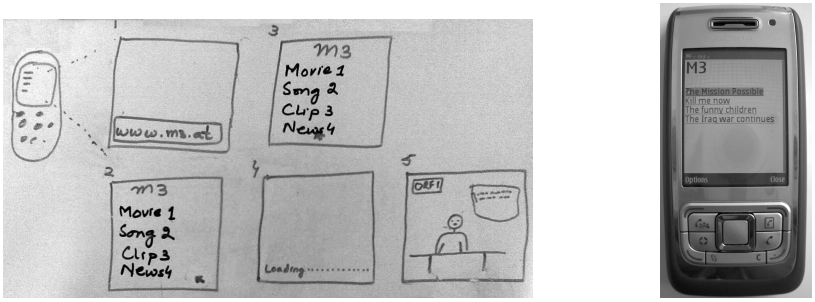


Figure 3.5: Iterative UI design workflow.

to show how this integrated process preserved and multiplied the benefits of the two approaches. That is, the UCD process helped evolving the application in conformance with user orientation and needs. On the other hand, the inherent quality of XP as being people-oriented in its practices and the agility of the overall process allowed the development team to work according to the wishes of the customer and outcomes of the usability testing procedures. The paper cites many references about researchers' view of human-orientation promoted by UCD and then draws a similarity of focus between XP and UCD. It conforms to the idea that that customer involvement is the most important success factor in software development projects [75]. A review of integration of agile software development approaches and UCD is given stating previous experiences of many researchers and usability experts. [44], [24], [74], [18], and [77] are some of the studies mentioned in this regard.

The following subsections provide a summary of the main sections of the paper to provide an insight into the XP and UCD in general, and the way these were adopted in the project. It clearly explains the team's approach to UCD. The paper provides a brief account of a usability study conducted in the context of m3 project as a working example of what has been established in other papers presented



(a) Paper mock-up. (b) Application on mobile.

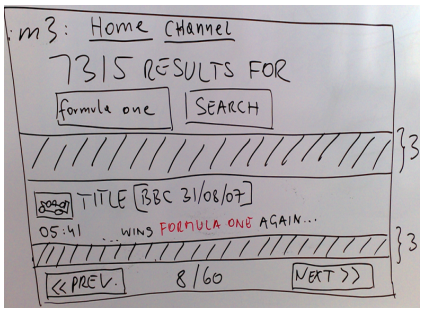
Figure 3.6: From paper mock-up to mobile: the first search-results screen.

about m3 project.

Similarities among XP and UCD This section lists down those practices of XP which can be seen as matching or fulfilling the needs of UCD [31]. Most importantly, the on-site customer of XP can act as the end user to test prototypes as suggested by UCD. Usability testing can be embodied very easily into the continuous integration and test-driven-development practices of XP, to run usability tests on regular basis whenever UI is modified or developed. Iterative development process is inherent both in XP and UCD, therefore, here again the two methodologies fit together very nicely.

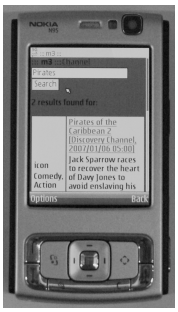
Project This section introduces the project, the team, and goals associated with the project. The main features and functionality of the m3 application constitute the main content of this section. It gives a detailed overview of the search, channels, media feeds, and clip detail functionality provided in the application.

Approach to User-Centered Design This section outlines the process of the UCD which is adopted for the project. It details the step by step procedure of the used to incorporate the UCD into the XP process. The pictorial view of this procedure is provided in the form of a loop named as "workflow" for UCD (see figure 3.5 in the original text) which is self explanatory and can easily be followed for this kind of XP-UCD process. A de-



(a) Paper mock-up.

(b) HTML mock-up.



(c) Final Application.

Figure 3.7: An additional HTML mock-up: a refactored search-results screen.

tailed implementation of all the steps involved in this workflow model is also provided in this section.

Usability Study The basic approach to UCD followed for m3 application was to make usability evaluations in small iterative steps. The prototypes of the UI of the system were designed and tested throughout the development process. The whole process of designing and testing of UI is depicted in the form of a loop which shows the real process followed and the people involved in the process. Figure 3.5 (in the original text) presents this process. The process used low-fidelity as well as high-fidelity mock-ups according to the requirements of the situation. End-user tests, user studies, personas, extended unit tests, usability tests, and usability expert evaluations were integrated into the process at different phases of development as suggested by the customer and the usability engineers.

Figures 3.8, 3.14, 3.11, 3.12, 3.13 have been presented as part of this publication.

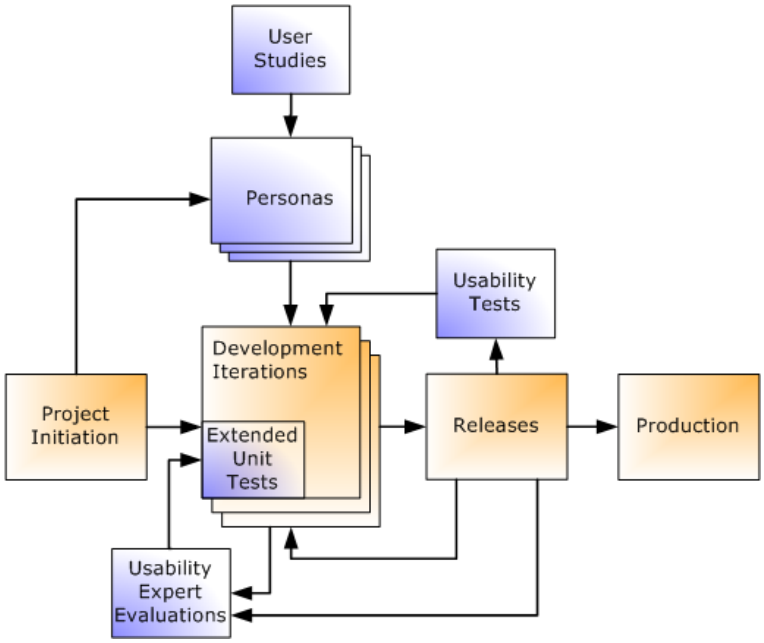


Figure 3.8: The integration of HCI instruments into XP [112].

Figure 3.8 shows the integration of XP and UCD using HCI instruments.

Figure 3.14 shows an example for prototype. The prototype of the home page is shown in the figure.

Figure 3.11 showing detail page with usability fixes in red circles.

Figure 3.12 shows the recommended menu layout and arrangement.

Figure 3.13 shows the space on top of the "Clip Detail" page which should be used more efficiently.

3.3.3 " Concept and Design of a Contextual Mobile Multimedia Content Usability Study" (Item 8 in the List of Publications)

Summary:

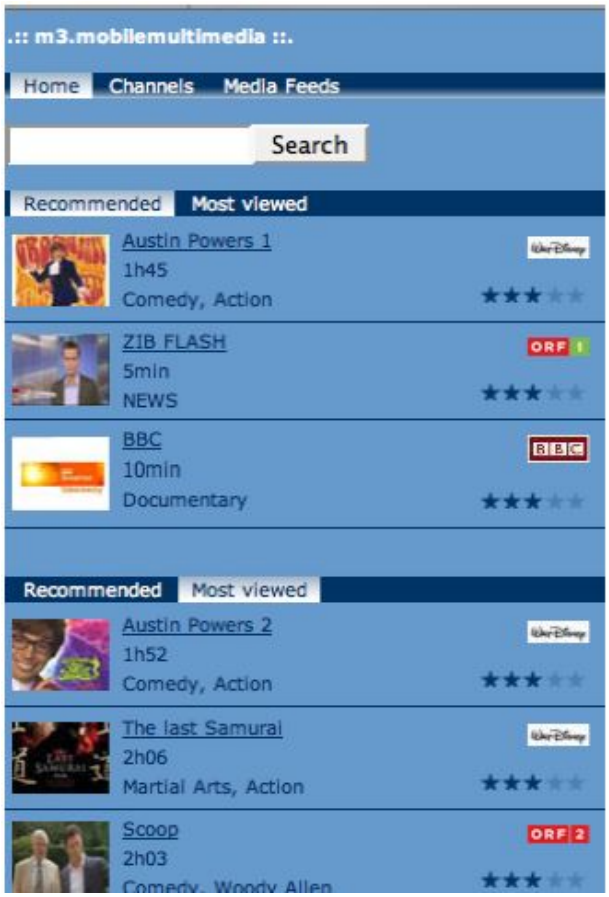


Figure 3.9: The prototype of the home page

This paper presents the setup of a usability study which was conducted in October 2008 in the context of m3 application. The main purpose of the study was to gain an insight into the field of mobile user-experience. The study was focused on gathering the scientifically relevant data which could help to analyze the pattern of user consumption of multimedia content, in order to facilitate further development and improvement of the application. Understanding the coherence among factors like content type, consumption times and consumption contexts was also an important goal of the study. Both qualitative and experience data were considered important for the significant results of this study.

Figure 3.10: The prototype of the Channel page showing the calender

The context of this study is the development process that was followed during application development. To get maximum benefit of user-centered design approach XP methodology was employed for the development of m3 application which facilitated the integration of multiple HCI instruments into the iterative development, hence evolving a comprehensive iterative UI design process ([112] [48] [64]).

The paper briefly states different aspects of the application development, UI design process, and user-centered design procedure adopted for m3 application. The highlights of the main sections of the paper are summarized in the following sections.

Application This section gives a detailed outline of the application under study and provides a tour to the main features of the application's user interface. The user interface is grouped into three top level pages. The main functionality of the application is depicted with the help of screen-shots of the application which show the main page called "Home" (see Figure 3.14 in the original text) containing feature for Search, and to displays top-rated and most recent clips. The page also pro-

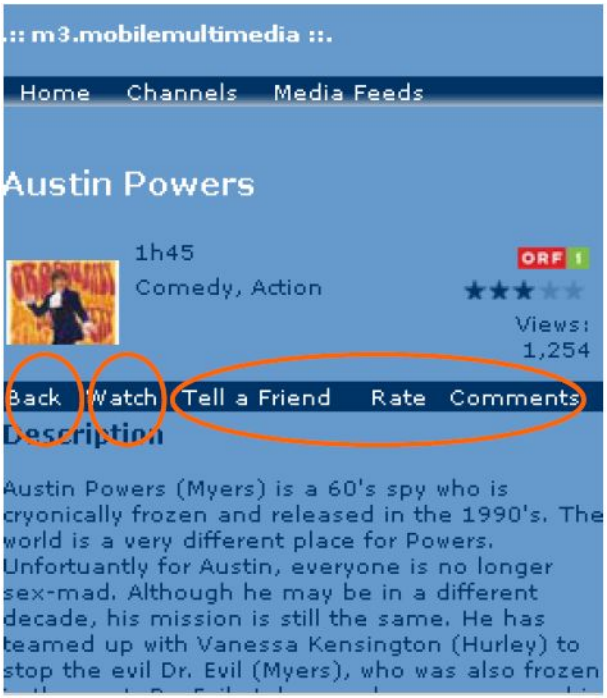


Figure 3.11: The menu entries without any visual separation

vides links to the other two pages named as "Channels" and "Categories" page. Details of functionality and usage of each feature is explained. These features include a searchable list of TV and radio channels along with their daily schedules, list of the whole available multimedia content filtered by particular categories under which the content is filed, a special page for each clip giving specific details about the clip along with other Web 2.0 features like clip rating and comments.

Selection of Respondents The basic criteria used to select respondents for the study is given in this section. 16 respondents were selected within the age group of 18 and 35 years. They were selected on the basis of their interests in watching different types of multimedia content. The topics of interest were politics, economics, technical studies, music and other entertainment categories. Respondents belonging to a particular interest group were to be directed to watch some particular type

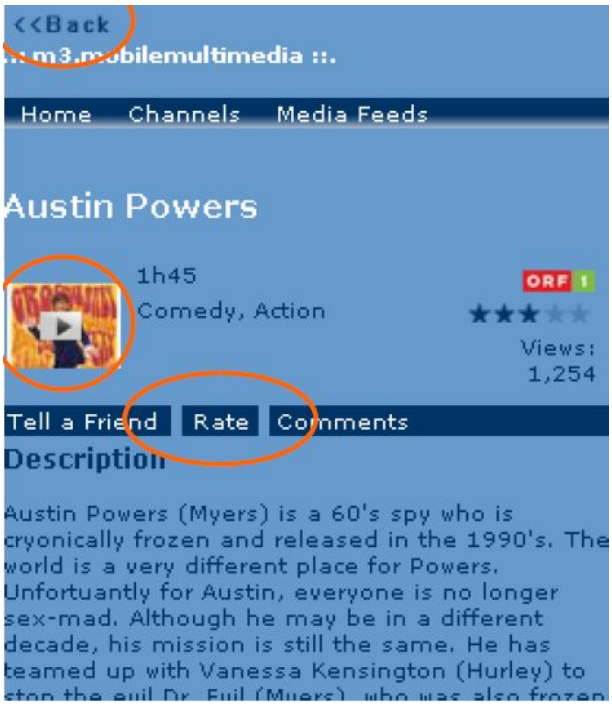


Figure 3.12: Improvements of menu layout and arrangement

of content on regular basis. For example, the respondents belonging to politics and economics interest group were advised to watch news related clips on regular basis throughout the span of the study. Another selection criteria for the respondents was their experience with using and consuming multimedia content on mobile phones.

Study Setup The overall setup of the study to be conducted is explained in this section. It gives details about the usability evaluation methods to be used and the choice of the type of content used for the study. It was decided to use two different methods for the study, namely "Diary study" and "Contextual overview". It was decided to use same respondents to conduct both methods. To follow a logical setting it was decided to first conduct the diary study and then the contextual overviews. All respondents were going to fill in questionnaires asking for some general information about them and their ex-

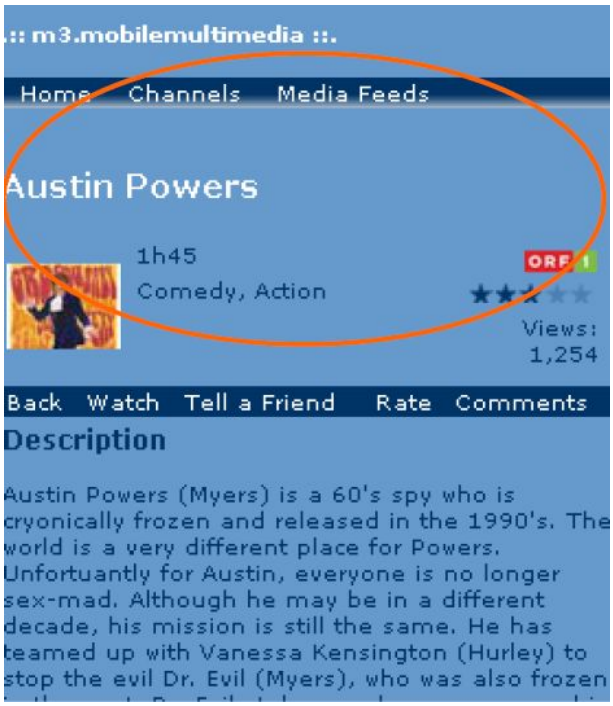


Figure 3.13: Usability fix: Use the space on top of the "Clip Detail" more efficiently

perience about multimedia content consumption.

Media Content To be able to gather relevant data, four specific content category types were selected, representing different information and entertainment levels. It was also taken care of that all clips were entirely different in audio-video content and in other static details. The clips of longer duration were selected for diary study, and for contextual interviews clips of shorter duration were selected to control the length of interviews.

Diary Study In the diary study each respondent was supposed to use the application for one week on the mobile device that was to be provided. The activities of the respondents while using the application were to be recorded in different log files maintained under the web server and streaming server of the application. The "user-tracking" implemented in the applica-

tion was meant to provide comprehensive information on the clips consumption (for example, when, how many and how long video clips were watched by different respondents).

Contextual Interviews In this study method the respondents had to use the application in four controlled and different "video" sessions. During each contextual interview (one video session in a specific context) a respondent was going to be accompanied by a usability engineer. The process included filling in a pre-questionnaire covering general and demographic data as well as personal experience of mobile multimedia consumption. The next step was to watch four videos from different content type categories and then provide a qualitative observations to the interviewer. A post-video questionnaire was also designed for respondents to collect more personal views of the respondents about the application. The application logs were maintained to collect the data. Different locations were to be selected to conduct contextual interviews to get significant results from the individual contextual interviews.

Expected Results and Conclusion This section provides a list of results that were expected to be captured from that study. The main results expected from diary study were the information about coherence between time of day of multimedia content consumption and content type, coherence between context of consumption and content type, along with the statistical data about average number of watched clips and average time a clip was watched. The expected results from contextual interview included availability of different contexts, availability of qualitative feedback from user on content types in different contexts and the availability of user experience data concerning context variables like light, noise, and being around in a public place. The results were to be correlated to the age and gender of the respondents.

Figures 3.14, 3.15 are given reflecting the process carried out during this stage of process.

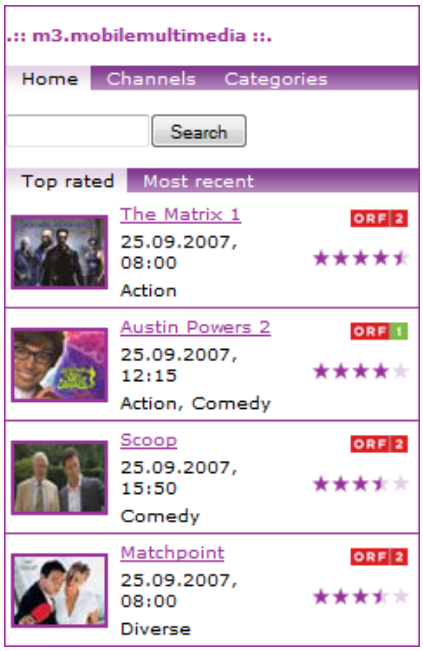


Figure 3.14: Home Page.

3.4 "Integration of Extreme Programming and User-Centered Design: Lessons Learned" (Item 10 in List of Publications)

Summary: This paper has been presented and published in the proceedings of the 10th international conference on agile software development. This paper is the sum-up of the usability process that has been conducted during the application development. It highlights the HCI instruments used and the process that was used to apply these instruments. There have been a lot of issues regarding the usability of mobile applications which are tried to be tackled in the context of this project, by employing a combined XP and UCD process in order to benefit from the practices of both methodologies. A lot of studies have been conducted in this context which show the growing interest of agile and HCI communities in integrating both methodologies.

The paper gives details about m3 project by defining this integra-

Figure 3.15: Categories Page.

tion process, and the way it was evolved and followed. It also highlights the important aspects of the HCI instruments that were used by providing a review of each instrument, and stating the lessons learned by the team. The paper starts by providing a brief account on the project and the development team. The main theme of the paper is to review what was done and learned by the team during the three years of the development process and the use of UCD approach from the beginning of the project. It gives a detailed account of a retrospective meeting conducted by the team along with the official off-site usability engineer who has been working with the team in close collaboration and whose usability related feedback had been incorporated into the application on regular basis. The HCI instruments that have been used for the usability engineering process are also discussed separately, highlighting how these were applied and how the team adjusted to their integration into the development process. Below is the brief summary from the main sections of the paper.

User Studies User studies are a way to get information about the

end users to uncover their requirements and expectations from the application. In the context of m3 project laddering interviews, field studies, and a large field trial study was conducted with 100 plus real end-users located throughout Austria. This trial study made use of diary studies, contextual interviews, laboratory usability tests, questionnaires, and focus groups.

Personas Personas were introduced into the process in the beginning but these were not actually used in the way required for these to deliver results. The personas were then explicitly reintroduced by the usability engineer to make the benefits of using personas apparent to the development team. Use of personas helped in refining the focus of the development towards on-site customer and the real-end user.

Lightweight Prototypes Both low-fidelity and high-fidelity mock-ups were used for UI development and usability evaluation of the application by the customer. To minimize the risk of developing only on-site customer focused application ad-hoc input was taken regularly from the off-site usability engineer. Conducting formal small usability tests after every release and access to an on-site usability engineer to further reduce this risk was a suggestion given by the projects off-site usability engineer.

Usability Expert Evaluations Usability expert evaluations were provided by the off-site usability engineer at different stages during the development by using instant messaging, email, and video-conferencing. That input was needed while writing UI related stories as well as during the implementation of the UI. A suggestion was to train the usability engineer in writing stories so that he could write usability related stories according to the usability requirements of the mobile platform.

Usability Tests Usability tests with real end users were conducted by the usability engineers, with developers as observers [49]. It was noticed that involvement of developers in the usability process refined the team's focus on the usability aspects of the application which was beneficial on the development side. To easily incorporate the results of the usability study it was suggested to conduct small usability tests after every three it-

erations and extensive usability tests after each release when a considerable amount of development on the UI design was made.

Chapter 4

Analyzing the m3 Experience

This chapter presents a theoretical analysis of the XP process implemented as the development methodology for m3 application.

4.1 Introduction

The m3 XP process passed through many stages of modifications and refinement during the development of the application. The research papers produced based on the m3 project are presented in chapter 3 which outline how the XP process evolved for the m3 project. As one of the main research goals of the project was to analyze the XP practices in order to find their best suited implementation for the team and the project, therefore, a conscious and continuous effort was made to understand the contribution of each practice towards the development of the application. The social and technical aspects of the practices effecting the team and the project were also investigated. The retrospective meetings which were held after every release played a vital role in this regard. A substantial amount of effort was put in these meetings by all team members to reflect on individual XP practices and on the overall XP process.

This chapter is based on the authors publication titled "Learning from Experience: The analysis of an Extreme Programming Process" [97] presented in the list of publications. In that publication the author specifically investigated the team's self learning process of XP practices. This self learning resulted in the evolution of the

XP process for m3 project, which has been presented in [45], [98], and in [97]. The following section gives an account of the XP process as it evolved in different stages and provides a theoretical analysis of the reasons for modifying or refining the implementation of the different practices during the application development.

4.2 Learning from Experience: Analysis of the m3 Process

Software development is a dynamic process which is affected by the changes in the product market, as well as, in the user requirements. Agile development methodologies tend to tackle this problem by incorporating an iterative process to answer the continuous flow of user requirements throughout the development process. XP, which is a widely practiced agile methodology, asks for a customer on-site who works in close collaboration with the development team and manage the changing requirements into the development process. The inclusion of customer input in the routine development process is not an easy task, as it requires a lot of effort from the development team and flexibility in the process itself. Also, the setup of a software development process varies with the type of organization and the product that is being developed. This process adaptability must be paired with flexibility in the process which needs to be continuously molded according to the changing business demands. Keenan [58] defines different ways of process molding and tailoring which are termed as static process tailoring and dynamic process tailoring. Static tailoring is defined as the changes made in the process before starting the project and dynamic tailoring is the "continuous process adaptation" made during the software development. Process adaptation is quite ignored in the first version of XP [8], introduced by Kent Beck. However, in the later versions of the book ([9] [10]) it is said that the process should be adapted according to the requirements of the organization and the product. This opens the doors for XP process improvement which provides software development organizations with mechanisms to evaluate their existing processes to identify possibilities for improving, implementing improvements and also to evaluate the performed improvements [26] [93].

4.3 Process Tailoring for Process Improvement

Agile software development principles give a high value to the dynamic process tailoring during the on-going projects. Regular process retrospectives greatly help in this regard [59]. Agile principle of self-organization also dictates that enough flexibility exists within the process and the development teams have enough authority to adjust the process according to the needs [38]. Release and iteration-wise agile retrospectives as part of the XP process help overcoming the shortcomings of traditional post-project retrospectives by opening a space for the dynamic process improvement [19]. These retrospectives provide a formal platform for discussing the problems and suggesting improvements in the process and incorporating them dynamically in the coming development cycles, which are not possible using traditional retrospectives that allow learning only from finished projects [61]. The process adjustment and improvement in this project is mostly based on continuous team reflections made on the process after iterations and releases. The team's experience also shows that start of a new iteration or release is the most plausible check-point to introduce a change in the process.

4.4 XP Process Background and Setup

Although it has been suggested by many experts that a step wise introduction of agile practices must be made and many agile practitioners like Eckstein [22] suggest implementing one technique at a time and addressing the most pressing problem first.

The m3 team opted to use the "big bang" [69] concept of methodology implementation for the said project. As the project was a "green-field" project and none of the team members had any prior experience of the XP practices, thus, they decided to use only XP, not mixed with any other development process, in order to experience it in "pure" form. The standard books of XP by Kent Beck [8] [9] [10] were used as initial guide for setting up the process details. The team used the read-discuss-and-learn loop to understand the process. No initial training was taken by any of the team members. It is also felt by Lindvall [68] that agile methodologies require less training and practices like pair programming are a type of continuous training which is better than explicit training. He further

suggests that the application and implementation of agile practices can be learned by self training as many successful teams have done it.

In the said project the initial process setup included the practices of planning, pair programming, daily stand-up meetings and small releases and iterations. Infrastructure setup was made for continuous integration and a coding standard was adopted when actual programming tasks were started. Different experience reports describing XP paradigm were also studied. The team also applied some of the practices in the way that "successful" teams had defined. One of the main ideas that were taken from another team's experience was to pair-work in blocks of time [27]. Two-hour working sessions were defined for a whole eight-hour day. Hence each working day consisted of three two-hour pair-working sessions and breaks in between. This was somewhat like the "Pomodoro" approach [30] which proposes to work in 25-minute sessions and each session is followed by a five minutes break. The initial setup of some of the practices as defined in [45] is summarized below:

- The first release was taken as a one-month release with four iterations of one week each.
- It was a common decision that every task, including coding and non-coding tasks, will be done in pairs. (Many managerial and infrastructure setup related tasks have been done by the team in the first release. The team has also worked on a scientific paper writing assignment which is also done in pairs).
- Three types of stories were identified by the team: Application, Science and business.
- The business person, also a permanent member of the team, was identified as the on-site customer.
- The work was done 40 hours per week.

During the initial stages of the application development the team members had different concepts and views about the XP process. Figure 4.1 shows the subjective analysis of the XP practices used in first three releases (that is, in R1, R2 and R3) of the project. The team used Shodan 2.0 survey [63] to analyze the XP practices adherence in the team. As the team was learning and implementing the

XP process at the same time and was working to figure out the suitable way of using different practices the graph of individual practices can be seen go up and down from one release to another. For example, in case of pair programming the team had been experimenting with different styles and schedules for doing pair programming and was not much satisfied with most of them so the graph is gradually going down from first release to third release. Whereas morale of the team starting to rise again from release 2 to release 3 as it started to learn and understand the logic behind using most of the XP practices.

