EMPIRICAL STUDY ON EXTREME PROGRAMMING

by

Sharifah Lailee Syed-Abdullah

A thesis submitted in partial fulfilment of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science
University of Sheffield
April 2005

ABSTRACT

The focus of this research is on the human side of an agile methodology because most of the other research on methodologies tends to focus on the technical aspect.

The context for this research is the Software Engineering Observatory at the University of Sheffield, a research facility, which is run by the Verification and Testing (VT) research group. The objective of this observatory is two-fold: firstly, it is to create an environment for the training and development of skills that are associated with the successful construction of a software solution with a real commercial client, and secondly, it is for the carrying out of research work that would be impossible to do in the real software industry. The observatory allows empirical researchers to observe, question or interview software developers working on real industrial projects.

The general relevance of this research lies in the ability to evaluate the effectiveness of an agile methodology by first, identifying the difficult practices in the XP methodology and the reasons for the difficulties, with the intention to improve the methodology. Cognitive theory indicates that for a new approach to be accepted easily, it must conform to the ways the brain accepts information, stimulates the mind, and thus motivates the developers.

The research demonstrates qualitatively and quantitatively the effect of this improvement on the software developers. Comparison studies between the Extreme Programming (an agile methodology) with the Discovery Method (a design-led methodology) were conducted to evaluate the effect of the XP methodology in term of the work related well being, the work group cohesion, the positive affectivity and finally the quality of the software. To achieve generalisability for some findings, data was collected from an XP team in IBM, Hursley, United Kingdom.

ACKNOWLEDGEMENTS

No research would have been possible without the assistance and willing cooperation of many individuals and groups. I am taking this opportunity to record my deepest gratitude to all individuals, directly or indirectly, who have contributed towards the completion of this thesis.

First, I would like to express my heartfelt gratitude to my supervisor Prof. Mike Holcombe for his expert guidance, confidence and stimulating suggestions throughout my research process. I would like also to express my profound appreciation to my research panel committee, Dr. Amanda Sharkey and Eur Ing Dr. Anthony Cowling, for taking the time out of their busy schedules to advise me on the progress of the research.

Thanks are also due to my employing institution, University Teknologi MARA, Malaysia, who has granted me the scholarship. Without the UiTM's commitment and generosity, it would not have been possible for me to embark on this research project.

I am greatly indebted to my parents, Syed-Abdullah Syed-Ali and Sharifah Kamariah Syed-Ahmad, my husband, Mohd Suhaimi Mohd Jusoh and six children, Diyana, Hazwan, Aqilah, Haseef, Syafiqah and Fathin. The research project seems to have a way of life for imposing sacrifices on my family.

I would like to express my appreciation to the members of the Verification and Testing research group for their support during the administration of the experimental procedure and the validation of the research work, especially to Dr. Marian Gheorge, Dr. Tony Simons, John Karn, Dr. Francisco Macias, Chris Thompson, Dr. Phil McMinn and Simon Coakley.

I would also like to thanks all of my friends, in particular, Dr. Rabiah Ahmad, Zainah Ibrahim, Dr. Jarita Duasa and Dita Sardjono for their kindness, help, support and warm friendship.

I am greatly indebted to Paul Miller for taking the time out of his busy schedule to edit the final draft of this thesis.

My sincere thanks go to all of the students in the Software Hut 2002-2004 and Genesys 2001-2003 and also the IBM team, who have kindly participated in this research.

My warm appreciations to the office staff and the support team in the Computer science department. Finally, my thanks to the reviewers of my work whom I have mentioned in the respective chapters.

LIST OF PUBLICATIONS

The progress of this research produced 8 papers, which provided the feedback to the research. These papers are:

1. Syed-Abdullah, S., Holcombe, M., and Gheorge, M. "Empirical Study on An Agile Methodology: An Action Research Approach," The 8th Annual Conference on UK Academy for Information Systems, PhD Consortium, University of Warwick, 2003, p. 245.
   This paper discusses the early introduction of an action research strategy into the study and reports the early result of the impact of the XP methodology on the SE teams.

2. Syed-Abdullah, S., Holcombe, M., and Gheorge, M. "Practice makes Perfect," in: *Lecture Notes in Computer Science LNCS 2675,* M. Marchesi and G. Succi (eds.), Springer-Verlag, Berlin Heidelberg, Germany, 2003, pp. 354-356.
   This paper presents the early evidence of the pattern of the relationship between the number of practices and the quality of software produced by the teams.

3. Syed-Abdullah, S., Holcombe, M., and Gheorge, M., "Agile Methodology in Practice" in the Research Memoranda CS-03-04. (2003)
   This paper presents the results of the early investigation into the relationship between the number of XP practices used and the quality of software developed, as perceived by the clients.

4. Syed-Abdullah, S., Holcombe, M., and Gheorge, M. "The Impact of an Agile Methodology on the Well being of Development Teams," *Empirical Software Engineering (to appear),* 2005.
   The paper presents the results of the first experiment on work related well being. The paper highlights the evidence of using the XP methodology in the most dynamic and in the stable work environment.

5. Syed-Abdullah, S. and Holcombe, M. "The Impact of the XP methodology on Work related Well being (to appear)," in: *Measuring Quality Requirements in Information Systems*, 2005.

   This chapter of a book presents the results of the first experiment and the replication made in the following year, on the impact of using the XP methodology on the work related well being.

6. Syed-Abdullah, S., Holcombe, M., Karn, J., Cowling, T., and Gheorge, M. "The Positive Affect of the XP methodology", in *the 6th International Conference on eXtreme Programming and Agile Processes in Software Engineering, XP 2005*, University of Sheffield, United Kingdom, 18-23 June 2005

   This paper describes a longitudinal study on how XP methodology acts as a positive affect inducer to the SE teams. The results show strong correlation between positive affectivity and the XP practices. This finding helps to provide empirical evidence that the XP methodology does have an impact on the positive affectivity of the developers

7. Cowling, A.J., Karn, J., Syed-Abdullah, S., and Holcombe, M. "Adjusting towards XP: An observational study of inexperienced developers" in *the 6th International Conference on eXtreme Programming and Agile Processes in Software Engineering, XP 2005*, University of Sheffield, United Kingdom, 18-23 June 2005

   This paper describes attempts by researchers from the University of Sheffield to introduce XP to relatively inexperienced student developers. Various research methods were used to ascertain student perceptions of XP, ranging from focus group interviews to ethnographic methods. This paper describes some of the important findings and provides evidence relating to common problems encountered when students attempt to adjust to XP.

8. Karn, J., Syed-Abdullah, S., Cowling, A.J., and Holcombe, M. "A study into the effects of personality type and methodology on cohesion in software engineering teams," *Journal of Business and Information Technology* 2005.

This paper describes work done to gain a qualitative understanding of how cohesiveness relates to personality type, performance and adherence to a particular methodology (XP).

The relation of these papers to the contribution of this thesis is presented in the following table:

Chapter	Title	Papers
3	Research methodology	Empirical Study on An Agile Methodology: An Action Research Approach
4	The Adoption of the XP methodology	Adjusting towards XP: An observational study of inexperienced developers
5	Positive Affect of the XP methodology	The Positive Affect of the XP methodology
6	XP and the Work Group Cohesion	A study into the effects of personality type and methodology on cohesion in software engineering teams,
7	The Impact of XP on Work related Well being	• The Impact of an Agile Methodology on the Well being of Development Teams *(to appear)*, *Empirical Software Engineering Journal* • The Impact of the XP methodology on Work related Well being *(to appear)*," in: *Measuring Quality Requirements in Information Systems (book section)*,
8	XP and the Software Quality	• "Practice makes Perfect," in: Lecture Notes in Computer Science LNCS 2675 • "Agile Methodology in Practice" in the Research Memoranda – CS-03-04

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1 Introduction

In recent years, efforts have been made to produce high quality software in dynamic environments. The problem of an ever changing requirement demands new approaches in developing software. The emergence of agile methodologies seems to be the answer to this phenomena and acceptance of this approach has grown with each year. Tools support has been developed and introduced to encourage the migration to this methodology. However, there is still resistance from several quarters, especially the development teams, in adopting this radical approach in software engineering. The purpose of this research was to investigate the reasons for these phenomena and to study the effects of an agile methodology, specifically the Extreme Programming (XP) methodology, on software development teams.

The focus of this research is on the human side of the methodology because most of the other research on methodologies tends to focus on the technical aspects. Studies reported by DeMarco and Lister (DeMarco et al. 1987) indicate that the overwhelming majority of failed projects are due to human problems;

> *There was not a single technological issue to explain the failure...the major problems of our work are not so much as technological as sociological in nature.*
>
> *The main reason we tend to focus on technology ...is not because it is crucial but it's easier to do....Human interactions are complicated and never very crisp and clean in their effects, but they matter more than any other aspect of the work. If you find yourself concentrating on the*

> *technology rather than the sociology, you're like the vaudeville character who loses his keys on a dark street and look for them on the adjacent street because, as he explains, "The light is better there." [p. 4-6]*

The context for this research is the Sheffield Software Engineering Observatory (SSEO) at the University of Sheffield, a research facility which is run by the Verification and Testing (VT) research group. One of the challenges in the SE curriculum is to make students more professionally aware. The curriculum at the University of Sheffield introduces students to a set of engineering practices that enable them to create sophisticated software systems for industrial clients. The curriculum provides an in-depth coverage of the management processes, software tooling, and design activities for software development. This is an effort by the University of Sheffield to create a professional SE environment within the university (Cowling 1994; Cowling 1997; Holcombe et al. 2003a). The objective of this observatory is two-fold: firstly, it is to create an environment for the training and development of skills that are associated with the successful construction of a software solution with a real commercial client, and secondly, it is for the carrying out of research work that would be impossible to do in the real software industry. The observatory allows empirical researchers to observe, question or interview software developers working on real industrial projects

The observatory consists of two programme modules: the Software Hut and Genesys Solutions. The Software Hut is a course module for the second year undergraduates, the students work on a project for a commercial client while several academics act as overseers to ensure that the students are working well together. This module has the expressed aim of fostering teamwork and communication skills and the students are strongly encouraged to take these aspects seriously. The second module consists of the fourth year in addition to the MSc students in the Department of Computer Science who run a professional software house known as Genesys Solutions. Genesys as the module will be refer to hereafter, was set up as a response to the challenges of trying to introduce the

entrepreneurial dimension as part of the higher learning curriculum. The concept is to allow the students to run the software house with minimum intervention from the lecturers, whose function is more as a consultant (Holcombe 2001).

XP was created in response to problem domains whose requirements changed, and to address the problem of the project risk. XP begins with 4 values; Communication, Simplicity, Feedback and Courage. It then builds up to the 12 practices that XP projects should follow. Many of the XP practices were created and tested as part of the Chrysler C3 project (Hendrickson 1999). Beck introduced XP as a solution to the problems encountered by the more formal methods (Beck 2000). The emphasis of this methodology is on improving communication between the developers and the managers, and is highlighted as one of the main values. The second value is simplicity. Complex requirements must be simplified to enhance understanding between all of the team members because a greater understanding promotes communication, even between the most erudite of scholars. The humanistic aspect of these two values aims to promote good teamwork, which in turn is an important ingredient towards developing quality software.

1.2 Research Goals

The general relevance of this research lies in the ability to evaluate the effectiveness of a methodology. The specific goals of this research are to:

Goal 1: Identify the difficulties in adopting the Extreme Programming practices and the reasons for these difficulties.

The first goal deals with the issue of identifying the difficult practices and reasons for the difficulties, with the intention to improve the methodology. To achieve this goal, the study used a qualitative approach; a focus group interview for data collection, and the innovation acceptance framework (Frambach et al. 2002) as a guide. The information gained was used to identify ways to improve the

presentation of the methodology and therefore the understanding of the specific practices.

Goal 2: Demonstrate whether the improvement made to the teaching of the Extreme Programming can be utilised to facilitate the understanding and application of the method.

The second goal of this research focusses on the improvements made to the teaching and coaching of the methodology to facilitate an understanding and thus, the application of this methodology. Cognitive theory indicates that for a new approach to be accepted easily, it must conform to the ways the brain accepts information, stimulates the mind, and thus motivates the developers. The research demonstrated qualitatively and quantitatively the effect of this improvement on the software developers.

Goal 3: Demonstrate the role of the Extreme Programming methodology as an inducer of positive affect.

The third goal deals with the role of the Extreme Programming methodology in manipulating the positive affect on the software engineering team members. Comparison studies between the Extreme Programming (an agile methodology) with the Discovery Method (a design-led methodology) were conducted to test this goal empirically.

Goal 4: Demonstrate whether the Extreme Programming practices can improve the work group cohesion amongst the members of a software engineering team.

The fourth goal of this research deals with the effect of the agile methodology in increasing the work group cohesion amongst members of the same software engineering project. Comparison studies between the Extreme Programming (an agile methodology) and the Discovery Method (a design-led methodology) were conducted to test this goal empirically.

Goal 5: Demonstrate whether the Extreme Programming practices can improve the work related wellbeing of a software developer.

The fifth goal of this research deals with the effect of the agile methodology in reducing the work related anxiety and depression; and in increasing the contentment and the feeling of enthusiasm in the dynamic software engineering projects. Comparison studies between the Extreme Programming and the Discovery Method (a design-led methodology) were conducted to test this goal empirically.

Goal 6: Determine whether the Extreme Programming methodology can be utilised to improve the quality of the software.

The focus of this goal is to conduct a comparison study to determine whether the XP methodology has the necessary techniques to improve the quality of the developed software.

1.3 Motivation and Solution

The significance of this research lies in the fundamental ability to understand and recognize the problems of adopting the 12 practices in Extreme Programming methodology. This research shows that in order for the software engineering team to effectively adopt these practices, improvement to the presentation of this methodology is pertinent. The significance of this improvement is two-fold: the evaluation of the effect of the methodology on the software developers and the effect of this improvement on the quality of the developed software. The evaluation of this methodology is conducted by testing several hypotheses about the effect of this methodology in comparison to a design-led methodology.

To address these rather imposing goals;

First, related work from the literature is detailed in order to discover a reasonable method to achieve the goals, to uncover pertinent research and to focus the investigation.

Second, the comparison and the longitudinal studies were conducted in the Sheffield Software Engineering Observatory (SSEO). To achieve generalizability for certain findings, data was collected from an XP team in IBM, Hursley, United Kingdom.

Third, for the purpose of validation, the triangulation of qualitative and quantitative approaches was used. In addition, a partial validation of several findings was made with data collected from industry.

1.4 Summary of the Remaining Chapters

The above investigation is reported in the following order:

Chapter 2 details the survey of related work. It provides background information necessary to understand the work.

Chapter 3 defines the detailed research methodology undertaken to complete this research. It describes the steps of the process and the rationales behind the decision taken to combine several well established research methodologies in software engineering, information science and social science. It details the comparison studies and the longitudinal study conducted throughout this research.

Chapter 4 discusses the qualitative findings of the research. First, the problems of adopting the 12 practices in the Extreme Programming (XP) are addressed and are followed by the recommended improvements to this methodology. Then it continues to discuss the theoretical reasons in cognitive science for these improvements. Evidence of the effect of this improvement is demonstrated qualitatively.

Chapters 5, 6, 7 and 8 present convincing quantitative findings of this research. Each chapter discusses the results of the comparison studies between an agile methodology and a design-led methodology. The results from the agile methodology are then associated with the number of practices adopted in order to discover possible causal effects with work related wellbeing, work group cohesion

and the quality of the developed software. This chapter also discusses the comparison between the final result with results from the early observations to establish the evidence for the improvement to both the research and the agile methodology.

Chapter 9 summarizes the contributions of this research and also suggests the directions for future research.

There is repetition of sentences in several chapters to facilitate the reading of this thesis.

CHAPTER 2
BACKGROUND

2.1 Introduction

This chapter provides a synopsis of the background information necessary to understand the work.

2.2 Agile Methodologies

The traditional methodologies imposed a disciplined process upon software development, with the aim of making software development more efficient in order to produce better quality systems. The detailed process places a strong emphasis on planning and was inspired by other engineering disciplines. The most frequent criticism of these methodologies is that they are bureaucratic. The several phases in the system development slow down the development process. The second problem with these methodologies is that the requirements specifications are not flexible. In reality, it is difficult to get the software customer to identify their requirements. Even if the requirements can be identified, the business world is forever changing.

As a reaction to these problems, a new group of methodologies evolved, these are known as **agile methodologies.** Agile methodologies welcome change and unpredictability. These new methodologies are more adaptive rather than predictive, and more people-oriented rather than process-oriented. Adaptive approaches are better when the requirements are uncertain or volatile. If the requirements are not stable, it is difficult to develop stable designs and follow a planned process as practised in the traditional methodologies. According to Fowler, executing an adaptive process is not easy because it requires a very effective team of developers. The team needs to be efficient in both the quality of the individuals, and the way they blend together (Fowler 2002). The few most notable agile

methodologies are Extreme Programming (XP) (Beck 2000), Dynamic Systems Development Method (DSDM) (Stapleton et al. 1997), SCRUM (Schwaber 2001), Feature Driven Design (FDD) (Coad et al. 1999), CrystalClear (Cockburn 2001; Cockburn 2004) , and Agile Modelling (Ambler 2002).

Agile Modelling (AM) is a practice-based methodology for effective modelling of software-based systems. AM is not a prescriptive process, that is, it is not a complete software process. AM focusses on a portion of the overall software process which is needed with other, full-fledged process such as XP, DSDM, SCRUM, or the Unified Process (UP). For XP projects, AM explicitly describes how to improve productivity through addition of modelling activities whereas with UP projects it describes how to streamline modelling and documentation efforts to improve productivity (Ambler 2002).

The **Crystal** methodology focuses on three properties Frequent Delivery, Reflective Improvement and Osmotic Communication because these properties are find in all of the software projects. Frequent delivery in this methodology refers to the deployment of software to the clients at the end of each iteration for the production use. If the team cannot deliver the system every month, user viewing becomes the critical alternative. Reflective improvement refers to the time taken by the team to think and reflect new strategies for improving each delivery. Reflective improvement mechanism allows the team to make adjustment to the change in the team member, technology and task. Osmotic communication refers to the information flows into the background hearing of the team members, so that they pick up relevant information as though by osmosis. This is accomplished by having the members in the same room (Cockburn 2004).

The **SCRUM** methodology assumes that the systems development process is an unpredictable, complicated process which is described as an overall progression. It defines the system development process as a loose set of activities that combines known, workable tools and techniques to develop software. The SCRUM approach is an enhancement of the commonly used iterative or incremental object-oriented

development cycle. The primary difference between the defined methodologies such as waterfall and spiral , and the SCRUM approach is that SCRUM assumes that the analysis, design and the development process are unpredictable. Operating in an unpredictable and complex environment requires management controls, to avoid falling into a chaos state. SCRUM uses the OO techniques to manage control and these controls are reviewed, modified and reconciled at every review meeting (Schwaber 1996; Schwaber 2001).

Feature Driven Design (FDD) was created by Peter Coad and Jeff De Luca, and combines the key advantages of agile methodologies with model-driven techniques that scale to the largest teams and projects. FDD is a design-oriented agile process, which follows the belief that a strong design (yet one that allows room for flexibility) will create a process that is better managed and thus more efficient. The project is divided into "features," which are small pieces of the project that possess some customer value. FDD creates design, code, and code inspection schedules that may seem strangely un-agile, but these schedules lack the depth and mounds of paperwork associated with a system completely specified in the requirements phase, instead relying on people and their roles to address the details as needed. The simplified design schedule also serves as a bridge of communication between manager and developer. Perhaps the most important part, the design schedule, can be used to establish a baseline for productivity and can be used to estimate future product development and serve as an important component of contract negotiation (Palmer et al. 2002).

The **Dynamic Systems Development Method (DSDM)** is about people, not tools. It is about truly understanding the needs of a business, delivering software solutions that work and delivering them as quickly and as cheaply as possible. This approach provides a framework of controls and best practice for Rapid Application Development. It was created by a consortium of organisations and it has been proved, since its publication in January 1995, to be extremely effective in delivering maintainable systems which match the needs of the business better than those produced using traditional lifecycles. DSDM uses an iterative process based

on prototyping and involves the users throughout the project life cycle. This approach achieves delivery to tight timescales through shortening communication lines between users and developers, between analysts and designers, between and across team members, and between differing levels of management. The mechanisms by which these communication lines are shortened differ from one to another. DSDM provides guidance on how to decide what sort of documentation it is necessary to control and why. DSDM is independent and can be used in unison with other frameworks and development approaches, for example with Prince2, RUP and eXtreme Programming (XP). The DSDM development process consists of 7 phases. The first one is before the project has officially started. Then there are the project studies, which in this document are considered to be one phase. Then there are three more phases that consist of iterative cycles, which are repeated as necessary to complete the project. Then there is the post-project phase, where the project is maintained. The project flow may move between the different phases in different directions (Stapleton 2002; Stapleton et al. 1997).

Extreme Programming Method

The XP methodology was created in response to problem domains whose requirements change and also to address the problem of project risk. XP begins with 4 values; *Communication, Simplicity, Feedback* and *Courage.* It then builds up to a dozen practices, which all XP projects should follow. Many of the XP practices were created and tested as part of the Chrysler C3 project. Beck (Beck 2000) introduces XP as a solution to the problems encountered by the formal methods. XP focuses on 4 humanistic values, these are **communication, simplicity, testing** and **courage,** and also how each of them are interrelated. XP does not arise out of nothing but it is an improvement on the existing methods.

XP embraces the notion that features, which provide the most business value to the customer, must be developed first because the real goal of this approach is to deliver the software that is needed when it is needed. Requirements are written as user stories, which are chunks of functionality which are valuable to the customers.

After the user stories are written, a release plan is created that specifies exactly which user stories are going to be implemented for each system release. These stories are translated into a programming task and tested during the various iteration phases. Iterative versions of the systems must be frequently released to the customer, in order to get feedback. It is important for the customer to give feedback immediately, so that the changes which are made can have an impact on the overall development. This is the core feature of XP, which allows it to adapt to any changes in the requirements

The **communication** between developer and manager, which can be lacking in other methods, is highlighted as one of the main values, which must be emphasized. XP encourages *communication* by having the developers *collectively owning* all of the code and *work in pairs*. ***Collective code ownership*** considers that the code belongs to the team and not to the individual developers. It encourages every developer to contribute new ideas to all segments of the project and allows any developer in the team to add functionality, correct errors or refactor the code. Gittins, Hope and Williams identify that the existence of the test procedures, which prevent poor code from entering the system, give the developers the confidence to allow other developers to modify the codes (Gittins et al. 2001).

Pair programming is a practice that requires two developers to sit side by side in front of a computer. One person types and thinks tactically about the methods being created, while the other thinks strategically about how the methods fit into the class. Each partner must explain what they are doing and this encourages the development of new ideas and an improvement on the old approaches. To facilitate a collective code ownership and continuous integration, pair programmers must swap partners amongst the team. Studies on pair programming have shown that this practice improves the quality of design and the implementation without a sacrifice in productivity (Lui et al. 2003; Succi et al. 2002; Williams 2002). Pair programming changes the environment from criticism and competition to learning and cooperation. These are some of the ingredients required for improving group cohesiveness.

The second value is **simplicity**. Complex requirements must be simplified to enhance an understanding between all of the team members because understanding promotes communication. XP *simple* design evolves through constant ***refactoring***, which is guided by suitable ***metaphor*** and implemented in accordance to common ***coding standards***. Refactoring is when redundancy is removed, unused functionality is eliminated and obsolete designs are rejuvenated (Wells 2001) and system metaphor is a story that everyone (customer, programmer and managers) can tell about how the system works (Beck 2000). The reason for using a metaphor is to achieve a common vision, a shared vocabulary, and generativity and architecture.

Coding standards keep the code consistent and make it easy for the entire team to read, understand and refactor (Beck 2000) but specific coding standards are not important as long as the standard supports collective code ownership (Jeffries 2001; Jeffries 2002). To provide *feedback* during the development, XP encourages the programmers to write ***unit tests*** before coding, because it helps the developers to really consider what needs to be done. Unit tests encourage immediate *feedback* because they reassure the developers that when all of the coding is completed, the unit tests are also completed. A developer creates one test to define some small aspect of the problem, followed by creating a simple code to pass the test. Then the developer creates a second test and new codes are added to the previous code to pass the new test. The ***testing*** and ***integration done incrementally and continuously*** allow a team to absorb unexpected changes, and the team's *courage* is revealed by how it responds to stress.

On-site customer is a practice which requires the customer to sit with the development team on a full-time basis. It is the customer's duty to assist in the writing of stories, to answer questions and to set priorities to the project. Holcombe (Holcombe 2002) is more realistic in this practice, by balancing between having customer on site full time, with one hardly there at all, and he proposes regular visits and meetings at both the development team site and the business site. In

reality, not all customers could afford to adhere to on-site customer practice. This is not because they are not serious but it is due to other managerial issues.

The humanistic aspect of the communication and the simplicity aspect promote good teamwork because it is an important ingredient towards developing quality software. A stable teamwork will facilitate continuous testing, which will enhance *courage* because members are confident of producing better and well-tested software. The satisfaction of producing quality software is very important because it will boost the confidence of the team to produce more challenging software. Beck only gives a glimpse of what XP can do to promote a better understanding and to improve the working environment amongst software development teams. Hendrickson suggested a few questions when evaluating whether a project adheres to XP or not. When judging a project status as Extreme, a person should look into the 4 values of the Extreme development process: communication, feedback, simplicity and courage, instead of focusing solely on the 12 practices (Hendrickson 2000). He further states that a team that truly follows the 12 practices has courage, implying that it is difficult to adopt the 12 practices in one attempt, which is the case with Secure Trading (Gittins et al. 2001). In this study, the authors mentioned that only selective adoption of XP practices is possible.

2.3 Discovery Method

The Discovery Method provides a completely guided approach to modern object-oriented and component-based software development. The method covers the part of the software lifecycle ranging from initial requirements elicitation, through task, object and subsystem identification, to detailed coding specifications. The method aims to support both expert and trainee software engineers; it seeks to ensure a guaranteed standard of quality in development, for the lowest cost. *Discovery* is mostly known for the way in which it directs the developer's attention using sharply-focused techniques which build upon each other in a clear and obvious way. A key feature is the exploitation of *discovery procedures*, self-reinforcing analytical techniques having a trigger, a feedback loop and a completion criterion.

A linked sequence of *discovery procedures* forms a *discovery path*, a development route leading from one modelling stage to the next.

The *Discovery Method* is informed by insights from Gestalt psychology, delaying the formation of fixed object concepts, which might introduce early bias into the models. It is a *transformational* method, in which analysis models evolve gradually into system designs with maximum cohesion and minimal coupling, and are integrated with existing libraries and application frameworks. It is a *selective* method in which techniques are chosen for their single focus and fitness-of-purpose in context. It is a *participatory* method in which the client is continually involved. The *Discovery Method* uses an adapted subset of UML notations, whose semantics have been clarified such that models have a formal underpinning and may be related to each other by transformation. The iterative, incremental and parallel development strategy may be described within the OPEN Process Specification.(Simons 1998; Simons 2003)

2.4 Empirical Software Engineering

Software engineering is concerned with the techniques to develop, run, and maintain application systems. Empirical software engineering is the sub discipline of software engineering that investigates software engineering techniques by using empirical methods, these are the case study, inquiry and experiment. Empirical software engineering has been pushed by internationally recognized research institutes such as the Software Engineering Institute (SEI), the Empirical Foundations of Computer Science (EfoCS), the Empirical Software Engineering Laboratory (ESEL), the Institute of Experimental Software Engineering (IESE), the Centre for Advanced Empirical Software Research (CAESAR), the Empirical Informatics Research (EIR), and NASA's Software Engineering Research Network (SEL) (Zendler 2001)

Experimental software engineering is the sub-discipline of empirical software engineering that uses experiments to validate, improve, and select software

engineering techniques, which permit an efficient application development. Zendler gave a brief history of the experimental software engineering research carried out since 1967. In this paper, Zendler identified that most of the published software engineering experiments adhere to three criteria: at least two software engineering techniques were studied, subjects (software developers) applied software engineering techniques and produced artefacts, and the artefacts were measured by software metrics. Most of these experiments were focused only on investigating analysis, design, implementation, a test and maintenance technique or quality assurance and reuse techniques. He suggested a need for new research, which included investigating the effect of formal versus non-formal methods on the quality of an application system (Zendler 2001).

2.5 Human Factors in the Software Engineering

In an engineering text, it is unusual to find a chapter on human factors. On the contrary, the business world has long recognized the important role humans play. Human are important in increasing productivity anywhere, including in the design of machines. In software engineering, there is a need for more research to understand the human factors because understanding them can help to identify the possible ways to increase productivity. One of the critical factors in software engineering is the productivity of the software developer. Therefore, a study of the human factors, specifically of the software developers, is important for several reasons. Two reasons are; firstly, effective managers must be able to understand their staff as individuals and how they interact as a group because computer systems are developed by people. Secondly, if the limitation and the ability of these developers are not taken into account when designing and developing new software engineering techniques, then these techniques will not be used in the best possible way.

Surveys by Lyytinen and friends (Lyytinen et al. 1987; Lyytinen et al. 1999) suggested that 50% of all systems development projects end in failure. Developing new products is a difficult process in any industry, but the software industry is

particularly notorious for delivering poor quality products that are late and over budget (Sheremata 2002). Further studies by Ewusi et al (Ewusi-Mensah et al. 1994) identify that often human issues play a substantial, if not the primary role, in these failures. Nevertheless, research on system development still suggests that the human issues are not properly addressed (Clegg et al. 1997; Clegg et al. 1994). Human aspects of system development are those issues, which have a discernible impact on the working practices and the environment of people who interact directly with the system.

The technical orientation of the systems development methodologies, such as SSADM, Yourdon and SSAD, gives very little recognition to human support beyond urging the developers to consult users during the requirement and design phases (Hornby et al. 1992). Methodologies which are more organisationally oriented such as ETHICS (Doherty et al. 1998) and Soft System Methodology (Checkland et al. 1990) are by comparison rarely used by IT professionals. In a survey on IT professionals with management responsibilities, it emerged that the majority of these professionals acknowledged the importance of human issues such as training, health, and motivation in the software development process. It also emerged from the survey, there is the need to prioritise the system development activity, so that effort is concentrated in those areas of greatest organisational importance; the necessity of acknowledging that as information systems operate in a highly dynamic environment, it is essential that the future needs are thoroughly examined, and translated into a technical specification that is sufficiently flexible; and also the importance of timing the system integration and implementation that is perceived as organisational disruption, so that greater acceptance can be achieved (Doherty et al. 1998).

The significance of the above findings is that it is the IT professionals who are starting to recognize the importance of the human issues but, due to the absence of the necessary tools, techniques and the rewards to address these issues, they have been led to concentrate more on the tangible technical issues such as transitional and system integration. As pointed out by studies (Doherty et al. 1998; Hornby et

al. 1992), the system development process is still primarily a technological driven process that rewards for delivering technically sound systems on time and to budget.

2.6 Cognitive Theory in the Software Engineering

The activity of software development is a cognitive skill and, like all such skills, is subject to the limitations of the human brain. There is a great diversity in individual abilities, reflecting differences in intelligence, education and experience, but all of these seem to be subject to some basic constraints regarding human thinking. These result from the way in which information is stored and modelled in the human brain. Although the distinction between information and knowledge is not rigid, a possible view is that neural information processing involves the integration of new and existing information to create knowledge. It is therefore worth looking at the human processing model to identify the effects this might have on software engineering techniques.

There is considerable evidence to suggest that the format of information influences learning (Ainsworth et al. 2003; Schnotz 2002; Verdi et al. 2002). The literature in the cognitive area also highlights the sequence for presenting both forms of the information. The studies have proven empirically that the conceptual diagram must be presented first and the text later for the simple reason that 'text never describes a subject matter with enough detail to allow only one kind of envisioning' (Schnotz 2002) and also because presenting the text first will cause interference with the picture presented afterward (Verdi et al. 2002). Previous research has shown that the more difficult a learning content is, the higher is the frequency of looking at an adjunct visual display (Schnotz 2002) and the supportive function of a visual display seems to be especially evident with learners of low prior knowledge (Hmelo-Silver et al. 2004) and low verbal skills (Carney et al. 2002; O'Donnell et al. 2002). Studies in software engineering have shown that practitioners are clearly of the view that adding structure to the software development process is beneficial. In practice, developers avoid cumbersome methods, which are more often time

consuming and costly but are willing to adopt flexible approaches such as those which are component-based, evolutionary or interactive in nature. The findings also showed the dominant use of several techniques reflecting the desire to use useful technique regardless of their original purpose (Benbasat et al. 1987; Clegg et al. 1997).

2.7 Work Group Cohesion

Most of the software engineering activities involve teamwork. Half of the software developer's time is spent in communicating with other team members or clients, while only a small percent is spent in working alone. Therefore, a successful team combination and effective communication are essential for increasing the productivity of a software development team. Studies in work related group cohesion identify that the member's personalities contribute to the success or failure of a software engineering project. Personalities can change depending on individual circumstances, the environment and the personalities of the co-workers. Work by Weinberg (1971) looked into the personality traits that are needed by the software engineers. These traits included the ability to be adaptive and also to withstand a certain amount of stress. These findings lead to the study of techniques to assist software engineers to be more adaptive and resilient in the ever-changing world of business demands and technology (Weinberg 1971).

Work group cohesion refers to the degree to which team members have close friends in their immediate work unit, the attraction to the members of the group and the satisfaction with the members of the group (Yoo et al. 2001). Group cohesion is an outcome of the group development process. According to Klein (1995), the increased task performance by cohesive groups is due to more frequent, less-inhibited, task-related communication. Cohesion has been proposed to be one of the important determinants of work group performance (Klein et al. 1995; Mullen et al. 1994).

Research into job-related diversity has discovered that factors such as age and gender have less impact than personality. Research into work group diversity has concluded that diversity can be a 'double-edged sword' specifically that it can result in a high-quality solution whilst at the same time decreasing the cohesion level (Webber et al. 2001). Previous research reported that organisational culture is considered as one of the potential contextual factors that influence group processes (O'Reilly et al. 1989). Organisational culture is a characteristic of an organisation, not of the individuals, which is rooted in the values, the beliefs, and the assumptions held by the organisational members and that individuals in a collective oriented organisation tend to encourage other members to categorize themselves as a unit (Chatman et al. 1998). In their study, Chuang and colleagues investigated and showed empirically that an organisation with a high culture value and a high emphasis on collectivistic values can promote the benefit of group diversity. The study also showed that the degree of intensity and the cultural content embedded within the members of the workplace has an impact on the work group functioning (Chuang et al. 2004).

2.8 Positive Affect and Negative Affect

The term 'affect' represents a broad category of affective processes, including emotional experiences, moods, and traits or dispositional affect. Emotions are brief states that involve cognitive, physiological, and behavioural processes that help individuals to respond quickly to threats or opportunities; they are relatively short and are directed at specific events or stimuli. Moods, in contrast, have a longer duration, lasting hours or days, and are less directly to focus on anything specific. Traits or dispositional affect, reflects stable individual differences in the tendency to experience and express certain emotions and moods. [For a review of the definitions see (Anderson et al. 2004)].

In a study of the role of affect on human life, Norman and colleagues (Norman et al. 2003) show that affect makes humans smart because affect is always passing judgments and presenting them with immediate information about the world. The

affective signals work through neurochemicals, bathing the relevant brain centres and changing the way humans perceive, decide, and react. These neurochemicals change the parameters of thought, adjusting such things as whether reason is primarily 'depth first' (focused, not easily distracted) or 'breadth first' (creative, out of the box thinking, but easily distractible). Affect came early in evolutionary history, preceding the evolution of humans and playing an essential role in survival. The fast-acting system helps people to navigate through life. Affect also has a major impact on how well people are able to perform tasks. Negative affect focusses the mind, leading to better concentration. In cases of an immediate threat this is good, for it concentrates processing power upon the danger. When creative problem solving is required this is bad, for it leads to narrow, tunnel vision. Positive affect broadens the thought processes, making it more easily distractible. When the problem requires focus, this is bad, but when the problem is best addressed through creative, out-of-the-box thinking, then this is precisely what is needed. Affect therefore regulates how a person solves problems and performs tasks. Negative affect can make it harder to do even easy tasks: positive affect can make it easier to do difficult tasks.

In another study, positive affectivity refers to an individual's disposition to be happy across time and situations (Watson et al. 1984); negative affectivity is an individual's disposition to experience discomfit across time and situations (Watson et al. 1987) and both of them are personality variables. Empirical evidence suggests that positive affectivity and negative affectivity might explain the variations in employees' job satisfaction. Works by Staw and colleagues have shown that variations in job satisfaction can be explained by an individual's dispositional affectivity (Staw et al. 1986; Staw et al. 1985). The authors have shown that employees who were predisposed to be happy (positive affectivity) are more likely to have a higher job satisfaction than those who are predisposed to experience discomfort (negative affectivity). The finding was validated by Brief *et a*l (Brief et al. 1988) , who argued that negative affectivity should be controlled in the studies of job attitudes.

People who find a positive meaning during adversity and keep a positive affect during ordinary events may have a greater ability to cope with adverse circumstances and bounce back quickly from them. Maintaining a positive affect during stress helps in coping with the adversities. Positive emotions create an upward spiral helping a person to see positive meaning in all events and circumstances. When a person feels everything that happens has a positive meaning, it is much easier to generate even more positive emotions. When thinking and attention are broadened by positive emotions, they begin to believe that there is an opportunity behind every adversity. Previous studies (Ashby et al. 1999; Carnevale et al. 1986; Isen 2001) indicate that in most circumstances positive affect enhances problem solving and decision making, leading to cognitive processing that is not only flexible, innovative, and creative, but also thorough and efficient.

Negative emotions create a downward spiral. Depressed, anxious or irritated mood saps the energy out of human. Negative mood dampens the spirit of the thought. As thinking narrows down, a person might not see solutions that might be right under their eyes. Humans begin to pay inadequate attention to the environment around them, therefore missing the helpful cues and details surrounding them. As a result, humans see options to be far more limited than they actually are. Pessimism and hopelessness might discourage people (Norman 2002) from taking appropriate and timely actions. The feedback from worsening circumstances, breakdown of relationships and deteriorating health, over time, creates a sense of failure and a distorted view of the future.

Research by Watson et al (Watson et al. 1984; Watson et al. 1987; Watson et al. 1985) indicates that emotional experience is dominated by two broad factors: Negative Affect (NA) and Positive Affect (PA). Both factors can be measured either as a state (short term mood fluctuations) or as a trait (stable and consistent individual differences in general affectivity level). The traits represent predispositions to experience the corresponding mood state factors. There is now overwhelming evidence that moderate fluctuations in feelings can systematically affect cognitive processing [for a review, refer to (Ashby et al. 1999; Isen 1993)].

Research by Isen and others have shown that daily positive affect experienced by people, improves creative problem solving (Isen et al. 1987), and cognitive organisation, flexibility and decision making tasks (Carnevale et al. 1986)

2.9 Work Related Well Being

Research has suggested that four factors have a significant effect on the well being of an employee and these are job design, performance monitoring, human resource practices and team leader's support (Frenkel et al. 1998; Knight et al. 1998). Studies on job design have demonstrated that control, variety and the demands placed on the employees are important predictors of well being. With regards to job control, a study has shown that high job control is positively associated with well being (Holman 2002) and job satisfaction (Batt et al. 1995). Findings by Holman showed that having high control over work methods and procedures, a low level of monitoring and a supportive manager, would appear to have the most significant effects on the employee's wellbeing. This finding is in line with other research, which found that a high level of monitoring has a negative association with wellbeing (Chalykoff et al. 1989).

2.10 Management Style

Management style is more than personality. It is also link with behaviour. Management style is essentially a dynamic process in a group whereby one individual influences the others to contribute voluntarily to the achievement of group tasks in a given situation (Cole 1996). Some of the management style theories are *authoritarian versus democratic*, *people versus task orientation*, and *contingency approaches*. Managers have a basic choice of being either authoritarian or democratic, even though, theoretically the best approach is democratic but the style depends on the other elements such as group, task and individual in the organisation. The main weakness of this approach is that it places too much emphasis on the manager's behaviour to the exclusive of the other elements. The people versus task orientation considers the effectiveness in dimension, where

situation is the consideration in which the leadership is exercised. The people orientation encourages employee to participate in decision making while the task orientation is more toward giving directive and task rather than people. The contingency approach is also known as action-centred management, incorporating the concern for task and for people. It stresses that the management style has to be related to the overall situation and therefore, has to be adaptive.

A mechanistic and a hierarchical style of management was needed when the experience with a product concept was high, while a looser style of management and more autonomous structures were employed when experience was low (Olson et al. 1995). Projects where the style of management fits the newness of the project resulted in products that were higher in quality than those projects whose style of managing was too bureaucratic or too informal. This study indicates the importance of determining the style of management when coordinating and supervising the team activities. Previous studies has shown that excessive monitoring may have an opposite effect on performance to the one intended (Holman 2002) because excessive monitoring may, over the long term, cause the employees to devote their cognitive resources to dealing with their anxiety, rather than focusing on providing better services (Kuhl 1992).

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Introduction

This chapter discusses the methodological approach of the research. The second section addresses the philosophy of the two research methodologies – qualitative and quantitative, used in this research. The third section discusses the strategies applied to accomplish the goals and the research model developed to guide the researcher when conducting the experiments. Section four highlights the set up of the Sheffield Software Engineering Observatory (SSEO) while section five discusses the four methods used to collect data and the different analysis required for each method. The final section discusses the validity and reliability issues addressed in this research.

3.2 Research Methodology

There are two methodological approaches in this research: quantitative, which is often associated with positivism, and qualitative, which is associated with interpretative research.

3.2.1 Quantitative methodology

Literature by (Aczel 1996; Black 1999; Blaikie 2003; Pallant 2001) were the key references drawn to outline the quantitative approach of this research. Quantitative methods are generally designed to collect data in a form suitable for statistical analysis and should be objective, non-reactive, representative and collected using standard measures (Stone et al. 1984). Data are collected mainly through a) social surveys which have the capacity for generating quantifiable data on large numbers of people who are known to be representative of a wider population in order to test theories or hypotheses; b) experiments which involve at least two groups – an

experimental and a control group; c) the analysis of previously collected data; d) structured observation where the researcher records observations in accordance with a predetermined schedule and quantifies the resulting data and e) content analysis by performing quantitative analyses of the content of media ((Bryman 1992).

For this research, the quantitative data was collected through surveys of the Genesys Solution students, experiments on the Software Hut students and the analysis of the previous performance documents.

3.2.2 Qualitative methodology

Literature by (Avison et al. 1988; Denzin et al. 2000; Gallier 1991; Greenbaum 1998; Kemmis et al. 2000; Patton 2002) were the key references drawn on to define the qualitative approach of this research. Qualitative research established itself as a method of inquiry for the study of human and group life. It involves the use and collection of a variety of empirical materials- case studies; personal experience; cultural texts and products that describe routine and problematic moments and meaning in individuals' lives (Denzin et al. 2000). Denzin and Lincoln stated that qualitative researchers deploy a wide range of interconnected interpretive practices, hoping always to get a better understanding of the subject matter at hand. The word ***qualitative*** implies an emphasis on the **qualities *of entities*** and on ***the process*** and ***meanings*** that are not experimentally examined or measured in terms of quantity, amount, intensity or frequency (Denzin et al. 2000). The principle advantage of using qualitative methods is that they force the researcher to delve into the complexity of the problem rather than abstracting from it, thus resulting in a richer and more informative discovery.

Denzin and Lincoln (2000) suggested:

> *Qualitative research is inherently multimethod in focus (Brewer & Hunter, 1989). However, the use of multiple methods, or triangulation, reflects an attempt to secure an in-depth understanding of the phenomenon in question. Objective reality*

can never be captured. Triangulation is not a tool or a strategy of validation, but an alternative to validation (Denzin, 1989a, 1989b, p.244; Fielding & Fielding, 1986, p. 33; Flick, 1992 p. 194). The combination of multiple methods, empirical materials, perspectives and observers in a single study is best understood, then, as a strategy that add rigor, breadth, and depth to any investigation (see Flick, 1992, p. 194) [(Denzin et al. 2000) p.2].

Qualitative methodology refers to the approach of studying the social world in order to describe and analyse the culture and behaviour of humans and their groups from the point of view those being studied ((Bryman 1992). Qualitative research requires that a) researchers collect data directly at a particular setting through observation or interviewing b) data is collected in the form of words or picture such as interview transcripts, field notes, memos and official documents c) data tends to be analysed inductively d) the main concern is with a person's perceptions of their life (Fraenkel et al. 1993).

3.3 Research Design

During the UK Academy for Information Systems (UKAIS), PhD Consortium, the advice of Prof. Philip Powell from University of Bath and Dr. Roger Beresford from University of Portsmouth were sought to verify the early concept of the research design especially the use of an action research strategy. The philosophy and paradigm adopted for this research was positivism and interpretism, which is in accordance with the most popular and most accepted approach taken by researchers in information system research. The strategies used in this research are the action research strategy and the experimental strategy.

3.3.1 Action Research Strategy

Action research is an established strategy in social and medical sciences which is gaining popularity in the scholarly investigation of information systems. It involves research intervention in real life contexts in order to improve contexts and at the

same time generate relevant scientific knowledge. The improved characteristic of action research is based on the belief that a researcher's positive intervention fosters involvement, cooperation and information exchange with the organisation members, which in turn leads the researcher to understand better the context being observed. Action research aims to contribute both to the practical concerns of people in an immediate problematic situation and to the goals of social science by joint collaboration within a mutually acceptable ethical framework [Rapoport, 1970] [reproduced in (Avison et al. 1999)]

An action research strategy was adopted primarily for the following reasons:

Cogenerative inquiry

Both the researcher and the participants generate the inquiry through collaborative communication processes in which all participants' contributions were taken seriously (Greenwood et al. 2000). The knowledge and experiences gained by the students in developing real life software projects contribute to the inquiry into the Extreme Programming methodology. The responses received from the students gave the researcher an insight into the problems faced in adopting this radically new methodology approach by novice developers. In turn the researcher helped in coaching (Beck 2000; Fowler 2002) the students to improve the understanding of the 12 practices. It is important to remember that in action research, intervention is not considered as violation of validity but rather the flexibility of the strategy (Davison et al. 2000; Gallier 1991). The experience gained during the pilot study showed that it was a necessity for the researcher to act as an advisor to the managers and as a coach to the students.

Cyclic Learning Process

The aim of an action research is to learn from experiences, seek solutions and apply that solution to bring about changes (Dick 2001; Greenwood et al. 2000). By adopting this strategy, experiences from the Sheffield Software Engineering Observatory (SSEO) have given an insight to the problems faced by the XP teams.

Researchers in this field discovered that experience is more valuable, if a researcher enters the experiences with expectation because he or she will be on the look out for unmet expectations and therefore maximises the learning process (Dick 2001). Experiences by other researchers (Gittins et al. 2001; Johansen et al. 2003; Macias et al. 2003) help the researcher to focus on certain XP practices. Reflection plays an important role in the cyclic learning process of action research. Reflections were made after interviews were conducted with the students. The analyses of the reflections are necessary to understand the problems faced by the students, and to make suggestions to facilitate the learning process.

Produce valid empirical result

In order to produce evidence on certain issues any intervention made must be explicitly linked to the issues at hand. This is achieved through an action research strategy because interviews and reflections enable the researcher to identify issues-related problems arising amongst the students. Interventions were made to ensure that the XP teams adhered as closely as possible to the 12 practices. By addressing these issues early in the research, it was hope that the study would produce valid empirical results.

Context centred

An action strategy focusses on solving real problems, and so the central inquiry processes of action research are linked to solving the practical problem in specific locations (Greenwood et al. 2000). Since the objective of this research was to understand the effects of XP methodology on students, the focus was centred on students using XP methodology to develop software projects with real clients.

3.3.2 Experimental Strategy

This is a research technique that allows researchers to determine how selected variables (independent variables) influence an outcome (dependent variable). Researchers use experimental design to make judgments about causes and effects.

In order to achieve the goals discussed in the first chapter, this strategy was also used. An experiment is a form of empirical study where the researcher has the control over some of the conditions in which the study takes place and control over the independent variables being studied; an operation carried out under controlled conditions in order to test a hypothesis against observation (Basili et al. 1999). Basili and colleagues acknowledged that experimentation in software engineering is difficult because of the complexity of the process, the time required to set it up and most importantly theories in software engineering are human based, thus the variation in human ability tends to obscure the effects. Therefore the credibility and validity of a study depends on how conclusions are drawn. The difficulties in the software engineering experimentation make it less likely that the validity types can all be satisfied at the same time. Still researchers are challenged to design the best study that the circumstances make possible, trying to rule out all the alternative explanations of the results and to generalize those results to the setting of interest. Although the 'perfect' study is difficult to obtain, a researcher must report the study in such a way that others can verify the findings.

Fenton and Pfleeger (Fenton et al. 1997) suggest six steps for carrying out a formal experiment. First, stating the objective of the study. Second, translate the objective into hypothesis. Third, set up the subject and environment. Fourth, execute the experiment. Fifth, conduct the analysis to ensure that the data is valid and useful before statistical testing is carried out. The final step is to document the experimental material and conclude the findings. Proper documentation is important to allow replication of the experiment. There has been a growing awareness amongst the empirical software engineering community of the importance of replicating studies (Basili et al. 1986; Basili et al. 1999; Deligiannis et al. 2002; Shull et al. 2004). In conducting the comparison study between the two methodologies, experimental strategy was used and the steps described by Fenton and Pfleeger were adhered to as closely as possible. Replication in software engineering domain is defined more broadly to include replication that allows details of the experiment to change so that certain threats to validity are addressed and also replication that varies variables (dependent variable) intrinsic to the focus

of the evaluation. Basili and colleagues identified 3 types of replication: 1) Replications that do not vary any research hypothesis, 2) Replications that vary the research hypotheses and 3) Replications that extend the theory (Basili et al. 1999). The replications carried out during the second and third year of this research conform to the first and second type of replication.

Experiments were set up with a minimum of two groups, one group, called the control group, was given a design-based methodology by the researcher. The other group, called the experimental group, was given an XP treatment by the researcher. Everything else about the two groups was as similar as possible.

3.3.3 Research Model

To provide a working definition of the relationships between the various strategies, a research model (Figure 3-1) of this research was constructed to give a clear picture of the components that the research examined.

