Aligning XP with ISO 9001:2000 - TickIT Guide 5.0
- A case study in two academic software projects.

David Vitoria

School of Engineering
Blekinge Institute of Technology
Box 520
SE - 372 25 Ronneby
Sweden
This thesis is submitted to the School of Engineering at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Software Engineering. The thesis is equivalent to $2 \times 20$ weeks of full time studies.

**Contact Information:**

Author: David Vitoria  
Address: Lindblomsvägen 102, 08252-1  
SE - 37233 Ronneby  
Sweden  
E-mail: ddelano@computer.org

University advisor(s):  
Kari Rönkkö  
School of Engineering

School of Engineering  
Blekinge Institute of Technology  
Box 520  
SE - 372 25 Ronneby  
Sweden  
Internet : www.tek.bth.se  
Phone : +46 457 38 50 00  
Fax : +46 457 271 25
Abstract

Since last four to five years the buzz of continuous growing of agile development has been spread all around the world, specially Extreme Programming as the most important methodology of this kind. From the other side, ISO 9001:2000-TickIT Guide 5.0 has been established mainly in Europe as one of the well-known Quality Management Systems, in order to create a continuous software process improvement throughout software organizations that is mainly related to a certification process. This thesis is developed to mainly answer the question of how Extreme Programming and the standard ISO 9001:2000 with its interpretation for software development as TickIT Guide 5.0 can be used together, in order to have a continuous software process improvement with the agility to respond quickly to environment changes and satisfy the customer needs and expectations.

<table>
<thead>
<tr>
<th><strong>A</strong></th>
<th><strong>Questionnaire</strong></th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.1</td>
<td>Opening the interview</td>
<td>100</td>
</tr>
<tr>
<td>A.2</td>
<td>The customer company</td>
<td>100</td>
</tr>
<tr>
<td>A.3</td>
<td>The project</td>
<td>100</td>
</tr>
<tr>
<td>A.4</td>
<td>XP and TickIT 5.0</td>
<td>101</td>
</tr>
<tr>
<td>A.5</td>
<td>Closing the interview</td>
<td>102</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>B</strong></th>
<th><strong>Manifesto for Agile Software Development</strong></th>
<th>103</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.1</td>
<td>Principles behind the Agile Manifesto</td>
<td>103</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>C</strong></th>
<th><strong>Agile development v/s TickIT Guide 5.0</strong></th>
<th>105</th>
</tr>
</thead>
</table>

| **D**   | **Acronyms List**                            | 109 |
List of Figures

4.1 Courage formula in XP ........................................... 29
4.2 Twelve XP practices .............................................. 32

6.1 Iterations in W.H.A.T. Project ................................ 52
6.2 Individual Task Velocity Metric in W.H.A.T. project. ........ 56

7.1 Pair Programming in WAIS Project ............................... 68
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Colors used in some tables</td>
<td>4</td>
</tr>
<tr>
<td>2.1</td>
<td>Quality Management Principles behind TickIT 5.0</td>
<td>16</td>
</tr>
<tr>
<td>2.2</td>
<td>ISO 9001:2000 structure in TickIT Guide 5.0</td>
<td>16</td>
</tr>
<tr>
<td>4.1</td>
<td>Different XP practices</td>
<td>31</td>
</tr>
<tr>
<td>4.2</td>
<td>XP Values</td>
<td>41</td>
</tr>
<tr>
<td>4.3</td>
<td>XP Principles</td>
<td>41</td>
</tr>
<tr>
<td>4.4</td>
<td>XP Practices</td>
<td>41</td>
</tr>
<tr>
<td>4.5</td>
<td>XP Activities</td>
<td>42</td>
</tr>
<tr>
<td>4.6</td>
<td>Roles used in XP</td>
<td>42</td>
</tr>
<tr>
<td>6.1</td>
<td>XP practices used in project</td>
<td>51</td>
</tr>
<tr>
<td>6.2</td>
<td>XP activities used in W.H.A.T. project</td>
<td>54</td>
</tr>
<tr>
<td>6.3</td>
<td>TickIT clauses compared with XP, W.H.A.T. project</td>
<td>63</td>
</tr>
<tr>
<td>6.4</td>
<td>Summary of XP Practices results in W.H.A.T. project</td>
<td>63</td>
</tr>
<tr>
<td>6.5</td>
<td>Summary of XP Activities results in W.H.A.T. project</td>
<td>64</td>
</tr>
<tr>
<td>6.6</td>
<td>Comparison summary of TickIT clauses with XP, W.H.A.T. project</td>
<td>64</td>
</tr>
<tr>
<td>7.1</td>
<td>XP practices used in WAIS. project</td>
<td>68</td>
</tr>
<tr>
<td>7.2</td>
<td>XP activities used in WAIS project</td>
<td>69</td>
</tr>
<tr>
<td>7.3</td>
<td>TickIT clauses compared with XP, WAIS project</td>
<td>76</td>
</tr>
<tr>
<td>7.4</td>
<td>Summary of XP Practices results in WAIS project</td>
<td>77</td>
</tr>
<tr>
<td>7.5</td>
<td>Summary of XP Activities results in WAIS project</td>
<td>77</td>
</tr>
<tr>
<td>7.6</td>
<td>Results of comparing TickIT clauses with XP, WAIS project</td>
<td>77</td>
</tr>
<tr>
<td>8.1</td>
<td>Comparison XP Practices results in both projects</td>
<td>79</td>
</tr>
<tr>
<td>8.2</td>
<td>Comparison XP activities results in both projects</td>
<td>79</td>
</tr>
<tr>
<td>8.3</td>
<td>Results of comparing TickIT clauses with XP in both projects</td>
<td>80</td>
</tr>
<tr>
<td>A.1</td>
<td>Roles used in XP</td>
<td>101</td>
</tr>
<tr>
<td>A.2</td>
<td>XP Practices</td>
<td>101</td>
</tr>
<tr>
<td>A.3</td>
<td>XP Activities</td>
<td>101</td>
</tr>
<tr>
<td>A.4</td>
<td>Conflictive areas in TickIT with project X</td>
<td>102</td>
</tr>
<tr>
<td>C.1</td>
<td>Comparison between agile values and QMS principles, TickIT Guide 5.0</td>
<td>106</td>
</tr>
<tr>
<td>C.2</td>
<td>Comparison between agile principles and QMS principles, TickIT Guide 5.0</td>
<td>108</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

There is no single development, in either technology or management technique, which by itself promises even one order-of-magnitude improvement within a decade in productivity, in reliability, in simplicity.

- Fredrik P. Brooks, Jr. (Brooks, 1995, p. 179)

The goal of this chapter is to illustrate the research background, research aims, and research objectives. In addition, the reader can find the structure of this thesis.

1.1 Background and motivation

Software process improvement models were developed to help software organizations to create good quality software products, using the knowledge obtained from the Total Quality Management in the software industry. There are three well know models CMM\(^1\) (Humphrey, 1989, Paul et al., 1993, Weber et al., 1993, Paul et al., 1995), CMMI\(^2\) (Ahern et al., 2003, Chrissis et al., 2003, Kupla & Johnson, 2003) and ISO 9001:2000 series\(^3\) (i.e. ISO 9000:2000, ISO 9001:2000, ISO 9004:2000), others models include ISO/IEC 15504 (Zahran, 1998), BOOTSTRAP (Zahran, 1998) and Trillium (Zahran, 1998). ISO 9001:2000 series contains standards for the manufacturing industry, but its interpretation to the software world is established in the TickIT 5.0 Guide for Software Development (Leakey & Restell, 2001) and it has been used mainly in Europe, supported by the UK and Swedish software industry. The standardization process started in 1991.

Agile methodologies were born in the mid's 90s due to traditional software development methods encountering problems in meeting product delivery deadlines. This meant that projects often exceeded budgets and changed constantly. The best-known agile methodologies are eXtreme Programming (XP) and SCRUM, but there are others such as Feature Driven Development (FDD),

\(^1\)http://www.sei.cmu.edu/cmm/
\(^2\)http://www.sei.cmu.edu/cmmi/
\(^3\)http://www.iso.org
Dynamic System Development Method (DSDM), Lean Development, Crystal, and Adaptive Software Development that belong to the same type of methodologies. All of these methodologies adhere to the agile manifesto created in February 11-13, 2001 in Salt Lake City, Utah, USA. Therefore it would be beneficial to know if ISO 9001:2000 - TickIT Guide 5.0 are compatible and if they can work together.

1.2 Research aim, questions and objectives

In this section the aim, research questions and the objectives of this thesis will be described.

1.2.1 Aim

The aim of this thesis is to investigate the relationships between XP practices and TickIT 5.0 and how software organizations can benefit from it. The main question in this research is:

How is XP related with TickIT Guide 5.0?

All the following research questions depends on it.

1.2.2 Research questions

The research questions that will be answered in this Case Study are the following:

- Are the agile development values and principles complementary with Quality Management Systems(QMS) principles and ISO 9001:2000 - TickIT Guide 5.0 or do they contradict each other?
- What are the strengths and weaknesses of each model?
- Which XP practices, activities, and roles were developed by each team?
- Are they using XP?
- Is an XP project compliant with TickIT 5.0 Guide?
- Can these models complement each other? In which areas?
- What are the benefits of using these models together?

1.2.3 Expected outcomes

By answering these questions the reader shall see how these models can complement each other. The answers shall also define the relationships between XP practices and TickIT Guide 5.0 clauses; and propose how a TickIT certificated organization can build their processes to easily respond to change using XP. This is achieved by use of a comparison and analysis of two academic software projects.
1.3 Objectives

Described below are the thesis objectives:


2. Develop a case study with projects that have been working in XP and shall be ISO 9001:2000 certificate according to TickIT Guide 5.0.

3. Make a comparison between the two projects using the literature mentioned above in order to determine the relationships between XP practices and ISO 9001:2000 -TickIT Guide 5.0 clauses.

1.4 Thesis structure

This thesis has 10 chapters and four appendices. In the following paragraphs is summarized the thesis structure.

**Chapter 1 Introduction:** Contains the introduction to this Msc. thesis. It gives an overview of the principal aspects of this report, and includes the background, research questions, expected outcomes, aims, objectives of this investigation, and thesis structure.

**Chapter 2 ISO 9001:2000 - TickIT Guide 5.0:** It is about ISO 9001:2000 - TickIT Guide 5.0, and describes which software process improvements methods are used in the software industry. The necessity for TickIT Guide 5.0 is explained as well as the QMS principles behind the standard. This is followed by a description of the different clauses in the TickIT Guide 5.0, and ultimately outlines the strengths and weaknesses of TickIT Guide 5.0.

**Chapter 3 Agile Software Development:** In this chapter agile development is described and discussed in terms of its values and principles.

**Chapter 4 Extreme Programming:** The fourth chapter is a discussion about Extreme Programming; describing and discussing its values, principles, practices, activities, and roles. Moreover, some other important features to consider about Extreme Programming(XP) are described in the final section of this chapter.

**Chapter 5 Research strategies and techniques:** This chapter describes the research strategy used in this thesis. A single-case study with and embedded design was used. Also outlined is the technique used for data-collection that was primarily interviews. Finally the methods used for data verification and validation were explained.

**Chapter 6 W.H.A.T. project:** This chapter begins with an overview of the W.H.A.T. project. This is followed by a description of the XP practices, activities, and roles that are used in this project. Finally a comparison is made between TickIT clauses and XP practices used in the project.

**Chapter 7 WAIS project:** Chapter seven begins with an overview of the WAIS project. This is followed by a description of the XP practices, activities, and roles that are used in this project. Finally a comparison is made between TickIT clauses and XP practices used in the project.

**Chapter 8 Case Study Analysis and Results:** This chapter describes the
results of the analysis of the contents of chapters 6, 7. The strengths and weaknesses of the use XP in both projects in relation to TickIT Guide 5.0 are discussed.

**Chapter 9 Conclusions and Recommendations:** This chapter details the conclusions and recommendations arising from the research carried out in this thesis. It also answers answering the questions developed in chapter 1.

**Chapter 10 Further research:** The tenth chapter outlines further research that can be developed from the findings of this thesis.

In addition this Msc. thesis contains three Appendices. Appendix A contains details of questionnaire used for the interviews. Appendix B outlines the Agile Manifesto and principles. Appendix C is a comparison between agile values, principles; QMS principles and ISO 9001:2000 clauses, and finally Appendix D contains a list of Acronyms used in this report.

In chapters 6, 7, and 8 the following colors were used in some tables. Their meaning are described in the following Table 1.1.

<table>
<thead>
<tr>
<th>Color</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Attention, problem.</td>
</tr>
<tr>
<td></td>
<td>Warning, small problem.</td>
</tr>
<tr>
<td></td>
<td>Pass, No problem</td>
</tr>
<tr>
<td></td>
<td>Not applicable.</td>
</tr>
</tbody>
</table>

Table 1.1: These are the colors used in some tables in this report

### 1.5 Reading Guidelines

This section describes who should read this report and how it should be read this report.

#### 1.5.1 Who should Read this Report

This report is intended for all software and management professionals, academics, and students who want to know about XP, the most well-known and used agile methodology, and ISO 9001:2000-TickIT Guide 5.0, a quality standard for software development.

- Software professionals and managers who are working for an ISO 9001:2000 certificated organization and want to know how to apply and adopt XP in their projects. Also, those who look for the benefits of applying XP in software projects.

- Software and computer science students who want to know about XP and the ISO 9001:2000 standard, and how they can work together.

- For academics that want to help their students in to understand how to use XP and TickIT Guide 5.0 together.
1.5.2 How to Read this Report

The report is intended to be easy to read and understand. With that purpose a color scheme for a comparison between ISO 9001:2000 - TickIT Guide clauses and XP developed in two academics projects. In that way it is easier for the reader to go through the comparison.

If the reader already has knowledge of TickIT Guide 5.0 and not about XP or agile methodologies he or she can skip chapter 2.

If the reader has knowledge about XP, but not about TickIT Guide 5.0 and the agile concepts behind XP, he or she can skip chapter 4.

If the case that the reader knows about agile development and XP, but not about TickIT Guide 5.0, then he or she can skip chapter 3 and 4.

In the case that the reader knows about TickIT Guide 5.0, agile methodologies and XP, he can skip chapters 2, 3, 4 and follow with the rest of the report.

Finally, if someone does not have enough time to read the chapters in depth. They can read the summaries of each chapter.

1.6 Summary

In this section one can understand the basic aspects of the report. For example, what is the background and motivation for this Msc. thesis, a description of the research aims, objectives and questions. Finally, one can obtain some guidelines about who should read this report and how the report should be read.
Chapter 2

ISO 9001:2000 - TickIT Guide 5.0

TickIT Guide 5.0 is the software interpretation of the quality standard ISO 9001:2000 using a quality management system as a base.
- David V. Délano, May 15th 2004

The goal of this chapter is to give an overview of the ISO 9001:2000 TickIT Guide, its principles, objectives, clauses, cost and benefits.

2.1 Introduction

Software process improvement started with the work developed by Watts Humphrey, associated with the Software Engineering Institute (SEI)\(^1\) in USA, as an initiative from the Federal Government of USA in 1987. The basis of that work are developed in the book ‘Managing the Software Process’ (Humphrey\(^1\) 1989), where a framework for process improvement was developed.

The basic idea was to apply the Total Quality Management\(^2\) (TQM) concepts that were developed in the manufacturing industry to the software industry. Number of years later SEI developed and the Capability Maturity Model (CMM) (Humphrey\(^1\) 1989, Paulk et al.\(^3\) 1993, Weber et al.\(^3\) 1993, Paul et al.\(^3\) 1995) based on his previous work. After that many models were created for different kinds of industries, but SEI incorporate them in only one model called Capability Maturity Model Integration (CMMI). For more information about this model, see (Ahern et al.\(^3\) 2003, Chrissis et al.\(^3\) 2003, Kupla & Johnson\(^3\) 2003).

In Europe the International Standard Association created the ISO 9001:2000 Standard for the manufacturing industry. Due to the fact that software development has its own particular problems such as huge complexity, high level of

\(^{1}\)http://www.sei.cmu.edu

\(^{2}\)Total Quality Management: ‘a constant endeavor to fulfill, and preferably exceed, customer needs and expectations at the lowest cost, by continuous improvement work, to which all involved are committed, focusing on the processes in the organization’ (Bergman & Klefsjo\(^3\) 2004, p. 34)
abstraction, errors that can be introduced easily, etc (Ahern et al. 2003). Then from 1991 in a joint effort between BSI Standards in UK and the Swedish Association for Testing, Inspection and Certification (SWETIC) that created the TickIT project. Due to the popularity in Europe of the ISO 9001:2000 - TickIT Guide 5.0, this standard will be used for this investigation instead of CMM.

There are 11 certification bodies, in addition, 1140 certificated companies are active through 44 countries, mainly in Europe, although there are some in USA, Canada, Japan, Australia, Egypt, India, Iran, etc. from all around the world. Moreover, there are 37 provisional TickIT auditors, 14 TickIT auditors, and 78 lead auditors.

The TickIT guide 5.0 has a main relationship with the following standards:

- ISO/IEC 12207:1995 Information Technology Software Life-Cycle Processes

ISO 9000:2000 is an International Standard that defines the principles, fundamentals concepts and terms underlying a Quality Management System (QMS). Its definition from the standard cited by (Bergman & Klefsjö 2003, p. 455) as ‘a management system to direct and control an organization with regard to quality’. The interpretation of this standard for software products is in Part E of TickIT Guide 5.0 (Leakey & Restell 2001).

These concepts and principles are used in the ISO 9001:2000 Standard to define the requirements for a QMS where each stage of the creation of a product is established in terms of quality (TickIT 5.0 Executive Overview 2000).

ISO 9001:2000 can be defined as a set of guidelines for QMS and is also related to the management of improvement within an organization (Bergman & Klefsjö 2003, p. 455). Moreover, the same principles and concepts defined in ISO 9000:2000 are used there. This standard is also used in TickIT guide 5.0 Part E. Part F of TickIT Guide is related to ISO/IEC 12207:1995 where a software life-cycle is defined, in order, to be implemented as a QMS (TickIT 5.0 Executive Overview). A correlation between the different clauses of part E and F is defined in the same guide (Leakey & Restell 2001).

The core of the TickIT guide 5.0 is ISO 9001:2000 Part E of the guide that will be explained in section 2.4 on page 10, because of the focus in this chapter is on that part of the TickIT 5.0 guide.

The principles behind ISO 9001:2000 are summarized in Table 2.1 on page 16.

3http://www.bsi-global.com/index.xalter
4http://www.setic.org
5The data obtained are from November 2003 http://www.tickit.org/certbods.pdf
6The data obtained are from April 2004 http://www.tickit.org/cert-org.htm
7The data obtained are from May 2004 http://www.tickit.org/Audlist3.pdf
8Emphasis from the author
9quality: 'the degree to which a set of inherent characteristics fulfills the requirements, i.e. needs or expectations that are stated, generally implied or obligatory'
based on TQM’s cornerstones as are described in (Bergman & Klefsjö 2003, c. 1). There are five TQM cornerstones:

- Focus on customers
- Base decision on facts
- Focus on processes
- Improve continuously
- Let everybody be committed.

Committed leadership is needed to make these five cornerstones interactive.

2.2 ISO 9001:2000 - TickIT Guide 5.0: QMS Principles

The eight quality management principles behind ISO 9001:2000 - TickIT Guide 5.0 (Leakey & Restell 2001) are described below (ISO 2000).

1. Customer focus
   Companies need to understand that they depend highly on the customer relationship. In order to survive they need to comprehend the current and future needs of the customer as well as their expectations and requirements.
   
   This principle is in connection with the definition of quality stated in ISO 9000:200010, and also with how Bergman and Klefsjö define quality of a product in (Bergman & Klefsjö 2003, p. 24)11.

   Finally, the authors (Bergman & Klefsjö 2003, p. 37) mention that this principle is about searching and identifying what customers want and need. This starts the process of how to fulfill customer desires with the manufacture and development of a specific product.

2. Leadership
   The managers should establish policies, mission and goals of the company, in order that all employees should follow their guidelines. They are also responsible for creating the necessary environment for the people working in the organization, in order that they can be committed to the company’s goals.
   
   The authors in (Bergman & Klefsjö 2003, p. 35) remarks the importance of management commitment in a quality initiative. Moreover, they have a whole chapter dedicated to leadership (Bergman & Klefsjö 2003, c. 17).

3. Involvement of People
   The core of any organization are people, and their commitment make possible that their abilities can be developed for company’s benefit. In order, to be successful with quality the authors in (Bergman & Klefsjö 2003, p. 45) determine that is necessary to create an environment for

---

10See quality in footnote 9 on the preceding page
11Quality of a product: ‘is its ability to satisfy, or preferably exceed, the needs and expectations of the customers’ (Bergman & Klefsjö 2003, p. 24).
participation to obtain continuous customer satisfaction with quality improvements. They add that everyone in the company should be committed and participate in an active form in this approach. This not only will be reflected in a better quality of the product, but also in the quality of the process.

Finally, they add that certain individual characteristic of the individual should be taken in consideration and developed to create collaboration, for example, self-reliance, conversational abilities, purposefulness, co-creativity, and ability to learn from the experience (Eklund & Lund 1999) cited by (Bergman & Klefsjö 2003, p. 46).

4. Process Approach

Good results are achieved with efficiency when the activities and their resources are administered as a process.

Process is defined by (Bergman & Klefsjö 2003, p. 40) as ‘a set of interrelated activities that are repeated over time’. They add that the process goals is customer satisfaction with the product developed with a minimum amount of people and resources. The authors remarks that this process should be a core part of the whole organization.

Finally, they add that the process should be measure as a whole, not with individual entities. This process view is detailed in (Bergman & Klefsjö 2003, c. 19).

5. System approach to management

Comprehending, determining and managing process that are related to each other as an entire system makes that the company is more effective and efficient in reaching their goals and objectives. This principle can be related to the general system theory developed by Ludvig von Bertalanffy in (von Bertalanffy 1976).

6. Continual improvement

One key objective within a company should be to follow a continuous process improvement for the development as an organization. Moreover, this objective should be permanent (Bergman & Klefsjö 2003, p. 42) if the company do not make this process a permanent one, they can lose market shares.

The base of continual improvement as is the improvement cycle “Plan-Do-Check-Out” explained in (Bergman & Klefsjö 2003, c. 9).

The authors also remark that the core rule of continuous improvement is ‘there is always a way to get improved quality using less resources’ (Bergman & Klefsjö 2003, p. 43) that turns to a win-win situation for everyone in the supplier company and the customer company. According to the authors in (Bergman & Klefsjö 2003, p. 44) continuous improvement can be taken using small and bigger increments.

7. Factual approach to decision making

This principle says that the decisions should be made using the analysis of concrete facts, that should be well-founded as the authors mention in (Bergman & Klefsjö 2003, p. 38). They also add that is necessary to split what are natural variations and variations due to distinguishable causes, this process requires knowledge and ability to perform it. Moreover, the
facts collected should be both quantitative and qualitative. Finally, they remark that a strategy is needed to perform the link between decisions that should be taken and the facts related to it. In manufacturing companies they have statistical and management tools to perform this link and are explained in (Bergman & Klefsjö 2003, c. 10, 22) respectively, and some of them have been used in project development as the Gantt Chart.