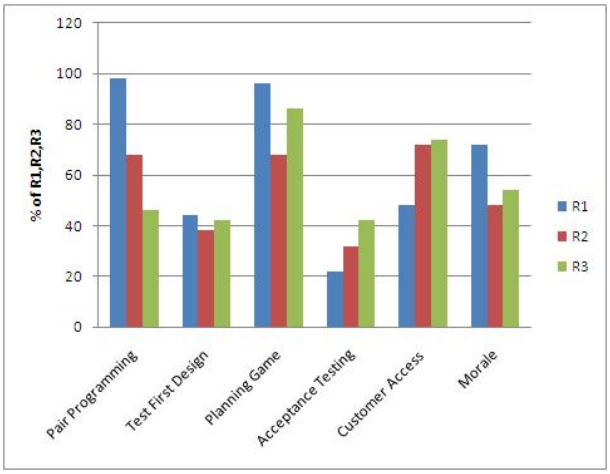


Figure 4.1: Subjective analysis of XP practices after third release using Shodan 2.0 survey [63]

4.5 Iteration-wise Improvement Process

Agile practitioners greatly advise agile project teams to reflect on and iteratively improve their behavior in a logical way [91]. For the said project team the practice of conducting iteration and release retrospective has played the focal role in understanding the process and in improving the team behavior. The team had been conducting these retrospectives at the end of the iteration and each release to take a critical view of all the major events occurred in the concluding development cycle. The team members discussed

any technical, business, process or team related issues that came up during the concluding development cycle. The issues were then resolved with the consent of the whole team. The team members especially emphasized over XP practices that were being used and discussed them in detail. Any practice was then partially or completely modified (as needed) to improve the overall process. These end-of-release/iteration retrospectives have proved to be very useful as they provide an opportunity to settle down the details of the process without disturbing the actual work. Also, these retrospectives have allowed the team to learn-by-experience and to have a fresh-start again. Table 4.1 shows a set of XP practices for which the team was satisfied that these were working to a greater extent as compared to other practices after three releases in the application development.

Table 4.1: Key practices of release R1, R2 and R3

- Co-location
- Release/Iteration planning
- Pair programming
- Test-driven development
- Sustainable Pace
- On-site customer
- Continuous integration

XP practices were always at the center of discussion as research in different aspects of XP was the goal of all team members. The team had even taken one week out from one of the releases when some major changes in process implementation were to be made. As the team was working in an academic research setting, therefore, it was possible to do so. This ability to self organize the development process has not only helped the team to improve but also to understand the practices for which none of them had any previous training. Each of the practices had gone through a continuous adjustment loop over multiple development cycles. This iteration-wise improvement of practices had made it easy for the team to clearly understand the situation and make the change accordingly.

The rest of this section outlines each implemented practice one by one and defines the way the details of the practices were modified since their initial setup defined in section 4.4. The main reasons of the modifications are also provided.

4.5.1 Release and Iteration Planning Process

The planning process had been a two-stage activity since the start of the project. The internal details of the planning process had gone through major changes at many stages. These details are briefly explained below:

Release and Iteration length

Release length had been changed from one to two months and then to three months. After a couple of two-month releases, when the planning process matured and estimates became better the team decided to start with three months release cycles. This helped to get rid of extra administrative work for the planning process and also to save time which was being spent in organizing release planning meetings which, at that time, were also attended by the delegates from the project partners. Similarly, it was decided to have two-week iterations instead of one week iterations to save the time spent every week on iteration planning.

User-Stories

The following items had been discussed and tackled regarding user stories since the initial process setup.

- Different Story Types: Initially the team was using three types of stories, that is, application related stories (called application stories), and business related stories (called business stories, for example, stories to prepare for a business meeting with project partners), and research/science related stories(called science stories, for example, writing a technical paper for a conference). After some time another story type was defined called "miscellaneous" stories which included developer stories as well as management related tasks. Many problems were encountered due to this division and need of different story types. The major problem was due to different planning requirements for different story types. For example, science and business stories required too much time to complete and hence there was always a feeling that it was not possible to give suitable amount of time to actual development work in iterations. Figure 4.2 which shows the graph of project velocity over 24 iterations

clearly depicts this problem. Although a lot of work had been done from iteration 11 to iteration 17 but none of it was related to application development, rather it was almost all for non application related stories. That is why the velocity of application development became zero. As no time could be given to application related stories. Also, estimating stories like writing a paper was quite illogical.

Finally, with the consent of the whole team, it was decided to plan and estimate only application stories. Science related tasks and business related tasks and other managerial tasks are done whenever required but are not calculated towards iteration and release velocities. The developer stories (infrastructure stories and spikes) were defined but are attached with some application story.

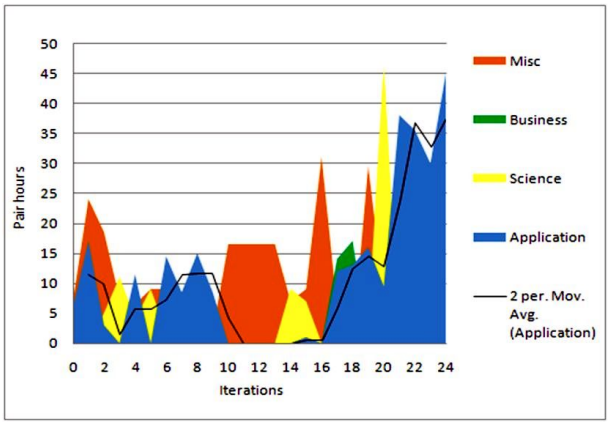


Figure 4.2: Velocity of application and non-application related stories after first three releases. Showing distribution of effort among application and other story types

- Planning Stories: As the team proceeded with planning and estimating different types of stories it was understood that planning non-technical stories, such as business related and science stories, was very difficult in the sense that such stories could not be estimated properly. Business and science stories could not be divided into smaller stories. Planning those big stories in any release and iteration meant that more than 50% of the working velocity was eaten up by them and sometimes

virtually no time was left for application stories. As described above, the team first took out business stories from planning process. That solved a lot of planning and estimation problems and the working velocity got better. Seeing the improvement, the team did the same with science and miscellaneous type stories. Developer stories were considered technical debt and were done along with application related stories when required.

- Story Estimation: From the beginning of the project the team had been estimating stories in real time (in days and in hours) and the time was calculated for pairs (as pair days and pair hours). Some of the developers were not comfortable with this type of estimation as due to the unpredictable nature of some stories it was impossible to estimate them for some logical time duration. The team then started estimating stories in story-points. The team used poker game cards bearing numbers 0, 1, 2, 3, 5, 8, 13. A given story was estimated by all the developers simultaneously by showing up a card of his/her estimate. Close estimates by all developers resulted in a consensus and too much different estimates resulted in a discussion between the highest and lowest bidder. After discussion an agreement was supposed to be reached and the story was re-estimated.

- Story Writing: Many changes were made in the story card template and the way the stories were written. The team had understood that writing a story in a form that it was actually understandable and featured all required ideas when the developers actually start working on the story was also a big task. It was not as trivial as writing a description of a task. Even a simple statement could introduce many misunderstandings between developers and the customer at the end. The same sentence was usually understood differently by people. That meant that the customer had to remain with the developers all the time to clarify the actual sense of the story whenever demanded by any developer. It became quite difficult as the customer was also the business person and had to deal with the business side of the project. Also, generally, even if the customer is there all the time with an XP team it can become very cumbersome for the customer to explain story cards again and again throughout the release. Therefore, after an year's ex-

perience with story writing, the team had shifted to the story writing template specified by [81] and found it quite useful in getting rid of all the ambiguities and misunderstandings about stories during iterations. Although it took some time for the team to understand and accurately use the new story template but when all team members had clearly understood it they were actually very happy with that as the developers considered it as the best thing to clarify everything about a story at planning time.

- Story Size: In the beginning the team did not pay too much attention on the size of the story. Big stories had also been a source of bad planning and estimation. After much experimentation, with the consent of the whole team, and according to the suggestion given by some XP practitioners, it had been decided to write stories of considerably small sizes. Also, the stories were decided to be written by the customer together with a developer one day before the planning day. Previously, when the team was strictly following "customers write and developers estimate" it was quite difficult for the customer to understand the depth of stories and extent of the detail needed to make it unambiguous and clear enough to convey the real matter. Also when the customer was putting up stories at the release planning day or adding stories at the iteration planning, many stories were starting a point of long discussions among the development team and the customer in order to understand and clarify the story. Sometime it meant re-writing each and every story. Most of the time it usually resulted in anguished developers and depressed customer after the planning meetings.

4.5.2 Fixed Time slots

Two hour blocks system of pair programming was dropped after a couple of months of practice. Working in two-hour blocks was not a good experience. It was felt by the developers that it was tiring and too restrictive. Due to this reason the pair programming timings were soon made flexible (in the range of one to three hours) to introduce more agility. Pairing times finally depended upon the pair partners. As there were only five developers so frequent pair switching was anyway not possible.

4.5.3 Pair Programming

The practice of pair programming had been applied since the beginning of the project. The team composed of 5 developers with one developer always working solo. Any person could accept to work as solo and would work on either a spike task or any other task which was logical to be done alone.

In the beginning all types of tasks, including administrative and science related tasks, were performed by pairs only. But with the exclusion of non application related stories the pair work was also then required only for programming and design related tasks as it had become clear with experience that pair working on non-technical tasks was not as productive as it was expected.

4.5.4 Customer Role

Initially, customer role was not well defined. User stories were written both by developers and the customer. As developers also had a say in selecting the stories for release and iterations it lead to long discussions during planning meetings. Also, due to unclear definition of acceptance criteria by the customer it was difficult for the developers and the customer to understand when to mark stories as finished. With this experience finally it was decided that the customer was the only one to define and accept stories. Stories were marked as finished only when these were formally accepted by the customer.

4.5.5 Usability Engineer Role

Initially the development team was directly communicating with the off-site usability engineer of the project. The developers could ask the usability engineer directly for usability related feedback on the user interface design mock-ups. The customer, as well as, the product manager were also communicating with the usability engineer regarding the same or some other issues. It sometimes created confusion among the developers and the customer. Due to this reason, it was decided that in general the customer and the product manager would do all the correspondence with usability engineer regarding the usability feedback. The usability engineer and the developers might talk directly with each other if it was necessary

on some technical grounds. Also, the developers and usability engineer were communicating for research related issues, for example, usability tests and user studies.

4.5.6 Test First Design

Only one of the developers had a previous experience of test driven development (TDD). Rest of the team took some time to get used to this type of coding habit. Some of the team members even had reservations in the beginning about the usefulness of the practice in some programming assignments. So it was not made a mandatory practice for the team. During the project the team attended a test-driven development workshop which clarified much of the misconceptions of team members about TDD. In the end every team member started making a conscious effort to use TDD and tried to use the practice in routine coding assignments.

4.6 Learning Protocol: Tips and Tricks

The author has tried to gather some learning experience in the form of tips for new developers and teams who want to start with XP process. The tips presented here are given in the context of learning issues and do not cover an experienced professional's point of focus.

- Although not necessary, as suggested in [68] but for new developers short training workshops especially for pair programming, planning and customer related tasks can be very helpful and productive.
- Customer role should be made explicit so that the developers have very clear idea where their opinion is required and where they do not need to make any comment.
- Instead of experimenting with story writing, story card template and velocity calculation new developers and teams should begin with some already experimented templates. This will save them from getting lost in the small details and instead they will be able to concentrate on the logic behind using the practice of story writing and velocity calculation.

- A culture of short discussions among team members should be established from the very beginning so that everyone gets used to the way of communication of each other which is very important to take positive participation in planning and reflection meetings. Someone's personal negative point of view about any team member can hinder in the discussion as well as in the development process. Short stand up meetings in which every team member can take part can be a good training for more important planning and reflection meetings.
- In the context of m3 project the team has learned a lot towards the usability process and its integration on XP. It is thus understood that the role of the usability engineer should be clearly defined and understood by every team members. This is necessary for positive communication among managers, customers, developers and usability engineers. Also, if the project is supposed to be a usability intensive application development (for example a web application) then some prior training in usability studies and exercises will make it possible for the developers to understand this concept, which will be very helpful in latter stages in the development.

4.7 Conclusion

The adaptability of the XP process has helped the m3 team to mold individual practices according to the requirements of the project. The reason behind taking an overview of the change and improvement made in the process since the beginning of the project is to find a pattern between the changes in the business requirements and their effects on the process. It is learned that a need to modify the implementation of a practice arises when it appears not to be helping the development process and this becomes very visible when some major change in the business requirement is to be accommodated. We have seen many changes in the implementation of planning and pair programming practices due to this issue.

Dynamic method tailoring and continuous retrospectives on the process go side by side. The ability of learning lessons from experience during a process cycle and the provision to tailor the process according to the change required as soon as it becomes logical is a

great benefit to agile teams.

Experiences of other agile teams and practitioners have also helped in tailoring the process. Although, this report provides a purely theoretic picture of the team's experience, still this analysis may be sufficient for the general agreement on many facts of experience. For example, the "big bang" way of deploying all XP practices at once may lead to slow and complicated process adaptation for a team. As it happened in the said project that the team tried to master a lot of practices like planning different story types, pair programming, customer involvement, and test-first development at the same time. Each of these practices should be given enough time separately to let the developers and other team members take maximum benefit from them.

As we have also learned from the experience of other teams, our experience will also be of interest specially for those teams who are working under similar circumstances as being in academic setting and working on a research cum business project.

Also, we have understood that not only experience of developers counts towards the success of the project but also the experience of the XP customer as being an on-site customer also counts towards good and proper planning and hence in " on-time product delivery" by the team.

Finally, the analysis of the whole XP process discussed in this chapter depicts that XP is not a strict process that needs to be followed on as-it-is basis. It leaves a room for experimentation and self-learning. Hence, provided with some basis guidance, it presents a good opportunity for new developers to learn and understand the overall structure and requirements of software development activity.

Chapter 5

Framework of Knowledge Management in Extreme Programming Education

This chapter gives details of the case study selected for the research. The chapter outlines the research goals defined for the study, study setup, design and organization of the case study. An outline of the actual study conducted in order to collect the data for the empirical analysis of the study is also presented. The chapter concludes by giving the framework in the form of guidelines for an XP course organization and management.

5.1 Introduction

As the ultimate goal of the research was associated with defining a teaching framework for graduate software design methodology course an ongoing university software engineering course was taken as the case project. The author took the role of the instructor to facilitate the research and at the same time to guide the participants of the course, hence fulfilling the conditions for an action research project. The following goals were identified which define the basis of the research.

5.1.1 Research Goals

- The major goal was identified as of defining a framework for structure and organization of an agile software development methodology course in order to analyze the learning and teaching issues of XP practices and agile knowledge management. The desired objective was to provide students a "meaningful experience" with learning a software development methodology which becomes useful and helpful for them while working in software development industry in professional life.
- To develop an innovative way of teaching XP so that the students not only learn XP as a software development methodology but can also visualize the development process as an organized activity to achieve a goal. At the same time the students should be able to grasp the logic behind individual values, principles, and practices of the methodology, and see how the practices fit together and contribute towards software development.
- To expose and emphasize the knowledge management aspects of individual XP practices and to demonstrate how formal as well as informal knowledge management helps in the learning process.
- To build a culture of student involvement in learning activities and the activities of fellow students on top of this knowledge management structure in order to create a knowledge sharing environment.
- To highlight the importance of communication among students by integrating social structure of XP practices and role playing activities with pedagogical techniques of communication.
- Along with this the research was to define a general framework for teaching software engineering and other subjects which have an extended objective to teach the theoretical bases as well as to present the practical aspects of that theoretical base.

5.2 Basis of Study Setup

The setup of this study is greatly inspired by the educational framework suggested by Hazzan and Dubinsky [33] which is based on a collection of 14 principles to teach Software development methodologies (SDMs). A snapshot of this framework is given in figure 5.1. Hazzan and Dubinsky have used the concepts of XP to exemplify these principles. This framework is a follow-up of their experience with teaching a course of XP at a university [32]. Van der Duim et al have analyzed the use of industrial projects in industrial setting for educational software engineering projects [103]. They have highlighted different educational and organizational problems occurring due to this real life setup for education. It is pointed out that pure industry setup introduces a lot of complexity for the students, which is due to their lack of technical knowledge and structure and experience to deal with real life situations, the lack of social and human content in industry-like settings, and the lack of participation by some team members in each team. van der Duim et al have based their study on the seven point guidelines for education given by Chickering and Gamson in [16], which are given in table 5.2.

Knowledge management is another important concept worth adding to the guidelines provided by Hazzan and Dubinsky (figure **??**) and by van der Duim et al. (figure **??**). Levy and Hazzan describe in [66] that although not new the combination of knowledge management and agile development methodologies is emerging as a requirement of business organizations. Both knowledge management and agile development methodologies not only benefit each other but also tend to diminish the shortcomings and risks associated with one another. Bjørnson and Dingsøyr ([12]) conclude that agile companies are more satisfied with their knowledge management activities as compared to traditional software development organizations. This eventually leads to the concepts of agile knowledge management (AKM) which removes barriers of long term organizational goals and the corporate hierarchy of roles, which exist in business organizations to "promotes collaboration and sharing" of meaningful knowledge [72]. Project management also applies to this

Table 5.1: Framework for teaching SDM by Hazzan and Dubinsky

Principle 1: Inspire the SDM's nature
Principle 2: Let the learners experience the SDM principles as much as possible
Principle 3: Explain while doing
Principle 4: Elicit reflection on experience
Principle 5: Establish diverse teams
Principle 6: Assign roles to team members
Principle 8: Manage time
Principle 9: Be aware of the abstraction levels
Principle 10: Inspire a process of an on-going and gradual improvement
Principle 11: Use metaphors or "other world's concepts"
Principle 12: Consider the awareness needed for the implementation of the SDM practices
Principle 13: Listen to participants' feelings towards the presented SDM
Principle 14: Emphasize the SDM in the context of the software world

situation as business organizations need a prompt reaction to change propagated by timely collaboration and sharing of knowledge among stakeholders, hence integrating AKM and project management concepts. All this knowledge management structure revolves around each and every person associated with the development process including developers, managers, customers, other stakeholders, and also usability experts associated with the team and the project.

Table 5.2: Good practices by van der Duim et al.

Good practice : Encourages contacts between students and faculty
Good practice : Develops reciprocity and cooperation among students
Good practice : Uses active learning techniques
Good practice : Gives prompt feedback
Good practice : Emphasizes time on task
Good practice : Communicates high expectations
Good practice : Respects diverse talents and ways of learning

5.3 Study Setup and Case Selection

The author perceived her mental model of learning and teaching software development methodologies from her experience of working with XP for m3 project, as described in chapter 3 and chapter 4. The author visualized a set of guidelines for herself (given in 4.6), which were further enhanced by the concepts of knowledge management in learning environments along with the concepts defined in section 5.2 and including the concepts of active learning (for exmple, [13]), role playing (for example, [37] [3]), using students as teaching devices (for example, [86]), the use of experimentation in classroom (for example, [6]), and the benefits of simulation in teaching software engineering concepts (for example, [20],[84]).

To reach the goals associated with this study, outlined in section 5.1.1, an XP methodology teaching course was reorganized and restructured. It was a regular course being conducted at Graz University of Technology since 2005. Due to the structural and organizational flexibility provided by the university the course was the appropriate case study providing a platform for defining, implementing and analyzing the perceived XP teaching model. The objective was to introduce XP on a platform which could help the students to visualize the actual working of the methodology in real industry setup while eliminating the dangers of introducing extra industry-related complexities (as mentioned in [103]). At the same time, emphasizing the knowledge management concept by bringing

out this inherent quality of the practices of XP and making the students understand its role in learning which in turn teaches sharing of knowledge. To enhance the effectiveness of this model setup the concepts of project management related to software development industry, including corporate roles and the communication channels which exist among them were also introduced. This provided an environment simulating the industry like situation at an academic setting, providing the students an experience of simple but real software development project, with an ultimate goal to analyze how to shrink the conceptual gap between industry and academia without effecting the academic environment of university [25]. Many experience reports have been presented by researchers and practitioners showing different aspects of teaching XP in classrooms. [87],[78],[110],[80],[94],[28],[35],[32],[43] are only some of them which the author had studied in the context of planning and organizing this course. [57] and [73] report on the of XP practices especially pair programming to teach basic programming skills. Hedin et at. [36] has specifically reported on teaching XP to a large group of students which was also the case with the course selected for this research.

5.4 Organization of the Extreme Programming Teaching Framework

The XP teaching framework is defined in the form of the organization and structure of the software engineering course selected as the case study. The course has been a regular university course and is part of the computer science curriculum of Graz University of Technology. The course is attended by both bachelor and master level students. Both groups of students attend this course in different semesters of their study programs. Due to this reason the said class is a mix of students with different levels of education, different experience of computer science study, and also some of the students (mostly master level students) even have work experience of software development industry. This also adds a further dimension in the proper organization of the course so that the course is of interest for all of the attendees according to their existing level of knowledge.

Extreme programming methodology is being taught in this course

as a software development methodology since 2005. Since then the course has been evolved in organization and structure using different pedagogical aids and techniques. Mostly, attempts have been made to make the course as practical-oriented as possible. It has been the main critical feedback given by the students in previous years' end-of-course reviews.

The author participated in organizing and conducting this course offered in summer semester in 2009 [96]. With an academic objective to transfer the personal experience of working with XP methodology ([47] [98] to the students which the author had learned through hard experience ([97] [45]). The research interests of the author defined in previous sections also motivated her to take full participation in organizing and arranging the structure of the course so that it fulfills the education needs of the students and at the same time become a case study for the research.

The rest of this chapter presents the course organization, the knowledge management perspective of the practices of XP, and the implications of project management and industrial setup in order to provide the students with an experience of industry simulated environment.

5.5 Course Organization

The course was divided into two clearly defined and organized parts, named as "XP-Visualization" phase and "XP-Realization" phase. As the names depict the first phase was more about learning the concepts and design of different values, principles and practices of Extreme Programming methodology and the second phase was about implementing those practices in a software development project. This section defines the structure and organization of both phases; giving the details of how the two parts were managed and scheduled. Table 5.3 gives the general outline of course specifying the clear separation between the two phases and the amount of time spent on each topic covered in the class. The time is mentioned as "XP-days" which is equivalent to eight hours. This unit of time also suits the concept of XP and makes sense as it was attempted to teach XP in XP way.

Table 5.3: Course Outline

Topic	**Duration**
XP-Visualization Phase	
Introduction and orientation	1 XP-day
XP hour	1 XP-day
Planning game: planning process in detail	1 XP-day
UIPaper mock-up design and usability exercises	1 XP-day
Test Driven Development exercise	1 XP-day
Mid-term examination	1/4 XP-day
Mini-project	3/4 XP-day
XP-Realization Phase	
Release 1. Iteration 1.	2 XP-days
Release 1. Iteration 2.	2 XP-days
Release 2, Iteration 1.	2 XP-days
Trade show/ project display Preparation	1 XP-day
Final exam and project presentation/tradeshow	1 XP-day

In the context of the research interests of the author, a number of formal surveys were also designed and conducted during the course. The questions of the surveys were focused on gathering information regarding the applicability of XP methodology as a software engineering course and also the acceptance of the XP practices by the students. Also the students were asked to give their opinion about the way the course was organized, especially regarding the project management and industry simulation concept presented and highlighted in the way the course was conducted. For this reason three different surveys were scheduled at different stages during the course. The details of the surveys will be given in the next chapter. Along with the surveys informal feedback from students was also collected through planned and unplanned discussion sessions, planning and retrospective meetings during the whole semester, also from the course "wiki" which also contributed to promote the project management and industry simulation concepts of the course. This research base provided a mix of qualitative and quantitative research methods which is thought to provide an opportunity to gain a better understanding of the factors that impact students' experience of XP methodology [76].

5.5.1 Part-I : XP-Visualization phase

XP-Visualization phase was designed keeping in mind the necessary and basic concepts of XP, and the level of knowledge of majority of the students about software development and programming technique. The time constraint had also been given importance as the whole duration of the course was 14 weeks. Table 5.4 provides an outline of this phase. The objective of this phase was to introduce XP practices in most meaningful and constructive way, and at the same time to provide the students with an opportunity to get a hands-on experience of XP methodology. In the same context the knowledge management concepts associated with XP practices were also elaborated so that the students get the optimal learning benefit from the course. All XP practices included in XP-Visualization phase (refer to table 5.4) were carried out as part of the exercises presented to the students using the concepts of active learning [13] to motivate the students by using unconventional ways of teaching and for faster distribution of knowledge among student software development teams [96].

The knowledge management and issues with learning XP methodology [54], explored and analyzed by the author in the learning experience case study [97] explained in chapter 5, prompted the course organizers to decide that the emphasis would be placed more on providing training in the XP practices than on scope of the application to be developed in the second phase of the course. For this reason, the modes of active learning were adopted further enhanced by the implementation of role playing strategies ([37] [3]), teaching devices [86], the concepts of experimentation in classroom [6], and the use of simulation in teaching software engineering concepts [20], in all exercises carried out in this phase. Each exercise was meant to teach particular XP practices. The idea of exercises was further refined using the concepts defined by different XP coaches and world renowned agile practitioners, presented by them at different international XP and agile methodologies related workshops and other research and training forums.

The following sections briefly mention the main features of each of the exercises conducted during this phase.

Table 5.4: Outline of XP-Visualization Phase

Day-1	Introduction to course structure and organization Introduction to XP and Agile.
Day-2	XP Hour: Full day exercise including brief introduction of XP process involving students in XP roles, imposing co-location, story writing exercise, prioritization and estimation, demonstrating pairing concepts, acceptance testing, concept of releases, iterations, and velocity calculation.
Day-3	Planning Game: Planning process in detail, role of customer on-site, feature identification, story template, distinction of roles in story writing and estimation. Concept of estimation in points and in real time. Concept of planning according to previous iteration's velocity.
Day-4	Paper Mock-up and Usability exercises: Concept of designing paper mock-up and introduction to usability testing, concept of integration of usability testing into XP process.
Day-5	Test Driven Development exercises: Introduction to TDD, unit testing, test first approach, code sharing, code integration.

Extreme Hour Exercise

The basic idea of this exercise was to introduce the whole concept and working of XP process in the simplest way and in the shortest period of time. This exercise is mainly used by agile practitioners and coaches to introduce XP to software development teams who do not have any prior experience with XP. This, usually a one-hour exercise, is known as "extreme" exercise as it encompasses every aspect of the XP methodology in a very short amount of time. In the context of this course a non-programming but "creative" problem was taken as a "project" for this exercise. The students were divided into teams, just like real XP teams, where each team had a manager, a customer, a coach and some developers. The developers had to design a paper model of an aircraft according to the specifications provided by the customer. The task was divided into two iterations of 30 minutes each. The exercise included the planning process, development phase, acceptance testing and velocity calculation for the iterations worked. Starting the course with such an exercise for students proved helpful in introducing the XP methodology, all in one

go, and also in developing their interest in the subject from the start of the course. It also helped in increasing their understanding of the XP practices when these were covered in detail later in the course. From this exercise the students received not only the overview of the whole XP process but it also gave them a clear idea of roles that exist in XP teams. The roles were introduced intentionally at this initial stage as the students were required to choose their roles they were supposed to play for the rest of the course, a requirement of the planned structure of the course.