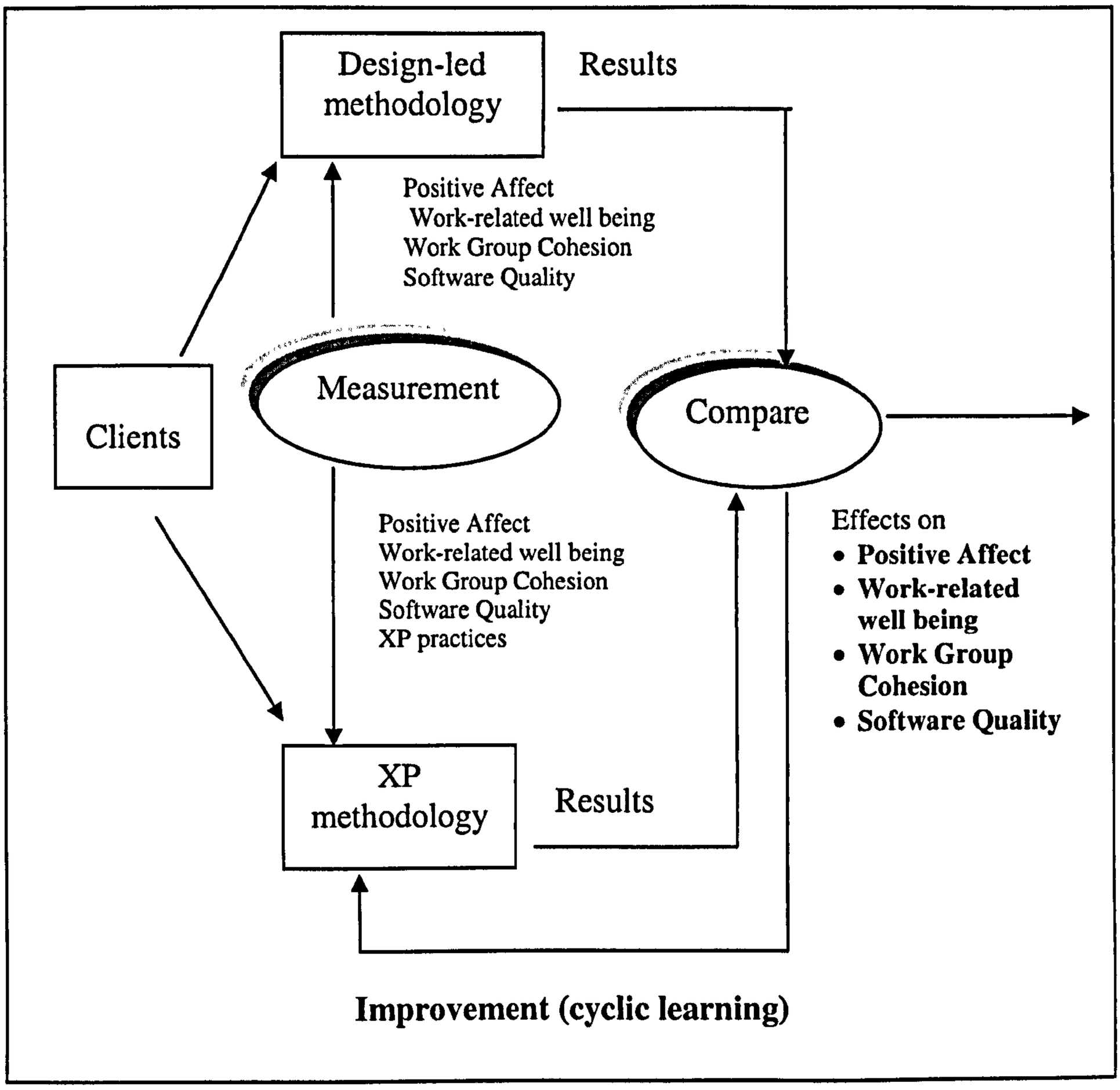


Figure 3-1 The Research Model

3.4 Subject

Throughout the course of this research, data was collected from two sources. The first two sets of data were collected from the Sheffield Software Engineering Observatory (SSEO) and the third set was from an XP team in IBM at Hursley Park, United Kingdom. Everyone connected with the SSEO, students, staff and researchers agreed to sign a non-disclosure agreement at the start of the project. This is essentially a guarantee that no information about specific projects can be passed on to any parties outside the SSEO. As a result of this agreement only the names of a selection of recent projects can be given.

3.4.1 Sheffield Software Engineering Observatory (SSE Observatory)

The objective of the Sheffield Software Engineering Observatory (SSEO) is twofold. First, is to create an environment for training and developing skills that are associated with the successful construction of a software solution with a real commercial client and secondly is for the research works that are impossible to carry out in the real software industry(Cowling 1994; Cowling 1997; Holcombe et al. 2003a).

Group Formation

In the SSEO, the students were encouraged to choose their own software development team members. The formation of these groups was a combination of 'natural' groups and 'constructed' groups. Natural groups consist of members who have been attending similar modules before attending the SSEO modules or members living in the same residential hall. The 'constructed' groups were teams which have fewer team members and required the lecturers' assistance to form a complete team. Through developing software together, communication and interaction amongst the team members were promoted and improved at an early stage. This aspect of team formation was important to observe, for the study of cohesion and communication between team members.

The SSE Observatory consists of two modules:

3.4.1.1 Software Hut

The Software Hut module consists of the 2nd year undergraduate students from Computer Science and Engineering degrees, and 3rd year students from Mathematics and Computer Science degree. The computer science students were required to complete all the subjects in Level 1 and the first semester of Level 2 subjects before enrolling in the Software Hut class. The subjects which are related

to the Software Hut projects are ***Introduction to Programming, Requirements Engineering, Object Oriented Programming*** and ***System Design and Testing*** modules in the Level 1 and the related subjects in the Level 2 are ***Functional Programming, Systems Analysis and Design*** and ***Database Technology*** modules.

During week 1, each client was required to give a brief introduction of their company and its requirement, to the students. After the briefing, the students were given a week to compare skills amongst team members with the skills needed to build a solution for each client. After the students made their evaluations based on their strengths and weakness, their choices were submitted to the lecturers. Final decisions regarding clients-students teaming were made by the lecturers to ensure equal distribution. The teaming was made as close as possible to the request to ensure that the students were satisfied with the decisions.

Every week, each team was required to deliver minutes of the team's meetings and the timesheet. After the 4th week, each team was required to submit the *Requirements* document. At the end of the semester, the teams were required to submit working software, software documents and a self-report document. The self-report document was a document where the team members described their experiences developing software with a real client using the respective methodology.

3.4.1.1.1 Software Hut 2002

Setting

In the Software Hut 2002 class, students were divided into teams consisting of 4 to 5 members. With a total population of 96 students, there were 20 teams during this study. Teams 1 through 10 were required to use the XP methodology to develop their software, while teams 11 through 20 were allowed to use any of the traditional software methodologies. During the second week, all of the students were given a team management course to familiarise themselves with managing a group project. Throughout the semester, which consists of 12 weeks, the XP students were

introduced to the XP methodology. The first lecture was on an overview of XP approach and was given during a 2 hours lecture. The XP students were informed of the XP book (Holcombe 2002) available online on the Genesys Solution intranet, document templates available in the Genesys internal website and other references about XP practices on various XP websites. The XP teams were also introduced to the practice of pair programming during a lab session at the beginning of the semester. While the XP teams were introduced to the XP methodology, the other students were reviewing the techniques used in the traditional methodology. The lectures were conducted as parallel sessions but the topic on Testing was given to both teams as a combined session.

At the beginning of the semester, each team was required to select a client. Each client was from a local business or organisation. For this session, four projects selected by the lecturers were Small Firms Enterprise Development Initiative (SFEDI), School of Dentistry, University of Industry, and National Cancer Screening Service (Table 3-1).

Project	Design-based teams	XP teams
SFEDI	18,20	5, 7, 8
Dentistry	12, 14, 17	2, 6
UFI	11, 13, 19	1, 9
NHS	15, 16	3, 4, 10

Table 3-1 Distribution of team and project for Software Hut 2002

Weaknesses of the Experiment

There were several weaknesses identified in this study. First, there was the absence of any coaching for the XP teams, thus the reasons for the team members' failure to adhere to most of the XP practices. Second, there was a lack of proper monitoring on the Design-based teams thus analysis of the project documents produced by these teams reflected that the teams were using 'unholy' amalgamation of

techniques to develop the software. Third, the assignment of methodologies was not at random and this might have contributed to the most organized teams being assigned to XP methodology while the less organized teams were assigned the design-based methodology. The Mann-Whitney statistical test conducted showed a significant difference in the average previous marks for Design-based teams (M = 50.90, SD = 5.49) and XP teams (M = 56.50, SD =56.50; $z(20) = 1.98$, $p = 0.048$), confirming the statement above. With these weaknesses discovered, the improvements were made before the second study commenced.

3.4.1.1.2 Software Hut 2003

Improvements made to the Experiment

During the second study, the management introduced several improvements to overcome the weaknesses mentioned in the section above. First, the assignment of the methodologies was changed, with the odd numbered teams assigned to use Design-based methodology and the even numbered teams were required to use the XP methodology. The Mann-Whitney statistical test conducted showed no significant difference in the average previous marks between the Discovery teams (M = 59.11, SD = 7.441) and the XP teams (M = 58.75, SD = 6.042). Second, the coaching of doing pair programming and developing test cases was introduced to the XP teams. Third, the decision was made to use and to monitor only one design-based approach, Discovery method (Simons 1998). This is a design-based approach developed by a lecturer in this university. Fourth, after observing the difficulties faced by the XP teams, the management decided to allow *partial adoption* of the XP practices. The decision was made to enable the students to focus more on the selected practices that were identified as essential to the development of better quality software.

The XP development process was carried out in 2-ways: *full adoption* and *partial adoption. Full adoption* referred to the decision by the managers to implement all of the 12 of practices and *partial adoption* referred to applying only part of the practices as shown by the table (Table 3-2).

No	Essential	Partial	Optional
1	Pair programming	Planning game	Frequent release
2	Testing	On-site customer	Refactoring
3	Coding standard	Continuous integration	
4	Collective ownership		
5	System metaphor		
6	Simple design		
7	15-hours		

Table 3-2 Partial Adoption of the XP practices

Setting

In the Software Hut 2003 class, students were divided into teams consisting of 4 to 5 members. With a total population of 80 students, there were 16 teams during this study. The even-numbered teams were required to use the XP methodology to develop their software, while odd-numbered teams were requested to use Discovery methodology. During the first week, all of the students were required to partake in a management game to break the ice amongst the team members. The XP teams were introduced to the methodology during the next 4 weeks and were also informed of the references available in the internet. In addition to lectures, the XP teams were coached on doing pair programming, developing story cards and testing before coding during three lab sessions. While the XP teams were introduced to the XP practices, the other team members were revising the Discovery technique during the lecture and lab sessions. The lectures were conducted as parallel sessions.

Project	Discovery team	XP team
Domestic Violence	1, 11, 17	6, 12, 14
Pharmaco	3, 5, 7	4, 8, 16
Control Engineering	9, 13, 15	2, 10

Table 3-3 Distribution of team and project for Software Hut 2003

For this session, three projects selected by the lecturers were Domestic Violence Unit, Pharmaco and Control Engieering (Table 3-3). Contrary to the previous year, the students were asked to submit their choices of client after the briefing.

Weaknesses of the Study

There were several weaknesses identified in this study. First, the XP teams were more focused to use the practices that were categorised as essential and there were weak attempts to use the rest of the practices. Second, since this was the first semester that coaching was introduced; therefore the coaches lacked the experience needed to coach properly. Third, the number of XP teams for each project was not balanced, two projects having 3 teams while another project having only 2 teams. This caused a problem during the statistical data analysis. Fourth, even though the XP approach emphasised early coding, the review and the releases of the project and project documents for the XP teams were the same as the Discovery teams. The schedule did not encourage the XP teams to code early and therefore the benefit of getting early feedback from the clients was lost in the process.

3.4.1.1.3 Software Hut 2004

Improvements made to the Experiment

During the final study, the weaknesses identified in the previous study were rectified. First, the assignment of the team-methodology was more random. Second, the XP teams were required to apply *full adoption* of the 12 practices. Third, the coaching focused on the pair programming, test-first design and the application of various practices. Fourth, to enhance understanding, the structure and the relationship of the practices were illustrated, in graphical forms during the XP lectures. Fifth, the use of an X-machine diagram was introduced to the XP teams. Sixth, to assist the students to follow the schedule effectively, separate handouts were given to the Discovery teams and XP teams.

Setting

In the Software Hut Spring 2004 class, students were divided into teams consisting of 4 to 6 members. With a total population of 60 students, there were 12 teams during this study. The distribution of team to methodology was made randomly as shown in the Table 3-4.

Project	Discovery teams	XP teams
Fizzilink	2, 7	3, 5
Dating	4, 11	6, 9
Debt Collection	1, 8	10, 12

Table 3-4 Distribution of team and project for Software Hut 2004

3.4.1.2 Genesys Solutions

Genesys Solutions is a student-run software company. The company was set up as a response to the challenges of trying to introduce the entrepreneurial dimension as part of the higher learning curriculum. The concept is to allow the fourth year MEng and advanced MSc students to run the software house with minimum

intervention from the lecturers, whose functions are more as a consultant (Holcombe et al. 2003b). The students are usually divided into a **Marketing** team, whose main function was to promote the company through public liaison with the clients and improving the Genesys website, a **Research and Development (R&D)** team, whose main task was to ensure that all hardware and software application required by the development teams are available, and finally the **Software development** teams, which made up the rest of the company. The main income for the company came from the projects, which was developed by these teams. In this company, all of the development teams were required to use the XP methodology where possible.

The following table lists some of the projects developed and maintained by the teams throughout this research (Table 3-5).

Genesys Year	New Projects	Maintenance Projects
2001-2002	• Smartcare • Field Labs • Medi Link	• Nursing School
2002-2003	• Sport Science • Ted Hancock • ELTC	• Nursing School • Field Labs
2003-2004	• Doncaster Violence Working Party • Pay4 Text • I-Sky	• Ted Hancock • ELTC

Table 3-5 List of some of the projects developed or maintained in the Genesys Solution Company.

3.4.1.3 Advantages of conducting study in the SSE Observatory

There were several advantages when the research was conducted in this environment:

The first advantage was the ability to carry out a longitudinal study where the researcher was able to follow the life cycle of a project (Ted Hancock project). The ability to study the development of this project using the XP methodology and then conduct a partial participative observation during the maintenance of this project enabled the researcher to deduce and make recommendations to introduce diagrammatic structure to the design practice in the XP methodology. Participative observation referred to the participation of a researcher in the process studied. The decision to conduct a participative observation was made when the team that inherited the maintenance project encountered problems in understanding the whole system and the interconnected program modules. Realising that the problem was due to the non-existence of an overview diagram that connected the various story cards, database and program code, literature review was made on cognitive theory. This leads to the proposal of including diagrammatic structure into the XP design practice. Since a PhD colleague was researching on the Extreme X-machine diagram (XXM) (Thomson et al. 2003), a spin-off from the current X-machine model (Holcombe 2000), there was a concensus agreement to test this technique in the study. The weaknesses of introducing the Extreme X-machine (XXM) in this study was realised when the students found it difficult to comprehend the structure without any additional literature on the subject. The discussions between the researchers and the explanations to the XP teams were intensified to fill the gap in the knowledge about the diagram. In addition, a decision was made to allow the teams to use the alternative X-machine, which has more literature on it. The reasons for choosing this diagrammatic structure and other improvements are discussed in Chapter 4.

The second advantage of conducting studies in SSEO was the ability to observe the difference in acceptance level of the XP practices amongst the developers. The longitudinal study of the students when they were in Software Hut 2001, where they were first introduced to the XP methodology and when they enrolled in the Genesys 2003, provides the researcher with an enriched experience in the user acceptance of this methodology across time. This insight provided the researcher

with the necessary information to recommend improvements for teaching the XP methodology as discussed in Chapter 4

3.4.2 IBM in Hursley Park, United Kingdom

The third set of data was collected from a team that applied XP practices to develop software in the organisation. The data collected was qualitative and quantitative. Due to the small size of the sample, to make any significance difference on its own, the quantitative data was not tested in this study.

3.5 Data Collection and Analysis

For this research, four methods of data collection were collected to add rigour, breadth and depth to the study. The discussion in this section includes the analysis of each technique used.

3.5.1 Focus group interview

The focus group approach was used in this study because team members develop the software and the existence of team interactions helped to release inhibitions amongst the team members, to activate forgotten detail of experiences and also to generate better data through wide ranges of responses. Focus group interviews were also chosen because it was the most appropriate method to study attitudes and experiences; to explore how opinions were constructed (Kitzinger 1995) and to understanding behaviours, values and feelings, (Patton 2002) . The early interview sessions were conducted to get a glimpse of the team members understanding of the new methodology. The difficulties encountered during the early interview sessions resulted in the development of an XP Activities table (Appendix 3-A). The use of the XP practices was measured through the focus group interview, analyses of the project documents and the participative observations. Evidence was documented quantitatively using the XP Activities table (Syed-Abdullah et al. 2003b).

The table was used to explain further about XP practices and the feedback from these interviews were analysed to improve it. The cyclic learning process (Greenwood et al. 2000) helped the researcher and the students to understand the XP practices as it was practised in the SSEO. The second interviews were conducted at the end of the project. For this interview, the XP Activities table was used as a guide to assist the students in identifying the relevant XP activities applied by the team. In comparison, the time taken to complete the second interview was shorter and the discussions were more precise. Open-ended questions were used as a guide for the interview sessions (APPENDIX 3-B).

Content Analysis

The challenges of analyzing the interview data lies in making sense of the massive amount of data, which involved reducing the volume of this information by identifying significant patterns, and constructing a framework to communicate the essence of what the data revealed (Denzin et al. 2000). The interview data was transformed into transcripts, and organised according to the pattern that emerged during the analysis. The XP Activities table was developed from the analytical insight and interpretation of this pattern. This method of organising and analysing the interview data was recommended in the qualitative study research (Denzin et al. 2000; Patton 2002). The *Atlas.ti* package was used to help in the coding of the interview text and linking of these codes to the semantic networks **(the sequence of this process is shown in APPENDIX 3-G)**. Decisions were made from these transcripts, vague at first then increasingly explicit and grounded (Strauss et al. 1990). The meanings emerging from the data had to be validated for their plausibility, sturdiness and confirmability. There were two levels of interview conducted with every batch of students. The purpose of the second interview was to do a follow-up and clarifying interview due to the gaps or ambiguities discovered during the analyses of the data. Social scientists (Miles et al. 1994; Patton 2002) acknowledge that data collection and analysis in qualitative inquiry are integrative, iterative, synergistic and interactive in nature. Due to the voluminous repetitive data collected using this approach, a new strategy was devised by distributing the XP

Activities table during the latter interview sessions in order to focus the discussion according to the existing pattern.

3.5.2 Documents

There were 2 types of documents used for this research.

3.5.2.1 Project Documents

The project documents include story cards, requirement document, user guide and the program codes. Software project documents were amongst the most important sources of data in this research and the analyses of these documents were used to help in the validation of the information gathered during the interview sessions.

Content Analysis

The information in these documents was analysed through content analysis and used to quantify the use of the XP practices. The availability of the self-report documents and minutes of the meetings helped the researcher to understand better the inter-relationship between the members, the problems faced and solutions agreed by the groups in applying the XP practices.

3.5.2.2 Assessment Documents

There were two types of assessment document used for calculating the performance level of each team member in a development team. The first document was past grades achieved in the other modules and the second document was the assessment documents produced by the clients (APPENDIX 8-A) and the lecturers (APPENDIX 8-B and 8-C). The teams performance level for the past and the current semester was derived from these documents.

Statistical Analysis

Statistical methods such as Independent t-test, Mann-Whitney test and Spearman Rank Order Correlation were used to analyse the documents.

3.5.3 Questionnaires

Questionnaires were used in this research because they facilitated the collection and analysis of mass data quickly and avoided bias when interpreting the qualitative data. Initially, the questionnaires were administered through the internet but due to the poor response rate (10% only), the approach was changed to face-to-face. Even though this approach was time consuming, the response rate increased to 85-95%.

Statistical Analysis

The reliability of a scale indicates how free it is from the random error. Two frequently used indicators of a scale's reliability are test-retest reliability and internal consistency. For this research, internal reliability indicator was used because the three scales were designed to measure the fluctuation of mood over a period of time the software was developed. Internal consistency refers to the degree to which the items that made up the scale are all measuring the same underlying attribute. Internal consistency can be measured in several ways but for this research, the most common Cronbach's coefficient alpha was used.

3.5.3.1 Work related Well being.

Work related anxiety, contentment, depression and enthusiasm were measured using the 12–items anxiety-contentment and depression-enthusiasm scale developed and improved by Prof. Peter Warr (Warr 1990) who is currently working with the Institute of Work Psychology at the University of Sheffield. Early discussion about the scale was made with the author himself and later, an in-depth discussion on the application and the analyses of this scale was made with Carolyn Axtell, a senior researcher from the same department. The scale was used to measure the extent to which software team members were anxious or contented, depressed or enthusiastic about their project across time and project. Respondents

were asked to think of the past few weeks and indicate the extent to which they felt gloomy, calm, uneasy, enthusiastic, cheerful, worried, contented, tense, depressed, optimistic, relaxed and miserable. The validity and reliability of this scale have been demonstrated in other studies (Axtell et al. 2002; Sevastos et al. 1992). Measurement was on a 5-point scale, and the respondents were asked to indicate the degree of agreement and disagreement with each item. Responses ranged from 1 "very slightly" or "not at all" to 5 "extremely" (APPENDIX 3-C). The 12-items were grouped into four categories: anxiety, contentment, depression and enthusiasm (APPENDIX 3-D).

3.5.3.2 Positive Affectivity (PANAS)

The Positive Affectivity of the Positive and Negative affectivity (PANAS) scale was used to measure the positive affect of the 2 methodologies on the students. Positive affectivity was measured using the 10 items used to assess the degree to which an individual was predisposed to be happy across time and situation (positive affectivity). The scale was developed and improved by Watson and colleagues (Watson et al. 1984). The validity and reliability of this scale has been demonstrated in other studies (Watson et al. 1997; Watson et al. 1987; Watson et al. 1985). Measurement was on a 5-point scale, and the respondents were required to indicate the degree of agreement and disagreement with each item. Responses range from 1 "very slightly" or "not all" to 5 "very" or "extremely". (APPENDIX 3-E)

3.5.3.3 Work group Cohesion

The scale used for measuring work group cohesion was developed by Price and Mueller (Price et al. 1986). Work group cohesion was measured using the 9 items used to assess the degree of cohesiveness amongst the members of the working team. The validity and reliability of this scale has been demonstrated in other study (Agho et al. 1992). Measurement was on a 5-point scale, and the respondents were asked to indicate the degree of agreement and disagreement with each item.

Responses range from 1 "very slightly" or "not all" to 5 "very" or "extremely". (APPENDIX 3-F)

3.5.4 Observation.

Short-term observation was used during the study. Observation was made of the pair programming process and the management style used through out this research. The observation was made during the client's meeting, the management meeting and also when the team met to start a project. Each observation lasted between 10 to 20 minutes and was made randomly. In addition to the short-term observation, participative observation was also made in the Genesys solution environment. The decision was made to include participative observation when the team that inherited a maintenance project encountered problems in understanding the whole system and the interconnected program modules.

3.6 Statistical Method

Literature by (Aczel 1996; Black 1999; Blaikie 2003) and (Pallant 2001) were the key references drawn on to define the quantitative analysis of this research. Throughout the process of analysing the data, assistance was sought and obtained from a colleague to check the administration of the statistical procedures. She is an academic who has been actively involved in the social science research for several years. She has recently completed her PhD research at the University of Sheffield.

One of the most difficult part of the research process is choosing the correct statistical technique to analyse the data. In choosing the right technique, several factors must be addressed. These include:

The types of research questions being addressed. In this research, most of the analyses are on exploring the relationship between variables (such as the relationship between the number of the XP practices applied in the project and the level of different emotions experienced by the SE teams) and exploring the differences between the groups (such as differences between two methodologies)

The type of items and scales that are included in the questionnaire to address the above questions. In this research, the scales chosen to measure the different emotions were validated by other researchers. The scores and the dependency of the variables are discussed in the specific section.

Nature of the data. The quantitative data collected in this study is categorical (such as the types of methodology, rating of the project, and the project number) and continuous (such as total score for wellbeing and positive affect)

The parametric or the non parametric technique. The decision to use parametric test is made if the sample is small (less than 30) and the non-parametric test is used when the sample size is large (30 and more). Unfortunately, non-parametrics techniques tend to be less 'powerful', that is they may not detect differences or relationships, even when they actually exists.

The statistical analysis methods chosen for this research are

1. **Pearson Correlation** is a parametric test for exploring the *strength of the relationship between two continuous variables* and is used when all of the teams are grouped as XP and Design only. This is because the *sample size is large* and the nature of the data is *categorical and continuous*
2. **Spearman's Rank Order Correlation** is the non parametric version of the Pearson Correlation test. This test is use because when the data is grouped into the different types of project, the size of the data is reduced to less than 30 for each project.
3. **T-tests** are used when there is the need to compare the mean score of the continuous data of two groups (XP and Design), or two sets of data (before treatment and after treatment). There are two types of T-test:

   a. **Independent sample t-test**

   This test is used to compare the *continuous* data (such as the mean score of the different measures in the wellbeing scales) for *two different groups* (such as XP and Design).

b. Paired sample t-test

This test is used to compare *continuous data* on *two different occasions* (before treatment and after treatment).

The above tests are used when the comparison is only between methodologies because the *sample size is large* enough to use a parametric test. In the event of a smaller sample size such as when the data is further split into different types of project, a non parametric test must be used

4. **Mann-Whitney test** is the non parametric version of the independent sample t-test.
5. **Wilcoxon Signed Rank** is the non parametric version of the paired sample t-test.
6. **Mixed between within ANOVA** is used when analyzing the variance between the different groups (between methodologies) and the variability within each of the groups (within the different time interval or treatments). This test is used because it informs the researcher of the interaction effect of the week, and the methodologies in this study. In addition, the SPSS package generates a line graph that allows the researcher to inspect visually the relationship between the different variables

3.7 Special Measurement Concerns

In this research, the study of software development methodology in action was by short-term observation, interviewing and collecting questionnaires from people. Using these approaches, several special issues were encountered:

1. Protecting the anonymity of the subject. This was achieved through coding the name of the respondents. Anonymity of the clients was achieved by not disclosing the full name of the company and discussing the projects.
2. Minimizing the interference with the development process was achieved through

- Coaching sessions where the researcher made sure that the teams adhered to as many XP practices as possible and at the same time advised the students on developing better quality software and producing the required documents.

- Observing the pair programming process from a distance of a few metres from the target group.

3.8 Validity and Reliability

Literature from (Denzin et al. 2000; Erlandson et al. 1993; Lincoln et al. 1985) were the key references drawn on to define the steps taken to ensure the validity and reliability of this research. In qualitative research, the trustworthiness of the investigation can be addressed by using 4 criteria: credibility, transferability, dependability and confirmability. Karlstrom (Karlstrom 2002) provided the following comparisons between qualitative and quantitative criteria in addressing the various validity issues.

3.8.1 Credibility

Credibility, which corresponds to the internal validity in quantitative research, aims to ensure that the subject of the inquiry was accurately identified and described. Credibility was assessed by determining whether the description developed through the inquiry in the SSEO setting was acceptable to the members of the empirical software engineering domain. Lincoln outlined five strategies to accomplish credibility: a) *activities* – prolonged engagement, persistent observation and triangulation b) *peer debriefing* c) *negative case analysis* d) *referential adequacy* and e) *member checking* (Lincoln et al. 1985).

Activities

Prolonged engagement refers to sufficient time being spent in the research setting in order to understand the activities of the respondent. This is to ensure that the researcher can gain understanding of the setting, build the trust of the respondents and also to overcome distortion and biases by the researcher herself.

During the course of this research, the researcher attended meetings between the development teams and the management team, examined the existing documents for the maintenance project and coached the XP practices. By examining the legacy software documents, the researcher gained an insight into the problems faced by the teams inheriting XP documents and through coaching, trust and rapport were built between the researcher and the teams. This helped to identify the solutions inherent to the various problems faced by them.

Participative observations contribute further to the credibility of the study by "..understanding the events that occur and its relationship that exist in the real context the same way that they are understood by a person who is part of the context" (Erlandson et al. 1993). The relevance of the study to the social context was obtained through the use of the SSEO where the teams dealt with the real clients requesting projects that were applicable in the respective business world. The decision to stop collecting data was made when the analysis of the quantitative data yielded the same result and the examination of the qualitative data only showed redundant answers to the unstructured questions.

Triangulation is often used to elicit the various and divergent constructions of reality that exist within the research setting by collecting information about different events and the relationships from different point of view (Erlandson et al. 1993). For this research, triangulation was achieved by collecting both qualitative and quantitative data from the Software Hut and Genesys Solution students.

Peer Debriefing

Peer debriefing is a process of exploring the inquiry that otherwise might only implicitly remain within the inquirer's mind (Lincoln and Guba, 1985). There is a need to review perceptions and to gain insights and analyses of professionals outside the research setting.

In this research, peer debriefing was done by discussing specific research work with fellow researchers conducting studies in the SSEO and various people in the

Institute of Work Psychology, IE Testing, IBM Hursley and others in academic and the IT industry met during conferences or workshops attended by the researcher. Some of these professionals are mentioned in the respective chapters of this thesis.

3.8.1.1 Negative Case Analysis

Negative case analysis is a process of revising hypotheses with the objective of constantly refining the hypothesis until it accounts for all known cases (Lincoln et al. 1985). The present research did not engage in this strategy because of the time scales demanded by this analysis.

3.8.1.2 Referential Adequacy

Referential adequacy refers to the need to archive all data in order to enable the testing of the result and to facilitate future data analyses and interpretation of this research. Even though this strategy has its advantages, but it also has some drawbacks regarding personal and confidential data. To accomplish this strategy and at the same time maintain data confidentiality the names of the subjects were coded.

3.8.1.3 Member checking

Member checking is a strategy to increase the level of credibility in the findings. In relation to this research, the findings from the interview data were discussed with members from another team without mentioning specific the team. This activity was carried out during the data collection phase when the members were asked about their experience in understanding and using the XP methodology. During this activity, controversial points raised by the previous members were discussed to discover the resistance pattern amongst the members and also to discover the strategy used by the teams in overcoming those issues. Member checking was also achieved through discussion with fellow researchers conducting short term observation on the SE teams in the SSEO.

3.8.2 Transferability

Transferability, which corresponds to the external validity in quantitative research, addresses how far outside the observed domain the results are applicable. In quantitative research, transferability is possible because the respondents are selected and can represent the general population (Erlandson et al. 1993). To achieve transferability in this research, data was collected from the industry (an XP team in IBM, United Kingdom) and the findings were compared with those obtained from the SSEO.

3.8.3 Dependability

Dependability refers to the reliability in quantitative research, which addresses whether the process of the study produces the same results, independent of time, researcher and methods. Techniques such as credibility principles, triangulation of research methods, replication of steps and inquiry audit are often use to address this issue (Lincoln et al. 1985). To add credibility to this research, triangulation of qualitative and quantitative methods, and replication of experiments were used.

3.8.4 Confirmability.

The last criterion for validity and reliability is the confirmability. Confirmability corresponds to the objectivity in quantitative research, addresses the issues of researcher bias and ensures that the researcher affects the results as little as possible. On the contrary, for this research where the action research approach was used, confirmability referred to the "attempts to establish trust among the subject is the confirmability of the data themselves" (Erlandson et al. 1993).

3.9 Conclusion

The present chapter has presented the methods and strategies used to accomplish the goals set in Chapter 1. The experiences gained in establishing the research model helped the researcher to understand better the different strategies and the

weaknesses of the studies conducted. This enable the researcher to suggest the various improvements as discussed in subsection 3.4.1.

CHAPTER 4

THE ADOPTION OF THE XP METHODOLOGY

4.1 Introduction

The term *innovation* is an interactive process initiated by the perception of a new market or new service opportunity for a technology-based invention which leads to the development, production and market tasks striving for the commercial success of the invention (Garcia et al. 2002). Garcia *et al* further stressed that an invention does not become an innovation until it is processed through the production and the marketing tasks, and finally diffused into the marketplace.

This chapter discusses the elements required for the XP methodology to diffuse, and how the students in the SSEO and an XP team in the IBM UK adapted the methodology to suit the purpose of their projects.

The longitudinal studies into the diffusion of this methodology in a controlled environment (SSEO) were carried out in several stages: the exploratory study (Genesys 2001 and Software Hut 2002) and the follow-up studies which saw several improvements made and the effects of these improvements on the next batch of students (Genesys 2002-2003 and the Software Hut 2003-2004). The exploratory study identified several weaknesses (as discussed in Chapter 3) and a modified experiment was repeated on the Software Hut 2003 teams, following the conceptual framework of "innovation acceptance in an organization" developed by Frambach and colleague (Frambach et al. 2002) (APPENDIX 4-A and 4-B). The advice of Dr. Ruud Frambach, was sought and obtained to understand and to apply the framework in this study. Steps were taken to ensure that the external influences such as organizational facilitators and social usage were allowed to influence the members' attitude towards the XP practices. Since this study used an action research approach, the steps taken to influence the teams' attitude were not

considered as intervention but as a flexibility of the strategy to ensure that the students have equal chance of understanding the new methodology. The variables for the organizational facilitators were lectures (education), coaching (training) and the managers and the coachers (organizational technical support), whereas for the peer usage, the information about the XP team in the IBM UK and the success stories of the previous XP teams were discussed. Following the observations from the second study, further improvements were made and the experiment was repeated on the Software Hut 2004 teams. The study of the XP team in IBM, UK was carried out for several months to add validity to the findings in the SSEO.

The study was important because it revealed how implementation barriers blocked the methodology from successful implementation. The study discovered several aspect of the software methodology, such as knowledge barrier and sociological factors, as some of the features that need to be addressed

4.2 Diffusion of an Innovation

Diffusion is a process by which an innovation is *communicated* through certain channels over time amongst the members of a social system (Rogers 1995). *Communication* is a process in which participants create and share information about new ideas with one another to reach a mutual understanding. The newness in the idea means that some degree of *uncertainty* is involved in diffusion. Uncertainty is the degree to which a number of alternatives are perceived with respect to the occurrence of an event and the relative probability of these alternatives. Uncertainty implies a lack of predictability, of structure and of information.

Diffusion is a kind of social change, defined as the process by which alteration occurs in the structure and function of a social system. When new ideas are invented, diffused, and are adopted or rejected, leading to certain consequences, social change occurs. Diffusion in this context refers to both planned and spontaneous spread of new ideas. The successful adoption of a new technology implies the successful diffusion of an innovation by the people of the organisations.

Four main elements in the diffusion of innovations are:

An innovation

Innovation is an idea, practice, or object that was perceived as new by an individual or other unit of adoption. Newness in an innovation need not just involve new knowledge but can include persuasion or a decision to adopt. Technological innovation such the XP approach created one kind of uncertainty about its expected consequences in the mind of potential adopters, as well as representing an opportunity for reduced uncertainty in another sense. Technological innovation usually has at least some degree of benefit for its potential adopters. This advantage is not always very clear-cut, at least not to the intended adopters. They are seldom certain that an innovation represents a superior alternative to the previous practice that it might replace (Rogers 1995). As defined by Garcia et al (Garcia et al. 2002) product innovativeness is a measure of potential discontinuity of a product which can generate in the marketing and technological process. It is the capacity of a new innovation to influence the firms' existing technological resources, skill and knowledge and to create a paradigm shift in the software engineering and technological domain; and the market structure in the industry.

Communication channels

As defined by Rogers (Rogers 1995), communication is a process in which participants create and share information about new ideas with one another to reach mutual understanding. The diffusion process is the information exchange through which individuals communicate new ideas through communication channels either through mass media channels or interpersonal channels. Mass media channels are channels involving radio, television and newspaper whereas the interpersonal channel refers to face-to-face exchange of information. Even though mass media enables information to reach many, interpersonal channels are often more effective in persuading individuals to accept new ideas. This is so because individuals do not evaluate innovations solely on scientific studies but also on subjective evaluation by relevant individuals.

Time

According to Rogers (Rogers 1995), the time dimensions is involved in the diffusion process through *innovation-decision process*, *the innovativeness of the adopter* and the *innovation rate of adoption.* Innovation-decision process refers to the process phase of early knowledge of the innovation to forming an attitude towards it, follow by the decision to reject or adopt, subsequently the implementation, and the confirmation of the innovation. The innovation-decision period is the time required to pass through these stages. Innovativeness of an adopter refers to the degree to which an adopter is relatively early in adopting new ideas than the other members of the system and thus an adopter can be classified as an innovators, an early adopter, early majority, later majority or a laggard. The measure of innovativeness and the classification of the adopters are based on the relative time at which an innovation is adopted. The innovation rate of adoption is the relative speed an innovation is adopt. The rate is often represented by the S-shaped curve but the slopes vary from one innovation to another. These variations are often determined by the perceived advantages as discussed in the later part of this chapter.

Social system

A social system is defined as a set of interrelated units that are engaged in joint problem-solving to accomplish a common goal (Rogers 1995). In this element, issues that affect the diffusion process are the system's social structure, the effect of norm on diffusion, the roles of opinion leaders and change agents, types of innovation-decisions made and the consequences of the innovation.

4.3 Factors Affecting the Diffusion of an XP methodology in the SSEO

According to Urban and Hauser (Urban et al. 1993), the key notion to the success of new products is to understand the voice of the users in terms of perceived needs and to establish a relationship between the customer inputs and how the products

are designed, produced and improved. The aim of this study was to accumulate the views of software students on the ease of use and the difficulties in using the XP practices for developing software. For this study, only a small set of variables was selected because the aim was to demonstrate the change in the effects of the variables rather than to provide an exhaustive set of variables on the methodology.

Several categories of variables may influence the adoption and diffusion of an innovation. These variables are *innovation characteristics*, *adopter characteristics*, *internal environment characteristics* and *external environment characteristics* (Frambach et al. 1998; Frambach et al. 2002; Patterson 1996; Waarts et al. 2002). For the purpose of this research, the focus was more on the innovation characteristics. To illustrate the findings, quotations from the interview transcripts and the observation notes were used. These quotes represent the dominant theme of the responses received and observation made.

The other three variables; adopter characteristics, internal and external environments were not included in the discussion because the SSEO, as a company set up for the main purpose of research, has different characteristics than the normal company and therefore does not experience the direct effect of the internal and external environment as other existing software houses in the industry do.

4.3.1 Innovation Characteristics

Innovation characteristics are the ideas about the value of an innovation in terms of the advantages and disadvantages compared to the existing methodologies. Rogers (Rogers 1995) stated that the five perceived innovation characteristics are:

4.3.1.1 Perceived Advantages and Disadvantages

This is the degree to which an innovation is perceived as better than the idea it supersedes. The greater the perceived relative advantage of an innovation, the more rapid its rate of adoption may be. The degree of the relative advantage may be measured in economic terms such as faster development, less maintenance and cost

saving (Chen 2003), strategic advantage and prestige are also important factors to consider.

After analysing the first batch of interview data (Software Hut 2002), it was discovered that the students were more enthusiastic in using XP methodology because of the perceived ease of use, faster development time and the prestigious Extreme name. Further analysis of the interview transcripts and the observation notes disclosed that the enthusiastic environment was created during the lectures given and management process of the project.

Observations made on the second batch of the interview data (Software Hut 2003) showed a different understanding of the new methodology. There was a lack of motivation amongst the XP teams in using the new approach. The lectures given to this batch of students failed to create the enthusiastic feeling amongst the students and this may contribute to the lack of satisfaction in being chosen to do the methodology.

Interview sessions conducted with the Software Hut 2004, showed a mixed feeling between enthusiasm and lack of satisfaction amongst the students.

4.3.1.2 Compatibility

This is the degree to which an innovation is perceived as being consistent with the existing technologies and past experiences of potential adopters. Suppliers commonly prefer to develop new technologies that have a good fit with important characteristics of their first target market and then the product extensions and improvements are further made to fit the needs of the later groups of adopter

Analysis of the 12 practices adopted by the teams in the Software Hut 2002 and Genesys 2001 revealed that the teams readily adopted practices that are descendent from a previous methodology or other proven management practices. These practices included coding standards, continuous integration and testing.

Coding standard

It was observed that once a specific coding standard was identified by the organization, the students readily accepted and incorporated the coding standard into their codes. The students found this practice easy to follow because adhering to programming standard has been instilled in the students during the earlier programming and software engineering courses.

> *we **decided** in term of commenting in C. general comment convention, ...We pretty much stick to the general convention.*

Frequent release/ review

It was impossible to do frequent releases because the scale of the software projects was small and the duration of the development phase was short, therefore the students adapted it to frequent reviews.

> *By showing him, we are able to identify our progress and also to show/ identify where we go wrong as well. It is good to be able to discuss and get him to say yes.*

Continuous integration

The next practice that was readily accepted and applied by the teams was the *continuous integration.* During the sixties, early studies of integration in organizational management proposed early and frequent integration of effort amongst the subsystem such as Marketing and R&D in order to increase efficiency and speed in accomplishing tasks. Research suggests that organizations can identify problems early by receiving information faster or receiving a greater range of information. Later, studies on software development have found indications of similar effects. Researchers later focus on early and frequent integration of code to help deliver software projects on time, improve quality and decreases defects in the codes. (For a review refer to (Sheremata 2002)). Continuous integration is made possible by introducing the students to CVS, the Concurrent Versions System, the dominant open-source network-transparent version control system. The system is useful for students because it has client-server access methods that allowed the students to access the latest code from anywhere with an internet connection. It also

has the unreserved check-out model to version control which helped the students to avoid artificial conflicts when the members started to load the different version of the software being developed.

> *Yes, we integrated constantly using the CVS.*
>
> *It (CVS) is good. We used it to manage and integrated the program. It allows us to do without complex and it allow you to monitor changes. ...We have previous experience with CVS and Eclipses last year in graphic. I was promoting it from the beginning. So it just takes off in Genesys in our case.*
>
> *Designing incrementally helps us to build a better solution because we knew more after that. We discovered that we couldn't build something less than the existing modules Designing is good because we can visualize it but when we get to code it, we can actually feel and see how they work and show it to the client.*
>
> *When we started coding quite early, it is quite good because it gives us an idea of what to do. If we find that the design has a problem, it gives us the chances to go back and modify it.*

Testing

> ***The more experience I get writing tests, the more I discover I can write tests for non-functional requirements- like performance or adherence of code to standards.*** *Kent Beck [p.45, (Beck 2000)]*

Testing was another practice that the teams accepted 'easily'. Holcombe et al (Holcombe et al. 2001) stated that the principle purpose of testing is to detect and remove faults in a software system. Two activities identified for the testing practice were ***test-first programming*** and ***frequent testing***. Test-first programming refers to the necessity for programmers to build a set of test cases and run them before any code is written. Even though test-first programming is a practice taken from an engineering field (Andrews 1999), practising test-first programming in a software environment proved to be a difficult practice to follow. Some of comments by the students were:

I didn't like it because when we have to write the trivial thing, it takes a lot longer to write the test than to write the code.

I find it better to write the test afterwards... because then we know what exactly we are doing, how the test be tested and pick up bits where you know your code have fall out.

You cannot really know what the tests are for until you sit and write the code.

But you can't be right about the function until you have written the function. We like to test after we write the code but we write the test because that is XP.

Observing this difficulty, the management decided to introduce a computational modelling X-machine to the students to facilitate the development of test cases before coding begins. Holcombe et al (Holcombe et al. 2001) defined the X-machine model as a simple and elegant way of visualising the dynamics of a software system. The model identifies the set of events and inputs that produce observable change; these are mouse clicks, data entry, sensor inputs etc. and the observable outputs such as screen displays, commands to peripheral devices etc. The key function is the business processes that are then integrated into a state-based machine model, the stream X-machine. This provides a mechanism for building extremely powerful test strategies (Holcombe et al. 1998).

The test-file table [p.85-86, (Holcombe 2002)], part of the X-machine modelling was introduced to the students in Genesys 2001 and Software Hut 2002. This table helped the teams in doing the test-first programming. Later, after observing that the students do not fully utilise the table, the management decided to include coaching as part of the training approach for the students and a more encouraging result on XP practices was observed. Coaching does indeed improve the understanding and usage of this table and other practices such as pair programming.

We write the function header then we write the test for it. We write the cases before we write the function.

We have it before writing the program, we edited the test detail, we write the blank detail, we run it and it fails.

We do the enforcing of the test first coding amongst the team members since XP glorify the test first coding. We have to make sure that people stick to it.

When we write the test first before it make us think about the function.

Observations made also indicated that difficulties in doing test-first programming was due to the inexperience of the students when building a new project. This accounts for the facts that the Genesys students were able to produce more test cases and adopt partial test-first programming than students in Software Hut. Observations made on students in Genesys indicated that the experience in practising test-first programming in previous projects (when the students were in the Software Hut class module) albeit partially also helped them to generate the test cases first before the code.

The second activity in the testing practice which the teams did was frequent testing. Frequent testing is the act of testing the code often to ensure that the code achieves what it is meant to do. Frequent testing was easily accepted because this practice was instilled in the students during the previous programming class modules.

4.3.1.3 Complexity

Complexity refers to the degree to which an innovation is perceived as difficult to understand and use. In the early study in Genesys 2001and Software Hut 2002, it was discovered that the students found it difficult to understand the 12 practices because of the inter-relationship of each practice with each other. Therefore reducing the complexity of the relationship is critical to the understanding of the methodology. This was achieved through improving the graphic presentation and incorporating the X-machine model into the planning game.

Improving the Presentation of XP practices

In order to achieve better understanding of these practices, a new presentation of the methodology and its practices was made to the students in the Software Hut 2004. The new presentation included illustrating the XP process in a diagrammatic form (APPENDIX 4-D) and categorising the practices (APPENDIX 4-C). After the introduction, it was discovered that the students understood the different practices easily. There was a change in the samples of quotes taken from students before and after the new presentation was made. Below are the samples of quotes from the students in the SSEO, before introducing the new presentation:

> *At the start we didn't know what we were doing because there is no structure but we figure it out.*
>
> *We have conflicting information to begin with. With people telling us XXM and story cards and others telling us to start programming without story cards.*

The following are the samples of quotes from the students in the Observatory, after the management introduced the table and process flow during Software Hut 2004:

> *Yeah, the basic is easy (to understand). It is easier than UML. It seems to be easy to apply to the project*
>
> *The planning game (documentation) makes it simple because we didn't really know how to do the planning game.*

Incorporating X-Machine diagram into the planning game.

Observation on the students in 2001 and 2002 revealed that the students found it difficult to write and apply the story cards effectively because they need a breadth-first approach. Lack of experience hinders the students from developing and producing complete story cards that can be useful as references when they started to code. Studies related to psychology of programming observed that experts adopt a planning mechanism based on a breadth-first approach, while novices, who often rely on their understanding of programming languages, adopt a depth-first approach (Robilliard 1999).

The story card should be kept simple. They didn't help because there was quite irrelevant information on the story cards. Some of the thing we don't really know until we have start programming.

Quite a lot of the part we have to write or rewrite the story cards as we go along the implementation....so we can afford to throw some of them.

We didn't use that much story cards towards the end though. Once we got on with the code, we didn't use much story cards. The story card helps to gain an idea of what it is going to be. Once we already done all that we got the first few (codes) started....then the story cards are no longer suitable.

When we have to do the testing stuff, we have to go back to that stuff in the story cards.

Students progress from a systematic planning activity to an opportunistic one with the evolution of the design, a process that is not always balanced. The introduction of the X-machine diagram was seen as a solution to introduce a depth-first approach before proceeding to broaden the software solution with story cards (APPENDIX 4-E). the X-machine model as described by Holcombe (Holcombe 2000) was chosen over other models not only because it provides simple notations for describing a system but it is also flexible and simple to use. Another advantage was its ability to abstract the detail from the state space.

In addition to the X-machine diagram, the students were also advised to use the existing HTML link to aid in connecting story cards with the overview X-machine diagram (APPENDIX 4-E). The following interview quotes indicate the resulting effect of these introductions:

The story cards help us to gain the idea what the system is going to be, once we got the first page, we knew how it is shaping up, we knew what they are going to do, so we didn't use the story cards (anymore)..

We did the X-machine first and then the story cards and then back to X-machine (detail)..... We showed him (the client) the first couple of screen at first, a prototype really and then the X-

machine (brief diagram overall). He likes the one (X-machine), which show the name because he could see all the section (of the story cards) that was coming off the X-machine but he did not read them (story cards) thoroughly ...

We used the flow diagram (X-machine diagram) for him (client) to understand the functions and passes.

Otherwise we can't visualize that (the software). If one person is talking about doing something, the best way to explain it is by drawing it, otherwise we will get confuse with the word. Diagram is good for showing the overall picture ... and then if you want to know more, then you go to the source code that was commented. (G03)

We draw what is supposed to work and from that write a story. The XM helps because it is a simple way of representing. If he used a lot of time doing it (diagrams) then we wouldn't have time doing the prototype.(S04)

I think XM were really good and helpful. We didn't get frustrated from figuring how to do them. The XM is literally a bubble to represent the system that link to one another

Drawing the diagram is not a waste of time because it builds your knowledge of what exactly to do, the process of asking, getting the facts.

Simple design

Waarts (Waarts et al. 2002) observed that in the later stage of the innovation diffusion, process improvement was needed to fit the needs of the other target groups, thus making it easier for these groups to switch to the new technology. Simple design in XP refers to the lack of diagrammatic structure but instead the developers proceed to write user stories. A user story is an agreement between the developers and the client, which must provide something of value to the clients (Beck et al. 2001)

Genesys is a special software house whereby the whole work force is replaced every year. Thus the company is in a dire need of an effective communication tool

to convey sufficient information from previous students to the next students. The documents produced must be self explanatory and easy for self learning of the projects. Research in cognitive science has proven that visual displays provide effective support for communication between writer and learner. Longitudinal study on Genesys revealed that the simple documentation on software design advocated by the XP pioneers was 'not effective' especially when a 'new' team acquire a maintenance project. A study of the two teams doing the maintenance tasks revealed the difficulties encountered by the members in comprehending the overall project. This was due to the non-existence of diagrammatic figures in the project documentation. Realising this weakness in the XP documentation, the next batch of students were encouraged to use the Xtreme X-Machine (XXM) diagrammatic tools for designing and reengineering the projects.

In the early study of the projects developed by the Genesys 2001, it was observed that the projects, which lacked overview design diagrams, created serious problem in understanding the project, amongst the next batch of students in Genesys 2002. There was a lack of connectors between story cards produced and the developed code, which made it difficult for these students to understand and maintain the software efficiently. Even though the lack of design in the XP approach shortened the time taken to develop and deliver the project, the follow up study on the same project showed more time was needed by the new team (Genesys 2002), inheriting the project, to understand the code and its interrelationship with the various databases.

