Second LACCEI International Latin American and Caribbean Conference for Engineering and Technology (LACCEI'2004)
"Challenges and Opportunities for Engineering Education, Research and Development"
2-4 June 2004, Miami, Florida, USA

Extreme Programming (XP): Development of Software Industrial Projects and Their Impact in the Academic Environment.

Carlos Ventura
Computer Science Undergraduate Program Chair, ITESM, Chihuahua, Chih, Mexico.

Rodolfo Castello
Dean, School of Engineering and Computer Science, ITESM, Chihuahua, Chih, Mexico.

Abstract
Extreme Programming has been considered as an optional methodology for software development. XP is based on certain practices, which provide an informal method for fast development. Thus the cost is reduced since the analysis, design, and coding are done at the same time. Hence this kind of software development approach permits changes in both, priority, and non-critical functionalities without big scale rework. This document studies the impact on academia and customer satisfaction, by using XP on the development of an industry strength software project by freshman Software Engineering undergraduate students.

Keywords
Extreme Programming, Software Engineering, Software Engineering Education.

Introduction

Software development is treated often like an art, where the programmer sits in a desk and began to write down programs. Quality is often too dismissed due time constrains and client eager to see a functional application. More specifically, Software Engineering graduates students that begin to work on real projects don't have enough experience in real world problems. Hence, they often don't take care of the quality factor. This paper explores the usage of Extreme Programming (Beck, 1999) in introductory courser of the Software Engineering undergraduate program. This project will reinforce the educational model of the Tecnologico de Monterrey (Vicerrectoria, 2003) which promotes the auto study skills; thus the students will extend their criteria and abilities in analysis, design and development.

One of the most notorious problems during development is the continuous change of requirements. This change generates undesired costs to the client, who can't perceive the difficult that those changes impose to the develop team.

There are methodologies that show the effects of the change on requirements in the development of a system (O'neal and Carver, 2001). However, although these changes can be prioritized, the modification of requirements lead to a big scale effort. Even with those methodologies, changes or improvements in a system could generate a slowdown on work schedule, or in the worst case, project failure (Lam and Shankararaman, 1999). There exist models that study the impact of requirements change, such as NRM

(Kobayashi and Maekawa, 2001), but as they mention, “the stakeholders (the people immerse in the software project) rarely gives concrete and correct answers of the product that they want. Hence, there are researches about the risks that implies the change of functionality and requirements (Schneidewind, 2001).

Extreme Programming (Beck, 1999) is based in the analysis, design, codification and implementation of little pieces of code. This methodology permits greater flexibility if changes on priority or functionality are required. One of the principal practices that XP establishes, is the integration of the client into the develop team. In fact, the client is practically part of the team. Although in the project that is presented in this article, the client will not be with the developer team 100% of the time, a solution is proposed for this kind of cases.

This article describes a research topic which will employ some concepts of XP in freshman students in software engineering. XP has been used satisfactory on programming courses at universities (Jovanovic et al, 2002). In our project, the concepts of XP will be used on a real project that will be fully functional on the company, from where the client belongs. The impact of XP on the development of a real project made by students is the key point of this research.

The second section describes in a generic form, the practices and principles of XP. The third section mentions the general characteristics of the system to be developed. Section four enumerates the modifications to the methodology of XP that we propose for our purposes. Section five describes our preliminary results. Finally, section six shows the conclusions and future work for this research.

2. Extreme Programming (XP)

Introduced in 1996, Extreme Programming (Schneider and Johnston, 2003), (Beck, 1999), (Wagner, 2001) is based on the simplest solution of a problem, which fulfills the requirements of the client´s project. XP assumes that the client will be always available for the developers. XP promotes pair programming practice, code testing, and continuous integration of code. XP also promotes the rapid delivery of small applications. However, XP is often rejected as a development methodology by software companies. This is because some practices are supposed not to be optimal, e.g. Pair programming. Pair programming establishes that two programmers are at the same computer with one keyboard/mouse/monitor. The justification of this practice is that, while one programmer is working on the code, the other is auditing the code. In this way, the validation and verification process is optimized. Pair programming permits one programmer to check for syntax, logical, and semantics errors, among others. Thus errors are corrected at this pace, avoiding their presence in later stages of the development. In fact, there exists empirical evidence that shows that pair programming can reduce errors (Jensen, 2003). Even with these studies about the reduction of errors, managers see this practice as a waste of human resources. Often, managers prefer that every programmer should be working on different codes. Eventhough, this approach could be true from the managerial point of view, the code could have more errors, be less efficient, and furthermore, the code will suffer a delay.