Planning Game

As learned from experience ([45],[98]), novice XP teams are feared to get lost in finding an appropriate and fitting planning process for the project as well as for the team. The guidelines and general outline of a planning process provided in classic literature of XP ([8],[10],[9]) need adjustment according to the nature of specific teams and specific project. Otherwise a lot of time is feared to be spent on discussions among the team members about the process and on decision of what, when and how to do things [45]. For this reason, in the context of this course, an in-depth training of the planning process was organized. The planning game exercise was carefully designed keeping in mind the students' existing knowledge of the XP and logic of planning. The exercise emphasized on the following points:

- The logic behind planning. Why we do it?
- Clear definition and separation of XP roles (for example, customers, developers, managers, etc.).
- User stories writing, and estimation.
- Concept of prioritization of user stories and their scope.
- The overall planning process.

To make things easier for the students, a full day planning workshop was conducted in which the students were given a programming project to plan. Students were divided into teams and different roles were assigned to team members so that they could understand the concept and duties of each role by actually performing it. Story

writing style and concept was introduced with the help of a general template for story writing provided by the course organizers. It provided them a quick understanding of writing unambiguous stories which is necessary to promote the concept that stories should be written in a way so that the developers can clearly understand what the customer really wants. The planning workshop was controlled by scheduling the planning activities into a proper time table which further clarified the way the whole planning process was to be performed. Through this exercise the students were also able to understand each role as they were guided which role was to do what at any particular stage in the process.

Usability Testing Exercise

An attempt was made to introduce the concept of usability testing of applications. The idea was to show the students how usability testing process could be integrated into the XP life cycle [49]. This exercise could be emphasized only at a very basic level as most of the students did not have any prior knowledge about usability testing. Only some of the master level students had some prior theoretical knowledge of usability testing and its concepts. Due to time constraints it was not possible to introduce the concepts of usability testing to an extent that an exhaustive usability testing exercise could be done. So it was decided to do a very shallow introduction of the technique and details were left to the students' choice if they wanted to use the concept in detail in the second part of the course.

TDD Exercise

Test driven development is another one of the main concepts of XP. A TDD workshop was arranged to teach basics of test driven development and the concept behind test-first programming. The idea was to let the students understand the technique and to organize their programming skills instead of doing only cow-boy style programming. Again, the same concept of team designing and conceptual exercise was employed using a programming problem but the main emphasis was put on testing only, instead of on the whole XP process.

5.5.2 Part-II : XP Realization Phase

This phase was about implementing all visualized XP practices during application development project as the second major part of the XP course. The idea was to let the students understand the software development process in a broader spectrum while applying XP practices. Efforts were made to create a close-to-real situation by assigning development and management side roles which exist in XP. The students were asked to participate in teams and work only according to the requirements of their roles in the development process.

To further introduce the real world business organizations concepts and requirements the roles hierarchy was also introduced in teams but only at level of communication. All roles in all teams were given same importance as is the spirit of XP.

As a preparation of the second part of the course the following concepts were pre-decided and explained to the course participants.

Team Composition

As already explained in section 5.5.1, to emphasize that all work should be done in XP-teams the concept of roles was introduced in the beginning of the course. It was explained that during the first part of the course the teams were to be created randomly for each XP exercise. But from the start of the XP-realization phase permanent teams were needed to be made for the rest of the course. The teams were decided and created by the course instructors. Each team was to have a manager, a customer, a coach, and a number of developers depending upon the size of the team. The coach was also considered as part of development team. The teaming process was completed during the first part of the course and all teams' members were assigned the roles they were going to play in XP-realization phase. But as defined earlier these teams started working only in the second part of the course. The idea was to let the students to feel the emotional and social pressure of joining a new team and a new project, which is very common in software development industry and always happens when a new project is started.

Roles and Role Selection

It was explicitly mentioned in the introductory lecture of the course that the students would be assigned different roles mostly according to their own choice. The roles were defined according to the actual roles that exist in real software development teams. The following roles were defined:

- Manager
- Customer
- Coach
- Developer

To make roles assignment transparent and to assign appropriate roles fitting according to a student's personality, character and personal interests the students were asked to fill in a questionnaire which posed different questions concerning their general aptitude, social and communication habits, programming experience and knowledge, and also about the students analytical abilities. The performa provided a self evaluation and grading criteria on the basis of which the students were able to suggest which role (or roles) best suited them and which they wanted to play in the teams. The final role assignments were done by the course instructors keeping in view the students' choice, on the basis of the analysis of their response to self evaluation questionnaire, as well as also on the availability of a role within teams, as for example no team could have two managers.

A snapshot of the performa is given in table 5.5.

Team Distribution

The course was attended by almost 100 students. Ten teams were made and each team was assigned roles as defined in section 5.5.2. Extra care was taken to distribute the female students among the teams as they were only the 15% of the total course participants. They were randomly distributed among different teams to eliminate

Table 5.5: Roles Self Assessment

Roles Self Assessment	
Grade the following as 1(highest) - 5(lowest)	
Role	Description
Team Leader	You meet 80% or more of all important and urgent decisions in less than 3 minutes. Only for 15% of all important and urgent decisions you need more information and more time.
Analyst	You can analyze technical problems very well and can compile one nearly perfect solution for it. You possess exact specialized knowledge and you know the necessary tools in some areas, which are relevant for your work and/or our exercise (Java, IDEs, programming).
Designer	You are good in brainstorming and in generating creative ideas and solutions. You are often optimist.
Team member	A good atmosphere in the team is important for you. Good relations among coworkers in your team is very important for you, and you help to make it possible. If one of your team colleagues has technical or problems within personal ranges, you recognize this very fast and talk with the concerned person in order to find a good solution. You engage yourself for the common work.

the factor of male and female programmers. Each team had at-least one female member.

Table 5.6: Project Timeline

Release 1. Iteration 1.
Release 1. Iteration 2.
Release 2. Iteration 1.
Preparation of project presentation/tradeshow
Final exam and project presentation/tradeshow

Project

All teams were given the same project, with some preliminary design requirements but decision about the major functionality was left up to the customers of the teams. They were supposed to decide what extra features they wanted their application to have. The table 5.6 shows the project timeline.

5.6 Knowledge Management Perspectives

Along with all the general issues of knowledge management (KM) that have been discussed in sections 5.2 and 5.3 there were some more pressing reasons for a profound knowledge management in organizing and managing this course. These reasons are explained in the following lines:

Diverse Technical Base The students registered in this course have different knowledge of software engineering and related subjects as they come from many different academic faculties. Due to this reason there is also a difference in the courses that they have completed so far.

Diverse Technical Background The students registering for the course are in different semesters of their study programs, that is why they also have different technical backgrounds. For example, some undergraduate students are in second semester of their study program while others are in registered in fourth semester. Also, many students already have work experience of software development industry, some of them even have several years of experience. This is very helpful in KM when students work in teams, as great care is taken to distribute experienced students evenly over the teams.

Different level of Education The course is offered to both graduate and undergraduate students. The participants are of different age groups, with difference in their technical knowledge base, as well as in practical experience with software development.

According to the format of the course, and for better and diverse knowledge experience on behalf of the students, there is a need to

distribute students among teams with these diverse technical skills and backgrounds. It is also taken care of that female students are equally distributed although they constitute on average around 15 percent of the whole class.

5.6.1 XP in Knowledge Management and Educational Context

This section briefly defines some of the team practices of the XP that have been applied, to date, in the context of this course, their KM issues and the educational context in which these are applied.

Pair Programming

Pair programming defines an interactive coding session among two developers sitting together in front of the same computer and sharing a keyboard and a mouse. Both developers work on the same piece of code at the same time. They act as a driver and navigator. The driver is the developer who controls the keyboard and the mouse at a time and the other partner acting as a navigator guiding through design and test process. According to the format of the practice the driver and navigator swap their role very frequently thus both get opportunity to code [9].

Knowledge Management Factor

This practice involves very informal and spontaneous communication but generates a lot of knowledge as both developers, pairing together, share their technical knowledge and experience regarding design, coding and testing [15]. This practice is characterized as providing a great learning experience as it gives an opportunity to the pair partners to learn from the experience of each other and at the same time apply their own knowledge in the domain. To make sure that the knowledge generated and shared by the pairs is also distributed to the whole team the pair partners are switched very frequently, and this also helps the whole team to come together [15] [9]. This practice embodies a social process as it allows not only to share technical and situational knowledge but also lets the developers to know better about each others' strengths and weaknesses.

Educational Context

In the context of said course the students are asked to work on different coding and non-coding assignments. In the first part of

the course the students have done many non-coding process-related assignments. It has been a good experience to ask the students to work in pairs on writing assignments, and on tasks like designing paper mock-ups. This has been done in order to teach them designing and coding related tasks as well as the pair-working practice before they actually start pair programming. This practice is of great importance in the context of this course as it is defined that there is a mix of students with different skill levels and educational background. This practice greatly helps in providing almost the same knowledge level among the whole class. Also, the pair learning concept is introduced using this practice.

Retrospectives

In XP, project retrospectives allow continuous learning by having a flash-back on what has been done in a previous development cycle [9]. These are postmortem reviews, taken by the whole team together, which allow to learn from what is done and then refine the process if required. These retrospectives are performed on a regular basis during a software development process and hence allow to refine the knowledge and apply it in the same project.

Knowledge Management Factor

The practice of retrospective makes it possible to identify obstacles that have hindered the process or the development one way or the other. The team also discusses the success factors in order to make sure that that the whole team understands how things work better. The team may also raise points of concern regarding any technical or management related issues. In this way it provides a platform for communicating one's own voice to all concerned people [15].

Educational Context

In the first part of the course, this practice is used to get feedback from students about their learning experience. In every lecture the students are given exercises about XP practices. As an after-class assignment the students are asked to write a brief review report on what they did in the exercise, how well they managed it, and also about their experience - did they like it the way it was conducted or did they have problems. These reviews done by students are allowing them to speak their thoughts and convey their problems to their course organizer in a very informal but organized way. This

also reveals how well they are understanding the concepts of XP. Also, as these reviews are published on the course wiki site and are visible to all course participants, this has resulted in easy sharing and distributing knowledge among the whole class.

Planning Game

Extreme programming process divides the whole development process in releases and iterations. Each release consists of more than one iteration. Before each release and iteration the planning meetings are conducted to discuss and settle system requirements as well as business requirements. These planning meetings are attended by the whole team including all developers, managers and customers [9].

Knowledge Management Factor

The planning is always done with the whole team. This makes the whole process visible to everyone. Each team member knows what is going to be done the next development cycle (release or iteration). Any change in the previous plan also becomes visible to everyone. This transparent process removes any danger of miscommunication among developers, customers and managers.

Educational Context

The students attending this course are greatly encouraged to make use of this practice. They were encouraged to do a small planning exercise while they were starting to prepare for their exam. According to feedback given by the students it was a new experience for them to properly plan how they were going to prepare for the exam and it was very time efficient, it helped them to be more organized and to do more in less time.

On-Site Customer

Extreme programming requires a customer with the development team to allow direct communication among the customer and the developers when needed. This helps the customer, who is investing lots of his money into the project, to make sure that the development is done according to the specified requirements [9].

Knowledge Management Factor

Direct communication among the customer and the development team is very important. In conventional software engineering pro-

cess the flow of information from customer to developers, and vice versa, is through the management. This results in late delivery of information to the customer and developer parties and even, sometimes, in loss of important information. The inclusion of a customer in the XP team removes these problems and provides quick customer feedback to the developers which makes their task easier. On the other hand, being with the development team all the time makes it convenient for the customer to introduce new requirements at the earliest.

Educational Context

The students are divided into teams during first part of the course and are assigned customer, manager, coach, and developer roles. The students taking the role of customer get a good training in acting as the customer of the product that his/her team develops. Although the real customer tasks come in the second part of the course but they also get to play their role in the first part of the course in a couple of non-programming exercises, and even in exam preparation. They learn how to formulate their requirements in a form that developers can understand, and to specify the acceptance criteria to understand the success or failure of a task at the end.

Story Cards

In XP, customers write the requirements of the system to be developed in the form of a narrative which is called a story. These stories are usually written on paper cards and are displayed in the room where the development team works.

Knowledge Management Factor

The story cards are usually displayed on a wall or at a prominent place in the XP work room. This makes the development plan visible and transparent to every one in the team. All developers know what their colleagues are working at, and for the managers and the customers it provides an efficient way of looking at the status of the project.

Educational Context

In many different exercises during the course the students playing the customer role in each team write stories for the developers specifying the tasks to be done. In this way they learn how to specify their requirements in a form that the other members of their team are able to understand. The customers are encouraged to discuss

the stories first with their managers and then they ask one of the developers to help them in formulating the stories. The story cards help the developers to organize their work and to visualize at what stage they are working.

Stand-Up Meetings

A stand-up meeting is a short and efficient means of communication among the whole XP team. Theoretically, a stand-up meeting is called every morning before starting to work. It is also attended by all team members. Every one tells very briefly what he/she did the day before and what they expect to do that day. It is called a stand-up meeting as it is kept very short by forcing everyone to stand, not involving any detail question and answering. If there are any points of discussion then these are discussed outside the scope of stand-up meetings [9].

Knowledge Management Factor

As defined above, it is a very quick means of sharing important knowledge with the whole team. Everyone gets a very quick review of the progress since the last stand-up meeting and also about who is doing what during the rest of the day. It provides enough knowledge to everyone to organize themselves.

Educational Context

It has become customary in the course to start every lecture with a stand-up meeting. This practice was initially adopted to give the students a practical experience about stand-up meetings in an XP way. But now that this has been practiced many times it is accepted as a good way of sharing some very important knowledge at the beginning of the lecture. In the start of the course the organizer encouraged every one to speak although only some students actually participated. The idea was to let the students have a courage to speak in front the whole class. That training worked and more students started to participate in stand-up meetings. Now that student groups are created and roles are defined, the teams managers are asked to represent their teams in stand-up meetings. In XP-realization phase this practice is performed individually within teams.

5.6.2 Examination as a KM Tool

The students are required to take examination at the end of each part of the course. The examination preparation is also introduced as an XP process where the customer is asked to make planning of what to prepare for the examination. The customer writes the preparation tasks in the form of stories. The whole team works with the customer in defining the stories for the exam preparation. The stories are worked on in pairs, hence performing pair learning. To further enhance the learning process the examination is also taken in pairs. Two students sit together, discuss the questions and write answers which are according to the combined knowledge of the pair. This way of examination spreads knowledge and let the students learn from each other.

5.6.3 Wiki as a Source of Knowledge Management in Classroom XP

Since the start of the course, a wiki site has been set up to facilitate collaboration among all course participants. Wikis are known to be helpful in education systems and in promoting learning [4]. The course's initial format was based on individual students, which was then defined for pairs and then for teams. The students have been given different assignments which they have worked on individually, in pairs and in teams. To further promote learning and sharing knowledge which they gather in lectures and in exercises the students upload their assignments on the wiki. This has allowed to make work assignments, and reviews of class exercises done by the students, visible to every one and also it is helping the course organizer to review and understand how the course is working out. The course wiki is providing the following modes of knowledge management:

Support for Personal Portals

Each student has been encouraged to use his/her personal wiki page to put every kind of information which could be useful for others as well as for the course. In the very first class all participants of the course are advised to put their contact information on the wiki-page. They are also encouraged to mention their knowledge and skill of programming tools and methodologies, their technical likes

and expertise and also what they intend to gain from the course. The students are also advised to personalize their wiki pages so that the class as well as the course organizer is able to have a better and fast acquaintance with the students. The XP exercises review assignments which they are asked to put on their wiki pages also serve as a good way of knowledge sharing and this has helped the course organizer to understand how students are following the course and what aspects of the course need to be refined.

Team Portals

As the students start working in teams, each team is given a team portal which is used by all team members to put information about their team organization, their meetings, team assignments and all other team related activities. Each team portal shows the team members, their roles, and their expertise and skills. This externalization of knowledge on team level portrays a specific KM model which is required from the general educational aspect as well as from the point of view of teaching XP.

The Scrapbook: Role-based Experience and Feedback

Each team's wiki portal manages a scrap book for all roles, for example, developers, customer, etc., in the team. Each role enters its review of the practice and exercise at the end of each work day. Thus, this scrap book works as an on-line retrospective of the course.

5.7 Implications of Project Management and Industrial Setup

5.7.1 Work Environment

As mentioned before, attempts were made to simulate an industry-like situation as closely as possible. Shared work place and work environment play a very important role in XP ([8] [10] as XP methodology asks for a team work and that the whole team sits together all the time. For this purpose all teams were asked to arrange a team room for them, ideally a place where they could sit together and work for the rest of the course. The minimum requirements for

the rooms were to have place for team meetings and pair programming sessions. Different options were also provided for arranging the work place to facilitate the information sharing and communication among the whole team [98]. Consequently, each student team arranged a routine work place along with facilities to display story cards, place for pair programming, planning and reflection meetings.

5.7.2 Daily Routine

To start with a proper training in project management and to make it an organized experience for the students a formal daily routine was defined for this phase. In the beginning it was followed strictly and then the teams were asked to take over and organize themselves according to their convenience, hence giving a clear idea of how it should be done, and, at the same time, saving the spirit of agility. A general sketch of the "minimum" daily routine is given in table 5.7 which was followed by all student teams during the whole XP-Realization phase.

Table 5.7: General daily routine

A short overview of the day's activity by the instructor
Teams meeting in their XP rooms
Stand-up meeting
Planning (only on planning days)
Manager and Customer : Administrative tasks
Development team: Coding sessions, pairing
End of day: Progress presentation by managers to the instructor
End of day: Refactoring of the code by the development team

5.7.3 40-Hours Week

It was also decided to work according to "40-hours week" theme of XP methodology. To do so the course was organized as a full day activity once a week for the whole semester. This provided an opportunity to the students to have consecutive eight hours for the course once a week during which they could work with their team members and work with core practices of XP (stand-up meeting, planning game, pair programming, refactoring and retrospective

meetings) on daily basis. This, otherwise, would not have been possible if the course time was distributed on multiple days in time slots of two or three hours during the week.

5.7.4 Communication among Course Organizers and Teams

It was decided to make the communication among the course organizers and student teams in a way to simulate the communication style, flow, and hierarchy of business organizations. It was done in a kind of official and professional manner to give the students a feeling of working in a software development industry. Also it provided an opportunity to teach communication skills which are an integral part of not only teaching but also of an important aspect of XP [55]. All course related instructions were sent to the managers of the student teams by email from the course organizer. The teams' managers were then responsible to organize their teams according to the given instructions. This exercise was greatly liked by the teams and it especially helped the managers to play their role more properly in the teams.

5.7.5 Changing Teams during Project

Another concept of real software development organizations is that in real life people (employees) are not made to work with the same team or even on the same project from start to end of the projects. The team members are changed, their roles are changed within the possible limit, and even people are hired and fired. To simulate this concept on a small scale the teams of the best programmers of each team were changed after first release of the project. Each team selected its own best programmer through voting without telling them what would be done with this criterion. The best programmers were then given a "promotion" by sending them to some other team. Although in the beginning the students did not like the idea of being forced to leave their team but then gradually they understood the idea behind this, and due to the practices like pair programming, stand up meetings, and collective code ownership which were followed in all teams the new team members quickly adjusted in their respective teams.

5.7.6 Customer-Manager Workshop

To further motivate the students about the idea of an XP customer a member of a commercial XP team was invited who was acting as a customer in his team since three years. The customer gave a talk to all student managers and customers regarding his personal experiences of performing the duties of a customer. It proved to be a good experience as the students were able to ask questions which they had in mind regarding their own experience with the roles they had played during the course. It was also suggested by the students that this kind of workshop would be of much more benefit if conducted at different stages during the course, especially during planning exercises.

5.7.7 Project Presentation/Trade show

The course concluded with a presentation of the projects developed by all teams. Managers of all teams prepared a thorough presentation of their projects and presented it to all course participants. This activity was called Trade-show and it was the only time when all teams came to know about the way the projects were developed by the other teams.

5.7.8 Planned and Surprise Meetings/Visits

To add another dimension of industry-like setting the teams received planned and surprise visits from the course instructor during their routine work. The objective was to keep a check on the working of the teams and also to evaluate their work for the grading of the course. Likewise, the course instructor arranged occasional meetings with the managers of the teams to inquire about their teams' activity and status of the work, any problems the teams were facing or to discuss any issues related to project or XP process prevailing with specific teams.

5.8 Summary and Conclusion

We can gather all this experience of XP learning and teaching and accumulate it in form of guidelines for organizing, teaching, and managing a course of practical XP for graduate students. This set of

guidelines constitute a flexible framework for teaching such a course of XP in a university environment and in the author's opinion fulfills the minimum requirements of course organization and management in general, and especially to teach XP.

5.8.1 Guidelines for Course Organization and Management

In the light of what the author perceived from her understanding and learning of XP during m3 project (refer to section 4.6) and the goals defined for this case study (refer to section 5.1.1) the author defines the following guidelines as a framework of structure and organization of XP teaching embedding KM in different forms.

- Organize the course in large time blocks, for example, four hours a time twice a week or eight hours once a week (as is done by the author). This gives an opportunity to incorporate XP practices already in organizing the course. Also on the other hand it makes the management of the course easier.
- Depending upon the total number of students make teams of about six to eight students (in the case defined above it was nine on average which was just ideal).
- Distribute the students into teams as fairly as possible. A class of computer science students is usually a mix of excellent, good and average programmers. Distribute them equally in all teams so that the knowledge can be distributed evenly among all students.
- Make an even distribution of male and female students. Female students are usually good in general management of teamwork.
- Instead of directly starting with fixed teams let the students know each other in the whole class by making random teams in the beginning. Working in fixed teams from the very beginning restricts the knowledge distribution and is also against the general concept of agility.
- Introduce roles in the teams. Roles of XP are ideal for teaching programming as well as management. The team size defined above also makes it easy to have a manager, a customer, a coach and a number of developers in each team.

- It is better to take students' consent in assigning roles. You can always impose a role onto a team member if a given role is not available in a team any more.
- Employ some general technique for personality evaluation before assigning roles. This way it will be possible to assign roles according to personality of students, and also the students will be able to give their consent about a role more confidently as they themselves will understand what they are good at according to the traits of their personality. But try using some simple way for personality evaluation and do not make it more technical and time consuming than required.
- Role assignment should be done before making teams. This way it is more flexible and gives a better opportunity for even distribution of expertise among teams.
- Clearly defining and specifying theory and practical part of course will be very helpful for the students to visualize their goals at a particular stage in course. Also managing theory and practice in two separate parts is more easy and simple.
- Instead of using lecture based approach use different teaching aids to make the theory part more interesting for the students. This also provides them with a good learning experience. Arranging lectures in the form of short and to-the-point workshop-style exercises makes it very easy to teach the logic and actual working of XP practices. As there is always a time constraint with university courses therefore it should be decided before hand how much time can be alloted to each practice (as in the case described above not all practices were taught in separate exercises and workshops. It is also possible to define exercises which demonstrate more than one practice).
- Practices like planning game, story writing and estimation, and customer focus can be demonstrated using logical and simulation exercises. Practices requiring technical knowledge, for example pair programming and test driven development should be taught using programming exercises.
- Follow a set schedule for almost everything including theory and practice. Although it is against the spirit of agility but too

much agility in the class room will result in chaotic situation, especially during workshops, and the goals of the workshops will not be met.

- Try to specify activities for each role. For example in planning game explicitly define the role and activity of managers, customers and developers. No member of a team should be sitting idle most of the time. It will reduce the interest of the students in certain roles. For some practices it is not possible to involve all roles, for example, in acceptance testing exercise most of the work is done by the customers and developers do not have enough work to do.
- From management point of view giving the same project to all teams is more convenient.
- Guide and facilitate the students to arrange their team rooms or work places.
- Define fixed time slots for iterations and releases.
- Simulating the corporate industry style communication link among the course organizer, managers and customers will also provide a good learning and training experience for managers and customers.
- Define protocols for the duties of managers, customers, coaches, and developers during project.
- Arrange workshops for managers and customers, ideally inviting professionals from the software development industry. This will provide them a good guidance about what they are doing.
- Define a schedule for daily routine during project and make sure that all teams follow it. Specify the schedule as minimum requirement and leave room for additional activities so that the teams can also learn to organize themselves.
- Supervise the work of each team during project time and make sure the practices are being properly followed.
- A protocol defining what to look for during visits to the teams will be helpful in comparing activities of different teams.

- Control the work of each team by asking them to report on the work done by the team at the end of the day. It can be done by having arranged meetings as well as surprise visits to teams.
- Introduce real life situation by changing goals of the application being developed as the XP project.
- Award teams with appreciation certificates that perform better than others during the course and after the final project.
- Provide a collaboration platform for all students and teams so that they can communicate as well as share knowledge easily and efficiently. For example, setting up the course wiki and allowing the students to manage personal as well as team portals is a good way to supervise teams' activities as well as knowledge management, also provides an opportunity to the course instructor to understand students view about the different aspects of the course.
- In case of some social or communication related problem among team members ask the manager to work as moderator and resolve small issues occurring in team work.
- It is also ideal to have student assistants for supervising and guiding the teams during project. Define a protocol of duties for the student assistants.

Chapter 6

Data Analysis and Results

This chapter presents the details of the analysis and discusses the results based on the study conducted in the context of this research. The chapter presents the details of the surveys conducted and the notable results gathered from the analysis of the surveys' data are discussed in detail.

6.1 Introduction

One of the main objectives of this course was to use it as a research base for analyzing XP in the educational context, the acceptance and adherence to XP practices by the students and also to find out the students' opinion about the way the course was organized. To do so it was decided to collect data from more than one source using two different questionnaire-based surveys. Observational data was also collected from course wiki site. That data was not formally analyzed but was used as a continuous feedback from the students and was continuously incorporated by the author into the course organization and structure. Two surveys were designed, as part of the course, conducted during the course at predefined stages. The students were informed about this research base in the start of the course. They were given a brief introduction of the surveys to be conducted, and also the motivation behind conducting these surveys. For this purpose participation in the surveys was made necessary for all students. It was made clear that for the validity of the results it was necessary that all students take serious participation in the surveys. The surveys were designed as short and concise questionnaires

and were to be done via an on line survey portal.

The following subsections give a brief introduction of the surveys which have been used to collect data for the empirical and descriptive analysis of the course.