> *There are many omissions, inconsistencies and errors that would make it difficult for another team to gain a good understanding of the project from the story cards.*

Observing that the lack of an overview diagram of the whole project increased the time taken for understanding, a stream X-machine diagram was added as part of the design documentation. This model was chosen because it fits the requirement of simple design specified by the XP methodology. The model was introduced in concurrent with research on graphical representation, where considerable evidence

has suggested that the format and sequence of presenting the information influenced the understanding of subject matter (Ainsworth et al. 2003; Schnotz 2002; Verdi et al. 2002). Previous studies have shown that conceptual diagram is best presented first before text for the simple reason that 'text never describes a subject matter with enough detail to allow only one kind of 'envisioning'(Schnotz 2002) and presenting the text first will cause interference with the picture presented afterward (Verdi et al. 2002).

For this reason, management in the Software Hut 2004 requested the students to use an X-machine diagram to draft the overall diagram before writing the story cards. The lack of diagram had been known to inhibit better understanding as findings in cognitive science have shown. Research by Hmelo-Silver et al (Hmelo-Silver et al. 2004) has shown that the more difficult the subject content is, the higher is the frequency of looking at adjunct visual display and the supportive function of these displays. This was evident with learners of low prior knowledge.

> *I think with the X-machine, we have little screen example of what you have on the screen to represent the bubbles so that we do not have to look at the screen to know exactly what each screen is meant to do.*
>
> *The diagrams (X-machine and screen) help during the discussion with the client.*

4.3.1.4 Trialability

Trialability is defined as the degree to which an innovation may be experimented on a limited basis. New ideas that can be tried on the installment plan are generally adopted more quickly because they represent less uncertainty and make learning a possibility. The ability to introduce XP practices incrementally in an organization enables it to be adopted more quickly because the approach does not require major changes. Even though literature in XP methodology stated that the practices must be adopted as a whole to see the advantages of them, the flexibility of the practices, which allow them to be compartmentalized, allows organizations to experiment

with these practices in stages as conducted by the teams in the SSEO and in IBM UK.

Sheffield Software Engineering Observatory (SSEO)

The trialability of this methodology was carried out during the course of this study. The adoption of the XP methodology was carried out in 2-ways: full adoption and partial adoption. *Full adoption* referred to the decision by the managers to implement all of the 12 practices and *partial adoption* referred to the adoption of some of the practices (Refer to Table 3-2) .

During the study in Software Hut 2002, the XP students were requested to apply full adoption to develop the software. The effects on the students using this methodology were then compared to the effects on the students using the design-based approach. At the end of the term, the managers conducted group interviews with every team and discovered that the XP teams had difficulties to understand and to apply all of the practices. Realising this problem, the managers decided on *partial adoption* of the XP methodology in the next study. In the partial adoption strategy, the students concentrated on understanding and applying several practices only.

Again the effect of this decision was compared with the effect of using the design-based methodology. The poor result obtained by XP students made the management decided to apply the full adoption strategy. The differences in the results are discussed in Chapter 5, 6, 7 and 8.

4.3.1.5 Observability

Observability is the degree to which results of an innovation are visible to others. The ability to observe the result enables others in the social system to seek innovation-evaluation information and stimulates discussions of new ideas amongst the adopters. The ability to run the XP approach in the Sheffield Observatory

enables the result of the experiments to be visible to the outside world. Quantitative results of the experiments are discussed in Chapter 5, 6, 7 and 8.

Display-based problem solving to system metaphor

In nearly all of the teams interviewed, the students acknowledge the advantages of using screen design to system metaphor in assisting them to capture the client's requirement better. System metaphor refers to the world concepts applied to in order to improve our understanding of the concept discussed (Dubinsky et al. 2003). It allows the students to comprehend one domain of experience in terms of another (Lakoff et al. 1980), indicating that to use metaphor effectively, students need to possess the entire understanding of the domain and not selected concepts. Lakoff (Lakoff et al. 1980) explained that the focus of the definition usually correspond to the natural kind of experience, which is actually a product of the human bodies, human interaction with physical environment and human interactions with other human within the culture (p. 117). Metaphor deals with a view of definition that is different from the standard view. The standard view seeks to be "objective", and it assumes that experiences and objects have inherent properties that the human can understand (p.119).

To use metaphor, the students need to be experienced and able to identify the exact metaphor for the association. Unfortunately, in most cases the experiences of these students and the clients were at a cross roads. The second reason for failing to use a metaphor is that metaphor is never specific. There is the element of ambiguity in the metaphor and to use ambiguity to combat ambiguity is definitely a recipe for a disaster. The third reason for not adhering to this practice corresponds to the ideas generated in the cognitive science concerning distribution and substitution. Distribution involved the current state of problem and substitution referred to the developer using perceptual skills to make inferences concerning problem implications, goal selection, or problem constraints (Walenstein 2002). Larkin (Larkin 1989) argued that with display-based problem solving (DBPS) nearly all or the most relevant problem solving state can be easily communicated, fewer logical

inferences were needed, allowing more perceptual inferences to be utilized to overcome ambiguity and also display allowed less deliberation on the part of the students thus giving them local control of the ambiguity.

Analyses of the interview data discovered that when the students were faced with the ambiguity of the clients' requirements, met with the high level of uncertainty and confronted with the different level of understanding regarding the business functionality, the teams involved in these empirical studies shared a common approach to overcome these difficulties. In all of the cases, the students preferred to use display-based problem solving than to use a metaphor to achieve common vision with the clients. Interestingly, the analysis found that the answers given by the students fall into the three categories outlined by Larkin (Larkin 1989).

The students expressed that by using this approach, they discovered that the clients were able to identify their requirement easily.

> *The screen is very easy to use, just a click of a button. There is only basic functionality. Obviously it depends on the client. For our client, it is as simple as possible. The easiness should be incremental, if the client has a need to use it.*

> *Until he can see it on the screen, we will not know what he is talking about or what he wants himself. I think it give him confidence to tell you. It is easier to point to certain screen than only saying what needs changing than trying to explain.*

> *We didn't do it (screen design) from scratch but use the browser. It solves the problem of what worked, faster. It helps us because he doesn't know what he wants.*

> *We gave him screen shots of what can be done... This approach speed up his process of understanding.*

The students used the screen to avoid ambiguity between the clients and them.

> *We generated the computer screen and showed it to the client so that he (client) could give us the feedback.*

We showed him the first couple of screen at first, a prototype really..

We achieved the common vision with the screen.

...when we discovered that it is slightly different it gives you the chances to change what needed changing. This is good thing because it gives you the idea of what the client want...

In dealing with distributed project, team needs to go further to demonstrate the security of data amongst the clients:

We made the prototype of that. We just used simple user interface, we have two computers, and we link them up to show how the agency can interact.

The students used the screen to permit only a little deliberation, therefore giving them the local control of the ambiguity.

We try to visualize everything by having the prototype working and developing the screen...

It makes you realize that you have to get something down and show them earlier on. We start coding quite early. Doing design is good because we can visualize it but doing the coding, we have something that is working, we get to try it (design) out and then realised that it was not it then we can go back and modify the design. In this way we can make it more efficient.

The discussion (with the client) is more on interface-to-interface rather than function or methods.

User involvement

There was a growing acceptance of the user involvement within the software engineering field. The term user involvement generally refers to a direct contact with the users and can be broadly characterised as being somewhere on the continuum from informative, through consultative to participative.(Kujala 2003) The term was often used as a synonym for 'user participation', 'user

centredness'(Heinbokel et al. 1996), 'user orientation' and 'consulting user'(Kujala 2003).

The goal of a user-centred design is the development of useful and usable products (Kujala, 2003). One of the principles of user-centred design is the early and continual focus on users, and it is generally agreed that usability is achieved through the involvement of potential users in the system design (Kujala, 2003; Karat, 1997). Damodaran's (Damodaran 1996) study on user involvement showed that several benefits arise, such as a more accurate user requirement, which resulted in avoiding the development of the unused system features and a higher level of user acceptance due to the greater understanding of the system. On the contrary, Heinbokel et al (Heinbokel et al. 1996) empirical research on user usefulness showed the negative effects of having user participation or user orientation during the software development. His study disclosed four problems that had arisen: first, the users developed more sophisticated ideas in the course of the development process and therefore, they intervened more frequently later on. Second, users often feared job loss or worsened working conditions as a result of the new software and therefore, were not interested in participating constructively. Third, user representatives were often unpredictable in their demands for changes and often these interventions were shown to disrupt the software development process. Fourth, user orientation may lead to higher level of aspiration and therefore, resulting in a higher degree of stress and a lower degree of team effectiveness.

The suggestion that XP methodology included an on-site customer should be carefully considered. The term on-site customer referred to having a real customer involved full time with the development team. The function of a customer at the site is not merely as a consultant but to be involved in writing functional tests and making decisions on project priority and scope for the team (Beck 2000). The availability of the communication technology such as mobile phones and email can lessen the importance of an on-site customer. The second factor that reduced the importance of this practice is the level of knowledge that the clients possessed. In the highly volatile projects, that is projects with unpredictable user requirement, the

lack of technological knowledge and indefinite user requirements render the clients ineffective to be at the developer's site. In some cases, the user knowledge had become implicit through automation. The third factor to be considered is the availability of the appropriate user all of the time. Throughout this study, all of the clients were reluctant to be stationed more than one third of the developmental time due to other commitments to their companies. The fourth factor is the existence of several user groups. The process of achieving compromises between the groups can be a harrowing experience to the students involved.

With regards to this practice, there were mixed responses from the students. Some expressed the need to have on-site customer only when the client is not prompt at clarifying certain issues:

> *We could use on-site customer more at the beginning...(when).*
>
> *Our customer thought he knows what he want but actually he doesn't really know what he wanted.*
>
> *Occasionally we emailed to him but he doesn't really read his mail.*
>
> *We do get feedback on everything, only it wasn't immediate.*

While others do not need the on-site customer...(because)

> *The client does give us constant feedback.... We get back feedback from him by email.*

4.4 The XP team in the IBM Hursley Park, United Kingdom

To add validity to the findings in the Sheffield Observatory, data was collected from an XP team in the software industry. Some of the information below is based on personal correspondence (APPENDIX 4-F and 4-G) and from a paper written by the team (Mitchell et al. 2003).

In the early 2003, ten developers were assigned to deliver an internal component for WebSphere Application Server, one of IBM's large middleware products. The team recognized the need to adapt to the changing requirements, as well as pick up experience and learn new technologies along the way. Therefore they decided to use a more flexible approach to the development process, to enable them to measure the progress effectively, and to break large, seemingly unmanageable chunks of work into something more understandable. An analysis of different agile delivery methods was undertaken, and the Extreme Programming (XP) which seemed to fit the requirements most readily, was selected. Since that was the first effort to adopt XP methodology, the team sought the approval of the management team. The project lasted approximately one year and at the end, the team managed to complete and deliver the component on time.

Effort was made to collect qualitative and quantitative data from the team. The researcher was able to collect the first reading for the quantitative data but unfortunately, before the second reading could be collected; three of the team members were reassigned to another project. The sample size of 3 is too small to enable generalisability of the quantitative data between the Genesys teams to the industrial team. Therefore, effort was focused on collecting the qualitative data only. The collection of qualitative data from IBM was made through a teleconference, and from email correspondences between the researcher and the team. The decision to use these methods was made because of the timetable problem in conducting a focus group interview with a team containing members who were reassigned to different teams within the company.

Planning game

At this point, a release plan was produced identifying the user stories scheduled for delivery in each iteration within each release. Based on customer input, highest priority user stories were scheduled at the beginning of the release so we could,

therefore, focus our efforts on the most important functions first.(Mitchell et al. 2003)

However, we still had to adhere to the standards imposed upon us by the parent project, which meant completing a number of documents that would provide status to the larger project community. Although this was extra overhead and duplication, we considered this a customer requirement; that is, our customer wanted us to track progress a certain way, and so we respectfully obliged.(Mitchell et al. 2003)

Frequent release

This practice provided the greatest confidence that the most important functions would be delivered with success.

Continuous interation

We followed XP practices closely in order to deliver our code iteratively. ..It took a few iterations for us to find the best day to begin an iteration... we also found that we needed a mid-iteration checkpoint meeting so we could ask the question, "How much more work have we got to do for this iteration and are we going to do it all?"... Asking this question enabled us to identify potential problems early and establish appropriate remediations. (Mitchell et al. 2003)

On-site customer

...not feasible different 'customers' spread across globe.

Forty-hour week

final week.. yes - XP team rarely work late/weekends

Story cards

Write stories with customer , no – customer representative writes stories

Discuss stories yes – but sometimes through an intermediate (non-XP) 'architecture' team representing all customers

Discuss user interfaces yes – via intermediate (non-XP) 'architecture' team on which each customer is represented

Coding standard

Coding standard yes – automated checking – but we haven't been all that strict

Identify standard yes – we agreed on the look and feel of the code at the start – our own standard

Naming convention yes – sometimes this is dictated to us by the encapsulating middleware product conventions

Quality

The combination of test-driven development, continuous integration with fully automated regression testing, pair programming, and team design certainly improved our code quality. Additionally, refactoring was actively encouraged and allowed code to be refined and improved over time. An agreed-upon common set of coding standards ensured consistency. (Mitchell et al. 2003)

Lack of code ownership, inexperience, and our (correct) reliance on the unit tests, did sometimes result in inelegant code, but refactoring later in the project cycle rectified this(Mitchell et al. 2003)

In summary, the team discovered that the XP experience was highly valuable, and had dramatically changed their view on how to deliver software, particularly with regard to scenario-driven development and testing. The team also felt that it was very important to consider the personality balance within the team since some members found it an uncomfortable environment to work. Research by Gitins et al (Gittins et al. 2004) supported this observation because the study discovered that social compatibility affected the experience of pair programming among the developers.

It was observed that there was no big contradiction between the findings in the Observatory and the IBM team. Broadly, the findings from IBM support findings

from the Observatory. However, because of the different ways the data was collected, it is not possible to claim the full generalisability of the result.

4.5 Discussion

Early research (Soloway et al. 1986) had demonstrated that developers worked effectively when the techniques they were required to use supported the structure of knowledge in their knowledge base. The findings in this chapter provide an insight into the practices that were easily adopted by the teams and the reasons why the development teams were more receptive toward these practices. The cognitive, psychological and management theories attached to these choices of practices have provided the reasons why certain practices were easily used and some practices were not easily accepted. The empirical findings in this study added to the existing evidence of the need to adapt some of the practices to suit the working environment of the developers (Gittins et al. 2004; Robinson et al. 2004; Sharp et al. 2003).

CHAPTER 5

THE POSITIVE AFFECT OF THE XP METHODOLOGY

5.1 Introduction

Past research has shown that a positive affect induction leads to a greater cognitive flexibility and facilitates creative problem solving across a broad range of settings. Research works by Isen et al (Isen 2001), Ashby (Ashby et al. 1999) , Aspinwall et al (Aspinwall 1998), Carnevale et al (Carnevale et al. 1986) suggest that positive affect increases a person's ability to organize ideas in multiple ways and to access alternative perspectives and also to improve performance in several tasks that are typically used as indicators of creativity or innovative problem solving.

Therefore, it is the intention of this chapter to explore the possibility of the XP methodology as a positive affect inducer and to discuss the findings of a longitudinal study on the possible impact of the selected XP practices on the positive affectivity of the developers. To achieve this, a comparison study was conducted on the Software Hut 2003 developers followed by a replication of the study on the Software Hut 2004 developers.

The first focus of this chapter is theoretical, and it is aimed at addressing the XP practices that are related to the feedback. The next section of the chapter discusses the result of the comparison studies. Findings revealed that the XP methodology does have an impact on the positive affectivity when most of the practices were implemented and the variable for the style of management was controlled.

5.2 The Positive Affect of the XP Practices

In this study, XP methodology was chosen as a positive inducer because of the existence of several XP practices that warrant feedback to the developers. Positive feedback about one's performance has been known as a positive affect inducer (Estrada et al. 1997). On the other hand, research by Aspinwall *et al* (Aspinwall 1998; Isen et al. 1978) and Isen *et al* (Isen et al. 1988; Isen et al. 1978) in the educational and interpersonal relations domain, have also shown that careful processing of the negative information is essential for obtaining feedback about one's progress towards goals, to formulating appropriate performance standards, and to evaluating behavioral alternatives. The accumulating evidence suggests not only that a positive mood does not necessarily compromise processing of negative information, but also positive affect may predict increased attention to negative information and a more careful, thorough processing of such information when the information is self-relevant or important. The XP practices associated with feedback seeking are simple design, pair programming, continuous testing, continuous integration and frequent review (release).

Studies by Aspinwall (Aspinwall 1998) and Muraven et al (Muraven et al. 1998) note that people must have a surplus of resources such as time, energy and attention to engage in a proactive behaviour. In the XP approach, the developers experienced a surplus of time during the coding because less time was engaged in the designing phase. With Beck's (Beck 2000) simplified design, he is actually releasing the stressful task of creation, thus liberating the mind to be more creative and innovative. By reducing the technical aspect of the design, the mind was able to approach the problem solving task through a breadth first approach. Design is only an early manifestation of ideas, whereas the coding process allows the developers to realize their idea in a more concrete way. This approach is considered as a positive affect inducer because it allows feedback on the design through the programming code. The ability to see the advantages and identify the flaws in the design allows the developers to be more creative in the next part of the system. This is the reason why simple design can accommodate a flexible requirement because

the process of creating part of the system in this manner allows the developers to be more innovative in the problem solving process.

It was observed that the practice of pair programming started with the initial socializing amongst the pair thus creating a positive mood amongst them before any formal programming commenced. The positive mood which is experienced and the attention of the two developers allow the pair to engage in a more proactive behaviour. The ability to discuss the advantages and disadvantages of certain coding ideas enables the pair to seek improvements and to avoid specific weaknesses. Even though pair programming was not a favourite practice, because it was perceived as difficult due to being time consuming and at a different level of programming experience, nevertheless, at the end of the project, the members often acknowledge that their creative ideas were explored much more during this process. Studies on pair programming have provided the evidence about the benefits of pair programming (Cockburn et al. 2000; Succi et al. 2002; Williams 2002). The existence of more attention paid to a portion of a system during pair programming, influences the pair to be more proactive. Pair programming enables the developers to seek and receive feedback about their programming ideas immediately and in a very conducive manner. This is achieved because of the presence of a partner, who is involved directly in developing the same part of the system concurrently, and this enables the pair to thrash out ideas and this in turn makes the pair feel satisfied (Cockburn et al. 2000) and confident concerning their work. With pair programming practice, positive affect is induced through early socializing, more attention and immediate feedback amongst the pair.

Continuous testing allows feedback on the developed code. In the normal software testing domain, testing is usually left to be performed at the end of the development cycle thus leaving a very short time for complete testing. In this situation, often the developers were faced with products that have too many defects, as the bugs were discovered too late. The benefit of testing as the software is developed is that the developers are always certain that the software developed is always test compliant. Continuous testing is a practice that is structured so that different levels of testing

can be conducted as the solution is being built. In the study by Trope and Promerantz (Trope et al. 1998), participants in whom positive affect has been induced showed greater interest in the part of the test they had failed than did neutral mood participants. The emphasis of the continuous testing enables the developers to feel more confident about the correctness of the code and therefore bolster their confidence and self-esteem.

Continuous integration is another feedback seeking practice, which allows the developers to address performance problems earlier in the development process. The more frequently the developers were able to test the integrated system, the more often they were able to check the functional integrity of the application as some problems do not manifest themselves until they are in the integration environment, such as when a database application is finally tested in a genuine load. The ability to address the performance problems early and to continuously improve the system allows the developers to enjoy a level of self regard or positive affect. Developers using this practice had the advantage of bolstering their self-esteem continuously, as they worked to perfect the functionality of the integrated system. Therefore, less time was required to address bottlenecks at the end of the process, and as a result developers felt less pressure to maintain a breakneck coding pace.

Frequent release, or in this study, it is more appropriately referred to, as review, is another practice that commands feedback. Feedback from the client, be it positive or negative, is also a positive affect inducer. Accumulating evidence suggests that positive affect can create an increased interest in information about one's liabilities. A study by Trope and Neter (Trope et al. 1994) has shown that prior positive experience subsequently increased the interest in feedback of high rather than low self-relevance, even when the feedback was expected to diagnose weaknesses rather than strengths.

The above theoretical study of XP practices identified these practices as being a positive affect inducer. The studies in the next section were carried out to determine

empirically, whether teams using these practices would experience higher positive affectivity than the teams using the design-based Discovery methodology.

H_1: The XP team will experience a higher level of positive affectivity than the Design team at the end of the project.

H_2: The number of the XP practices is positively correlated to level of the positive affectivity of the members in the XP teams.

To test these hypotheses, two comparison studies were carried out on the Software Hut developers in 2003 and 2004.

5.3 Comparison Studies

To measure the developers' state of the positive affect, the positive affect scale of the Positive and Negative Affect Schedule (PANAS) was used. Positive affect was induced by introducing and requiring the XP methodology to be used by half of the development teams. The studies do not include the negative affect because previous research has shown that positive affect operated as a single construct, indicating that the fluctuation of the positive affectivity, has no effect on the negative affectivity of a person (Anderson et al. 2004).

5.3.1 Study 1 (SOFTWARE HUT 2003)

The Positive Affect scale showed a satisfactory internal consistency coefficient, Cronbach $\alpha = 0.87$ during the first reading and $\alpha = 0.88$ during the second reading. At the beginning of the project, independent sample t-test was conducted to test the difference between the two development teams. The result showed that there was no significant difference between the Discovery team (N = 18, Mean score = 27.78, SD = 8.59) and the XP team (N = 16, Mean score = 30.19, SD = 5.85), indicating that all of the developers have a similiar dispositional state (Figure 5-1, Tpos1).

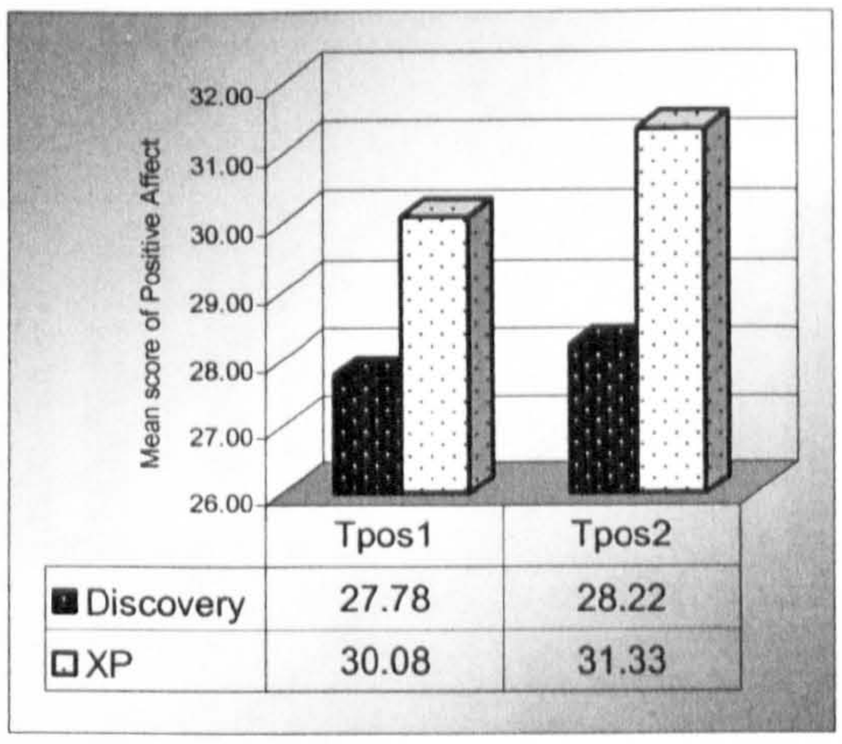


Figure 5-1 Bar graph oF the positive affect of the two teams before treatment (Tpos1) and after treatment (Tpos2), [Software Hut 2003]

To test the above hypothesis, mixed between-within ANOVA was conducted. There was no significant effect for both time and method between the two teams [Discovery ($N = 18$, $M_1 = 27.78$, $SD_1 = 8.59$ and $M_2 = 28.22$, $SD_2 = 6.79$) and XP ($N = 16$, $M_1 = 30.19$, $SD_1 = 5.85$ and $M_2 = 30.94$, $SD_2 = 6.56$)]. Even though the data for the XP team showed a greater increase in the mean score for positive affect after completing the project, the difference was not significant between the two methodologies (Table 5.1, Tpos2).

The teams were then grouped according to project (Figure 5-2) and a further analysis was conducted. The mixed between-within ANOVA test conducted does not indicate any significant difference between the two methodologies for both times (week 2 and week 11). The readings were taken for all of the projects. There was no main effect relationship between the time and the methodologies, indicating that the result may be moderated by other factors. Factors that should be considered as the moderator of effect are style of management and the partial adoption of the XP practices during this study. These weaknesses in the study are discussed in detail in Chapter 3.

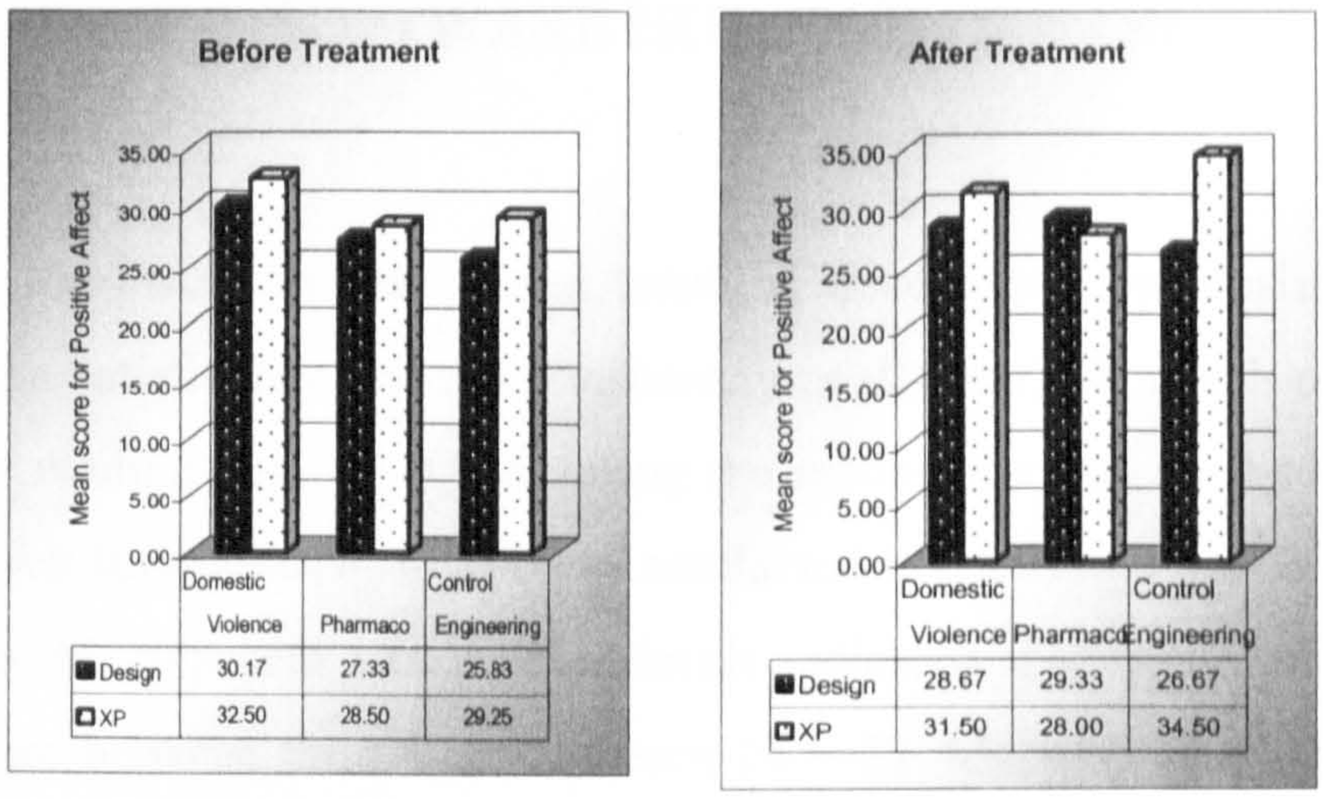


Figure 5-2 Bar graph of the positive affect before the treatment and after the treatment according to project (Software Hut 2003)

Project	Method	N	M1	SD1	M2	SD2
Domestic Violence	Discovery	6	30.17	(10.07)	28.67	(6.055)
	XP	6	32.50	(4.23)	31.50	(6.77)
Pharmaco	Discovery	6	27.33	(7.76)	29.33	(6.38)
	XP	6	28.50	(6.92)	28.00	(6.29)
Control Engineering	Discovery	4	25.82	(8.79)	26.67	(8.66)
	XP	4	29.25	(6.60)	34.50	(6.14)

Table 5.1 Descriptive Statistic for Software Hut 2003

Nevertheless, a comparison of the mean score for positive affect amongst the XP teams (Table 5.1) showed that the team developing the Control Engineering project (volatile project) experienced a higher positive affect [M_1 =29.25 : M_2 =34.50] than the teams developing Domestic Violence project [M_1 =32.50 : M_2 =31.50] and Pharmaco project [M_1 =28.50 : M_2 =28.00].

The relationship between the number of XP practices and the positive affectivity was investigated using the Spearman's Rank Order correlation. The analysis showed no correlation between the two variables [$r = 0.062$, $n = 16$, $p = 0.82$] in all of the projects. The results from this study, therefore, do not support both of the hypotheses stated earlier.

5.3.2 Study 2 (SOFTWARE HUT 2004)

During the study on the second batch of developers, the Positive Affect scale showed a satisfactory internal consistency coefficient, Cronbach $\alpha = 0.930$ during the first reading and $\alpha = 0.928$ during the second reading. At the beginning of the project, an **independent t–test** was conducted to determine the difference between the two development teams. The result indicated that there was no significant difference between the Discovery team (N = 32, Mean score = 29.50, SD = 9.69) and the XP team (N = 28, Mean score = 30.29, SD = 6.41) indicating that all of the developers have a similiar dispositional state at the beginning of the project (Figure 5-3 – at Tpos1).

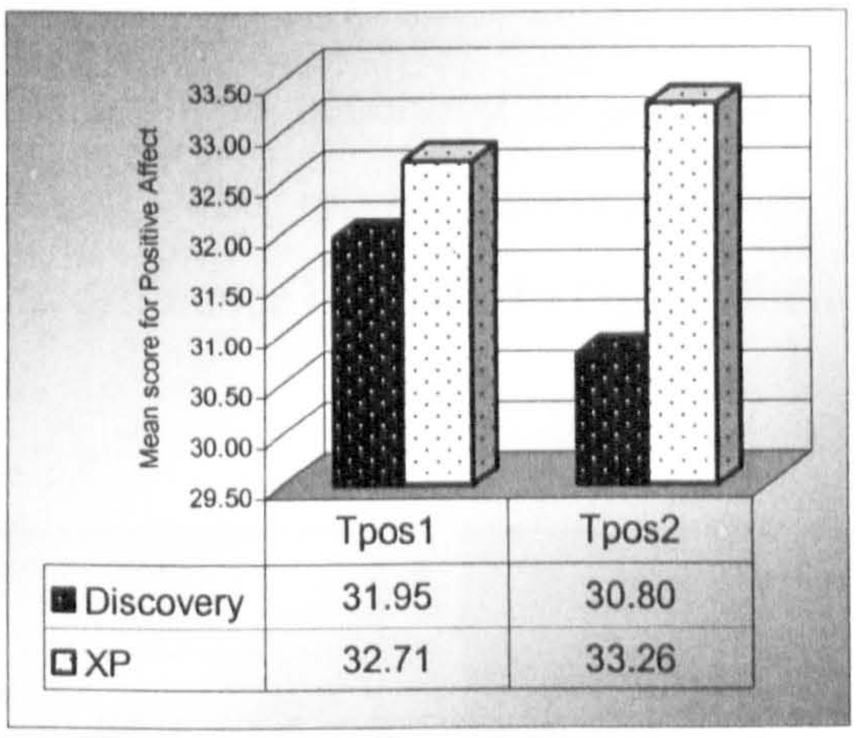


Figure 5-3 Bar graph of the positive affect of the two teams before the treatment (Tpos1) and after the treatment (Tpos2) [Software Hut 2004]

To test whether there was a difference in the positive affect after the methodology treatment, the **mixed within-between ANOVA** test was conducted. The result indicated a statistically significant main effect of the week [F (1, 59) = 4.56, p = 0.037], however, the effect size was small (eta squared = 0.074). The **paired sample t-test** was then conducted to determine whether there was a statistically significant difference in the mean score of positive affect for week 2 and week 8. The result indicated a significant difference in the positive affect for XP teams between week2 ($N = 27$, $M_1 = 30.30$, $SD_1 = 6.533$) and week 8 ($M_2 = 34.30$, $SD_2 =$

8.892), t(27) = 2.64, p = 0.014). The results showed that the XP methodology has the ability to increase the positive affectivity of the developers. However, the reading for the Discovery teams showed no significant difference in the positive affect for members between week 2 (N = 32, M_1 = 29.50, SD_1 = 9.69) and week 8 (M_2 = 30.25, SD_2 = 7.78) indicating that the Discovery methodology does not have a significant impact on the positive affectivity of the developers.

The **Independent t-test** was conducted to compare the positive affectivity between the two methodologies and the result showed no significant difference between the two readings [Tpos2 for Discovery and XP, F(1,57) = 1.379, p = 0.2455] indicating that even though the XP approach has some impact on the positive affectivity but the impact was not significant enough between the two methodologies. (Figure 5-3 Total Positive Affectivity before treatment [Tpos1] and after treatment [Tpos2].).

The teams were then grouped according to projects (Figure 5-4) and a further analysis was conducted. The mixed between-within ANOVA test conducted indicated a significant difference between the two methodologies and the projects.

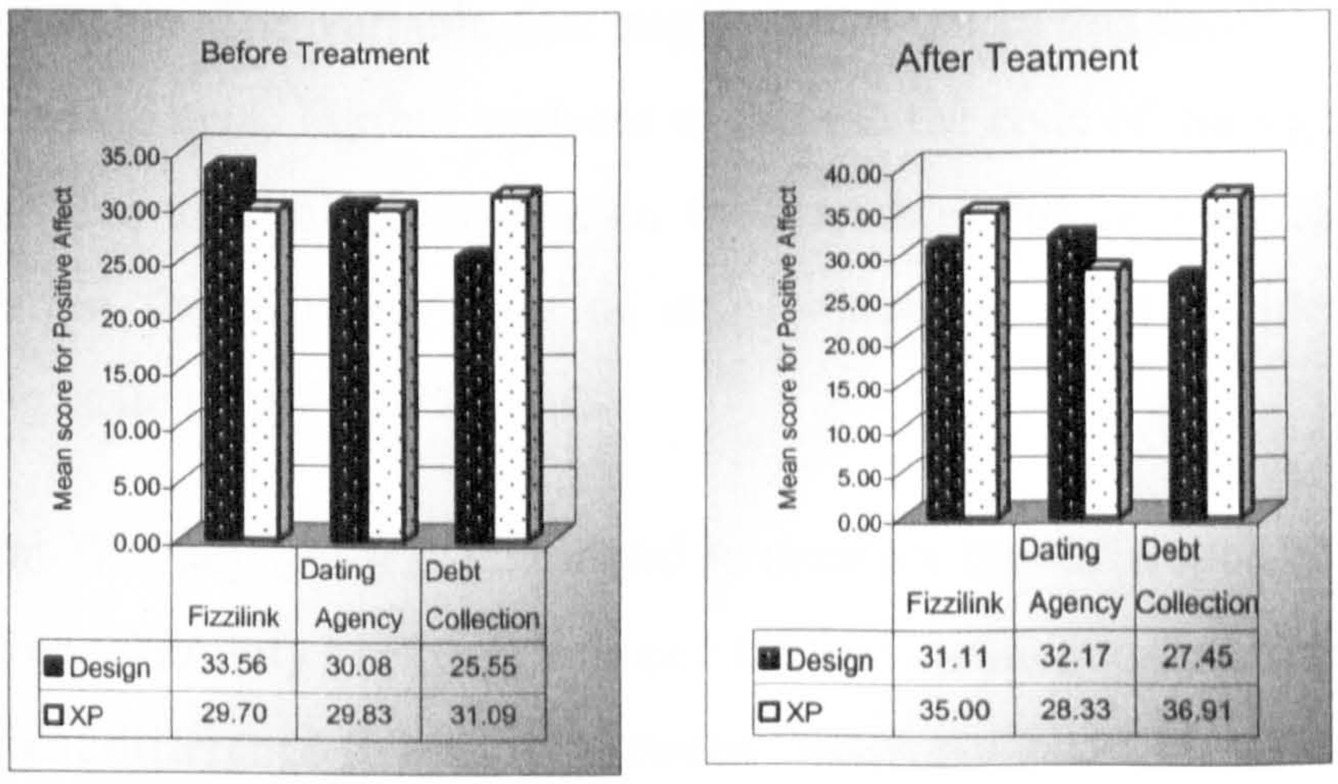


Figure 5-4 Bar graph showing the positive affect before the treatment and after the treatment according to project (Software Hut 2004)

Wilcoxon Signed Rank test was conducted to measure the change in positive affectivity amongst the three projects (Table 5.2). In all of the Discovery teams, there was no significant difference in the positive affectivity reading.

Project	Method	N	M1	SD1	M2	SD2
Fizzilink	Discovery	9	33.56	(6.912)	31.11	(6.51)
	XP	10	29.70	(6.46)	35.00	(8.01)
Dating	Discovery	12	30.08	(11.06)	32.17	(8.78)
	XP	6	29.83	(3.06)	28.33	(9.52)
Debt	Discovery	11	25.55	(9.29)	27.45	(7.44)
Collection	XP	11	31.09	(8.22)	36.91	(8.55)

Table 5.2 Descriptive Statistic for Software Hut 2004

However, in the case of the XP teams, there was a *significant increase* in the positive affectivity for developers doing the Debt Collection project [$M_1 = 31.09$, $SD_1 = 8.22$; $M_2 = 36.91$, $SD_2 = 8.55$, $Z(11) = 2.627$, $p = 0.009$] whereas the developers have shown *no significant increase* in the positive affectivity for Fizzilink project[$M_1 = 29.70$, $SD_1 = 6.46$; $M_2 = 35.00$, $SD_2 = 8.01$] and *no significant decrease* for Dating Agency project [$M_1 = 29.83$, $SD_1 = 3.06$; $M_2 = 28.33$, $SD_2 = 9.52$]. The result indicated that the influence of XP methodology was mediated by another variable. The variable that might contribute to this result was the style of project management. Further analyses to include the style of management were not conducted due to insufficient data on this variable. Future work should take into consideration the different style of management in order to identify a proper measuring scale to collect this data.

The Mann-Whitney test was conducted to measure the difference in mean score of the positive affectivity (TPOS8) between the two methodologies. The result showed a significant difference in reading between Discovery team [$M = 27.45$, $SD = 7.44$] and XP team [$M = 36.91$, $SD = 8.55$, $Z(22) = 2.432$, $p = 0.013$] for the Debt Collection project but no significant difference for the other projects.

The findings in this study showed significant increase in the positive affectivity for teams managed by a more 'neutral manager' (Debt Collection) compared to the

teams managed by a 'result-oriented' manager (Fizzilink) and a 'technical-oriented' manager (Dating). An effective manager possesses the ability to influence team members towards the achievement goals. According to contingency theories, the managers can influences the group through *autocratic-democratic continuum model, Fiedler's model, path-goal model* or *leader-participation model.* All of the managers in this study adopted a path-goal model, which described the leader as being responsible for helping the subordinates to achieve their goals by clarifying the paths to these goals. Throughout the study, it was observed that the neutral manager exercised no influence on both teams (XP and Discovery) allowing the team members to follow the selected methodology as closely as possible. On the contrary, the result-oriented manager has the tendency to demonstrate partiality towards the XP methodology thus creating a pro-XP environment even amongst the Discovery team. One of the Discovery teams made known they tend to do a few activities considered as part of the XP practices such as continuous integration supported this observation. This may explained the insignificant difference of the positive affectivity between the two methodologies. XP methodology emphasis simplicity in design and more focus on the coding process. The presence of a manager who is technical-oriented spelled a different version of an XP approach. During the development phase, the XP teams involved with the Dating project were observed to be more focused on the technical aspect of the design than the XP teams developing the Fizzilink and Debt Collection projects. This may explain the decrease in the positive affectivity amongst the team members at the end of week 8. It might be possible to improve the result if the variable for style of management is controlled. The result in this study **does support** the first hypothesis only if the management style has no influence on the methodologies used.

The relationship between XP practices and positive affectivity was investigated using the Spearman's Rank Order correlation. There was a significantly strong correlation between the two variables for every project: Fizzilink [$r = 0.570$, $n = 10$, $p = 0.085$], Dating [$r = 0.828$, $n = 6$, $p = 0.042$] ** and Debt Collection [$r = 0.665$, $n = 11$, $p = 0.025$] ** indicating that the XP practices are strongly associated with the

positive affectivity. This result supports the second hypothesis that XP practices do have a positive impact on the affectivity of the developers.

5.4 Discussion

In Study 1, even though there was no significant difference between the methodologies due to the weaknesses in the way the experiment was conducted, it is interesting to observe the impact of the XP methodology on the unpredictable project (Control Engineering). The findings in this study suggest that the XP methodology works best in an unpredictable environment rather than in the domain where the problem is known and understood. In the latter case, the most effective solution will be to write an appropriate algorithm using a structured method to deal with it. This is supported by the result discussed in Chapter 7 and 8. In dealing with an unpredictable condition, a more flexible method of developing the software is needed. Norman et al (Norman et al. 2003) refer to this as 'weak method,' in the artificial intelligence domain. It is referred to as 'weak; because its power lies in its capacity to help to deal with the unexpected problems so that it complements strong, algorithmic methods by adding robustness in the unanticipated situations. The teams using a more flexible approach, such as the XP methodology, were able to incorporate the constant changes made by the client and thus still maintain their positive mood.

The second finding in Study 1, clearly points out that for the XP methodology to be more effective in all domains, the developers need to incorporate all of the 12 practices. This is supported by an earlier study on the quality of the software produced (Syed-Abdullah et al. 2003b) and the discussion in Chapter 8, when only partial adoption of the methodology was imposed. Failure to use all of the 12 practices albeit with adaptation, has resulted in the failure to achieve the optimum outcome in term of affectivity, quality, well being and work group cohesion, which has been shown in the respective chapters.

The findings in Study 2 support the finding in the Study 1, that by incorporating all of the practices, the developers were able to experience a significantly higher positive affectivity. When a person experiences a positive affect, they show a greater preference for a larger variety of actions and see and think of more possibilities and options to solve whatever problem is faced. People with a positive affect are more likely to take action because they are proactive. Research on positive affects found that positive emotions such as joy, interest, contentment, pride and love broaden the scope of thinking and actions, thus they tend to approach and explore novel objects, people and situations under such emotions. The study suggests that when people experience joy and mild contentment, they are more likely to think of a wider range of action, become more resilient over time and are more likely to develop long-term plans and goals.

The second finding in Study 2, shows a significant increase in positive affectivity for teams managed by a more neutral manager compared to the teams managed by a result-oriented manager and a technical-oriented manager. For the Fizzilink project, the reason that both teams (Discovery and XP) experienced a similiar increase in the positive affectivity project might be due to the way the teams were managed. The high increase in the level of positive affectivity for both teams managed by the result-oriented style of management is only to be expected. Previous study by Appelbaum et al (Appelbaum et al. 1998) has shown that a major management feature that can lead to success is a deliberate bias towards implementing solutions to problems and also improving the performance by achieving an agreement amongst the employees.

Research in management behaviour has shown mixed findings in the effect of the manager's orientation on the job satisfaction. Job satisfaction is used as a measurement to positive affectivity because previous studies have found no direct relationship between management orientation and positive affectivity. Study by Brief at al (Brief et al. 1995) has shown a consistent finding with past research, that positive mood induction has been found to yield increases in pro-social behaviour

and this same kind of behaviour has been found to be positively associated with job satisfaction. Nevertheless, research within the management field has shown extensive studies on the relationship between the management orientation and the job satisfaction. To quote a few studies, researchers such as Euske et al (Euske et al. 1982), Holdnak et al (Holdnak et al. 1993) and Savery (Savery 1994) have reported a positive relationship between management behaviour and job satisfaction. On the contrary, Pool (Pool 1997) found a negative relationship between the structure of management behaviour and job satisfaction, while Hampton et al (Hampton et al. 1986) found no relationship between the variables.

The third finding in Study 2 shows a strong correlation between positive affectivity and the XP practices. This is to be expected because of the existence of the practices such as simple design, pair programming, continuous testing, continuous integration and frequent review (release) which commands feedback. This finding helps to provide empirical evidence that the XP methodology does have an impact on the positive affectivity of the developers.

CHAPTER 6

XP AND THE WORK GROUP COHESION

6.1 Introduction

This chapter describes the empirical studies, which address the aspect of work group cohesion amongst the members of the software development teams. The question of interest is whether an Extreme Programming (XP) methodology has any distinct effect on the cohesiveness amongst the software developers. This chapter presents a series of results that show the relationship of the XP methodology to the work group cohesion. In the second section, there is a presentation of the results of the comparison study between methodologies. In the third section, an in-depth understanding of the effect of the XP practices on the work-group cohesiveness is focused on the teams in Genesys Solutions Company. An examination of the teams in both environments (Software Hut and Genesys Solutions) in the Sheffield Observatory shows that there is group diversity in the form of demographics, personality and functionality. The final section briefly discusses the relationship between the XP practices and the group cohesion. The two parameters, personality and functionality, were part of a research project carried out by another PhD student. Demographic diversity refers to the differences in race, culture and age, while functional diversity refers to the differences in skill and knowledge (Pelled 1996). During the initial analyses, the assistance of Isabel Evans, a consultant from IE Testing, UK, was sought and obtained. The advice and recommendation given was used to improve the study in Genesys 2003 and Software Hut 2004.

The first purpose of this study was to assess empirically whether there are any differences in the level of the work group cohesion between the XP teams and the Discovery teams. The hypothesis was defined as follows:

H_A: The XP teams will experience a higher level of work group cohesion than the designed-based teams.

The study of the relationship between the number of the XP practices and work group cohesion level was not continued because the result of the comparison studies showed that there were other factors that affect the level of cohesiveness amongst the SE teams.

6.2 Comparison Studies

In this section, the results of the two comparison studies are discussed.

6.2.1 Experiment 1 (Software Hut 2003)

During this study, the work group questionnaire was administered during the 10th and 11th week. The weakness of this study was that there was no data on work group cohesion at the beginning of the project.

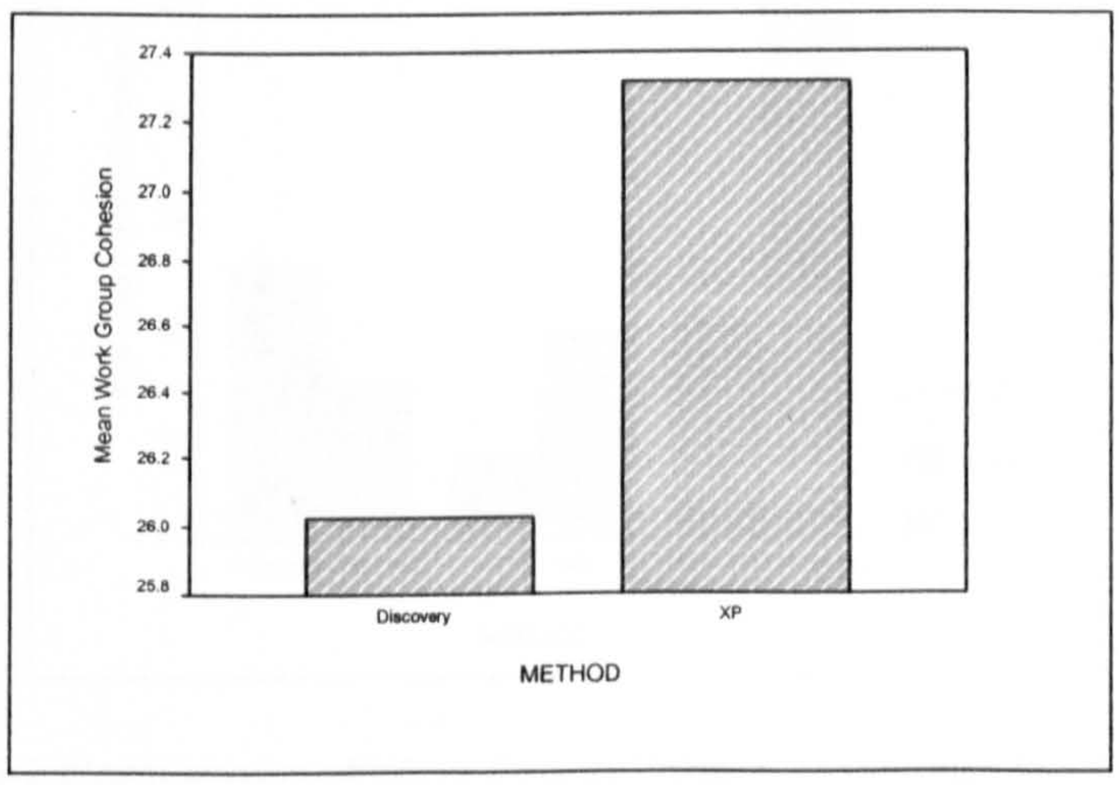


Figure 6-1 Comparison of work group cohesion level between the two methodologies (Software Hut 2003)

Presented in Figure 6-1 is the bar graph that depicts the differences in the mean score between the 2 teams. This graphic comparison showed that the XP teams experienced a higher level of work group cohesion compared to the Discovery

teams. A statistical **independent sample t-test** was conducted to investigate the degree of cohesiveness amongst the teams. The magnitude of the differences in the mean scores was very small (eta squared = 0.016), indicating **no significant difference** in scores for the Discovery (M = 24.30, SD = 8.26), and the XP teams (M = 26.25, SD = 7.019); t (75) = -1.120, p = 0.27 at week 11.

From the plotted graph (Figure 6-1 and Figure 6-2) , the bar graph supported the observation made that the members using the XP approach were more cohesive than the Discovery teams but the statistical result showed no significant difference between the two teams. The data was grouped, according to the type of project and a bar graph was plotted to explore the effect of the methodology and the project on the cohesion level of the working groups. Figure 6-2 illustrates that the level of work group cohesiveness was higher amongst the XP teams for the Pharmaco and the Control Engineering projects, while for the Domestic Violence project the Discovery teams was slightly more cohesive.