8. Mutual beneficial supplier relationships
This principle remarks that the customer and supplier company are per-se independent, but they can both benefit from a relationship that improve their capability to create value. If the reader wants to know about the supplier process on TQM, it is described in (Bergman & Klefsjö 2003, c. 13).

For more information about the principles, see (ISO 2000).

2.3 ISO 9001:2000 - TickIT Guide 5.0: Objectives
TickIT has two purposes that are presented below (TickIT 5.0 Executive Overview 2000):

• To increase and stimulate all people involved in software development about:
  – The meaning of quality in software development process.
  – How quality can be reached in software organization, in two folds, products and services.
  – How can be continuously developed the quality in a QMS.

• To provide software companies with a practical framework to effectively certificate their quality in processes and products through:
  – Publishing a guide that provide an understanding and interpretation for software companies about the ISO 9001 series.
  – Create and provide the necessary training, requirements and registration for auditors with demonstrable experience and ability in IT and SW projects.
  – Establish the rules that are necessary to accredit Certification Bodies that have experience with software companies.

2.4 ISO 9001:2000 - TickIT guide 5.0: Clauses
The TickIT 5.0 guide can be divided in five main clauses that will be explained in this section (Leakey & Restell 2001, Part E). These clauses will be used to compare ISO 9001: 2000 with XP and two students projects in chapters 6, 7, and 8.

• 4 Quality Management System
4.1 General Requirements
An organization need to define, develop, document and update a QMS, in addition to this its effectiveness shall be improved continually, and all of this shall be in accordance to ISO 9001:2000.

The organization need to define what are the processes involved in the QMS, define what kind of interaction will be between processes and their sequence.

Define what are the methods that will be used to guarantee effectiveness of the processes control and operation. In addition, the organization shall guarantee that all the needed resources and information used to support that daily operation of the QMS and its monitoring.

Moreover, the organization shall ensure that the processes can be monitored, analyzed and measured.

Finally, the organization shall establish the necessary actions to reach their planning results and follow a continuous processes improvement.

4.2 Documentation Requirements
This section of Leakey & Restell (2001, Part E) define what are the documents that are necessary in a QMS. That are the following: quality policy; quality objectives; quality manual; a document with the defined processes needed by ISO 9001:2000; documents established by the company to control, plan, and operate effectively their processes (e.g. project plans, project specifications, etc.); and finally the record needed by Leakey & Restell (2001, Part E) (e.g. review records, test results, audit reports, etc.). In addition, all these documents have to be defined, maintained and implemented.

5 Management Responsibilities

5.1 Management commitment
Top managers shall demonstrate that they are committed to build and implement a QMS and that should be effectively improved continuously.

The notion that customer meetings are as important as requirements defined in the organization shall be communicated in the whole organization. They are in charge to define the quality policy, and the quality objective are determined.

In addition, they shall conduct management reviews, and finally they can assure the resources needed for the QMS.

5.2 Customer focus
The work of top managers is to be sure that customer requirements are determined and achieved with the purpose to obtain customer satisfaction.

5.3 Quality Policy
Top management assure that the quality police is the right for their organization. In addition, that the quality policy comply with the QMS requirements and its effectiveness is improved continuously.

The quality policy also should create a framework where the quality objectives are reviewed and established.
The quality policy should take the organization where top management want to go.

5.4 Planning
The quality objectives should be established in the organization through all levels. These quality objectives should be follow the quality policy and be measurable.

Quality objectives should be used to measure quality and its defined characteristics as functionality, usability, reliability, portability, efficiency, etc. as are defined in the quality model of ISO/IEC 9126. Moreover, quality of the project should be also taken in consideration as cost, time, plan, etc.

Planning a QMS is developed in order to meet the quality objectives and requirements. In addition, integrity of the QMS should be developed continuously through maintenance of changes. Moreover, is remarked that planning should be a twofold activity, one is planning as an entire company, and planning for a functional level, and planning for improvements.

All of these should be ensure by top management.

5.5 Responsibility, authority and communication
Responsibility and authority is formally defined and communicated to the entire company. The responsibility to that is from top managers.

Some examples of project responsibilities and authorities should be taken in consideration for reporting lines, interfacing with the customer, stakeholder communication, etc. In addition, to this in the software release stage there is also a need to define responsibilities and authorities.

Top managers shall establish a middle manager that take care of the QMS, make the reports to them and ensure that everyone is aware of the customer needs and requirements.

Finally, Top managers also should assure that internal communication for the QMS is the appropriate.

5.6 Management review
Top managers shall perform continuous and planned reviews of the QMS to assure that is performing according to the quality policy and quality objectives. Moreover, it should support improvements to the QMS and its related processes and documents.

Reviews inputs are: results of audits, customer feedbacks, recommendations for improvements, etc.

Reviews outputs are: decisions in terms of the resources needed, improvement of the product, etc.

6 Resource Management

6.1 Provision of resources
The organization shall determine what kind and which amount the resources are needed for implementing, improving and maintaining the QMS. To achieve customer satisfaction through achieving its requirements.
6.2 Human resources

The adequate people should work to improve the quality of the product due to their personal skills, training and experience. People should be competent in what they are doing.

For example, people should have the necessary skills in some languages and testing techniques, comprehend what is the architecture that have been used, methods, etc.

6.3 Infrastructure

The company is in charge to provide the necessary infrastructure to achieve product requirements. The infrastructure goes from buildings, hardware, software to services (e.g. network connection).

6.4 Work environment

Working environment should be defined, managed to obtain conformity to the product requirements.

7 Product Realization

7.1 Planning of product realization

The product development should be in accordance to the requirements of the QMS. Documentation is needed, in order that conform the procedures in the organization (Bergman & Klefsjö 2003, p. 466).

7.2 Customer-related processes

Customer requirements should be specified and reviewed with other process related requirements defined in the organization. After that they can be communicate to the customer (Bergman & Klefsjö 2003, p. 466).

7.3 Design and development

Product development should be planned and controlled by the organization. In addition, the organization should define activities for validation, verification and review of the product. Records of the performed activities in the development and design should be defined and maintained (Bergman & Klefsjö 2003, p. 466).

7.4 Purchasing

Control of the purchasing process is necessary, in order that the product to be purchased fulfil the requirements of the design or development. The organization can select specific supplier knowing their skills to deliver a product that meet the necessary requirements (Bergman & Klefsjö 2003, p. 467).

7.5 Production and service provision

The development and service provision should be controlled, validate using instructions, equipment and procedures. Moreover, the product traceability should be controlled, and recorded in documents or other artifacts. Finally, the organization should be aware of customer property (Leakey & Restell 2001, Part E 7.5).

7.6 Control of monitoring and measuring devices

Measurement and monitoring should be taken to the devices involved in the software development, as example, new software that have been added, hardware, or software tools. It should be determined the
processes involved on it. All of these, in order, to conform customer requirements (Leakey & Restell 2001, Part E 7.6).

- **8 Measurement, analysis and improvement**
  - **8.1 General**
    Monitoring, measurements, analysis and improvements should be planned and established in the organization, in order to conform the customer requirement with the product to be developed, and improve the processes involved on it (Leakey & Restell 2001, Part E 8.1).
  - **8.2 Monitoring and measuring**
    Monitoring and measure customer satisfaction and dissatisfaction, in order that the organization, really knows how well the QMS is working. Internal audits should be defined, and implemented as well as measures to monitor the processes and the characteristics of the products, in order to verify conformity with the requirements (Bergman & Klefsjö 2003, p. 467).
  - **8.3 Control of nonconforming products**
    Products that do not conform the requirements should be founded and controlled in order to do not deliver or use these kind of products. Documentation of responsibilities and authorities should be developed to deal with nonconforming products. All of these should performed by the organization (Leakey & Restell 2001, Part E 8.3).
  - **8.4 Analysis of data**
    Collection and analysis of adequate data should be done to determine how effective and suitable is the QMS. Then it is possible to perform an evaluation of what improvements can continuously made to the system. This should be related to customer satisfaction, suppliers and the characteristic of processes and products (Leakey & Restell 2001, Part E 8.4).
  - **8.5 Improvement**
    The organization as whole is in charge to improve continuously the QMS using the needed actions as defining a plan and finding the opportunities for continuous improvements, in order to do that corrective and preventive actions can be taken. Moreover, other activities can be used as analysis of data, management reviews, etc (Leakey & Restell 2001, Part E 8.4).

### 2.5 ISO 9001:2000 TickIT Guide 5.0: Strengths and Weaknesses

The strengths and and weaknesses related to implement ISO 9001:2000 TickIT Guide 5.0 in a software organization (Leakey & Restell 2001, Part A) are described in the following subsections.

#### 2.5.1 Strengths

The benefits are related to use the QMS defined in TickIT, that define the necessary process for continuous improvement. The benefits are the following
according to (Leakey & Restell 2001, p. A10):

- An organization can improve the quality of their products.
- It can improve the efficiency of the processes.
- It can decrease costs due to failure\(^7\).
- The staff is highly satisfied and do not want to go to another company.

### 2.5.2 Weaknesses

Weaknesses are mainly related to costs for developing a ISO 9001: 2000 certification process using TickIT Guide 5.0.

- There is a cost related to the time involved by people involved in the company to initially create a QMS, moreover it there is a cost related to an external consultant that can help the organization to implement a QMS (Leakey & Restell 2001, p. A9).

- There is also an effort that the organization need to develop, in order to be prepare for the TickIT audit and cooperate with TickIT auditors as is described (Leakey & Restell 2001, p. A9).

- The fees that the organization shall pay for the first TickIT audit and the next ones to maintain the certification (Leakey & Restell 2001, p. A9).

- The effort and time that people should spend on improving and use the established QMS (Leakey & Restell 2001, p. A10).

- Usually the companies that are going for a certification process, only care about the certification, but not about a continuous process of improvement. This comes from the experience of the author in a software development company in his home country where the top manager says ‘In two year we will be certificate CMM level 2’, that is not the focus on CMM or TickIT, the aim here is on continuous process improvement.

- Improvements usually can fail due to mismanagement. (Nilsson 2003). The certification process needs commitment from senior managements, without it is impossible to carry out all the necessary changes in the organization, in order to develop a successful process.

- It turns into ambitious plans and goals (Nilsson 2003). The goals and plans are to high to develop them.

- TickIT requires documentation and some people start to create a lot of it, but then the company lose its mobility to deal with changes. The tricky issue is to develop only enough documentation to fulfill the requirements and do not lose mobility.

---

\(^7\)failure costs are cost involved in correcting defects; overrun costs of budget and schedule; indirect costs associated to people that are dissatisfy with their daily work; and finally indirect cost because of bad quality products (Leakey & Restell 2001, p. A10).
2.6 Summary

- In the introduction, the reader can find where the ISO 9001:2000 - TickIT Guide 5.0 come from, where people are using this certification, and obtain a brief overview of what is.

- Then the reader can obtain an overview of its principles in section 2.2 on page 8. See Table 2.1.

<table>
<thead>
<tr>
<th>Code</th>
<th>Quality Management Principles</th>
</tr>
</thead>
<tbody>
<tr>
<td>QMSPrinciple1</td>
<td>Customer Focus</td>
</tr>
<tr>
<td>QMSPrinciple2</td>
<td>Leadership</td>
</tr>
<tr>
<td>QMSPrinciple3</td>
<td>Involvement of People</td>
</tr>
<tr>
<td>QMSPrinciple4</td>
<td>Process Approach</td>
</tr>
<tr>
<td>QMSPrinciple5</td>
<td>System approach to management</td>
</tr>
<tr>
<td>QMSPrinciple6</td>
<td>Continual improvement</td>
</tr>
<tr>
<td>QMSPrinciple7</td>
<td>Factual approach to decision making</td>
</tr>
<tr>
<td>QMSPrinciple8</td>
<td>Mutual beneficial supplier relationships</td>
</tr>
</tbody>
</table>

Table 2.1: Quality Management Principles behind ISO 9001:2000 - TickIT Guide 5.0

- The objectives are summarized in section 2.3 on page 10.

- The clauses in ISO 9001:2000 - TickIT Guide 5.0 are summarized in section 2.4 on page 10. See Table 2.2.

<table>
<thead>
<tr>
<th>ISO 9001:2000, TickIT 5.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 Quality Management System</td>
</tr>
<tr>
<td>5 Management responsibility</td>
</tr>
<tr>
<td>6 Resource Management</td>
</tr>
<tr>
<td>7 Product realization</td>
</tr>
<tr>
<td>8 Measurement, analysis and improvement</td>
</tr>
</tbody>
</table>

Table 2.2: ISO 9001:2000 structure in TickIT Guide 5.0

- Finally in section 2.5 on page 14, the reader can find the cost and benefits of the ISO 9001:2000 TickIT Guide 5.0.
Chapter 3

Agile Software Development

*Software development is a cooperative game of invention and creation.*
- Alistair Cockburn (Cockburn [2001], p. 29)

The goal of this chapter is to give to the reader an overview of what is agile development and which are its values and principles.

### 3.1 Introduction

Summarizing what the authors say in (Highsmith & Cockburn [2001]), in order to deal with continuous changes in today economy, due to Internet and dotcoms fast growing and contracting, it is necessary to change the way of developing software.

As Highsmith and Cockburn remark ‘the question today is not how to stop change early in a project but how to better handle inevitable changes throughout its life cycle’ (Highsmith & Cockburn [2001]).

Traditional methods are focus on anticipating and eliminate changes, but that caused that a huge amounts of companies on Internet business failed (Highsmith & Cockburn [2001]). Huge amount of documentation and follow a heavy process in order to detect errors and measure, it is not the answer for the fast changing and deliveries on today projects.

As is explained in (Highsmith & Cockburn [2001]) today companies need to survive using quick time-to-market products, in a manner to win a market place to their competitors, they need agility to be able to adapt and survive in this new changing environment.

As it mentioned before there is a need of new concepts, principles and methods, the first concept that should be defined is *agility* shortly as:

> ‘Agility is the ability to both create and respond to change in order to profit in a turbulent business environment’ [1]

---

[1] Emphasis from the author
There are several concepts inside this definition that should be clarified. The first one is *ability* that is a property that belongs to a person, with these skills people can create and respond to change. Here we have the second concept *create* that is means that the person or team should be active in order to create new types of products, and the last concept is *response*, people should be alert to respond to the fluctuations of the changing environment. The fourth concept is *change* the three previous concepts have to deal with it. Finally, the last one is *profit*, the aim of a company is to survive and have profit from its business, in order to do that, the four previous concepts are needed. For a more detailed definition see (Goldman et al. 1995, p. 42).

In a manner to deal with all the five concepts there is a need of an evolutionary development with high quality inside, that change the definition of software development, as is described in the next paragraph.

In order to deal with changing environments agile development rely a lot on people characterized in four key factors, as is described by Cockburn and Highsmith in (Cockburn & Highsmith 2001) to develop software in an agile way, and are: amicability, talent, skill, and communication.

A quote from the previous article is the following phrase, representing how important people is in agile development.

‘Agile development focuses on the talents and skills of individuals, molding the process to specific people and teams.’


In the same article (Cockburn & Highsmith 2001) they remark that agile development is best suited for exploratory projects where are involved complexity, high levels of change and extreme pressure.

These are the major strengths in agile development, but it should be good to discover where are the weaknesses of this type of development, for that Barry Boehm’s article about ‘Get Ready for Agile Methods, with Care’ (Boehm 2002) addresses these matters, where he analyzes several areas in software development as: developers, customers, requirements, size, etc. One of the issues to take in consideration according to what Barry Boehm explains in (Boehm 2002) is that agile development relies a lot on the tacit knowledge shared by the team, instead plan-driven approach relies on explicit knowledge as plans and architectures.

In (Beck & Boehm 2003), the authors discuss about agility and discipline reflected in the discipline that an agile methodology as XP need and from the perspective of comparing the so called agile with plan-driven development. They are living metaphorically in two different environment, agile development lives in highly changing environments and plan-driven development is living in environments with changes that occur not very often. The question is the following, can they be mixed? According to Barry Boehm the answer is yes, using an risk-driven development. Jim Highsmith, also address this question using another approach that he calls agile software development ecosystems, see (Highsmith 2002, c. 25,26).

Agile development respond to this new environment with short iterations where they can deliver to the customer a small part of a software product that respond
to the customer needs and expectations. Due to the description from previous paragraphs, the software development needs a new definition instead of the old one linked to processes and plans from \(\text{IEEE Std 610.12-1990}\). A good definition of software development in this new environment is the following:

‘Software development is a cooperative game of invention and creation.’
- Alistair Cockburn (Cockburn 2001, p. 29)

Then there are some concepts that should be clarified in the above definition as **cooperative game**. It is a cooperative game because there are people involved playing some roles as project managers, customers, developers, testers, etc. and they need to interact in order to win the game, as the same a rugby team or soccer team where there are some established roles and some rules in order to play and win the game. They need each other to be successful. The second point is about **invention** and **creation**, because software companies always are creating new products and innovating in their risky business.

MacCormack investigates some companies that are involved on Internet business as Microsoft and he realize that they were not working as traditional development describe it. They are working on a new form to develop software (MacCormack 2001).

Then it is necessary now to describe the agile development values and principles, in the next sections.

### 3.2 The Agile Manifesto in detail

In February 2001, seventeen leaders of this new way of developing software had a meeting in Salt Lake City, Utah, USA. They were developing new methodologies since the mid 90’s as Extreme Programming (XP)\(^2\) (Beck 1999\(a\)) \(^{\text{http://www.extremeprogramming.org}}\), \(\text{http://www.controlchaos.com}\), \(\text{http://www.mountaingoatsoftware.com/scrum/}\), \(\text{http://jeffsutherland.org/scrum/}\), Dynamic Software Development Method (DSDM)\(^6\) (Stapleton 2003), Crystal Methodologies\(^7\) (Cockburn 1998\, 2000\, 2001), \(\text{http://www.crystalmethodologies.org}\), Lean Development (LD) (Poppendieck & Poppendieck 2003), Adaptive Software Development (ASD) (Highsmith 2000), and Feature Driven Development (FDD) (Coad et al. 1999).\(^8\)

They created the Agile Alliance\(^8\) to support this new way of develop software, more methodologies are now created and this is continuing to grow. The core of the agile software development are the values presented in the Agile Manifesto\(^9\) (Beck et al. 2001) and other twelve principles that will be discussed in this section. In the statements in the following paragraphs more importance is given to the left part of the statement than the right part. The left part of the
statements is highlighted in bold text. Also some comments were added from
one author outside the agile development world to these values.

• ‘Individuals and interactions over processes and tools’.
  The first point regarding this statement is that focus on individuals in
  the team is better than focus on the individuals as resources in the pro-
  cess. The second point is that the interactions and community aspects
  between skilled people influence new solutions on the challenges that ap-
  pear in a project. In addition, this means that using an undocumented
  process with excellent communication is preferable to using highly doc-
  umented processes with bad communications between people (Cockburn
  2001, Abrahamsson et al. 2001). These two factors a.k.a individuals
  and interactions influence the success of the project (Kutschera & Schäfer
  2002).
  Comments
  Robert L. Glass comments in (Glass 2001, p. 13) about this value that
  agile proponents are right about the people factor, that influence software
  quality and productivity a lot. He cites as example the book Software
  Engineering Economics (Boehm 1981) where productivity is influenced in
  a high degree by personal/team capability. He also adds that it is so im-
  portant that even SEI created People Capability Model\(^{10}\) to complement
  CMM.

• ‘Working software over comprehensive documentation’.
  Working software is the principal product of the software development
  where the customer can evaluate if the systems has been built or not
  (Cockburn 2001). The software shall be tested on regular intervals then
  the code becomes simpler and easier to change in the future, and the doc-
  umentation should be down to barely sufficient (Cockburn 2001, Abra-
  hamsson et al. 2001). Huge amount of documentation do not necessarily
  mean that the problem has been well understood (Kutschera & Schäfer
  2002).
  Comments
  Here Robert L. Glass (Glass 2001, p. 13) is also with the agile movement,
  about working software is a main issue in software development, but he
  says that documentation is also necessary, in order to follow the mainte-
  nance phase. This is because documentation is also a part of the software
  product.

• ‘Customer collaboration over contract negotiation’.
  This stress the issue that between customer and supplier should be a col-
  laborative relationships to develop software, not the old us and them, it
  shall exist only our team. Good relationships between these two actors
  are necessary for an successful project and product (Cockburn 2001). Collab-
  oration emphasis over strict contracts is useful when the project involves
  small to medium size project, but when large projects are taken place,
  more defined contracts are needed (Abrahamsson et al. 2001). Moreover,

\(^{10}\)For more information about People Capability Model, see http://www.sei.cmu.edu/cmm-p/
collaboration is necessary for a well-understanding of the customer situation (Kutschera & Schäfer 2002).

**Comments**

In this matter Glass says (Glass 2001, p. 13) that there is a tie between both perspectives agile and traditional approach, because customer collaboration is very important to be a successful project, but there is a need of the contract as a framework to create this collaborative approach.

- **‘Responding to change over following a plan’**

  This last issue means that the team should adapt to follow fast changes in the development. Agile methodologies contain planning tasks and mechanisms with whom is possible to change the priorities during the development to match customer’s needs using short times to deliver working software (Cockburn 2001, Kutschera & Schäfer 2002). In the report developed by Pekka Abrahamasson and colleagues (Abrahamasson et al. 2001) is remarked that the team i.e. software developers and customers should have competence, good information and authorization to deals with the challenges described above.

  **Comments**

  In this case, there is also a tie between the agile community and the traditional one, according to the author of (Glass 2001, p. 14), because it depends on the project characteristics how much plan and how much response to change a team should use. For example, according to him greater the project, the greater the plan, and the smaller response to change; but from the other side, smaller projects need greater response to change and smaller plans.

### 3.3 The Principles in the Agile Manifesto

In this section will be explained the principles from the agile manifesto and also some comments will be added from one author not involved in the agile development about these principles.

- **‘Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.’**

  The aim of agile methodologies is to delivering software that satisfy customer’s needs and expectations proving that serves to business objectives (Fowler & Highsmith 2001, Cockburn 2001). Deliver regularly is needed for obtaining a continuous winning spirit in the team and receive fast feedback to change the priorities throughout the development. Regular deliveries interval should be negotiated with the customer to fit a project based rhythm (Cockburn 2001).

  **Comments**

  Robert L. Glass (Glass 2001, p. 14) says that is not with the agile development in this matter, because according to him the customer should be involved at the beginning and at the end of the project, the continuous involvement of the customer can cause to much noise in the design and development phase, but he also add as sooner the customer obtain feedback from what product is developing, the better is.
• ‘Welcome changing requirements, even late in development. Agile processes harness change for the customers competitive advantage.’

Agile methodologies have some techniques to deal with changes in every stages of the software development e.g. use of continuous and early delivery with iterative development. If the company can face to the change quickly then this turns into a competitive advantage in the market for the customer company and supplier company. They can win market place (Cockburn 2001). According to (Fowler & Highsmith 2001) is better embrace the change than try to prevent it.