It is important to mention that XP can’t be applied to all kind of projects (Duncan, 2003). This methodology can be applied only with small workgroups, teams with at most fifteen programmers, and coding is the most important task. XP is based in the interaction with the client to resolve this important question: which is the most important part of the system that is required to be implemented immediately? These selected parts of the system are known as *stories*. This process must be understood as the approach of the client to the technical part of the system. He must know how every stage of the system is developed, not only the visualization from the manager point of view. This is the reason that indicates that a project under XP will fail if the client isn’t present during all the stages of the development and, furthermore, if he doesn’t know all the characteristics of the system. Figure 1 shows this aspect of the XP process.

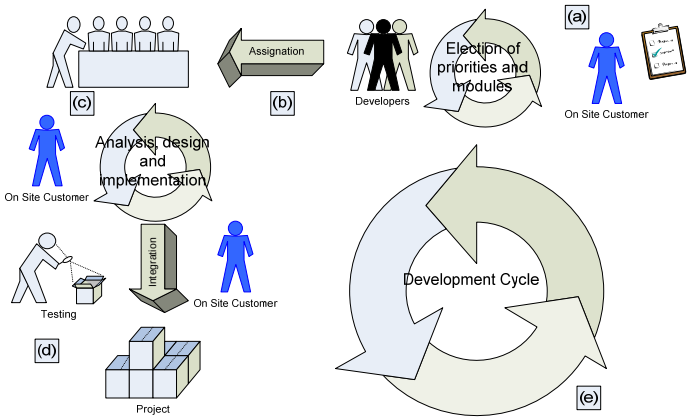

Figure 1.- Development Cycle

(a) The client is available on all stages of development. He is the responsible of choosing which of the modules will be developed (order). Hence, he must establish priorities and, if required, the changes on requirements. (b) Developers estimates the duration of a story, and then, are assigned to them. (c) The story is analyzed, designed, implemented, and tested. (d) If the code pass all tests is integrated to the project. (e) These steps are repeated until the system is finished.

The development cycle of XP is based in a set of common practices and concepts that have been used through various successful projects that have used them (Schneider and Johnston, 2003), (Duncan, 2003), (Beck, 1999), (Maurer and Martel, 2002). These practices are described below.

1. *Planning game*. Customers decide the scope and timing of releases based on estimates provided by programmers.
2. *Small Releases*. The system is put into production in a few months, before solving the whole problem. New releases are often made.
3. *Metaphor*. The shape of the system is defined by a metaphor shared between the customer and programmers.
4. *Simple Design*. At every moment, the design runs all the tests, communicates everything the programmers want to communicate, contains no duplicate code, and has the fewest possible classes and methods. This rule can be summarized as, “Say everything once and only once”.
5. *Tests*. Programmers write unit tests minute by minute. These tests are collected and they must run correctly.
6. *Refactoring*. The design of the system is evolved through transformations of the existing design that keep all test running.
7. *Pair Programming*. All production code is written by two people at one screen/keyboard/mouse.
8. *Continuous integration*. New code is integrated with the current system after no more that a few hours. When integrating, the system is built from scratch and all tests must pass or the changes are discarded.
9. *Collective ownership*. Every programmer improves any code anywhere in the system at any time if they see the opportunity.
10. *On-Site Costumer*. A customer sits with the team full-time.
11. *40 hour weeks*. No one can work a second consecutive week of over-time. Even isolated overtime used too frequently is a sign of deeper problems that must be addressed.

12. *Open Workspace*. The team works in a large room with small cubicles around the periphery. Pair programmers work on computers set up in the center.
13. *Just Rules*. By being part or an Extreme team, you sign up to follow the rules. But they're just rules. The team can change the rules at any time as long as they agree on how they will assess the effects of change.