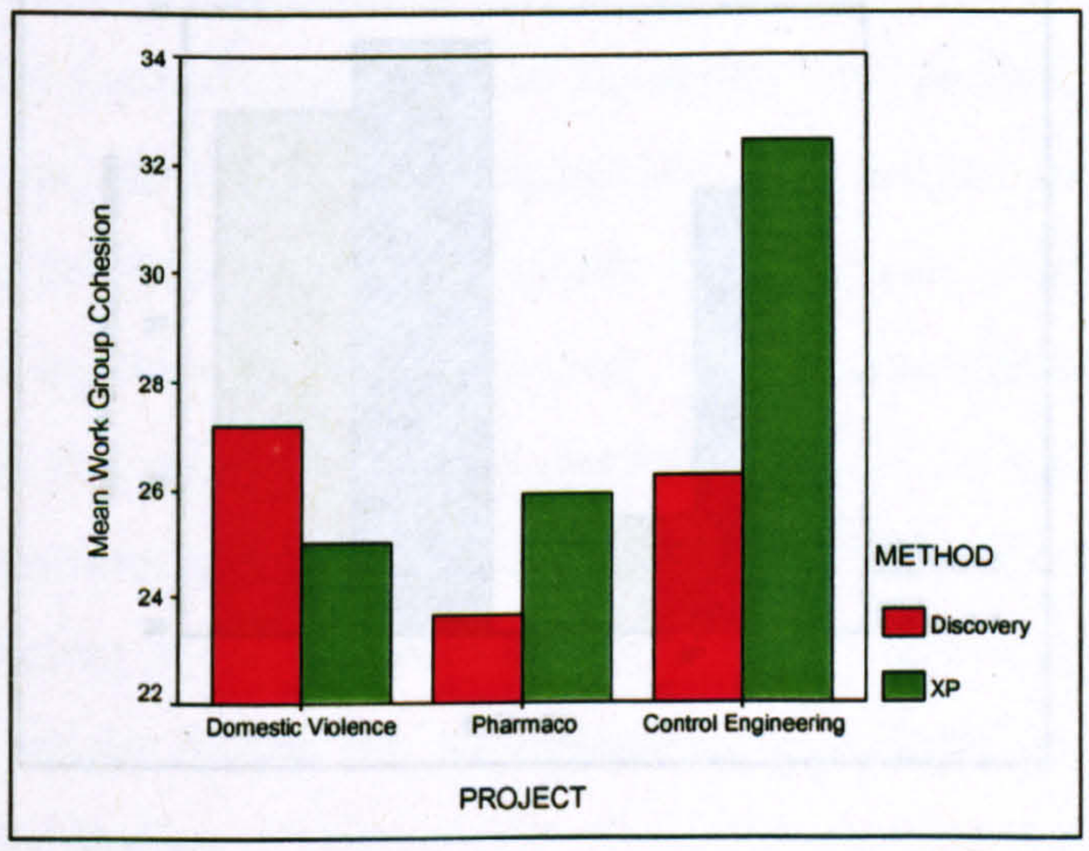


Figure 6-2 Comparison of Work Group Cohesion between the Discovery teams and the XP teams according to project for Software Hut 2003

A Mixed between-within ANOVA test was conducted to explore the impact of the methodology and the project on the degree of cohesiveness amongst the teams as measured by the Work Group Cohesiveness scale. Subjects were divided into 3 groups according to the projects (Domestic Violence, Pharmaco and Control). Even

though the graph indicates some differences in the level of cohesion between the groups, the result showed **no statistically significant effect** of the project and the methodology on the work group cohesiveness amongst the development teams [F(2, 11)= 0.002].

The result **rejected the hypothesis H_A** indicating that the XP teams do not experience a higher level of work group cohesion than the Discovery team.

6.2.2 Experiment 2 (Software Hut 2004)

The weakness of not having any initial readings before the treatments was rectified during this study. Two readings were taken; pre-treatment, which was collected during week 2, and post-treatment, which was collected during week 8. The cumulative work group cohesion scores for both of the methodologies are shown in the graph, Figure 6-3.

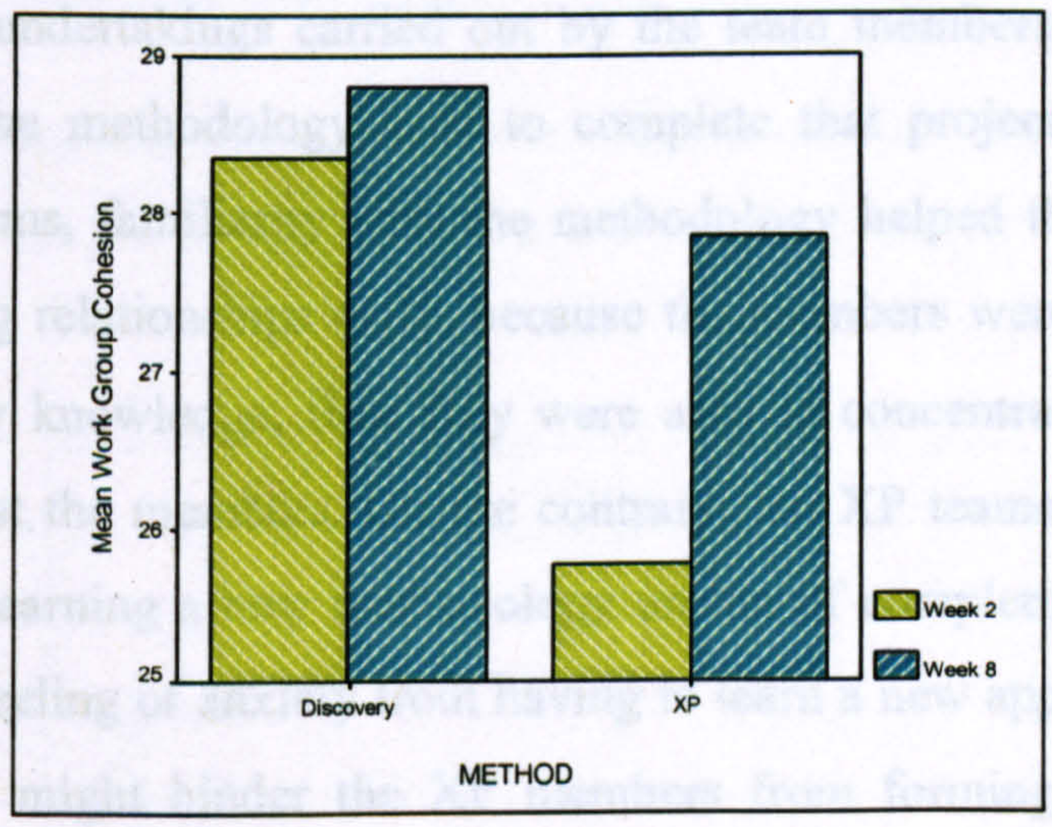


Figure 6-3: Bar Graph comparing the work group cohesion level between the two methodologies during week 2 and week 8 (Software Hut 2004).

Before the Treatment

To compare the mean scores of the work group cohesion between the XP teams and the Discovery teams, an **Independent-sample t-test** was conducted. The result

showed **a significant difference** in the cohesive level for the Discovery teams[1](M= 28.34, SD = 3.45), and the XP teams [M=25.77, SD = 5.33; t (63) = 2.26, p= 0.028) during the second week. However, the magnitude of the differences in the mean score was very small (calculated eta squared = 0.07). The analyses indicated that at the beginning of the project, the Discovery members were closer to each other than the XP teams. The factors that might have contributed to this condition were the familiarity of the members and the familiarity with the future tasks. In analysing the combination of the team members, it was observed that both the XP teams and the Discovery teams consisted of a mixture of 'natural' and 'constructed' teams. Natural groups consisted of members who have been attending similar modules before attending the Software Hut module, or members living in the same residential hall. The 'constructed' focus groups were teams which have fewer team members and required the managers' assistance to form a complete team.

The next factor that was analysed was the future task. The term future task referred to all of the undertakings carried out by the team members, which included the project and the methodology used to complete that project. In the case of the Discovery teams, familiarity with the methodology helped the members to foster better working relationships early, because the members were not concerned with acquiring new knowledge, thus they were able to concentrate on developing the bonds amongst the members. On the contrary, the XP teams were faced with the challenge of learning a new methodology on top of completing their first real life project. The feeling of anxiety from having to learn a new approach was one of the elements that might hinder the XP members from forming better bonds at the beginning of the project.

After the Treatment

After the treatment (using the different methodologies), the result of the test showed **no significant differences** in the work group cohesion level between the

[1] Data produced from Out T-test.sav

Discovery teams (M=28.81, SD=4.27) and the XP teams [M=27.84, SD= 5.47, p=0.43).

Referring to Figure 6-3, the bar graph illustrates that the cohesion level amongst the XP teams increased dramatically compared to the Discovery teams. To test whether there was a significant change in the group cohesion level following the intervention of methodologies, the **paired-samples t-test** was used. The result of the statistical test showed that there was a **statistically significant increase** in the work group cohesion level for the XP teams from the second week (M = 25.77, SD = 5.33) to week 8 (M=27.84, SD= 5.47), t (31) =-2.82, p<0.009. The calculated eta squared = 0.21, indicating a small effect size. As depicted by the gradual increase for the Discovery teams, there was **no significant increase** for work group cohesiveness amongst these teams from week 2 (M = 28.34, SD = 3.45) to week 8 (M=28.81, SD = 4.27), t (32) = -0.64, p < 0.53.

Comparison according to project.

The second comparison study was conducted, to analyse the work group cohesion level between the two methodologies in different projects.

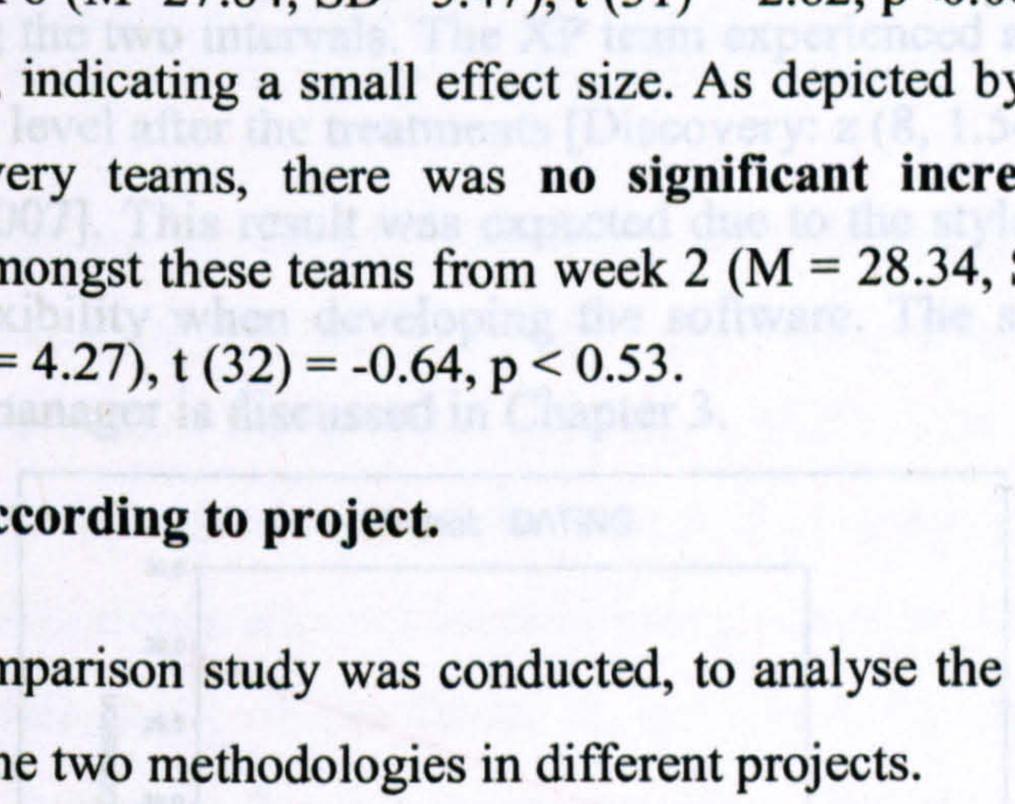


Figure 6-4 Line graph illustrating the differences in cohesion level between the XP teams and Discovery teams in the Fizzilink project.

In the Fizzilink project, the line graphs for both the Discovery and the XP teams are very similar. A statistical test, the **Mann-Whitney test** was conducted and showed **a significant difference** between the two methodologies at week 2, with the Discovery teams experiencing a higher group cohesion level (N= 9, M_1=29.11, SD= 2.26) than the XP teams [(N=10, M_1 =22.8, SD=3.85), z= 3.163, p=0.002]. There was also **a significant difference** between the two methodologies during week 8, the Discovery teams experienced a higher group cohesion level (M_2=31.89, SD= 3.26) than the XP teams [(M_2=26.60, SD= 4.00), z= 2.664, p = 0.008]. The **Wilcoxon Signed Rank test** was used to compare the difference between the readings during the two intervals. The XP team experienced **a significantly higher** group cohesion level after the treatments [Discovery: z (8, 1.544) = 0.123 and XP: z =(9, 2.68) = 0.007]. This result was expected due to the style of management that encouraged flexibility when developing the software. The style of managing the project by the manager is discussed in Chapter 3.

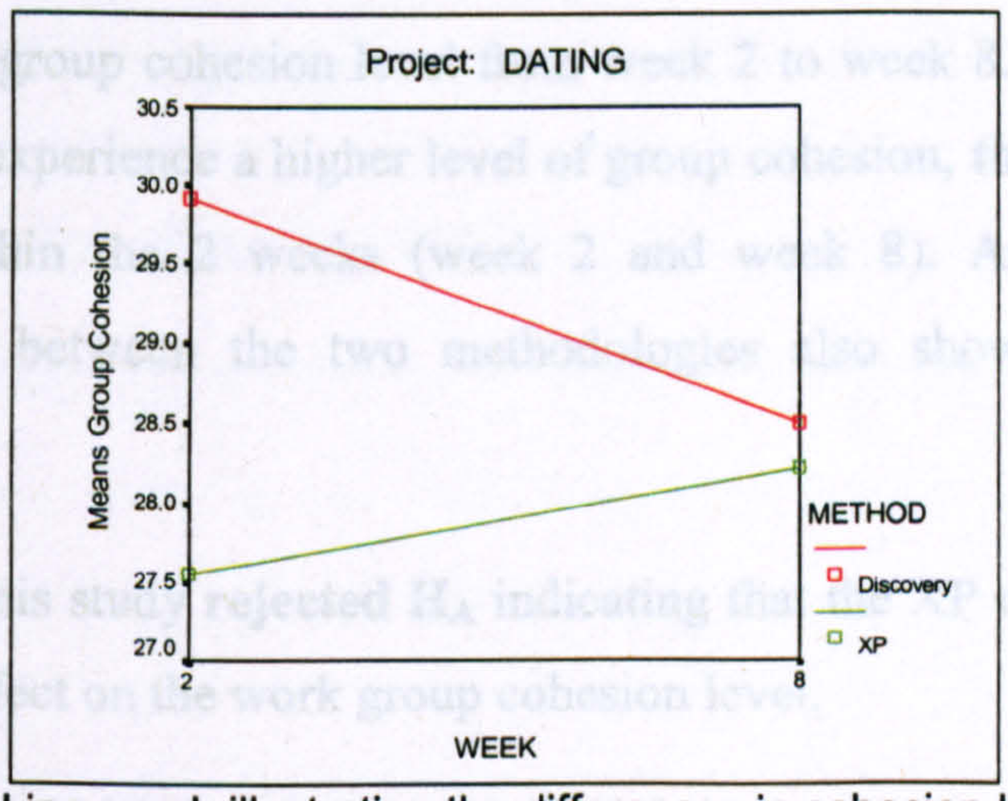


Figure 6-5 Line graph illustrating the differences in cohesion level between the Discovery teams and XP teams in the Dating project.

In the **Dating project** (Figure 6-5), the test conducted **does not show any significant differences** within week 2 and week 8 for both methodologies and also between the two methodologies. The Discovery team experienced (N=12, M_1= 29.92, SD= 2.97) (M_2=28.22, SD= 7.68) while the XP teams experienced (N= 9, M_1=27.56, SD=5.57) (M_2=28.22, SD=7.68). Interestingly, the Discovery teams experienced a decrease in the group cohesion level as they progressed towards the end of the project.

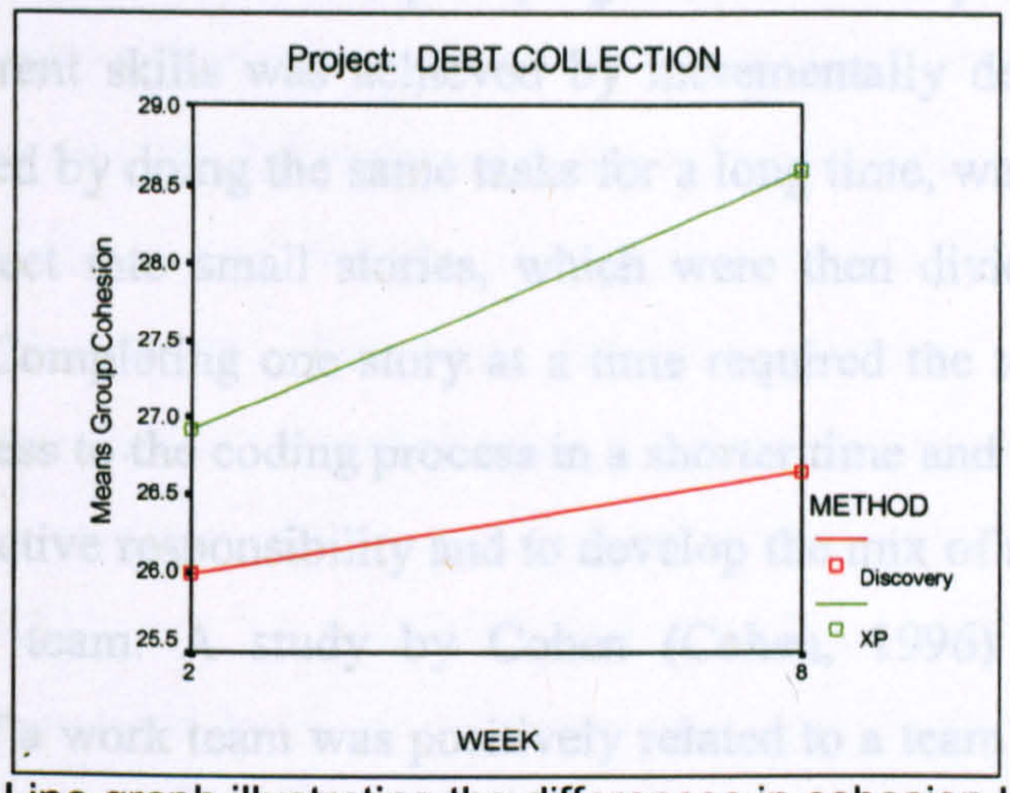


Figure 6-6 Line graph illustrating the differences in cohesion level between XP teams and Discovery teams in Debt Collection project

A different pattern was observed in the **Debt Collection project** (Figure 6-6). Both the Discovery teams (N=11, M_1= 26.00, SD= 3.68) (M_2= 26.64, SD= 4.78) and the XP teams (N= 12, M_1= 26.92, SD= 5.55)(M_2= 28.58, SD= 4.85) experienced an increase in the group cohesion level from week 2 to week 8. Even though the XP teams seem to experience a higher level of group cohesion, **the difference was not significant** within the 2 weeks (week 2 and week 8). Analysis of the group cohesion level between the two methodologies also showed **an insignificant difference.**

The results in this study **rejected H_A** indicating that the XP methodology does not have a direct effect on the work group cohesion level.

6.2.3 Discussion

The results showed the influence of an "approach of work" on the group cohesion, even though there was no significant relationship between the two variables. The result was in line with the general theories on work design, which suggested that the group can humanize work, with group tasks designed to create meaningful work. The results have shown that a variety in the tasks encouraged the team to learn and to use the different skills, and to rotate between jobs which reduced the boredom of any repetitive work. In this study, a job rotation was achieved by

changing the tasks between the pair programmers. The process of learning and using the different skills was achieved by incrementally developing the project. Boredom, caused by doing the same tasks for a long time, was avoided by dividing the whole project into small stories, which were then divided amongst the pair programmers. Completing one story at a time required the team to progress from the design process to the coding process in a shorter time and enabled them to share a sense of collective responsibility and to develop the mix of skills necessary for an effective work team. A study by Cohen (Cohen, 1996) has shown that the effectiveness of a work team was positively related to a team's belief in a common task and this was a mutual belief. The improvement made to the coaching approach in the second and third study showed that job design, in this study the methodology; does have an effect (if not a direct effect) on the group cohesion. This was supported by a study on teamworking which discovered that training was a necessity to promote an employee's involvement, and commitment and cohesiveness in the workplace (Levine 1995).

6.3 Case Study – Genesys 2003

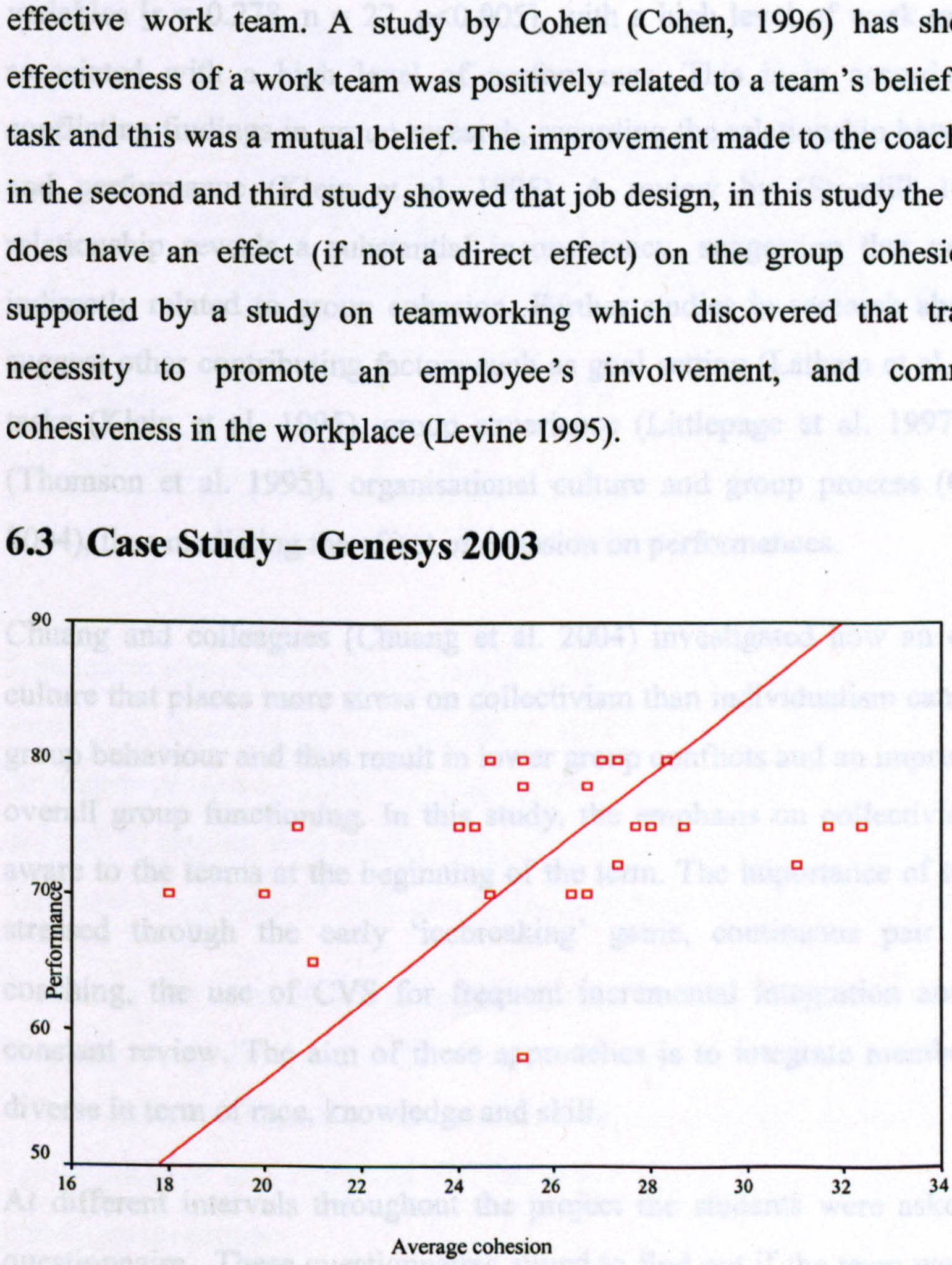


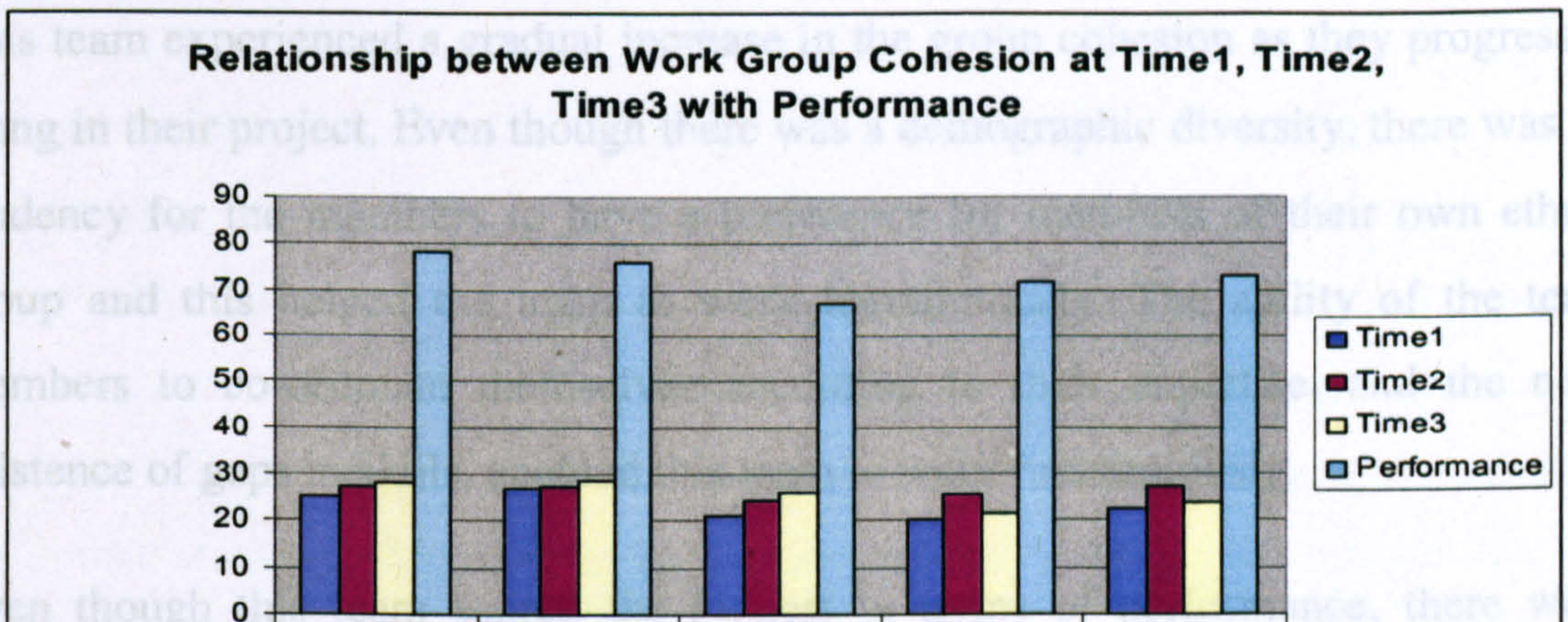
Figure 6-7: Average Cohesion and Performance

A scatter plot (Figure 6-7) was plotted to examine the relationship between the average work group cohesion and the performance. The average cohesion level was used to study the relationship, rather than the cohesion level at Time 3, because this data represented the overall feeling of cohesiveness amongst the team members. The relationship between the two variables (average cohesion level) and (performance) was then investigated using the Pearson product-moment correlation coefficient. The analysis shows a small positive correlation between the two variables [$r = 0.278$, $n = 27$, $p<0.005$], with a high level of work group cohesion associated with a high level of performance. This is in accordance with the conflicting findings in group research, regarding the relationship between cohesion and performance (Klein et al. 1995). A review by (Stogdill 1948) on this relationship reveals a substantial inconsistency, suggesting that performance is indirectly related to group cohesion. Further studies in research about the group suggest other contributing factors such as goal setting (Latham et al. 1990), group tasks (Klein et al. 1995), group experience (Littlepage et al. 1997), personality (Thomson et al. 1995), organisational culture and group process (Chuang et al. 2004), thus mediating the effect of cohesion on performances.

Chuang and colleagues (Chuang et al. 2004) investigated how an organisational culture that places more stress on collectivism than individualism can influence the group behaviour and thus result in lower group conflicts and an improvement in the overall group functioning. In this study, the emphasis on collectivism was made aware to the teams at the beginning of the term. The importance of teamwork was stressed through the early 'icebreaking' game, continuous pair programming coaching, the use of CVS for frequent incremental integration and the client's constant review. The aim of these approaches is to integrate members who were diverse in term of race, knowledge and skill.

At different intervals throughout the project the students were asked to fill in a questionnaire. These questionnaires aimed to find out if the team were working as a cohesive unit and what factors were preventing them from becoming an effective unit. The students were also asked to complete an online version of the MBTI, and

this data was used to analyse the personality types of the individual team members. In Figure 6-8, the three times represent the levels of cohesiveness when the questionnaires were handed out and the performance represents the average overall performance for a particular team.



	A1	A2	A3	A4	A5
Time1	25	26.75	21.25	20.33	22.6
Time2	27.5	27.5	24.25	25.33	27.4
Time3	28	28.75	25.75	21.67	23.8
Performance	78.33	76	67	71.67	73.2

Figure 6-8 Work Cohesion and Performance

Five teams were studied throughout the project; they were all software development teams. A decision was made not to study the marketing or the research and development teams as they were not working on any projects therefore it would be difficult to explain their results in the context of an SE project. It was observed that there were 2 patterns of work group cohesion experienced by the Genesys teams:

Pattern1: The team members experienced an increase in the cohesion level as they progress along the project. This pattern is consistent with previous studies that cohesiveness amongst the team members increases with time due to the increased communication and improved coordination (Teasley et al. 2002) during the group development process.

Pattern2: The team experienced an increase in the group cohesion level during the first term but the cohesion level started to decrease during the second term. Overall,

the cohesion level is higher at the end of the year than when they started the project. Possible factors to be considered for this pattern are group diversity (Webber et al. 2001) and team potency (Pearce et al. 2002).

Team A1

This team experienced a gradual increase in the group cohesion as they progressed along in their project. Even though there was a demographic diversity, there was no tendency for the members to have a preference for members of their own ethnic group and this helped the team to work harmoniously. The ability of the team members to co-ordinate themselves according to their expertise, and the non-existence of gaps in skills, enabled this team to work harmoniously.

Even though this team scored the highest in terms of performance, there were glimpses of a task related conflict during the focus group discussion. Task related conflict refers to any disagreement amongst the team members about task issues, including the nature and importance of the task goal. Studies on task conflicts have shown both positive (Jehn 1995), (Lovelace et al. 2001) and negative (Liang et al. 1995) results. In the case of team A1, the conflict was more on the practice of pair programming rather than on the project related task. The experience of using XP in previous projects enabled some of the members to recognise the strengths and weaknesses of its practice. These are the members who appointed themselves as XP enforcers, creating a slightly uncomfortable environment for those who were reluctant to work in pairs all the time. Nevertheless, the enforcers managed to make the pair programming practice compulsory, despite the reluctance; hence this could have contributed to the slight increase in the group cohesion level. The task conflict in this case seems to have a more favourable effect on the performance of the team.

Another factor that might contribute to this pattern of cohesiveness is that the group recognised some of their members' expertise. Prior group experience allowed group members to understand the ways in which other members might be able to

contribute and this had the advantage of increasing the group cohesiveness and performance. Studies have found that a team's experience leads to more trust in the members' expertise and the group that were trained together were better able to recognise another members' expertise (Liang et al. 1995).

Team A2

The group diversity for this team is more demographic than functional. The cohesiveness amongst the team members was very high from the start of the project. During the two focus group discussions, it could be sensed that the members were very comfortable with each other. The maturity amongst some of the members helped to foster a better interpersonal relationship because these members are known to occasionally break the 'developers block' during the project meeting by taking a break at the nearest pub or discussing topics other than those which are task related.

Examinations of the XP practices used by this team revealed that they were the team that had applied the most practices. The presence of a member who had used some of these practices before helped the team to adhere to the selected XP practices as closely as possible. To practice pair programming needs an effort on the part of all the team members and in the case of this team, members reported that pair programming activities had been an enjoyable way of working. The members found that the changing of partners amongst the pairs can be easily accommodated. This may be due to the improvising approach exercised by the team; whereby if the time does not permit them to change partners at certain times, the members exchanged part of the system. The use of CVS, a version management tool also helped the team to foster collective ownership amongst the members. The interpersonal relationship between the members helped the team to appreciate the flexibility of the methodology by modifying it to suit the team and the project.

Studies by Klein et al (Klein et al. 1995) have shown that the relationship between cohesiveness and performance is mediated by the group processes and the goal processes. In these studies, they reported that cohesion is very effective in getting groups to set difficult goals and remain committed to those goals. These studies have shown that cohesiveness was significantly related to commitment and to goal setting regardless of whether it is a self-group setting or an organisational setting. The studies have also shown that cohesive groups are more effective in attaining those goals than non-cohesive groups but the goals of cohesive groups are not necessarily goals for high performance or performance related goals.

Team A3

This team's diversity is in terms of demographics, functionality and MBTI type. The task related diversity that is present in this team should have been able to propel it to perform better but the tendency to task avoidance by a specific member of the team also had an effect on the group's effectiveness. The added political problem faced by the client, in addition to the SE problems, seemed to be the worst case scenario for any developers to face. Nevertheless, the group managed to progress with their project, albeit slowly. The presence of a sensing-feeling personality in this team was the driving force in keeping the group going as a team. This behaviour was typical of a sensing-feeling type as they managed to act as a peace broker both in the positions of inter-team understanding and in a spirit of reconciliation with other members of the team. This sensing-feeling personality along with another member who had experienced working with the required methodology, even though not adhering to them properly before, helped the team to use as many XP practices as possible. Previous research has shown that task experience can lead to ability-based enhancement when conditions promote transfer of specific knowledge or strategy (Littlepage et al. 1997).

This team experienced the first pattern of group cohesiveness. The factor that might contribute to this pattern is the effort made by the team members to adhere to the practices that are easier to use, such as pair programming and collective ownership.

Due to the composition of the team and with a very few timetable clashes, members managed to meet regularly in order to apply the activities that improved cohesiveness amongst them. Previous experience in working in pairs helped the members to progress to the next stage of pair programming. That is the changing of partners between the pair. In addition to the method of doing pair programming, the team also practiced having different test partners. These interactions are possible in reducing any emotional conflict that might arise in the team. The decision by the team to use CVS, which managed the various versions of the project, was also a factor to be considered in contributing to the cohesiveness of the group. Finally, the decision made by the consultants (primarily Prof. Mike Holcombe the head of the VT research group) to continue as a generic project can be considered as a contributing factor towards group cohesiveness.

Team A4

A series of studies by Littlepage et al (Littlepage et al. 1997) has shown that task experience and group experience can lead to a higher group performance. Task experience refers to the experience with similar tasks, and group experience refers to the experience of working with group members. During the focus group discussion, the group explained that even though three of them had some limited working experience before joining the company, those experiences did not include the challenge of working as a team as demanded by the XP approach. Cohesiveness amongst the members started to diminish as the project became more complex and the members reverted to working individually in order to complete the project on time. The above factors might have contributed to the group experiencing the second pattern of cohesiveness.

Team A5

The team members were diverse in terms of demographics and skills. The differences in their experience on task and group work contributed to the team experiencing the second pattern in the group cohesiveness. The focus group interviews conducted with the team revealed that this team did not function as well

as the other teams because of the loafing behaviour and effort-avoidance of particular team members. This in turn led to the feeling of resentment towards some of the group members. Research has discovered that teamwork does not always increase the participants' mindful engagement in learning and thus improve its outcomes, and the existence of social-psychological effects such as understanding and empathy can also debilitate team performance (Solomon et al. 1989).

The discussion on XP applications identified that the team did not truly adhere to most of the XP practices when developing their projects. Pair programming was not fully practiced whereby the team members were supposed to pair as much as possible and in addition to changing the partners. The nearly non-existence of this practice amongst some of its members is a possible contribution to the second cohesive pattern experienced by this team.

6.4 XP practices and the Work Group Cohesion

The Pearson correlation test was conducted to find out the relationship between the level of the work group cohesion and the number of XP practices used. In all of the studies (Study 1, Study 2 and Study 3), there was no significant relationship between the two variables, indicating that the number of XP practices has no direct impact on the level of work group cohesion experienced by each team member. The result indicates the possibility of a mediating factor, such as the group development process, that might contribute to the results discussed in Section 5.2.1.1 and Section 5.2.1.2.

6.5 Discussion

In this study, the results for the sample as a whole show that there was no direct impact of the XP methodology on the work group cohesion, even though previous research has shown a positive relationship between job design and group effectiveness and group cohesion. More research is needed to clarify the different

activities associated with each practice and there is a need to find the correct scale to conduct this study.

CHAPTER 7

THE IMPACT OF XP ON WORK RELATED WELLBEING

7.1 Introduction

This chapter describes the empirical studies, which address the aspect of work related well being amongst members of the software development teams. The question of interest is whether the XP methodology has any distinct effect on the well being of the software developers. To understand how XP can increase enthusiasm, results are interpreted with references to cognitive, affective and managerial properties of the practices studied. The result needs further investigation on the effects of individual practice on the wellbeing of Software Engineering (SE) teams.

Section 2 outline the hypotheses to be tested while Section 3 discusses the experiment carried out in the SSEO. Following the discussion of the difference in wellbeing level between the two studies, Section 4 summarizes by discussing ways the practices affect the developers' wellbeing.

7.2 Work Related Well being

The study described in this section addressed the humanistic aspects of Extreme Programming with particular emphasis on the wellbeing of developers in three different levels of project dynamism and how this was influenced by the intrinsic relationship of the 12 practices in Extreme Programming (XP). In this study, the four factors examined were the software development teams, the methodologies used, the dynamism of projects developed and the wellbeing variables (Warr 1990) measured against the teams.

The first purpose of this study was to assess empirically whether there are any differences in the well being level between the XP teams and the Discovery teams. The hypothesis was defined as follows:

H_A: The XP teams will experience a better level of well being compared to the designed-based teams.

The second purpose of this study was to explore the relationship between the 12 XP practices and the well being level of the development teams. The hypothesis was defined as follows:

H_B: The higher number of XP practices used is associated with a higher level of well being, experienced by the software team members.

Job related anxiety, depression, contentment and enthusiasm were measured using the 12–items anxiety-contentment and depression-enthusiasm scales developed by Warr (Warr 1990). The scale was used to measure the extent to which the members were anxious or contented, depressed or enthusiastic about their project at varying intervals. Respondents were asked to think of the past few weeks and to indicate the extent to which they felt gloomy, calm, uneasy, enthusiastic, cheerful, worried, contented, tense, depressed, optimistic, relaxed and miserable (APPENDIX 3-C).

It was discovered that the effects of the agile methodology were positively related to the level of the project' dynamism and the more XP practices were used to develop the software the better was the feeling of wellbeing. The contribution of this study is a set of tested hypotheses, along with the arguments in the form of the supporting evidence.

7.3 Comparison Studies

To test the first hypothesis, two comparison studies were conducted on the development teams in the Software Hut in the year 2003 - 2004. The wellbeing

scale consists of four variables, which are related to each other – anxiety-contentment and depression-enthusiasm (APPENDIX 3-D). The first test conducted examined all of the wellbeing items as a total score. Then to understand further the impact of the methodology, four hypotheses regarding its impact on the constructs were developed and tested separately. The hypotheses to be tested were as follows:

H_1: Developers using the XP methodology will experience lower levels of anxiety than those using the Discovery methodology.

H_2: Developers using the XP methodology will experience higher levels of contentment than those using the Discovery methodology.

H_3: Developers using the XP methodology will experience lower levels of depression than those using the Discovery methodology.

H_4: Developers using the XP methodology will experience higher levels of enthusiasm than those using the Discovery methodology.

The reading on the wellbeing constructs were taken during two specific intervals. There were an 8-weeks gap for the Software Hut 2003 and a 6-weeks gap for the Software Hut 2004.

7.3.1 Software Hut 2003

The first batch of data was collected from students in the Software Hut 2003. This batch of students had the knowledge and early experience of using the Discovery method. **Independent sample t-test** was used to compare the total mean score for well being variables and the result showed no significant difference between the intervals for both methodologies; Discovery (N=20, M_1= 46.60 SD = 5.46; M_2 = 37.25, SD = 9.96) and XP (N=19, M_1=46.74, SD = 5.98; M_2 = 37.32, SD =7.79). The similarity in the mean score for wellbeing is illustrated in Figure 7-1

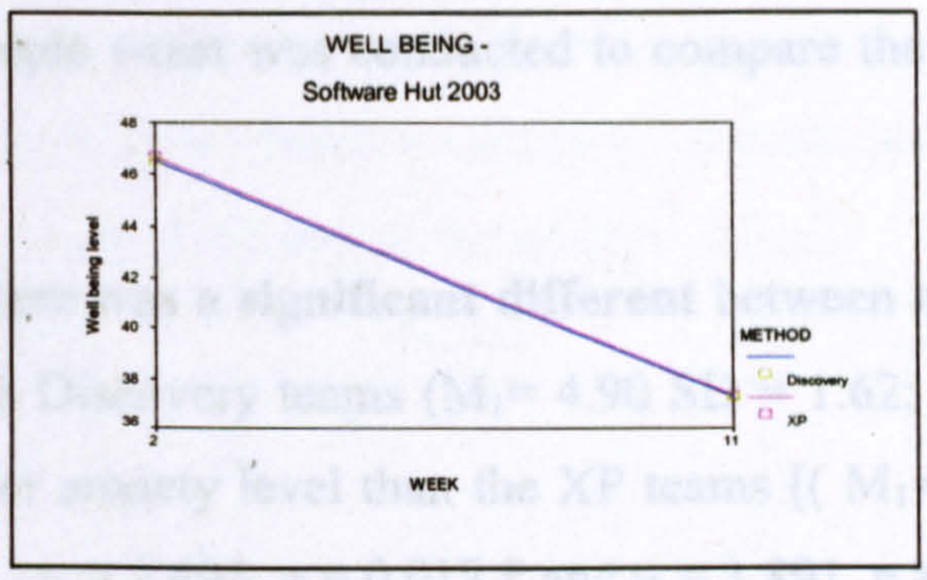


Figure 7-1The well being measure between the two methodologies (Software Hut 2003)

To identify the effects that each methodology had on the well being measure, the scale was divided to measure four different factors – anxiety, contentment, depression and enthusiasm (Figure 7-3).

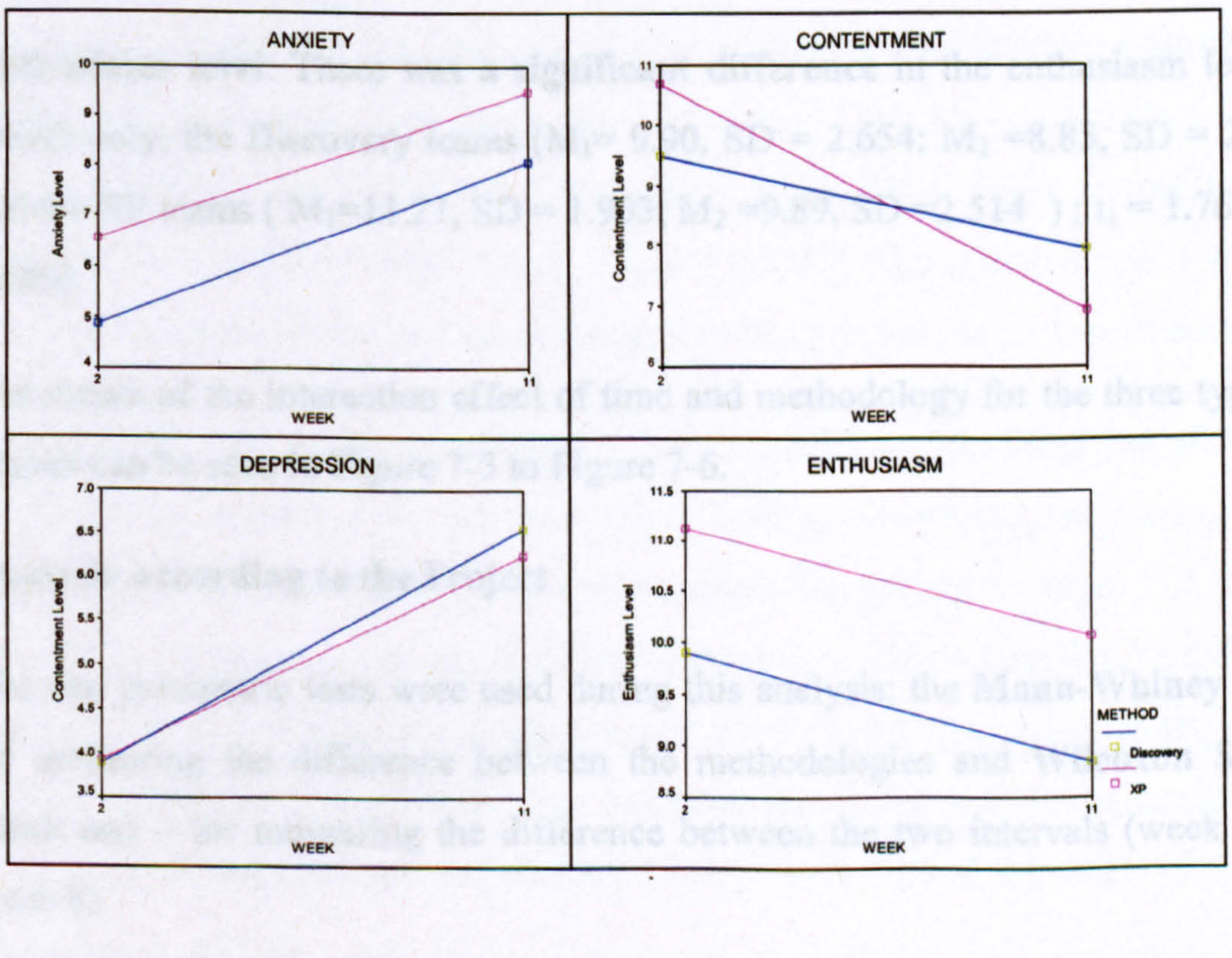


Figure 7-2 The interaction of the methodology and the time for four different variables in the wellbeing scale (Software Hut 2003)

Independent sample t-test was conducted to compare the mean score of the two methodologies.

Anxiety level. There was **a significant different** between methodologies at week2 and week 11 with Discovery teams (M_1= 4.90 SD = 1.62; M_2 = 8.10, SD = 3.24) experiencing lower anxiety level than the XP teams [(M_1=6.68, SD = 2.69; M_2 = 10.00, SD =3.02) ; t_1 = 2.495, p = 0.018 * and t_2 = 1.891, p = 0.066].

Contentment level. There was **no significant different** in contentment level during the two intervals; Discovery teams (M_1=9.50 SD =3.035; M_2 = 7.95, SD = 2.911) and XP teams (M_1=10.32, SD = 2.358; M_2 =6.74, SD = 2.130).

Depression level. There was **no significant different** in the depression level between Discovery teams (M_1=3.90 SD = 2.29; M_2 = 6.55, SD = 4.045) and XP teams (M_1= 4.11, SD = 1.912 ; M_2 = 6.32, SD = 3.667).

Enthusiasm level. There was **a significant difference** in the enthusiasm level at week2 only, the Discovery teams (M_1= 9.90, SD = 2.654; M_2 =8.85, SD = 2.159) and the XP teams (M_1=11.21, SD = 1.903; M_2 =9.89, SD =2.514) ; t_1 = 1.764, p = 0.086].

The nature of the interaction effect of time and methodology for the three types of project can be seen in Figure 7-3 to Figure 7-6.

Analysis According to the Project

The non parametric tests were used during this analysis; the **Mann-Whiney test** – for measuring the difference between the methodologies and **Wilcoxon Signed Rank test** – for measuring the difference between the two intervals (week 2 and week 8).

Anxiety to project

Figure 7-3 graphs the direction and magnitude of the effects of the methodologies on the anxiety level for the three types of project. Anxiety refers to the feeling of

being tensed, worried and anxious in developing software. In the most dynamic project, it was observed that students using the XP methodology seem to experience a lower level of anxiety at the end of the project.

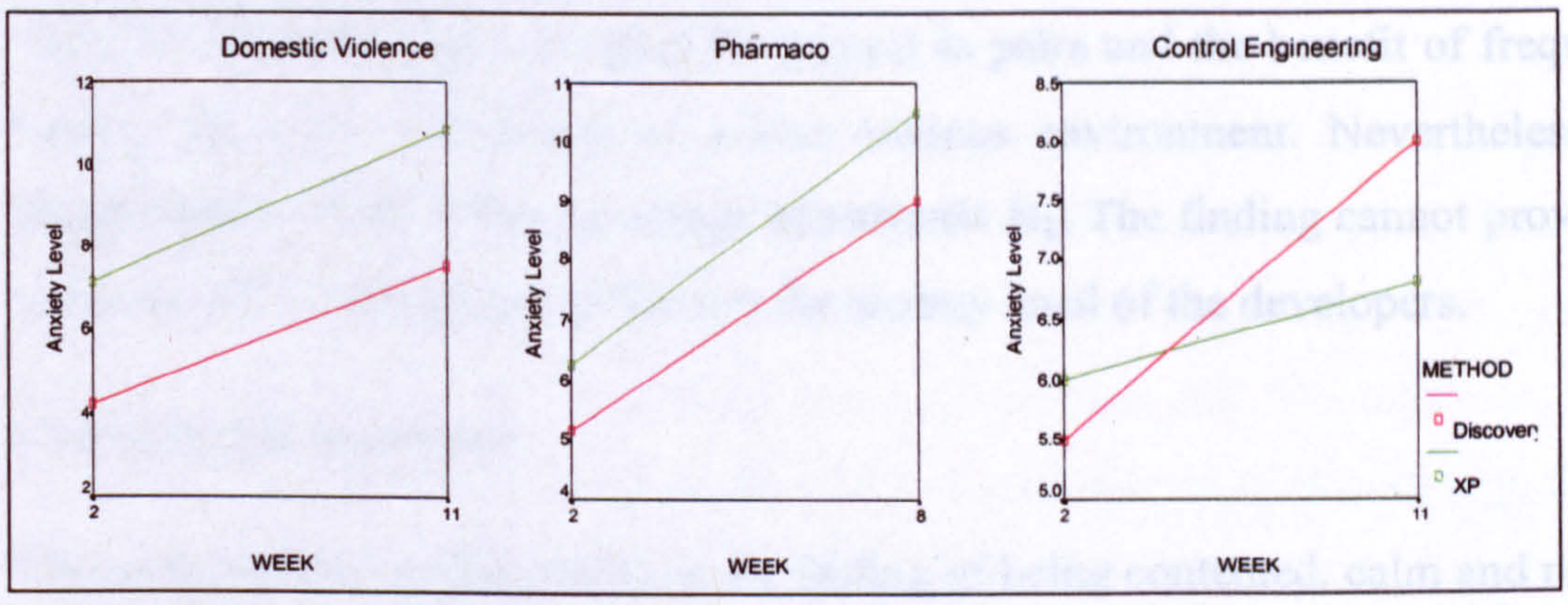


Figure 7-3 The interaction of methodology and time for anxiety towards project (Software Hut 2003)

For both the Domestic Violence and Pharmaco projects, the anxiety level of the XP teams was higher than the Discovery teams but the statistical test conducted showed that teams developing the Domestic Violence project only exhibit **a significant difference** [Discovery teams (N=8, M_1= 4.25 SD = 1.03; M_2 = 7.50, SD = 3.46) ; XP teams (N=7, M_1=7.14, SD = 2.79; M_2 = 10.86, SD = 1.68)] during the two intervals [(z_1 =2.20, p = 0.027 ; z_2 = 2.1, p = 0.036)]. The possible reason for this result is due to the Discovery methodology treatment in the earlier course module.