Comments
In his article Robert L. Glass (Glass 2001, p. 14) is not with the agile development on this topic. He says that the sentence should be rephrased to something like ‘Be prepare to accept change...’, the more late the change are made in the development, more is the cost and rescheduling.

• ‘Deliver working software frequently, from a couple of weeks to a couple of months, with a preference for the shorter time scale.’

In this principle are defined the working interval to deliver working software. The use of three month to four month intervals is proposed by (Cockburn 2001). He takes as example Winifred project (Cockburn 1998) cited by (Cockburn 2001) where they work with two customer reviews inside the deliver intervals. However, he agree with (Fowler & Highsmith 2001) where smaller deliver interval can be used when the development team can follow the rhythm of changing request and fast feedback.

Comments
According to the author (Glass 2001, p. 15) this is related to the previous principle and again he is with the traditionalist in this matter, because according to him they are talking about small projects.

• ‘Business people and developers work together daily throughout the project.’

According to (Cockburn 2001) if the business people do not spend the enough time with developers the project can fail. If there is not a daily basis communication between them, the project can be damage. Agile supporters start to work with not well-defined requirements, that is why the business people cooperation is needed to fill this space (Fowler & Highsmith 2001).

Comments
According to Robert L. Glass (Glass 2001, p. 15) the problem here is that daily basis collaboration that in most of the project it is not possible to follow. Also he says that is better to call customer than business people, in order to have a broad concept rather than business.

• ‘Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.’

Individuals are key in the project, without them there is no project. Motivation can be linked with community, friendship, and work satisfaction
Therefore maximizing people factor have high importance in agile development. Trust between managers and the team is a necessary element to success in a specific project (Fowler & Highsmith 2001).

Comments

The author (Glass 2001, p. 15) in this case is with the agile development, motivated people is important, but also he wants to add that skills are also important as key factors. In relationship with environment he also considers it as a key factor as was described in Peopleware book (Demarco & Lister 1999) cited by him.

• ‘The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.’

The principal thing here is to communicate information it is not high amount of documentation, it is people understanding. However, the problem is not huge documentation against not documentation it is about how to balance the amount of documentation and knowledge transfer (Fowler & Highsmith 2001). In (Cockburn 2001) chapter 3 and 4 discuss about this issue.

Comments

Robert L. Glass (Glass 2001, p. 15) says here that it is with the agile development, because the richness and instantaneously of face-to-face conversation, it can be reproduced with other devices. He also says that it is needed that the team sit in the same room, in order to have fluently face-to-face conversations.

• ‘Working software is the primary measure of progress. Entrust on working software instead of documentation and plans. Agile methodologies stress the fact of early and evolutive development. Projects can be divided in small chunks in order to develop and test incrementally (Cockburn 2001). Using these techniques the customer can see the project process early in the development and obtain a feedback of what are the risk that project is facing (Fowler & Highsmith 2001).

Comments

The author of (Glass 2001, p. 15) is again with the agile development fellows, because for management working software it is key issue as they measure the progress of a software development as the software itself.

• ‘Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.’

This principle involves that the team should work regularly, not with continuous overtime, because it can affect the work progress due that people are more tired not only during overtime but also in regular work. This can cause that more defects are included in the development on a regular work day (Cockburn 2001).

In addition, as is expressed in (Fowler & Highsmith 2001) sustainable development is about working with the same interval of hours daily throughout the whole project and remain healthy and creative.
CHAPTER 3. AGILE SOFTWARE DEVELOPMENT

Comments
Also in this case the author of (Glass 2001, p. 15) support this idea from the agile development, because of human health it is good work 40 hours a week as XP recommend, and not with unhuman schedules.

• ‘Continuous attention to technical excellence and good design enhances agility.’

If the design is well performed with good patterns that result in well-defined design, then it is very easy to make changes on it currently and further in the development. Moreover, the design is tested and improved regularly for good comprehension on later delivery (Cockburn 2001). Design is performed throughout the whole development in each chunk of increment, therefore as is expressed in (Fowler & Highsmith 2001) ‘design quality is essential to maintaining agility’.

Comments
Again the author of (Glass 2001, p. 15) is with the agile development, because if the team has some people with great knowledge then it should be easier for them to create a good quality product using design improvements practices.

• ‘Simplicity the art of maximizing the amount of work not done is essential.’

Making simple design in a software project where changes be made easily it is more difficult, but better for maintenance and development than a design where changes cannot be made effectively or it takes a long time to make the changes, because it is easy to add something to the code (Glass 2001, p. 15) (Cockburn 2001, Fowler & Highsmith 2001).

Comments
Robert L. Glass (Glass 2001, p. 15) does not agree with this principle, because the interpretation that someone as XP developers obtain from it is to go straight to the code, and gives only some minutes to design. He says that is necessary make a good design first and then start coding it, if not the team should develop the code again and again as the design change, taking a lot of time from the development.

• ‘The best architectures, requirements, and design emerge from self-organizing teams.’

An issue is that there is not an agreement on the emergent stuff in these paragraph. Cockburn says in (Cockburn 2001) that these properties emerge from the self-organizing teams in small steps. Jim Highsmith and Martin Fowler in (Cockburn 2001, Fowler & Highsmith 2001) instead, say that all those properties emerge due to few rules and high interactions in the team, also they add that this is couple with incremental development. However, they agree that the architecture also should be adaptable on through the time.

Comments
The author of (Glass 2001, p. 15) has a tie with this point, because there is no supporting evidence that self-organizing teams are the best team or the opposite that a strong management gives better results.
‘At regulars intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.’

Agile methodologies are not issues that the team can adopt and copy easily. Fowler and Highsmith recommend to start with one part of the methodology first, moreover methodologies can be tailored for every specific project they are not universal for every project (Fowler & Highsmith 2001).

As Cockburn says ‘it is fitting to end where we began’ (Cockburn 2001), that means adapt, fit, and improve the methodology for your own project and through it.

Comments
Finally, according to Robert L. Glass (Glass 2001, p. 15), he supports this agile principle, because teams perform better with something that they believe and can adapt to their current development, than some issue that is imposed by upper management.

3.4 Summary

• In the introduction the reader can find why this agile development starts, what is agility, and a new definition of software development.

• After that, the reader can find a description and discussion about the values that are mentioned in the agile manifesto.

• Finally the reader can find a description and discussion from another actor outside agile development about its principles.
Chapter 4

Extreme Programming

Text:

XP is a lightweight methodology for small-to-medium sized teams developing software in the face of vague or rapidly changing environment.
- Kent Beck (Beck 1999b, p. XV)

The goal of this chapter is to perform an overview of the different Extreme Programming (XP) aspects as for example, its values, principles, practices, roles and activities.

4.1 Introduction


A more detailed definition of XP than what was defined by Kent Beck in (Beck 1999a, pg. XV) is given by Ron Jeffries in (Jeffries 2002).

‘Extreme Programming is a discipline of software development based on values of simplicity, communication, feedback, and courage. It works by bringing the whole team together in the presence of simple practices, with enough feedback to enable the team to see where they are and to tune the practices to their unique situation’.

Extreme programming was first used for the famous C3 or Chrysler Comprehensive Compensation project where Kent Beck and his team (or ‘The Best Team in The World’) (Hendrickson 2000a) were involved (Anderson et al. 1998, Beck 1999a) to develop a large-payroll system to Chrysler using Smalltalk as language of development and XP practices. However there is a lot of discussion
if it was a successful project as is described in some Wiki\textsuperscript{1} pages (Cunnigham 2004\textit{a}). XP is composed of a set of values, practices, principles and activities that will be described in the following sections of this chapter. In order to review where can be the problems in an XP project, see (Stephens & Rosenberg 2003).

### 4.2 XP values

The values that are behind the five principles of XP are \textit{simplicity}, \textit{communication}, \textit{feedback} and \textit{courage} (Beck 1999\textit{b}) that can be matched with the values in (Anderson et al. 1998) where \textit{testing} can be \textit{feedback} in the previous book and \textit{aggressiveness} is replaced by \textit{courage}. The concept of feedback is used because it covers more than only testing, and \textit{courage} instead of a more negative word as aggressiveness. All these four values are described below:

- **Simplicity**
  
  It makes software development simple. Programmers should care about what they are writing in the moment, not for the future. If later new features are needed that will be without to much effort, because the code is simple and easy to understand (Anderson et al. 1998, p. 25). Make complex things simple, the same as for example in martial arts where the apprentice practice simple movement everyday, to make complex movements in the future.

  The previous paragraph contradict the old programming way of thinking, that programmers should make the code general, in order to future support, but this premise turn the code more complex and difficult to understand, using XP premise of simplicity the project can go faster in this extremely fast world that we are living (Anderson et al. 1998, p. 25).

  Summarizing the simplicity value in one sentence is ‘do the simplest thing that possible work’ (Anderson et al. 1998, p. 25). It is possible compare Ken Beck’s metaphor of simplicity with gambling, because you bet on the simple things on the present than complex things for the future (Beck 1999\textit{b}, p. 30).

  **Comments**

  Matt Stephens says that the focus on simplicity in XP practice is about throwing away code, where the team is working in code that is good today, but it is necessary to through away in the future, because of changes in the requirements (Stephens 2001\textit{a}). According to him it is better to not bet and spend the required time in up-front design and then code what it was designed. The tricky issue it is to know when to stop making simple things (Stephens & Rosenberg 2003, p. 344, 345). For more information, see (Stephens 2001\textit{a}).

- **Communication**

  Communication is an important part of the software development, for instance if there is lack of communication between customer and developers,
manager and customer, manager and developers or between developer, the project could possible fail (Beck 1999a, p. 30).

The customers in XP should be committed with the development, they should be one of the members of the team. No more “them” and “us” culture (Anderson et al. 1998, p. 25). The software development team should act as a rugby team were everyone are working to win the match, XP team is working to be successful with the development. It should not act as the customer was an adversary and was not a part of the team.

In order to deal with what customer wants and what developers can do, the XP project based their relationships using four variables (Anderson et al. 1998, p. 25): scope, quality, resources, and time.

Scope is what the system do and what should be done in the future. Quality are the measures used to fulfill the customer needs and expectations, as for example, reliability, correctness, security, etc. Resources are people, technical equipment and the room where the XP will work. Time is the how long will be the project (Anderson et al. 1998, p. 25).

Communication can be related with simplicity, because if developer make more simple applications, they are more simple to communicate and understand by the customer.

Comments
There is nothing wrong with communication, but relate communication as a tool to replace requirements documentation it is not good thing as was explained in (Stephens 2001a).

• Feedback
Feedback from one point of view is about how users and developers see how the project or the application is going. As is mention in (Beck 1999b, p. 32) the team is more confident in what they are doing if they have an accurate feedback of the application

Developers make unit tests in order to know where are the bugs in the systems, to later fix them and improve the quality of the product. They obtain with unit tests a feedback of how the application is going (Beck 1999b, p. 32).

Customers make functional or acceptance testing where they can see if the application meets their needs and its behavior. They obtain feedback from the system. In addition, customers make use stories, developers at the same time make the estimations for each use story, therefore the customers obtain feedback of how long will take to develop a specific story (Beck 1999b, p. 32).

For the release is necessary that 100% of the use stories have been tested and pass the test (Anderson et al. 1998, p. 28).

From another point of view feedback it is not only related with testing, it is related from the continuous communication between customers and developers that are located in the same space, because they are always communicating between each other obtaining feedback from the other part.

Later developers can easily change on the system what the customer wants. This is because their code is simple through use of feedback from the customer (Beck 1999b, p. 32). This is how these three XP values of communication, simplicity and feedback are related.

Comments
According to (Stephens 2001a), customer negotiation takes place every time that there is a meeting with the customer, but according to him programmers are not the adequate people to negotiate with the customer. It is good to deliver the product early, but one or two weeks are very difficult to accomplish without the necessary experience. See his other article about ‘Fear of Non-Progress’ to go deeper in the subject (Stephens 2001f).

- **Courage**

These three values working together turns to courage (Anderson et al. 1998, p. 28), see below Fig. 4.1.

Kent Beck shows a story in (Beck 1999b, p. 33,34) where they detect some problems in the architecture of the system few weeks before the delivery of the system. They fixed the problem, but that turns into more problems, because they had rebuild 50% of the tests, but they did it. Later they began to redesign all the code that they have due to changes in the tests. That courage (Beck 1999b, p. 33,34) is build on well understanding of the system due to simplicity, feedback with customer and good channels of communications.

**Comments**

Again according to Matt Stephens’ article (Stephens 2001f), there is nothing against courage as a concept, but in XP they apply courage to throw away code and start to see what happen next day, that can be a problem of delay in the project, because the first time is one day, and then another; and suddenly the project is out of control. It should be good to write the right code at the first time.

**4.3 Core XP Principles**

The core XP principles are five rapid feedback, assume simplicity, incremental change, embracing change, and quality work according to (Beck 1999b c. 8). These principles will be described below:

- **Rapid feedback**

Rapid feedback can be divided in two parts as is mention in (Beck 1999b, p. 37, 38), feedback from the customer and developer point of view.

  From developers point of view with feedback they learn how produce code, design and tests with good quality, this occur quickly in a time frame of seconds or minutes (Beck 1999b, p. 38).

  In addition, customers learn how make a good contribution to develop the application in rapid time frame of days (Beck 1999b, p. 38).

  The core of this principle is ‘to get feedback, interpret it, and put what is learned back into the system as quickly as possible’ (Beck 1999b, p. 38).

- **Assume simplicity**

According to Kent Beck (Beck 1999b, p. 38), developers should make the
design and code very simple using XP practices as refactoring, testing and supplemented with good communication. Developers should make the code simple and trust that more complexity can be added later.

- **Incremental change**
  As is described in (Beck 1999b, p. 38) change is made using small steps through the development, and planning. The same approach should be taken for changes in the team and XP implementation in the company.

- **Embracing change**
  The team should try to change only the most problematic areas in the application and maintain the other parts (Beck 1999b, p. 38).

- **Quality work**
  Quality reflects that the team works is well-done using the others three variables as scope, cost, time (Beck 1999b, p. 38).

Supplementing to these principles there are others less important principles that will listed here but not described, see (Beck 1999b, c. 8) for more details.

- Teach learning
- Small initial investment
- Play to win
- Concrete experiments
- Open, honest communication
- Work with people's instincts, not against them
- Accepted responsibility
- Local adaption
- Travel light
- Honest measurement

### 4.4 XP Practices

Different authors named several practices as is presented in Table 4.1 on the next page.

Ward Cunningham’s Wiki website ([Cunningham](http://www.extremeprogramming.org/xmags/xpmag_whatisxp.htm)) is more up-to-date, and so his list will be used. Then according to that there are twelve core practices

---

2 Refactoring is the process of changing a software system in such a way that it does not alter its external behavior of the code yet improves its external structure. ([Fowler et al., 1999](#))


4 Kent Beck's article, *Embracing Change with Extreme Programming*


in XP that can be categorized in programming, team and processes practices as Ron Jeffries mentions in (Jeffries 2002) as follow and in Fig. 4.2, on the following page:

Programming practices: test driven-development, simple design, design improvement, pair programming.

Team practices: collective code ownership, continuous integration, sustainable pace, coding standard.

Processes practices: whole team, planning game, small releases.

- **Planning game**
  According to (Jeffries 2002) the planning game can be divided in two parts:
  - Release Planning.
    In this stage the customer defines what kind of features wants in the product and prioritize them, programmers make an estimation of each

<table>
<thead>
<tr>
<th>Kent Beck</th>
<th>Robert C. Martin</th>
<th>Ron Jeffries</th>
<th>Ward Cunningham</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Planning Game</td>
<td>The Planning Game + User Stories</td>
<td>The Planning Game</td>
<td>The Planning Game</td>
</tr>
<tr>
<td>Small releases</td>
<td>Short cycles</td>
<td>Small releases</td>
<td>Small releases</td>
</tr>
<tr>
<td>Metaphor</td>
<td>Metaphor</td>
<td>Metaphor</td>
<td>System Metaphor</td>
</tr>
<tr>
<td>Simple design</td>
<td>Simple design</td>
<td>Simple design</td>
<td>Test-Driven Development</td>
</tr>
<tr>
<td>Testing</td>
<td>Acceptance Tests + Test-Driven Development</td>
<td>Customer Tests + Test-Driven Development</td>
<td>Design Improvement</td>
</tr>
<tr>
<td>Refactoring</td>
<td>Refactoring</td>
<td>Pair programming</td>
<td>Pair programming</td>
</tr>
<tr>
<td>Pair programming</td>
<td>Pair programming</td>
<td>Collective ownership</td>
<td>Collective code ownership</td>
</tr>
<tr>
<td>Collective ownership</td>
<td>Continuous integration</td>
<td>Continuous integration</td>
<td>Continuous integration</td>
</tr>
<tr>
<td>Continuous integration</td>
<td>Sustainable pace</td>
<td>Sustainable pace</td>
<td>Sustainable pace</td>
</tr>
<tr>
<td>40-hour weeks</td>
<td>Customer Team Member</td>
<td>The Whole Team</td>
<td>The Whole Team</td>
</tr>
<tr>
<td>On-site costumer</td>
<td>Coding standards</td>
<td>Coding standards</td>
<td>Coding standards</td>
</tr>
<tr>
<td>Coding standards</td>
<td>Open workspace</td>
<td>Coding standards</td>
<td>Open workspace</td>
</tr>
<tr>
<td>Open workspace</td>
<td>Coding standards</td>
<td>Open workspace</td>
<td>Continuous integration</td>
</tr>
<tr>
<td>Just rules</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.1: Different XP practices
feature. Ron Jeffries in (Jeffries 2002) remarks that the initial estimations could not be so accurate, but then with continuous iterations and reviews, they turn more accurate i.e. priorities and estimations. When the priorities and estimations are added to each feature, a release plan for the project can be done.

Iteration Planning.

This when the customer and programmers meet together to deliver working software every two weeks. The level of detail is bigger than the release plan, the customer shows what of the features of the release plan he or she wants for the next two weeks, programmers divide the features in tasks and estimates their tasks.

The tasks for the customer side as is described in (Beck 1999b, p. 55) are to determine about: scope of the project, priority of the features, composition of releases, dates of releases.

The tasks for the programmer side (Beck 1999b, p. 55) are to determine about: estimations of the features, technical consequences, process, detailed scheduling.

For more information about the planning game, see (Martin 2002, Beck 1999b, c. 4, c. 15) or (Beck & Fowler 2000).

Comments
XP according to Matt Stephens (Stephens 2001b, p. 345, 346) has a good approach on this practice, in order to plan each iteration, time estimations, effort, etc. But, also he adds that the project should not lose the big picture of the whole development.

- **Small Releases**
  Kent Beck mentions in (Beck 1999b, p. 56) that all releases should be a small as possible, but with the maximum quantity of business features developed. The release should contain features 100% implemented not 50% or 70%. If it is not possible, more time should be added to the release cycle.
  Ron Jeffries in (Jeffries 2002) added that in each iteration should be shipped working software, that is means the application should be executed, tested and the requirements of the customer included, everything at 100%. Moreover frequency is important as well depending on which kind of software is delivered.
  Robert C. Martin in (Martin 2002, p. 16) explains that two kind of plans are used to develop software using XP, one of them is the iteration plan that have a time frame of few weeks where the software may be or not deliver in production. The other plan is called release plan, where the working software is delivered into production and its time frame is about two or three months.

**Comments**
In XP the recommended time for each iteration is one or two weeks according to Matt Stephens (Stephens 2001b, p. 346-348) experience a good size should be one month, where the team can develop all the necessary features. Small releases can help the team to get feedback early in the development from the customer. For more information about this topic see (Stephens 2001b, c. 11)

- **System Metaphor**
  Ron Jeffries (Jeffries 2002) defines metaphor as ‘a common vision of how the program works’. As Alistair Cockburn mention in (Cunmigham 2004d) the important issue here is that people share some common understanding from their past experiences.
  Kent Beck’s remarks in (Beck 1999b, p. 56) that a metaphor should ‘helps everyone on the project understand the basic elements and their relationships’. According to him a metaphor is similar to that other people call ‘architecture’, but with the adding that should put the XP Team to follow some way of cohesion.

**Comments**
The system metaphor should not replace the architecture in a software development as Matt Stephens (Stephens 2001b, p. 348), but from another point of view he adds that a system metaphor can help the team to understand the architecture.

- **Simple design**
  The design is keep it simple through the developments, using the developers test-driven development and refactoring. XP fits the design for the present system features ready for future changes in an incremental or iterative way (Jeffries 2002).
As a recommendation Robert C. Martin argue in (Martin 2002 p. 15) that XP design should begin without thinking in infrastructure, as e.g. database or middleware. The infrastructure is added when the development force XP team to incorporate it.

According to (Martin 2002 p. 16) a good design should follow the rules listed below:

‘Consider the simplest thing that possibly work.
Your are not going to need it(YAGNI).
Once and Only Once’.

Comments
The problem here with simple design is that people are thinking in a daily basis and are evolving the design, the code, and test all the team, that is time consuming. They usually takes like 10 to 30 minutes to make the up-front design. According to Matt Stephens in (Stephens 2001c) it is better to spend some time with the up-front design and then start to code and then evolve the design. More comments about this you can find in (Stephens 2001b c. 12).

• Design Improvement (formerly Refactoring)
Doing design improvement in an XP project is a practice where the programmer delete duplicate code. In addition, programmers should increase ‘cohesion’ and decrease ‘coupling’ as is recommended in Martin Fowler’s book (Fowler et al. 1999) as signal of good code design.

Kent Beck’s (Beck 1999b p. 58) recommend to use design improvement only when the code push to use it, e.g. duplicate code.

Martin Fowler (Fowler et al. 1999 p. xiv) define refactoring the former name for design improvement as ‘a process of changing a software system in such a way that it does not alter the system behavior of the code yet improves it internal structure’.

Refactoring should made when there is something wrong in the code or as Kent Beck says ‘code smells’, according to William Wake (Wake 2001, p. 27) there are some problems in the code when:

‘Classes that are too long.
Methods are too long.
Duplicate code.’

etc.

Robert C. Martin in (Martin 2002, p. 16), add that this design improvement should be done every hour or half hour, followed by testing what was done. For more in deep coverage about design improvement see (Martin 2002 c. 5) or Martin Fowler’s book (Fowler et al. 1999).

Comments
Matt Stephens in his article (Stephens 2001c) says that the effort that people should put in refactoring is a huge one, spending a lot of time rewriting the code every time, if the necessary time it is not put it in working with an up-front design, and then only refactor some parts of it. See (Martin 2002 c. 9) for more information about this subject.
• Test-driven development (programmers tests + customer tests) 
(formerly unit tests + acceptance tests)
Test-driven development can be divided in two, first programmers test 
and customers tests, but that have always the same idea of first test then 

code. These two approach will be described below:

Programmer Tests.
The idea is that programmers should create the tests first and then code. 
The first test should fail, because no code have been created, and then the 
programmers should create the code to pass the test, and then turn the 
cycle to add one more test following by the code.
Ron Jeffries in (Jeffries 2002) remarks that one of the benefits of extreme 
programming is that 100% of the code is tested.