6.2 XP Practices Survey

A questionnaire regarding XP practices was sent out to the students three times during the whole semester. The timing of the survey was selected carefully to analyze the learning process and the acceptance of the XP practices by the students. The survey was conducted at the following stages:

- In the very first lecture of the course when students had no previous university training of any agile methodology,
- After the end of XP-visualization phase,
- At the end of XP-realization phase.

6.2.1 Methodology

The survey was conducted through an on line web-based survey portal to eliminate the chances of manual and human errors in data collection [79]. The survey was designed as a descriptive survey as the goal was to collect students' responses about their perspective and experience with XP in general and specifically about all practices. The survey mostly provided closed-ended questions as multiple-choice and point-scaled. Most of the questions were designed using 5-point Likert scale [67]. Open-ended questions were also provided to collect students response about benefits and drawbacks of the practices. The survey was meant to provide a descriptive analysis by defining frequencies and cross-tabulation method as the goal was to understand the opinion of students about practices of XP during different phases in the course [82]. The qualitative research methods were also used in some cases to answer the "why" and "how" questions regarding some aspects analyzed.

The questionnaire posed 14 main questions (one for each practice) and each question was divided into five sub-subquestions. The questions were mainly adopted from Shodan 2.0 survey [63]. The

questions' structure was kept similar for all practices so that the students could concentrate more on the question and not on the structure. It also made the design of the questionnaire more simple. Apart from these 14 questions there were three more questions regarding general information about the students.

The analysis of this survey shows how the different practices of XP were learned and understood by the students during the course of the training. The author has followed the analysis and reporting method used by Salo and Abrahamsson in [92] because of the similarity of goals with the analysis of the survey. Other surveys and related studies which have been considered include [17],[100],[85],[88], and [89]. All these surveys have explored and analyzed different aspects including usability, applicability, etc., of agile and especially XP methodology. The questionnaire is provided in appendix A.

6.3 XP Education Perspective Survey

This survey was conducted only once, at the end of the course. It posed questions regarding the way the course was organized and about the exercises that were carried out to teach different XP practices. The main objective in conducting this survey was to analysis the students' acceptance of the course organization and teaching framework nd also to get a general evaluation of the course from them. This survey was also organized as a descriptive survey and contained questions regarding general educational background of students and their current and past experience with using XP practices. The survey was designed a a combination of open-ended and closed-ended multiple choice questions. The descriptive analysis was done following the same methodology which was used for XP practices survey defined in section 6.2. The survey questionnaire is provided in appendix **??**.

6.4 Analysis and Results

This section provides results of the analysis of the surveys mentioned in the previous sections in this chapter. The results are self explanatory and can be followed very easily. The results are analyzed from the point of view of teaching and learning aspects of XP.

The data is gathered during the course the main objective of which was to introduce XP methodology in the frame work of knowledge management, semi-industrial setup and of role playing. The results emphasize the students' perspective about using XP as a software development methodology during and after the course.

6.4.1 Analysis of XP Practices Survey

This questionnaire was filled in by 87 students (with 80% male 20% female students) and at three stages during the course, that is, before the start of the training in the very first lecture, after XP-visualization phase was finished, and after XP-realization was finished at the end of the course. The results were presented in percentage of students. In the first survey only 8% of students said they had already done XP to some extent and 37% of the students said they knew nothing about XP. The rest of the students had either read about it or only heard about the methodology. Therefore, the responses of the all the students in first survey were based upon only on their prior knowledge about the methodology. The second questionnaire was filled in by the students after XP-visualization phase was finished in which XP practices were taught using lectures, classroom workshops, and small exercises. Up to that point they had not used the methodology in a full software project. Third time the questionnaire was filled in by the students when they had been working with XP since four months and they had classroom training of XP-visualization phase and of practicing XP in the project during XP-realization phase. The questions regarding XP practices measured the aspects like their "knowledge about a practice, experience of working with the methodology, easy to learn, easy to apply, helpfulness, enjoyable, already widely used, and easy to introduce in the team". Different 5-point scales were used, for example [-2(not at all), -1, 0, 1, +2(absolutly positive)] and [Nothing, Heard about,Read about, Done it, Everything]. The data from the three questionnaires was analyzed separately to find out the opinion of students about XP while in a specific phase in the course, and at the end the results from all the three questionnaires were compared to see the effect on the student learning and their interest and opinion about XP practices after different phases of the course.

Comparison analysis

Following are some of the comparison analysis results based on specific aspects about all XP practices.

- The aspect of "Helpfulness" of XP practices. This comparison shows how helpful were the different practices during different phases in the course.
- The aspect of "Easy to learn" for XP practices was analyzed in comparison as one of the goals of the study was to test the XP course organization framework for its effectiveness in increasing the learning of the methodology.
- The aspect of "Enjoyment" related with learning and practicing the XP practices in general and in relation with the way the course was organized and structured. This was also done to see how much the mode of teaching was able to grasp the interest the students in learning XP methodology. This also showed how well the students were taking the team work and other social aspects related to XP practices.

Discussion of Comparison Analysis Results

- Aspect of Helpfulness Figure 6.1 shows the comparison analysis of XP practices based on the aspect of "Helpfulness" using the data collected three times through the questionnaire of XP practices survey. Blue coloured bars represent the data collected before training started, red coloured bars represent the data collected after the XP-visualization phase was completed, and green coloured bars represent the data collected at the end of the course after XP-realization phase was completed. Y-axis gives the names of all practices for which the data was collected. X-axis shows the percentage of students who agreed to the aspect of helpfulness of particular practices in the software development process. For example, the gradual increase in the graph of planning game from first questionnaire to the third questionnaire shows that the students gradually understood the logic behind using the practice and considered it helpful for the development process. Similar trend exists for the practices of small releases, collective code ownership,

continuous integration, sustainable pace, and for daily stand-up meetings. The graph for the practice of metaphor shows almost 40% decrease in the popularity of the practice for the aspect of helpfulness. It was because most of the students argued that they did not feel the usefulness of the practice in the context of the project at that small scale. Most of the students considered talking about and consciously following the same metaphor for their project as extra overhead. Even though in the begining they thought that it was a useful concept to be used in projects. Some practices, for example, simple design, pair programming,coding standards, and the practice of whole team, lost their importance in the XP-visualization phase but then again the students foundout that they were actually helpful in actual software development.

- Aspect of learning The graph of "Easy to learn" 6.2 aspect related to XP practices shows trends of the similar nature as shown in the aspect of "Helpfulness". More than 60% students voted that planning game was easy to learn after the last questionnaire, as compared to 32% votes in the first questionnaire and only 18% votes in the second questionnaire. This is due to the fact that students initial knowledge of the practice was based on only what they had learned in books. The students general feedback was that too many small details had to be followed during the planning game, which most of the students thought as waste of time in XP-visualization phase as they wanted to start coding right away. But when they worked on the complete programming project in XP-realization phase and actually planned the project development they understood the practice and learned it easily. Same is the case with small releases, simple design, TDD, refactoring, collective code ownership, coding standards, and sustainable pace. It is obvious that this trend existed for all practices realted to the tasks of coding. The students gradually learned and understood the practice of onsite customer and the concept of whole team, starting from the begining to the end of the course after working on the project in the XP-realization phase.

- Aspect of Enjoyment Figure 6.3 shows with which of the practices students enjoyed working during different phases of the

course. Working in Small releases was voted high as it made the programming tasks plan-able and easier. They enjoyed refactoring, collective code ownership, working with sustainable pace, having on-site customer, and daily stand-up meetings also worked a lot for them in the XP-realization phase. Pair programming showed somewhat negative trend, but only with a decrease of 5-10% from first questionnaire to the last questionnaire.

Individual Questionnaire Analysis

Different aspects related to the XP practices were analyzed after XP-visualization and XP-realization phases to see how the training in the two phases worked for the students. Following are some of the results given by the the analysis.

Discussion of Individual Questionnaire Analysis

Planning Game - Easy to Introduce In figure 6.4 it can be seen that about 32% of the students said that planning game was easy to introduce and 11% students were absolutly positive about it. Only 4% of the students showed their reservations about the aspect after XP-reaslization phase.

Planning Game - Easy to Apply Figure 6.5 shows that 38% of students can apply planning game easily and 10% of the students are absolutly positive about this aspect. So the students are more sure about the application of the practice as compared to the fact that they can easily introduce it in a team.

Metaphor - Easy to Introduce During XP-visualization phase the teams said that metaphor for an application was not easy to introduce and use. But after using this concept during the application development most of the students were satisfied with that practice, although there were many who said they did not need it. Figure 6.6 shows 43% positive and 22% neutral response for this aspect after XP-realization phase.

Simple Design - Helpful Helpfulness of the practice of simple design was voted positive by a total of 75% students (with 50% being absolutly positive and 25% positive). Only a total of 5% students gave negative response. Rest of the students remained neutral.

Table 6.1 shows a ranking of all practices on the aspect of helpfulness after XP-realization phase. Small release is ranked first followed

by planning game and pair programming. Metaphor is placed in the last position.

Table 6.1: Ranking Practices on the basis of Helpfulness

XP Practice	**Percentage**	**Rank**
Small Releases	83	1
Planning Game	82	2
Pair Programming	81	3
On-Site Customer	79	4
Simple Design	78	5
Refactoring	77	6
Coding Standards	82	6
Daily Stand-up Meetings	76	7
Whole Team	73	8
Sustainable Pace	72	9
Test Driven Development	71	10
Collective Code Ownership	69	11
Continuous Integration	69	11
Metaphor	52	12

6.4.2 Analysis of XP Education Perspective Survey

The survey gathered data about the past experience of students in the field of computer science. 62.86% of students was studying computer related subjects for 0-2 years. 52% of the students had 0-2 years experience of working in software development industry. Only 16.87% students had 4-6 years of experience of the field of computer science, mostly as a student and in very few cases as a professional. 37.35% students said they wanted to work as a developer in software development teams. 25% students said they would like to take a management post. 50% of the students rated themselves as good to excellent programmers. Only 4.82% students said that they were not programmers at all.

Following are the results concluded from the descriptive and qualitative analysis of the survey regarding XP methodology.

- 61.45% students said that they liked the concept of pair programming, someone actively working along while doing a programming task.

- 48% students rated programming expertise as most important skill for a software developer. 27.71% said working in teams was important for a software developer.
- 55.42% of the students said that they were satisfied with using XP during application development, 16.87% of students said they wanted to use some other methodology. The main reasons specified were short time for the project as a lot of time was spent on learning technical skills and practices like test driven development. Some of the students argued that XP practices were adding overhead. Rest of the students remained neutral.
- 65.65% students said they will advocate XP methodology in their future work. Some students argued that it would be better if selecting some practices from the whole set of XP process was possible.
- 80% of the students said TDD was the harder practice to realize, 12% of the students said that pair programming was also difficult to do properly because of too much difference in technical knowledge of students.
- 53.01% students said they liked team work and 34.94% students said they were selective about their team mates. 7.23% students said they preferred working alone.
- 78.31% students were in the favour of collective code ownership.
- 81.93% students liked the idea of task prioritization and estimation.
- 38.55% of students was in the favour of daily standup meetings, 48.19% said it was a good practice only when there was some information to share. 8.43% students said they would attend a standup meeting if sharing views with others was not mandatory.
- 56.63% students said having an onsite customer was the best thing for a development team. 13.25% students said the customer should be a trained customer to be productive for the team.
- 74.70% students said open workspace raised the productivity of the team. But still only 22.89% ranked it as an important

practice (reference table 6.3). Most of the students considered practtices like pair programming and coding standards as being more important as compared to open workspace.

- 51% of students concluded that working on application development with XP methodology was fun and it was a success factor for a software project.

XP Practices Ranking

Table 6.2: XP practice ranking done by students according to "Dislike" aspect

XP Practice	**Percent**	**Rank**
metaphor	37.35%	1
Test Driven Development	25.30%	2
Short Releases/Iterations	15.66%	3
Acceptance/Functional Testing	14.46%	4
Pair Programming	13.25%	5
Collective Code Ownership	12.05%	6
Planning Game	9.64%	7
Simple Design	9.64%	7
40 Hours Week	9.64%	7
Standup Meeting	7.23%	8
Coding Standards	7.23%	8
Refactoring	7.23%	8
Open Work Space	7.23%	8
Continuous Integration	4.82%	9
On Site Customer	4.82%	9

Table 6.2 shows a ranking of the practices according to the aspect of "dislikeness" measured using XP education perspective survey. For example, although the practice of metaphor was considered as easy to introduce in a team (refer to figure 6.6) but it was ranked as being least helpful (refer to table 6.1) and was also disliked by most of the students.

Table 6.3 ranks the practices according to their importance at the end of the course (measured in XP education perspective questionnaire). Here again the practice of metaphor is ranked as least important. Pair programming is ranked as the most important practice for a development process.

Table 6.3: XP practice ranking done by students according to "Importance" aspect

XP Practice	**Percent**	**Rank**
Pair Programming	72.29%	1
Simple Design	67.47%	2
Test Driven Development	59.04%	3
Standup Meeting	57.83%	4
On Site Customer	55.42%	5
40 Hours Week	55.42%	5
Planning Game	51.81%	6
Collective Code Ownership	51.81%	6
Coding Standards	45.78%	7
Continuous Integration	38.55%	8
Short Releases/Iterations	36.14%	9
Refactoring	32.53%	10
Acceptance/Functional Testing	28.92%	11
Open Work Space	22.89%	12
Metaphor	20.48%	13

6.5 Summary of Results and Discussion

All the data is collected and analyzed using descriptive and qualitative data analysis techniques as the main idea was to collect students' opinion about the course, the XP practices, the way the course was taught, and the effect of the semi-industrial role-based knowledge emphasized training of XP in university environment. The key results concerning XP in general and XP practices are as follows: 62.65% students said they will advocate for XP in future, team work (53.01%), collective code ownership (78.31%), usefulness of story estimation and prioritization (81.93%), standup meetings (38.55%), onsite customer (56.63%), common workspace (74.70%). In XP education perspective survey (refer to table 6.3) practices of pair programming, simple design and TDD were ranked first, second and third with 72.29%, 67.47% and 59.04% positive responses, respectively as being most important practices for the development process. In the same way, 40-hour week was voted by 55.42% students and planning game received 51.81% of positive votes. Metaphor ranked last and was the most disliked practice. Refactoring and acceptance testing also received only 32.53% and 28.92% votes with

respect to their importance felt by students during project. 51% students voted for the fun factor related to working with XP.

Although a descriptive analysis is performed on the results collected from different sources of data but it is not attempted to prove any hypothesis. The data collected under same aspect using different sources showed common as well as contradictory results in some cases. These cases need to be analyzed in detail using more specific research methods. For example, although practice of metaphor was found easy-to-introduce after XP-realization phase but it was ranked as most disliked practice at the end of course in XP education perspectives survey (refer to section 6.3). Also it showed a drop from 98% to 58% while comapred on the basis of "helpfulness" after three xp practices surveys (refer to figure 6.1). Also there is a need to correlate the aspect of importance of XP practices measured in XP education perspective survey with the aspects of helpfulness, enjoyable, easy to learn and easy to introduce. The analysis presented here is not fully exhaustive, the author has presented some of only those results which were calculated keeping in mind the main goals associated with the research conducted.

Being descriptive in nature and also because of human nature the results are supposed to be affected by the personal and subjective opinions, level of knowledge of students, immaturity of concepts on behalf of students, provide only a very limited view on the concepts and aspects which were tried to be measured [92]. All questions in the survey provided "do not know" or neutral (0) options to avoid incorrect input which is not actually meant by students. This concept is also used and accepted and also used by Salo and Abrahamson in [92].

Apart from all these points and facts described above, the results show positive aspects of the XP teaching in general and the improvement in learning and understanding of XP practices from one phase to another phase (refer to figure 6.2, 6.1, and 6.3). This was what mainly intended to see from the analysis.

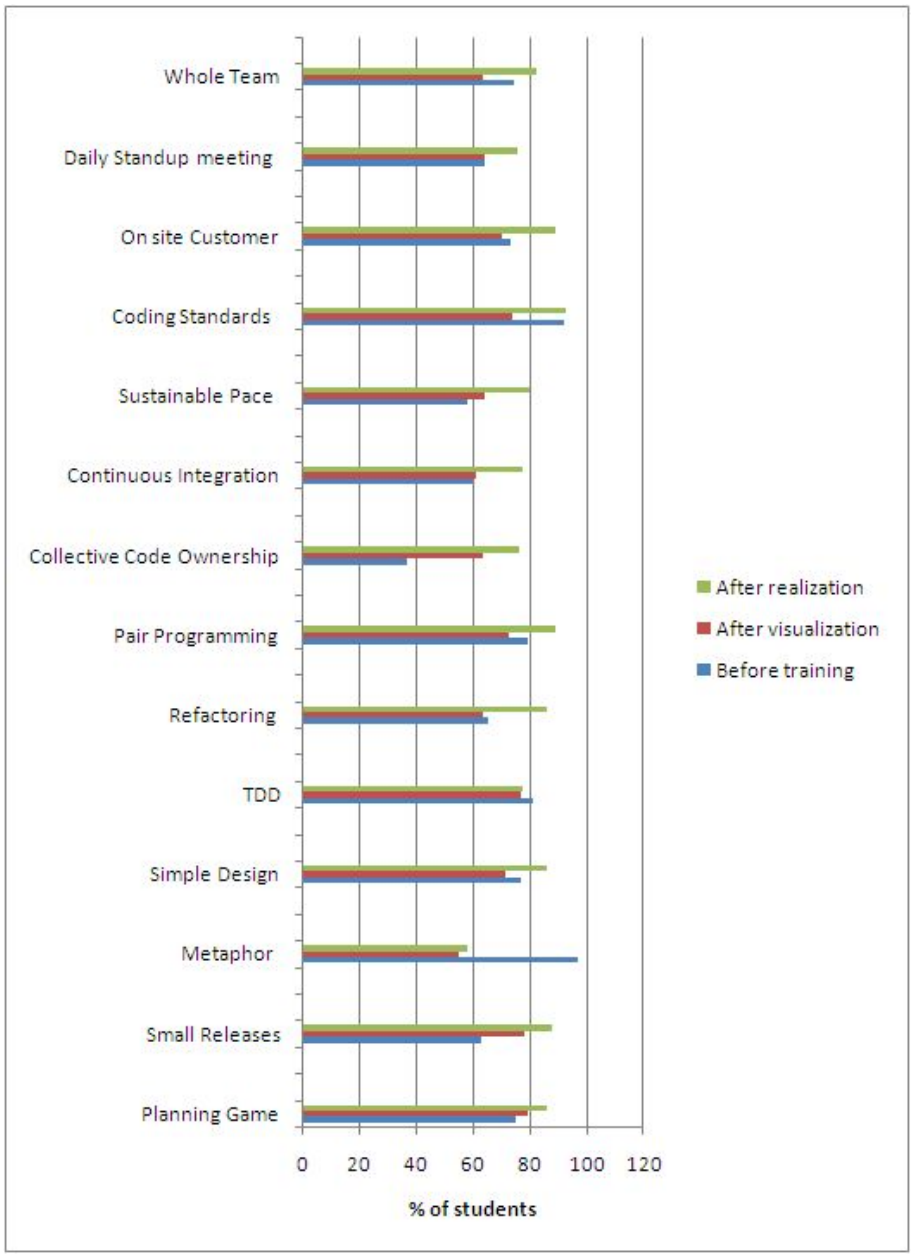


Figure 6.1: Comparing the aspect of "Helpfulness" - increased or decreased from one phase to the other phase of XP training

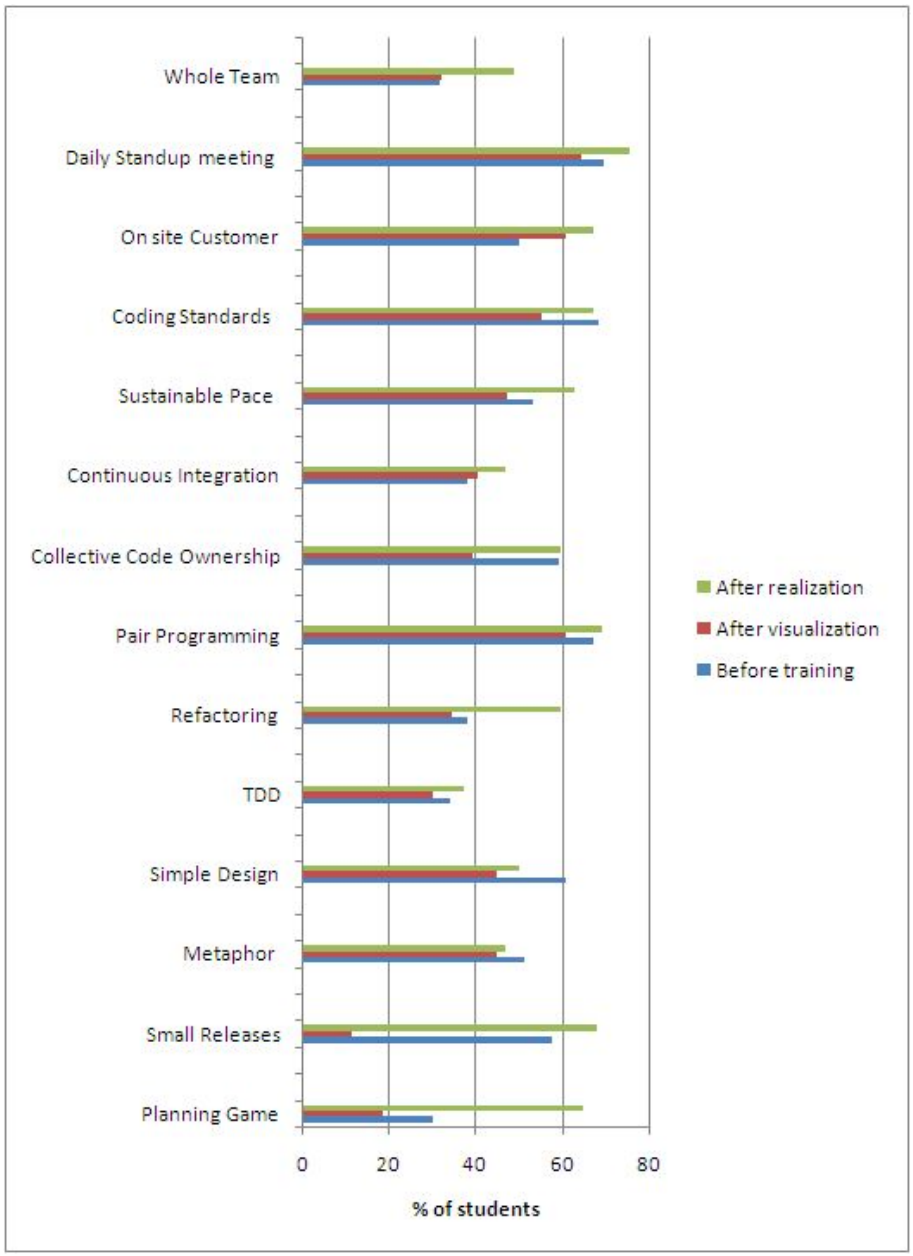


Figure 6.2: Comparing the aspect of "Easy to Learn" - increased or decreased from one phase to the other phase of XP training

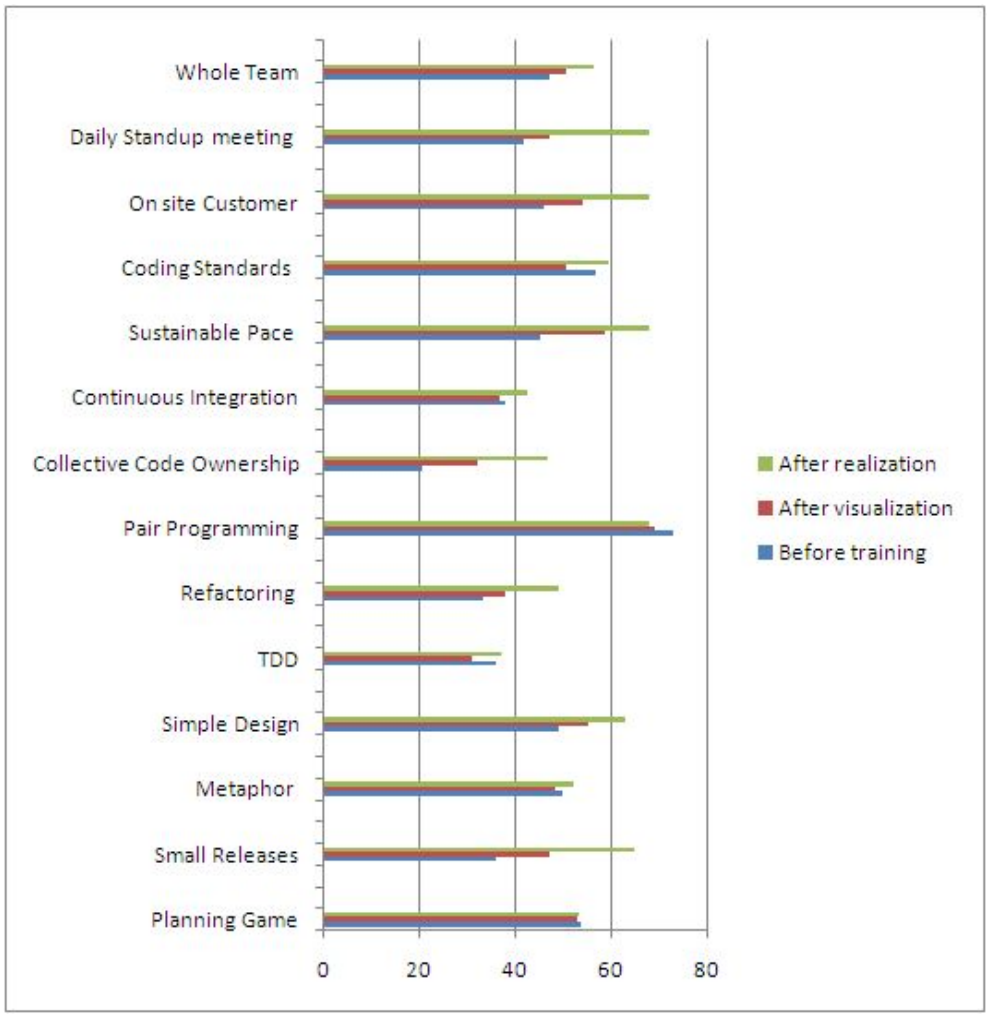


Figure 6.3: Comparing the aspect of "Enjoyment" - increased or decreased from one phase to the other phase of XP training