However, the XP teams experienced a higher initial anxiety level for all of the projects. The teams had the disadvantage of having to learn the new methodology within a certain time frame in order to complete the project on time. The lack of knowledge had been perceived as a disadvantage in a competition that awarded the wining team with a computer laptop for each member. Previous research has shown that the perceived lack of knowledge has been associated with the increase in the anxiety level amongst the team members (Axtell et al. 2002; Clegg et al. 1997).

There was a **significant positive relationship** between methodology and anxiety for the Domestic Violence project, indicating that for a stable project a more structured methodology such as the Discovery method was more suitable. This is to be expected as shown by previous research (Norman et al. 2003).

However in the most dynamic project, where the requirements were vague, the ability of the XP methodology to capture only partial requirements through the use of a simple story helped to prevent a drastic increase in the anxiety level. In addition, the advantage of coding the project in pairs and the benefit of frequently testing the code contributed to a less anxious environment. Nevertheless, the **insignificant result failed to accept hypothesis H_1**. The finding cannot prove that using the XP methodology will lower the anxiety level of the developers.

Contentment to project

The contentment variable refers to the feeling of being contented, calm and relaxed when developing and completing the software project. A similar pattern was observed in all of the three projects (Figure 7-4).

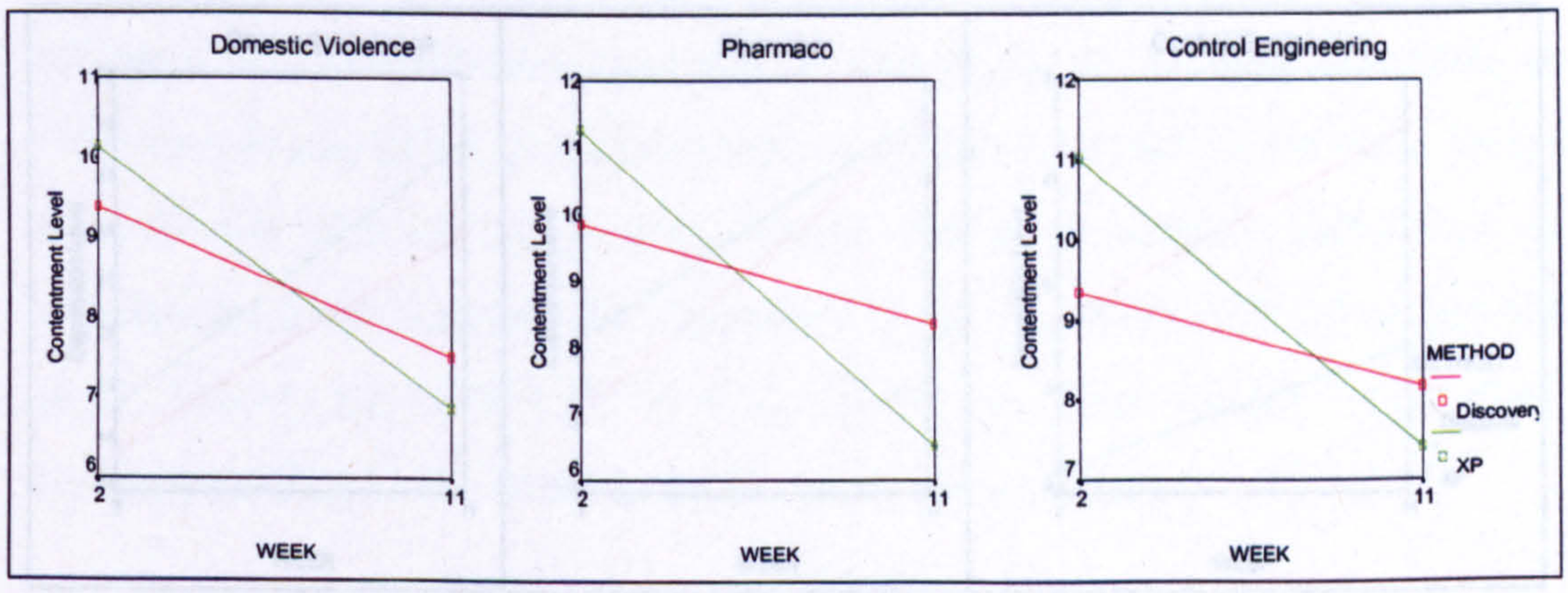


Figure 7-4 The interaction of methodology and time for contentment towards project. (Software Hut 2003)

Both teams experienced a decrease in the level of contentment at the end of the project. Initially the XP members seemed to experience a higher level of contentment but the rate of decreasing was much faster than the Discovery teams. The statistical test conducted showed that the XP teams developing the Pharmaco project (M_1=11.25, SD=0.96; M_2=6.50, SD=2.08, z=1.84, p=0.06) and the Control Engineering project (M_1=11.00, SD=2.55; M_2=7.4, SD=2.07, z=2.041, p=0.041) showed a significant decrease in the contentment level between the two intervals. Nevertheless, the comparison between the methodologies showed **no significant difference, therefore the hypothesis (H_2) was rejected**. The finding failed to

prove that using the XP methodology will cause the contentment level of the developers to be higher than the Discovery teams.

Depression to project

Depression refers to the feelings of gloom, being miserable and depressed towards the project. The XP teams developing the Domestic Violence and the Pharmaco projects experienced a higher depression level at the beginning of the project, nevertheless there was no significant increase between the two intervals. In the Control Engineering project where there was a higher level of uncertainty, the XP teams experienced a lower depression level than their counterpart. The statistical test conducted showed no significant difference between the methodologies and between the intervals.

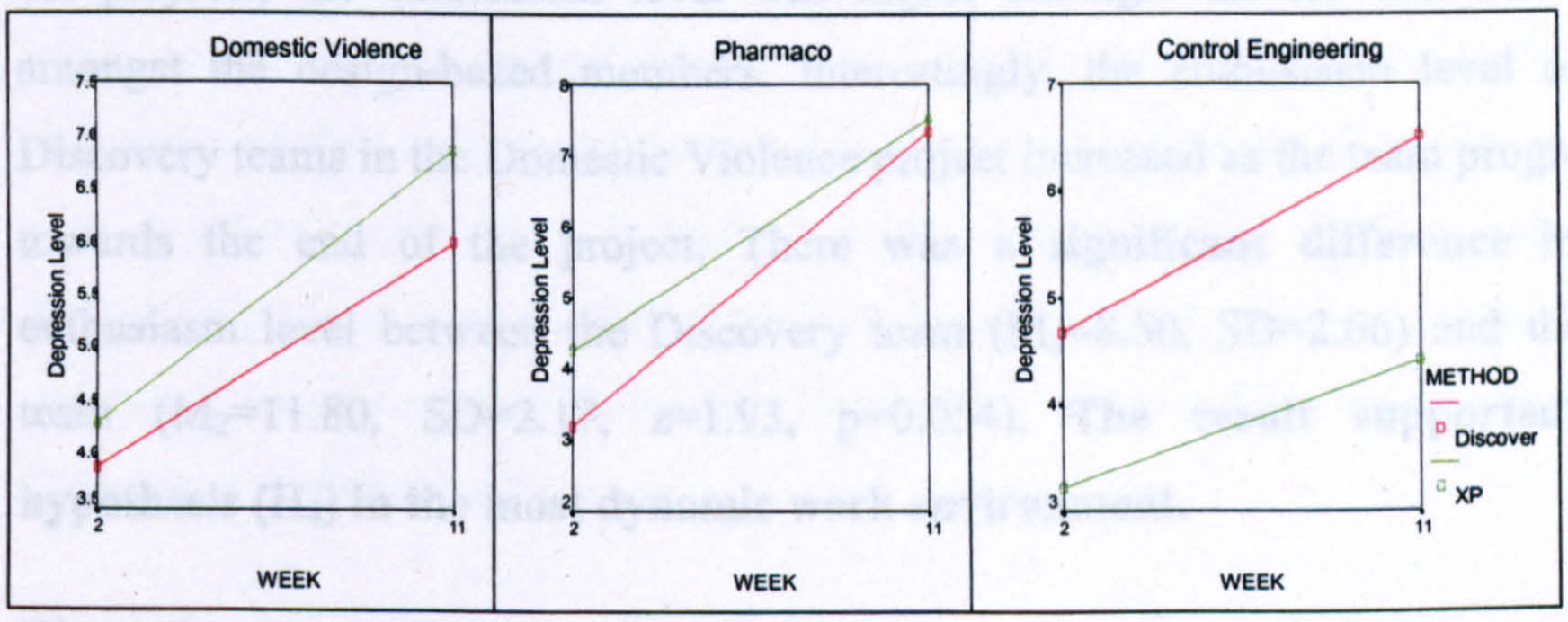


Figure 7-5 The interaction of methodology and time for depression towards project. (Software Hut 2003)

The analysis of the depression variable provided the evidence that XP members experienced a lower depression level in the most dynamic project but the statistical test conducted showed **no significant different** between the methodologies; Discovery (M_1=4.67, SD=3.141; M_2=6.50, SD=3.674) and XP (M_1=3.20, SD=0.447; M_2=4.4, SD=2.608). **Therefore the third hypothesis (H_3) is not accepted.**

Enthusiasm to project

Enthusiasm is the measurement of feeling enthusiastic, optimistic and cheerful towards the project being developed.

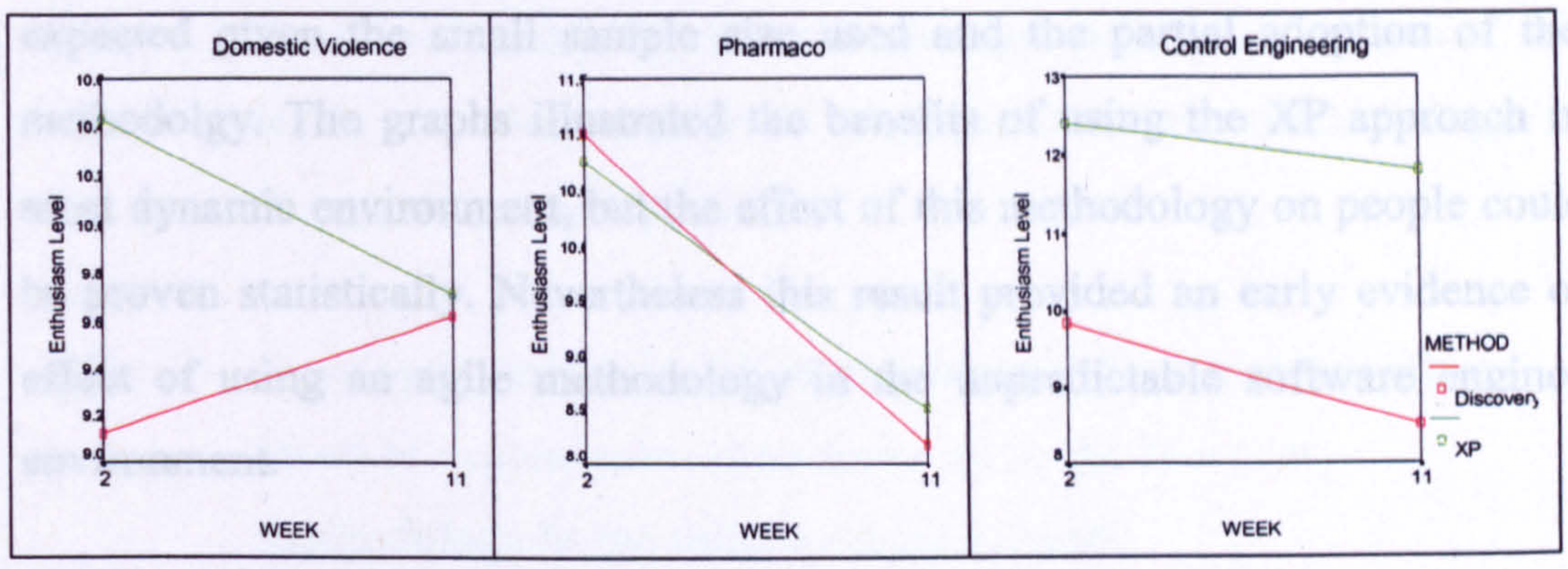


Figure 7-6 The interaction of methodology and time for enthusiasm towards project (Software Hut 2003)

The analysis of the enthusiasm variables revealed that XP members started with a higher level of enthusiasm for the Domestic Violence and Control Engineering projects but the feeling started to decrease as the project progressed. At the end of the projects, the enthusiasm level was higher amongst the XP members than amongst the design-based members. Interestingly, the enthusiasm level of the Discovery teams in the Domestic Violence project increased as the team progressed towards the end of the project. There **was a significant difference** in the enthusiasm level between the Discovery team (M_2=8.50, SD=2.66) and the XP team (M_2=11.80, SD=2.17, z=1.93, p=0.054). **The result supported the hypothesis (H_4) in the most dynamic work environment.**

Discussion

The results indicated that the anxiety level ($p < 0.05$) was significantly lower amongst the Discovery team developing the stable project and the enthusiasm level ($p < 0.10$) was significantly higher amongst the XP teams developing the most dynamic project. While the other results showed no significant difference between the two methodologies, it was clear that the XP teams experienced a higher level of wellbeing – apart from contentment – than the design-based team in the most dynamic work environment.

Statistically, the results failed to accept hypothesis (H_A) that is the XP developers will experience a higher level of wellbeing. Instead the study showed that only the last hypothesis (H_4) was accepted in the most dynamic project. The result was

expected given the small sample size used and the partial adoption of the XP methodolgy. The graphs illustrated the benefits of using the XP approach in the most dynamic environment, but the effect of this methodology on people could not be proven statistically. Nevertheless this result provided an early evidence of the effect of using an agile methodology in the unpredictable software engineering environment.

7.3.2 Software Hut 2004

The second batch of data was collected from students in the Software Hut 2004. The previous hypotheses were modified to include the different level of project dynamic. Nevertheless, this study could not include unpredictable project because unpredictable environment is an ongoing process based on the user requirement. While conducting the second experiment, it became clear to the researcher that replication of the previous experiment was not possible because all of the projects were at either stage 1 or 2, in term of project dynamic. The Fizzilink and Debt Collection projects were rated 2 to represent dynamic project and the Dating project was rated 1 to represent stable project. The rating was given, after a discussion with the respective managers.

The following hypotheses were developed to test the working environment available during this study.

H_1: Developers using the XP methodology will experience lower levels of anxiety than those using the Discovery methodology in the most dynamic project.

H_2: Developers using the XP methodology will experience higher levels of contentment than those using the Discovery methodology in the most dynamic project.

H_3: Developers using the XP methodology will experience lower levels of depression than those using the Discovery methodology in the most dynamic project.

H_4: Developers using the XP methodology will experience higher levels of enthusiasm than those using the Discovery methodology in the most dynamic project.

The hypotheses were tested using the **repeated measures ANOVA** with week as within-participant factor and intervention to methodology as between-participant factors. Since the hypotheses were that the work related wellbeing will change over time as a function of the intervention to the methodology for the different types of project, an interaction was predicted. (Figure 7-7)

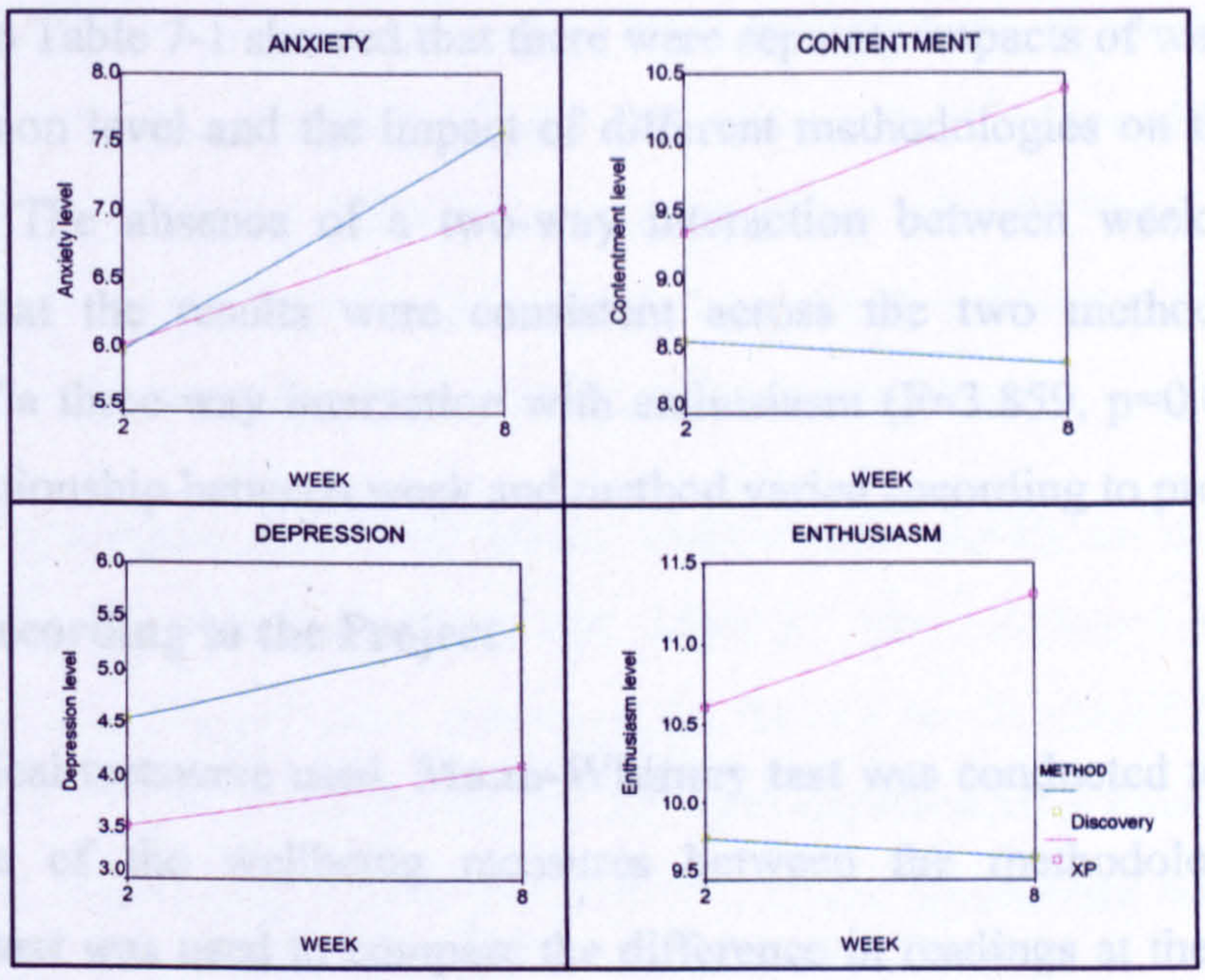


Figure 7-7 The interaction of the methodology and the time for the wellbeing variables (Software Hut 2004)

The nature of the interaction effect between week and methodology can be seen, in Figure 7-7 . The intervention of an XP methodology lowered the anxiety and the depression level and increased the contentment and enthusiasm level amongst the developers.

Variables	Anxiety		Contentment		Depression		Enthusiasm	
	F	df	F	df	F	df	F	df
Week	5.306**	1	0.000	1	4.518**	1	0.248	1
Method	0.516	1	1.740	1	7.038**	1	2.201	1
Project	3.453*	1	0.120	1	0.338	1	0.544	1
Week*Method	0.774	1	1.214	1	0.210	1	0.071	1
Week*Method*Project	0.203	1	2.239	1	0.107	1	3.859*	1

Note: p<0.1 *; p<0.05**; p<0.01***; p<0.001****

Table 7-1 Repeated measures ANOVA for all of the wellbeing variables (Software Hut 2004)

The result in Table 7-1 showed that there were separate impacts of week on anxiety and depression level and the impact of different methodologies on the depression level only. The absence of a two-way interaction between week and method indicated that the results were consistent across the two methodologies. The presence of a three-way interaction with enthusiasm (F=3.859, p=0.054) signified that the relationship between week and method varied according to project.

Analysis According to the Project

Two statistical testswere used, **Mann-Whitney test** was conducted to compare the mean score of the wellbeing measures between the methodologies and the **Wilcoxon test** was used to compare the difference in readings at the two intervals (week 2 and week 8).

Anxiety to project

Graphs in Figure 7-8 show that the anxiety level was lower amongst the XP teams in two of the three projects.

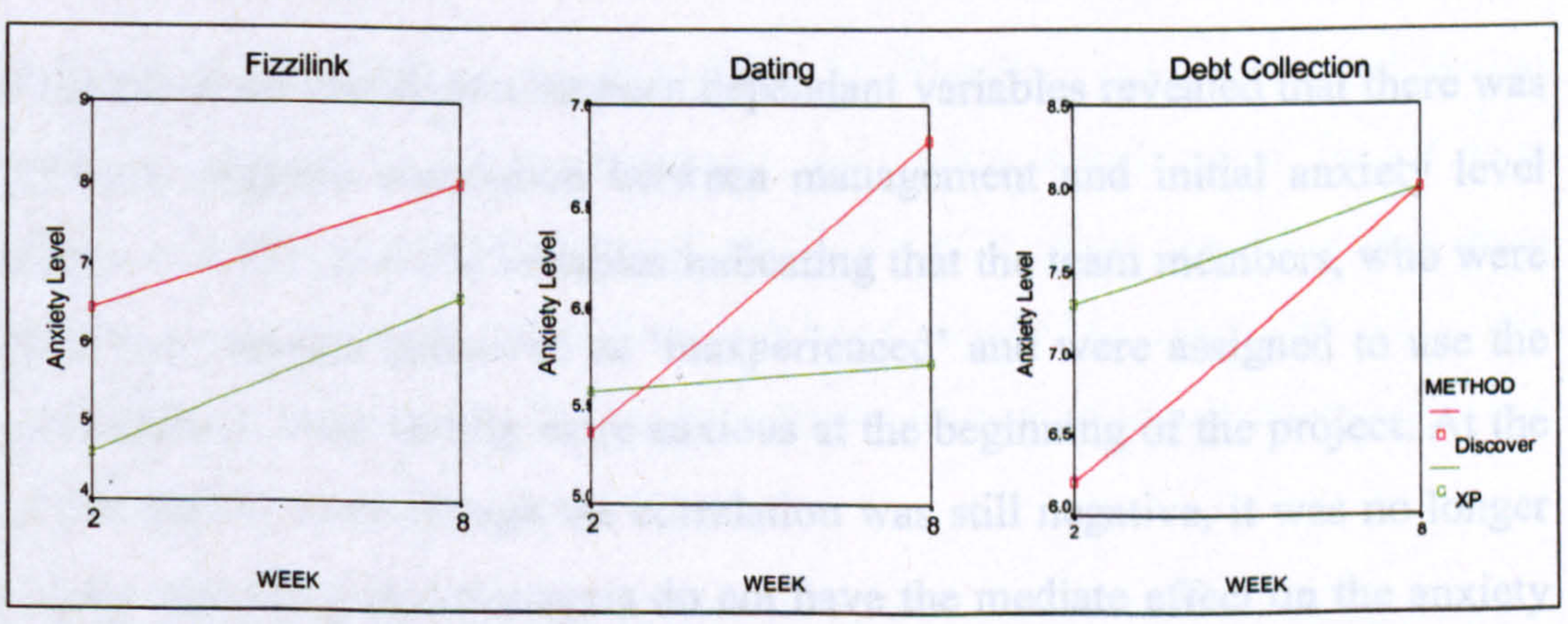


Figure 7-8 The interaction of methodology and time for anxiety. (Software Hut 2004)

As the project progressed, all of the members started to experience a higher anxiety level due to the increase in the project's difficulties and the changes in the user requirements.

In the **Fizzilink project**, there was **an initial significant increment** in the anxiety level for both the Discovery team (N= 9, M_1=6.44, SD=1.810; M_2=8.00, SD=2.550,

z=1.807, p=0.071*) and the XP team (N= 10, M_1=4.60, SD=1.506; M_2=6.60, SD=3.026, z=1.725, p=0.084*). There was also **a significance different** in the anxiety level between the methodologies (z= 2.195, p = 0.028**) but the difference was **no longer significant** at the end of the project indicating that the rate of increment for XP teams was higher than the Discovery teams.

In the **Dating project**, Discovery teams experienced a significant increment in the anxiety level (N= 9, M_1=5.36, SD=2.656; M_2=6.82, SD=2.523, z=1.657, p=0.097*) but there was an insignificant increment for the XP teams (M_1=5.57, SD=3.409; M_2=5.71, SD=2.289). Even though the XP team experienced a lower anxiety level but **the difference** at the end of the project **was not significant** between the two methodologies.

The XP teams in the **Debt Collection** project experienced a higher anxiety level but the different between the methodologies was **not significant** during the two intervals, Discovery teams (N= 10, M_1=6.20, SD=2.150; M_2=8.00, SD=3.266); XP teams (N= 13, M_1=7.31, SD=3.038; M_2=8.00,.SD=2.614).

Examination of the correlation between dependant variables revealed that there was a significant negative correlation between management and initial anxiety level (week2) [r = -0.417, p<0.05] variables indicating that the team members, who were assigned to a manager perceived as 'inexperienced' and were assigned to use the XP methodology, were feeling more anxious at the beginning of the project. At the end of the project, even though the correlation was still negative, it was no longer significant, indicating that managers do not have the mediate effect on the anxiety feeling at the end of the project. Examination of the Discovery teams showed **no significant correlation** between the two variables, indicating that the team members were not anxious or depressed when assigned to the different managers. The contributing factor that might explain this result was that previous knowledge and experience of using the Discovery methodology helped to lessen the anxiety level amongst the members.

The result rejected H_1, indicating that the methodologies have no direct effect on the level of depression of the SE teams.

Contentment to project

There were mixed findings in the contentment level of the development teams (Figure 7-9). For the Debt Collection and Fizzilink projects, the XP teams were found to experience a higher contentment level than the Discovery members.

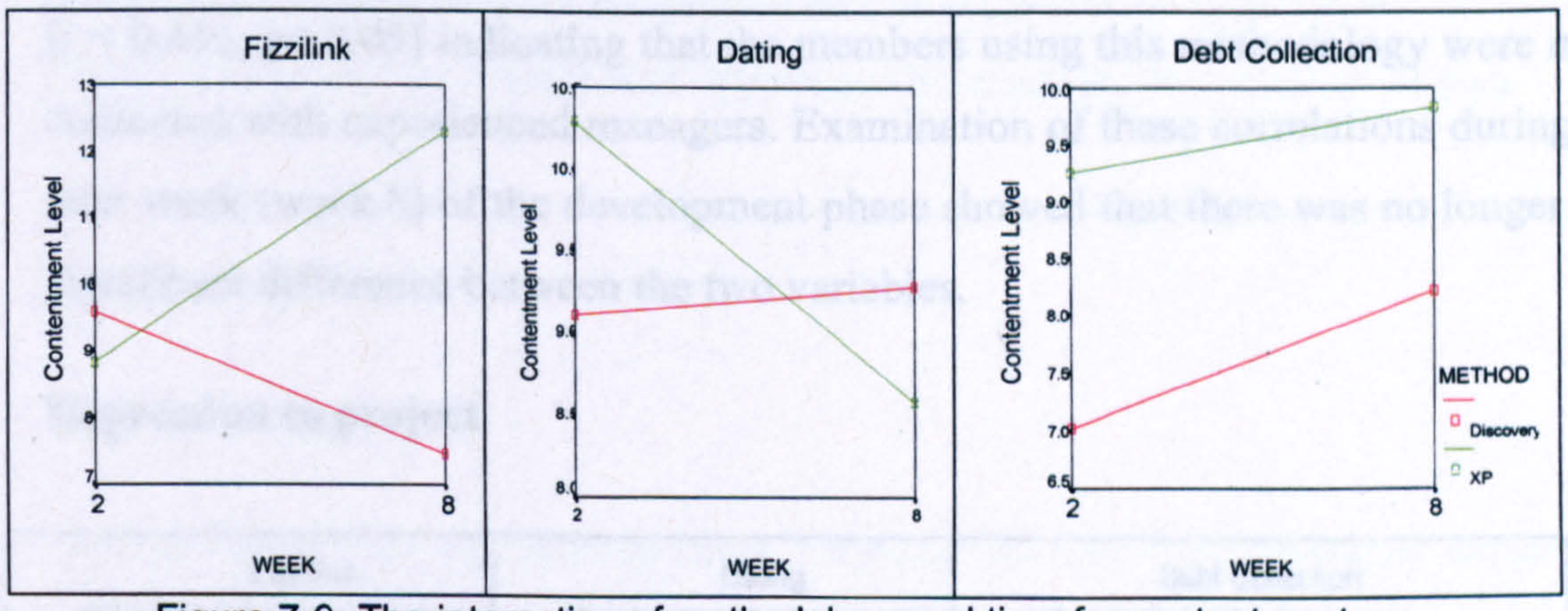


Figure 7-9 The interaction of methodology and time for contentment towards project (Software Hut 2004)

At the beginning of the project, both teams in the **Fizzilink project** showed similar contentment level. As the work progressed, the contentment level of the XP team increased (M_1=8.80, SD=3.584; M_2=12.30, SD=9.487) while the Discovery team experienced a decrease in the contentment level (M_1=9.56, SD=0.882; M_2=7.44, SD=2.789) but the difference in the contentment level after the treatment was **not significant.**

In the **Dating project**, the XP team experienced a decrease in the contentment level (M_1=10.29, SD=1.799; M_2=8.57, SD=2.070) while the Discovery team experienced a gradual increase in the contentment level (M_1=9.09, SD=3.113; M_2=9.27, SD=2.573). There was **no significant difference** in the contentment level after the methodology treatment.

In the **Debt Collection project**, the contentment level amongst the XP members was significantly higher at the beginning of the project, [(M_1=9.25, SD=3.137;

M_2=9.82, SD=2.167); z= 1.747, p=0.081]. At the end of the project there was **no significant difference** in the contentment level between the two methodologies.

The **results rejected H_2**, that the methodologies have not influence the contentment level of the SE teams.

Examination amongst the Discovery teams showed that there was a significant positive correlation between management and initial contentment (week2) variables [r = 0.411, p< 0.05] indicating that the members using this methodology were more contented with experienced managers. Examination of these correlations during the later week (week 8) of the development phase showed that there was no longer any significant difference between the two variables.

Depression to project

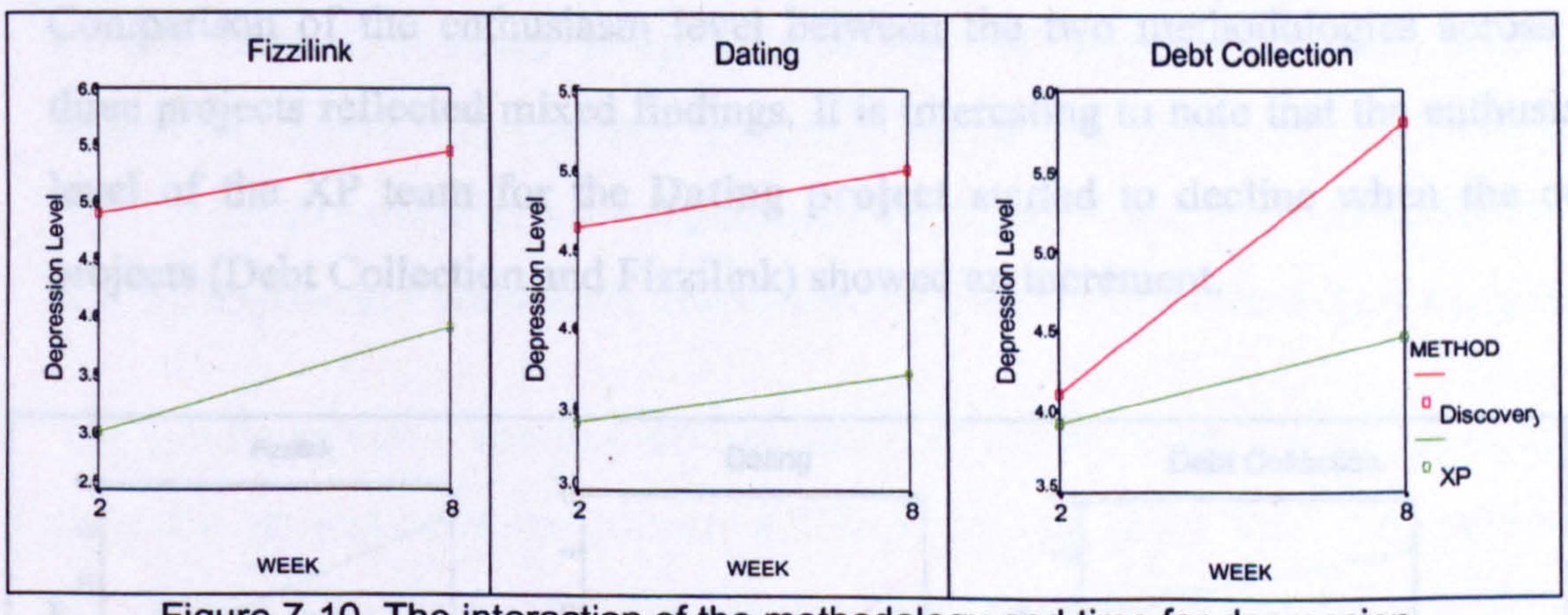


Figure 7-10 The interaction of the methodology and time for depression towards project. (Software Hut 2004)

Discovery teams experienced a higher depression level in the **Fizzilink project** (M_1=4.89, SD=1.833; M_2=5.44, SD=2.186) than the XP teams (M_1=3.00, SD=0.089; M_2=3.90, SD=2.183). There was a significant difference in depression level before the project started (week 2) [z= 2.643, p=0.008**]. There was no significant difference in the increment rate of the depression level for the two methodologies.

In the **Dating project**, there was **no significant difference** in the depression level between the two methodologies [Discovery teams (M_1=4.64, SD=2.461; M_2=5.00, SD=2.098) and XP teams (M_1=3.43, SD=0.787; M_2=3.71, SD=1.254)].

In the **Debt Collection project**, Discovery teams experienced a significant increment in the depression level between the two intervals [(M_1=4.10, SD=1.595; M_2=5.80, SD=2.860); z= 2.060, p=0.039**] while the XP teams had a lower depression level throughout the project (M_1=3.92, SD=1.498; M_2=4.46, SD=1.898). The result showed **no significant difference** between the two methodologies.

The **results rejected H_3**, indicating that the methodologies have not influence on the depression level of the SE teams

Enthusiasm to project

Comparison of the enthusiasm level between the two methodologies across the three projects reflected mixed findings. It is interesting to note that the enthusiasm level of the XP team for the **Dating project** started to decline when the other projects (Debt Collection and Fizzilink) showed an increment.

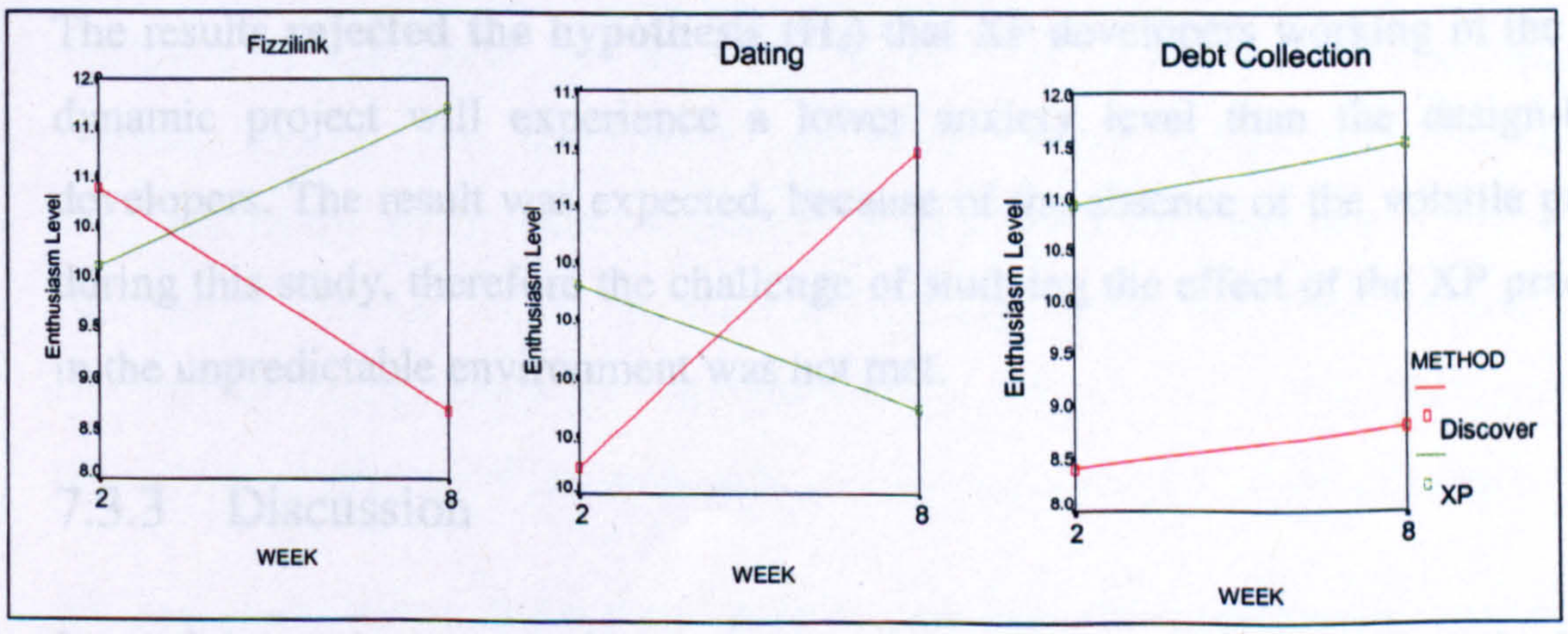


Figure 7-11 The interaction of methodology and time for enthusiasm towards project (Software Hut 2004)

Examination amongst the Discovery teams showed that there was a significant positive correlation between management and initial enthusiasm (week2) variables [r = 0.393, p< 0.05] indicating that the members using this methodology were more

enthusiastic with experienced managers. Examination of these correlations during the later week (week 8) showed no significant differences amongst the two variables.

In the **Fizzilink project**, XP teams experienced **an insignificant increase** in the enthusiasm level as the project progressed [(M_1=10.10, SD=2.183; M_2=11.70, SD=3.335) while the Discovery teams experienced **a significant decrease** in the enthusiasm level [(M_1=10.89, SD=1.764; M_2=8.67, SD=2.872); z=2.35, p=0.020].

The opposite result can be seen in the **Dating project**, where the XP teams experienced **an insignificant decrease** in the enthusiasm level [(M_1=10.71, SD=1.113; M_2=10.29, SD=2.360) while the Discovery teams experienced **an insignificant increase** in the enthusiasm level (M_1=10.09, SD=3.270; M_2=11.18, SD=2.676). The result showed **no significant difference** between the methodologies.

In the **Debt Collection project**, both the XP teams (M_1=10.92, SD=2.499; M_2=11.54, SD=1.808) and the Discovery teams (M_1=8.40, SD=3.134; M_2=8.80, SD=2.616) experienced **an insignificant increase** in the enthusiasm level. There was **no significant difference** in enthusiasm level between the methodologies.

The results **rejected the hypothesis (H_4)** that XP developers working in the most dynamic project will experience a lower anxiety level than the design-based developers. The result was expected, because of the absence of the volatile project during this study, therefore the challenge of studying the effect of the XP practices in the unpredictable environment was not met.

7.3.3 Discussion

In explaining the anxiety level, earlier exposure to a design-based methodology such as the Discovery Method helped in reducing initial feeling of anxiety amongst the design-based members. However in the unpredictable environment such as the Control Engineering project, where the requirements were vague, the ability of the

XP method to capture only partial requirements through the use of a simple story helped to prevent a drastic increased in the anxiety level. In addition, the advantage of coding in pairs and testing the code frequently contributed to a less anxious environment. Even though agile members experienced a higher level of initial anxiety due to the lack of knowledge in methodology and project, the analyses revealed that the increased rate was reduced after week 5. From the interview sessions carried out during the final week, it was indicated that the three XP coaching sessions conducted improved the members understanding of *using* story cards, *developing* test cases and *practising* pair programming.

One explanation for this finding was that, in the unpredictable environment, where the developers were allowed to capture only partial requirements at the early stage, the XP approach enabled the members to proceed quickly to the coding procedure and testing stage of the project. The ability to move forward gave the members a sense of control and progress of the project. In addition, the emphasis of several stages in testing, which was done simultaneously with the coding, permitted feedback seeking to takes place. Feedback seeking occurred when the developers purposely monitored their work for performance information and has the advantages of reducing uncertainty about work performances(Ashford 1989) and also of managing impressions (of oneself or of the client) by presenting team members as competent (Moses et al. 2003). The initial findings indicated that constant testing, pair discussions and client reviews which result in constant feedback to the team, were considered as treatments for depression. It was noted that all of the XP teams practised testing vigorously and simultaneously with coding. In comparison, the designed-based teams only tested their software vigorously at the end of the project as was expected in a traditional approach.

Feeling enthusiastic throughout project development is often a utopian ideal, especially in an unpredictable project. The ability of the XP approach to cut the project complexity into small stories, and solve them in stages according to the client's priority, allowed the XP members to retain their enthusiastic feelings. By presenting complex software as a sequence of small simple stories, the members

had a clear measure of successful achievement as well as a retreat to a previous successful step upon failure. Enthusiasm was also maintained through constant unit and integration testing that was conducted throughout the project. Communications amongst the team members and with the clients also supported the feeling of enthusiasm.

The ability to discuss complex projects through simple stories and simple design encourages teamwork. By changing pairs constantly, the XP approach created knowledge-based developers rather than job-based workers. The flexibility associated with the tasks that these developers can perform was likely to be an enhanced value in a dynamic environment and less value in the stable environment (Lepak et al. 2003). Other factors such as communication skills and personality are vital in modern SE.

7.4 XP practices and the Work Related Well being

To explore the relationship between the XP practices and the level of work related well being amongst the development teams, 4 hypotheses were developed. Spearman's Rank Order Correlation was used to calculate the strength of the relationship between the two variables and the result shows significant correlations for all of the hypotheses tested.

H_5: The higher the number of XP practices used, the lower the reported anxiety level will be

When the number of XP practices used was higher, lower anxiety levels amongst the developers were reported (Spearman coefficient -0.579, $p < 0.05$).

H_6: The higher the number of XP practices used, the higher the reported contentment level will be

When the number of XP practices used was higher, higher contentment levels amongst the developers were reported (Spearman coefficient 0.468, $p < 0.10$).

H_8: The higher the number of XP practices used, the lower the reported depression level will be

When the number of XP practices used was higher, lower anxiety levels amongst the developers were reported (Spearman coefficient -0.293, $p < 0.05$).

H_9: The higher the number of XP practices used, the higher the reported enthusiasm level will be

When the number of XP practices used was higher, higher enthusiasm levels amongst the developers were reported (Spearman coefficient 0.440, $p < 0.10$).

7.4.1 Discussion

The above results indicated that teams who gained a deep understanding of the XP practices were able to utilise them in a more efficient manner and because of the specific human related aspects of the XP methodology the members using them were able to experience better work related well being even in the most challenging situation such as the ever changing user's requirement. The result supported the second purpose of the study:.

H_B: The higher number of XP practices used will be associated with a higher level of well being experienced by the members

This study also backs up earlier research into happy SE teams (Melnik et al. 2002).

7.5 Conclusion

XP methodology has been introduced primarily as an answer to the rapidly growing and volatile software industry. The study reveals that XP methodology has a positive impact on the wellbeing of the developers and that uncertainty and complexity increases the need for flexibility, adaptability and speed (Blomqvist et al. 2002). In this study, the XP teams were showed the ways to manage the clients' uncertainty through the flexibility of story cards and the speed of releasing the

functioning part of the system; while the project complexity was controlled through the use of a simple design, refactoring, testing and pair programming.

CHAPTER 8

XP AND THE SOFTWARE QUALITY

8.1 Introduction

The purpose of this chapter is to presents the findings of a longitudinal study of the possible impact of XP practices on the quality of developed software. To achieve this, empirical data was gathered for three consecutive years from the Software Hut environment and is divided into three parts. First, the prior academic performance is envisaged as an important variable to ensure that every team has an equal opportunity to develop quality software. Second, the assessment of the software quality by the managers and the clients are the descriptive variables which will provide the statistics for the findings. Finally, the number of XP practices employed by the teams. Even though the data was gathered qualitatively, the necessity of quantifying this data is considered important in order to conduct the required statistical tests and thus providing some information concerning the impact of using the XP methodology.

This chapter is structured as follows: Section 2 discusses the comparison study between the groups that used the XP methodology and the control group. The discussion is presented in a chronological sequence starting with a discussion on the findings from Software Hut 2002 to findings from Software Hut 2004. This section is subdivided into 3 parts: the first part discusses the weaknesses discovered and the improvements made during each phase of the study. The second part discusses the result of the comparison study and the third part is more focused on the possible impacts of the XP practices on the external quality of the software. Quality in this study is more focused on the external quality being the marks awarded by the client because the client's satisfaction is the most challenging goal to achieve. The findings revealed that the client's satisfaction has a positive relationship with the number of the XP practices used by the teams. Section 3 summarises the findings of

the three consecutive studies. The results indicated that throughout the three years, the XP teams managed to develop higher quality software in 70% of the projects studied. Section 4 provides a discussion on the XP practices pattern, which was discovered during this study.

8.2 Software Quality in the Software Hut Environment

For this study, two methods of data collection were used. First, the analysis of the documents, and secondly, the focus group interview. There were two types of documents produced by the teams; the project documents and the assessment documents. The project documents consisted of short minutes of weekly meetings, three interim reports, the user manual and the installation guide. The project assessment documents consisted of the client assessment sheet and the lecturer assessment sheet. From this point onwards, the assessment by the client is referred to as the external quality and the assessment by the lecturer is referred to as the internal quality.

The external quality is measured using 10 items (APPENDIX 8-A) and the internal quality was measured using 6 items for the Discovery teams (APPENDIX 8-B) and 7 items for the XP teams (APPENDIX 8-C). The decision to focus on the external quality only was made, after considering the 'noise' factor associated with the internal quality. Analysis of the pattern in the internal quality revealed that by knowing the student's previous academic performance and the weekly performance, there was a tendency to look beyond the finished product to measure the teams' capabilities.

At the end of the term, the client will choose the best software without being aware of the methodology used by the different teams. Before the final presentation, the clients were briefed on the structured marking scheme, which they were required to follow when awarding the marks to the teams. The clients were required to fill in the assessment document after every team had delivered their software.

In order to test that the members in the development groups for the two methodologies were equally distributed, statistical analysis was conducted on marks earned in the previous course modules. The selected course modules were *Introduction to Programming, Requirement Engineering, Object Oriented Programming, System Design and Testing, Functional Programming, System Analysis and Design* and lastly *Database Technology.* A statistical test was used to compare the average marks from the previous modules for the Discovery teams and the XP teams. The Mann-Whitney statistical test conducted showed a **significant difference** in the average previous marks for the Design-based teams ($\underline{M}$ = 50.90, $\underline{SD}$ = 5.49) and the XP teams ($\underline{M}$ = 56.50, $\underline{SD}$ =56.50; z(20) = 1.98, p = 0.048), for members during the study in Software 2002. The analysis of the overall marks for the members in Software Hut 2003 and Software Hut 2004 showed **no significant difference** between the Discovery teams ($\underline{M}$ = 56.88, $\underline{SD}$ = 8.048) and XP teams ($\underline{M}$ = 57.08, $\underline{SD}$ = 6.763; t(49) = 0.096, p = 0.92).

The first purpose of this study was to assess empirically, whether there are any differences in the quality of the software developed by the XP teams and the software developed by the Discovery teams. The hypothesis was defined as:

H_A: The XP teams will develop better quality software than the designed-based teams.

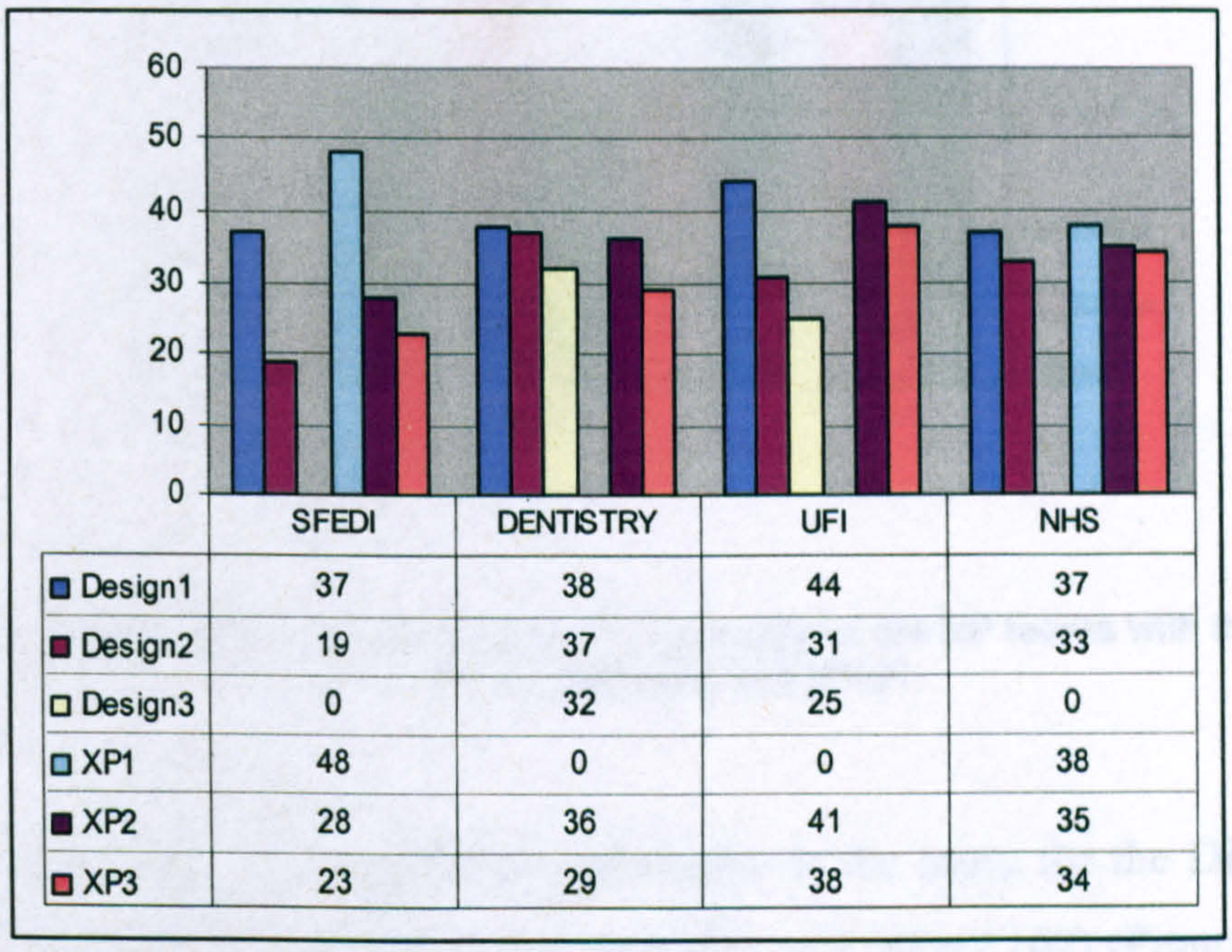
The second purpose of this study was to explore the relationship between the 12 XP practices and the quality of the software developed by the XP teams. The hypothesis was defined as:

H_B: A high number of XP practices used is associated with a high quality of software developed by the SE teams.