Customer Tests.
Each user story that represent a feature in the XP development has as-
sociated an acceptance test (Martin 2002) that is determined by the XP 
customer and implemented by the team. A suggestion in (Martin 2002, 
p. 13) and (Jeffries 2002) is that tests should be automated, because in XP 
time is really precious. These automated tests are executed continuously.

The correctness of the systems is showed to the customer when all tests 
pass. Consequently the application is continually growing and evolving 
(Martin 2002, p. 13).

Another suggestion by Robert C. Martin in (Martin 2002, p. 13) is that 
developers or the QA department can implement their own testing tools 
for acceptance testing. For more information about customer tests, see 
(Martin 2002, c. 3). For information about test-driven development, see 

Comments about programmers tests
Programmers test or unit tests cannot be the requirements, the test are 
created to find bugs in the code. The whole process cannot rely on tests. 
What is happen when the tests are wrong and are testing incorrectly the 
functionality or design? Testing is good according to (Stephens 2001), 
but it should be related with the necessary requirement specification. For 
more information about his comments about unit testing, see (Stephens 
2001, c. 8).

Comments about customer tests
Customer tests cannot replace the requirements, as a Macro Excel of tests. 
Sometimes programmers make customer tests, but this is not specifically 
their role (Stephens 2001).

• Pair programming
Development is made pair of programmers, sitting in front on the same 
computer. Ron Jeffries (Jeffries 2002) argues that this is a really good 
practice, because the whole code is reviewed, designed, and tested by a 
fellow from the XP team. Moreover, if pairs are switched through the 
team knowledge is shared to everyone working in the XP team. Also, 
individuals skills are improved.
Robert C. Martin (Martin 2002, p. 13) remarks that pair should switch 
at least once per day. One programmer code and the other is observing 
the code looking for errors and improvements.
Laurie Williams in (Williams & Cockburn 2001) reports that there is
a 15% of increasing quality using pair programming, but also a 15% of overhead. For more information about pair programming, see (Williams & Kesler 2003).

Comments
According to Matt Stephens (Stephens 2001), programming is more an alone practice, where the programmer should work on their problem by themselves. The pair can interrupt the process of thinking. Also he adds that pair programming needs that people should be on the same level of experience to pair programming.

• **Collective code ownership**
The entire XP team is responsible for the developed code there is no personal responsibility for each part of the code or technology (Martin 2002, p. 14).

Martin mentions in (Martin 2002, p. 14) for example that programmers can work with the database one day, another day with servers, another act as a tester. XP team members should be multifunctional, they should work for the team.

Ron Jeffries argue in (Jeffries 2002) that one problem that XP can have with collective ownership is that people can not understand certain code that should change, but he mentions that XP resolve that problem using pair programming and using the tests that were implemented before, as a result each member of the team has knowledge of the whole project.

Comments
As Matt Stephens comments in (Stephens 2001b, p. 354,355) that if the team follow collective code ownership, there is a huge more importance of well-done standard that everyone can understand it.

• **Continuous integration**
The integration of the code is performed several times per day (Jeffries 2002). If the code is integrated once a week, that can be a problem because developers should investigate where is the problem in order to understand the causes of it, and in one week more code is developed then too much effort can be focus on it.

The pair should check that their changes do not affect another part of the system developed by another pair of programmers (Martin 2002, p. 14).

Kent Beck mentions in (Beck 1999b, p. 59) that one machine can be used only for integration issues for one pair of programmers, in addition, the pair should assure that all the tests pass.

Comments
According to Matt Stephens this is a good practice in XP, he only add that instead of continuous integration, should be frequent integration (Stephens 2001b, p. 354,355).

• **Sustainable pace (formerly 40-hour weeks)**
As is described in (Jeffries 2002) a project is a long term activity, then the XP can be developed a constant rate of velocity in a manner to have energy through all the project life cycle. In that way the Team can be alert to act quickly when some changes appear throughout the project.

Robert C. Martin (Martin 2002, p. 15) compare a software project with a marathon where the athlete is maximizing his energy thorough the whole
run waiting for the final sprint. The final week is the only one that the team can work overtime in XP project.

Comments
About this practice Matt Stephens (Stephens 2001b, p. 355) says that this is a good practice, because people in the team do not burn out. If it is possible to obtain it, do it if not sometimes people should work overtime to reach a deadline.

- The Whole Team (formerly On site Customer)
All the team members is in one place called open workspace, defined for Robert C. Martin in (Martin 2002, p. 15) as the large room where the XP team work that have cubicles in the periphery around a center that have a several tables configured for three to four computers with two chairs each where developers work using pair programming, in addition to that it can used a large white-board where the team posts the use stories, designs, task assignments and everything that the project need.
From another point of view the XP team (Jeffries 2002) is composed by one customer that represent the business perspective to define the requirements, another members of the team are the developers, testers that help the user to make the user tests, also there is the figure of the coach that enabling the management process of the Team, more details of the roles will be in the next subsection. Kent Beck in (Beck et al. 2001, p. 60,61) stress the importance of the customer as ‘someone who will really use the system when it is in production’.

Comments
Matt Stephens says in (Stephens 2001b) that on the first book about XP (Beck 1999b), the role of the customer on site was that the customer should be available every day of the project answering questions from people in the project. The problem is that usually customers do not have the time to getting involve in a daily basis with the team. Then XP evolve from one single business analyst as a proxy customer to a group of business analysts in Questioning Extreme Programming (McBreen 2002, p. xvi) cited by (Stephens 2001b) where the first choice was an error according to Ken Beck. If the reader wants to know about another comments to this practice, see (Stephens 2001b, c. 5).

- Coding standards
Using a coding standards helps the XP team to understand all the code that has been written as basis for the practice of collective ownership (Jeffries 2002).
In addition, Kent Beck mentions in (Beck 1999b, p. 61,62) that the standard should not be imposed to the team and the code should be written ‘Once and Only Once’ (Beck 1999b, p. 61,62) as a rule, duplicate code should be avoided.

Comments
Again according to Matt Stephens this is a good practice, where a documented code standard should be shared and understood by the whole team (Stephens 2001b, p. 356, 357).
4.5 XP Activities

XP is composed by the following four activities that are self-explanatory, see (Beck 1999b) for details:

- **Coding**
  XP focus is on coding, developers start almost immediately to code in a very simple way, trying to maintain the code as simple as possible. It supposes that the code should be clear to everybody in the project, in order that all the team can make changes on it.

- **Testing**
  Using with coding, testing is used in two forms as customer tests and programmers tests. According to XP developers should test first and then code. Confidence is obtained always testing your code.

- **Listening**
  XP needs a continuous feedback from the customer, developers should continuously listening and understand what the customer wants.

- **Designing**
  Design in XP is related to making the code design simple, and not to the design phase of the development. They use the refactoring approach to make the code simple, where for XP that symbolized good design.

4.6 XP Roles

There are some roles that an XP team should fulfil, these will be explained in the following paragraphs.

- **Programmers**
  They are the core of XP, without them there is no software developed. The XP programmer should practice design improvements, simple design, learn pair programming, test-driven development, make the estimations from the use stories, and determine what are the tasks that should be taken in order to develop each use story (Beck 1999b, p. 141,142).

  **Comments**
  In the article (Stephens 2001d) Matt Stephens’ says that programmers in XP are not only programmers, they should play different roles as testers, designers, and analysts. The question is, how many programmers can work in so many different roles?

- **Customer**
  It is the other 50% of the heart of XP, as Beck’s says ‘The programmer knows how to program. The customer knows what to program’ (Beck 1999b, p. 142,143).
  The Customer tasks are writing use stories, define the customer tests, determine the priorities for each use stories that should be explained to the team. A good customer is one who will use the system to be developed (Beck 1999b, p. 142-144).

  **Comments**
Matt Stephens in his article about the XP (Stephens 2001d) says that the daily involvement of the customer in the development, and also sometimes writing acceptance tests, can convert him in some kind of technician, forgetting the business scope of the product.

- **Tester**
  Tester is the person who teach the customer how to identify and create customer tests (i.e. acceptance tests). Tester executes the integration tests and make some graph, in a manner to show the XP team the results (Beck 1999b, p. 143).

- **Comments**
  Again as Matt Stephens remarks in his article (Stephens 2001d), the problem here is that programmers are working as testers. A tester needs another way of thinking a more destructive one, they can find bugs where programmers do not know. They test also in an application level, not only in a code level.

- **Tracker**
  He or she is who estimates the project velocity, using the feedback from the programmers asking and listening them about what they are doing in the current moment. He should be careful to do not interrupt the project too many times, Ward Cunningham mentions in (Cunnigham 2004b) that two times per week should be enough.

- **Coach**
  Kent Beck defines this role as who ‘is responsible for the project as a whole’ (Beck 1999b, p. 145). He or she enables that the project goes for the right path and keeps people working on the current features for the actual iteration. He has a good comprehension of XP and the project than the rest of the team (Beck 1999b, p. 145,146).

- **Consultant**
  Sometimes a consultant is needed in an XP project. His or her role is to teach the team the technical skills that are needed to solve the problem that they have (Beck 1999b, p. 146,147).

- **Big Boss**
  Big Boss is the person that trusts on the XP team that they do their tasks to complete a release. Continuously checking of the team performance is part of this role responsibility. In addition, if they want to change something the shall explain why that change is needed (Beck 1999b, p. 147,148).

### 4.7 Comments in general about XP

The XP practices reinforce each other and that requires high discipline in the team. Other comments and problems can be found in Matt Stephens and Doug Rosenbergs’ book *Extreme Programming Refactored: The Case Against XP* (Stephens & Rosenberg 2003).

- **High discipline needed.**
  XP is a highly discipline methodology, that is means that involves a high
level of team discipline to work with all XP practices. The team should use all of them or none at all, because those practices reinforce between each other, with the possible exception of Unit Testing (Stephens 2001b). Moreover, Stephens remarks that it is necessary that a good XP coach works with XP in a project, if not the team could not perform all the practices required by XP.

In addition, see Alistair Cockburn’s comments on a WikiPage8, where he compares XP with another high discipline methodology as Personal Software Process (PSP)9 created by Watt Humpeys at SEI.

• **No detailed written requirements**

  Requirements are written in a high level of abstraction using use stories10, the design is developed through the project, not big up-front design (Stephens 2001c).

• **No big up-front design**

  Only little time is consider to develop an up-front design. Huge changes in the architecture and design are developed throughout the development, for that constant refactoring is needed (Stephens 2001c).

• **Constant refactoring**

  Design is improved along the project, but this can cause an overtime working on it. Also some bugs can be introduced with the small changes on it if the team does not have unit tests for all the components (Stephens 2001c).

• **Unit Tests**

  Unit tests are fine they check the correctness of the code, but what happens when sometimes the design is not correct, unit tests do not check that. According to Stephens is needed a full time design expert that check this is needed (Stephens 2001c). In terms of XP this is done by Pair Programming.

• **Pair programming**

  According to Stephens 2001c there is a need of pair programming, because they do not have written the requirement specification, they can talk each other and with the customer that should be on site. There is a need that both programmers have the same experience and know each other in order to develop Pair Programming.

• **Customer on-site**

  There is a customer on site that should take business decisions, in order to deal with lack of requirement specification, but what is happen when customer on site is only a junior one? (Stephens 2001c).

---

8See [http://www.c2.com/cgi/wiki?HighDisciplineMethodology](http://www.c2.com/cgi/wiki?HighDisciplineMethodology)

9See [http://www.sei.cmu.edu/tsp/psp.html](http://www.sei.cmu.edu/tsp/psp.html)

10Use stories are notes used to described requirements with a high level of abstraction in XP (Cockburn 2000 p. 187).
4.8 Summary

- The introduction starts to define what is XP and also brings to the reader a little history of XP where started and which are the components of it.

- After that in section 4.2 on page 27 the reader can find a description of what are the values behind XP. See Table 4.2.

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPValue1</td>
<td>Simplicity</td>
</tr>
<tr>
<td>XPValue2</td>
<td>Communication</td>
</tr>
<tr>
<td>XPValue3</td>
<td>Feedback</td>
</tr>
<tr>
<td>XPValue4</td>
<td>Courage</td>
</tr>
</tbody>
</table>

Table 4.2: XP values

- Section 4.3 on page 29 are described and commented the principles behind XP. See Table 4.3.

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPPrinciple1</td>
<td>Rapid feedback</td>
</tr>
<tr>
<td>XPPrinciple2</td>
<td>Assume simplicity</td>
</tr>
<tr>
<td>XPPrinciple3</td>
<td>Incremental change</td>
</tr>
<tr>
<td>XPPrinciple4</td>
<td>Embracing change</td>
</tr>
<tr>
<td>XPPrinciple4</td>
<td>Quality work</td>
</tr>
</tbody>
</table>

Table 4.3: XP Principles

- The next section 4.4 on page 30 is about the twelve XP practices, where were added some comments about each practice. See Table 4.4 with the twelve XP Practices.

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPPractice1</td>
<td>Whole Team</td>
</tr>
<tr>
<td>XPPractice2</td>
<td>Simple Design</td>
</tr>
<tr>
<td>XPPractice3</td>
<td>Collective Code Ownership</td>
</tr>
<tr>
<td>XPPractice4</td>
<td>Coding Standards</td>
</tr>
<tr>
<td>XPPractice5</td>
<td>Design Improvement</td>
</tr>
<tr>
<td>XPPractice6</td>
<td>Planning Game</td>
</tr>
<tr>
<td>XPPractice7</td>
<td>Test-driven development</td>
</tr>
<tr>
<td>XPPractice8</td>
<td>System Metaphor</td>
</tr>
<tr>
<td>XPPractice9</td>
<td>Small Releases</td>
</tr>
<tr>
<td>XPPractice10</td>
<td>Pair Programming</td>
</tr>
<tr>
<td>XPPractice11</td>
<td>Continuous Integration</td>
</tr>
<tr>
<td>XPPractice12</td>
<td>Sustainable pace</td>
</tr>
</tbody>
</table>

Table 4.4: XP Practices

- The activities are described in section 4.5 on page 38. The following Table 4.5 shows these activities.

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
- In section 4.6 on page 38 a description of what are the roles involved in XP with some comments for some of them. A list of XP roles is showed in the following Table 4.6.

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPRole1</td>
<td>Programmers</td>
</tr>
<tr>
<td>XPRole2</td>
<td>Tester</td>
</tr>
<tr>
<td>XPRole3</td>
<td>Coach</td>
</tr>
<tr>
<td>XPRole4</td>
<td>Customer</td>
</tr>
<tr>
<td>XPRole5</td>
<td>Big Boss</td>
</tr>
<tr>
<td>XPRole6</td>
<td>Consultant</td>
</tr>
<tr>
<td>XPRole7</td>
<td>Tracker</td>
</tr>
</tbody>
</table>

Table 4.6: Roles used in XP

- Finally, in section 4.7 on page 39 some general problems of XP are described.
Chapter 5

Research strategies and techniques

An interview is a conversation, usually between two people. But it is a conversation where one person - the interviewer - is seeking responses for a particular purpose from the other person: the interviewee.
- Bill Gillham (Gillham 2000, p. 1)

The goal of this chapter is to present the research strategies and techniques used to obtain the desired results fulfilling the objectives of this thesis.

5.1 Introduction

According to (Creswell 2000, p. 18) there are three paradigms or approaches of empirical research, that are named as quantitative, qualitative or mixed research.

In this Master thesis a qualitative research was done in order to obtain a good understanding about the problem that is being investigated, also because the projects do not have a huge amount of statistical data in order develop a mixed research or quantitative research. In qualitative research the focus is to study the objects in their own environment. The researcher in this case try to understand the phenomenon and its causes using the explanations obtained from people (Denzin & Lincoln 2000, p. 3). Qualitative research use several interpretative practices that are usually interconnected to obtain a better understanding of the subject problem to study (Denzin & Lincoln 2000, p. 3), as for example: case study; personal experience; introspection; life story; interview; artifacts; etc. Each practice make the world visible from their own perspective that integrated together conform a big puzzle. Usually more than one practice is developed in one research. The questions are usually open-ended that gives more freedom to the researcher to dig into the problem giving to them more freedom.
Using the categorization that is developed in (Wohlin et al. 1999, c. 2) there are three principals research strategies: survey, experiment, and case study. According to (Yin 2002, p. 1) each research strategy differ on three items: the questions, the control that the researcher has over the object of investigation, and how contemporaneous or historical is or was the event.

A Case study was developed, because the event is investigated in two contemporaneous software projects where the researcher do not have control on them, and the investigator can use different techniques for data gathering.

In the next sections of this chapters the reader will know which strategy and techniques that were used to gathering the data.

5.2 Case Study

A case study definition can be developed in two part, quoted from (Yin 2002, p. 12):

1. A case study is an empirical inquiry that
   • investigates a contemporary phenomenon within its real-life context, especially when
   • the boundaries between the phenomenon and context are not clearly evident

2. The case study inquiry
   • copes with the technically distinctive situation in which there will be many more variables of interest than data point, as one result
   • relies on multiple sources of evidence, with data needing to converge in a triangulating fashion, and as another result
   • benefits from the prior development of theoretical propositions to guide data collection and analysis.

A case study has several stages according to (Yin 2002), each stage was fulfilled to develop this Msc. thesis:

• Design the case study(this chapter),
• Develop the case study(this chapter),
• Analyze the data gathering(chapters 6 on page 49, 7 on page 65, 8 on page 78),
• Develop conclusions and recommendations(chapters 9 on page 84).

The type of case study developed was an exploratory case study (Yin 2002), in order to answer the research questions described in the Introduction. It is mainly exploratory, because the phenomenon of the comparison of XP and TickIT 5.0 it has not been investigated before.

Another classification of case studies are single-case study and multiple-case study. In this case was selected a single-case study, because in order to perform a good multiple-case study the researcher need to perform between 6 to 10 case studies according to (Yin 2002, p. 46), moreover some replication characteristics
should be founded in each study, but in this case software projects are always different between each other.

The type of design selected using the recommendations founded in (Yin 2002, p. 40) was a single-case study design with embedded units of analysis, in this case two software development projects.

Finally, a single-case study will be developed using two embedded units of analysis (i.e. two academic software projects), after that chapter a comparison between both project will be held. Moreover, both projects were using XP as development methodology and were ISO 9001:2000 certificate using the TickIT Guide 5.0.

5.3 Data Collection

For data collection different approaches are used in order to create the case study chapter and one chapter about comparison between the two projects.

These techniques for data collection were interviews; documentation of one project (i.e. quality manual, quality policy, etc.) and the result of the audit of both projects; archive records as photos of how they work in an ordinary day in both project; and two hours of direct observation in both projects. Each technique of data gathering has its own pros and cons as the reader can see in (Yin 2002, p. 78).

5.3.1 The Interviews

As is defined in (Stewart & Jr Cash 1997, p. 1) interview is ‘an interactional communication process between two parties, at least one of whom has a predetertermined and serious purpose, and usually involves the asking and answering questions’.

In this thesis all the interviews were taped also notes were taken. Two groups-interviews were performed (i.e. face-to-face) where the interviewees were the project leaders of each project and the quality responsible. One more interview was developed to go more deep in some details that were not covered in the first interview with one of the project leaders. Finally, one more interview was performed to the TickIT advisor of the two projects. The face-to-face interviews were selected, because the cost to perform the interviews for the author was affordable, and brings to the researcher a possibility of depth coverage of the investigated situation. Pros and cons of face-to-face interview can be founded in the following book (Gillham 2000, c. 2), where the author mentions some characteristics that the researcher should take in consideration as the number of people to be interviewed, people accessibility, etc.

The interviews performed in this thesis cover all the stages in the so called proving interview as is mentioned in (Stewart & Jr Cash 1997, p. 108): determining the purpose of the interview (the interview in this case was used as a data gathering to develop this case study), research the topic (i.e. the topic was defined in this Msc. Thesis), structuring the interview (see the questionnaire section 5.3.2 on the following page in this chapter), selecting the interviewees

---

1 the whole documentation in one project was in Swedish and unfortunately this author do not have a high knowledge of it, only basic

2 Emphasis of the author
(see section 5.3.3 on the next page in this chapter), selecting the interviewers
(the author of this report), conducting the interview (see section 5.3.4 on the
following page in this chapter), preparing the story or report (i.e. this thesis).

All the structured is presented in the Appendix A where is detailed the ques-
tionnaire.

An interview has a structure that can be divided in three main parts as is
described in (Stewart & Jr Cash 1997, c. 3): opening, body and closing, that
should be taken in consideration.

Opening are the first minutes or seconds that the interviewer and the inter-
viewee spend to establish the first contact on the interview, that can be a great
influence for the rest of the interview. Some opening techniques are detailed in
(Stewart & Jr Cash 1997, c. 3). There are also additional non-verbal communi-
cation techniques, that goes from how the interviewee look, to how the rest of
the body is following the conversation.

The body is in essence the questionnaire to be developed and the topics to be
followed. This goes according to (Stewart & Jr Cash 1997, p. 63) from infor-
mal interviews where the questionnaire is prepared in a high level and formal
interviews where the questionnaire is highly detailed to perform the interview.
The recommendations of the authors of the book (Stewart & Jr Cash 1997) is
to have defined at least an interview guide, the guide in this case was the ques-
tionnaire presented in the Appendix A. After that an interview schedule should
be prepared.

Closing it is also important, because it is helps to establish a good relationship
between the interview and interviewee for next interviews or researches, and
also for the results of the interview. Functions, guidelines and techniques for
closing an interview are described in (Stewart & Jr Cash 1997, c. 3).

5.3.2 Questionnaire

As is mention in (Stewart & Jr Cash 1997, c. 4) questions can be divided in
three categories: (1) open and closed, (2) primary or secondary, (3) neutral or
leading.

The questionnaire was developed using questions that were open, primary and
neutral questions, because the researcher can obtain an in deep coverage of the
subject and gives a lot of freedom to the interviewee.

Closed questions were not used, because they are very restrictive in what
the research is looking for, and almost give the answers to the interview as is
mention in (Stewart & Jr Cash 1997, p. 81).

The primary questions used are mentioned in Appendix A but also some
secondary questions come out when the interview were developed to obtain
more information from the interviewee.

Leading questions were not used, because that will influence the result of this
research, then only neutral questions were developed to give the interviewee the
freedom to give the answer that he or she wants, for more information about
leading and neutral questions, see (Stewart & Jr Cash 1997, c. 4).

Questions were divided in three main issues, the customer company, the project
itself, the comparison about TickIT and XP, and what they think about in the
closing area as the reader can see in Appendix A.
5.3.3 Selecting the interviewees

The interviewees that were selected were the people with knowledge in the software development of these projects, the project leaders and the quality managers.

Their knowledge go through the different phase of the development, i.e. from planning to development. In addition to these, they have a little bit of more understanding about Extreme Programming than their fellows.