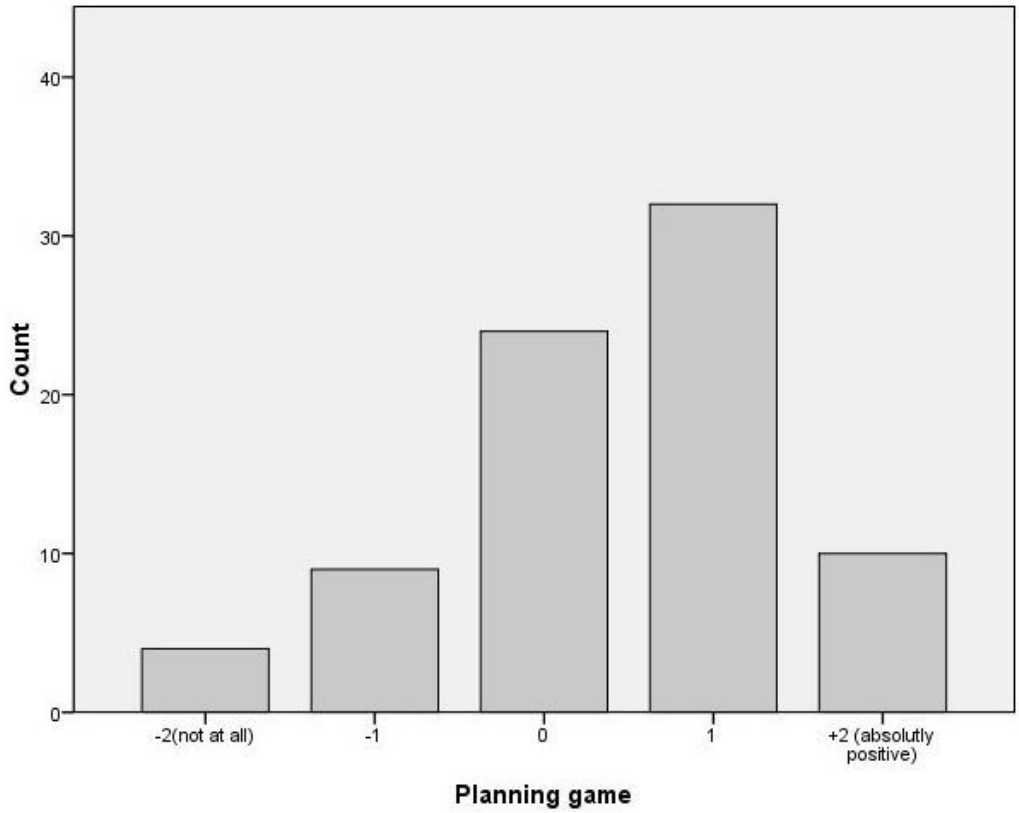


Figure 6.4: Aspect of "Easy to Introduce " for planning game after XP-realization phase

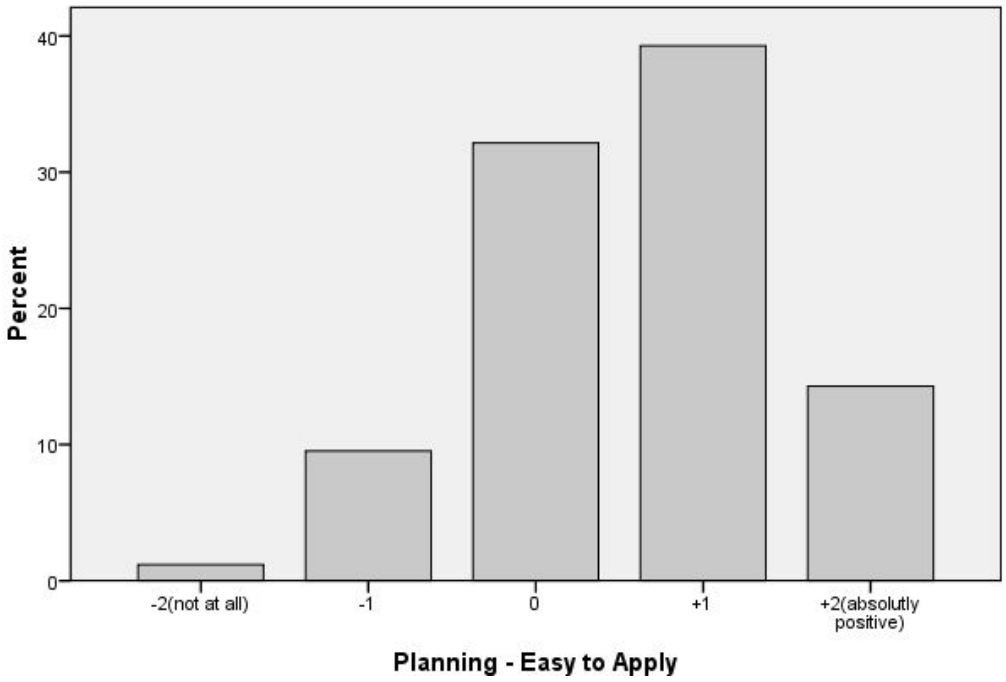


Figure 6.5: Aspect of "Easy to Apply " for planning game after XP-realization phase

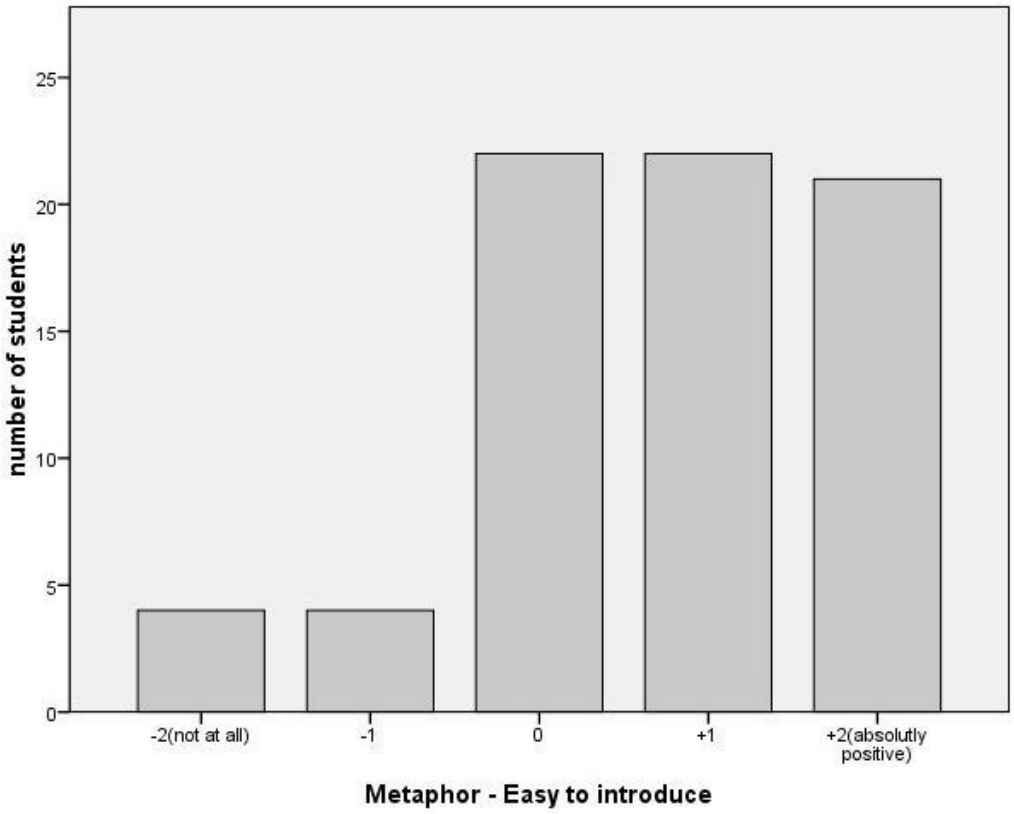


Figure 6.6: Aspect of "Easy to Introduce " for metaphor after XP-realization phase

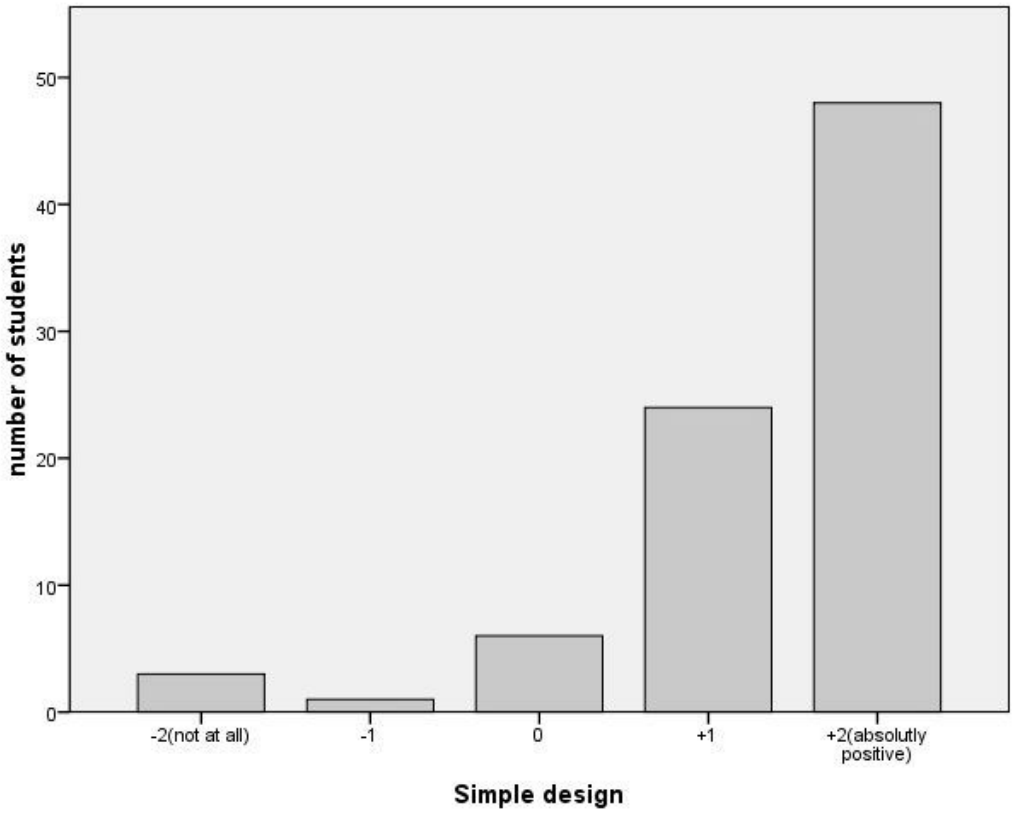


Figure 6.7: Aspect of "Helpful " for simple design practice after XP-realization phase

Chapter 7

Limitations and Future Work

In the following sections the author provides a summary of results, the limitations of the study as well as some perceived future directions which can be taken from the result of this study. A general conclusion is provided at the end of this chapter

7.1 Summary of Data Collection and Results

The software engineering course selected as case study was organized and structured keeping in mind the research goals set by the author. To make it simple and less precarious both for the author herself and for the students the analysis component was embedded into the course structure. The data collection was planned within the course outline and was placed at logical points and stages during the course so that the flow of teaching and learning was not disturbed and students had enough and required extent of knowledge about the XP process to fill in the questionnaire share at that time. It has helped in maintaining the validity and completeness of the data. The other reason for embedding the data collection process into the routine work was that it normalized the "extra work" concept in the minds of the students and they considered questionnaires as part of the course just like other activities performed in the context of the course. The data was collected from multiple sources. Two surveys were conducted with the help of questionnaires designed specifically according to the specifications of the course. One

of the surveys namely "XP practices survey" was conducted using an iterative approach to analyze the XP learning process and its relationship with the kind of training being provided. This survey was conducted, with similar questions and format, three times during the course at different stages of the course. The second survey was conducted only once, that is, at the end of the course as it posed questions regarding the overall XP teaching and learning process, the students' personal view about the XP process, and about the course structure and organization in general.

The analysis of the data collected through XP practices survey presented in chapter 6 provides an insight into different aspects of XP practices. It reflects the XP learning process and its relationship with the kind of training being provided. The data collected after the lecture-cum-workshop based training done in XP-visualization phase reflects the students extent of knowledge after learning the concepts in a controlled semi instruction-based environment (discussed in detail in chapter 5). The data collected after all-programming XP-realization phase reflects the students' concepts about the XP practices after they have learned the XP methodology by instruction and learning (in XP-visualization phase) and from the practical experience (in XP-realization phase). The data was also collected from other sources such as from the course wiki site which collected students' views about daily routine work and exercises.

All this data is analyzed to view the actual output of the study. The results are discussed section 6.5 in chapter 6. The descriptive analysis performed on the data provides students perspective about the aspects of "knowledge about a practice, experience of working with the methodology, easy to learn, easy to apply, helpful, enjoyable, already widely used, and easy to introduce in the team".

7.2 Limitations of the Study

The author acknowledges the methodological limitation of this research. The design and the structure of the framework were adapted from the general guidelines provided by different experts in the field of software engineering education and from the professional in the industry. But it also contains personal bias on behalf of the author as her knowledge about the domain is based on her one time personal

experience of working with the methodology. Although the span of the experience is the continuous work using the XP methodology for three years but still the factor of non-exhaustive exposure of the domain is involved. Similarly the data collected for the analysis is also based on what was the personal idea of the author, although the data collection surveys were designed and adapted from a previously tested study.

The data analysis was also performed using descriptive and qualitative methods as the main research idea was to collect the views of the participants (students) of the course about XP practices in general. No formal hypothesis was defined which needed to be proved. The reason was that the author was aware of the unavailability of multiple case studies to cross compare the results to have an accurate finding about any hypothesis. Although cross-comparison analysis of the coding metrics gathered from the student projects can give a better reflection on the study performed but using only the analysis of the coding metrics was not considered to be adequate for an extensive research study design.

7.3 Future Research

The study provides an opportunity of several directions for future research. It can be replicated to similar case studies involving teaching of XP. As the author is a member of academia therefore this direction is of utmost priority. The author plans to apply the study and the framework on some other university based courses in different universities. The findings of the study conducted are adequate to make several simple and logical improvements in the structure of the framework. The analysis of the XP education perspective survey outlines specifies points of improvement in the exercises and structure of the workshops done in XP-realization phase.

The case replication can also be performed by choosing case studies from industry to use it as a training framework for professional software engineers having none, little or more knowledge about the agile software development paradigm. The content and format of the workshops can be adjusted to include more directions of agility and can be made more advanced according to the level of knowledge of the sample population on which it is to be applied. The analysis structure can be readily applied to such cases with some

modification in the questionnaires with respect to the data collection requirements of the case study.

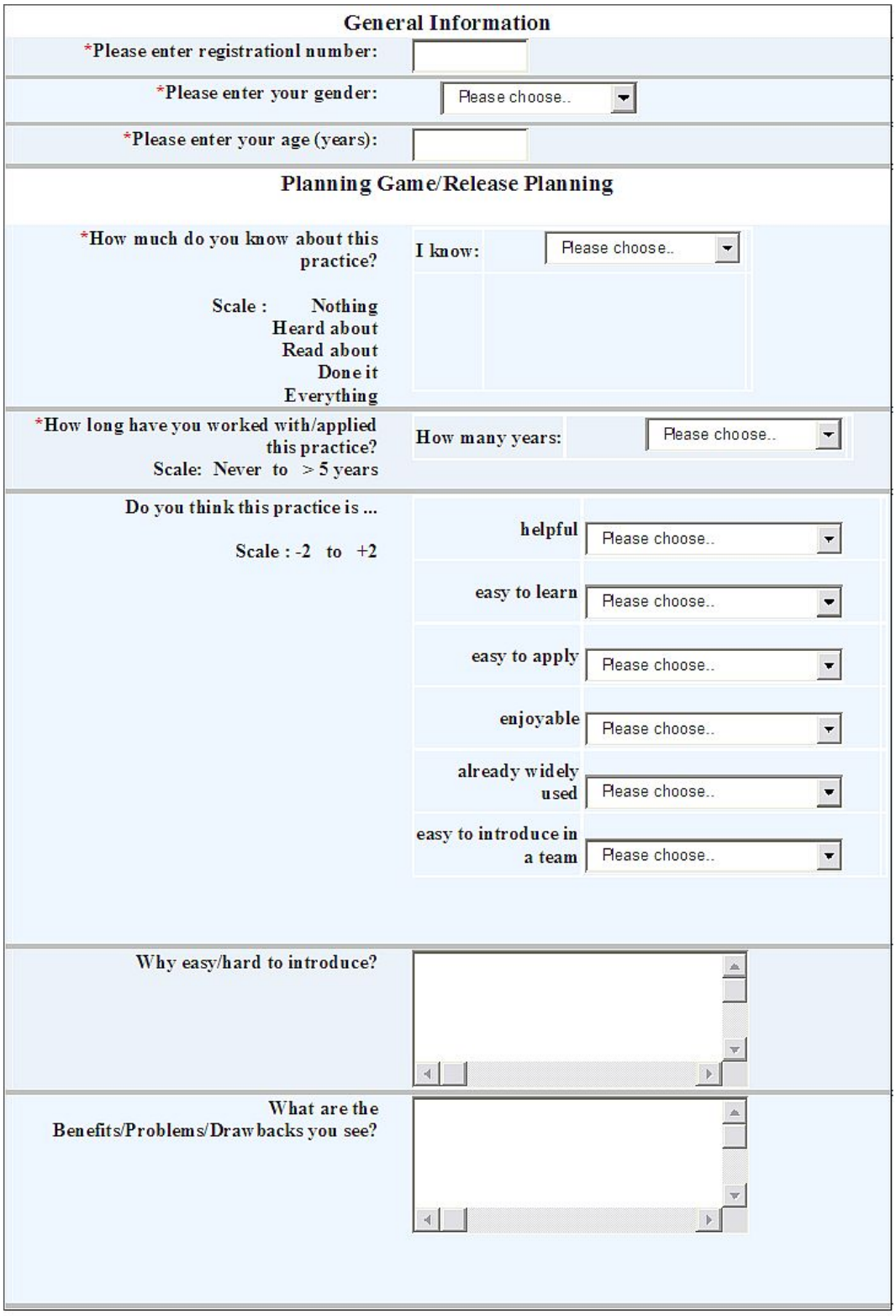
The author plans to adjust the framework to test it for other SDMs as well and to make a comparison study of the usefulness of the framework for coaching and training students and/or professionals using XP, and other agile and non-agile SDMs.

7.4 Conclusion

The study opens the doors for further improvement in software engineering education. It has attempted to simulate industry-like situation in an academic environment which is not as complicated as the real industry setup so that the students can concentrate on learning XP instead of facing other non-educational activities along with the learning process. Role playing activity has been liked by the students because of the fact that the students were able to choose their roles according to their preferences and personality evaluated with the help of a simple personality evaluation form. Many of the students were of the opinion that they were able to identify their personal qualities and apply it in the roles of managers, customers, and developers. The role specific training was also possible as each role was given only role specific tasks to perform in XP-realization phase. In XP-visualization phase the roles were swapped continuously first to provide an opportunity to all students to test their abilities in different roles and also to get knowledge about the working of all roles which contributed to the better understanding of other roles played by their colleagues in XP-realization phase. The overall performance of the framework has been satisfactory from the point of view of the author and also supported by the results and observations of the data analyzed. This further gives a chance to research into the field of design, structure, and organization of software development courses.

Appendix A

XP Practices Survey Questionnaire



General Information

*Please enter registrationl number:

*Please enter your gender: Please choose..

*Please enter your age (years):

Planning Game/Release Planning

*How much do you know about this practice?

Scale : Nothing
Heard about
Read about
Done it
Everything

I know: Please choose..

*How long have you worked with/applied this practice?
Scale: Never to > 5 years

How many years: Please choose..

Do you think this practice is ...

Scale : -2 to +2

helpful: Please choose..

easy to learn: Please choose..

easy to apply: Please choose..

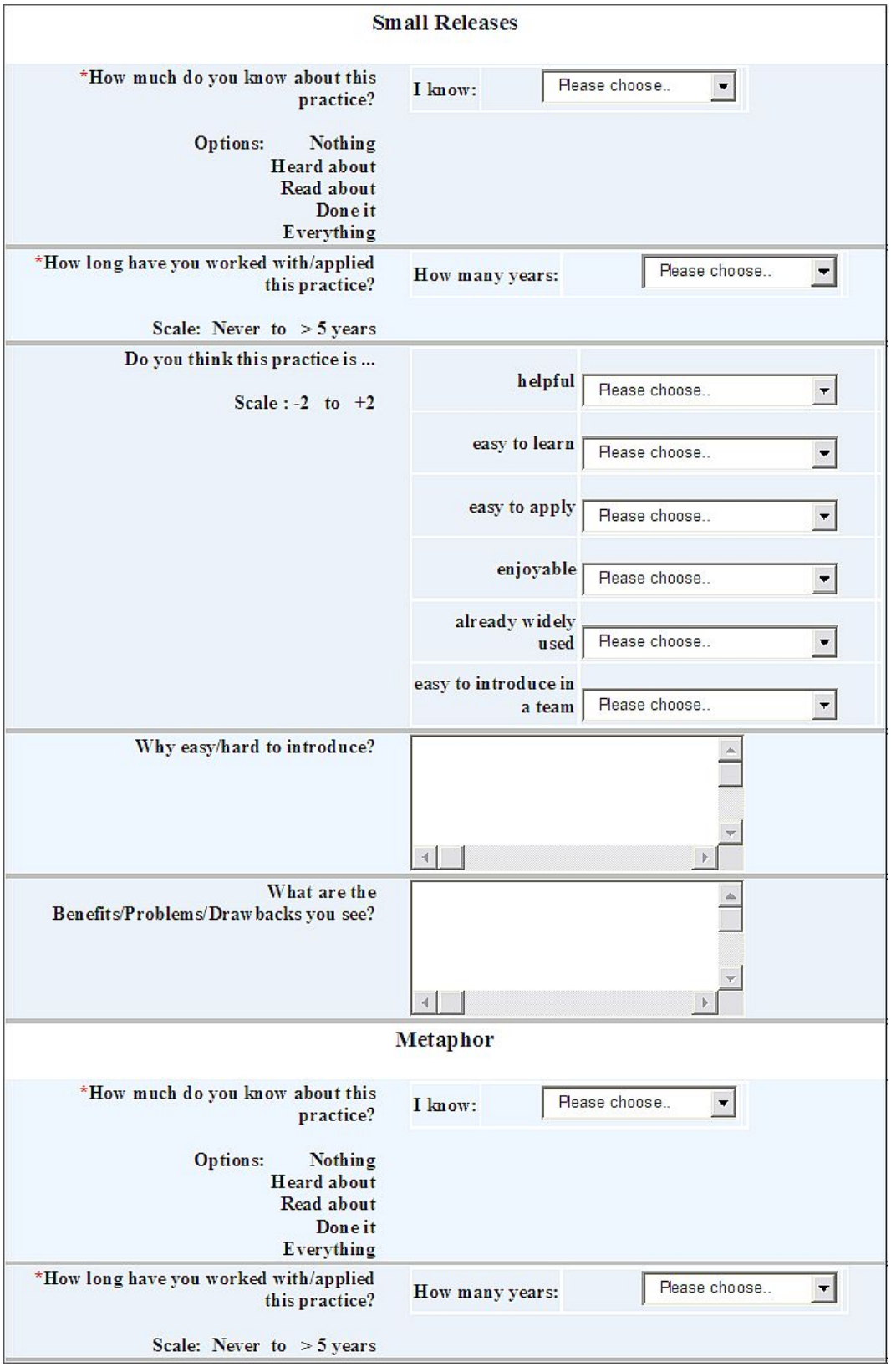
enjoyable: Please choose..

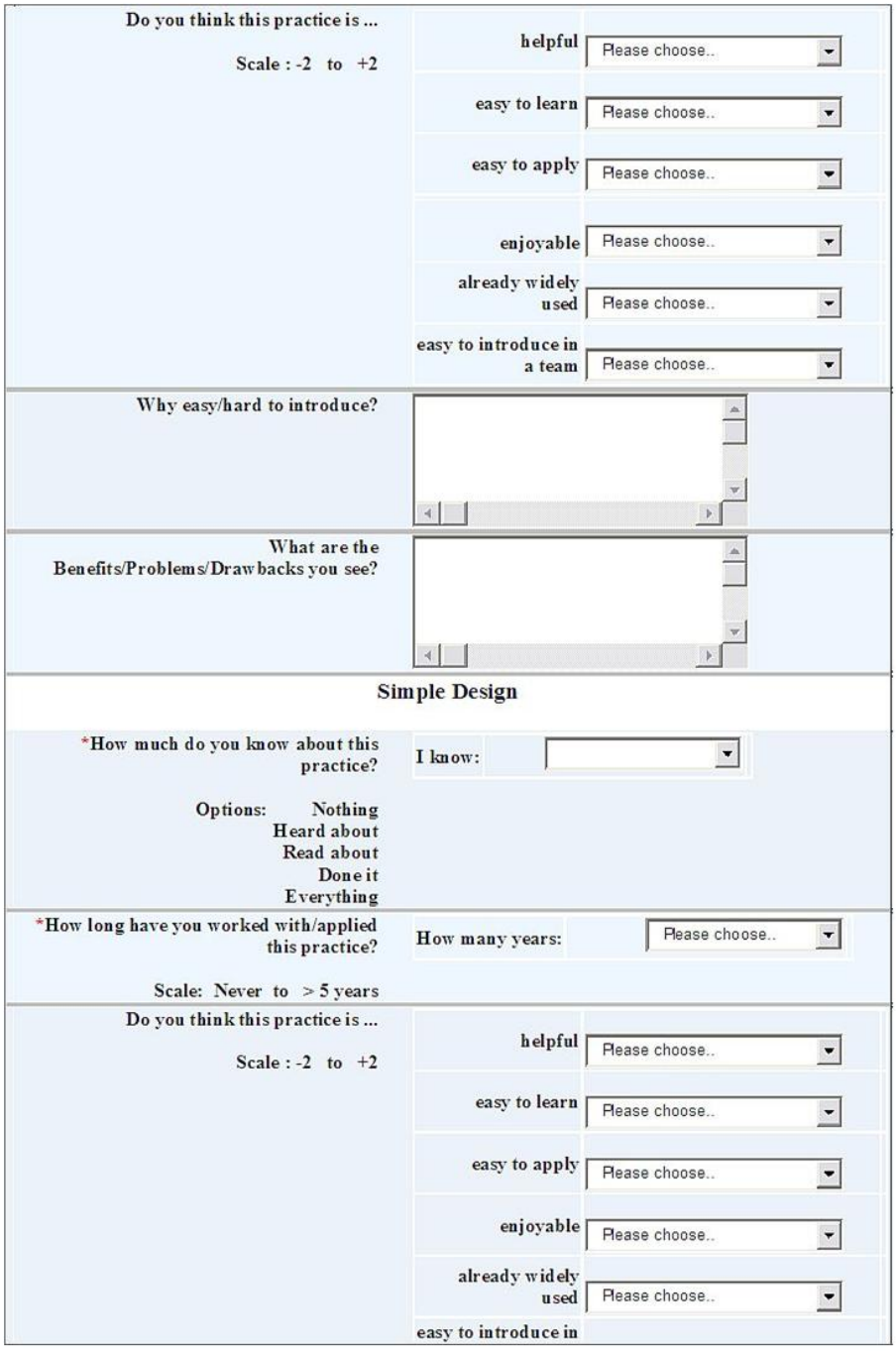
already widely used: Please choose..