8.2.1 Software Hut 2002

The distribution of the teams between XP and Discovery was in the ratio 2:3, therefore the average mark was used for the analysis in this study. The XP teams

were required to use the *full adoption* of the XP practices but the observation and analysis of the documents revealed that the XP teams did not adhere to all of the practices, instead were selective in using them. The group members accepted practices, which were easy to understand and convenient to follow.



	SFEDI	DENTISTRY	UFI	NHS
Design1	37	38	44	37
Design2	19	37	31	33
Design3	0	32	25	0
XP1	48	0	0	38
XP2	28	36	41	35
XP3	23	29	38	34

Figure 8-1 Bar graph illustrating the internal quality according to project (Software Hut 2002).

At the end of week 12, 2 clients (DENTISTRY and UFI) chose software which was developed by the Design-based teams and the other 2 clients (SFEDI and NHS) selected the XP team's software (Figure 8-1). The average marks for the two methodologies were then compared using the Mann-Whitney non parametric statistical test.

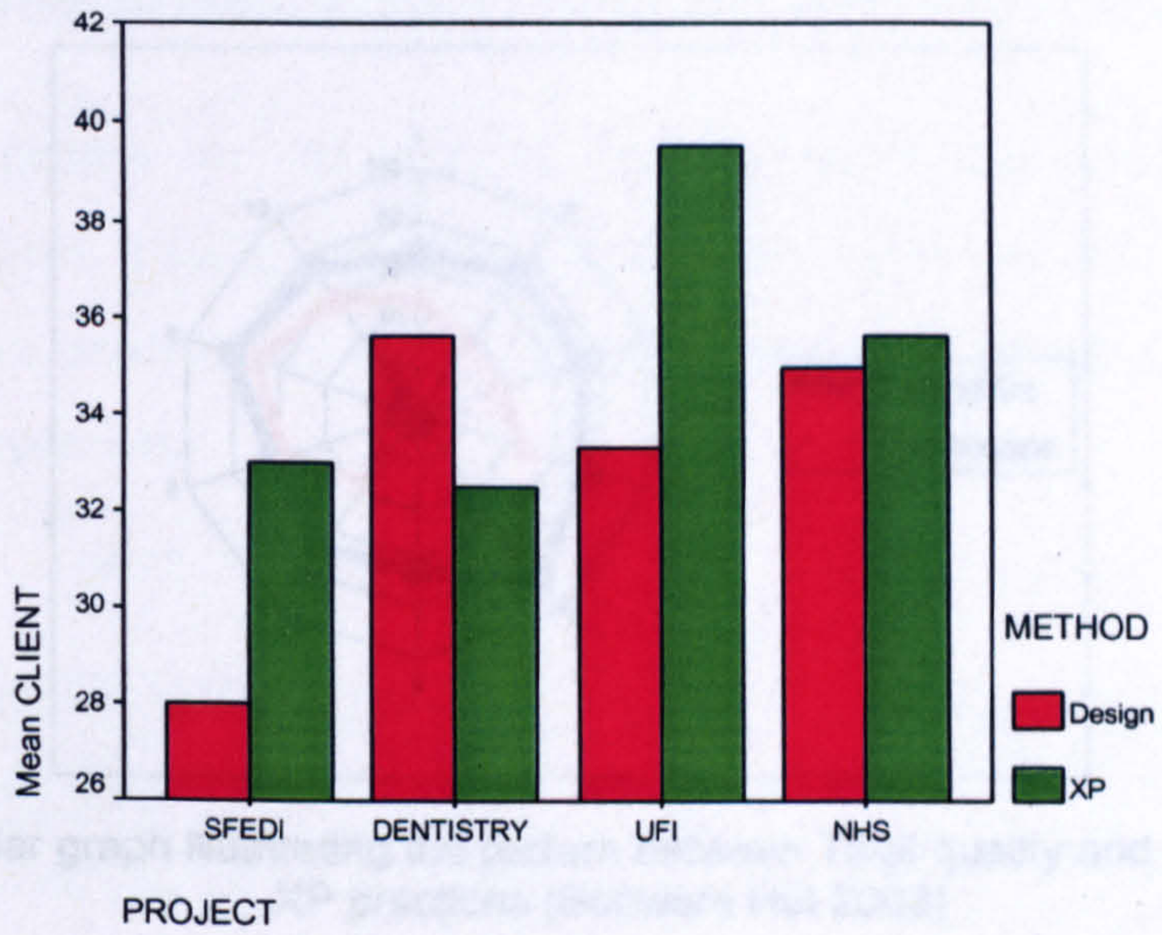


Figure 8-2 Bar graph comparing the external quality of the XP teams with the Discovery teams (Software Hut 2002).

The result showed **no significant difference** in the score for the Discovery teams (M=33.30, SD=7.134) and the XP teams (M=35.0, SD=7.102) (Figure 8-2).

The result rejected H_1 indicating that the type of software methodology has no effect on the quality of the developed software.

8.2.1.1 The implications of the XP practices on the Software Quality

Further analysis on the external quality, was made to understand the implication of the XP practices on quality. The radar graphs below illustrate 2 important points in the marking scheme of the Software Hut projects. The graph for the total quality (Figure 8-3) did not reflect any specific pattern between the total quality and the number of practices used by the teams. The pattern emerged when the total quality was divided, into the external quality and the internal quality (Figure 8-4).

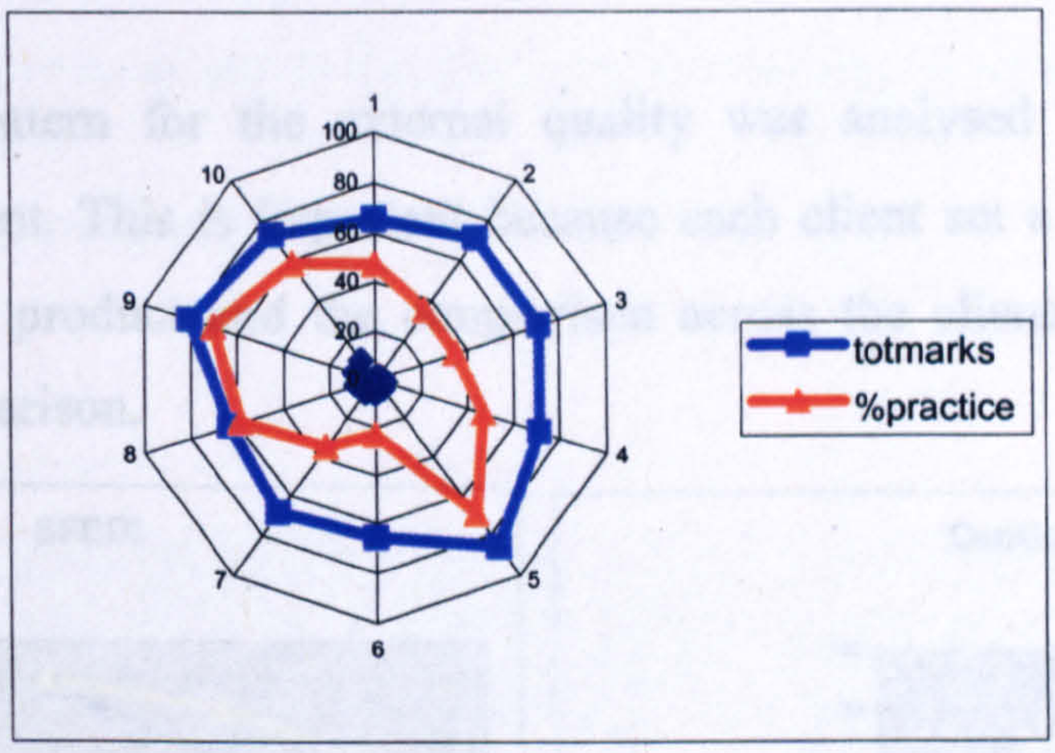


Figure 8-3 Radar graph illustrating the pattern between Total quality and the number of the XP practices (Software Hut 2002)

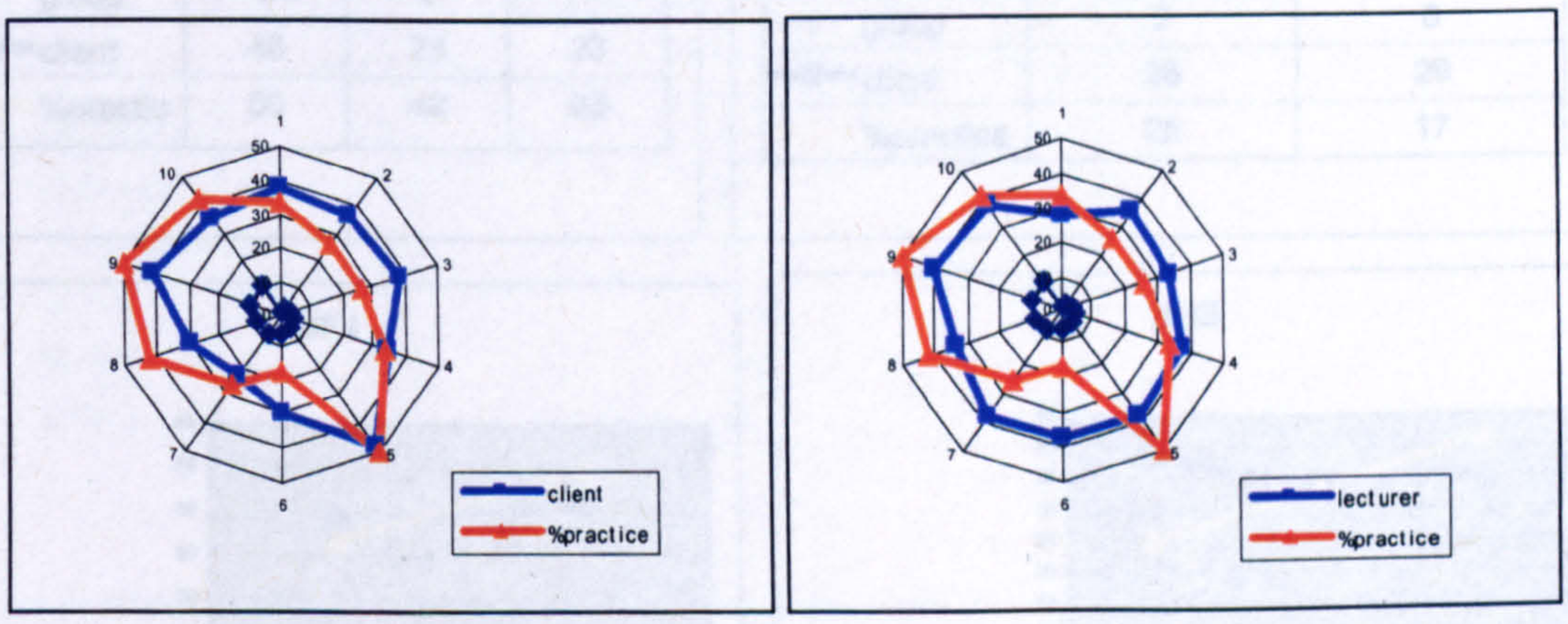


Figure 8-4 Radar graphs illustrating the pattern between external quality and internal quality with the number of the XP practices (Software Hut 2002)

The radar graph on the left is clearer in illustrating a pattern between the number of XP practices used and the external quality. This graph shows that the level of the external quality was higher for teams, which used the most practices (except team 3), whereas the graph on the right illustrates that there was little effect of the number of the XP practices on the internal quality. The correlation between the internal quality and the external quality was calculated using a non parametric test **Spearman Rank Order Correlation**. There was a weak positive correlation between the two qualities [$r = 0.176$, $n = 10$] indicating that a high external quality was associated with a high internal quality for some of the projects.

8.2.1.2 The implications of the XP practices on the External Quality

An overall pattern for the external quality was analysed further through the individual client. This is important because each client set a certain standard for their software product and the comparison across the clients did not reveal the accurate comparison.

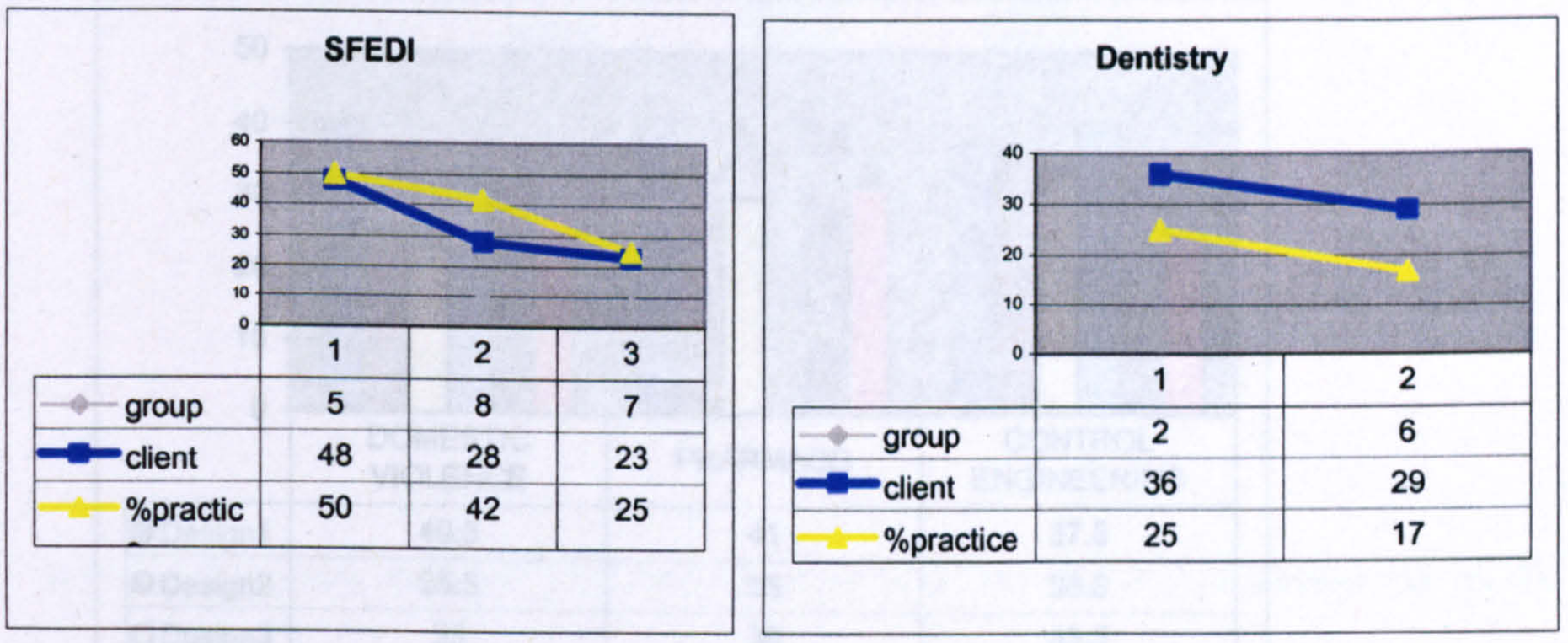
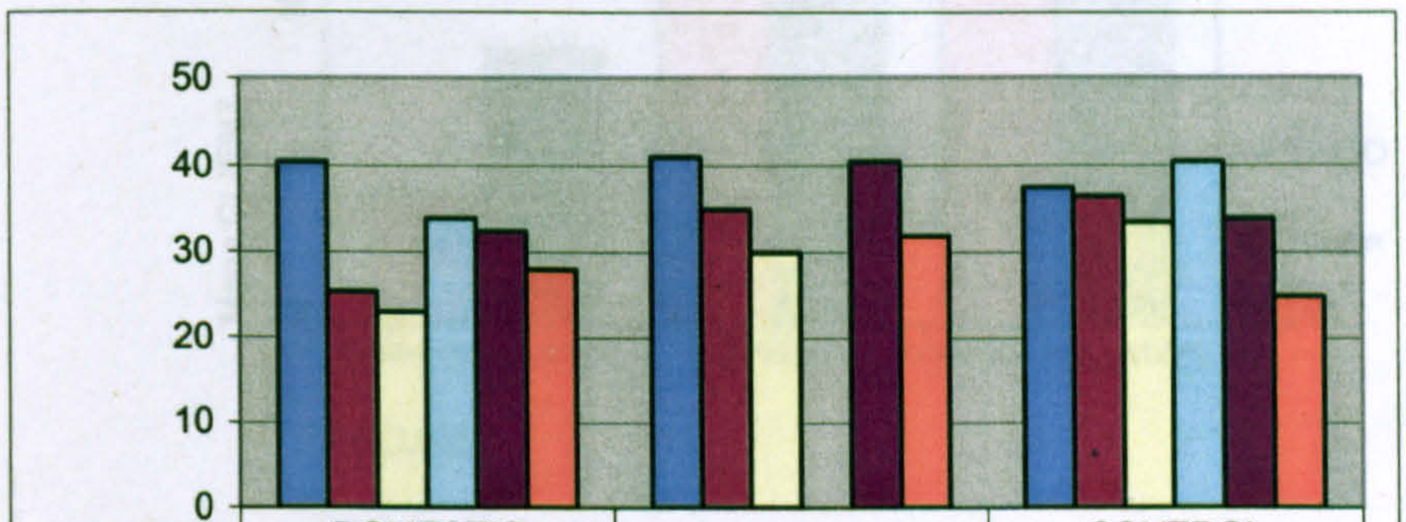


Figure 8-5 Line graph showing the relationship the client's mark and the XP practices for SFEDI, Dentistry, UFI and NHS (Software Hut 2002).

Analysis of the external quality according to clients revealed that the quality was parallel to the number of practices adopted by the teams (Figure 8-5). The line graph reflects a constant relationship pattern between practices and external quality in all cases except for team 3 (developing the NHS project). The Spearman test conducted revealed **an insignificant positive relationship** between the two variables [n=10, r=0.550, p = 0.100].

The **hypothesis H_B was rejected**; indicating that the number of the XP practices used did not have an effect on the quality of the developed software.

8.2.2 Software Hut 2003



	DOMESTIC VIOLENCE	PHARMACO	CONTROL ENGINEERING
Design1	40.5	41	37.5
Design2	25.5	35	36.5
Design3	23	30	33.5
XP1	34	0	40.5
XP2	32.5	40.5	34
XP3	28	32	25

Figure 8-6 Bar graph illustrating the external quality according to project (Software Hut 2003).

At the end of the term, 2 clients (Domestic Violence and Vigillex) selected software developed by the Discovery teams while the third client (the Control Engineering) selected the XP project (Figure 8-6). Again the average external quality for the two approaches were compared using the Mann-Whitney non parametric statistical test and the result showed ***no significant difference*** in the means score for the Discovery teams (M=33.61, SD=6.314) and the XP teams (M=33.31, SD=5.398)[Figure 8-7].

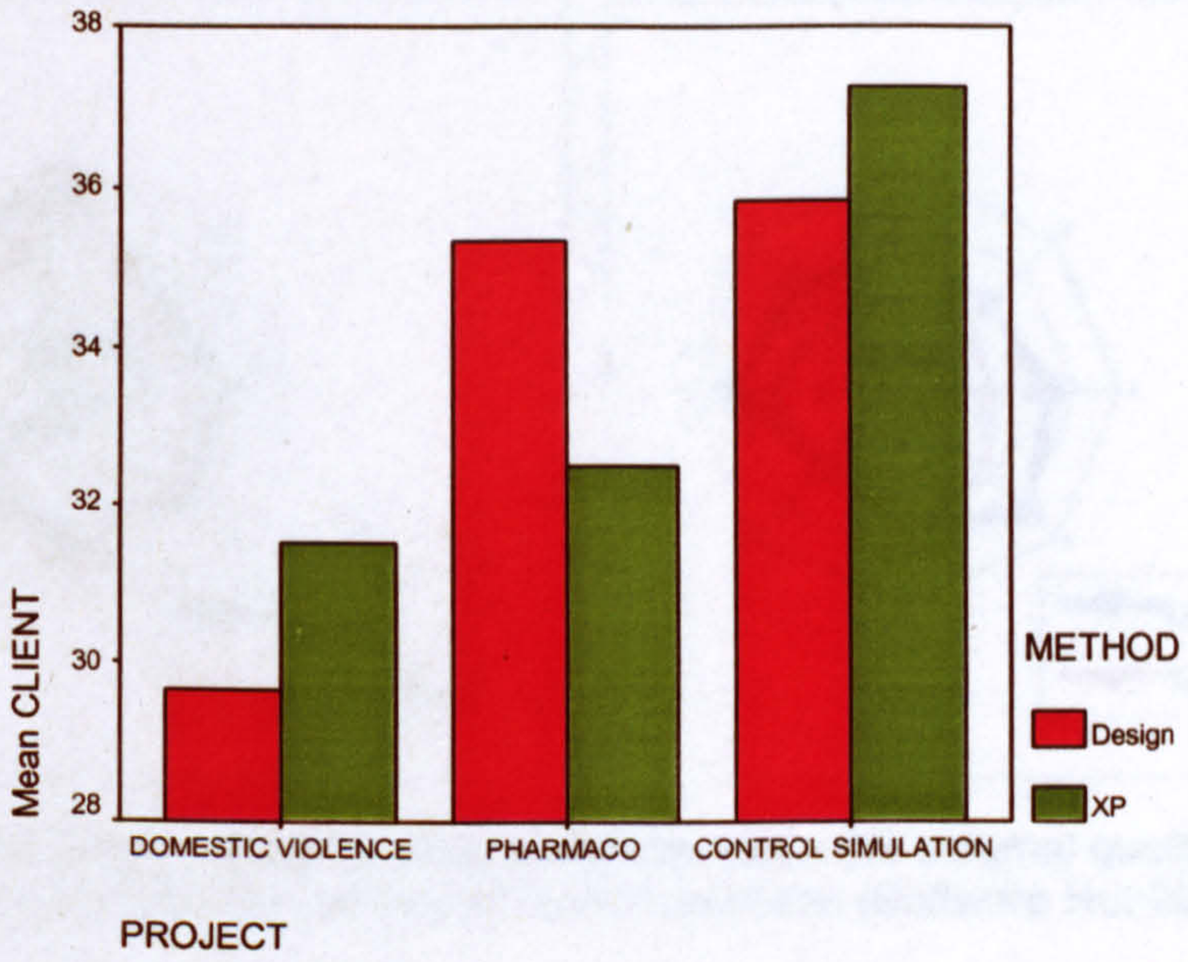


Figure 8-7 Bar graph comparing the average external quality between (Software Hut 2003).

8.2.2.1 The implications of the XP practices on Quality

The graphs illustrated in the diagram below (Figure 8-8 and Figure 8-9), show similar pattern for both internal and external quality. The pattern does not exhibit a clear relationship between quality and the number of practices used by the teams.

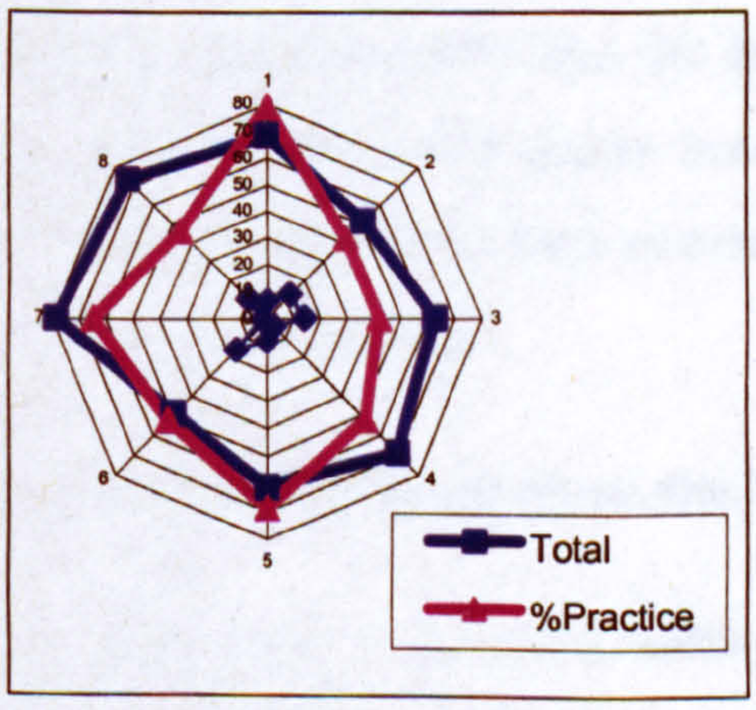


Figure 8-8 Radar graph illustrating the pattern between the Total quality and the number of XP practices (Software Hut 2003)

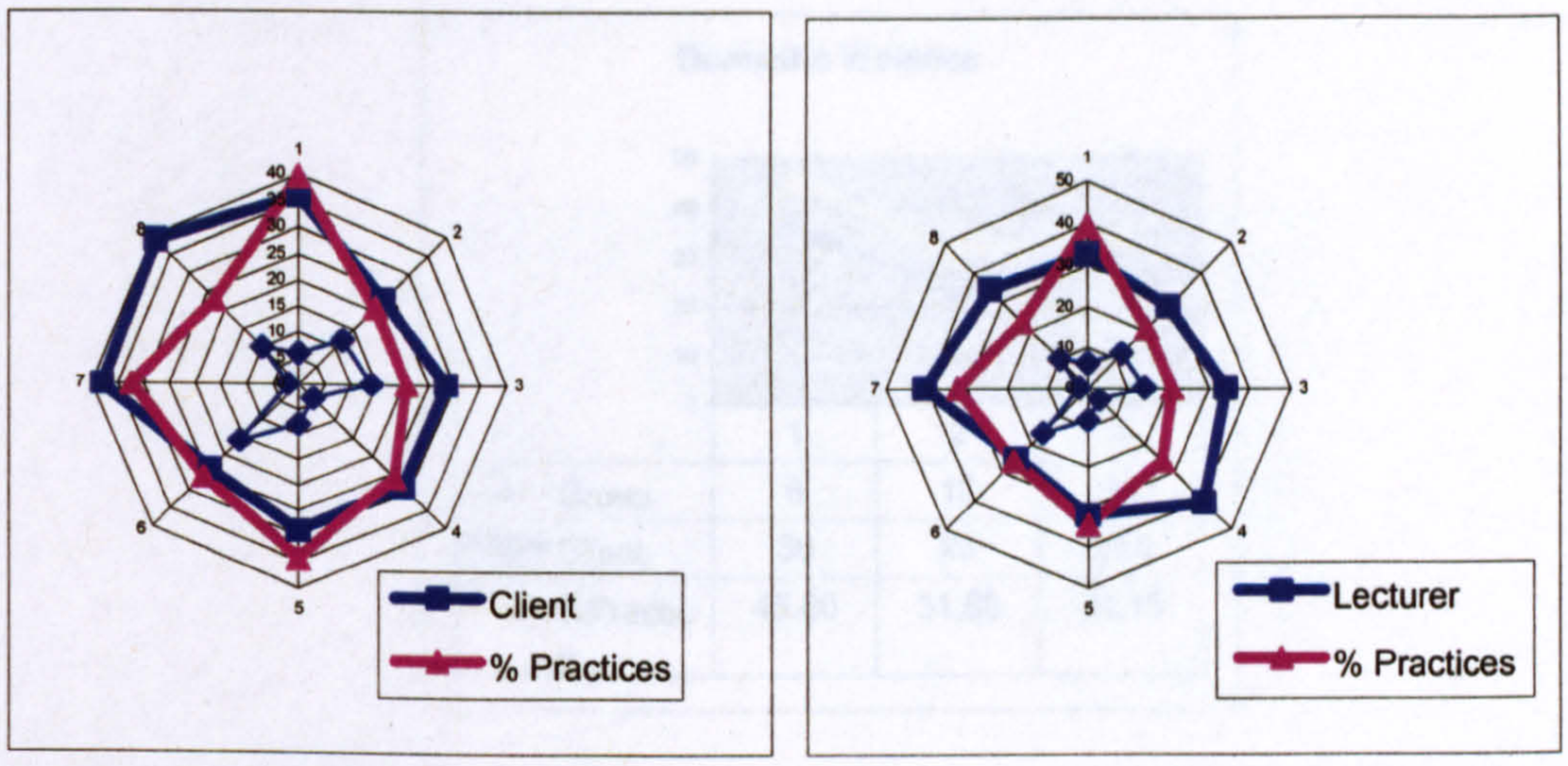


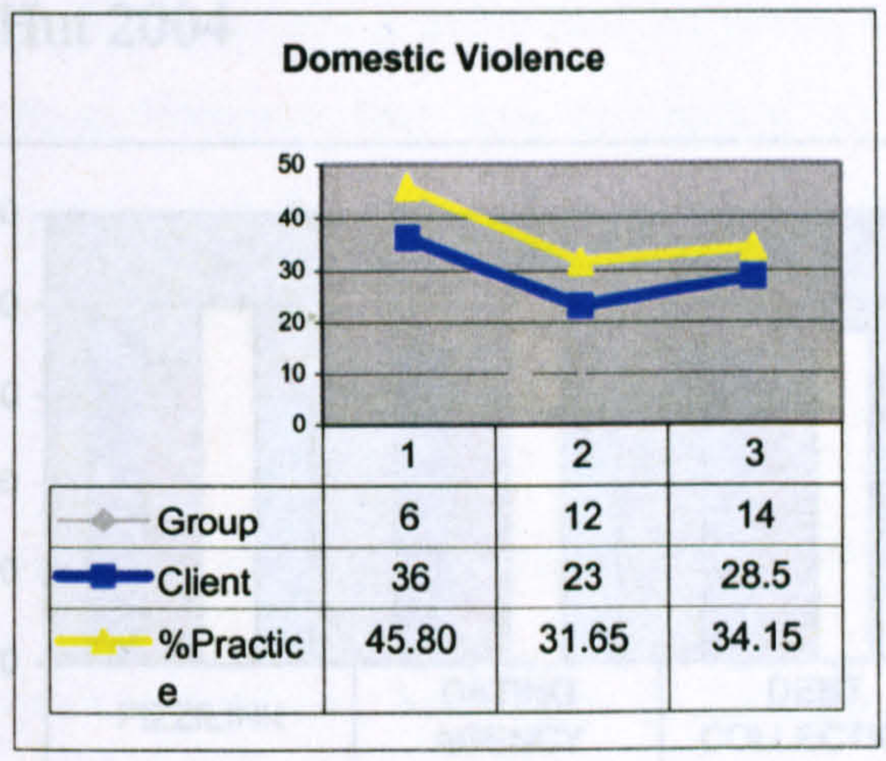
Figure 8-9 Radar graphs illustrating the pattern between the external quality and the internal quality, with the number of the XP practices (Software Hut 2003).

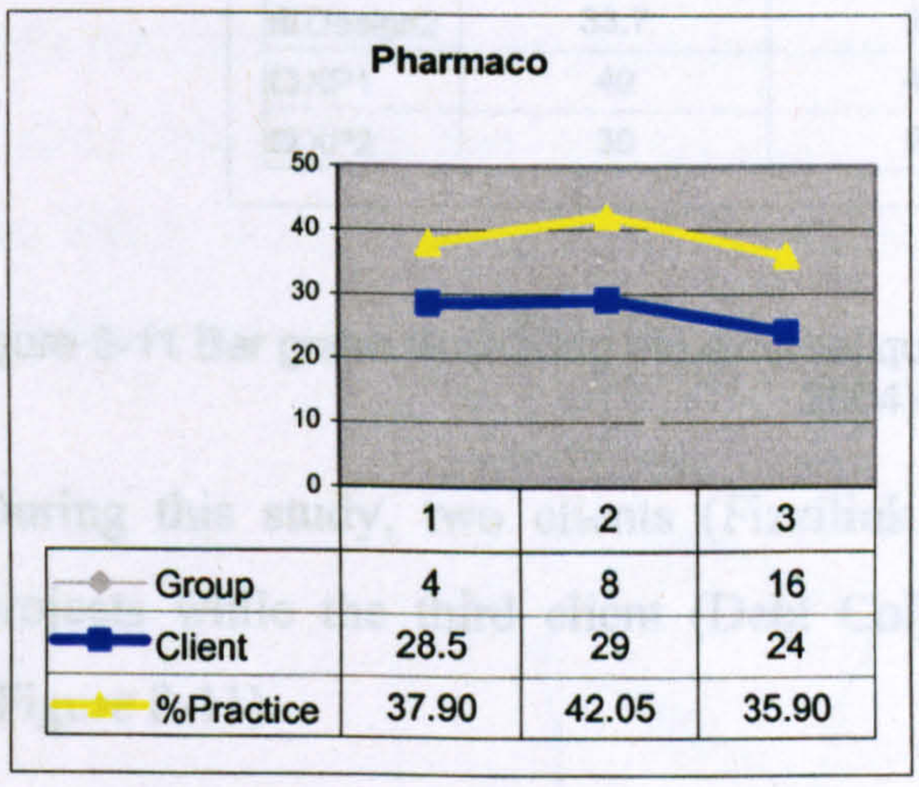
The graph for the total quality (Figure 8-8) did not reflect any specific pattern between the total quality and the number of practices used by the teams. The pattern associating the external and internal quality with the number of XP practices is not clear when the total quality was separated into external quality and internal quality (Figure 8-9), indicating that there was a small effect of the XP practices on both qualities. The possible contributing factor for this pattern is the implementation of partial adoption of the XP practices.

The correlation between the internal quality and the external quality is calculated and the result shows a strong positive correlation between the two qualities [$r = 0.704$, $p = 0.002$, $n = 17$] indicating that the high external quality is associated with high internal quality for most of the projects.

8.2.2.2 The implications of the XP practices on the External Quality

Further analysis of this relationship indicated a relationship between the external quality and the number of practices used by the teams for the two projects; Domestic Violence and Pharmaco. The relationship does not apply to the Control Engineering project. The factor that might contribute to this result is the weakness in the method used to quantify the use of XP practice. Future study should identify a proper metric to capture this data.





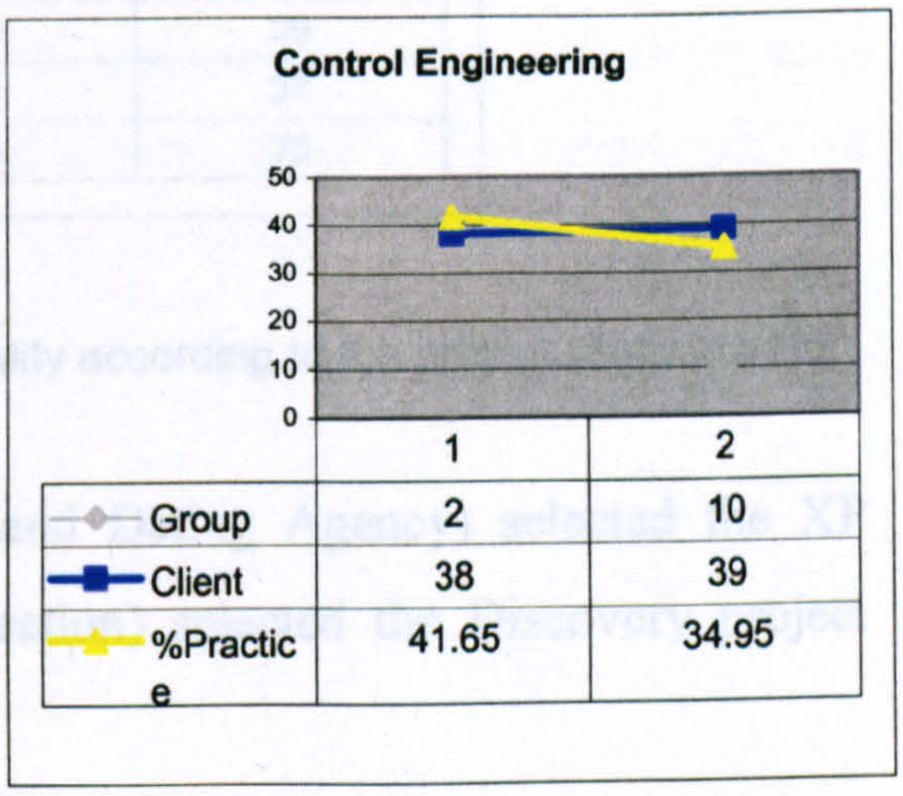


Figure 8-10 Line graphs showing the pattern between the client's mark and the XP practices for projects Domestic Violence, Pharmaco and Control Engineering (Software Hut 2003).

Analysis of the external quality according to clients revealed that the quality was parallel to the number of practices adopted by the teams (Figure 8-10). The line graph reflects a constant relationship pattern between practices and external quality in all of the cases The Spearman test conducted revealed **an insignificant positive relationship** between the two variables [n=8, r=0.448, p = 0.265].

The **hypothesis H_B was rejected**; indicating that the number of the XP practices used did not have an effect on the quality of the developed software.

8.2.3 Software Hut 2004

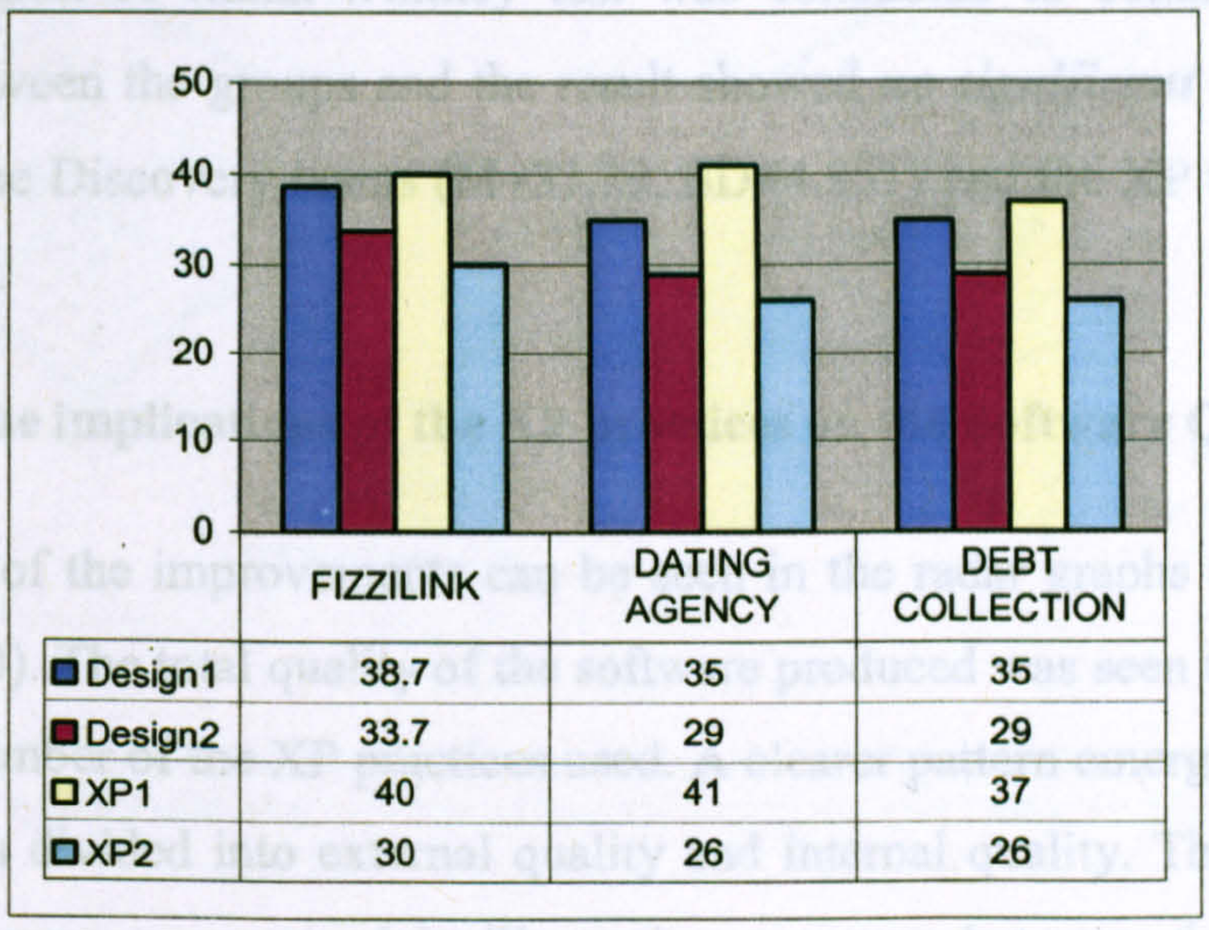


Figure 8-11 Bar graph illustrating the external quality according to the project (Software Hut 2004).

During this study, two clients (Fizzilink and Dating Agency) selected the XP projects while the third client (Debt Collection) selected the Discovery project (Figure 8-11).

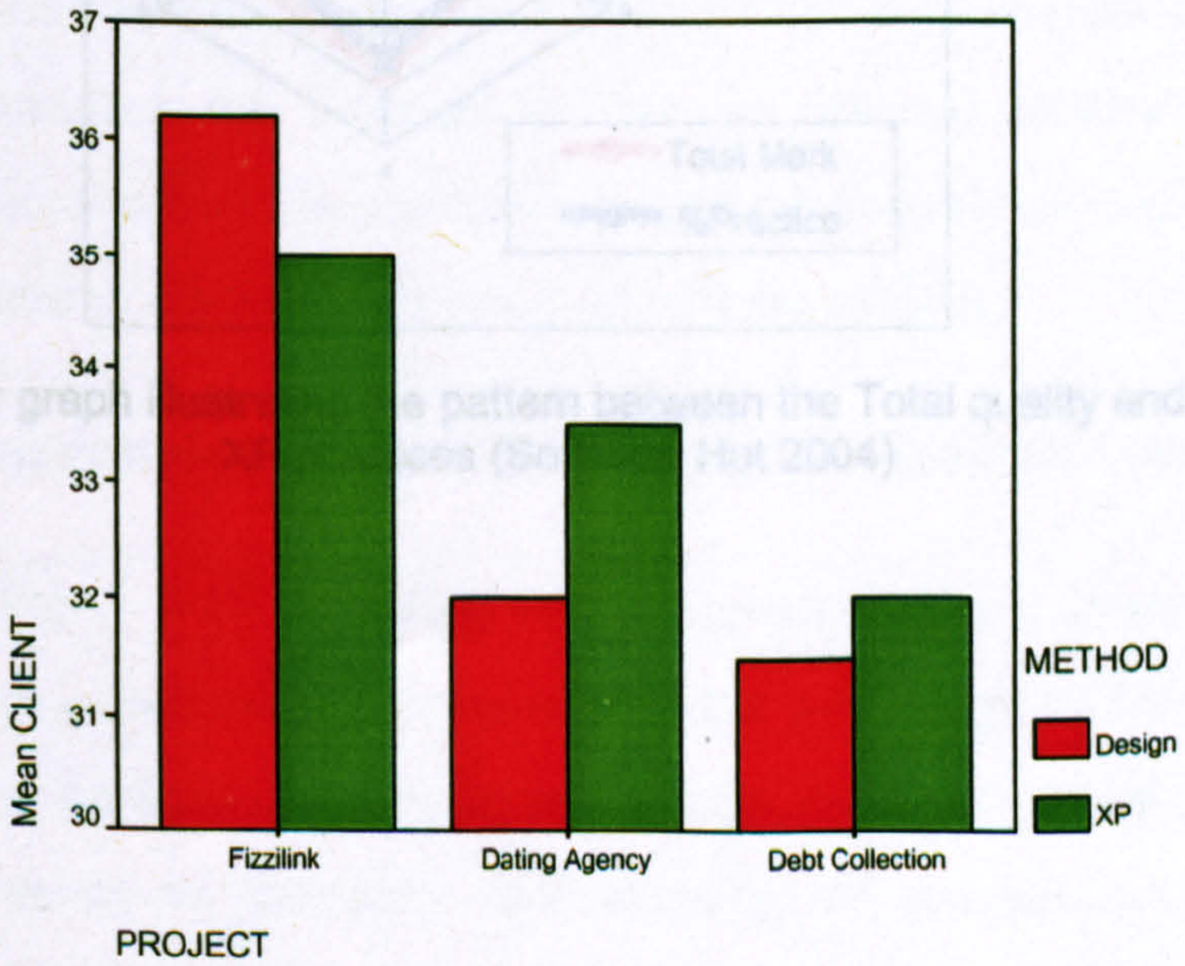


Figure 8-12 Bar graph comparing the average mark awarded to the XP teams and Discovery teams in Software Hut 2004

The bar graph (Figure 8-12) shows a difference of mean scores between the two methodologies. A Mann-Whitney test was conducted to compare the external quality between the groups and the result showed ***no significant difference*** in the score for the Discovery teams (M=33.23, SD=4.851) and the XP teams (M=33.50, SD=6.156).

8.2.3.1 The implications of the XP practices on the Software Quality

The effect of the improvements can be seen in the radar graphs (Figure 8-13 and Figure 8-14). The total quality of the software produced was seen to have a good fit with the number of the XP practices used. A clearer pattern emerged when the total quality was divided into external quality and internal quality. The radar graph on the left is more pronounced in illustrating a pattern between the number of XP practices used and the external quality.