5.3.4 The Interview procedures

Here are described the interview procedures:

- Mainly the interviews were developed using face-to-face conversation, but also were used emails to cover some of the issue that in the interview were not clear.
- The approximate time for each face-to-face interview was one hour.
- The approximate time for the electronic interview was one week.
- Mainly, people involved in the face-to-face interview answered the same questions, people involved in the electronic-interview answered a subset of the questionnaire. However, additional questions were developed through the face-to-face interview to know more about the specific project.
- All the face-to-face interviews were developed using a tape-recorder.

5.4 Data verification and validation

The data obtained through the interviews were of high importance to develop the case study analysis. Also to perform comparisons and develop the conclusions of this report. In order, to verify and validate the data obtained the following approach was developed. **Face-to-Face meetings**

In order to verify and validate the data results and analysis face-to-face meetings were developed with the projects leaders, quality managers and one programmer of each project.

**Cross-checking**

As all the people was working in the same room or location it was not so difficult to talk with people and verify the data from another person involved in the interviews, with the quality audits results obtained from an auditor outside the development, with two hour of observation in both projects, records obtained from both projects, and documentation obtained from one project.

5.5 Summary

- In the introduction of this chapter the reader can find a description of what kind of research paradigm was used and why, which strategy was used and the techniques developed.
- In section 5.2 on page 44 is defined which type of case study was developed in order to fulfill the objectives of this Msc. thesis.
In section 5.3 on page 45 describes how the data were collected, as for example which kind of interviews were used, the questionnaire characteristics, which persons were interviewed, and the interview procedures.

Finally, section 5.4 on the previous page describes how the data were verify and validate.
Chapter 6

Case Study - Part 1 (The
W.H.A.T. Project)

The project is targeted on Hulsfred Festival, but it could be use in other
festival too, and other arrangements as sports arrangements.
- David Sandberg, Research Interview, April 15th, 2004.

The goal of this chapter is to investigate the use of XP in the
WHAT project that shall be ISO 9001:2000 certificate.

6.1 Introduction

W.H.A.T. is the acronym for (Wireless Hotspot of Advanced Technology), whose
product is ‘Festival Compis’. The project objective is to solve a problem that
Hultsfred Festival has.

The problem is on communication, because the expected attendance this year
is about 30,000 visitors and 5,000 officials, moreover the place in Hultsfred where
the Festival will be held is about two square kilometers.

In previous years the Festival has been using runners with notes, GSM-phones,
and DECT-phones in order to organize themselves, but they do not think that
this is enough to obtain the appropriate information that they need and work
efficiently with the GSM-network that is easily overload by the number of calls
that the visitors make in that small area.

The product ‘Festival Compis’ helps the officials to improve their communica-
tion and work as for example, make decisions, responsibilities, etc.

In addition to this product a website is developed whose name is ‘HF-Net’
for simplify administration issues. This website will be available to all the re-
sponsible of the Festival, before it start, to create and organize the responsible
schedules.

1Hultsfred Festival is the largest and longest Festival in Sweden for young peo-
ple. It takes three days that will be 17th, 18th and 19th of June this year
2004, see http://www.rockparty.se/content/english/ for more information in English or
http://www.rockparty.se/ in Swedish
This project is developed by students at BTH for their Bachelor examination course, whose name is Large Software Engineering Project.

The project itself takes 20 weeks from early January until early June 2004. Rockparty is a non-profit organization in charge of the festival which is responsible for the 5000 functionaries. Musiclink AB is a commercial company owned by Rockparty which usually offers technical services, education and media distribution. During the festival, Musiclink AB is responsible for IT, videoproductions and, this year, also for the system they have developed.

The project is developed by seventeen students of the third year study of software engineering program, two human-computer interaction students that helps with the design of graphic user interfaces (GUIs) and one economics student that helps with marketing research of how this product can be used in other festivals or other arrangements like sports arrangements.

The language used for development is C#, using Dot Net Platform and Visual Studio.

The methodology used in this project is XP with continuous deliver every. The project iterations are not of the same length, some times are of 1 and a half weeks, another time three weeks, but mostly two weeks. There has been six iterations in the project and are projected eight iterations to finish the project.

### 6.2 XP Practices in W.H.A.T. project

Here are examined the different practices of XP in the project, see below on Table 6.3.

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Grade</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPPractice1</td>
<td>Whole Team</td>
<td>-</td>
<td>There is no really customer at the open space room, a proxy customer is used that can call the real customer to obtain the requirements.</td>
</tr>
<tr>
<td>XPPractice2</td>
<td>Simple Design</td>
<td>++</td>
<td>The team work almost always with simple design, but sometimes they add some complexity in the code.</td>
</tr>
<tr>
<td>XPPractice3</td>
<td>Collective Code Ownership</td>
<td>+</td>
<td>For few parts of the code they have collective ownership, because not everybody is pair programming.</td>
</tr>
<tr>
<td>XPPractice4</td>
<td>Coding Standards</td>
<td>++</td>
<td>They share coding standards through the whole group.</td>
</tr>
<tr>
<td>XPPractice5</td>
<td>Design Improvement</td>
<td>++</td>
<td>They made design improvements once or more time per day.</td>
</tr>
<tr>
<td>XPPractice6</td>
<td>Planning Game</td>
<td>++</td>
<td>They work with the planning game in each iteration, the first day of each iteration.</td>
</tr>
</tbody>
</table>
Table 6.1: XP practices used in project W.H.A.T.

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Grade</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPPractice7</td>
<td>Test-driven development</td>
<td>-</td>
<td>One problem is that the real customer do not want to made acceptance test. The test manager with the customer responsible develop acceptance tests, that are not checked by the customer, because he does not understand it. Unit test is made for some parts of the system. Problems are on how to test GUIs and Web Applications for example, how do you test components related to user interfaces. The team develop their own tools, but it is still difficult to test everything.</td>
</tr>
<tr>
<td>XPPractice8</td>
<td>System Metaphor</td>
<td>-</td>
<td>They have a system metaphor. They have the knowledge of it, but they do not know how to use it and what benefit obtain from it. This can be because it is one of the most difficult practices to understand in XP.</td>
</tr>
<tr>
<td>XPPractice9</td>
<td>Small Releases</td>
<td>++</td>
<td>The releases are not of the same length, from 1.5 weeks to 2 weeks. But it works for what they want with the customer.</td>
</tr>
<tr>
<td>XPPractice10</td>
<td>Pair Programming</td>
<td>+</td>
<td>Three or four pair programming and seven that do not pair programmer in the project. It is not working by few people, because people have different skill levels, then it is difficult to pair programming, they should be on the same level to pair programming.</td>
</tr>
<tr>
<td>XPPractice11</td>
<td>Continuous Integration</td>
<td>++</td>
<td>They have two different places to store the code, a trunk and branch. Developers work in the branch, but when a developer think that his code is ready for release, the code is copied in the trunk, to be tested and integrated with the other code by the Test Group.</td>
</tr>
<tr>
<td>XPPractice12</td>
<td>Sustainable pace</td>
<td>++</td>
<td>The team use 40-hour weeks development, but when people is ill they should work a little more to fulfil the gap when they are not in the project.</td>
</tr>
</tbody>
</table>

2A grade is given for each XP Practice. + : means that the referred practice is partially fulfilled ++: means that the practice is fulfilled, and -: means that the practice is not fulfilled.
In the following page, two iterations in the W.H.A.T. project are showed in Fig. 6.1. In this picture the reader can see the tasks developed in the first iterations, and when they were completed.

6.3 XP activities in W.H.A.T project

In this section the activities related to XP in the W.H.A.T. project are examined, see below on Table 6.2.
### CHAPTER 6. CASE STUDY - PART 1 (THE W.H.A.T. PROJECT)

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Grade</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPActivity1</td>
<td>Coding</td>
<td>+</td>
<td>It is hard that fifteen people share an understanding of the code. In the first iterations everyone was working sharing an understanding of the whole code, but as the project evolved and the size of the code was increasing, collective ownership was not possible to do. There are some people that is specialist in one part of the code as they obtain a huge knowledge about that part. Others have a great understanding of several parts of the code, but none for the whole code. Usually people work with their own part of the code.</td>
</tr>
<tr>
<td>XPActivity2</td>
<td>Testing</td>
<td>-</td>
<td>At the first iterations they made unit test and acceptance tests, for the whole code. The customer was supposed to write the customer tests, but the customer it was not interested at all in writing acceptance tests. They started developing the acceptance tests by themselves, but as they realize that have not meaning, because the customer it was not interested they did not do it since the third iteration. About programmers tests they are making unit test for some part of the system, but from other side there are people from the Test Group of four people that are performing, code reviews, and integration tests. It is up to them to perform the testing, there is no more test driven development.</td>
</tr>
<tr>
<td>XPActivity3</td>
<td>Listening</td>
<td>++</td>
<td>They have a customer meeting at the end of each iteration, where they have feedback from them. They demonstrate the product and what have been done since last iteration, and obtain what kind of change the customer needs. Also they have almost frequent meetings where everyone can talk about the problems, the solutions founded and how the development can be improved. The system is checked almost daily by the customer, because it is online right now. A lot of feedback is received from the customer, according to them it is a really good customer.</td>
</tr>
</tbody>
</table>
### 6.4 XP Roles in W.H.A.T. project

When the project started the roles were *project manager, the coach, the tracker, test manager, the developers*. The roles used currently in the development, according to the information obtained from W.H.A.T project plan ([Sandberg & Falck] 2004), and complemented with the interview are:

- **Project Leader**
  The previous roles of project manager and coach are now in one person the project leader. For their understanding the role of project manager is the same as the big boss, but they do not have someone with that characteristic in the team, that is why the role of the project leader is really similar to the coach in XP.
  The job of the project leader is to plan the project and that everything goes in the same direction in a manner that the project be successful. He is responsible for external communication of the project, schedule and be in charge of the team meetings, and try to solve the problems that the team has.

---

3idem previous grade.
CHAPTER 6. CASE STUDY - PART 1 (THE W.H.A.T. PROJECT) 55

- **Tracker**
  His responsibility are the project metrics and the quality of the product and process. He must be sure that the metrics are accurate and used through the project iterations and in the progress report. The tracker evolved to be the Quality Manager at the last three iterations.

- **Customer Responsible**
  This is the proxy On-Site customer that should act as an XP customer due to the impossibility to have the real customer in the room with the team. He understand the customer needs and is in charge of the customer relationships. His responsibility is to contact the real customer, in order to answer developer questions about the product. Moreover, he also is responsible to develop and plan user stories and tasks.

- **Developer**
  The developer is responsible to develop the system make the estimations for his work, helps in the planning meeting and design the application.

- **GUI Responsible**
  He is a developer responsible for the GUIs development, and make sure that the layout of the system is consistent through the development.

- **The Non-Functional Group**
  The Non-Functional Group is in charge for non-functional requirements and how these can be achieved.

- **Test Group**
  When the project started there was only one person that worked as tester, right now this role was changed and now is developed for a group of people. The responsibility of this group is to test and find defects, in order to deliver a good quality product.

- **Dispatch group**
  The dispatch group is responsible of the application delivery. They assure that the product was developed, tested and integrated before presentations in each iteration.

- **Backup responsible**
  He is responsible of the automated backup of the system each week.

- **Configuration responsible**
  The whole team is responsible for configuration issues, the project leader will allocate specific developers to this task when its needed.

6.4.1 Metrics Program

There are two types of metrics in this project, first the productivity and progress metrics and second the test metrics, more details see \[\text{Falck & Berntsson-Svensson 2004}\].

- **Productivity and progress metrics**
  These metrics are related with the metrics used in XP. The first one is Task Point that reflects the hours used to develop certain task.
TP = Task point were setting to 8 perfect hours\(^4\) at the beginning of the the development, but it can change due to personal skills.

The second one is Individual Task Velocity, in a manner to measure how much work is developed for one individual, see below Fig. 6.2.

\[
ITV = \frac{TP}{H}
\]

Figure 6.2: Individual Task Velocity Metric in W.H.A.T. project. \(H\) = hours worked on own Tasks, \(TP\) = Total number of task points completed during previous iteration.

The third one is a user story point that is equivalent to 40 ideal hours. The fourth one are the hours that a developers should investigate to implement a certain user story. The fifth one is the project velocity that are the number of estimated user story points used to work in one iteration using the knowledge from the previous iterations.

- **Testing Metrics**
  Testing metrics were not developed until the sixth iteration, after the TickIT audit, they realize that there is a need for testing metrics to fulfill TickIT audit. This can be viewed as a lacking in XP, that do not mention any testing metric. XP is more focus on working software than in metrics for the software improvement. One metric that the author suggest to the W.H.A.T team was to have registered per day the number of tests that passed and the number of test that did not pass.

### 6.5 Comparison between TickIT Guide 5.0 and XP, W.H.A.T. Project

The following Table was developed using the information obtained from the audit results and the interviews to W.H.A.T. Project members, in addition cross-checking was performed top verify and validate the results with project members and TickIT auditor.

<table>
<thead>
<tr>
<th>TickIT Guide Clause</th>
<th>Grade</th>
<th>Comments about agile development and W.H.A.T. project</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 Quality Management System</td>
<td></td>
<td>Some of the general requirements where developed by the team, but not by XP itself as the reader will see in the following rows of this table.</td>
</tr>
<tr>
<td>4.1 General Requirements</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.2 Documentation Requirements</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TickIT Guide Clause</td>
<td>Grade</td>
<td>Comments about agile development and W.H.A.T. project</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------</td>
<td>--------------------------------------------------------</td>
</tr>
<tr>
<td>4.2.1 General</td>
<td></td>
<td>Some documents were established and maintained, according to the standard (i.e. quality manual, quality policy, etc.). But, in the case of the design documentation were in a very high level. Use stories were considered as requirement specification. Records of the meetings were not available, and test records were established only in the last part of the project.</td>
</tr>
<tr>
<td>4.2.2 Quality Manual</td>
<td></td>
<td>The quality manual was established and maintained.</td>
</tr>
<tr>
<td>4.2.3 Control of documents</td>
<td></td>
<td>Documentation as quality documents and use stories were reviewed by another person than the author. Versions are developed by each document.</td>
</tr>
<tr>
<td>4.2.4 Control of records</td>
<td></td>
<td>Meetings were not taped. Test records were developed after TickIT audit. XP does not say anything about control of records.</td>
</tr>
</tbody>
</table>

5 Management responsibility

5.1 Management Commitment |       | The project leader was committed with the project working on progress reports; coding; developing the quality policies and manual; and meeting with the customer. He has high commitment on the project. |
5.2 Customer Focus         |       | Customer satisfaction was obtained from the XP view, showing a part of the product in each iteration. |
5.3 Quality Policy         |       | The quality policy was developed by the project leader and understood by the rest of team. |
5.4 Planning

5.4.1 Quality objectives   |       | In this case the nonfunctional requirement established in the quality objectives were measure only late in the development. XP has problems to deal with nonfunctional requirements, the same as the team. |
<table>
<thead>
<tr>
<th>TickIT Guide Clause</th>
<th>Grade</th>
<th>Comments about agile development and W.H.A.T. project</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.4.2 Quality management system planning</td>
<td></td>
<td>The item about planning for the whole organization in a strategic view is not applicable here, because they do not belong to a company, they are in a university project. About planning at a functional level, there are specified the quality plans, quality objectives, code standards, etc. From the other side product documents were poorly defined, for XP the documentation are code and tests that have been done through the development.</td>
</tr>
<tr>
<td>5.5 Responsibility, authority and communication</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.5.1 Responsibility and authority</td>
<td></td>
<td>Instead of having the whole responsibility in one person, the whole team is responsible for what is going to delivered. There are roles and responsibilities defined as the reader can see before in this chapter. Responsibility for the production phase was not defined, because this is an academic project.</td>
</tr>
<tr>
<td>5.5.2 Management representative</td>
<td></td>
<td>It was created the role of Quality Group that is responsible for well-function of the QMS. This was added to XP, it is not a part of it.</td>
</tr>
<tr>
<td>5.5.3 Internal communication</td>
<td></td>
<td>A high point in XP is that there is a huge level of communication, between the members of the team, because they work in the same room.</td>
</tr>
<tr>
<td>5.6 Management Review</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.6.1 General</td>
<td></td>
<td>They documented the reviews using the collaboration of the test group. This was defined only in the last part of the project (i.e. the last three iterations). It was difficult to known the status of the report (i.e. preliminary or accepted) For TickIT Guide 5.0 management reviews are keys and necessaries to implement a QMS.</td>
</tr>
<tr>
<td>5.6.2 Review input</td>
<td></td>
<td>id. clause 5.6.1</td>
</tr>
<tr>
<td>5.6.3 Review output</td>
<td></td>
<td>id. clause 5.6.1</td>
</tr>
<tr>
<td>6 Resource Management</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.1 Provision of resources</td>
<td></td>
<td>Not applicable, because people were students in their large project work.</td>
</tr>
<tr>
<td>6.2 Human Resources</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TickIT Guide Clause</td>
<td>Grade</td>
<td>Comments about agile development and W.H.A.T. project</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------</td>
<td>------------------------------------------------------</td>
</tr>
<tr>
<td>6.2.1 General</td>
<td></td>
<td>People were working as a testing group, to ensure the QMS and customer satisfaction. In a pure XP project the team will work as a whole for improving the quality of the product, and customer satisfaction.</td>
</tr>
<tr>
<td>6.2.2 Competence, awareness and training</td>
<td></td>
<td>Through the project, some of the more experience people were dedicated to it. Training was developed by the courses that they take at BTH, before developing this system.</td>
</tr>
<tr>
<td>6.3 Infrastructure</td>
<td></td>
<td>Not applicable, because the infrastructure was provided by BTH.</td>
</tr>
<tr>
<td>6.4 Work Environment</td>
<td></td>
<td>Not applicable, because the working environment was provided by BTH.</td>
</tr>
<tr>
<td>7 Product realization</td>
<td></td>
<td>As a difference, between XP and traditional way of doing planning is that XP is focus on the details of one or two iterations, it is not possible to plan beyond that.</td>
</tr>
<tr>
<td>7.1 Planning of product realization</td>
<td></td>
<td>As was specified before, requirements were specified in a high level, and how to prioritize them were not defined. The team prioritize them in an ad-hoc manner during planning meetings. Moreover, in TickIT Guide 5.0 is specified that requirements for delivery and post delivery should be taken in consideration. In this case, as it is an academic project without the delivery and post-delivery part, this is not applicable.</td>
</tr>
<tr>
<td>7.2 Customer-related processes</td>
<td></td>
<td>Review techniques are not under control. They were defined, but not concretely applied in the project. The reviewing activity is performed by the Testing Group.</td>
</tr>
<tr>
<td>7.2.1 Determination of requirements related to the product</td>
<td></td>
<td>One of the points that XP remarks, it is customer feedback using the customer on site. Instead of that they have a proxy customer that should contact the real customer and be or she perform well in contacting almost daily with the customer. Web access is provided to the customer to test the website created.</td>
</tr>
<tr>
<td>7.2.2 Review of requirements related product</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.2.3 Customer Communication</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.3 Design and development</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Chapter 6. Case Study - Part 1 (The W.H.A.T. Project)

#### 7.3.1 Design and development planning

As was mentioned before, planning is developed almost for the current iteration. Design it was planned in a high level of abstraction. This is because, XP highly propose refactoring the code to make a good design.

#### 7.3.2 Design and development inputs

Functional requirements were specified in the use stories, but nonfunctional requirements (e.g. reliability requirements, portability requirements, usability requirements, etc.) were developed only since the last two iterations. XP do not handle nonfunctional requirements.

#### 7.3.3 Design and development outputs

A detailed design specification will be provided using a reverse engineering tool. Architectural specifications were developed in a high level of abstraction. Tests were not recorded at the beginning of the development, but later at the end of the project, they recorded tests for some use stories, not for every in the last two iterations. Executable code and source code will be delivered to the customer.

#### 7.3.4 Design and development review

Most of the code and the architectural design were reviewed by the Test Group. Detailed design were not reviewed, because it was in a continuous process of design improvement of the code. The code was reviewed by the Test Group.

#### 7.3.5 Design and development verification

The verification process were developed by the test group. Traceability from requirements to code was not defined. That is a problem in XP projects. Most of the unit test were specified, but not the integration and systems tests.

#### 7.3.6 Design and development validation

Validation was performed by meetings with the customer, where they show a prototype of the product that was evolving through the time.
<table>
<thead>
<tr>
<th>TickIT Guide Clause</th>
<th>Grade</th>
<th>Comments about agile development and W.H.A.T. project</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.3.7 Control of design and development changes</td>
<td></td>
<td>Control of design was not performed, because they will deliver a detail design when the project will be completed. The design is defined only in a high level. They manage the change in the code using a configuration management tool. There was not defined a configuration management activity.</td>
</tr>
<tr>
<td>7.4 Purchasing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.4.1 Purchasing process</td>
<td></td>
<td>Not applicable, they use products that are licensed by the university.</td>
</tr>
<tr>
<td>7.4.2 Purchasing information</td>
<td></td>
<td>id. clause 7.4.1.</td>
</tr>
<tr>
<td>7.4.3 Verification of purchased product</td>
<td></td>
<td>id. clause 7.4.1.</td>
</tr>
<tr>
<td>7.5 Production and service provision</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.5.1 Control of production and service provision</td>
<td></td>
<td>Not applicable, because in the contract they still do not defined the responsibility of the product.</td>
</tr>
<tr>
<td>7.5.2 Validation of processes for production and service provision</td>
<td></td>
<td>Not applicable, see 7.5.1</td>
</tr>
<tr>
<td>7.5.3 Identification and traceability</td>
<td></td>
<td>As was detailed before, traceability was not developed from the requirements to the code. XP does not explain about that. But for example, it is possible to trace a requirement as a use story and who was the owner.</td>
</tr>
<tr>
<td>7.5.4 Customer property</td>
<td></td>
<td>Not applicable, see 7.5.1</td>
</tr>
<tr>
<td>7.5.5 Preservation of product</td>
<td></td>
<td>Not applicable, see 7.5.1</td>
</tr>
<tr>
<td>7.6 Control of monitoring and measuring devices</td>
<td></td>
<td>Devices as PDAs where tested and monitored through the development for functional requirements, nonfunctional requirements were not tested until the latest iteration. The real device to experiment with was given to the team in the latest iteration, in order, to test nonfunctional requirements.</td>
</tr>
<tr>
<td>8 Measurement, analysis and improvement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TickIT Guide Clause</td>
<td>Grade</td>
<td>Comments about agile development and W.H.A.T. project</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>8.1 General</td>
<td></td>
<td>Again, measurement of nonfunctional requirements was not developed until the last part of the project, due to lack detailed requirement specification in XP. The functional requirements were tested without any problem using the information in the use stories. Problems with measurements, analysis and monitoring of the product are detailed in the next clauses.</td>
</tr>
<tr>
<td>8.2 Monitoring and measurement</td>
<td></td>
<td>Customer satisfaction was developed through customer meetings at the end of each iteration where feedback can be obtained directly face-to-face with the customer, but it is not measure in any form.</td>
</tr>
<tr>
<td>8.2.1 Customer satisfaction</td>
<td></td>
<td>Not applicable, There were only people of the project working on it, there was not assigned an internal audit, only reviewers as the Test Group.</td>
</tr>
<tr>
<td>8.2.2 Internal Audit</td>
<td></td>
<td>Measurements of the process were developed using the task points and velocity of each developer, they do not have other measures to monitor the process.</td>
</tr>
<tr>
<td>8.2.3 Monitoring and measurement of processes</td>
<td></td>
<td>As was detailed before, they do not monitor nonfunctional requirements, functional yes. In the last iteration, they perform the measurement of nonfunctional requirements.</td>
</tr>
<tr>
<td>8.2.4 Monitoring and measurement of the product</td>
<td></td>
<td>They do not make any control over the part of the product that was not non compliant with the customer requirements, they throw it away as XP says.</td>
</tr>
<tr>
<td>8.3 Control of nonconforming product</td>
<td></td>
<td>The improvements were done without the analysis of data, only due to someone investigation in one topic.</td>
</tr>
<tr>
<td>8.4 Analysis of data</td>
<td></td>
<td>Continual improvement was held on morning meetings, with the customer meetings at the end of each iteration, and in the agile meetings at the end of each iteration. The results of this meetings are not stored, problems with the corrective and preventive actions are described in clauses 8.5.2, 8.5.3.</td>
</tr>
</tbody>
</table>
6.6 Summary

- In the introduction is described the W.H.A.T. project and how was developed in 20 weeks.