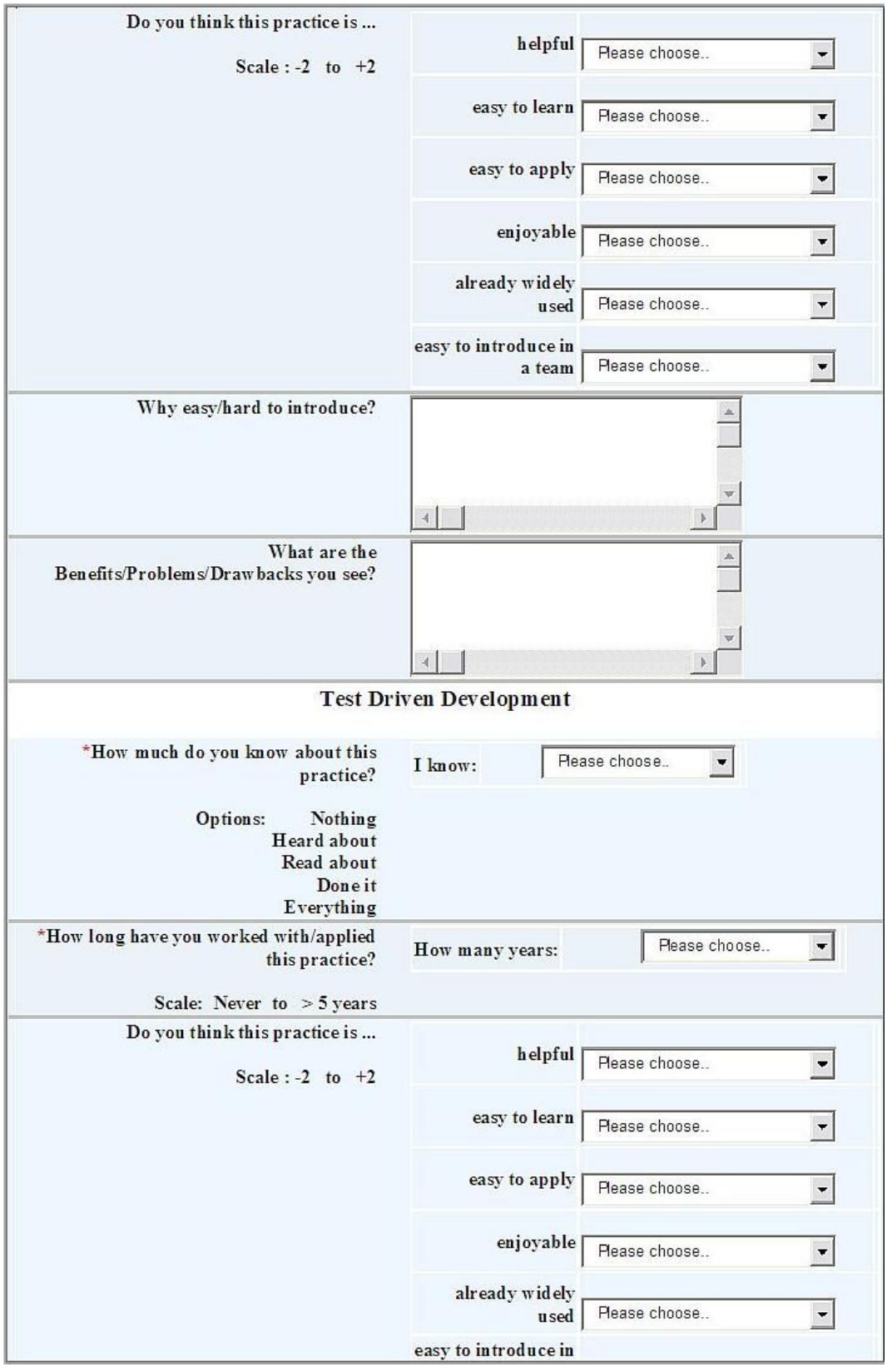
easy to introduce in a team: Please choose..

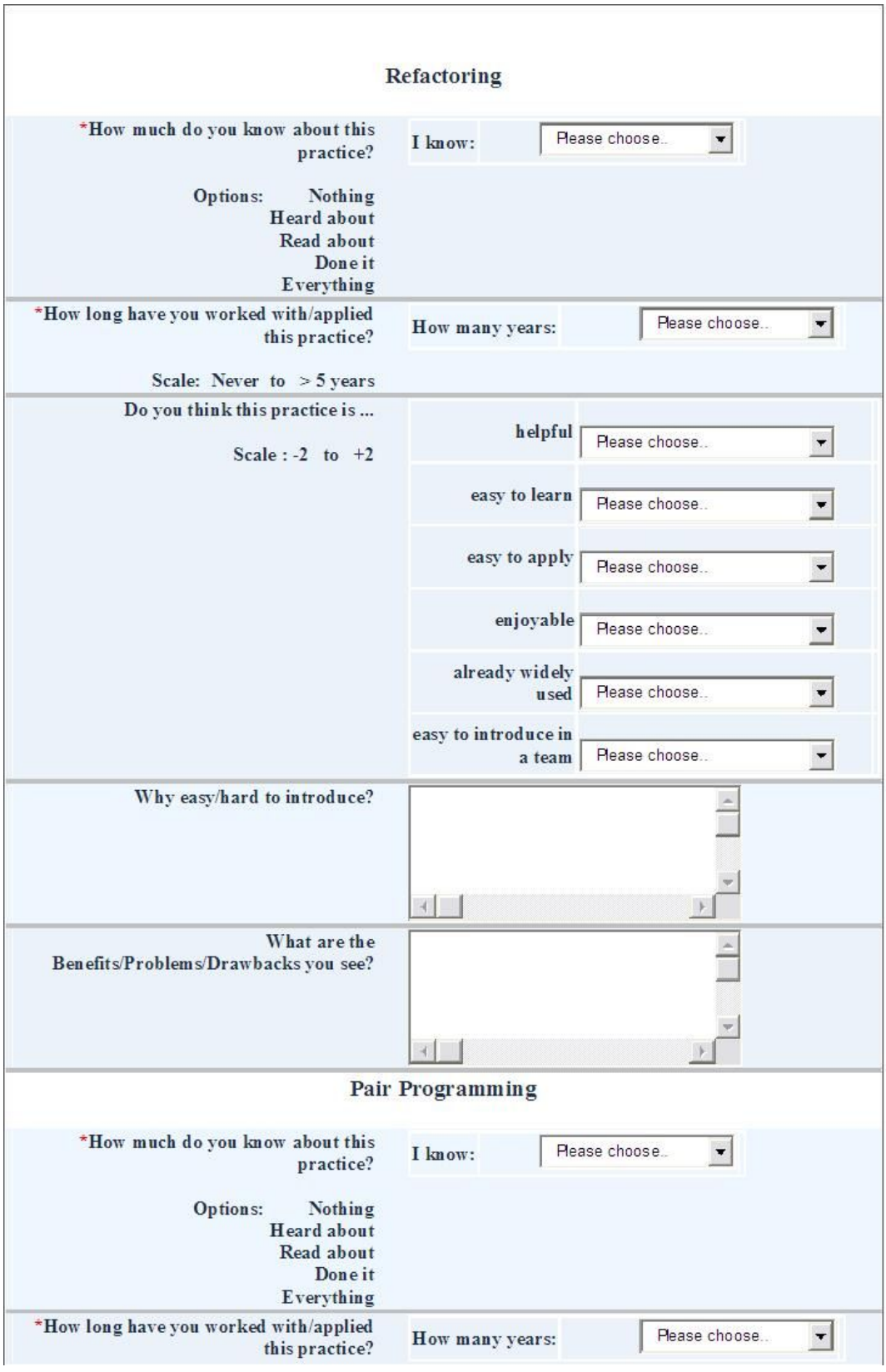
Why easy/hard to introduce?

What are the Benefits/Problems/Drawbacks you see?


Small Releases
*How much do you know about this practice?
Options: Nothing
Heard about
Read about
Done it
Everything
I know:
Please choose..
*How long have you worked with/applied this practice?
Scale: Never to > 5 years
How many years:
Please choose..
Do you think this practice is ...
Scale : -2 to +2
helpful
easy to learn
easy to apply
enjoyable
already widely used
easy to introduce in a team
Please choose..
Why easy/hard to introduce?
What are the Benefits/Problems/Drawbacks you see?
Metaphor
*How much do you know about this practice?
Options: Nothing
Heard about
Read about
Done it
Everything
I know:
Please choose..
*How long have you worked with/applied this practice?
Scale: Never to > 5 years
How many years:
Please choose..


Do you think this practice is ...
Scale : -2 to +2
helpful
Please choose..
easy to learn
Please choose..
easy to apply
Please choose..
enjoyable
Please choose..
already widely used
Please choose..
easy to introduce in a team
Please choose..
Why easy/hard to introduce?
What are the Benefits/Problems/Drawbacks you see?
Simple Design
*How much do you know about this practice?
I know:
Options: Nothing
Heard about
Read about
Done it
Everything
*How long have you worked with/applied this practice?
How many years:
Please choose..
Scale: Never to > 5 years
Do you think this practice is ...
Scale : -2 to +2
helpful
Please choose..
easy to learn
Please choose..
easy to apply
Please choose..
enjoyable
Please choose..
already widely used
Please choose..
easy to introduce in


Do you think this practice is ...
Scale : -2 to +2
helpful
Please choose..
easy to learn
Please choose..
easy to apply
Please choose..
enjoyable
Please choose..
already widely used
Please choose..
easy to introduce in a team
Please choose..
Why easy/hard to introduce?
What are the Benefits/Problems/Drawbacks you see?
Test Driven Development
*How much do you know about this practice?
Options: Nothing
Heard about
Read about
Done it
Everything
I know:
Please choose..
*How long have you worked with/applied this practice?
Scale: Never to > 5 years
How many years:
Please choose..
Do you think this practice is ...
Scale : -2 to +2
helpful
Please choose..
easy to learn
Please choose..
easy to apply
Please choose..
enjoyable
Please choose..
already widely used
Please choose..
easy to introduce in



Refactoring

*How much do you know about this practice?

Options: Nothing
Heard about
Read about
Done it
Everything

I know: Please choose..

*How long have you worked with/applied this practice?

Scale: Never to > 5 years

How many years: Please choose..

Do you think this practice is ...

Scale : -2 to +2

helpful: Please choose..

easy to learn: Please choose..

easy to apply: Please choose..

enjoyable: Please choose..

already widely used: Please choose..

easy to introduce in a team: Please choose..

Why easy/hard to introduce?

What are the Benefits/Problems/Drawbacks you see?

Pair Programming

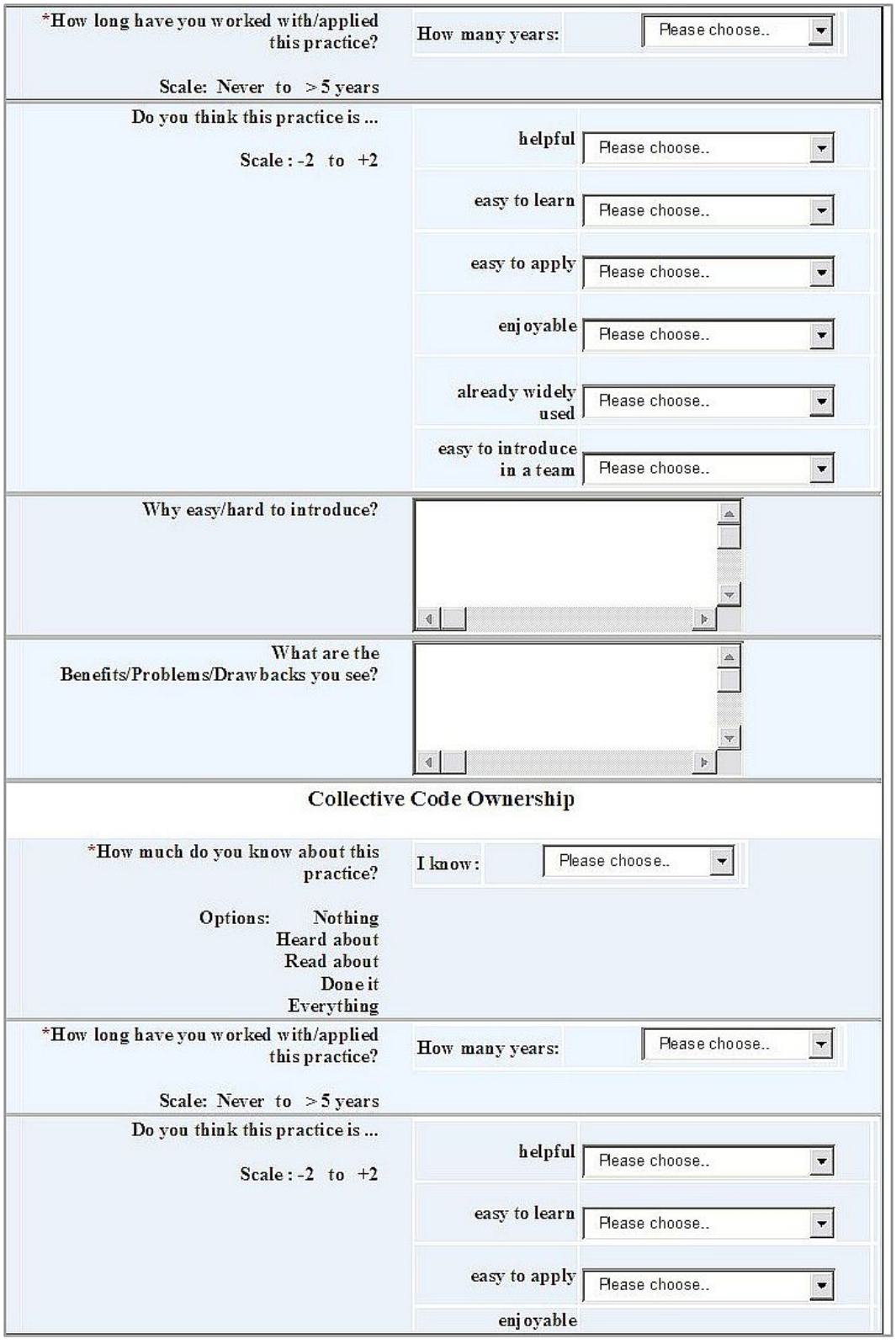
*How much do you know about this practice?

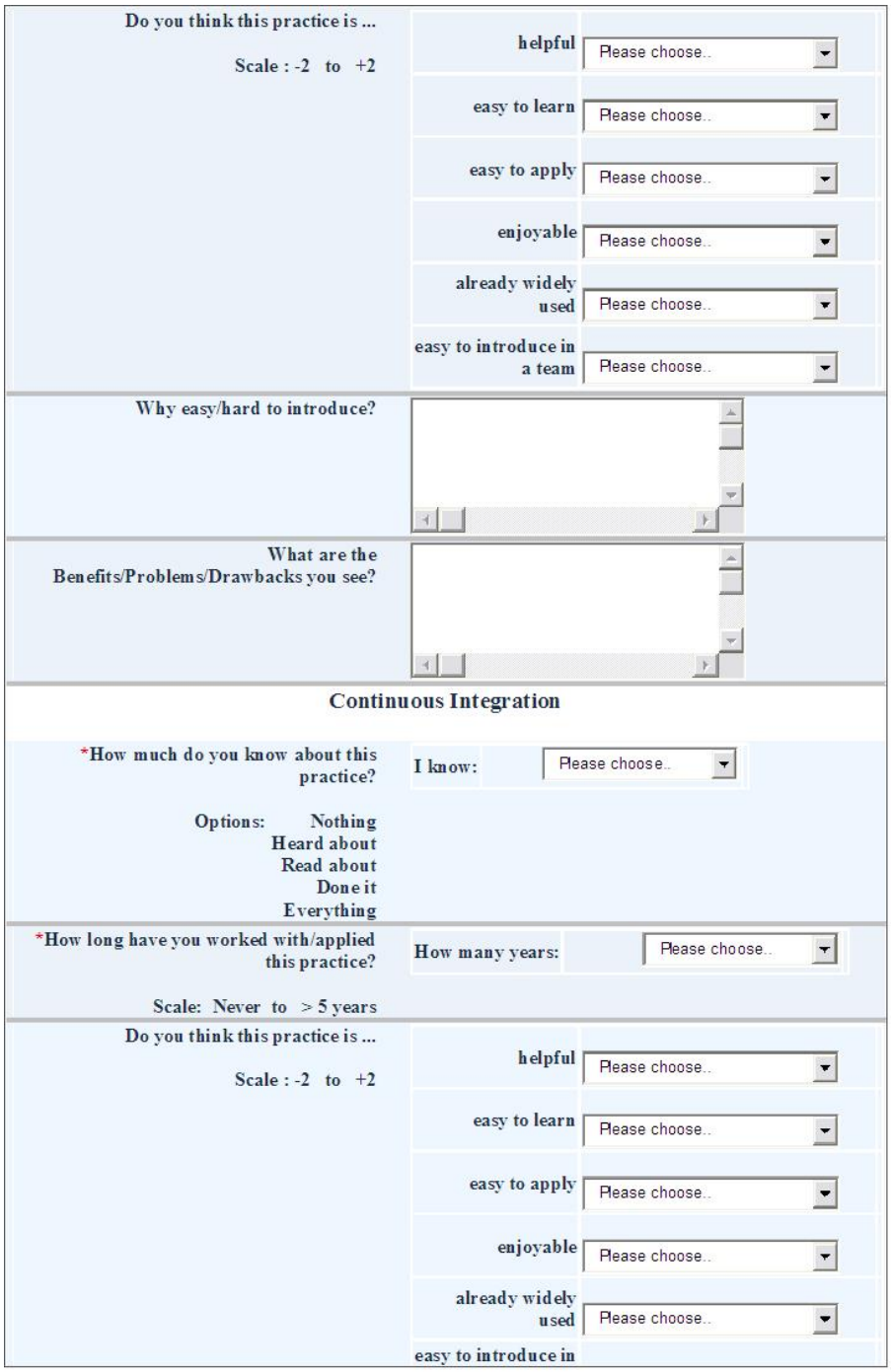
Options: Nothing
Heard about
Read about
Done it
Everything

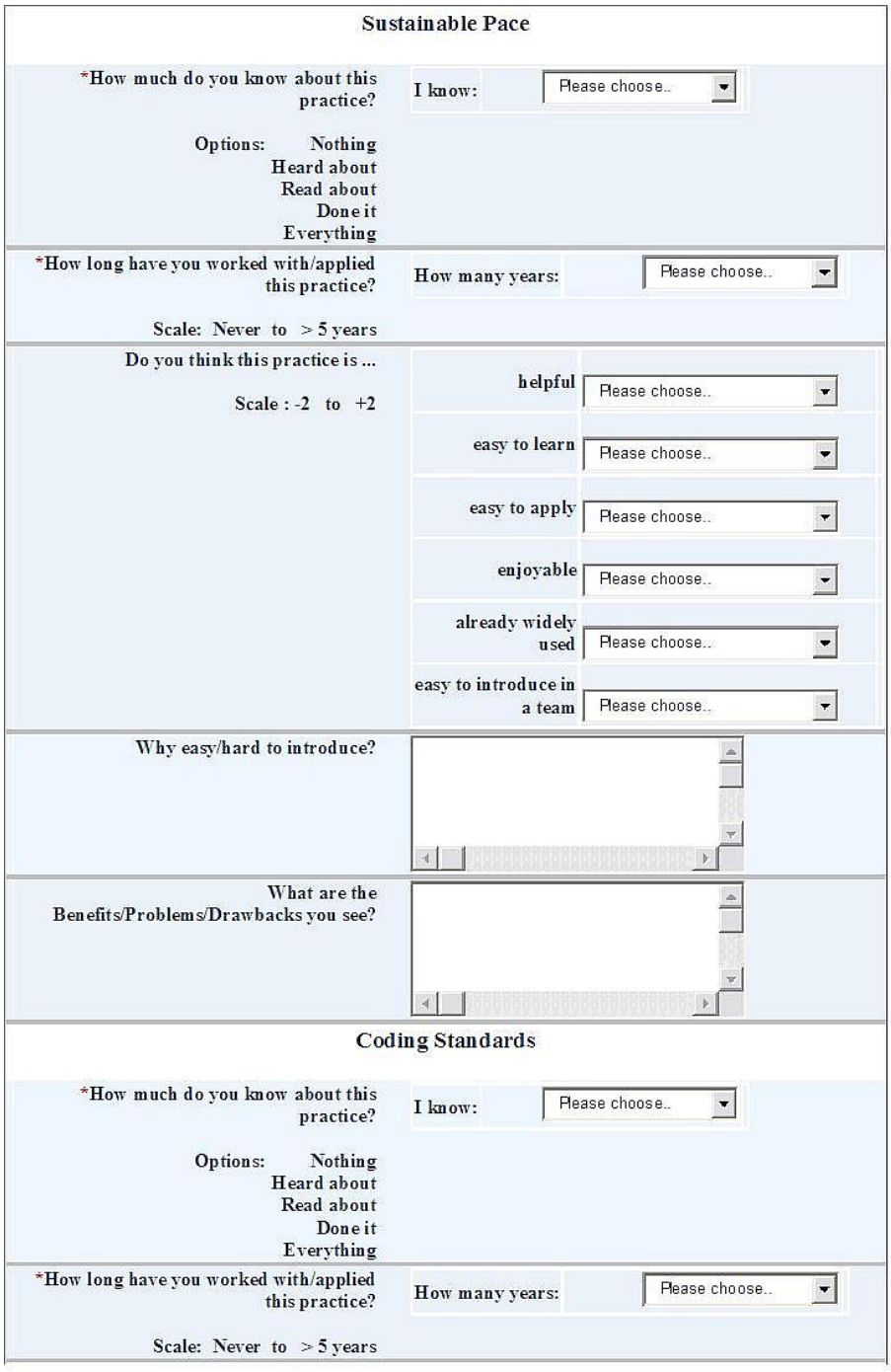
I know: Please choose..

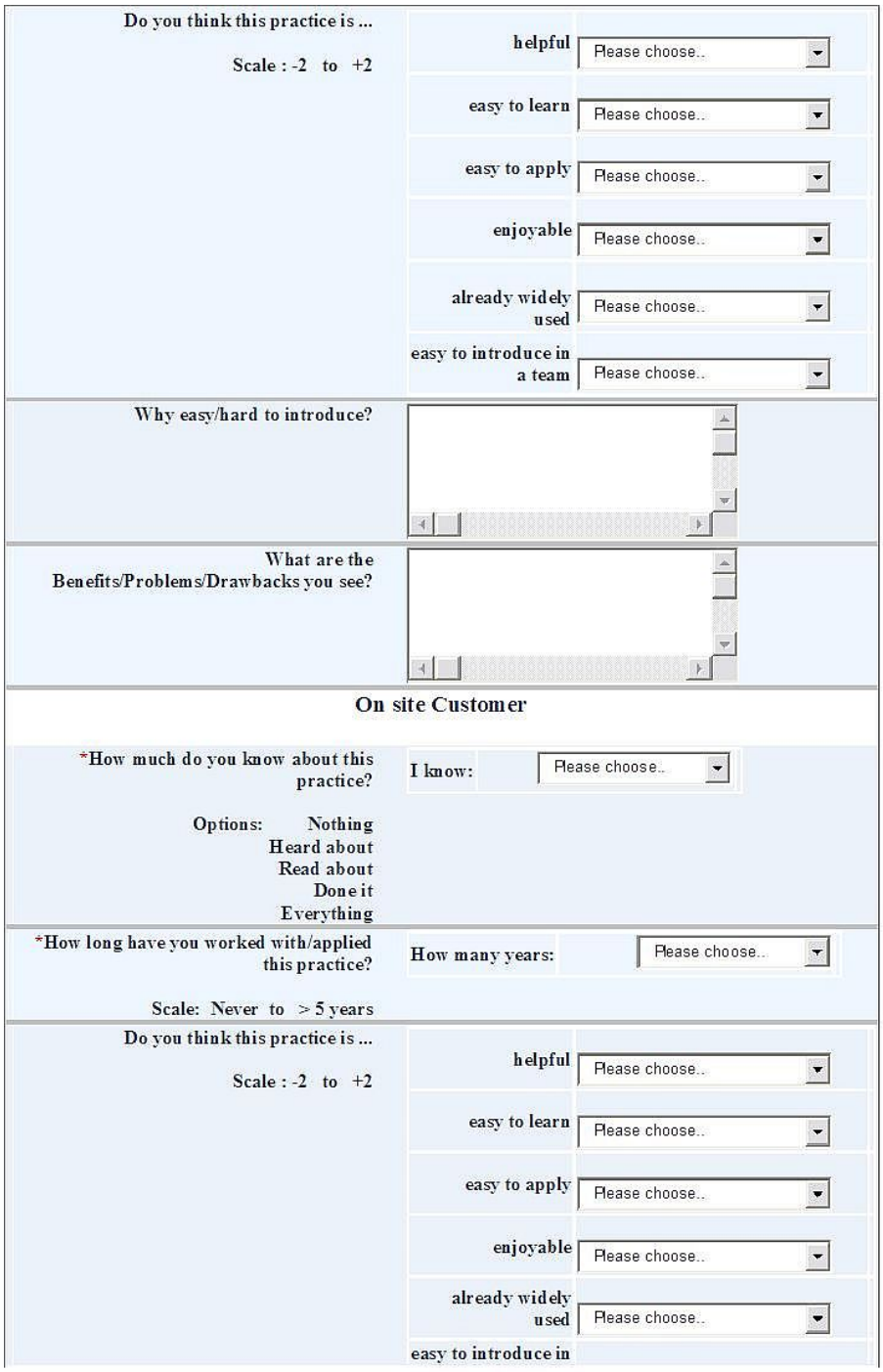
*How long have you worked with/applied this practice?