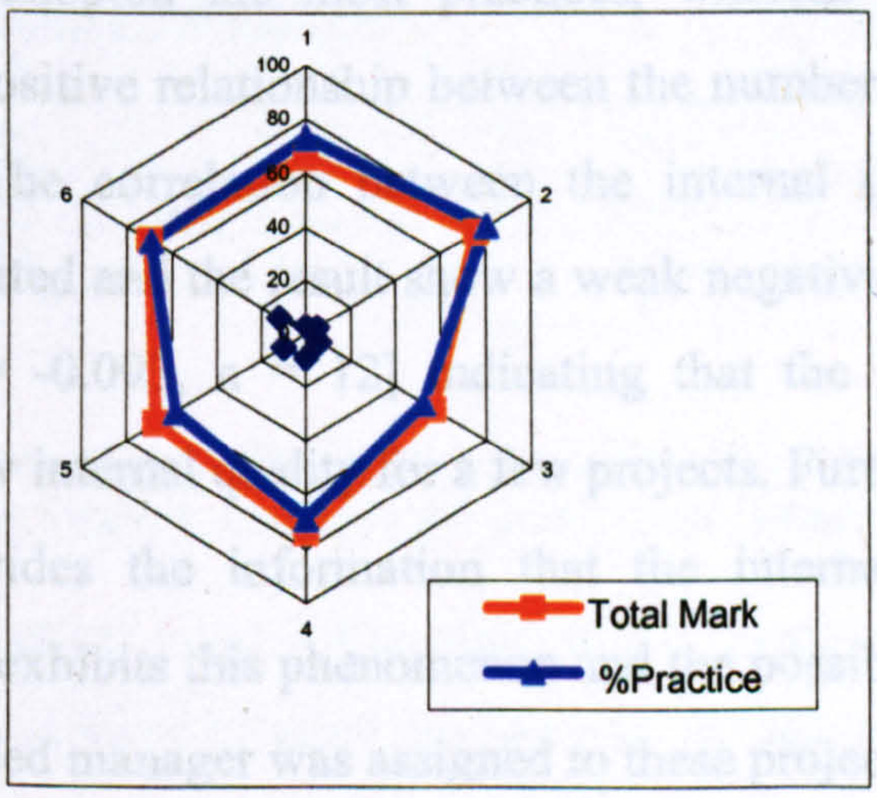


Figure 8-13 Radar graph illustrating the pattern between the Total quality and the number of XP practices (Software Hut 2004)

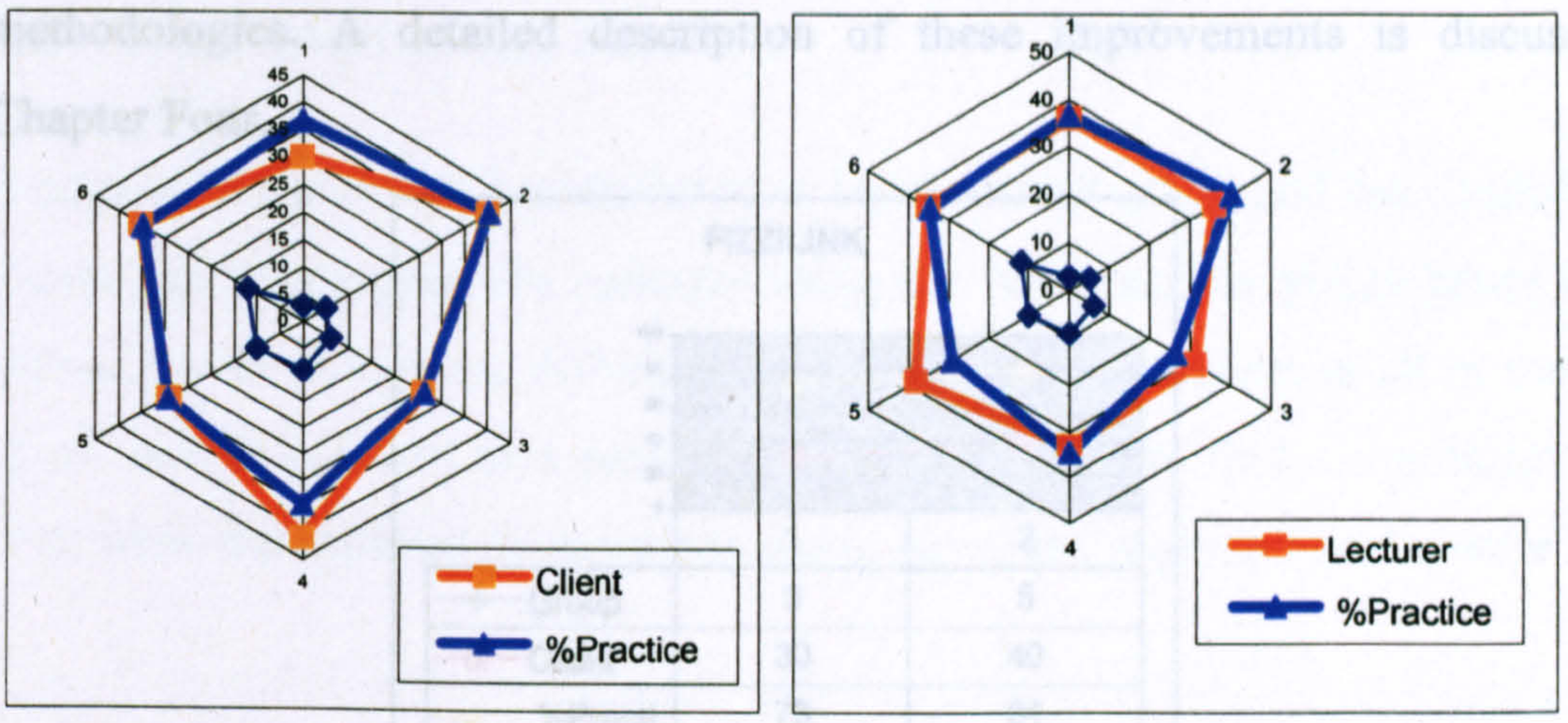


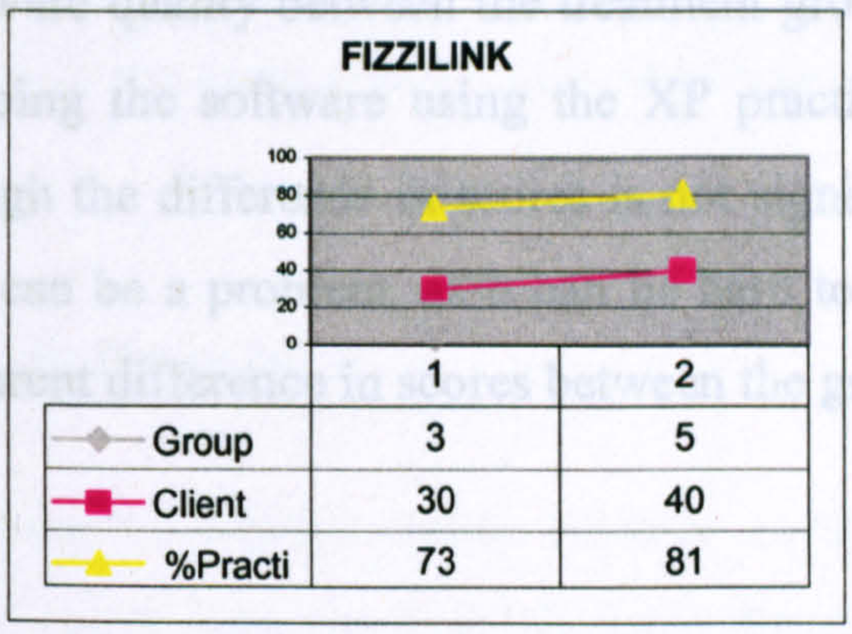
Figure 8-14 Radar graphs illustrating the pattern between the external quality and the internal quality with the number of the XP practices (Software Hut 2004)

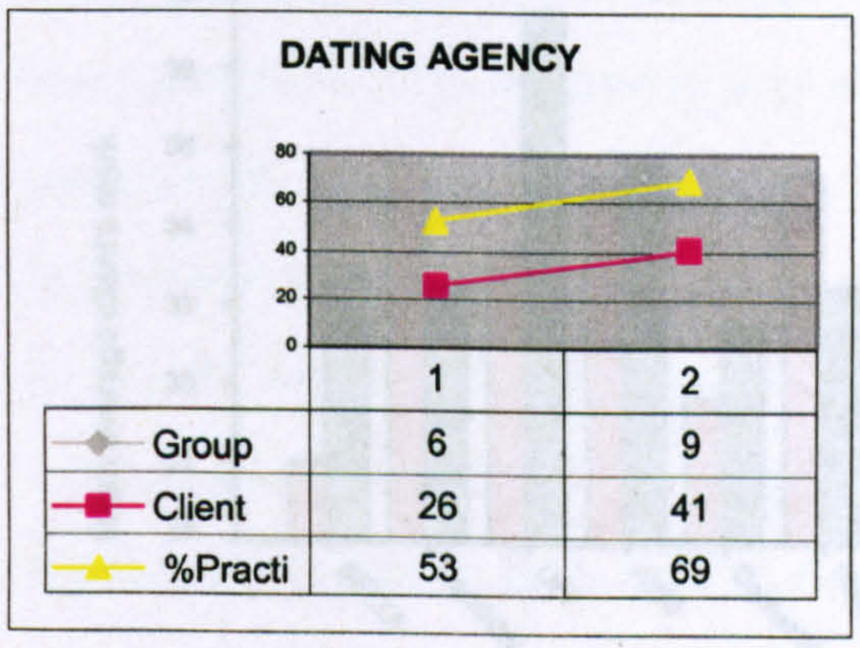
This graph (Figure 8-14) shows that the levels of the external quality were higher for teams which adopted the most practices, whereas the graph on the right illustrated a less positive relationship between the number of XP practices and the internal quality. The correlation between the internal quality and the external quality was calculated and the result show a weak negative correlation between the two qualities [r = -0.093, n = 12] indicating that the high external quality is associated with low internal quality for a few projects. Further analysis according to each project provides the information that the internal quality for the Debt Collection project exhibits this phenomenon and the possible explanation for this is that an inexperienced manager was assigned to these projects.

8.2.3.2 The implications of the XP practices on the External Quality

The graphs in Figure 8-15 show that external quality was positively related to the number of XP practices used. The increase in the percentage of the practices used in this study may be due to several factors such as the improvement to the teaching approach that leads to the early understanding of the various practices, the additional coaching sessions inside and outside the computer lab that enhance the application of the selected practices and also the acknowledgement and implementation of the different scheduling for the project sub-releases for the two

methodologies. A detailed description of these improvements is discussed in Chapter Four.





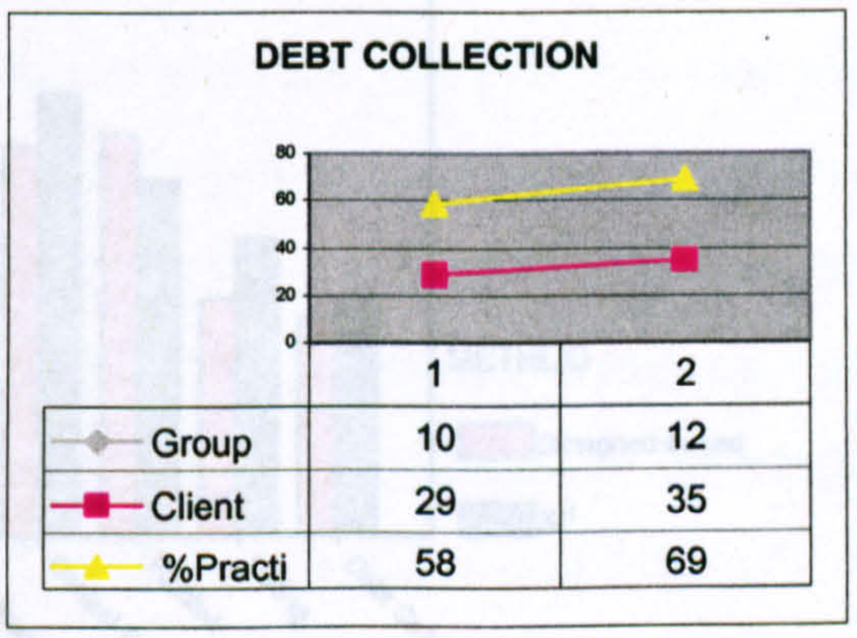


Figure 8-15 Line graphs showing the relationship between the external quality and the XP practices for projects Fizzilink, Dating Agency and Debt Collection (Software Hut 2004).

Analysis of the external quality according to the clients revealed that the quality was parallel to the number of practices used by the teams (Figure 8-15). The line graphs reflect a constant relationship pattern between practices and external quality in all of the cases The Spearman test conducted revealed **a significant positive relationship** between the two variables [n=6, r=0.749, p = 0.086].

The **hypothesis H_B was accepted,** indicating that the number of the XP practices has an effect on the quality of the developed software.

8.3 The Longitudinal Findings

Analysing the software quality between the treatment group and the control group shows that developing the software using the XP practices yields better quality software even though the difference in scores is not significant in all of the cases. Small sample size can be a problem, as it can be hard to find a significant result, even when the apparent difference in scores between the groups can be quite large.

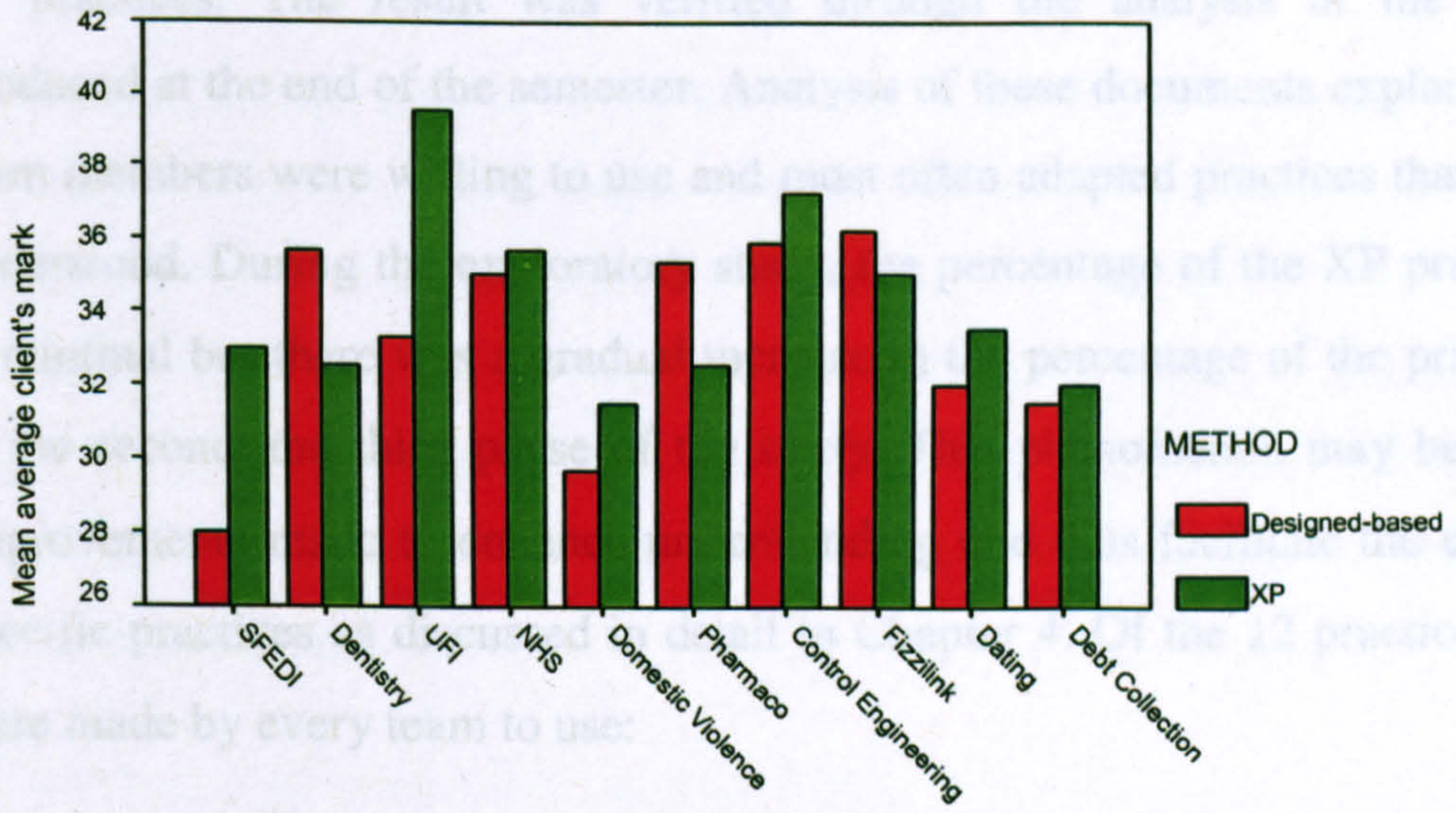


Figure 8-16 The comparison of the external quality for all teams in the Software Hut.

The mean average of the external quality for the XP teams was higher in 7 projects out of the 10 projects studied. This indicated that the teams using the XP methodology were able to produce better quality software than the control teams. The reason this was achieved, is discussed in detail in Chapter 4.

The second finding in this study is that the 12 practices must be used as a whole package; failure to adhere to a majority of these practices will lower the ability to develop software of the expected quality. This was supported by the analysis of the external quality on all of the projects. The result shows that the average mean

marks were higher for the XP teams when they were applying *full adoption* in Software Hut 2002 and Software Hut 2004 but the average mean mark was higher for the Discovery team when the partial adoption of XP practices was implemented. The findings in SOFTWARE HUT 2003 and SOFTWARE HUT 2004 provide the supporting evidence to the early study (Syed-Abdullah et al. 2003a; Syed-Abdullah et al. 2003b).

8.4 XP practices Pattern

The interviews conducted yielded certain patterns amongst the teams in using the 12 practices. The result was verified through the analysis of the documents produced at the end of the semester. Analysis of these documents explained that the team members were willing to use and most often adapted practices that were most understood. During the exploratory study, the percentage of the XP practices used is minimal but there was a gradual increase in the percentage of the practices used in the second and third phase of the study. This phenomenon may be due to the improvements made to enhance understanding and thus facilitate the usage of the specific practices as discussed in detail in Chapter 4. Of the 12 practices, attempts were made by every team to use:

Planning games

Every team planned their project but writing user stories was another matter. The requirement to include story cards in their documentation forced several teams to attempt to write the stories. In every document analysed, none of the clients write a story card. Analysis of the minutes for meetings during the exploratory and the second study indicated that the story cards were developed after the requirements were captured. This was because the knowledge on requirements engineering was carried over from the previous course modules. However, during the study 3, efforts were focused on writing the story cards after the overview diagram was completed. Continuous observation and analyses of the requirement document helped to validate that the story cards were written before any coding commenced.

Testing

Testing in this methodology consisted of Test-first coding, Unit testing and Functional testing. Testing was highlighted, as the most important part of the XP methodology. During the lectures and the coaching session, the procedure and elements needed for applying the three phases of testing were explained. Several teams produced detailed descriptions of the tests conducted throughout the projects and some teams included reports, which highlighted the importance of this practice and its impact on their project in terms of program integrity.

Pair programming

This practice was introduced to the students during the first lab session. The knowledge and experience gained during the early lab session helped them to decide whether to continue with this practice. Feedback obtained during the early interviews gave a negative impression on the acceptance of this practice. The members gave various reasons for not doing pair programming during the initial interviews but admitted the various advantages associated with the practice during the second interview sessions.

XP methodology commands rigorous testing being done on the software. Testing demands teamwork to ensure that testing was done exhaustively. Most of the team members acknowledged that they practiced pair programming readily during the testing sessions, which was nearly all the time in the Extreme programming. Some of the teams with 5 members initially divided themselves in the ratio of 3:2 for pair programming. From the second interview conducted, it was discovered that attempts by the developers to pair in groups of 3 were a waste of time. Realising that the third member does not contribute productively, the teams rearranged the members into a 2:2:1 ratio.

Collective ownership

Attempts to practice collective ownership were made by several teams. These teams swapped partners and in addition several teams also exchanged program modules

amongst the team members. Some teams did not change the members but exchanged modules to experience collective ownership amongst the members. The commentary document analysed indicated the teams' belief that having the other pair test their modules enabled the whole team members to understand the overall systems.

Frequent releases/ demonstration

Due to the time frame and the scale of the projects, it was more beneficial to conduct frequent demonstrations of the software developed at the different stages. The teams were advised to visit the client and to show the user interfaces during the reviews. The documents analysed revealed that some teams attempted to practice frequent review. The analyses identified that the teams, which were awarded the highest marks by the clients, constantly used this practice. Frequent release or in this study frequent reviews, gave the teams the benefit of improving communication, understanding and building rapport with the clients.

Coding Standard

There were attempts by several teams to follow a certain coding standard. Some teams referred to the existing coding standard available in the website, while the others referred to the Genesys Coding standard available in the company's intranet. In addition, some teams identified and formulated their own team standard according to the previous experiences of its members. In the documents analysed, the teams which did not follow any coding standard encountered incomplete and incompatibility problems during the development of the software.

8.5 Conclusion

The empirical studies presented in this chapter investigated the effect of using Extreme programming on software quality. The treatment focused on all of the aspects of the XP practices and the result of the treatment was compared with the

result of the Design-based group. The findings in this study showed that on the average, the teams using the XP practices developed better quality software.

The finding on the positive association of the external quality and the XP practices in the pilot experiment was corroborated by the second and third experiments. In order to generalise the finding external replication should be conducted. External replication refers to an experiment conducted in the industry.

In any case, the point should be emphasized that the present research is still exploratory in nature and just the first step of a series of experiments, which after modification of the treatment and stepwise inclusion of subjects in the industry, might yield more generalisable results in the future.

CHAPTER 9

CONCLUSION AND RECOMMENDATION

9.1 Introduction

This chapter summarizes the significance of this research, discusses its limitation and suggests some additional work for future research. The second section is a description of the summary of the contributions. The third section addresses the Goals of the researches individually. The fourth section is an explanation of the limitations of the research and the final section suggests some future work, some of which has been mentioned in the previous chapters.

In this research, initially only four factors were examined. These were the software development teams, the methodologies used, the dynamism of the projects being developed and the three humanistic scales; the positive affectivity, the work groups cohesion and the work-related well being parameters measured against the teams. The ability to include variations to some of the variables has strengthened the result of the findings. The variations made to the parameters include the level of the project's dynamism, the number of XP practices used and the style of project management applied. For the level of the project's dynamism, the rating was based on the observation made by the researcher and the managers. The flexibility of the XP practices allowed the variation of the practices to be included through the partial adoption and full adoption approach. When the project's dynamism was the same, a different style of project management emerged. The style used by these managers is a combination of several management techniques to ensure that the teams arrive at the required goal.

All of the empirical studies described in this research were carried out in the "Sheffield Software Engineering Observatory (Cowling 1994; Cowling 1997; Holcombe et al. 2003a) and were followed by an attempt to handle generalisability

in the industry with the IBM XP team. However, the effort to generalize the findings proved to be fruitful only in some of the studies because the team was disbanded before the second data collection was made.

9.2 Summary of the Contributions

The contributions of this research are several sets of tested hypotheses, along with the arguments in the form of the supporting evidence. This section presents the contribution of the research work and is summarised as follows:

C1 The first contribution of this research provided the empirical evidence on the affect of using the XP methodology in an unpredictable environment. The findings supported statements made by previous researchers about the need to use an agile approach for solving problem in an unpredictable software engineering environment. The results as discussed in the respective chapters highlighted the ability of the XP practices to deal with unexpected changes, therefore resulting in a better quality software and a higher feeling of the positive mood and the work related wellbeing among the SE teams.

C2 The second contribution of this research provided the additional evidence that in the domain where the problem is known and understood, the most effective solution will be to write an appropriate algorithm using a structured method to deal with it.

C3 The third contribution of this research is that it provides an insight into the psychological aspect of the Extreme Programming. While other work on agile methodologies has focused mainly on the anecdotal reference to the studies in work psychology, this research goes deeper to analyse the different aspects of the work environment and its impact on the developers. By providing empirical evidence of the impact on the developers, this study provides new information about Extreme Programming.

C4 The fourth contribution is the type of strategies used to carry out research methodology work, which is a combination of an experimental and an action research approach. The experimental approach is a familiar approach often used in the software engineering field but the action research technique, which is often applied in the information system domain, as far as the researcher is aware, has never previously been used in the analysis of the software methodology.

C5 The fifth contribution provided the empirical evidence that the practices are complementary to each other. To see the effectiveness of the methodology, it is best to use as many practices as possible, albeit with adaptation. The results provided a clear relationship between the number of the XP practices and the effect of the methodology on the developers.

C6 The sixth contribution is the empirical evidence on the need to include a diagrammatic model into the methodology. The study showed how the introduction of a simple model, such as the X-machine model, provided a balance between the depth-first and breadth-first approach.

9.3 Research Goals

This section of the chapter addresses each Goal individually, and describes how the studies and findings from Chapter 4 to Chapter 8 address the Goals defined in Chapter 1. Each Goal is listed at the beginning of each subsection, as a reminder of the goal statement.

Goal 1: To identify the difficulties in adopting the Extreme Programming practices and the reasons for these difficulties.

An initial attempt to address this goal involved conducting the focus group interview and participative observations. As the investigation continued, the difficulties, such as ensuring that the teams were following the process in the respective methodologies as closely as possible, were uncovered. This difficulty

warranted some sort of intervention which is not permissible in the existing experimental approach in the software engineering domain. The flaw to this experimental study was rectified by introducing the action research approach, which allows intervention as a way of ensuring better results in the comparison study. The focus of the investigation turned towards developing a model that will incorporate coaching the XP teams and monitoring the Discovery teams into the study. The research model is depicted in Figure 3.1 in Chapter 3.

To achieve the above goal, the data was collected qualitatively using focus group interviews and short term observations. To guide the process of collecting the correct data, the individual innovation acceptance framework(Frambach et al. 2002) was used as a guide (APPENDIX 4-A & 4-B). The information gained identified that the first difficulty encountered by the students stems from the lack of understanding of the relationship between the practices and the process flow of these practices.

During the course of the data collection, measurable activities, which represent the specific practices, have been identified (XP Activities table). These measurable indicators were demonstrated to be useful in understanding the least and the most favourable practices. This information, together with the interviews conducted in the Genesys Solutions environment, gave an insight into the process of adaptation happening amongst the novice developers. Early generalisability was achieved when a continuous informal correspondence was exercised with the XP team in the IBM at Hursley, United Kingdom. Even though the team was disbanded due to the team members being committed to other projects, the experience of using and adapting the XP practices amongst them has shown some similarities with those experienced by the Genesys developers. (See APPENDIX 4-F)

Goal 2: Demonstrate whether the improvement made to the teaching of the Extreme Programming can be utilized to facilitate the understanding and application of the method.

Following on from an understanding of the difficulties faced by the developers, efforts were made to understand the theories behind these difficulties. The second goal of this research was achieved through the graphical presentations. The introduction of the XP process flow and the categorization of the XP practices has facilitated early understanding of the methodology. In order to facilitate an understanding of the inherited projects, that is projects begun by the previous XP teams, the X-machine diagram was introduced. By introducing the diagrammatic feature into the XP methodology, it was observed that the time taken to understand both the methodology and the project was much less. Nevertheless, the researcher failed to collect quantitative data to verify this observation. The introduction of the three features was based on the cognitive theory which indicates that for a new approach to be accepted easily, it must conform to the way the brain accepts information, stimulates the mind, and thus motivates the developers. The quantitative results (as discussed in Chapter 5 to Chapter 8) and the qualitative result (as discussed in Chapter 4) demonstrated the effect of these improvements on the SE developers.

Early researchers have demonstrated that developers work effectively when the techniques they are required to use support the structure of knowledge in their knowledge base. These findings provide an insight into the practices that were easily adopted by the teams and the reasons why the development teams were more receptive towards these practices.

Goal 3: Demonstrate the role of the Extreme Programming methodology as a positive affect inducer

This goal is addressed in Chapter 5. The finding provides empirical evidence that the XP methodology does have an impact on the positive affectivity of the developers. The result shows a strong correlation between positive affectivity and

the XP practices. This is to be expected because of the existence of the practices such as a simple design, pair programming, continuous testing, continuous integration and frequent review (release), which command feedback. The members using a more flexible approach, such as the XP methodology, were able to incorporate the constant changes made by the client and thus still maintain their positive mood.

Goal 4: Demonstrates whether the Extreme Programming practices can improve the work group cohesion amongst the members of a software engineering team.

The fourth goal of this research deals with the effect of the XP methodology in increasing the work group cohesion amongst members of the same software engineering project. The finding in Chapter 6 answers this goal by demonstrating that the XP teams do experience higher work group cohesion. This pattern is consistent with previous studies which show cohesiveness amongst the team members increases with time, due to increased communication and improved coordination during the group development process (Teasley et al. 2002).

Goal 5: Demonstrates whether the Extreme Programming practices can lead to the improvement of work related wellbeing of software developers.

This Goal is addressed in Chapter 7. Analysis of the well being data shows that the practices have a significant positive impact on the well being of the developers in most of the projects developed. The result supports the hypothesis that XP developers working in the most dynamic project experience higher levels of enthusiasm than those using a design-based approach. The result reflects that uncertainty and complexity increases the need for flexibility, adaptability and speed (Blomqvist et al. 2002). In this study, the XP teams were shown ways to manage a client uncertainty through the flexibility of story cards and the speed of releasing

the functioning part of the system; whilst the project complexity was controlled through the use of a simple design, refactoring, testing and pair programming.

Goal 6: To determine whether the Extreme Programming methodology can be utilised to improve the quality of the software.

Goal 6 is addressed by presenting the results throughout the three years of the research. As reflected by the performances of the developers in the Software Hut, the average quality of the software produced by the XP methodology is higher for 7 out of the 10 projects studied. Nevertheless, the differences were not significant between the two methodologies. This study also revealed that the quality is parallel to the number of practices adopted by the teams. The line graph depicted in the study reflects a constant positive relationship pattern between practices and external quality in most of the cases studied. The adoption of the XP methodology, which was carried out in 2-ways: full adoption and partial adoption, allows the researcher to study the different impact of the two types of adoption on the quality of software developed using the XP methodology.

Taken together, the findings on positive affect, group cohesion and well being, suggests that XP methodology may be implicated in maintaining the well being over time. As improvements were made to the experiment, the findings suggested evidence that the practices can influence the positive affectivity amongst the developers. The positive affectivity experienced by the developers was then manifested in terms of interpersonal and intrapersonal relationships amongst the members. Since group cohesiveness is the outcome of a group development process, the increase in the groups cohesive level amongst the XP team members is to be expected. Previous researchers have shown that group cohesiveness is positively related to the well being of the group members.

9.4 Limitation of the Research

The methodological limitation of this research must be acknowledged. The data to measure the XP practices was collected qualitatively through interview sessions, a self-report from the development teams, progressive reports and the verifying report from the team which inherited a project. During the course of this study, activities which represent the specific practice have been identified. There is a need to identify experimental metrics to capture the attributes of interest, that is the number of practices used.

The second issue relates to the sample size. The small sample naturally raises questions concerning the statistical validity. Lack of validity is an obvious concern with a small sample but nevertheless in some of the studies, as discussed in the previous chapters, the results are strong and significant.

The third limitation is the issue of generalisability. The issue of generalisability is an issue about the type of respondent in the research. In the current research, like much of the research on comparison studies on the human factor, the use of students, even in a 'simulated' software engineering environment, is still a limitation that characterises most comparison studies. This limitation can only be addressed by replication and extension in the industry. This is not always possible for some of the comparative studies conducted here.

9.5 Future Research

Emerging from this study are several directions for future research. First, the software engineering field should broaden its scope of inquiry by including other less studied, yet potentially important, mediating variables such as project management style in an unpredictable environment and its effect on the emotional stability of the developers.

While this study has documented the findings of the work-group cohesiveness amongst the XP developers, future undertakings may choose to extend the work by

examining the relationship between group cohesion, wellbeing and the quality of software developed. Given the importance of the role of flexibility in software methodology, and its influence on the group cohesiveness in this study, such inquiry might help to improve the software engineering process for future unpredictable projects.

Third, future research should include the use of a pattern recognition technique to analyse the existing data in understanding how each practices affect the humanistic variables studied.

Fourth, future undertaking may choose to extend the work by applying the XP methodology in the artificial intelligence domain. This study should focus on the agility of the methodology and its impact on the three humanistic measures.

Finally, future research should examine to what extent the present findings can be generalized to other workplaces in the software engineering industry.

REFERENCES

Aczel, A.D. *Complete Business Statistics*, (3rd ed.) Times Miror Higher Education Group, Chicago, 1996, p. 869.

Agho, A.O., Price, J.L., and Mueller, C.W. "Discriminant validity of measures of job satisfaction, positive affectivity and negative affectivity," *Journal of Occupational and Organizational Psychology* (**65**), 11 December 1991 1992, pp 185-196.

Ainsworth, S., and Th Loizou, A. "The effects of self-explaining when learning with text or diagrams," *Cognitive Science* (27:4) 2003, pp 669-681.

Ambler, S.W. *Agile Modelling: Effective Practices for Extreme Programming and Unified Process*, (1 ed.) Wiley, 2002, p. 224.

Anderson, C., and Thompson, L.L. "Affect from top down: How powerful individuals' positive affect shapes negotiation," *Oranizational Behavior and Human Decision Processes* (95), June 2004, pp 125-139.

Andrews, D. "Software Engineering Education in the 21st century," *Information and Software Technology* (41) 1999, pp 933-936.

Appelbaum, S.H., St-Pierre, N., and William, G. "Strategic Oranizational Change: the role of leadership, learning, motivation and productivity," *Management Decision* (36:5) 1998, pp 289-301.

Ashby, F.G., Isen, A.M., and Turken, A.U. "A neuropsychological theory of positive affect and its influence on cognition," *Psychological Review* (106) 1999, pp 529-550.

Ashford, S.J. "Self-assessment in organization: A literature review and integrative model," in: *Research in Organizational Behavior,* C.L. L. and B.M. Staw (eds.), 1989, pp. 133-174.

Aspinwall, L.G. "Rethinking the Role of Positive Affect in Self-Regulation," *Motivation and Emotion* (22:1) 1998, pp 1-32.

Avison, D., and Fitzerald, G. *Information Systems Development: Methodologies, Techniques and Tools* Blackwells Scientific Publications, London, 1988, p. 309.

Avison, D.E., Lau, F., Myers, M.D., and Nielsen, P.A. "Action Research.," *Communications of the ACM.* (42:1), January 1999, pp 94-97.

Axtell, C., Wall, T., Stride, C., Pepper, K., Clegg, C., Gardner, P., and Bolden, R. "Familiarity breeds content: The impact of exposure to change on employed openness and well-being," *Journal of Occupational and Organizational Psychology* (75:2), June 2002, pp 217-231.

Basili, V.R., Selby, R.W., and Hutchens, D.H. "Experimentation in Software Engineering," *IEEE Transactions on Software Engineering* (12:7), July 1986, pp 733-743.

Basili, V.R., Shull, F., and Lanubile, F. "Building knowledge through families of experiments," *IEEE Transactions on Software Engineering* (25:4), July/Aug 1999, pp 456-473.

Batt, R., and Appelbaum, E. "Work participation in diverse settings:does the form affect the outcome, and if so, who benefits?," *British Journal of Industrial Relations* (33:3) 1995, pp 353-378.

Beck, K. *Extreme Programming Explained: Embrace Change* Addison Wesley, Longman, Reading, Mass, USA, 2000.

Beck, K., and Fowler, M. *Planning Extreme Programming*, (1 ed.) Addison-Wesley, New Jersey, 2001, p. 139.

Benbasat, I., Goldstein, D.K., and Mead, M. "The Case Research Strategy in the Studies of Information System.," *MIS Quarterly* (11) 1987, pp 369-384.

Black, T.R. *Doing quantitative research in social sciences: an integrated appraoch to research design, measurement and statistics* Sage Publications, London, 1999, p. 751.

Blaikie, N.W.H. *Analyzing quantitative data from description to explanation* Sage Publications, London, 2003, p. 352.

Blomqvist, K., Tikkanen, J., and Moller, K. "Patnering in High-Velocity Environment-Is the Interaction Theory Valid Approach," 18th annual IMP Conference, Dijon, France, 2002.

Brief, A.P., Burke, M.J., Atieh, J.M., Robinson, B.S., and Webster, J. "Should negative affectivity remain an unmeasured variable in the study of job stress?," *Journal of Applied Pschology* (73:2) 1988, pp 193-198.

Brief, A.P., Butcher, A.H., and Roberson, L. "Cookies, Disposition, and Job Attitudes:The Effects of Positive Mood-Inducing Events and Negative Affectivity on Job Satisfaction in a Field Experiment," *Oranizational Behavior and Human Decision Processes* (62:1), April 1995, pp 55-62.

Bryman, A. *Quantity and quality in social research* Routledge, London, 1992.

Carnevale, P.J.D., and Isen, A.M. "The influence of positive effect and visual access on the discovery of integrative solutions in bilateral negotiation.," *Organizational Behavor and Human Decision Process* (37) 1986, pp 1-13.

Carney, R.N., and Levin, J.R. "Pictorial Still Improve Students' Learning from Text," *Educational Psychology Review* (14:1), March 2002, pp 5-26.

Chalykoff, J., and Kochan, T. "Computer-aided monitoring: its influence on employee job satisfaction and turnover," *Personnel Psychology* (42:2) 1989, pp 807-834.

Chatman, J.A., Polzer, J.T., Barsade, S.G., and Neale, M.A. "Being different yet feeling similar: the influence of demographic composition and oragnizational culture on work process and outcomes," *Academy of Management Journal* (42) 1998, pp 273-287.

Checkland, P., and Scholes, J. *Soft System Methodology in Action.* John Wiley & Sons, 1990, p. 315.

Chen, M. "Factors affecting the adoption and diffusion of XML and Web services standards for E-business systems," *International Journal of Human-Computer Studies* (58:3), March 2003, pp 259-279.

Chuang, Y.-T., Church, R., and Zikic, J. "Organizational culture, group diversity and intra-group conflict," *Team Performance Management* (10:1) 2004, pp 26-34.

Clegg, C., Axtell, C., Damodaran, L., Farbey, B., Hull, R., Lloyd-Jones, R., Nicholls, J., Sell, R., and Tomlinson, C. "Information technology: a study of performance and the role of human and organisational factors," *Ergonomics* (40) 1997, pp 851-871.

Clegg, C., Waterson, P., and Clarey, N. "Computer supported collaborative working: lessons from elsewhere," *Journal of Information Technology* (9) 1994, pp 85-98.

Coad, P., Lefebvre, E., and De Luca, J. *Java Modeling in color with UML: Entreprise Component and Process* Prentice Hall PTR, 1999, p. 221.

Cockburn, A. *Agile Software Development*, (1 ed.) Addison-Wesley Publication Co., 2001, p. 256.

Cockburn, A. *Crystal Clear: A Human-Powered Methodology for Small Teams* Addison Wealey Professional, 2004, p. 336.

Cockburn, A., and Williams, L. "The Costs and Benefits Pair Programming," in: *Extreme Programming Examined,* G. Succi and M. Marchesi (eds.), Addison-Wesley Publishing Co., 2000, pp. 223-248.

Cole, G.A. *Management: theory and practice* Continuum, London, 1996, p. 461.

Cowling, A.J. "A Framework for Developing the Software Engineering Curriculum," Proceedings of ISEE 94, 1994, pp. 111-118.

Cowling, A.J. "Curriculum Support for Professionalism," in: *The Responsible of Software Engineer*, Springer Verlag, 1997.

Damodaran, L. "User involvement in the systems design process - a practical guide for users," *Behaviour and Information Technology* (15:6) 1996, pp 363-377.

Davison, R., and Vogel, D. "Group Support System in Hong Kong: an action research project.," *Info System Journal* (10) 2000, pp 3-20.

Deligiannis, I.S., Shepperd, M., Webster, S., and Roumeliotis, M. "A Review of Experimental Investigatins into Object-Oriented Technology," *Empirical Software Engineering* (7:3) 2002, pp 193-231.

DeMarco, T., and Lister, T. *Peopleware : productive projects and teams* Dorset House Publishing Company, New York, 1987, p. 184.

Denzin, N.K., and Lincoln, Y.S. "The Discipline and Practice of Qualitative Research.," in: *Handbook of Qualitative Research.*, Sage Publication, London, 2000.

Dick, R. "Grounded Theory: A Thumbnail Sketch at http://www.scu.edu.au/schools/gcm/ar/arp/grounded.html," 2001.

Doherty, N.F., and King, M. "the consideration of organizational issues during the system development process: an emprical analysis," *Behaviour and Information Technology* (17:1) 1998, pp 41-51.

Dubinsky, Y., and Hazzan, O. "Using Metaphors in eXtreme Programming Projects," *Lecture Notes in Computer Science.* (2675), May 2003, pp 420-421.

Erlandson, D.A., Harris, E., Skipper, B., and Allen, S. *Doing naturalistic inquiry: a guide to methods* Newbury Park, Calif; London:Sage Publications, London, 1993, p. 198.

Estrada, C.A., Isen, A.M., and Young, M.J. "Positive Affect Facilitates Integration of Information and Decreases Anchoring in Reasoning among Physicians," *Organizational Behavior and Human Decision Processes* (72:1), October 1997, pp 117-135.

Euske, K.J., Jackson, D.W.J., and Rei, W.E. "Factors contributing to the performance and satisfaction of branch managers," *Arizona Business* (29:2) 1982, pp 3-7.

Ewusi-Mensah, K., and Przasnyski, Z. "Factors contributing to the abandonment of information systems development projects," *Journal of Information Technology* (9) 1994, pp 185-201.

Fenton, N.E., and Pfleeger, S.L. *Software Metrics: a rigorous and practical approach*, (2nd ed.) International Thomson Computer Press, London, 1997, p. 638.

Fowler, M. "The New Methodology at http://www.martinfowler.com/articles/newmethodology.htm," 2002.

Fraenkel, J.R., and Wallen, N.E. *How to design and evaluate research in education*, (2nd ed.) McGraw-Hill, New York; London, 1993.

Frambach, R.T., Barkema, H.G., Nooteboom, B., and Wedel, M. "Adoption of a Service Innovation in the Business Market: An Empirical Test of Supply-side Variables," *Journal of Business Resarch* (41:2), February 1998, pp 161-171.

Frambach, R.T., and Schillewaert, N. "Organizational innovation adoption: A multi-level framework of determinants and the opportunities for future research," *Journal of Business Research* (55) 2002, pp 163-176.

Frenkel, S., Tam, M., Korczynski, M., and Shire, K. "Beyond bureaucracy? Work organisation in call centres," *International Journal Human Resouce Management* (9:6) 1998, pp 957-979.

Gallier, R. "Choosing Information Systems Research Appraoches," in: *Information Systems Research: Issues, Methods and Practical Guidelines*, R. Gallier (ed.), Alfred Walter Ltd., 1991, pp. 144-162.

Garcia, R., and Calantone, R. "A critical look at technological innovation typology and innovativeness terminology: a literature review," *The Journal of Product Innovation Management* (19:2), March 2002, pp 110-132.

Gittins, R., Bass, J., and Hope, S. "A Comparison of Software Development Process Expereinces," *Lecture Notes in Computer Science*. (3092/2004) 2004, pp 231-236.

Gittins, R., Hope, S., and Williams, I. "Qualitative Studies of XP in a Medium Sized Business.," 2nd International conference on Extreme Programming and Flexible Process in Software Engineering., Sardinia, Italy, 2001, pp. 122-126.

Greenbaum, T.L. *The handbook of Focus Group Research.*, (2 ed.) Sage Publications, 1998.

Greenwood, D.J., and Levin, M. "Reconstructing The Relationships between Universities and society through Action Research.," in: *Handbook of Qualitative Research.*, N.K. Denzin and Y.S. Lincoln (eds.), Sage Publication, London, 2000, pp. 85-106.

Hampton, R., Dubinsky, A.J., and Skinner, S.J. "A model of sales supervisor leadership behaviour and retail salespeople's job-related outcomes," *Academy of Marketing Science* (14:3) 1986, pp 33-43.

Heinbokel, T., Sonnentag, S., Frese, M., Stolte, W., and Brodbeck, F.C. "Don't underestimate the problems of user centredness in software development projects - there are many," *Behaviour and Information Technology* (15:4) 1996, pp 226-236.

Hendrickson, C. "DaimlerChrysler: The Best Team in the World," *IEEE Computer* (32:10) 1999.

Hendrickson, C. "When is it not XP? at http://www.xprogramming.com," 2000.

Hmelo-Silver, C.E., and Pfeffer, M.G. "Comparing expert and novice understanding of a complex system from the perspective of structures, behaviors, and functions," *Cognitive Science* (28:1) 2004, pp 127-138.

Holcombe, M. "What are X-Machines?," *Formal Aspects of Computing* (12:6) 2000, pp 418-422.

Holcombe, M. "Real Software Engineering Projects at the University of Sheffield," *Forum for Advancing Software Engineering Education (FASE)* (11:6) 2001.

Holcombe, M. *Extreme Programming for Real: a disciplined, agile approach to software engineering* University of Sheffield, Sheffield, 2002, p. 183.

Holcombe, M., Bogdanov, K., and Gheorge, M. "Functional Test Generation for Extreme Programming," 2nd International Conference on Extreme Programming and Flexible Processes in Software Engineering, Sardinia, Italy, 2001, pp. 109-113.

Holcombe, M., Cowling, T., and Macias, F. "Towards an Agile Approach to Empirical Software Engineering," Proceeding ESEIW 2003 Workshop on Empirical Studies in Software Engineering, Roman Castles, Italy, 2003a, pp. 37-48.

Holcombe, M., and Georghe, M. "Enterprise Skill in the Computing Curriculum," in: *Ingenia*, 2003b, pp. 56-61.

Holcombe, M., and Ipate, F. *Correct systems - building a business process solution* Springer, 1998.

Holdnak, B.J., and Busharadt, S.C. "An examination of leadership style and its relevance to shift work in an organizational setting," *Health Care Management Review* (18) 1993, pp 21-30.

Holman, D. "Employee wellbeing in call centres," *Human Resource Management Journal* (12:4), November 2002, pp 35-50.

Hornby, C., Clegg, C., Robson, J., McClaren, C., Richardson, S., and O'Brien, P. "Human and organisational issues in information systems development," *Behaviour and Information Technology* (11) 1992, pp 160-174.

Isen, A.M. "Positive affect and decision making," in: *Handbooks of Emotion,* M. Lewis and J. Haviland (eds.), Guilford, New York, 1993, pp. 261-277.

Isen, A.M. "An Influence of Positive Affect on Decision Making in Complex Situations: Theoretical Issues With Practical Implications," *Journal of Consumer Psychology* (11:2) 2001, pp 75-85.

Isen, A.M., Daubman, K.A., and Nowicki, G.P. "Positive affect facilitates creative problem-solving," *Journal of Personality and Social Psychology* (52) 1987, pp 1122-1131.

Isen, A.M., Nygren, T.E., and Ashby, F.G. "The influence of positve effect on the subjective utility of gains and lossess: It's not worth the risk.," *Journal of Personality and Social Psychology* (55) 1988, pp 710-717.

Isen, A.M., Shalker, T.E., Clark, M., and Karp, L. "Affect, accessibility of material in memory, and behavior: a cognitive loop?," *Journal of Personality and Social Psychology* (36) 1978, pp 1-12.

Jeffries, R. "Extreme Programming at http://www.armaties.com/extreme.htm," 2001.

Jeffries, R. "Circle of Life, Spiral of Death: Ways to keep your XP Project Alive and Ways to Kill it," in: *Extreme Programming Perspectives,* M. Marchesi, G. Succi, D. Wells and L. Willaims (eds.), Pearson Education, Inc, Boston, 2002, pp. 49-56.

Jehn, K.A. "A multimethod examination of the benefits and detriments of intragroup conflict," *Administrative Science Quarterly* (**40**) 1995, pp 256-282.

Johansen, K., Stauffer, R., and Turner, D. "Learning by Doing: Why XP Doesn't Sell," in: *Extreme Programming Perspectives,* M. Marchesi, G. Succi, D. Wells and L. Williams (eds.), Addison-Wesley, Boston, 2003, pp. 411-419.

Karlstrom, D. "Introducing Extreme Programming - An Experience Report," Third International Conference on eXtreme Programming and Agile Processes in Software Engineering, Sardinia, Italy, 2002.

Kemmis, S., and McTaggart, R. "Participatory Action Research.," in: *Handbook of Qualitative Research,* N.K. Denzin and Y.S. Lincoln (eds.), Sage Publication, 2000.

Kitzinger, J. "Qualitative Research: Introducing Focus Group.," *British Media Journal* (3:11) 1995, pp 299-302.

Klein, H.J., and Mulvey, P.W. "Two Investigations of the Relationships among Group Goals, Goal Commitment and Performance," *Organizational Behavior and Human Decision Processes* (61:1) 1995, pp 44-53.

Knight, D., and McCabe, D. "What happens when the phone goes wild? Staff, stress and space for escape for in a BPR telephone banking call regime," *Journal of Management Studies* (35:2) 1998, pp 163-194.

Kuhl, J. "A Theory of self-regulation: action versus state orientation, self discrimination, some applications," *Applied Psychology: An International Review* (41:2) 1992, pp 97-129.

Kujala, S. "User involvement: a review of the benefits and challenges," *Behaviour and Information Technology* (22:1), January-Februari 2003, pp 1-16.

Lakoff, G., and Johnson, M. *Metaphors We Live By* The University of Chicago Press, Chicago, 1980, p. 242.

Larkin, J.H. "Display-based problem solving," in: *Complex Information Processing: The Impact of Herbert A. Simon,* D. Klahr and K. Kotovsky (eds.), Lawrence Erlbaum Assocaites, NJ, 1989, pp. 319-341.

Latham, G.P., and Locke, E.A. *A theory of goal setting and performance* Prentice Hall, Englewood Cliffs, NJ, 1990.

Lepak, D.P., Takeuchi, R., and Snell, S.A. "Employment Flexibility and Firm Performance: Examining the Interaction Effect of Employment Model, Environmental Dynamism and Technological Intensity," *Journal of Management* (29:5), October 2003, pp 681-703.

Levine, D. *Reinventing the Workplace: How Business and Employees Can Both Win* Brookings Institute, Wahington D.C, 1995.

Liang, D.W., Moreland, R., and Argote, L. "Group versus individual training and group performance: the mediating role of transactive memory," *Personality and Social Psychology Bulletin* (**21**) 1995, pp 384-393.

Lincoln, Y.S., and Cuba, E.G. *Naturalistic inquiry* Beverly Hills, California; London Sage Publications, London, 1985, p. 416.

Littlepage, G., Robinson, W., and Reddington, K. "Effects of task experience and group experience on group performance, member ability and recognition of expertise," *Organizational Behavior and Human Decision Processes* (**69**) 1997, pp 133-147.

Lovelace, K., Shapiro, D.L., and Weingart, L.R. "Maximising cross-functional new product teams' innovativeness and constraints adherence: a conflict communications perspective," *Academy of Management Journal* (44) 2001, pp 779-794.

Lui, K.M., and Chan, K.C.C. "When Does a Pair Outperform Two Individuals," in: *Extreme Programming and Agile Processes in Software Engineering,* M.

Marchesi and G. Succi (eds.), Springer-Verlag, Heidelberg, Germany, 2003, pp. 225-233.

Lyytinen, K., and Hirscheim, R. "Information systems failure: A survey and classification of the empirical literature," *Oxford Surveys in Information Technology* (4) 1987, pp 257-309.

Lyytinen, K., and Robey, D. "Learning failure in information systems development," *Info System Journal* (9) 1999, pp 85-101.

Macias, F., Holcombe, M., and Gheorge, M. "Design-led & Design-less: One Experiments and Two Approaches," *Lecture Notes in Computer Science.* (2675) 2003, pp 394-401.

Melnik, G., and Maurer, F. "Perceptions of Agile Practices: A Student Survey," the 3rd International Conference on eXtreme Programming and Flexible Process in Software Engineering, Sardinia, Italy, 2002, pp. 241-250.

Miles, M.B., and Huberman, A.M. *Qualitative Data Analysis: an expanded sourcebook*, (2 ed.) Sage Publication, London, 1994, p. 338.

Mitchell, S., Owen, J., and Warr, K. "An adventure in Extreme Programming: Why ten developers will never be the same again.," *IBM WebSphere Developer Technical Journal*), August 2004.

Moses, S.E., Valenzi, E.S., and Taggart, W. "Are you hiding from your boss? The Development of a Taxonomy and Instrument to Assess the Feedback Management Behavior of Good and Bad Performer," *Journal of Management* (29:4), August 2003, pp 467-606.

Mullen, B., and Copper, C. "The relation between group cohesiveness and performance: An integration," *Psychological Bulletin* (2:115) 1994, pp 210-227.

Muraven, M., Tice, D.M., and Baumeister, R.F. "Self control as limited resource: Regulatory depletion patterns," *Journal of Personality and Social Psychology* (74) 1998, pp 774-789.

Norman, D.A. "Emotion and design: Atrractive things work better.," *Interactions Magazine,* (9:4) 2002, pp 36-42.

Norman, D.A., Ortony, A., and Russell, D.M. "Affect and machine design: Lessons for the development of autonomous machines," *IBM Systems Journal* (42:1) 2003, pp 38-43.

O'Donnell, A.M., Dansereau, D.F., and Hall, R.H. "Knowledge Map as Scaffolds for Cognitive Processing," *Educational Psychology Review* (14:1), March 2002, pp 71-86.

Olson, E.M., Walker, O.C., and Ruekert, R.W. "Organizing for effective new product development: The moderating role of product innovativeness," *Journal of Marketing* (59:1) 1995, pp 48-62.

O'Reilly, M.Q., Caldwell, D.F., and Barnett, W.P. "Work group demography, social integration and turnover," *Administrative Science Quarterly* (34:1) 1989, pp 21-37.

Pallant, J. *SPSS Survival Manual* Open University Press, 2001, p. 267.

Palmer, S.R., and Felsing, J.M. *A Practical Guide to Feature-Driven Development*, (1 ed.) Prentice Hall PTR, 2002, p. 304.

Patterson, D.W. *Artificial Neural Networks: Theory and Applications*, (first ed.) Prentice Hall, Singapore, 1996, p. 477.

Patton, M.Q. *Qualitative Research & Evaluation Methods*, (3 ed.) Sage Publications, 2002, p. 598.

Pearce, C.L., Gallagher, C.A., and Ensley, M.D. "Confidence at the group level of analysis: A longitudinal investigation of the relationship between potency and team effectiveness," *Journal of Occupational and Organizational Psychology* (75) 2002, pp 115-119.

Pelled, L.H. "Demographic diversity, conflict, and work group outcomes: an intervening process theory," *Organization Science* (7) 1996, pp 615-631.