- In section 6.2 on page 50 are analyzed the XP practices in the W.H.A.T. project. In addition, in the following Table 6.4 XP practices codes developed in Table 6.1 will be used to summarize the XP practices results.

<table>
<thead>
<tr>
<th>Grade</th>
<th>Practices</th>
<th>Number</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>++</td>
<td>XPPractice2, XPPractice4, XPPractice5, XPPractice6, XPPractice9, XPPractice11, XPPractice12</td>
<td>7</td>
<td>58.3%</td>
</tr>
<tr>
<td>+</td>
<td>XPPractice3, XPPractice10</td>
<td>2</td>
<td>16.7%</td>
</tr>
<tr>
<td>-</td>
<td>XPPractice1, XPPractice7, XPPractice8</td>
<td>3</td>
<td>25.0%</td>
</tr>
<tr>
<td>Total</td>
<td>of XP practices in W.H.A.T. Project</td>
<td>12</td>
<td>100%</td>
</tr>
</tbody>
</table>

Table 6.4: Summary of XP Practices results in W.H.A.T. project

- Section 6.3 on page 52 analyze the XP activities developed in the W.H.A.T. project. In addition, in the following Table 6.5 XP activities codes developed in Table 6.2 will be used to summarize the XP activities results.

<table>
<thead>
<tr>
<th>Grade</th>
<th>Activities</th>
<th>Number</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>++</td>
<td>XPActivity3, XPActivity4</td>
<td>2</td>
<td>50.0%</td>
</tr>
</tbody>
</table>

Table 6.5: Summary of XP Activities results in W.H.A.T. project
• In the next section 6.4 on page 54 the roles in W.H.A.T project are analyzed.

• Finally, in section 6.5 on page 56 the TickIT clauses are related to the XP practices in W.H.A.T. project. A summary of these results is developed in the following Table 6.6.

<table>
<thead>
<tr>
<th>Grade</th>
<th>Activities</th>
<th>Number</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>XPActivity1</td>
<td>1</td>
<td>25.0%</td>
</tr>
<tr>
<td>-</td>
<td>XPActivity2</td>
<td>1</td>
<td>25.0%</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>12</td>
<td>100%</td>
</tr>
</tbody>
</table>

Table 6.5: Summary of XP Activities results in W.H.A.T. project

<table>
<thead>
<tr>
<th>Color</th>
<th>Clauses</th>
<th>Number</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>4.2.4, 5.4.1, 5.6.1, 5.6.2, 5.6.3, 7.2.1, 7.3.2, 7.3.7, 7.5.3, 8.1, 8.2.1, 8.2.3, 8.3, 8.4, 8.5.1, 8.5.2, 8.5.3</td>
<td>17</td>
<td>33.5%</td>
</tr>
<tr>
<td>Yellow</td>
<td>4.1, 4.2.1, 5.4.2, 7.1, 7.2.2, 7.3.1, 7.3.3, 7.3.4, 7.3.5, 7.6, 8.2.4</td>
<td>11</td>
<td>21.5%</td>
</tr>
<tr>
<td>Green</td>
<td>4.2.2, 4.2.3, 5.1, 5.2, 5.3, 5.5.1, 5.5.2, 5.5.3, 6.2.1, 6.2.2, 7.2.3, 7.3.6</td>
<td>12</td>
<td>23.5%</td>
</tr>
<tr>
<td>Brown</td>
<td>6.1, 6.3, 6.4, 7.4.1, 7.4.2, 7.4.3, 7.5.1, 7.5.2, 7.5.4, 7.5.5, 8.2.2</td>
<td>11</td>
<td>21.5%</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>51</td>
<td>100%</td>
</tr>
</tbody>
</table>

Table 6.6: Comparison summary of TickIT clauses with XP, W.H.A.T. project
Chapter 7

Case Study 2 - (The WAIS Project)

The project is about making logistics between two different companies that using different business software how can they interact and communicate between each other.

The goal of this chapter is to investigate the use of XP in the WAIS project that shall be ISO 9001:2000 certificate.

7.1 Introduction

The customer company has education and make software development as applications for their customers, the name of the company is EC\(^1\), they have a three offices and approximately 100 employees in the South of Sweden. The contact is with the main office in Kristianstad.

The project is about making logistic between two different companies that have different software for their business. The team should solve problems of communication and interaction between different software. People in the company can still use their own programs that send the information to the product developed by the team involved in the WAIS project that after that send the information to the customer companies of EC.

This project is developed by students at BTH for their Bachelor examination course, whose name is Large Software Engineering Project.

The AIS project has been developed for two and half year and they are the last part of the project in charge of the software development. They worked in the pre-study for the huge project that has been developed.

The project was developed using 15 people, but they add that at the starting point they were 20 people. For requirements, they use use stories where one use story is usually one requirement.

The tools used in the project are XPlanner for the planning and report time

\(^{1}\)The website is [http://www.ec.se](http://www.ec.se)
in the current iteration. It is specially designed for XP, you have the use stories and time build in.

The language used for development is Java and they are using IntelligentJ IDEA, for server and client pages they are using JBox and JBOXEE, but before they were using Tomcat and JSP. For testing they used JUNIT and for configuration management they use CVS. They have an explicit requirement of using only free software.

For a better management they divide the team in two groups of 7 people, where each one has a group leader. The project was a huge one to be managed by only one people.

The contract was made really late in the development, it says that they should deliver an architecture, documentation and related issues. They are very flexible in which kind of functionality should be or not delivered.

They do not have a maintenance phase, they will deliver their product and then it should be maintained by people in EC. Their goal is to deliver a prototype to the customer that should be in charge to extended it and create the final product.

### 7.2 XP practices used in the WAIS project

Here are examined the different practices of XP in the project, see below on Table 7.1.

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Grade</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPPractice1</td>
<td>Whole Team</td>
<td>-</td>
<td>They do not have the customer on site, but from the other hand it is possible to contact him almost daily.</td>
</tr>
<tr>
<td>XPPractice2</td>
<td>Simple Design</td>
<td>++</td>
<td>They always code with the simple design as a mandatory rule. All the design has a big refactoring session every Monday to keep it simple. They use the knowledge obtained from the UML course to make the design simple.</td>
</tr>
<tr>
<td>XPPractice3</td>
<td>Collective Code Ownership</td>
<td>++</td>
<td>The system metaphor and the classes involved in the architecture with the CVS tool for configuration management help them to have collective code ownerships in each group.</td>
</tr>
<tr>
<td>XPPractice4</td>
<td>Coding Standards</td>
<td>++</td>
<td>They use code standards described in the document about Coding Standard, but was developed in Swedish.</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Grade</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPPractice5</td>
<td>Design Improvement</td>
<td>++</td>
<td>They have an upfront-design with the architecture, but they have some problems with the customer, and therefore they have to change the architecture, for example, now they are on their third architecture from the scratch. Now they start with an upfront-design and then they start making design improvements on it.</td>
</tr>
<tr>
<td>XPPractice6</td>
<td>Planning Game</td>
<td>-</td>
<td>At the beginning they were working with the planning game. They had to leave this XP practice, because of the size of the project. The project manager have meetings with the groups leaders and they told the programmers what to do, contradicting what the planning game says that everyone in the group should be involved in the planning game. It is more linked to the traditional way of a project leader to develop software were the project leader is in control of the project and not the whole team. Prioritization is made in ad-hoc manner, due to the knowledge that the group leaders know about their programmers.</td>
</tr>
<tr>
<td>XPPractice7</td>
<td>Test-driven development</td>
<td>+</td>
<td>They have acceptance tests, but only manually not automatically as is explained in the books of XP. They make integration tests, right now they have two computers only for integration. Unit tests are developed by the programmers.</td>
</tr>
<tr>
<td>XPPractice8</td>
<td>System Metaphor</td>
<td>++</td>
<td>They have a system metaphor and it is everywhere in the room, walls and tables. They share the knowledge of the metaphor that help them to have a good understanding of the system.</td>
</tr>
<tr>
<td>XPPractice9</td>
<td>Small Releases</td>
<td>++</td>
<td>They have small releases of two weeks except for one where they have a problem with the customer that turns the iteration into three weeks. The total number of iteration will be seven and right now they are on their sixth iteration.</td>
</tr>
</tbody>
</table>
### Table 7.1: XP practices used in WAIS project

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Grade</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPPractice10</td>
<td>Pair Programming</td>
<td>++</td>
<td>They always use pair programming through the whole development. That was confirmed by the author in one observation day. They are using pair programming only when they are programming, not when they are working on something else, as write a report or do some research. They also add that it is very difficult to find a programmer with your same skills, because if that it is not the case, the problem is if they have different skills one of them does not do anything. According to their experience with Pair Programming, the code has a better quality than you make it alone.</td>
</tr>
<tr>
<td>XPPractice11</td>
<td>Continuous Integration</td>
<td>++</td>
<td>They have huge problems at the beginning, but right now they separate the code in different modules, and as soon as they finish to test and develop the code they integrate the parts and start to test them.</td>
</tr>
<tr>
<td>XPPractice12</td>
<td>Sustainable pace</td>
<td>++</td>
<td>They have been working 40 hours per week with flexible turns of eight hours, i.e. one turn can be from eight to five with one hour break or from nine to six with one hour break.</td>
</tr>
</tbody>
</table>

The next Fig. 7.1 shows part of the WAIS team using pair programming XP practice.
7.3 XP activities used in WAIS project

In this section the activities related to XP in the WAIS project are examined, see below on Table 7.2.

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Grade</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPActivity1</td>
<td>Coding</td>
<td>++</td>
<td>They always work making the code simple in manner that everyone can understand and can make change on them. There are established code reviews to maintain it simplicity.</td>
</tr>
<tr>
<td>XPActivity2</td>
<td>Testing</td>
<td>+</td>
<td>The programmers made unit test using the concept of test-driven development. The acceptance tests are made manually, not automatically. The integration test is made on the machines available for that use.</td>
</tr>
<tr>
<td>XPActivity3</td>
<td>Listening</td>
<td>+</td>
<td>The knowledge and understanding of the project remains on the two group leaders. The project manager has a high level picture of what is going on in the project. The decision in the project are made by the two group leaders and the project manager, but not for the whole team.</td>
</tr>
<tr>
<td>XPActivity4</td>
<td>Designing</td>
<td>++</td>
<td>They use simple design for every part of the code. The knowledge of the architecture is shared for everyone. They make huge design improvements every week and the code is integrated once per iteration.</td>
</tr>
</tbody>
</table>

Table 7.2: XP activities used in WAIS project

7.4 XP Roles in WAIS project

In this section will be described the XP roles used in the WAIS project, some of them were merged and others were added to fulfill ISO 9001:2000 - TickIT Guide 5.0. The project started with all the XP roles, but as the development was evolved they made the necessary changes.

- **Project Leader**
  It is responsible for the contact with the customer, making progress report. He has a high level of understanding of the project, but he does not know in detail about each functionality that has been developed.

- **Group Leaders**
  group-leaders are responsible for the quality of the code of their respective

---

2 A grade is given for each XP Practice. + : means that the referred practice it is partially fulfilled. ++: means that the practice is fulfilled, and -: means that the practice it is not fulfilled.

3 idem previous grade.
groups, and for assign the tasks that come out from the use stories. They merge in the figure of the group leaders the roles of the tracker and coach.

- **Programmers and testers**
  They have programmers that act also as a testers, using XP practice of Pair programming.

- **Customer on site**
  They do not have the customer on site, but they can use MSN, or ICQ to contact the customer almost daily or webcams to ask questions to the customer.

### 7.5 Comparison between TickIT Guide 5.0 and XP in WAIS Project

The following Table it was developed using the information obtained from the audit results and the interviews to WAIS Project members, in addition cross-checking was performed top verify and validate the results with project members and TickIT auditor.

<table>
<thead>
<tr>
<th>TickIT Guide Clause</th>
<th>Grade</th>
<th>Comments about agile development and WAIS project</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 Quality Management System</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.1 General Requirements</td>
<td></td>
<td>Some of the general requirements where developed by the team, but not by XP itself as the reader will see in the following rows of this table.</td>
</tr>
<tr>
<td>4.2 Documentation Requirements</td>
<td></td>
<td>Some documents were established and maintained, according to the standard(i.e. quality manual, quality policy, etc.). But, in the case of the design documentation were in a very high level. Requirement specification was not valid, how to link between use stories and requirements specification. Records of the meetings were not available, and test records were not established for non-functional requirements.</td>
</tr>
<tr>
<td>4.2.1 General</td>
<td></td>
<td>The quality manual was established and maintained.</td>
</tr>
<tr>
<td>4.2.2 Quality Manual</td>
<td></td>
<td>As was specified before, requirement documentation were considered the use stories, because in XP there is no requirement specification, but they were in a high level of abstraction. The design documentation were also in a high level, this is because in an XP project there is a lot refactoring and not a lot of up-front design.</td>
</tr>
<tr>
<td>4.2.3 Control of documents</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### CHAPTER 7. CASE STUDY 2 - (THE WAIS PROJECT)

<table>
<thead>
<tr>
<th>TickIT Guide Clause</th>
<th>Grade</th>
<th>Comments about agile development and WAIS project</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2.4 Control of records</td>
<td></td>
<td>Meetings were not taped. Test error records were not statistically developed. XP does not say anything about control of records.</td>
</tr>
<tr>
<td>5 Management responsibility</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.1 Management Commitment</td>
<td></td>
<td>The project was divided in two projects the group leaders were committed to the developing of the product, as the same as the project leader, with the project working on progress reports; coding; developing the quality policies and manual; and meeting with the customer.</td>
</tr>
<tr>
<td>5.2 Customer Focus</td>
<td></td>
<td>Customer satisfaction was obtained from the XP view, delivering through iterations.</td>
</tr>
<tr>
<td>5.3 Quality Policy</td>
<td></td>
<td>The quality policy was developed by the project leader and groups leaders and understood by the rest of team.</td>
</tr>
<tr>
<td>5.4 Planning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.4.1 Quality objectives</td>
<td></td>
<td>There is no quality objectives defined. XP has problems to deal with nonfunctional requirements, the same as the team.</td>
</tr>
<tr>
<td>5.4.2 Quality management system planning</td>
<td></td>
<td>The item about planning for the whole organization in a strategic view it is not applicable here, because they do not belong to a company, they are in a university project. About planning at a functional level, they specified the quality plans, code standards, etc. For XP the documentation are code and tests that have been done through the development.</td>
</tr>
<tr>
<td>5.5 Responsibility, authority and communication</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.5.1 Responsibility and authority</td>
<td></td>
<td>In this case as they use a a more traditional way of management, where the project leader and groups leaders are taking the responsibility for the project. They divide in ad hoc manner the tasks and responsibilities through the team. Some of the responsibilities as the tracker were not clear for the person that was developed that function.</td>
</tr>
</tbody>
</table>
### TickIT Guide Clause

<table>
<thead>
<tr>
<th>Clause</th>
<th>Grade</th>
<th>Comments about agile development and WAIS project</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.5.2 Management representative</td>
<td>5.5</td>
<td>The do not have a management representative to contact the customer, everyone can contact him and obtain information from him. Also there is a quality responsible, but the main responsibility for quality it is the team.</td>
</tr>
<tr>
<td>5.5.3 Internal communication</td>
<td>5.4</td>
<td>A high point in XP is that there is a huge level of communication, between the members of the team, because they work in the same room.</td>
</tr>
<tr>
<td>5.6 Management Review</td>
<td>5.5</td>
<td>They have not a good system for management review. XP is based on daily reviews due to pair programming, but they do not documented it. They do not know what to do with them.</td>
</tr>
<tr>
<td>5.6.2 Review input</td>
<td>5.5</td>
<td>id. clause 5.6.1, moreover requirements specification was not longer updated. Use stories were reviewed only for the persons involved on it.</td>
</tr>
<tr>
<td>5.6.3 Review output</td>
<td>5.5</td>
<td>id. clause 5.6.1</td>
</tr>
<tr>
<td>6 Resource Management</td>
<td>5.5</td>
<td>Not applicable, because people were students in their large project work.</td>
</tr>
<tr>
<td>6.2 Human Resources</td>
<td>5.5</td>
<td>All the team is practicing test driven development. Then they are all in charge of the testing part. Through the project, some of the more experience people were dedicated to it. The courses that they took at BTH, before starting this large projects can be consider as training.</td>
</tr>
<tr>
<td>6.3 Infrastructure</td>
<td>5.5</td>
<td>Not applicable, because the infrastructure was provided by BTH.</td>
</tr>
<tr>
<td>6.4 Work Environment</td>
<td>5.5</td>
<td>Not applicable, because the working environment was provided by BTH.</td>
</tr>
<tr>
<td>7 Product realization</td>
<td>5.5</td>
<td>As a difference, between XP and traditional way of doing planning is that XP is focus on the details of one or two iterations, it is not possible to plan beyond that. Planning for each iteration was developed throughout the planning game. Records for controlling the process were not developed.</td>
</tr>
<tr>
<td>7.1 Planning of product realization</td>
<td>5.5</td>
<td></td>
</tr>
<tr>
<td>7.2 Customer-related processes</td>
<td>5.5</td>
<td></td>
</tr>
</tbody>
</table>
### 7.2.1 Determination of requirements related to the product

As was specified before, requirements were specified in a high level, and how to prioritize them were not defined. The team prioritize them in an ad-hoc manner during planning meetings. In addition to this TickIT Guide 5.0 talk about the requirements for delivery and post-delivery. These kind of requirements are not applicable in this academic project.

### 7.2.2 Review of requirements related product

Review techniques are not under control. Reviews were done by people that are involved in the task not by the other group. The reviewing activity is viewed in XP as pair programming. The requirement specification it is not up-to-date.

### 7.2.3 Customer Communication

One of the points that XP remarks it is customer feedback using the customer on site. They do not have the customer on site everyone can talk with him. Feedback is constant. The customer feedback it was not so god, because it always say yes to everything that they say. However, they have prototype demonstrations to the customer. Reports on the progress of the development were not defined.

### 7.3 Design and development

#### 7.3.1 Design and development planning

As was mention before, planning is developed almost for the current iteration. Design it was planned in a high level of abstraction, there is no in detail design that should be maintained. This is because, XP proposes refactoring the code to make a good design. Milestones for the development were not developed, because of lacking in management in XP. In these milestones, should be for the approval of customer requirements document, approval for architectural design, etc.
### 7.3.2 Design and development inputs
Functional requirements were specified in the use stories, but nonfunctional requirements were developed only since the last two iterations. XP do not handle nonfunctional requirements. Guides for production were not developed, because this project does not contemplate the installation and production phase.

### 7.3.3 Design and development outputs
The detailed design specification will be provided using a reverse engineering tool. Architectural specifications were developed in a high level of abstraction. Test results were not recorded. Executable code and source code will be delivered to the customer.

### 7.3.4 Design and development review
Most of the code and the architectural design were reviewed by the Test Group. Detailed design were not reviewed, because it was in a continuous process of refactoring of the code.

### 7.3.5 Design and development verification
The verification process were developed by the Test Group in charge of it. Traceability from requirements to code was not well defined. Peer reviews were developed by the Test Group.

### 7.3.6 Design and development validation
Validation was performed by meetings with the customer, where they show a prototype of the product that was evolving through the time.

### 7.3.7 Control of design and development changes
Control of design was not performed, because they will deliver a detail design when the project ends. The design is defined only in a high level. They manage the change in the code using a configuration management tool. There was not defined and established a configuration management activity, needed by TickIT Guide 5.0.

### 7.4 Purchasing

<table>
<thead>
<tr>
<th>7.4.1 Purchasing process</th>
<th>Not applicable, they use products that are free of charge.</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.4.2 Purchasing information</td>
<td>id. clause 7.4.1.</td>
</tr>
<tr>
<td>7.4.3 Verification of purchased product</td>
<td>id. clause 7.4.1.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TickIT Guide Clause</th>
<th>Grade</th>
<th>Comments about agile development and WAIS project</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.3.2 Design and development inputs</td>
<td>Red</td>
<td>Functional requirements were specified in the use stories, but nonfunctional requirements were developed only since the last two iterations. XP do not handle nonfunctional requirements. Guides for production were not developed, because this project does not contemplate the installation and production phase.</td>
</tr>
<tr>
<td>7.3.3 Design and development outputs</td>
<td>Yellow</td>
<td>The detailed design specification will be provided using a reverse engineering tool. Architectural specifications were developed in a high level of abstraction. Test results were not recorded. Executable code and source code will be delivered to the customer.</td>
</tr>
<tr>
<td>7.3.4 Design and development review</td>
<td>Yellow</td>
<td>Most of the code and the architectural design were reviewed by the Test Group. Detailed design were not reviewed, because it was in a continuous process of refactoring of the code.</td>
</tr>
<tr>
<td>7.3.5 Design and development verification</td>
<td>Yellow</td>
<td>The verification process were developed by the Test Group in charge of it. Traceability from requirements to code was not well defined. Peer reviews were developed by the Test Group.</td>
</tr>
<tr>
<td>7.3.6 Design and development validation</td>
<td>Green</td>
<td>Validation was performed by meetings with the customer, where they show a prototype of the product that was evolving through the time.</td>
</tr>
<tr>
<td>7.3.7 Control of design and development changes</td>
<td>Red</td>
<td>Control of design was not performed, because they will deliver a detail design when the project ends. The design is defined only in a high level. They manage the change in the code using a configuration management tool. There was not defined and established a configuration management activity, needed by TickIT Guide 5.0.</td>
</tr>
</tbody>
</table>
### TickIT Guide Clause

<table>
<thead>
<tr>
<th>Clause</th>
<th>Grade</th>
<th>Comments about agile development and WAIS project</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.5.1 Control of production and service provision</td>
<td></td>
<td>Not applicable, because in the contract they still do not define the responsibility of the owner of the product.</td>
</tr>
<tr>
<td>7.5.2 Validation of processes for production and service provision</td>
<td></td>
<td>Not applicable, see 7.5.1</td>
</tr>
<tr>
<td>7.5.3 Identification and traceability</td>
<td></td>
<td>As was detailed before, traceability was not developed from the requirements to the code. XP does not explain about that. But for example, it is possible to trace a requirement as a use story and who was the owner.</td>
</tr>
<tr>
<td>7.5.4 Customer property</td>
<td></td>
<td>Not applicable, see 7.5.1</td>
</tr>
<tr>
<td>7.5.5 Preservation of product</td>
<td></td>
<td>Not applicable, see 7.5.1</td>
</tr>
<tr>
<td>7.6 Control of monitoring and measuring devices</td>
<td></td>
<td>Not applicable in this case, they do not have any special device to test.</td>
</tr>
<tr>
<td>8 Measurement, analysis and improvement</td>
<td></td>
<td>Again, measurement of non-functional requirements was not developed until the last part of the project, due to lack specification in XP. The functional requirements were tested without any problem using the information in the use stories. They have the problems with the process and product analysis and measure as the reader will see in the next clauses. Problems with measurements are due to lack of measurements in XP and lack of experience in the team.</td>
</tr>
<tr>
<td>8.1 General</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.2 Monitoring and measurement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.2.1 Customer satisfaction</td>
<td></td>
<td>Customer satisfaction was developed through customer meetings at the end of each iteration where feedback can be obtained directly face-to-face with the customer. However, there is no analysis of non-conformities in the product.</td>
</tr>
<tr>
<td>8.2.2 Internal Audit</td>
<td></td>
<td>Not applicable, There were only people of the project working on it, there was not assigned an internal audit, the reviewers were the same programmers.</td>
</tr>
<tr>
<td>8.2.3 Monitoring and measurement of processes</td>
<td></td>
<td>Measurements of the process were developed using the velocity of every developer using XPlanner. They do not monitor and document the measurement process.</td>
</tr>
</tbody>
</table>
## 8.2.4 Monitoring and measurement of the product

As was detailed before, they do not monitor nonfunctional requirements, functional yes. In the last iteration, they perform the measurement of nonfunctional requirements.