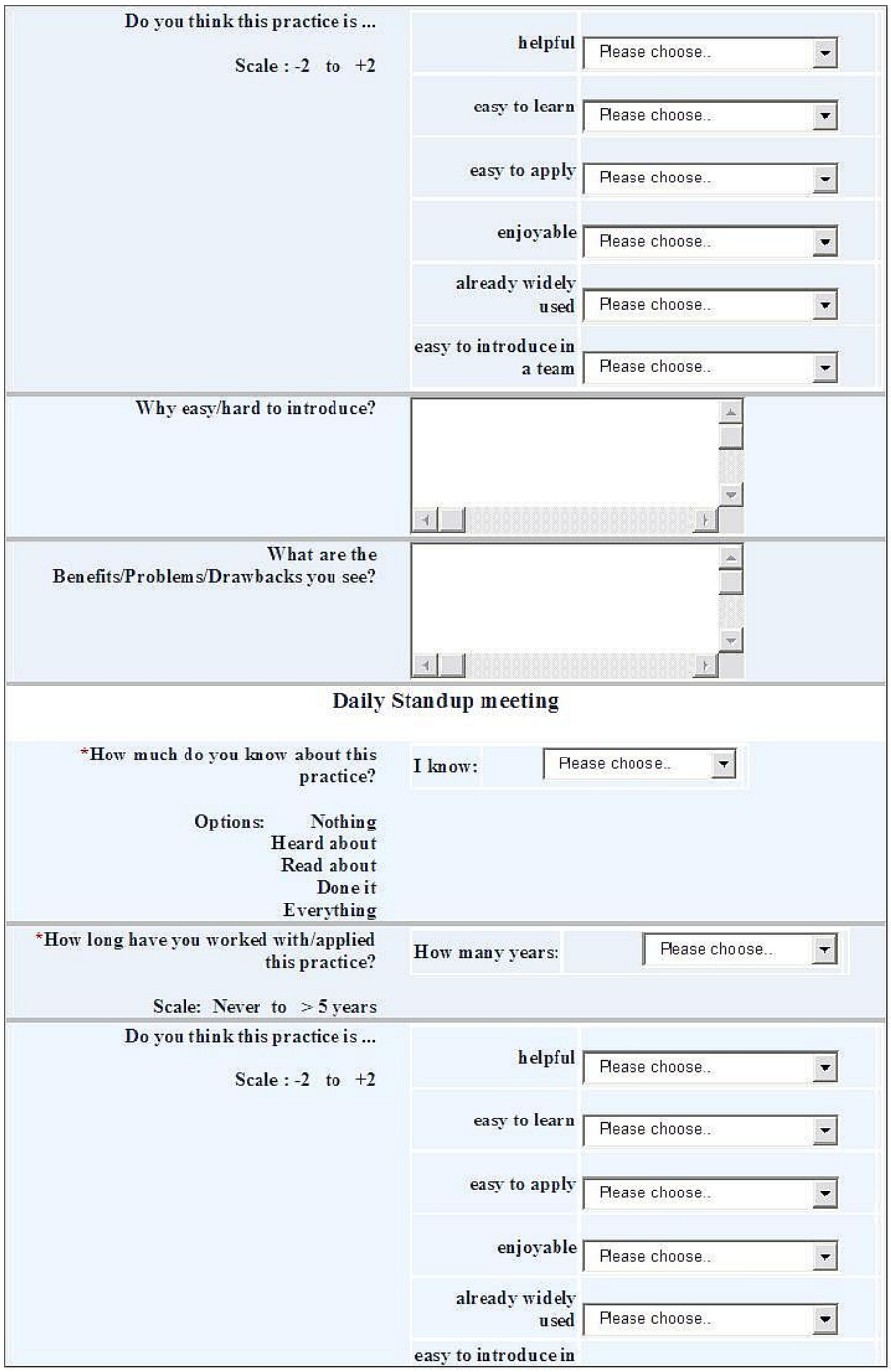
How many years: Please choose..

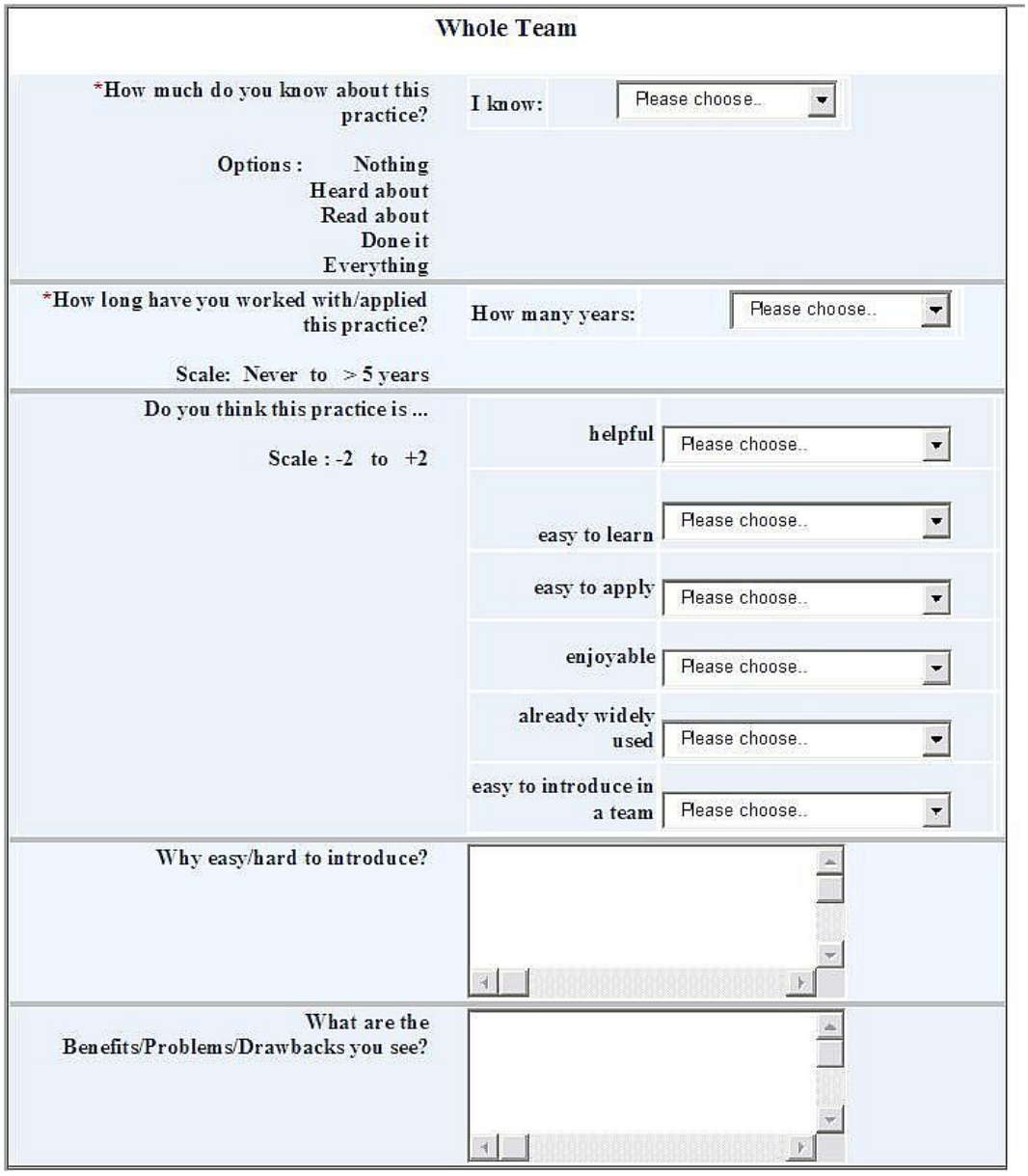

*How long have you worked with/applied this practice?
Scale: Never to >5 years
How many years:
Please choose..
Do you think this practice is ...
Scale : -2 to +2
helpful
easy to learn
easy to apply
enjoyable
already widely used
easy to introduce in a team
Please choose..
Why easy/hard to introduce?
What are the Benefits/Problems/Drawbacks you see?
Collective Code Ownership
*How much do you know about this practice?
Options: Nothing
Heard about
Read about
Done it
Everything
I know:
*How long have you worked with/applied this practice?
Scale: Never to >5 years
How many years:
Do you think this practice is ...
Scale : -2 to +2
helpful
easy to learn
easy to apply
enjoyable
Please choose..


Do you think this practice is ...
Scale : -2 to +2
helpful
Please choose..
easy to learn
Please choose..
easy to apply
Please choose..
enjoyable
Please choose..
already widely used
Please choose..
easy to introduce in a team
Please choose..
Why easy/hard to introduce?
What are the Benefits/Problems/Drawbacks you see?
Continuous Integration
*How much do you know about this practice?
Options: Nothing
Heard about
Read about
Done it
Everything
I know:
Please choose..
*How long have you worked with/applied this practice?
Scale: Never to > 5 years
How many years:
Please choose..
Do you think this practice is ...
Scale : -2 to +2
helpful
Please choose..
easy to learn
Please choose..
easy to apply
Please choose..
enjoyable
Please choose..
already widely used
Please choose..
easy to introduce in


Sustainable Pace
*How much do you know about this practice?
I know:
Please choose..
Options: Nothing
Heard about
Read about
Done it
Everything
*How long have you worked with/applied this practice?
How many years:
Please choose..
Scale: Never to > 5 years
Do you think this practice is ...
Scale : -2 to +2
helpful
easy to learn
easy to apply
enjoyable
already widely used
easy to introduce in a team
Please choose..
Why easy/hard to introduce?
What are the Benefits/Problems/Drawbacks you see?
Coding Standards
*How much do you know about this practice?
I know:
Please choose..
Options: Nothing
Heard about
Read about
Done it
Everything
*How long have you worked with/applied this practice?
How many years:
Please choose..
Scale: Never to > 5 years


Do you think this practice is ...
Scale : -2 to +2
helpful
Please choose..
easy to learn
Please choose..
easy to apply
Please choose..
enjoyable
Please choose..
already widely used
Please choose..
easy to introduce in a team
Please choose..
Why easy/hard to introduce?
What are the Benefits/Problems/Drawbacks you see?
On site Customer
*How much do you know about this practice?
I know:
Please choose..
Options: Nothing
Heard about
Read about
Done it
Everything
*How long have you worked with/applied this practice?
How many years:
Please choose..
Scale: Never to > 5 years
Do you think this practice is ...
Scale : -2 to +2
helpful
Please choose..
easy to learn
Please choose..
easy to apply
Please choose..
enjoyable
Please choose..
already widely used
Please choose..
easy to introduce in


Do you think this practice is ...
Scale : -2 to +2
helpful
Please choose..
easy to learn
Please choose..
easy to apply
Please choose..
enjoyable
Please choose..
already widely used
Please choose..
easy to introduce in a team
Please choose..
Why easy/hard to introduce?
What are the Benefits/Problems/Drawbacks you see?
Daily Standup meeting
*How much do you know about this practice?
Options: Nothing
Heard about
Read about
Done it
Everything
I know:
Please choose..
*How long have you worked with/applied this practice?
Scale: Never to > 5 years
How many years:
Please choose..
Do you think this practice is ...
Scale : -2 to +2
helpful
Please choose..
easy to learn
Please choose..
easy to apply
Please choose..
enjoyable
Please choose..
already widely used
Please choose..
easy to introduce in

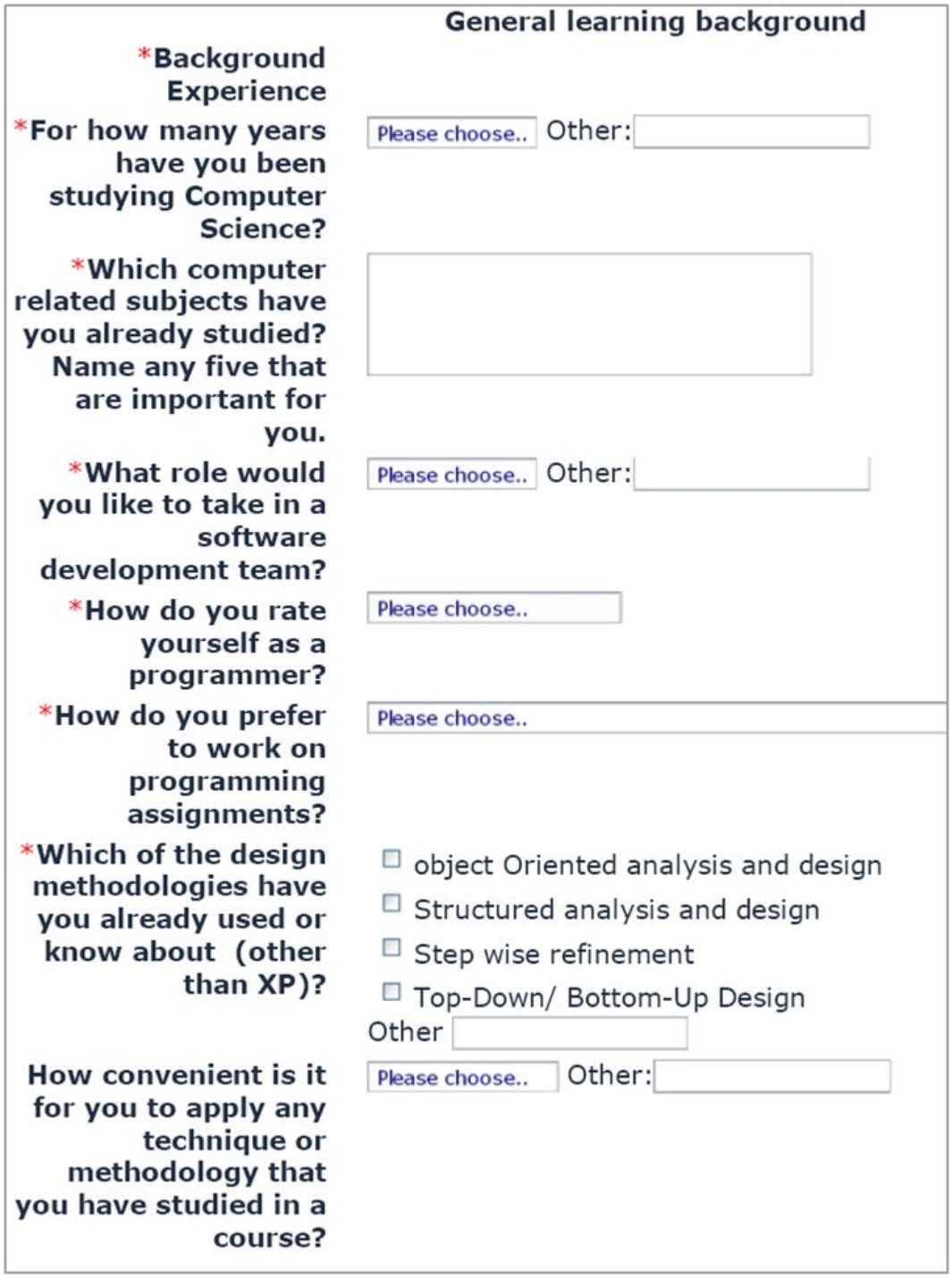


Whole Team

*How much do you know about this practice? Options : Nothing Heard about Read about Done it Everything	I know:	Please choose..
*How long have you worked with/applied this practice? Scale: Never to > 5 years	How many years:	Please choose..
Do you think this practice is ... Scale : -2 to +2	helpful	Please choose..
	easy to learn	Please choose..
	easy to apply	Please choose..
	enjoyable	Please choose..
	already widely used	Please choose..
	easy to introduce in a team	Please choose..
Why easy/hard to introduce?		
What are the Benefits/Problems/Drawbacks you see?		

Appendix B

XP Education Perspective Survey Questionnaire



General learning background

***Background Experience**

***For how many years have you been studying Computer Science?** Please choose.. Other:

***Which computer related subjects have you already studied? Name any five that are important for you.**

***What role would you like to take in a software development team?** Please choose.. Other:

***How do you rate yourself as a programmer?** Please choose..

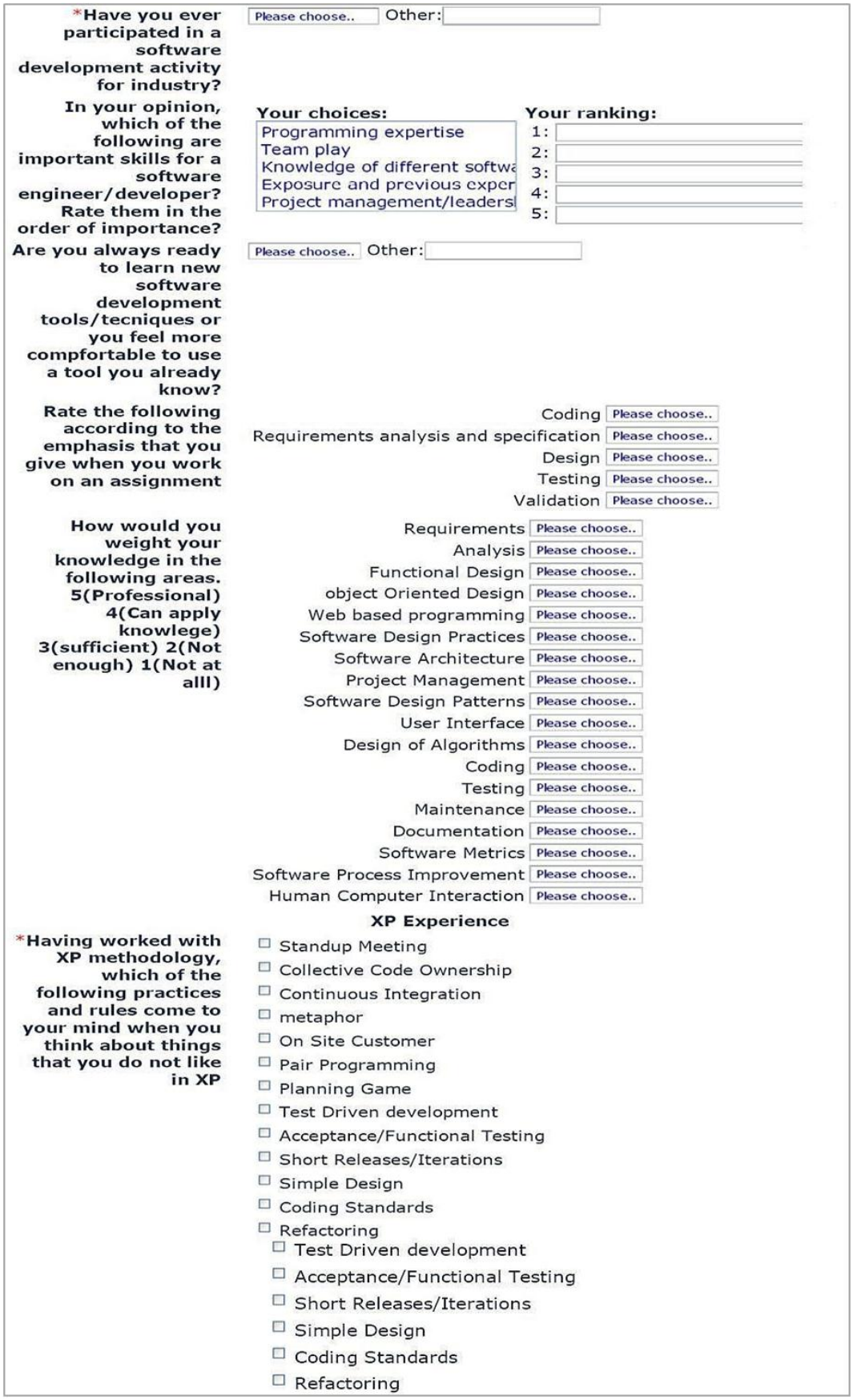
***How do you prefer to work on programming assignments?** Please choose..

***Which of the design methodologies have you already used or know about (other than XP)?**

- ☐ object Oriented analysis and design
- ☐ Structured analysis and design
- ☐ Step wise refinement
- ☐ Top-Down/ Bottom-Up Design

Other

How convenient is it for you to apply any technique or methodology that you have studied in a course? Please choose.. Other:


*Have you ever participated in a software development activity for industry?
Please choose..
Other:
In your opinion, which of the following are important skills for a software engineer/developer? Rate them in the order of importance?
Your choices:
Programming expertise
Team play
Knowledge of different softwa
Exposure and previous exper
Project management/leadersl
Your ranking:
1:
2:
3:
4:
5:
Are you always ready to learn new software development tools/tecniques or you feel more compfortable to use a tool you already know?
Please choose..
Other:
Rate the following according to the emphasis that you give when you work on an assignment
Coding
Requirements analysis and specification
Design
Testing
Validation
Please choose..
How would you weight your knowledge in the following areas. 5(Professional) 4(Can apply knowlege) 3(sufficient) 2(Not enough) 1(Not at alll)
Requirements
Analysis
Functional Design
object Oriented Design
Web based programming
Software Design Practices
Software Architecture
Project Management
Software Design Patterns
User Interface
Design of Algorithms
Coding
Testing
Maintenance
Documentation
Software Metrics
Software Process Improvement
Human Computer Interaction
XP Experience
*Having worked with XP methodology, which of the following practices and rules come to your mind when you think about things that you do not like in XP
Standup Meeting
Collective Code Ownership
Continuous Integration
metaphor
On Site Customer
Pair Programming
Planning Game
Test Driven development
Acceptance/Functional Testing
Short Releases/Iterations
Simple Design
Coding Standards
Refactoring
Test Driven development
Acceptance/Functional Testing
Short Releases/Iterations
Simple Design
Coding Standards
Refactoring

☐ Open Work Space
☐ 40 Hours Week
Other

***Did you work according to the defined XP process or you made some changes in any of the practices (Pair programming, Planning Game, Acceptance test, Test Driven Development)**

Please choose..
Comment:

***How did you actually implemented the practice of Test-driven-development**

Please choose.. Other:

***Your feeling about working in a team.**

Please choose..
Other:

Rank the given XP practices according to their importance for you.

Your choices:	Your ranking:
Pair Programming	1:
Planning Game	2:
Stand up Meeting	3:
Continuous Integration	4:
Collective Code Ownership	5:
metaphor	6:
Simple Design	7:
Test Driven Development	8:
Acceptance/Functional Testin	9:
Short Releases/Iterations	10:
Refactoring	11:
Open Workspace	12:
Forty Hours Week	13:
Coding Standards	14:
On-Site Customer	15:

***Having worked with XP methodology, which of the following XP practices and rules come to your mind when you think about good things in XP.**

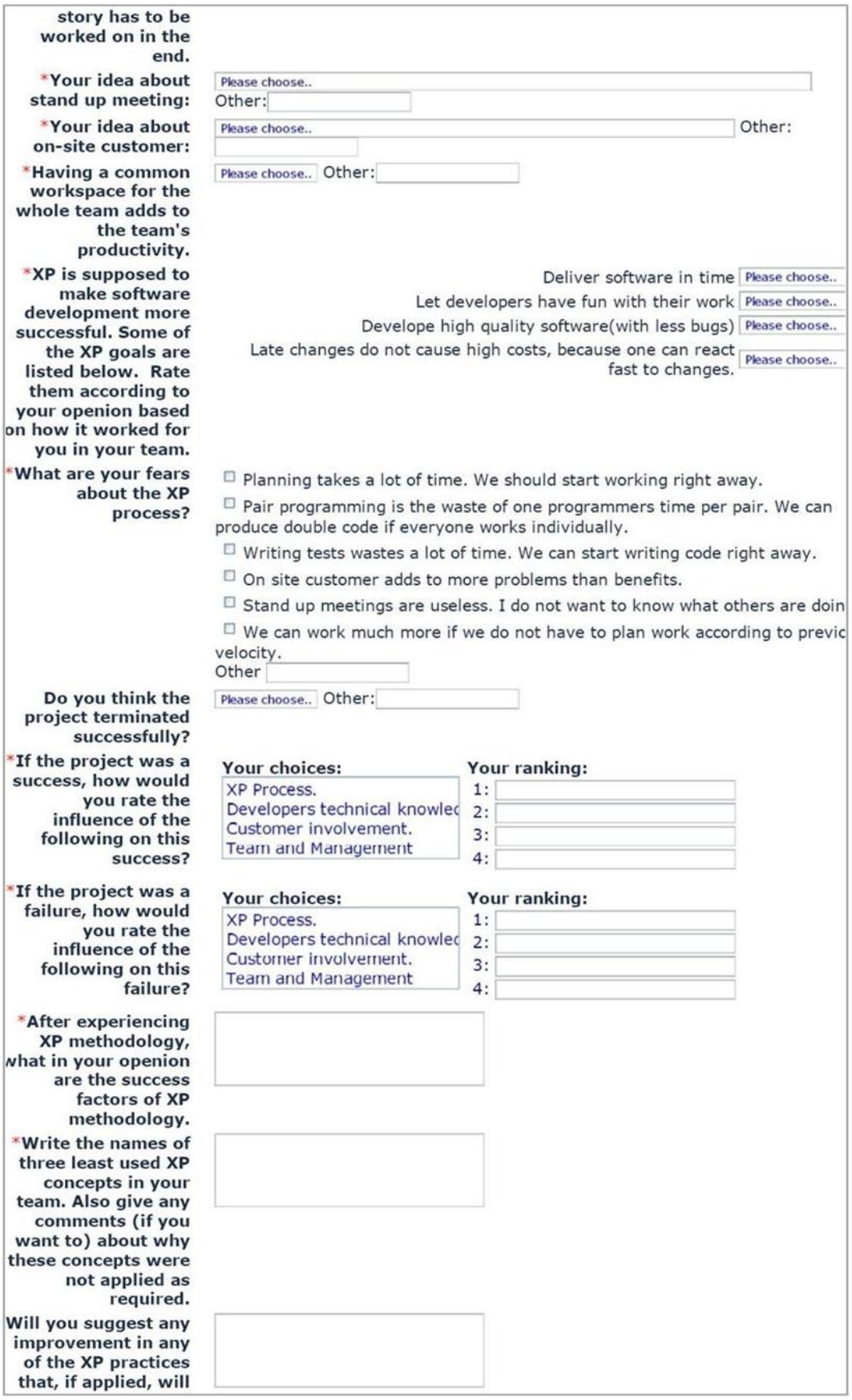
☐ Standup Meeting
☐ Collective Code Ownership
☐ Continuous Integration
☐ metaphor
☐ On Site Customer
☐ Pair Programming
☐ Planning Game
☐ Test Driven development
☐ Acceptance/Functional Testing
☐ Short Releases/Iterations
☐ Simple Design
☐ Coding Standards
☐ Refactoring
☐ Open Work Space
☐ 40 Hours Week
Other

***Your idea about collective code ownership:**

Please choose..
Other:

***Does it make sense for you to estimate and prioritize the stories when every**

Please choose.. Other:



story has to be worked on in the end.