Pool, S.W. "The relationship of job satisfaction with substitutes of leadership, leadership behavior, and work motivation," *The journal of Psychology* (131:3) 1997, pp 271-282.

Price, J.L., and Mueller, C.W. *Handbook for Organizational Measurement* Harper-Collins, Scranton, 1986.

Robilliard, P.N. "The Role of Knowledge in Software Development," *Communications of the ACM* (42:1), January 1999, pp 87-92.

Robinson, H., and Sharp, H. "The Characteristic of XP teams," Extreme Programming and Agile Processes in Software Engineering: 5th International Conference,XP 2004, Garmisch-Partenkirchen, Germany,, 2004, pp. 139-147.

Rogers, E.M. *Diffusions of Innovations*, (4 ed.) Free Press, New York, 1995, p. 518.

Savery, L.K. "Attitudes to work: the influence of perceived style of leadership on a group of workers," *Leadership & Organization Development Journal* (15:4) 1994, pp 12-18.

Schnotz, W. "Commentary: towards an Integrated View of Learning from Text and Visual Display," *Educational Psychology Review* (14:1), March 2002, pp 101-102.

Schwaber, K. "SCRUM Development Process," OOPSLA, Object-oriented Programming Systems, Languages and Applications, San Jose, California, 1996.

Schwaber, K. *Agile Software Development with SCRUM*, (1 ed.) Prentice Hall, 2001, p. 150.

Sevastos, P., Smith, L., and Cordery, J.L. "Evidence on the reliability and construct validity of Warr's (1990) well-being and mental health measures," *Journal of Occupational and Organizational Psychology* (65), 20 September 1991 1992, pp 33-49.

Sharp, H., and Robinson, H. "An Ethnography of XP practice," Proceeding of the Joint EASE and PPIG Conference, Keele University, UK, 2003, pp. 15-27.

Sheremata, W.A. "Finding and solving problems in software new product development," *The Journal of Product Innovation Management* (19) 2002, pp 144-158.

Shull, F., Mendoca, M.G., Basili, V.R., Carver, J., Fabbri, S., Travassos, G.H., and Ferreira, M.C. "Knowledge-Sharing Issues in Experimental Software Engineering," *Empirical Software Engineering* (9:1/2) 2004, pp 111-137.

Simons, A.J.H. "Object Discovery: A process for developing medium sized applications," Tutorial 14, ECOOP 1998, AITO/ACM, Brussels :, 1998, p. 109.

Simons, A.J.H. "The Discovery Method at http://www.dcs.shef.ac.uk/~ajhs/discovery," 2003.

Solomon, G., and Globerson, T. "When teams do not function the way they ought to," *International Journal of Education Research* **(13)** 1989, pp 89-99.

Soloway, E., and Sitharama, I. *Empirical Studies of Programers.* Ablex Publishing Corporation, Washington D.C., 1986.

Stapleton, J. *Dsdm: a Framework for Business Centred Development* Addison Wesley, 2002, p. 208.

Stapleton, J., and MacDonald, D. *DSDM: The Method in Practice* Addison Wesley, 1997, p. 192.

Staw, B.M., Bell, N.E., and Clausen, J.A. "The dispositional approach to job attitudes: A lifetime longitudinal test," *Administrative Science Quarterly* (31) 1986, pp 56-77.

Staw, B.M., and Ross, J. "Stability inn the midst of change: A dispositional appraoch to job attitudes," *Journal of Applied Pschology* (70) 1985, pp 469-480.

Stogdill, R.M. "Personal Factors Associated with Leadership: A Survey of the Literature," *Journal of Psychology* (25:January 1948) 1948, pp 35-71.

Stone, S., and Collin, H. "Basic social research techniques," Centre for Research on User Studies, University of Sheffield, Sheffield.

Strauss, A., and Corbin, J. *Basic of Qualitative Research:Grounded Theory Procedures and Techniques* Sage Publications, 1990, p. 270.

Succi, G., Marchesi, M., Pedrycz, W., and Williams, L. "Preliminary Analysis of the Effect of Pair Programming on Job Satisfaction," 3rd International Conference on Extreme Programming and Flexible Processes in Software Engineering (XP2002), Sardinia, Italy, 2002.

Syed-Abdullah, S., Holcombe, M., and Gheorge, M. "An Agile Methodology in Practice," CS-03-04, Department of Computer Science, University of Sheffield, p. 7.

Syed-Abdullah, S., Holcombe, M., and Gheorge, M. "Practice makes Perfect," in: *Lecture Notes in Computer Science LNCS 2675,* M. Marchesi and G. Succi (eds.), Springer-Verlag, Berlin Heidelberg, Germany, 2003b, pp. 354-356.

Teasley, S.D., Covi, L.A., Krishnan, M.S., and Olson, J.S. "Rapid software development through team collocation," *IEEE Transactions on Softwave Engineering* (28) 2002, pp 217-233.

Thomson, C., and Holcombe, M. "Applying XP Ideas Formally," 1st South-East European Workshop on Formal Methods (SEEFM03),, City Collage and South-Eastern European Research Center (SEERC), Thessaloniki, Greece, 2003.

Thomson, N.F., and Martinko, M.J. "The relationship between MBTI types and attributional style," *Journal of Psychological Types* (35) 1995, pp 22-30.

Trope, Y., and Neter, E. "Reconciling competing motives in self-evaluation: The role of self-control in feedback seeking," *Journal of Personality and Social Psychology* (66) 1994, pp 646-657.

Trope, Y., and Pomerantz, E.M. "Resolving Conflicts Among Self-Evaluative Motives: Positive Experiences as a Resource for Overcoming Defensiveness," *Motivation and Emotion* (22:1), Januari 1998, pp 53-72.

Urban, G.L., and Hauser, J.R. *Design and Marketing of New Products*, (second ed.) A Simon & Schuster Company, New Jersey, 1993, p. 701.

Verdi, M.P., and Kulhavy, R.W. "Learning with Maps and Texts: an Overview," *Educational Psychology Review* (14:1), March 2002, pp 27-46.

Waarts, E., van Everdingen, Y.M., and van Hillegersberg, J. "The dynamics of factors affecting the adoption of innovations," *The Journal of Product Innovation Management* (19), 13 April 2002 2002, pp 412-423.

Walenstein, A. "Foundations of Cognitive Support: Towards Abstract Pattern of Usefulness," 14th Annual Conference on Design, Specification, and Verification of Interactive Systems (DSV-IS'2002), Rostock, Germany, 2002.

Warr, P.B. "The measurement of well-being and other aspects of mental health.," *Journal of Occupational and Organizational Psychology* (63) 1990, pp 193-201.

Watson, D., and Clark, L.A. "Negative Affectivity: The disposition to experience aversive emotional states," *Psychological Bulletin* (96) 1984, pp 465-490.

Watson, D., and Clark, L.A. "Extraversion and its positive emotional core," in: *Handbook of Personality Psychology,* R. Hogan and J.A. Johnson (eds.), Academic Press, San Diego, Ca., 1997, pp. 767-793.

Watson, D., Pennebaker, J.W., and Folger, R. "Beyond negative affectivity: Measuring stress and satisfaction in the workforce," *Journal of Organizational Behavior Management* (8:2) 1987, pp 141-157.

Watson, D., and Tellegen, A. "Towards a consensual structure of mood," *Psychological Bulletin* (98) 1985, pp 219-235.

Webber, S.S., and Donahue, L.M. "Impact of highly and less job-related diversity on work group cohesion and performance: a meta-analysis," *Journal of Management* (27:2) 2001, pp 141-162.

Weinberg, G.M. *The Psychology of Computer Programming* Van Nostrand Reinhold Company, New York, 1971.

Wells, J.D. "Extreme Programming: a Gentle Introduction at http://www.extremeprogramming.org," 2001.

Williams, L. "Pair Programming: Why Have Two Do the Work of One?," in: *Extreme Programming Perspectives,* M. Marchesi, G. Succi, D. Wells and L. Willaims (eds.), Addison-Wesley, Boston, 2002, pp. 23-33.

Yoo, Y., and Alavi, M. "Media and Group cohesion: Relative Influences on Social Presence, Task Participation, and Group Consensus," *MIS Quarterly* (25:3), September 2001, pp 371-390.

Zendler, A. "A Preliminary Software Engineering theory as Investigated by Published Experiments," *Empirical Software Engineering* (6) 2001, pp 161-180.

APPENDICES

XP ACTIVITES TABLE

No	Practices	Activities/Products			
1.	**Planning games**	Write Stories cards	Write Task cards	Estimate task, cost	Clients write simple test sets
2.	**Coding standard**	Identify standard	Naming convention		
3.	**System metaphor**	Common vision	Shared vocabulary	Analogies	Architecture
4.	**Simple design**	Story cards	X-machine	XX-machine	
5.	**Testing**	Test cases	Unit testing	Functional testing	Test suite
6	**Refactoring**	Early identification with X-machine			
7.	**Short/ Frequent release**	Continuous review with client	Early feedback from clients	Changes in requiremen t	
8.	**Continuous integration**	Incremental integration			
9.	**Pair programming**	2 person	Swapping pair		
10.	**Collective ownership**	Repository	Testing partner	Swapping modules	
11.	**On-site customer**	Write stories	Discuss stories	Immediate feedback	Discuss user interfaces
12.	**Forty-hour week (final week)**	Less activities during last week	Completed project	Tested project	Integrated software

STRUCTURED QUESTIONS FOR INTERVIEW

Story Cards

1. Is generation of story cards helped in creating test cases for both functional and non functional parts of the system?

2. Did the predicted time and the time required for the implementation match? Suggestions ways to overcome this problem.

3. Is it possible to design classes or method from the story cards? If N—what was used to assist the design process?

X Machine

1. It is easy to draw and use XM diagram to design the software? Why ?

2. Has the XM diagrams helped the team to understand the overall system?

   How? Why?

Early Releases

1. When was the first release? (months or weeks after commencing the project).

2. The early release gave the team an opportunity to improve which of the following areas. Functional Improvements / User Interfaces / Design improvements/ Others, state

3. Are all the reviews (releases) tested thoroughly? Why?

4. Do early review (releases) help in fixing bugs better? How? Why?

5. Did the first release cover most of the functionalities mentioned in the specification document? What percentage of the estimated functionalities were captured?

6. How many releases were made before the final release?

Coding Standards

1. What are the socing standard used by the team? (suggests Genesys coding standards) If N-- State the reasons and the source for your standards.

2. There was hardly any degradation of code due to the difference in coding standards between two XP partners working on the project?. If No, how is the standard maintain?

3. XP partners worked efficiently in the presence of proper coding standards? How? Why?

Code Refactoring

1. Do the team reuse the code? Estimate percentage of the code being reuse?

   If N, why ?

2. Does reuse of code help to speed up the development process? How? Why?

Testing.

1. When were the test cases developed?

2. It is easy to develop test cases before coding? If **Y**, how this is achieved. If **N**, state the reasons

3. Does the client help with the test cases?

4. All codes undergo unit testing?

5. Unit testing was done to helped the team to Discover and correct bugs/ Boost members' confidence/ Fulfil the company's requirement / Others, state the reasons

How often is Functional testing carry out?

Understanding XP

1. It is easy to understand the concept and theory of XP? If **Y**, state the reasons

If **N**, why and suggest ways to overcome the problems

2. When did you start to understand XP?

Applying XP to the Project

1. It is easy to **capture user requirement** using XP methodology?

If **Y** what feature in the XP practices that facilitates your works?

If **N** State the reasons and suggest ways to overcome the problems

If (**Y** and **N**) identify which features of XP that facilitates and hinder the development process

2. It is easy to **design the software** using XP methodology? **Y** **N**

If **Y** what feature in the XP practices and how does it facilitates your works?

If **N** State the reasons and suggest ways to overcome the problems

If (**Y** and **N**) identify which features of XP that facilitates and hinder the development process

3. It is easy to **develop codes** software using XP methodology? **Y** **N**

If **Y** what feature in the XP practices that facilitates your works?

If **N** State the reasons and suggest ways to overcome the problems

If (**Y** and **N**) identify which features of XP that facilitates and hinder the development process

Maintaining XP projects (state the project:.....)

1. It is easy to maintain an XP project?

If Y, what are the features that help ?

If **N** why and suggest the ways your team had done to overcome the problems

2. Will the next group be able to maintain your project easily? If **Y**, what are the features that will assist the future group? If **N**, Why?

Costs and Estimation

1. The project was completed under the expected range of time. If **N**, state the reasons.

2. The jobs given to each member was comfortably executed and the team was not over burdened during development schedule. If N, state the reasons.

WARR'S WELL-BEING MEASURES

Name: _____ Team: __

Date: _____

This scale consists of a number of words that describe different feelings and emotions.

During the past week, how much of the time has **your software project** made you feels:

1- Very slightly or not at all
2- A little
3- Moderately
4- Quite a bit
5- Extremely

(Please tick / the appropriate column for each feeling)

No.	Emotions/Feelings	1	2	3	4	5
1	Tense					
2	Miserable					
3	Depressed					
4	Optimistic					
5	Calm					
6	Relaxed					
7	Worried					
8	Enthusiastic					
9	Anxious					
10	Comfortable					
11	Gloomy					
12	Motivated					

The Grouping in the Warr scale

WARR SCALE

Anxiety	Tense Worried Anxious
Contentment	Calm Relaxed Comfortable
Depression	Miserable Depressed Gloomy
Enthusiasm	Optimistic Enthusiastic motivated

Anxiety-Contentment
Depression-Enthusiasm

POSITIVE AFFECT AND NEGATIVE AFFECT QUESTIONNAIRE

Name: _____ **Team:**_____

Date (answering the questionnaire) : _____

This scale consists of a number of words that describe different feelings and emotions.

During the past week, how much of the time has **your software project** made you feels:

1- Very slightly or not at all
2- A little
3- Moderately
4- Quite a bit
5- Extremely or Very

(Please tick / in the appropriate column for each feeling)

Feeling/Emotion		1	2	3	4	5	Feeling/Emotion		1	2	3	4	5
1	Interested						11	Irritable					
2	Upset						12	Inspired					
3	Scare						13	Attentive					
4	Proud						14	Afraid					
5	Ashamed						15	Excited					
6	Determined						16	Guilty					
7	Active						17	Enthusiastic					
8	Distressed						18	Alert					
9	Strong						19	Nervous					
10	Hostile						20	Jittery					

WORK GROUP COHESION QUESTIONNAIRE

Name: _____ **Team** :_____ **Date :**_____

This questionnaire consists of a number of words that describe different situations, feeling and emotions. For each item indicate to what extend you have felt this way during this week. Circle a number that represent your feeling.

1 Not at all or slightly 2 A little 3 Moderately 4 Quite a bit 5 Very or Extremely

1.	To what extent are individuals in your project team *friendly*?	1	2	3	4	5
2.	How *often* do you do things socially with individuals in your project team outside of the project?	1	2	3	4	5
3.	How *often* do you discuss important personal problems with individuals in your project group?	1	2	3	4	5
4.	To what extent are individuals in your project team *helpful* to you in getting your work done?	1	2	3	4	5
5.	To what extent do you *trust* individuals in your project team?	1	2	3	4	5
6.	To what extent do individuals in your project team take an *interest* in you?	1	2	3	4	5
7.	To what extent do individuals in your project team take an *interest* in you?	1	2	3	4	5
8.	To what extent do individuals in your project team *do favours* for you at considerable cost to themselves?	1	2	3	4	5
9.	How much do you *know* about the individual in your project team?	1	2	3	4	5

The following 4 pages of the Appendix illustrated the sequence of the process from the MS document, analysed by using the Atlas.ti package and finally written in the thesis.

Page 184 is a sample of an interview transcript in the Word document

Page 185 is a screen shot of the package Atlas.ti.

1. The left column is the interview transcript as seen through the package. The transcript is known as P7 (Primary document 7). The text in blue is selected as a quotation. This quotation is selected to associate with the code Metaphor (the reason being the display screen is used to identify requirement and not metaphor).
2. The *Quotations* pop-up window shows all of the selected quotes from several transcripts. The quotation (7.1) is highlighted.
3. The *Codes* pop-up window shows all the themes identified. The code Metaphor (8.0) indicates that there are 8 quotations associated with the Metaphor theme.
4. The *Focussed Network on Metaphor* pop-up window shows the semantic network of one theme (metaphor) with 8 quotations associated with the code. The box (7.1) is selected (shown as back box) to highlight the association.

Page 186-187 is an output of the dialog (HU) selected for the Metaphor code.

Page 188 is the quotation as written in the thesis.

(Sample of an interview transcripts)

There is no sharing of test cases. We use the tool for testing which you can add the general test set. We run the tool (something like Junit) on all the classes. We debug, test it as we are writing the code. It helps a lot because you need to see what worked and what don't. we do it up to a point, then do it again. You write the test cases to meet the requirement but actually to measure it. You can write code and know that it work or you can write code and know it might work. If you test it, you can see the code in action. You are being analytical when testing than when coding.

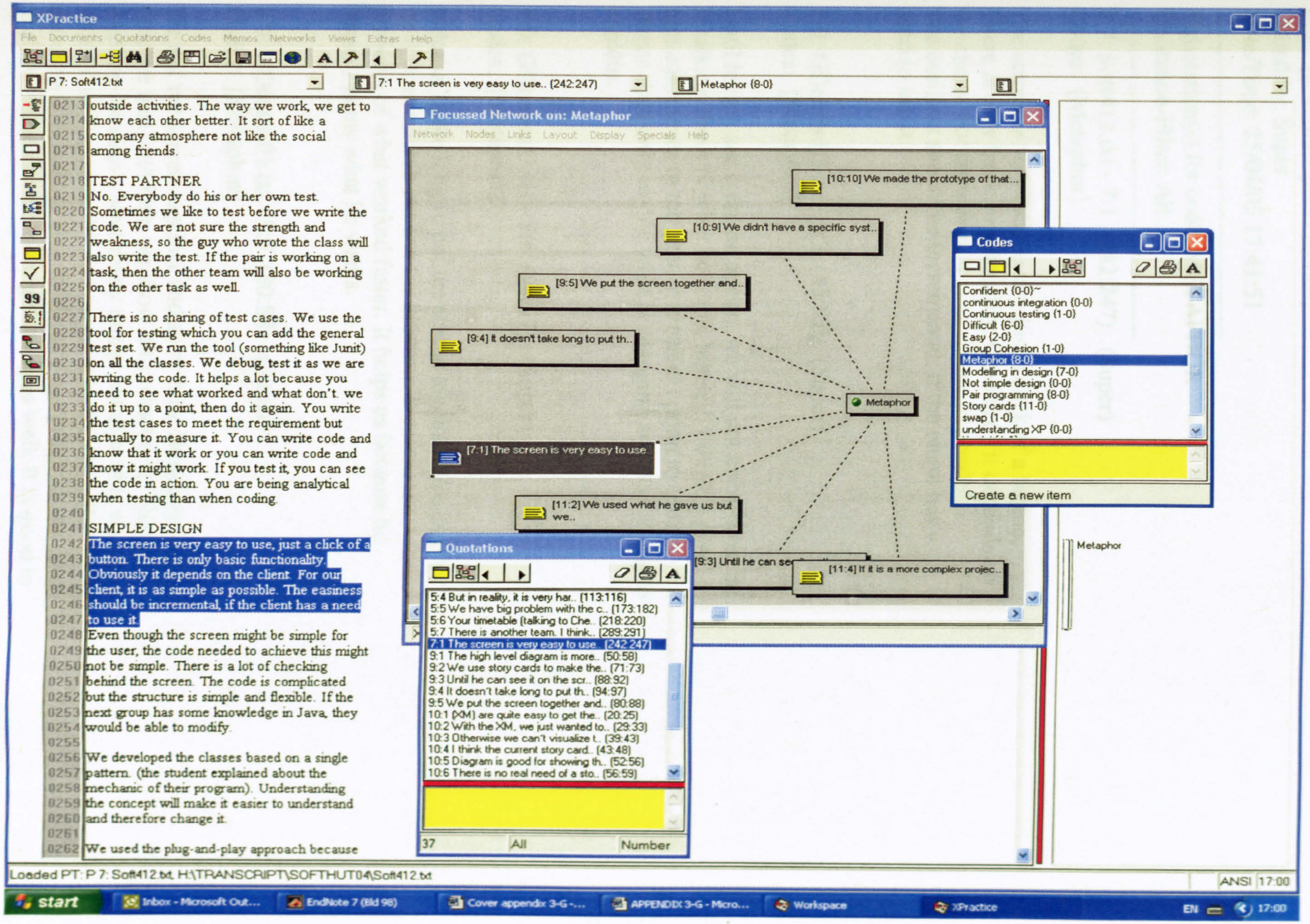
SIMPLE DESIGN

The screen is very easy to use, just a click of a button. There is only basic functionality. Obviously, it depends on the client. For our client, it is as simple as possible. The easiness should be incremental, if the client has a need to use it.

Even though the screen might be simple for the user, the code needed to achieve this might not be simple. There is a lot of checking behind the screen. The code is complicated but the structure is simple and flexible. If the next group has some knowledge in Java, they would be able to modify.

We developed the classes based on a single pattern. (the student explained about the mechanic of their program). Understanding the concept will make it easier to understand and therefore change it.

We used the plug-and-play approach because we were working in 3 different group or else no way we are going to integrate it. Basically, we developed 2 algorithms and then decided on the codes to implement it.


XPractice
File Documents Quotations Codes Memos Networks Views Extras Help
P 7: Soft412.txt
7:1 The screen is very easy to use.. (242:247)
Metaphor {8-0}
outside activities. The way we work, we get to know each other better. It sort of like a company atmosphere not like the social among friends.
TEST PARTNER
No. Everybody do his or her own test. Sometimes we like to test before we write the code. We are not sure the strength and weakness, so the guy who wrote the class will also write the test. If the pair is working on a task, then the other team will also be working on the other task as well.
There is no sharing of test cases. We use the tool for testing which you can add the general test set. We run the tool (something like Junit) on all the classes. We debug, test it as we are writing the code. It helps a lot because you need to see what worked and what don't. we do it up to a point, then do it again. You write the test cases to meet the requirement but actually to measure it. You can write code and know that it work or you can write code and know it might work. If you test it, you can see the code in action. You are being analytical when testing than when coding.
SIMPLE DESIGN
The screen is very easy to use, just a click of a button. There is only basic functionality. Obviously it depends on the client. For our client, it is as simple as possible. The easiness should be incremental, if the client has a need to use it.
Even though the screen might be simple for the user, the code needed to achieve this might not be simple. There is a lot of checking behind the screen. The code is complicated but the structure is simple and flexible. If the next group has some knowledge in Java, they would be able to modify.
We developed the classes based on a single pattern. (the student explained about the mechanic of their program). Understanding the concept will make it easier to understand and therefore change it.
We used the plug-and-play approach because
Focussed Network on: Metaphor
Network Nodes Links Layout Display Specials Help
[10:10] We made the prototype of that...
[10:9] We didn't have a specific syst..
[9:5] We put the screen together and..
[9:4] It doesn't take long to put th..
Metaphor
[7:1] The screen is very easy to use..
[11:2] We used what he gave us but we..
[9:3] Until he can se
[11:4] If it is a more complex projec..
Codes
Confident {0-0}~
continuous integration {0-0}
Continuous testing {1-0}
Difficult {6-0}
Easy {2-0}
Group Cohesion {1-0}
Metaphor {8-0}
Modelling in design {7-0}
Not simple design {0-0}
Pair programming {8-0}
Story cards {11-0}
swap {1-0}
understanding XP {0-0}
Create a new item
Metaphor
Quotations
5:4 But in reality, it is very har.. (113:116)
5:5 We have big problem with the c.. (173:182)
5:6 Your timetable (talking to Che.. (218:220)
5:7 There is another team. I think.. (289:291)
7:1 The screen is very easy to use.. (242:247)
9:1 The high level diagram is more.. (50:58)
9:2 We use story cards to make the.. (71:73)
9:3 Until he can see it on the scr.. (88:92)
9:4 It doesn't take long to put th.. (94:97)
9:5 We put the screen together and.. (80:88)
10:1 (XM) are quite easy to get the.. (20:25)
10:2 With the XM, we just wanted to.. (29:33)
10:3 Otherwise we can't visualize t.. (39:43)
10:4 I think the current story card.. (43:48)
10:5 Diagram is good for showing th.. (52:56)
10:6 There is no real need of a sto.. (56:59)
37 All Number
Loaded PT: P 7: Soft412.txt, H:\TRANSCRIPT\SOFTHUT04\Soft412.txt
ANSI 17:00
start
Inbox - Microsoft Out...
EndNote 7 (Bld 98)
Cover appendix 3-G -...
APPENDIX 3-G - Micro...
Workspace
XPractice
EN 17:00

HU: XPractice
File: [h:\transcript\XPractice]
Edited by: Super
Date/Time: 25/04/05 17:41:51
---
8 quotation(s) for code: METAPHOR
Quotation-Filter: All

P 7: Soft412.txt - 7:1 (242:247) (Super)
Codes: [Metaphor]

The screen is very easy to use, just a click of a button. There is only basic functionality. Obviously it depends on the client. For our client, it is as simple as possible. The easiness should be incremental, if the client has a need to use it.

P 9: GDentist03.txt - 9:3 (88:92) (Super)
Codes: [Metaphor]

Until he can see it on the screen, we will not know what he is talking about or what he wants himself. I think it give him confidence to tell you. It is easier t point to certain screen than only saying what is changing and than trying to explain.

P 9: GDentist03.txt - 9:4 (94:97) (Super)
Codes: [Metaphor]

It doesn't take long to put the screen together. We didn't do it from scratch but use the browser. It solves the problem of what worked faster. It helps us because he doesn't know what he wants.

P 9: GDentist03.txt - 9:5 (80:88) (Super)
Codes: [Metaphor]

We put the screen together and show him. Basically, every time we met with him, we show him something. The first meeting we didn't have many things but after that we add a little bit of functionality and reassure him that ... By showing him, we are able to identify our progress and also to show/ identify where we go wrong as well. It is good to

be able to discuss and get him to say yes.

P10: GDon03.txt - 10:9 (122:123) (Super)
Codes: [Metaphor]

We didn't have a specific system metaphor but we have a diagram.

P10: GDon03.txt - 10:10 (63:72) (Super)
Codes: [Metaphor]

We made the prototype of that. We just used simple user interface, we have two computers, and we link them up to show how the agency can interact. There were a couple of agencies on that day. There was a huge progress because some of the agency can say what they like and what they don't like. We found out a lot of them have reservation towards the system. With those that come, we can come to some sort of conclusion on that meeting.

P11: GTed03.txt - 11:2 (148:153) (Super)
Codes: [Metaphor]

We used what he gave us but we also suggested options of what can be done. We gave him screen shots that can be done. The whole admins side, where he wanted to learn so that he could make changes to his website that was impressive to the client. This approach speeds up his process of understanding.

P11: GTed03.txt - 11:4 (267:270) (Super)
Codes: [Metaphor]

If it is a more complex project, I would eliminate XM diagram and use the collaboration diagram instead. These diagrams have been in use already in the company.

(The quotation below as written in the thesis after modifying the original output from quotation)

The students expressed that by using this approach, they discovered that the clients were able to identify their requirement easily.

P 7: Soft412.txt - 7:1 (242:247) (Super)
Codes: [Metaphor]

The screen is very easy to use, just a click of a button. There is only basic functionality. Obviously it depends on the client. For our client, it is as simple as possible. The easiness should be incremental, if the client has a need to use it.

P 9: GDentist03.txt - 9:3 (88:92) (Super)
Codes: [Metaphor]

Until he can see it on the screen, we will not know what he is talking about or what he wants himself. I think it give him confidence to tell you. It is easier t point to certain screen than only saying what is changing and than trying to explain.

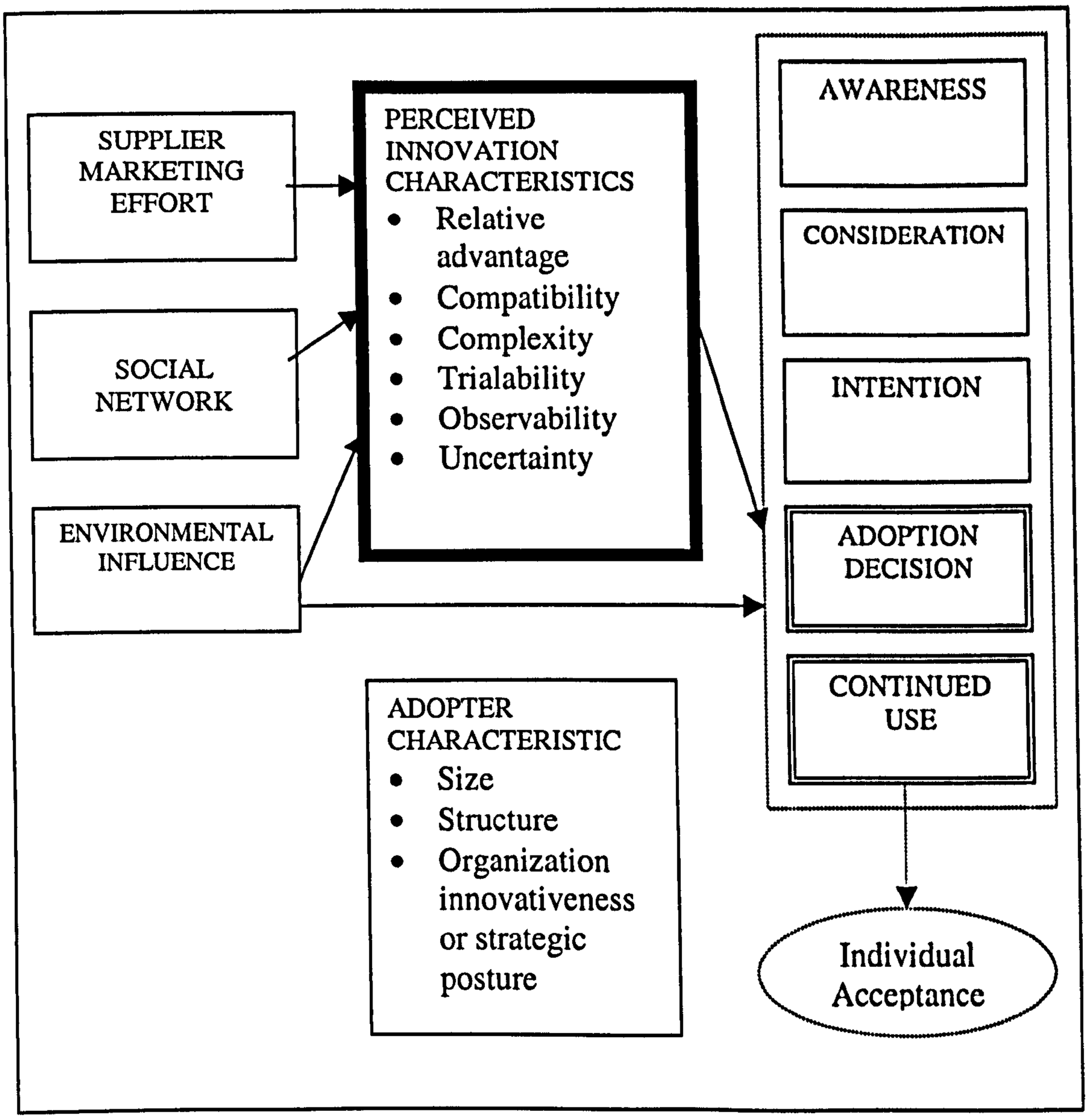
P 9: GDentist03.txt - 9:4 (94:97) (Super)
Codes: [Metaphor]

We didn't do it from scratch but use the browser. It solves the problem of what worked faster. It helps us because he doesn't know what he wants.

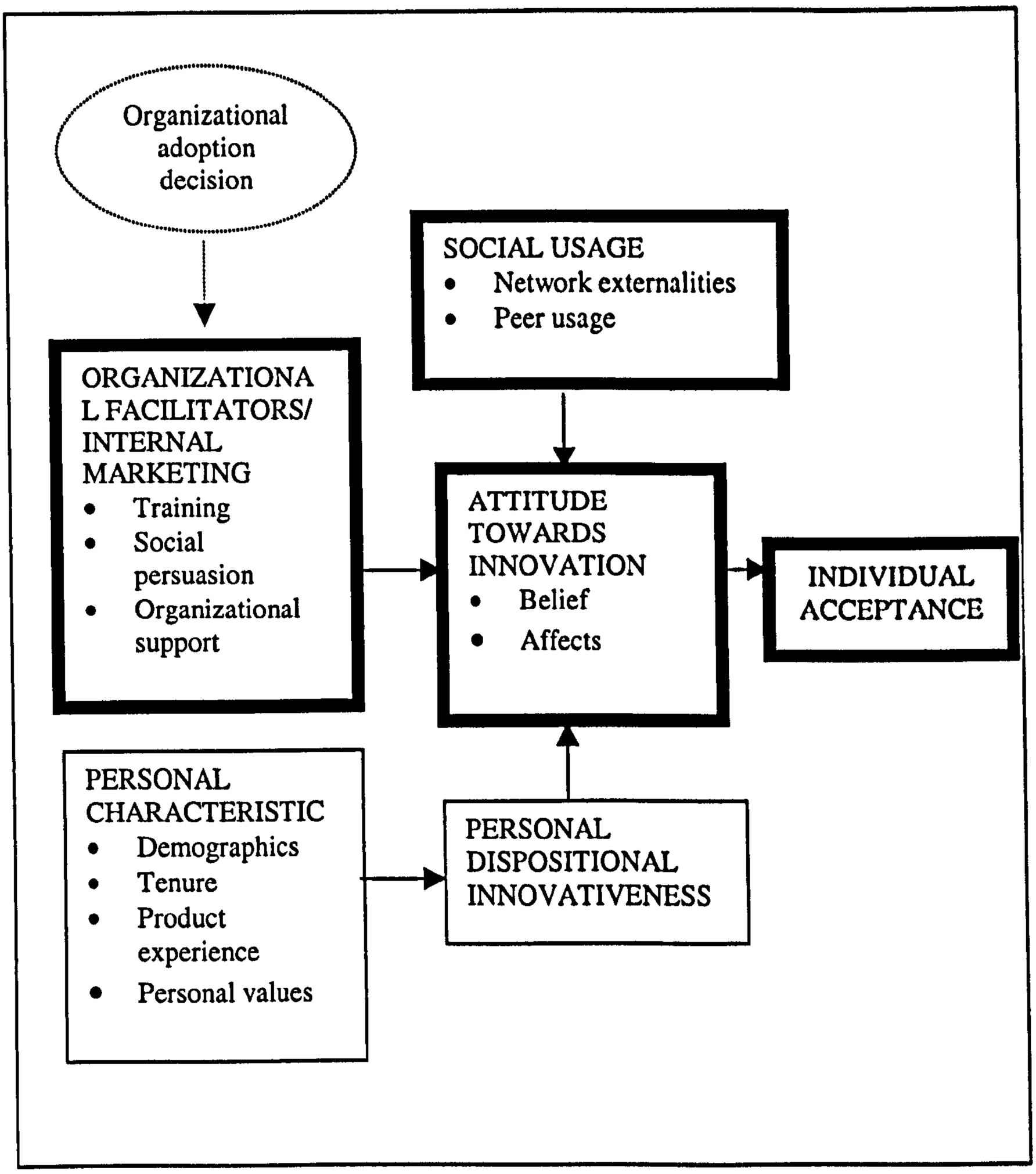
P11: GTed03.txt - 11:2 (148:153) (Super)
Codes: [Metaphor]

We gave him screen shots that can be done….. This approach speeds up his process of understanding.

A Conceptual Framework of Organizational Innovation Adoption



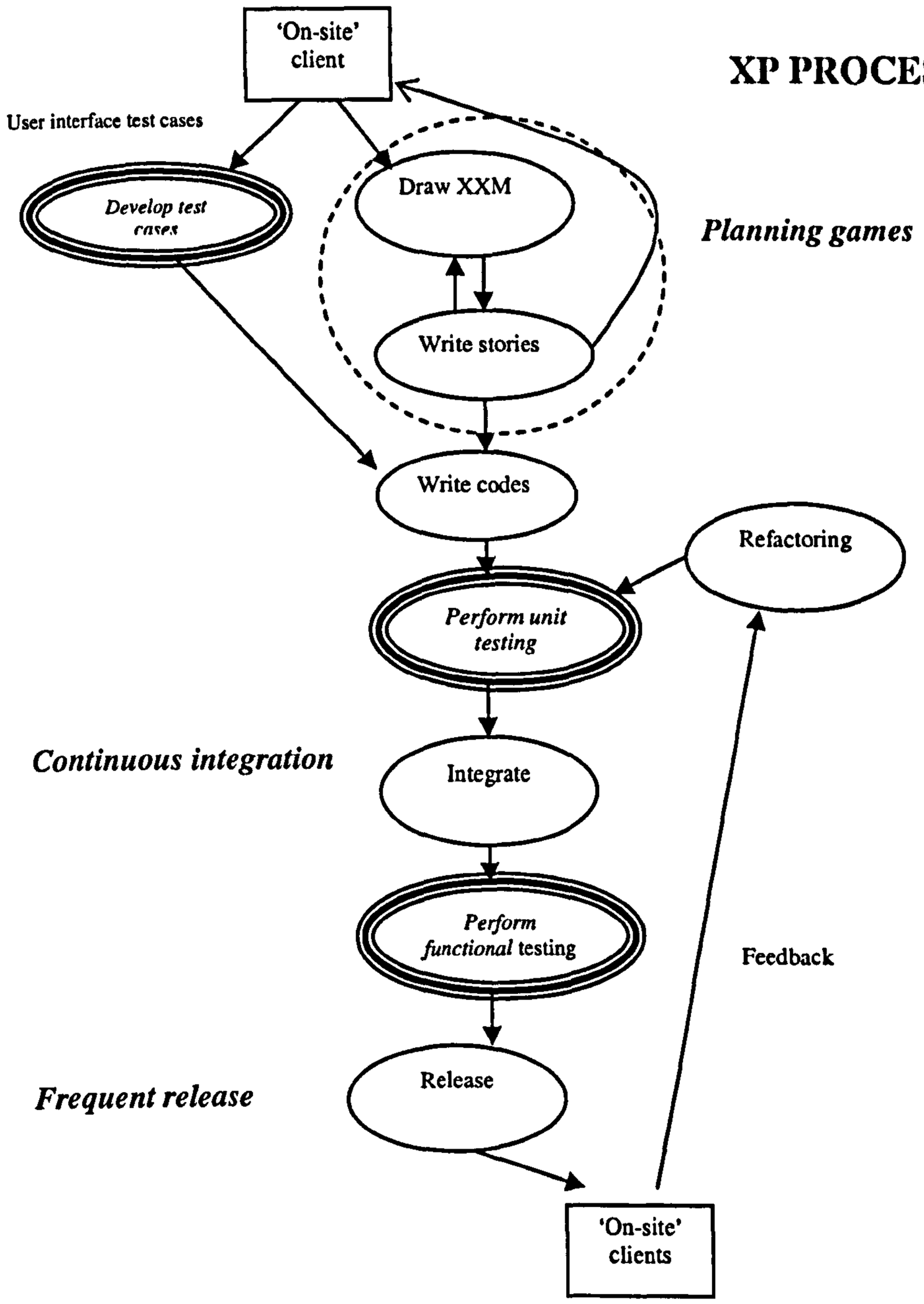
A Conceptual Framework of Individual Innovation Acceptance in Organizations



XP CATEGORISATION TABLE

Rule	Practical	Software Process	Product
Coding standard (Genesys)	Pair programming/tasking	Planning games	Test cases
System metaphor (architecture)	Collective ownership	Simple design	Story cards
15-hours week (maximum – every time)	On-site customer	Refactoring (whenever possible)	eXtreme X-machine diagram
		Testing (every time)	Tested codes
		Continuous integration	
		Frequent release	

XP PROCESS FLOW



XP PROCESS FLOW

Planning games

Continuous integration

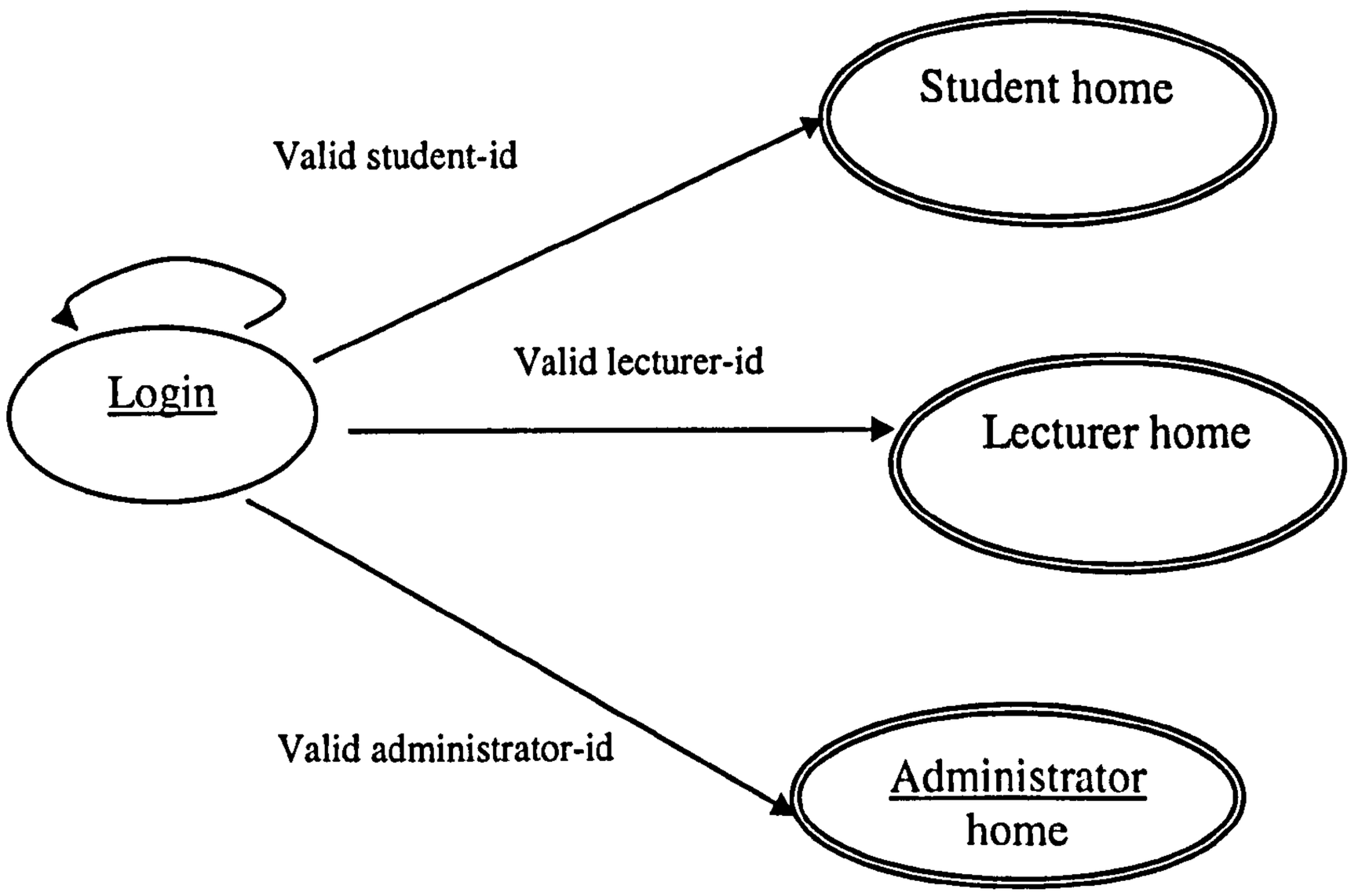
Frequent release

EXAMPLE

WEB BASED CONTROL PROJECT

WEB-BASED *CONTROL* PROJECT

MAIN XXM DIAGRAM



Administrator XXM Diagram

XP ACTIVITIES AT THE IBM, HURSLEY, UNITED KINGDOM

No	Practices	Activities/Products			
1.	Planning games **yes**	Stories cards **yes**	Task cards yes – we now have a large pinboard on which unfinished stories/tasks are placed according to iteration	Estimate task **yes**	Estimate cost **yes**
2.	Coding standard yes – automated checking – but we haven't been all that strict :o)	Identify standard yes – we agreed on the look and feel of the code at the start – our own standard	Naming convention yes – sometimes this is dictated to us by the encapsulating middleware product conventions		
3.	System metaphor no – we couldn't think of a good one at the start and never went back to it later	Common vision **yes**	Shared vocabulary **yes**	Analogies **sometimes**	Architecture yes – much of the architecture (how we fit into the middleware solution) is defined outside the XP team
4.	Simple design **yes**				
5.	Testing **yes**	Test cases **yes**	Unit testing **yes**	Functional testing yes (acceptance tests)	Test suite yes - automated
6	Refactoring **yes**	Yes – informal (i.e. as part of other user stories). It has been commented that			

No	Practices	Activities/Products			
		we could do this more)			
7.	Short/ Frequent release **yes**	Continuous review with client yes – often via intermediate representative (too many customers to be talking to all the time)	Feedback from clients **yes**	Changes in requirement **yes**	
8.	Continuous integration	Incremental integration **yes**			
9.	Pair programming yes	2 person **yes**	Swapping pair yes – swapping encouraged		
10.	Collective ownership **yes**	Repository yes – all product and test code (XP or not) is developed with a code repository	Testing partner – not sure what this is?	Swapping modules yes – well, everyone works on all areas	
11.	On-site customer no – not feasible different 'customers' spread across globe	Write stories with customer , no – customer representative writes stories	Discuss stories yes – but sometimes through an intermediate (non-XP) 'architecture' team representing all customers	Immediate feedback **sometimes**	Discuss user interfaces yes – via intermediate (non-XP) 'architecture' team on which each customer is represented .
12.	Forty-hour week (final week) yes - XP team rarely work late/weekends	Less activities during last week Last week of project? Haven't got there quite yet- we had some last minute customer user	Completed project nearly finished this release and about to start on the next release (although requirements	Tested project **yes**	Integrated software **yes**

No	Practices	Activities/Products			
		stories which have made things busier	gathering for the next release has been underway for some time)		

XP Questions

The following are some questions (I would like to know how your team achieve them) and the Genesys's (an in-house software company) attempted efforts to achieve them.

1. Pair programming (pp)

a. How easy do the team members accept pp?

From the interview sessions with the in-house developer, I discover that the students find it very difficult to do this practice due to their different programming ability in certain programming languages. The students who can program better said that pp slow them down and those who are slower refuse to do pp because they said that they are not learning by looking and prefer to learn the language by developing simpler modules. Pairing only started during the 3rd months or when they started to do serious testing.

b. What are the criteria that decide the partners?

In our case, we let the students decide among themselves. Most of them give comfortable with each other and timetable suitability as the criteria for choosing their partners.

c. How often do you swap between partners?

Partners are swap when the different pairs have problem with their work and the most skilful person will be switching from one pair to the other.

d. Swapping modules

In our case, swapping module is done for testing. *Is the swapping of modules part of the IBM policy or does the XP team practice it only?*

2. Frequent release

So far, we have not managed to do this. *How do your team manage to release part of the project? What kind of modules are possible to release?*

3. Testing

a. Testing partners

Since the in-house company do not have a specific testing team, the pairs will swap modules/ project to be tested by the other pair. *How is testing conducted in your team?*

b. Test first coding

I have been interviewing 2nd year and MSc students for my data. Only the MSc students easily adopts this practice and I attribute this to the lack of experiences on the part of the 2nd year students .
Do your team find this approach difficult, if so how do you overcome it?
Is every members involved in identifying the test cases?

This questions were answered through the paper written as quoted in the reference.

SOFTWARE HUT – CLIENT'S ASSESSMENT SHEET

This sheet is meant to help in the assessment of the suitability of the delivered system. The mark obtained for each team will contribute 50% towards their final module mark

Group number:_____

Documentation

Presentation	[5 marks]	.....
User Manual	[5 marks]	.....
Installation guide	[5 marks]	.....

Software system

Ease of use	[5 marks]	.....
Error handling	[5 marks]	.....
Understandability (use of appropriate language etc)	[5 marks]	.....
Base Functionality (completeness)	[5 marks]	.....
Innovation (extra features)	[5 marks]	.....
Robustness (correctness – doesn't crash)	[5 marks]	.....
Happiness with product	[5 marks]	.....

TOTAL _____

Signed: .....

SOFTWARE HUT -LECTURER ASESSMENT (DISCOVERY)

Marking Scheme (Discovery group)

Group number: _____

Requirement Documentation _____ [5 marks]
A set of user stories
A statement of requirements, signed off by the client.

A detailed design for the proposed system, using an appropriate language.
_____ [10 marks]
Comments

Completed the test results. _____ [10 marks]
Acceptance of test report.
Comments.

Complete code listing. _____ [10 marks]
This must satisfy the coding standards (where appropriate) together with full documentation.
Comments.

User Documentation. _____ [5 marks]
Installation instructions, maintenance guide/manual.
Comments

A commentary on the project (maximum 10 sides) containing: _____[10 marks]
A log of the project describing important milestones
A description of the group structure, the roles of individuals and the mechanism for communication used between group members.
A description of the quality control strategy (who did what, when and how the acceptance criteria was defined)
A list any references to the literature used during the project.
An evaluation of the group performance including an allocation of the proportional effort contributed to the project by individuals in the group. this statement must be signed by all members of the group.
Please supply copies of all team meeting minutes, showing responsibilities for actions, the progress and the revisions made to the project plan.
Comments.

Total process mark: _____ [50 marks]

This mark is combined with the client's mark which is also out of 50 marks

SOFTWARE HUT -LECTURER ASESSMENT (XP)

Marking Scheme (XP group)

Group number: _____

Requirement Documentation _____ [5 marks]
A set of user stories. A statement of requirements, signed off by the client.

A detailed specification of test cases for the proposed system, using an appropriate language.
_____ [5 marks]
Comments

Test management process. _____ [5 marks]
How do you develop and apply the test? Is there any scripts that you need to develop to automate the process? Use of tools.
Comments.

Completed the test results. _____ [10 marks]
Acceptance of test report.
Comments.

Complete code listing. _____ [10 marks]
This must satisfy the coding standards (where appropriate) together with full documentation.
Comments.

User Documentation. _____ [5 marks]
Installation instructions, maintenance guide/manual.
Comments

A commentary on the project (maximum 10 sides) containing: __[10 marks]
A log of the project describing important milestones
A description of the group structure, the roles of individuals and the mechanism for communication used between group members.
A description of the quality control strategy (who did what, when and how the acceptance criteria was defined)
A list any references to the literature used during the project.
An evaluation of the group performance including an allocation of the proportional effort contributed to the project by individuals in the group. this statement must be signed by all members of the group. evidence of pair programming.
Please supply copies of all team meeting minutes, showing responsibilities for actions, the progress and the revisions made to the project plan.
Comments.

Total process mark: _____ **[50 marks]**

This mark is combined with the client's mark which is also out of 50 marks