## 8.3 Control of nonconforming product

They do not make any control over the part of the product that was not non compliant with the customer requirements, they throw it away as XP says.

## 8.4 Analysis of data

The improvements were done without the analysis of data, only due to someone investigation in one topic.

## 8.5 Improvement

### 8.5.1 Continual Improvement

Continual improvement was held on Monday meetings and with the customer meetings at the end of each iteration. No analysis or documentation of the improvements have been held. Problems with corrective and preventive actions for improvements are shown in clauses 8.5.2 and 8.5.3.

### 8.5.2 Corrective Action

The team perform corrective actions with the results that they come out after Monday meetings, with daily meetings at the beginning, and meetings with the customer. There is no reviews of the corrective action that have been taken. For example, they do not store quantitative errors logs. Errors are written in a diary, that can be used after as a qualitative error report.

### 8.5.3 Preventive Action

They take preventive actions with the information obtained in agile meetings, Monday meetings, and meetings with the customer. There is no reviews of the preventive action that have been taken. The results have not been stored.

<table>
<thead>
<tr>
<th>TickIT Guide Clause</th>
<th>Grade</th>
<th>Comments about agile development and WAIS project</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.2.4 Monitoring and measurement of the product</td>
<td></td>
<td>As was detailed before, they do not monitor nonfunctional requirements, functional yes. In the last iteration, they perform the measurement of nonfunctional requirements.</td>
</tr>
<tr>
<td>8.3 Control of nonconforming product</td>
<td></td>
<td>They do not make any control over the part of the product that was not non compliant with the customer requirements, they throw it away as XP says.</td>
</tr>
<tr>
<td>8.4 Analysis of data</td>
<td></td>
<td>The improvements were done without the analysis of data, only due to someone investigation in one topic.</td>
</tr>
<tr>
<td>8.5 Improvement</td>
<td></td>
<td>Continual improvement was held on Monday meetings and with the customer meetings at the end of each iteration. No analysis or documentation of the improvements have been held. Problems with corrective and preventive actions for improvements are shown in clauses 8.5.2 and 8.5.3.</td>
</tr>
<tr>
<td>8.5.1 Continual Improvement</td>
<td></td>
<td>The team perform corrective actions with the results that they come out after Monday meetings, with daily meetings at the beginning, and meetings with the customer. There is no reviews of the corrective action that have been taken. For example, they do not store quantitative errors logs. Errors are written in a diary, that can be used after as a qualitative error report.</td>
</tr>
<tr>
<td>8.5.2 Corrective Action</td>
<td></td>
<td>They take preventive actions with the information obtained in agile meetings, Monday meetings, and meetings with the customer. There is no reviews of the preventive action that have been taken. The results have not been stored.</td>
</tr>
</tbody>
</table>

Table 7.3: TickIT clauses compared with XP, WAIS project

### 7.6 Summary

- In the introduction is described the WAIS project and how was developed in 20 weeks.
- In section 7.2 on page 66 are analyzed the XP practices in the WAIS project. In addition, in the following Table 7.3 XP practices codes develop-
CHAPTER 7. CASE STUDY 2 - (THE WAIS PROJECT)

oped in Table 7.1 will be used to summarize the XP practices results.

<table>
<thead>
<tr>
<th>Grade</th>
<th>Practices</th>
<th>Number</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>++</td>
<td>XPPractice2, XPPractice3, XPPractice4, XPPractice5, XPPractice8, XPPractice9, XPPractice10, XPPractice11, XPPractice12</td>
<td>9</td>
<td>75.0%</td>
</tr>
<tr>
<td>+</td>
<td>XPPractice7</td>
<td>1</td>
<td>8.3%</td>
</tr>
<tr>
<td>-</td>
<td>XPPractice1, XPPractice6</td>
<td>2</td>
<td>16.7%</td>
</tr>
<tr>
<td>Total of XP Practices in WAIS Project</td>
<td>12</td>
<td>100%</td>
<td></td>
</tr>
</tbody>
</table>

Table 7.4: Summary of XP Practices results in WAIS project

- Section 7.3 on page 69 analyze the XP activities developed in the WAIS project. In addition, in the following Table 7.5 XP activities codes developed in Table 7.2 will be used to summarize the XP activities results.

<table>
<thead>
<tr>
<th>Grade</th>
<th>Activities</th>
<th>Number</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>++</td>
<td>XPActivity1, XPActivity4</td>
<td>2</td>
<td>50.0%</td>
</tr>
<tr>
<td>+</td>
<td>XPActivity2, XPActivity3</td>
<td>2</td>
<td>50.0%</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0.0%</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>12</td>
<td>100%</td>
</tr>
</tbody>
</table>

Table 7.5: Summary of XP Activities results in WAIS project

- In the next section 7.4 on page 69 the roles in WAIS project are analyzed.
- Finally, in section 7.5 on page 70 the TickIT clauses are related to the XP practices in WAIS project. A summary of these results is developed in the following Table 6.6 on page 64.

<table>
<thead>
<tr>
<th>Color</th>
<th>Clauses</th>
<th>Number</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>5.4.1, 5.6.1, 5.6.2, 5.6.3, 7.2.1, 7.2.2, 7.3.2, 7.3.7, 7.5.3, 7.3.8, 7.8.1, 8.2.1, 8.2.3, 8.3, 8.4, 8.5.1, 8.5.2, 8.5.3</td>
<td>17</td>
<td>33.4%</td>
</tr>
<tr>
<td>Yellow</td>
<td>4.1, 4.2.1, 4.2.3, 4.2.4, 5.4.2, 5.5.1, 7.1, 7.2.3, 7.3.1, 7.3.4, 7.3.5, 8.2.4</td>
<td>12</td>
<td>23.5%</td>
</tr>
<tr>
<td>Green</td>
<td>4.2.2, 5.1, 5.2, 5.3, 5.5.2, 5.5.3, 6.2.1, 6.2.2, 7.3.3, 7.3.6</td>
<td>10</td>
<td>19.6%</td>
</tr>
<tr>
<td>Gray</td>
<td>6.1, 6.3, 6.4, 7.4.1, 7.4.2, 7.4.3, 7.5.1, 7.5.2, 7.5.4, 7.5.5, 7.6, 8.2.2</td>
<td>12</td>
<td>23.5%</td>
</tr>
<tr>
<td>Total of clauses</td>
<td></td>
<td>51</td>
<td>100%</td>
</tr>
</tbody>
</table>

Table 7.6: Results of comparing TickIT clauses with XP, WAIS project
Chapter 8

Case Study Analysis and Results

*Discipline is the foundation for any successful endeavor... The strength and comfort that come from discipline support the endeavor when things are difficult... Discipline creates well-organized memories, history and experience.*

*Agility is the counterpart of discipline... Agility applies memory and history to adjust to new environments, react and adapt, take advantage of unexpected opportunities, and update the experience base for the future.*

- Barry Boehm and Richard Turner ([Boehm & Turner](#), p. 1,2)

The goal of this chapter is to make a comparison between the two case studies analyzed in this report.

8.1 Introduction

This chapter presents the comparisons from the analysis performed in chapter 6 and 7 of two academic software projects, from the point of view of XP in section 8.2 and from TickIT 5.0 Guide point of view in section 8.3 and finally detailing the strengths and weaknesses of XP to fulfill TickIT Guide 5.0 from these two projects, see section 8.4 on page 80.

8.2 Comparing XP in both projects

In this section, XP practices will be compared first in both projects and then will follow with a comparison of XP activities.
8.2.1 Comparing XP practices in both projects

In the following Table 8.1, the reader can see a comparison of the XP practices in both projects.

<table>
<thead>
<tr>
<th>Grade</th>
<th>W.H.A.T.</th>
<th>WAIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>++</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>+</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>-</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Total of XP practices in both projects</td>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>

Table 8.1: Comparison of XP Practices results in both projects

One can see from previous Table 8.1 that one of the projects perform a little bit better in working with XP practices. However, if the reader compare both projects (see Table 6.1 and Table 7.1), fail in the whole team practice, because they do not have the customer on site to perform acceptance tests, and because of that they create a proxy customer in one project (i.e. W.H.A.T project) and in the other (i.e. WAIS project) everyone of the project could ask the customer. Moreover, they perform acceptance tests manually, not automatically as XP says to do it.

Major differences between the projects were that in one project they use test-driven development idea (i.e. WAIS project) and in the other only they perform the tests after the code was created (i.e. W.H.A.T. project). However, if the reader see the planning game practice the situation is viceversa.

Another difference is that in one project everyone pair programming (i.e. WAIS project), instead in the other only some part of the team developed that practice (i.e. W.H.A.T. project). If the reader see the practice of collective code ownership in one project were lacking of do it as the entire team, because of some of them do not pair programming (i.e. W.H.A.T project). Finally, the last difference between both projects is that in one project they understood the practice of System Metaphor (i.e. WAIS project) and in the other not (i.e. W.H.A.T. project).

8.2.2 Comparing XP activities in both projects

Here the XP activities will be compared in both projects, see Table 8.2.

<table>
<thead>
<tr>
<th>Grade</th>
<th>W.H.A.T.</th>
<th>WAIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>++</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>+</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Total of XP activities in both projects</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

Table 8.2: Comparison of XP activities results in both projects

The activity that is not fulfilled in one of the projects is testing (i.e. W.H.A.T. project), because in XP terms the testing activity should be taken in consideration before written the code, not after.
In both projects the coding activity is not entirely fulfilled, because in both projects do not have the complete code ownership activity developed.

### 8.3 Comparing TickIT Guide 5.0 clauses in both projects

The Table 8.3 shows the results of comparing XP with TickIT clauses in both projects. As the reader can see very similar results are obtained from both projects unless they take different approaches to fulfill TickIT Guide 5.0 using XP. In the WAIS project they created two subgroups to manage the project, and in the W.H.A.T. project they create a Test Group, to deal with quality aspects of TickIT Guide 5.0.

<table>
<thead>
<tr>
<th>Color</th>
<th>W.H.A.T.</th>
<th>WAIS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>Total of clauses</td>
<td>51</td>
<td></td>
</tr>
</tbody>
</table>

Table 8.3: Results of comparing TickIT clauses with XP in both projects

### 8.4 Strengths and Weaknesses with two XP projects to fulfil TickIT Guide

In this section are established the strengths and weaknesses of both projects using the information obtained through the TickIT comparison with the projects in the previous chapters.

#### 8.4.1 Strengths with two XP projects to fulfil TickIT Guide 5.0

- **Quality of the code**
  
  The quality of the code was reviewed in the observations hours that the author has with both groups. The code has in general a good quality in both projects, but for example there was in one of the projects (i.e. W.H.A.T. project) an administrative tool was developed with not good quality of the code. In addition, in both projects there were security problems, because they have the connection code to the database in the server pages instead of creating procedures in the database and call the procedures from the server pages.

- **Customer focus**
  
  XP is highly committed with the customer. In the W.H.A.T project a person acting as a customer is used, and fulfil the customer focus in TickIT.
In the WAIS project, everyone can communicate with the customer receiving feedback from him. In addition, continuous delivery is used to increase the customer satisfaction in both projects.

- **Internal Communication**
  This highly answered by XP, because everyone is in the same room, and can communicate orally with each other. The knowledge is transferred naturally through groups members in both projects.

### 8.4.2 Weaknesses with two XP projects to fulfil TickIT Guide 5.0

In this section there is a description of where are the problems with TickIT using XP as a methodology using the information obtained from the interview.

- **Documentation**
  They detect a documentation problem. TickIT wants that almost everything should be documented and with a high quality. Instead XP says that the Team should document what is ‘good enough’ for the project not everything. The documentation needed by TickIT are:
  - W.H.A.T. Test Plan (Sandberg 2004b).
  - W.H.A.T. Organization (Sandberg 2004d).
  - W.H.A.T. Risk Management (Sandberg 2004e).
  That was supplied by the team (i.e. W.H.A.T. project) as an extra, but according to them that is not a part of XP. The same for the other team (i.e. WAIS project).
  The requirement documentation is supplied by the use stories, according to the TickIT advisor the documentation was good but a little bit to abstract.
  According to him, the main problem is that they do not have documented any nonfunctional requirement. From the author point of view this is not only a problem of the W.H.A.T. or WAIS problem, it is a problem of XP where the documentation for them are only code, tests and user stories.

- **Risk management**
  Risk management is defined in the clauses 7.1, 7.2.2, 7.3.1, 7.3.4 and 8.5.3 in the TickIT Guide 5.0, see (Leakey & Restell 2001).
  In XP they do not mention about risk management, the team interprets that as they do not do it.
  XP have not write anything for the risk of the future. They deal with the problem in the present when they appear. The problem that the interviewee remarked in both projects was that they were working with a certain architecture that have certain problems with scalability, but when the problem emerged they could not do anything, and they had to change
the whole architecture.

In the W.H.A.T: team they take into consideration risk management in the latest iteration, where they start to discuss about risk in the agile meetings.

- **Up-front design**
  At the beginning of the project both teams want to follow what Kent Beck says that the team should not be working with any up-front design or as Ron Jeffries says that you should use 30 minutes in each use story for up-front design and architecture. The team interpretation was that they continuously improve the design.

  But, as the project was evolved they realize that it is necessary to create an up-front design for the system, because the size of it. It is not very detailed, but at least it is something to base on.

  Design issues are described in clause 7.3 in the TickIT Guide 5.0.

- **Measurements**
  They have only few measurements as is presented in the previous subsections, but they do not measure what happens with testing. For example, how many test pass and do not pass this day, in which percentage. XP does not explain about measurements in a project, that is why the team come out with some measurements, but not the necessary to fulfill TickIT.

  According to the TickIT auditor the measurement for testing it should be done at least in the integration and system level.

  From the TickIT auditor and author point of view there are some problems in the XP practice of test-driven development explaining about integration tests and system tests, because they mention only explicitly unit tests and for the team it was really difficult to come out with some practices of integration and system testing.

  But, as the project evolved they create a Test Group that are responsible for the tests and measurements of them.

  Measurements, analysis and improvements are established in clause 8 of the TickIT Guide 5.0.

- **Control of documents**
  XP is not specific about control of documents, instead TickIT needs a control of documents. The project leader in both projects is in charge of controlling the documents in the project. Control of documents are established in clause 4.2.3 in TickIT Guide 5.0

- **Project management**
  They use different approaches to manage the project, in one of the teams (i.e. WAIS project) they use the most traditional approach were people follow the commands of the groups leaders and the project leader. They have daily meetings, Monday meetings for continuous improvement of the project, but they manage in an ad-hoc manner, according to the author interpretation this is a lack of experience in managerial tasks from the leaders of the projects. From another point of view they come out by themselves to divide the project in two subproject.

  The other team (i.e. W.H.A.T project) uses a more agile approach for developing software with the planning game in each iteration, agile
meetings after each iterations and daily meetings for control of the project. They come out with this by themselves.

XP presents only the planning game as a managerial task, instead TickIT Guide 5.0 has a entire clause talking about management responsibility (i.e. clause 5).

- **Traceability**
  Traceability from customer requirements until code it is not defined in XP. It is supposed that the team should self-organize to answer that requirement of traceability.
  As their first project using XP both projects have some problems with traceability and control of it through the development.
  Traceability is established in the TickIT guide in clause 7.5.3.

- **Quality objectives**
  One project (i.e. W.H.A.T. project) have documented quality objectives as performance, stability, etc., that are related to nonfunctional requirements. They will be tested in the last part of the project. In the other project (WAIS project) the nonfunctional requirements will be not tested at all, because of time.
  In XP only functional or technical requirements are defined using use stories, but they do not explain how to deal with nonfunctional requirements.
  XP forget the big picture of the system with the nonfunctional requirements. For TickIT Guide 5.0 the quality objectives are established in clause 5.4.1.

- **Reviews and audits**
  The reviews and internal audits are made because of TickIT to increase the quality of the product, but they are not specified by XP. Although, internal reviews in XP can be viewed as Pair Programming where one pair is checking constantly what the other is written. In the case of W.H.A.T. as only few people pair programming, they need to change XP with some kind of Quality Group that is in charge for testing and reviews.
  In order to use pair programming there is a need of high discipline in the group, because it is needed to rotate the pairs.
  In the other project people pair programming without any problems pairs were rotated within other pairs, that can be considered as a review, but people do not documented it, as is needed by TickIT Guide 5.0, they are described in clauses 5.6 and 8.2.2.

**8.5 Summary**

In this chapter both projects were compared from the point of view of XP, that means from XP practices and activities were compared; and from the point of view of TickIT Guide 5.0 from its different areas of software development.
Chapter 9

Conclusions and Recommendations

The best processes in the world will not save a project from failure if the people involved do not have the necessary skills to execute the process; conversely, really good developers can make any process work.
- Pete McBreen [McBreen 2001, p. 35,36]

The goal of this chapter is to develop the conclusions and recommendations from this thesis.

9.1 Introduction

This chapter is divided in two sections the first one are the conclusions obtained developing this thesis. All the research questions that are defined in the introduction will be answered. The second one are the recommendations given to these XP projects to fulfill TickIT Guide 5.0.

9.2 Conclusions

In this section the answer to the research questions are developed.

- Are the agile development values and principles with Quality Management Systems (QMS) principles and ISO 9001:2000 - TickIT Guide 5.0 complementary or they conflict among each other?

The values in the agile development conflicts in some parts with QMS principles and ISO 9001:2000 - TickIT Guide 5.0, as for example in agile development persons are highly valuable, but for a QMS the process has a mayor importance, therefore if QMS relay more in the process the organization can easily change a person from one project, but in the case of agile development that is no so fluent. For agile proponents is the person
that has a high importance, and gives values to a project, improving the quality of the code and the process with it experience.

If the coach tries to replace one of the team members in a specific iteration, he has to teach her or him about the XP practices that are used in the project, as for example, test-driven development, or design patterns in order to make design improvements, therefore there is a lot of time involved on it, instead in a QMS there are some documents that the person should study in a time-frame not so long as in XP, and the person in this case works only in one specific task, instead in XP people should work in multiple tasks as programmer, tester, database manager, etc. throughout the development. He or she needs huge amount of knowledge.

Another discrepancy is that QMS rely mainly on planning and documents, that are described in TickIT Guide 5.0, they do not define working software as the essence of software development. From another point of view, customer focus is followed by both approaches, they agree on that part.

Talking about the principles in agile development and principles in QMS and the TickIT Guide 5.0, occurs the same as in the agile values, in some parts they agree and in other parts they are in conflict, however in other parts they do not agree or disagree. For example, they agree in continuous process improvement but disagree that the best deliverables come out from self-organizing teams, for TickIT Guide 5.0 it is highly valuable the project manager as the main source for quality in the product and not the team. They do not agree or disagree in working software as a primary measure of progress in software development. For more information, see Appendix ?? on page ??.

- What are the strengths and weaknesses of each model?

The principal strength of ISO 9001: 2000 - TickIT Guide 5.0 are that using the TickIT Guide 5.0 the organization can develop a QMS for continuous process improvements, that gives to the organization the opportunity to use efficiently its resources to increase the quality of the software products and processes. If the process turns to be more efficient the company can create better products and with that gain market shares in its business.

The weaknesses are that the process turns to be too documented and because of that it does not bring mobility to the company to deal with changing environments in today economy. Another weakness is the high cost that the company should pay for training the people involved in process, the cost for certification that should be renewed every two or three years, in a manner to evaluate that the company is in continuous process improvement. Moreover, another weakness is that senior management is not highly committed with the process and then the process will not have the maturity and efficiency that is necessary to confront changes. There is a need of senior management commitment, because the continuous process improvement process must be institutionalized in the whole organization, if not only some projects will use this process and turn it on more efficient and not the whole organization, in order to establish differences with its competitors.
Now from the point of view of XP its strengths are that people have more fun working with XP (i.e. pair programming (Williams & Cockburn 2001)), but as it was observed in the project when the author was talking with the team, some of them hate XP, but that is due of this is the first project using XP for their coaches, with an experience XP coach this will not happen. Another strength of XP as in the case of these projects is that the team continuously deliver part of the software product in each iteration, obtaining with that constant feedback from the user. One more issue in favor is that the quality of the code with XP is high due to a continuous process of design improvements.

Weaknesses in XP lack of an up-front design, they just code. Lack of traceability, is another issue and the author sent an email to Kent Beck and he answer that in every project he implements traceability from requirements to code, but how a team can implement traceability using XP it is not in the books. They (i.e. both groups) handle changes in the requirements or design with fast changes in the code and using continuous iterations to show how the code is evolving to the customer.

The team needs high discipline and experience to work with all the XP practices, if not XP does not work as it must be. Another important issues is that how to implement integration testing and system testing, in the book they do not explain about that and some teams as W.H.A.T. and WAIS are lacking on that. Measurements, risk management, and project management are the other areas where XP is lacking. These last areas can influence the control of the project that the teams have on it. Another main point is about scalability problems that the team has, because they self-organize change and adapt to overcome these scalability creating an hybrid approach of XP.