*Your idea about stand up meeting: Please choose.. Other:

*Your idea about on-site customer: Please choose.. Other:

*Having a common workspace for the whole team adds to the team's productivity. Please choose.. Other:

*XP is supposed to make software development more successful. Some of the XP goals are listed below. Rate them according to your openion based on how it worked for you in your team.

Deliver software in time Please choose..
Let developers have fun with their work Please choose..
Develope high quality software(with less bugs) Please choose..
Late changes do not cause high costs, because one can react fast to changes. Please choose..

*What are your fears about the XP process?

- ☐ Planning takes a lot of time. We should start working right away.
- ☐ Pair programming is the waste of one programmers time per pair. We can produce double code if everyone works individually.
- ☐ Writing tests wastes a lot of time. We can start writing code right away.
- ☐ On site customer adds to more problems than benefits.
- ☐ Stand up meetings are useless. I do not want to know what others are doin
- ☐ We can work much more if we do not have to plan work according to previc velocity.

Other

Do you think the project terminated successfully? Please choose.. Other:

*If the project was a success, how would you rate the influence of the following on this success?

Your choices:	Your ranking:
XP Process.	1:
Developers technical knowlec	2:
Customer involvement.	3:
Team and Management	4:

*If the project was a failure, how would you rate the influence of the following on this failure?

Your choices:	Your ranking:
XP Process.	1:
Developers technical knowlec	2:
Customer involvement.	3:
Team and Management	4:

*After experiencing XP methodology, what in your openion are the success factors of XP methodology.

*Write the names of three least used XP concepts in your team. Also give any comments (if you want to) about why these concepts were not applied as required.

Will you suggest any improvement in any of the XP practices that, if applied, will

make the practice more useful for the team and for the software development?	
***Which of the XP practices worked 100% for your team?**	
***Which of the XP practices that you used is most harder to realize?**	
Did it ever come to your mind during the project that you would have done better if you were to use some other development methodology?	Please choose..
If yes, then can you give any reason for that?	
***Will you actively advocate XP if you ever get a chance to work in a software development team?**	Please choose..
Any comments that you want to give about the process and the project.	
What is your opinion about the course?	Please choose.. Other:
What did you like in the whole course?	
What did you not like in the whole course?	

List of Figures

List of Tables

Bibliography

[1] Emma: Java code coverage tool. http://emma.sourceforge.net/.

[2] LinesOfCodeWichtel. www.andreas-berl.de/linesofcodewichtel/en/index.html.

[3] Steven K. Andrianoff and David B. Levine. Role playing in an object-oriented world. In *SIGCSE '02: Proceedings of the 33rd SIGCSE technical symposium on Computer science education*, pages 121–125, New York, NY, USA, 2002. ACM.

[4] Naomi Augar, Ruth Raitman, and Wanlei Zhou. Teaching and learning online with wikis. In *Proceedings of Beyond the Comfort Zone: 21st ASCILITE Conference*, pages 95–104. Jossey-Bass Publishers, 2004.

[5] Donald J. Bagert, Greg Hislop, Michael Lutz, Thomas B. Hilburn, and Thomas B. Hilburn. Guidelines for software engineering education version 1.0. Technical report, CiteSeerX - Scientific Literature Digital Library and Search Engine [http://citeseerx.ist.psu.edu/oai2] (United States), 1999.

[6] V R Basili, R W Selby, and D H Hutchens. Experimentation in software engineering. *IEEE Trans. Softw. Eng.*, 12(7):733–743, 1986.

[7] Kent Beck. Extreme programming: A humanistic discipline of software development. In *FASE*, pages 1–6, 1998.

[8] Kent Beck. *Extreme Programming Explained: Embrace Change (1st Edition)*. Addison-Wesley Professional, 1999.

[9] Kent Beck and Cynthia Andres. *Extreme Programming Explained : Embrace Change (2nd Edition)*. Addison-Wesley Professional, November 2004.

[10] Kent Beck and Martin Fowler. *Planning Extreme Programming*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2000.

[11] H. Beckman, G. O'Mary, J. Lawrence, and C.L. Parish. Tracking and evaluating industry/university collaborations for software engineering education and training. In *Software Engineering Education and Training, 1999. Proceedings. 12th Conference on*, pages 102–103, Mar 1999.

[12] Finn Olav Bjørnson and Torgeir Dingsøyr. A survey of perceptions on knowledge management schools in agile and traditional software development environments. In Pekka Abrahamsson, Michele Marchesi, and Frank Maurer, editors, *XP*, volume 31 of *Lecture Notes in Business Information Processing*, pages 94–103. Springer, 2009.

[13] Charles C. Bonwell and James A. Eison. *Active learning: Creating excitement in the classroom (J-B ASHE Higher Education Report Series (AEHE))*. Jossey-Bass, February 1991.

[14] David Carrington, Paul Strooper, Sharron Newby, and Terry Stevenson. An industry/university collaboration to upgrade software engineering knowledge and skills in industry. *Journal of Systems and Software*, 75(1-2):29 – 39, 2005. Software Engineering Education and Training.

[15] Thomas Chau and Frank Maurer. Knowledge sharing in agile software teams. *Logic versus approximation: Essays dedicated to Michael M. Richter on the occasion of his 65th birthday*, LNCS 3075:173–183, 2004.

[16] A. W. Chickering and Z. F. Gamson. Seven principles for good practice in undergraduate education. *AAHE Bulletin*, 39(7):3–7, 1987.

[17] T CHOW and D CAO. A survey study of critical success factors in agile software projects. *Journal of Systems and Software*, 81(6):961–971, June 2008.

[18] Larry L. Constantine and Lucy A. D. Lockwood. Usage-centered software engineering: an agile approach to integrating users, user interfaces, and usability into software engineering practice. In *ICSE '03*, pages 746–747. IEEE Computer Society, 2003.

[19] Esther Derby and Diana Larsen. *Agile retrospectives: Making good teams great.* Pragmatic Bookshelf, 2006.

[20] Anke Drappa and Jochen Ludewig. Simulation in software engineering training. In *ICSE '00: Proceedings of the 22nd international conference on Software engineering*, pages 199–208, New York, NY, USA, 2000. ACM.

[21] Yael Dubinsky and Orit Hazzan. A framework for teaching software development methods. *Computer Science Education*, 15(4):275–296, 2005.

[22] Jutta Eckstein. *Agile software development in the large: Diving into the deep.* Dorset House Publishing Company, Incorporated, New York, NY, USA, June 2004.

[23] Jacques Morel et al. Xplanner: (xp) project planning and tracking tool. http://www.xplanner.org/. Visited on 04.01.2008.

[24] Jennifer Ferreira, James Noble, and Robert Biddle. Agile development iterations and UI design. In *Agile 2007*, pages 50–58. IEEE Computer Society, 2007.

[25] Libero Ficocelli. Industry and academia: How can we collaborate? *appeared in Journal "Loading...,"*, 1(1), 2007.

[26] William A. Florac, Anita D. Carleton, and Julie R. Barnard. Statistical process control: Analyzing a space shuttle onboard software process. *IEEE Software*, 17(4):97–106, 2000.

[27] Troy Frever and Paul Ingalls. The pairing session as the atomic unit of work. *Agile 2006*, pages 165–169, 2006.

[28] Hmood Al Dossari Ghazy Assassa, Hassan Mathkour. Extreme programming: A case study in software engineering courses. In *Proceedings of the 1st National Information Technology Symposium, NITS, Riyadh, Saudi Arabia*, pages 233–240, 2006.

[29] W. Gibbs. Software's chronic crisis. *Scientific American,*, 3:86–95, September 1994.

[30] Federico Gobbo and Matteo Vaccari. The pomodoro technique for sustainable pace in extreme programming teams. In *XP 2008*, pages 180–184, 2008.

[31] Jan Gulliksen, Bengt Göransson, Inger Boivie, Stefan Blomkvist, Jenny Persson, and Äsa Cajander. Key principles for user-centred systems design. *Behaviour & Information Technology, Special Section on Designing IT for Healthy Work.*, Vol. 22 No. 6:397–409, 2003.

[32] Orit Hazzan and Yael Dubinsky. Teaching a software development methodology: The case of extreme programming. *Conference on Software Engineering Education and Training*, 1:176, 2003.

[33] Orit Hazzan and Yael Dubinsky. Teaching framework for software development methods. In *ICSE '06: Proceedings of the 28th international conference on Software engineering*, pages 703–706, New York, NY, USA, 2006. ACM.

[34] Orit Hazzan and James E. Tomayko. Human aspects of software engineering: The case of extreme programming. In *XP 2004*, pages 303–311, 2004.

[35] Görel Hedin, Lars Bendix, and Boris Magnusson. Introducing software engineering by means of extreme programming. In *ICSE*, pages 586–593. IEEE Computer Society, 2003.

[36] Görel Hedin, Lars Bendix, and Boris Magnusson. Teaching extreme programming to large groups of students. *Journal of Systems and Software*, 74(2):133–146, 2005.

[37] Tyson R. Henry and Janine LaFrance. Integrating role-play into software engineering courses. *J. Comput. Small Coll.*, 22(2):32–38, 2006.

[38] Jim Highsmith. *Agile project management: Creating innovative products*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 2004.

[39] Jim Highsmith and Martin Fowler. The agile manifesto. *Software Development Magazine*, 9(8):29–30, 2001.

[40] Thomas B. Hilburn. Software engineering education: A modest proposal. *IEEE Software*, 14:44–48, 1997.

[41] Thomas B. Hilburn, Greg Hislop, Donald J. Bagert, Michael Lutz, Susan Mengel, and Michael McCracken. Guidance for the development of software engineering education programs. *Journal of Systems and Software*, 49(2-3):163 – 169, 1999.

[42] Thomas B. Hilburn, Susan Mengel, Donald J. Bagert, and Dale Oexmann. Software engineering across computing curricula. In *ITiCSE '98: Proceedings of the 6th annual conference on the teaching of computing and the 3rd annual conference on Integrating technology into computer science education*, pages 117–121, New York, NY, USA, 1998. ACM.

[43] Mike Holcombe, Marian Gheorghe, and Francisco Macias. Teaching xp for real: some initial observations and plans. In *In Proceedings of 2nd International Conference on Extreme Programming and Flexible Processes in Software Engineering*, pages 14–17. Pearson Education Inc, 2001.

[44] Andreas Holzinger, Maximilian Errath, Gig Searle, Bettina Thurnher, and Wolfgang Slany. From extreme programming and usability engineering to extreme usability in software engineering education (XP+UE→XU). In *COMPSAC '05: Proceedings of the 29th Annual International Computer Software and Applications Conference (COMPSAC'05) Volume 2*, pages 169–172, Washington, DC, USA, 2005. IEEE Computer Society.

[45] Zahid Hussain, Martin Lechner, Harald Milchrahm, Sara Shahzad, Wolfgang Slany, Martin Umgeher, and Thomas Vlk. Optimizing Extreme Programming. In *ICCCE 2008: Proceedings of the International Conference on Computer and Communication Engineering, Kuala Lumpur, Malaysia*, pages 1052–1056. IEEE, 2008.

[46] Zahid Hussain, Martin Lechner, Harald Milchrahm, Sara Shahzad, Wolfgang Slany, Martin Umgeher, Thomas Vlk, and Peter Wolkerstorfer. User interface design for a content-aware mobile multimedia application: An iterative approach. In *Frontiers in Mobile and Web Computing: Proceedings of MoMM2007 & ii-WAS2007 Workshops*, volume 231, pages 115–120, Jakarta, Indonesia, December 2007. 5th @WAS International Conference on Mobile Computing and Multimedia (MoMM2007), 3-5 December, 2007, Jakarta, Indonesia ISBN : 978-3-85403-231-1.

[47] Zahid Hussain, Martin Lechner, Harald Milchrahm, Sara Shahzad, Wolfgang Slany, Martin Umgeher, Thomas Vlk, and Peter Wolkerstorfer. User interface design for a mobile multimedia application: An iterative approach. In *ACHI 2008, First International Conference on Advances in Computer-Human Interaction, February 10-15, 2008, Sainte Luce, Martinique, France*, pages 189–194. IEEE Computer Society, 2008.

[48] Zahid Hussain, Martin Lechner, Harald Milchrahm, Sara Shahzad, Wolfgang Slany, Martin Umgeher, Thomas Vlk, and Peter Wolkerstorfer. User interface design for a mobile multimedia application: An iterative approach. In *ACHI 2008, First International Conference on Advances in Computer-Human Interaction, Sainte Luce, Martinique, France*, pages 189–194. IEEE Computer Society, February 10-15, 2008.

[49] Zahid Hussain, Martin Lechner, Harald Milchrahm, Sara Shahzad, Wolfgang Slany, Martin Umgeher, and Peter Wolkerstorfer. Agile User-Centered Design Applied to a Mobile Multimedia Streaming Application. In *USAB 2008*, volume 5298/2008 of *LNCS*, pages 313–330. Springer Berlin / Heidelberg, November 2008.

[50] Zahid Hussain, Martin Lechner, Harald Milchrahm, Sara Shahzad, Wolfgang Slany, Martin Umgeher, and Peter Wolkerstorfer. Integrating Extreme Programming and User-Centered Design. In *PPIG 2008, The 20th Annual Psychology of Programming Interest Group Conference, Lancaster University, UK. 10th - 12th September 2008*, 2008.

[51] Zahid Hussain, Martin Lechner, Harald Milchrahm, Sara Shahzad, Wolfgang Slany, Martin Umgeher, and Peter Wolkerstorfer. Concept and design of a contextual mobile multimedia content usability study. In *ACHI 2009, Second International Conference on Advances in Computer-Human Interaction, February 1-7, 2009 - Cancun, Mexico*. IEEE Computer Society, 2009. To appear.

[52] Zahid Hussain, Harald Milchrahm, Sara Shahzad, Wolfgang Slany, Manfred Tscheligi, and Peter Wolkerstorfer. Integration of extreme programming and user-centered design: Lessons learned. In Pekka Abrahamsson, Michele Marchesi, and Frank Maurer, editors, *XP 2009*, volume 31 of *LNBIP*, pages 174–179. Springer, 2009.

[53] Zahid Hussain, Wolfgang Slany, and Andreas Holzinger. Investigating agile user-centered design in practice: A grounded theory perspective. In A. Holzinger and

K. Miesenberger, editors, *HCI and Usability for e-Inclusion. 5th Symposium of theWorkgroup Human-Computer Interaction and Usability Engineering of the Austrian Computer Society*, volume 5889 of *Lecture Notes in Computer Science*, pages 279–289, Berlin, Heidelberg, New York, 2009. Springer.

[54] SystemShare Inc. Extreme programming an effective framework for knowledge management. electronic, 2003. Accessed: May 2009.

[55] Daniel M. Johnson, Peter Sutton, and Neil Harris. Extreme programming requires extremely effective communication: Teaching effective communication skills to students in an it degree. *An IT Degree. Proceedings of 18 th ASCILITE 2001*, 2001.

[56] Timo Jokela and Pekka Abrahamsson. Usability assessment of an extreme programming project: Close co-operation with the customer does not equal to good usability. In *5th International Conference, PROFES '04*, pages 393–407, 2004.

[57] V. Jovanovic, T. Murphy, and A. Greca. Use of extreme programming (xp) in teaching introductory programming. In *Frontiers in Education, Annual*, volume 3, pages F1G23–22, Los Alamitos, CA, USA, 2002. IEEE Computer Society.

[58] Frank Keenan. Agile process tailoring and problem analysis (aptly). In *ICSE '04: Proceedings of the 26th International Conference on Software Engineering*, pages 45–47, Washington, DC, USA, 2004. IEEE Computer Society.

[59] Norman L. Kerth. *Project retrospectives: a handbook for team reviews.* Dorset House Publishing Co., Inc., New York, NY, USA, 2001.

[60] J. Kivi, D. Haydon, J. Hayes, and G. Schneider, R.and Succi. Extreme programming: A university team design experience. In IEEE, editor, *Electrical and Computer Engineering, 2000 Canadian Conference on*, volume Volume 2, of *Electrical and Computer Engineering, 2000 Canadian Conference on*, pages Page(s):816 – 820, 7-10 March 2000.

[61] A. S. Koch. *Agile software development evaluating the methods for your organization.* Artech House. Boston, 2005.

[62] Andrew J. Kornecki, Iraj Hirmanpour, Massood Towhidnajad, Roger Boyd, Theresa Ghiorzi, and Linda Margolis. Strengthening software engineering education through academic industry collaboration. *Software Engineering Education and Training, Conference on*, 0:204, 1997.

[63] Bill Krebs. http://agile.csc.ncsu.edu/survey/shodan_survey.html. Visited on 04.01.2008.

[64] Michael Leitner, Peter Wolkerstorfer, Reinhard Sefelin, and Manfred Tscheligi. Mobile multimedia: identifying user values using the means-end theory. In *Proceedings of the 10th international conference on Human computer interaction with mobile devices and services*, pages 167–175, Amsterdam, The Netherlands, 2008. ACM.

[65] Noel F LeJeune. Teaching software engineering practices with extreme programming. *J. Comput. Small Coll.*, 21(3):107–117, 2006.

[66] M. Levy and O. Hazzan. Knowledge management in practice: The case of agile software development. In *Cooperative and Human Aspects on Software Engineering, 2009. CHASE '09. ICSE Workshop on*, pages 60–65, May 2009.

[67] R. Likert. A technique for the measurement of attitudes. *Archives of Psychology*, 22(140):1–55, 1932.

[68] Mikael Lindvall, Vic Basili, Barry Boehm, Patricia Costa, Forrest Shull, Roseanne Tesoriero, Laurie Williams, and Marvin Zelkowitz. Empirical findings in agile methods. In *In Proceedings of the Second XP Universe and First Agile Universe Conference on Extreme Programming and Agile Methods - XP/Agile Universe 2002*, pages 197–207. Springer-Verlag, 2002.

[69] Kim Man Lui and Keith C. C. Chan. A road map for implementing extreme programming. In Mingshu Li, Barry W. Boehm, and Leon J. Osterweil, editors, *ISPW*, volume 3840 of *Lecture Notes in Computer Science*, pages 474–481. Springer, 2005.

[70] F. Macias. *Empirical Assessment of Extreme Programming.* PhD thesis, University of Sheffield., 2004.

[71] Michele Marchesi and Giancarlo Succi, editors. *Extreme Programming and Agile Processes in Software Engineering, 4th International Conference, XP 2003, Genova, Italy, May 25-29, 2003 Proceedings*, volume 2675 of *Lecture Notes in Computer Science*. Springer, 2003.

[72] John McBurnie. Agile knowledge management. Article online, January 2010.

[73] Charlie McDowell, Linda Werner, Heather E. Bullock, and Julian Fernald. The impact of pair programming on student performance, perception and persistence. In *ICSE '03: Proceedings of the 25th International Conference on Software Engineering*, pages 602–607, Washington, DC, USA, 2003. IEEE Computer Society.

[74] Paul McInerney and Frank Maurer. UCD in agile projects: dream team or odd couple? *Interactions*, 12(6):19–23, 2005.

[75] Marc McNeill. User centred design in agile application development. http://www.thoughtworks.com/pdfs/agile_and_UCD_MM.pdf.

[76] Grigori Melnik and Frank Maurer. Introducing agile methods in learning environments: Lessons learned. In Marchesi and Succi [71], pages 172–184.

[77] Thomas Memmel, Harald Reiterer, and Andreas Holzinger. Agile methods and visual specification in software development: A chance to ensure universal access. In Constantine Stephanidis, editor, *HCII Universal Access in HCI*, volume 4554 of *LNCS*, pages 453–462. Springer, 2007.

[78] Matthias M. Müller and Walter F. Tichy. Case study: Extreme programming in a university environment. In *23rd International Conference on Software Engineering*, pages 537–544. IEEE Computer Society, 2001.

[79] Peter M. Nardi. *Doing Survey Research: A Guide to Quantitative Methods.* Allyn & Bacon, 2005.

[80] J. Fernando Naveda, Kent Beck, Richard P. Gabriel, Jorge L. Díaz-Herrera, Watts S. Humphrey, Michael McCracken, and David West 0002. Extreme programming as a teaching process. In Wells and Williams [108], pages 239–239.

[81] Dan North. Whats in a story? DanNorth.net website (last visit : 2008.08.29), February, 11th 2007.

[82] A. N. Oppenheim. *Questionnaire design and attitude measurement / A.N. Oppenheim.* Heinemann, London :, 1968.

[83] Tim O'Reilly. What is web 2.0 - design patterns and business models for the next generation of software, September 2005.

[84] Mittermeir R., Bollin A., Hochmüller E., Jäger S., and Wakounig D. Ameise - an interactive environment to acquire project-management experience. In Eigenverlag Universität Klagenfurt, editor, *Methods and tools for development of semantic enabled systems and services for multimedia content, interoperability and reusability*, Proceedings of HUBUSKA Third Open Workshop, Klagenfurt, Austria, 27-28 April 2006, pages 79–92, 2006.

[85] Vinay Ramachandran and Anuja Shukla. Circle of life, spiral of death: Are xp teams following the essential practices? In Wells and Williams [108], pages 166–173.

[86] Valentin Razmov and Richard Anderson. Pedagogical techniques supported by the use of student devices in teaching software engineering. In *SIGCSE '06: Proceedings of the 37th SIGCSE technical symposium on Computer science education*, pages 344–348, New York, NY, USA, 2006. ACM.

[87] Patrick Reed. An agile classroom experience. In *AGILE 2008*, pages 478–483. IEEE, 2008.

[88] Donald J. Reifer. How good are agile methods? *IEEE Softw.*, 19(4):16–18, 2002.

[89] B. Rumpe and A. Schröder. Quantitative survey on extreme programming projects. *Extreme Programming and Agile Methods XP/Agile Universe 2002*, 2002.

[90] Per Runeson and Martin Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2):131–164, April 2009.

[91] O. Salo and P. Abrahamsson. Integrating agile software development and software process improvement: a longitudinal case study. *ISESE*, page 10 pp., 2005.

[92] O. Salo and P. Abrahamsson. Agile methods in european embedded software development organisations: a survey on the actual use and usefulness of extreme programming and scrum. *Software, IET*, 2(1):58 –64, February 2008.

[93] Outi Salo. *Enabling software process improvement in agile software development teams and organisations*. PhD thesis, University of Oulu, 2006.

[94] J. Schneider and L. Johnston. Extreme programming at universities - an educational perspective. In *ICSE '03: Proceedings of the 25th International Conference on Software Engineering*, pages 594–599, 2003.

[95] Jean-Guy Schneider and Lorraine Johnston. Extreme programming: Helpful or harmful in educating undergraduates? *Journal of Systems and Software*, 74(2):121–132, 2005.

[96] Sara Shahzad. Knowledge management issues in teaching extreme programming. In *9th International Conference on Knowledge Management and Knowledge Technologies, I-Know and I-Semantics, 2009.*, pages 278–288, September 2009.

[97] Sara Shahzad. Learning from experience: The analysis of an extreme programming process. In *Sixth International Conference on Information Technology: New Generations, 2009. ITNG '09.*, pages 1405–1410, April 2009.

[98] Sara Shahzad, Zahid Hussain, Martin Lechner, and Wolfgang Slany. Inside view of an extreme process. In Pekka Abrahamsson, Richard Baskerville, Kieran Conboy, Brian Fitzgerald, Lorraine Morgan, and Xiaofeng Wang, editors, *XP*, volume 9 of *Lecture Notes in Business Information Processing*, pages 226–227. Springer, 2008.

[99] Mary Shaw. Software engineering education: A roadmap. In *Proceedings of the 22 nd International Conference on Software Engineering, ACM*, pages 371–380. Press, 2000.

[100] A. Sillitti, M. Ceschi, B. Russo, and G. Succi. Managing uncertainty in requirements: a survey in documentation-driven and agile companies. pages 10 pp. –17, Sept. 2005.

[101] Bjornar Tessem. Experiences in learning xp practices: A qualitative study, xp2003. In Marchesi and Succi [71], pages 131–137.

[102] J Barrie Thompson. Software engineering practice and education: An international view. In *SEESE '08: Proceedings of the 2008 international workshop on Software Engineering in east and south Europe*, pages 95–102, New York, NY, USA, 2008. ACM.

[103] Louwarnoud van der Duim, Jesper Andersson, and Marco Sinnema. Good practices for educational software engineering projects. In *ICSE '07: Proceedings of the 29th international conference on Software Engineering*, pages 698–707, Washington, DC, USA, 2007. IEEE Computer Society.

[104] R. van Solingen, E. Berghout, R. Kusters, and J. Trienekens. From process improvement to people improvement: enabling learning in software development. *Information and Software Technology*, 42(14):965 – 971, 2000.

[105] Vasudeva Varma, Jocelyn Armarego, and Pankaj Jalote, editors. *Proceedings 22nd Conference on Software Engineering Education and Training, CSEET 2009, Hyderabad, India, 17-20 February. 2009*. IEEE Computer Society, 2009.

[106] Mario Žagar, Ivana Bosnić, and Marin Orlić. Enhancing software engineering education: a creative approach. In *SEESE '08: Proceedings of the 2008 international workshop on Software Engineering in east and south europe*, pages 51–58, New York, NY, USA, 2008. ACM.

[107] Mario Žagar, Ivana Bosnić, and Marin Orlić. Enhancing software engineering education: a creative approach. In *SEESE '08: Proceedings of the 2008 international workshop on Software Engineering in east and south europe*, pages 51–58, New York, NY, USA, 2008. ACM.

[108] Don Wells and Laurie A. Williams, editors. *Extreme Programming and Agile Methods - XP/Agile Universe 2002, Second XP Universe and First Agile Universe Conference Chicago, IL, USA, August 4-7, 2002, Proceedings*, volume 2418 of *Lecture Notes in Computer Science*. Springer, 2002.

[109] L. Williams and R. Upchurch. Extreme programming for software engineering education?, 2001.

[110] Laurie Williams, D. Scott McCrickard, Lucas Layman, and Khaled Hussein. Eleven guidelines for implementing pair programming in the classroom. In *AGILE '08: Proceedings of the Agile 2008*, pages 445–452, Washington, DC, USA, 2008. IEEE Computer Society.

[111] Laurie Williams, Sarah E. Smith, and Michael Rappa. Resources for agile software development in the software engineering course. In *Conference on Software Engineering Education and Training,*, pages 236–238, Los Alamitos, CA, USA, 2005. IEEE Computer Society.

[112] Peter Wolkerstorfer, Manfred Tscheligi, Reinhard Sefelin, Harald Milchrahm, Zahid Hussain, Martin Lechner, and Sara Shahzad. Probing an agile usability process. In *CHI '08: human factors in computing systems*, pages 2151–2158, New York, USA, 2008. ACM.

Deutsche Fassung:
Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008
Genehmigung des Senates am 1.12.2008

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Graz, am …… ……
(Unterschrift)

Englische Fassung:

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

…… ……
date (signature)