• Which XP practices, activities, and roles were developed by each team?

The teams were using a high percentage of XP practices up to 60% in one project and up to 75% in the other, the others were not completely fulfilled or not fulfilled. From these that were not fulfilled in one team was test-driven development (i.e. W.H.A.T. team) and in the other the planning game (i.e. WAIS team), but both were lacking on the whole team practices, because they do not have the customer on site, moreover of that in one team the system metaphor was not fulfilled (i.e. W.H.A.T. team).

Talking about the activities the only problem was that in one of the team the activity of testing was not fulfilled, because they do not have test-driven development (i.e. W.H.A.T team)).

About the roles they create new roles as the quality managers, and the quality team in one team (i.e. W.H.A.T team), and in the other team (i.e. WAIS team) they divide the team in two subgroups where the subgroup responsible have the role of tracker and coach. Moreover, in one group were created the roles of configuration responsible, and backup responsible, in addition to the others roles (i.e. W.H.A.T. team).

• Are they using XP (i.e. W.H.A.T. and WAIS teams)?
CHAPTER 9. CONCLUSIONS AND RECOMMENDATIONS

Being rigorous the teams were not performing XP, they were using some of the XP practices with some kind of hybrid approach, as the reader can see in chapters 6, 7. A question was posted in the Extreme Programming group in yahoo.com, about when an XP project is really an XP project and the answer of one of the fellows there was that it should be follow all the practices, if not it is not XP, see http://groups.yahoo.com/group/extremeprogramming/message/92270. From another point of view there is an article wrote by Chet Hendrickson, where he address that problem also and he says that the team can start with the 12 practices, but then can adapt them and modify them to tailor XP to their purpose, the main issue according to him (Hendrickson 2000b) are the four XP values feedback, simplicity, communication, and courage. There is another article wrote by Ron Jeffries about Are we doing XP? (Jeffries 2000) that talks about the same issue and he says that the team should inevitable tailor XP to their special situation, but always should focus on the benefits on the practices and should maintain XP values. He adds mainly courage. 'Applying the principles for best effect is what is most important' according him.

From the researcher point of view, a project should be develop the XP practices gradually throughout the project. In order to use all of them at the end of the project, if that is their first project with XP, but if they have experience with XP they should try to use all them, from the beginning of the project and continuously measure them, for a best efficiency in the project. People should take in consideration that all the XP are very interrelated and require discipline and experience to do develop the whole XP methodology. As a summary, in order that people can call that their project is using XP, they should work with all the practices, values, principles, and activities of XP.

- Is an XP project compliant with TickIT 5.0 Guide?

In the case of these projects were analyzed XP with each clause of TickIT guide 5.0 and in up to 30% of these, they have a problem with them, and up to 20% of them were warning and up to 24% were not applicable, only 26% pass the TickIT 5.0 audit. From that point of view in the case of these academic projects XP it is not compatible, the main problem here in both projects was that to much people was involved in both projects, Kent Beck’s recommend up to 10 people in an XP due to communication and complexity issues.

It should be really good to investigate what is happens in and XP project with up to 10 in the industry with people with experience in XP and good coaches.

The recommendations of this author is that the team should adapt and extend XP in order to satisfy all the requirements in ISO 9001:2000 TickIT Guide 5.0. For more about recommendations see the next section.

- Can these models complement each other? In which areas?

The complementary areas are: the creation of a quality system, customer satisfaction due to a constant feedback from them, internal communica-
CHAPTER 9. CONCLUSIONS AND RECOMMENDATIONS

Risk Management.
- Measurement.
- Traceability.
- Documentation.
- Nonfunctional Requirements.
- Management.
- Up-front Design.
- System and Integration Tests.

• What are the benefits of using these models together?

Using XP to deal with ISO 9001:2000 standard brings agility to the company that can face changing environments. It means that the software organization can develop software with quality and in less time than before obtaining market shares by doing that.

ISO 9001:2000 brings to XP a process for continuous improvement to check what part of the process and products can change for a better performance, quality, and customer satisfaction. The company can reduce with XP the cost of documentation, developing only the necessary documents to be ISO 9001:2000.

For unexperienced XP teams, TickIT Guide 5.0 brings a framework of quality practices, if they do not have the necessary discipline to follow all the practices in XP to deliver a software product.

9.3 Recommendations

The following recommendations are for improving XP in order to fulfil the ISO 9001:2000 clauses for software as are described in the TickIT Guide 5.0.

About the problems with integration testing and system testing that both projects have using XP the author have posted a question on the Extreme Programming group and the answer was that XP does not make this explicitly, but in practice they are using two approaches one of them is the use of MockObjects and the other is from ThoughtWorks, first published in the XP Universe Conference 2001 (Schuh & Punke 2001) where the unit test is continuing to growing and growing until is an integration test and system test, see http://groups.yahoo.com/group/extremeprogramming/message/92272.

XP is recommended for teams with up to 10 members, after that the team should have communications problems among different people. Quoting Kent’s Beck in Extreme Programming Explained. He remarks “Size clearly matters. You probably couldn’t run an XP project with a hundred programmers. Nor fifty. Nor twenty, probably. Ten is definitely doable.” (Beck 1999b, p. 157), for more information where a team should not use XP, see (Beck 1999b, c. 25). For larger projects it is necessary to complement XP with other agile management

1See the website http://www.thoughtworks.com/
methodologies as Scrum[^2] or DSDM[^3] for example, but that will not be XP anymore.

In order to fulfill all the XP practices it is required a highly disciplined group, with some people with a lot of experience in software development in order to share their knowledge through the group.

In a question that has given to Ken Schwaber[^4] in an informal way in the Scrum Master Certification[^5] days in April, Vienna this year, one of my fellows asked about how long does it take from a team have the necessary discipline to follow all the XP practices? He answered from 6 to 8 months.

XP brings together a highly coupled engineering practices, it is not a management methodology. The recommendation is to use XP@Scrum[^6]. Moreover, with Scrum the team can resolve scalability problems as is detailed in (Stephens 2001b, c. 15) with XP as Ken Schwaber explains in (Schwaber 2004, c. 9).

About XP and DSDM, the DSDM Consortium[^7] is developing at this moment a new framework that they called EnterpriseXP[^8]. Matt Stephens and Doug Rosenberg’s book *Extreme Programming Refactored: The Case Against Extreme Programming* (Stephens 2001a) presents another approach where they make some suggestions to XP where they modify some of the practices and add another ones. This is explained in (Stephens 2001a, c. 15).

In Barry Boehm and Richard Turner’s book ‘Balancing Agility and Discipline: A Guide for the Perplexed’ (Boehm & Turner 2003), the authors explain how to balance agile development and plan-driven development using a risk-driven approach. The advices that can be founded in the book are good for every agile development, not only for XP.

For the design, the recommendations are to use some of the design patterns described by Robert C. Martin in his book about *Agile Development and Patterns and Practices* (Martin 2002) with Martin Fowler book’s about refactoring (Fowler et al. 1999), and also spend some time with the up-front design in each iteration as Martin Fowler recommends in one of his articles about 20% of the development should focus on up-front design (Fowler 2001). It depends how large is the architecture and design the team should spend more time on up-front design, the team should balance the rate for each particular project.

For following a continuous improvement process it should be good that in the agile meetings after each iteration take in consideration project retrospectives[^9] activities to perform an iteration retrospective, however the process should be tailored for about 4 hours. After a project is finished a full project retrospective can be taken in consideration. The author of this thesis was talking with Norman Kerth author of ‘Project Retrospectives: A Handbook for Team Reviews’ (Kerth 2001) at the Scrum Gathering[^10] held in Vienna in April of this year, and he said that project retrospectives should be taken in consideration not only for projects improvement else to an organizational tool for institutionalization con-

[^3]: More information about DSDM, [http://www.dsdm.org](http://www.dsdm.org)
[^4]: Author of Scrum Development Process, his website is the same of Scrum, see footnote[^9]
[^5]: More information about Scrum Certification is available on, [http://www.certifiedscrum.com](http://www.certifiedscrum.com)
[^6]: More information about XP@Scrum, see [http://www.controlchaos.com/xpScrum.htm](http://www.controlchaos.com/xpScrum.htm)
[^7]: Idem footnote[^9]
[^8]: For more information about EnterpriseXP, see [http://www.enterprisexp.org/](http://www.enterprisexp.org/)
[^9]: For more information about project retrospectives, see [http://www.retrospectives.com/](http://www.retrospectives.com/)
[^10]: For more information about Scrum Gathering, see [http://gathering.scrumalliance.org](http://gathering.scrumalliance.org)
9.4 Summary

In this chapter are developed the conclusions were the answers to the research questions can be found, following with the recommendations that this author come out to change a little bit XP to fulfill the entire ISO 9001:2000 standard.
Chapter 10

Further research

If a process can be fully defined, with all things known about it so that it can be designed and run repeatably with predictable results, it is known as a defined process, and it can be subjected to automation. If all things about a process aren’t fully known-only what generally happens when you mix these inputs and what to measure and control to get the desired output-these are called empirical processes.
- Ken Schwaber (Schwaber, 1996)

The goal of this chapter is to describe the further research studies that can be held using this report as a base.

There is a series of further researches that can be handled as for example:

- What kind of problems can be found in the maintenance phase after an XP project was certificated using ISO 9001:2000 using the TickIT Guide 5.0?
  For example, how much documentation is needed to follow with the maintenance phase?, how much people from the development team is necessary to work together with the maintenance team?, in order that they can follow with the system updates, how long will be the process were both people will work together?

- It can be developed a research about XP and other software process improvement models as CMM or CMMI.
  For example, a comparison of the process areas in CMM and XP practices to fulfill CMM can be developed. The same in the case of CMMI. Is it possible that compatibility in a real project?. The project should be developed in a CMM certificate company.

- It can be performed a research about XP and Scrum to deal with a TickIT certification.
  In this case, a similar approach should be taken in consideration using this MSc. thesis as a guideline to work with academic projects or with
some industrial projects were the companies were using XP@Scrum and the company should be ISO 9001:2000 certificate.

- Another can be the use of XP and DSDM to deal with a TickIT certification. In order to do this, the study should be carried out in two years when EnterpriseXP should be finished an some companies can start using it.

- Finally, it can be done a research in the industry, about how the use XP to be TickIT certificate. The author was thinking to study about that and compare it with the results here at the university, but that was not possible due to problems to meet the person in the industry.

10.1 Summary

In this chapter were described the different kinds of further research that people can develop from the information obtained in this report, for example, studying the maintenance phase in an XP that is used for TickIT certification, the use of XP@Scrum or EnterpriseXP with TickIT Guide 5.0 and finally a comparison with a industry case study in a project that are using XP and is developed in a ISO 9001:2000 certificate organization, with both academic projects.
Bibliography


Cockburn, A. (2000), *Writing Effective Use Cases*, Crystal Collection for Software Professionals, 1st edn, Addison-Wesley, Massachusetts, USA.


Marchesi, M., Succi, G., Wells, D., Williams, L. & Wells, J. D. (2002), Extreme Programming Perspectives, XP Series, 1st edn, Addison-Wesley, Massachusetts, USA.


McBreen, P. (2002), Questioning Extreme Programming, XP Series, 1st edn, Addison-Wesley, Massachusetts, USA.


URL: [http://www.tickit.org/ti3q02.pdf](http://www.tickit.org/ti3q02.pdf)


Appendix A

Questionnaire

A.1 Opening the interview
1. Introducing myself
2. Introducing them
3. What is the interview about?

A.2 The customer company
1. What is the name of the company?
2. What does the company do?
3. How many employees has the organization?
4. How many offices has the company?
5. Where the offices are located?

A.3 The project
1. What is the name of the project?
2. What is the project about?
3. How many people have been involved?
4. How do you manage the project?
5. What tools do you use in the project?
6. What kind of contract do you use (e.g. fixed-contract, open-contract, feature-contract)?
7. What are the XP roles used in the project?
8. How many people are involved in each role?
APPENDIX A. QUESTIONNAIRE

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPRole1</td>
<td>Programmers</td>
</tr>
<tr>
<td>XPRole2</td>
<td>Tester</td>
</tr>
<tr>
<td>XPRole3</td>
<td>Coach</td>
</tr>
<tr>
<td>XPRole4</td>
<td>Customer</td>
</tr>
<tr>
<td>XPRole5</td>
<td>Big Boss</td>
</tr>
<tr>
<td>XPRole6</td>
<td>Consultant</td>
</tr>
<tr>
<td>XPRole7</td>
<td>Tracker</td>
</tr>
</tbody>
</table>

Table A.1: Roles used in XP

9. Did you modify any of the XP practices? How? Why?

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPPractice1</td>
<td>Whole Team</td>
</tr>
<tr>
<td>XPPractice2</td>
<td>Simple Design</td>
</tr>
<tr>
<td>XPPractice3</td>
<td>Collective Code Ownership</td>
</tr>
<tr>
<td>XPPractice4</td>
<td>Coding Standards</td>
</tr>
<tr>
<td>XPPractice5</td>
<td>Design Improvement</td>
</tr>
<tr>
<td>XPPractice6</td>
<td>Planning Game</td>
</tr>
<tr>
<td>XPPractice7</td>
<td>Test-driven development</td>
</tr>
<tr>
<td>XPPractice8</td>
<td>System Metaphor</td>
</tr>
<tr>
<td>XPPractice9</td>
<td>Small Releases</td>
</tr>
<tr>
<td>XPPractice10</td>
<td>Pair Programming</td>
</tr>
<tr>
<td>XPPractice11</td>
<td>Continuous Integration</td>
</tr>
<tr>
<td>XPPractice12</td>
<td>Sustainable pace</td>
</tr>
</tbody>
</table>

Table A.2: XP Practices

10. Have you use all the XP activities? In which degree?

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPAActivity1</td>
<td>Coding</td>
</tr>
<tr>
<td>XPAActivity2</td>
<td>Testing</td>
</tr>
<tr>
<td>XPAActivity3</td>
<td>Listening</td>
</tr>
<tr>
<td>XPAActivity4</td>
<td>Designing</td>
</tr>
</tbody>
</table>

Table A.3: XP Activities

A.4 XP and TickIT 5.0

1. Which are the conflictive areas in ISO TickIT to be complete by XP?
   - Examples
   - Documentation
   - Risk Management
   - Design Requirements
   - Architecture
   - Management
   - Measurement
Table A.4: Conflictive areas in TickIT with project X

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TickITArea1</td>
<td>Quality System</td>
</tr>
<tr>
<td>TickITArea2</td>
<td>Management Responsibility</td>
</tr>
<tr>
<td>TickITArea3</td>
<td>Resource Management</td>
</tr>
<tr>
<td>TickITArea4</td>
<td>Product Realization</td>
</tr>
<tr>
<td>TickITArea5</td>
<td>Measurement, analysis and improvement</td>
</tr>
</tbody>
</table>

2. If there were some problems, how do you solve those problems in order to accomplish TickIT standard using XP?

3. Did the project pass the TickIT certification?

A.5 Closing the interview

1. After the end project, how will be developed the maintenance phase?

2. Will you repeat the experience of XP next time in other project? Why?

3. Would you like to comment something else?
Appendix B

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- **Individuals and interactions** over processes and tools.
- **Working software** over comprehensive documentation.
- **Customer collaboration** over contract negotiation.
- **Responding to change** over following a plan.

That is, while there is value in the items on the right, we value the items on the left more\(^1\).

Kent Beck    James Grenning    Robert C. Martin
Mike Beedle   Jim Highsmith    Steve Mellor
Arie van Bennekum Andrew Hunt Ken Schwaber
Alistair Cockburn Ron Jeffries Jeff Sutherland
Ward Cunningham Jon Kern Dave Thomas
Martin Fowler  Brian Marick

B.1 Principles behind the Agile Manifesto

We follow these principles:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

2. Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage.

\(^1\)©2001, the above authors. This declaration may be freely copied in any form, but only in its entirety through this notice.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

4. Business people and developers must work together daily throughout the project.

5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

6. The most efficient and effective method of conveying information to and within a development is face-to-face conversation.

7. Working software is the primary measure of progress.

8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

9. Continuous attention to technical excellence and good design enhances agility.

10. Simplicity—the art of maximizing the amount of work not done—is essential.

11. The best architectures, requirements, and designs emerge from self-organizing teams.

12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.
Appendix C

Agile development v/s TickIT Guide 5.0

In this Appendix will be held a theoretical comparison between agile values and principles described in chapter 3 sections 3.2 on page 19, 3.3 on page 21 with QMS principles as were showed in chapter 2 section 2.2 on page 8 and TickIT Guide 5.0 clauses 2.4 on page 10. The following Tables C.1, C.2 were improved from the work of (Southwell 2002).

<table>
<thead>
<tr>
<th>Agile Values</th>
<th>Grade</th>
<th>Comments about QMS Principles and TickIT Guide 5.0 clauses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Individuals and interactions over processes and tools.</td>
<td></td>
<td>On QMS Principle 1, see section 2.2 on page 8 that talks about involvement of people, this agile value is reflected, but from another point of view in QMS there is a huge emphasis on QMS Principle 4, see section 2.2 on page 8 where is described a process approach, therefore as a conclusion TickIT Guide 5.0 promotes involvement of people under the umbrella of process approach. This is the opposite of this agile value.</td>
</tr>
<tr>
<td>Working software over comprehensive documentation.</td>
<td></td>
<td>There are a large amount of records and documentation that are needed through the entire TickIT Guide 5.0, as is defined in the clause 4.1 (Leakey &amp; Restell 2001) about general requirements of a QMS, that can cause to lose the focus that the most important issue is working software after all.</td>
</tr>
</tbody>
</table>
### Agile Values

<table>
<thead>
<tr>
<th>Agile Values</th>
<th>Grade</th>
<th>Comments about QMS Principles and TickIT Guide 5.0 clauses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer collaboration over contract negotiation.</td>
<td></td>
<td>QMS Principle 1 <em>customer focus</em>, that talks about fulfill customer needs and expectations, section 2.2 on page 8.</td>
</tr>
<tr>
<td>Responding to change over following a plan.</td>
<td></td>
<td>In TickIT Guide 5.0 there is nothing that specify about change in the requirements. They highly develop the idea of create a plan for the whole development, see clause 7.1 in TickIT Guide 5.0 (Leakey &amp; Restell 2001). Then it is possible to consider this as a contradicting point, between this agile principle and the TickIT Guide 5.0.</td>
</tr>
</tbody>
</table>

Table C.1: Comparison between agile values and QMS principles, TickIT Guide 5.0

### Agile Principles

<table>
<thead>
<tr>
<th>Agile Principles</th>
<th>Grade</th>
<th>Comments about QMS Principles and TickIT Guide 5.0 clauses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.</td>
<td></td>
<td>In the QMS Principle 1 <em>customer focus</em> it is explained that customer requirements should be satisfied, but there is nothing on TickIT Guide 5.0 that talks about time delivery or can contradicts it or the highest priority term.</td>
</tr>
<tr>
<td>Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage.</td>
<td></td>
<td>There is nothing in the TickIT Guide 5.0 that talks about changing requirements and be used as a competitive advantage with the customer.</td>
</tr>
<tr>
<td>Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.</td>
<td></td>
<td>Again there is nothing in the TickIT Guide 5.0 that talks about how frequently should be the software to be delivered, and can contradict or support this agile principle.</td>
</tr>
<tr>
<td>Business people and developers must work together daily throughout the project.</td>
<td></td>
<td>There is nothing that in the TickIT Guide 5.0 contradict this agile principle, but the important issue is that the collaboration between business people and developers is “daily”.</td>
</tr>
<tr>
<td>Agile Principles</td>
<td>Grade</td>
<td>Comments about QMS Principles and TickIT Guide 5.0 clauses</td>
</tr>
<tr>
<td>---------------------------------------------------------------------------------</td>
<td>-------</td>
<td>------------------------------------------------------------</td>
</tr>
<tr>
<td>Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.</td>
<td></td>
<td>The principle about involvement of people in QMS reflects the same thoughts around people motivation. The environment and support is related to clause 6 (<a href="#">Leakey &amp; Restell 2001</a>) about Resource Management where this issues should be provided by the organization to the team.</td>
</tr>
<tr>
<td>The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.</td>
<td></td>
<td>TickIT Guide 5.0 (<a href="#">Leakey &amp; Restell 2001</a>) focus on documentation through the whole guide rather than conversation, but if the reader see the QMS Principle 3 about people involvement in section 2.2 on page 8. They talk about that, but not explicitly in the TickIT Guide 5.0.</td>
</tr>
<tr>
<td>Working software is the primary measure of progress.</td>
<td></td>
<td>In the clause 8 of the TickIT Guide 5.0 (<a href="#">Leakey &amp; Restell 2001</a>) about measurement and analysis of the software that has been developed, but they say nothing about working software as a primary measure for customer satisfaction.</td>
</tr>
<tr>
<td>Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.</td>
<td></td>
<td>It is not mention explicitly in (<a href="#">Leakey &amp; Restell 2001</a>), but it may be related to QMS Principle 5, where a system approach to management is defined. It can be that as a result of working with software using a systemic view. see section 2.2 on page 8.</td>
</tr>
<tr>
<td>Continuous attention to technical excellence. and good design enhances agility.</td>
<td></td>
<td>In TickIT Guide 5.0 is not specified explicitly this issue, but it can be founded implicit in clause 6.2 (<a href="#">Leakey &amp; Restell 2001</a>) about competence, awareness and training related to human resources.</td>
</tr>
<tr>
<td>Simplicity -the art of maximizing the amount of work not done- is essential.</td>
<td></td>
<td>Again, nothing is writing about this in the TickIT Guide 5.0 (<a href="#">Leakey &amp; Restell 2001</a>).</td>
</tr>
</tbody>
</table>
Agile Principles | Grade | Comments about QMS Principles and TickIT Guide 5.0 clauses
--- | --- | ---
The best architectures, requirements, and designs emerge from self-organizing teams. | | The stress in the TickIT Guide 5.0 is on management responsibility as the reader can see in clause 5.0 ([Leakey & Restell](#) 2001) and also described in the leadership principle on QMS, see section [2.2](#) on page 8 about that. They work with the traditional way of management where the manager is responsible for the project and each one in the team is responsible for the quality of some part of the development. Instead in agile development the responsibility is in the whole team, that turns to be a self-organizing one.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly. | | This principle in agile development reflects continuous improvement that is described in the QMS Principle 6 of *continual improvement*, see section [2.2](#) on page 8.

Table C.2: Comparison between agile principles and QMS principles, TickIT Guide 5.0
Appendix D

Acronyms List

Below the reader can see the acronyms used in the entire report:

ASD  Adaptive Software Development
CMM  Capability Maturity Model
CMMI Capability Maturity Model Integration
DSDM Dynamic Software Development Method
FDD  Feature Driven Development
LD   Lean Development
PSP  Personal Software Process
QMS  Quality Management System
SEI  Software Engineering Institute
TQM  Total Quality Management
W.H.A.T. Wireless Hotspot of Advanced Technology
XP   Extreme Programming
YAGNI You are not going to need it