The raising of XP has its roots in the special demanding situations for some industry software projects. Some of the successful industrial projects developed under XP (Beck, 1999) are:

DaimelerChyrsler. The purpose was to create a payment system, since the current system had some critical deficiencies. After 33 weeks the new system was working and ready to be tested. The system was released in 1999. This project took ten programmers, and the duration was four years.

Ford Motor Company. Cost analysis system, a six years period and twelve programmers.
Tarriff System. This one had a three months duration and used three developers. The purpose was to estimate shipping and handling fees.

During the development of these projects, some weakness and disadvantages were found in using XP. The following are the common weaknesses that were detected (Class, 2001):

1. *Pair Programming*. Some times the concept of pair programming dislikes the programmers, and there are problems to accept another programmer side by side.
2. *Optimization*: code optimization could minimize design, thus the quality of the product could be poor.
3. Usage of short periods of planning.

However, from the Capability Maturity Model (CMM) point of view, XP may be a complementary methodology (Paul, 2001). While CMM says what to do, not how to do it, XP provides enough information about how to do and implement the development of a software product.

3. Project to be developed (SET Tools)

This project rises as result of the collaboration between the ITESM Campus Chihuahua and the local industry of Chihuahua. SET Tools (Simultaneous Engineering Team Tools) has the finality of showing the estimate time at which an engineering process is involved. When certain changes or processes are defined, dates are established for the accomplishment of their different stages. The work of the supervisors is to monitor how the schedule is followed by their engineering staff. The supervisors can change the due dates, tasks, and priorities, among others. The original system was developed under Microsoft Excel. All the information flows from spreadsheets and Microsoft Project files. Later, these data is processed and it is shown in a friendly interface. This project was chosen as a starting point, because the level of difficulty is moderate and there already existx one implementation (although is not the best); and besides, it will permit the integration and learning of XP to our students.

4. Proposal

Developers of the System will be Software Engineering freshman students. This will permit the application of the XP practices in the development of the Set Tools System. Certainly, all the practices of XP can't be applied (e.g., 40 hour week) to the project. In consequence, some modifications are proposed and adapted to a university environment. This project has intermediate difficulty level. However, students will count with the advise of their professor during certain stages of the development process. Certain aspects of XP will be modified in order to adapt it to some constrains, such as time, documentation, and the on-site client. The most important aspects of the methodology that will be modified are the next: On site Costumer, 40 hours working week, and workspace.

Because the students involved in the project are regular students that have other subjects, the project will have a schedule of 10 hour per week per team, under their professor supervision. While not counting with the possibility of an on-site customer, histories will be created every week. The histories will be evaluated by professors and assigned to the teams. The implementation will be delivered on the following week. As an additional method of documentation, scanners and digital cameras (NawRocki et al, 2002) will be used to archive interviews with the costumers. This will allow the team to have rapid access to documentation. The schema proposed is the following:

a) Every two weeks the client will meet the teams to evaluate current work.
b) The doubts and questions will be exposed on the meetings with the client.
c) A log will be used. This will permit fast access to information.
d) The meetings and interviews will be recorded by Digital Cameras. This information will be saved on disk, available for further revisions.
e) At the end of session, the modules will be evaluated and the students will receive feedback from the client.

An important aspect is to use one of the professors as the on-site costumer (Williams and UpChurch, 2001). The knowledge of the professor will ensure accuracy in minor decisions about the system.

5. Preliminary Results

This project counts with eleven students. Although it was an open project, without being assigned on a particular course, the project had good acceptation between the students. From an initially 15 people, 11 took the project and accepted the extra academic work. Students are regular in computer programming courses. Students with more experience were distributed in the teams, so they can take the leadership on the teams.

5.1 Understanding the system

The first part of the development was focused on the interface. All interfaces were given to different teams, so that they could familiarize with the current system. Some of the interfaces needed special algorithms to display information on screen. Due this fact, students developed the algorithms and programmed them. In the first meeting, when the results were presented, the customer was very satisfied with the actual work developed by the teams.

5.2 The gap between developers and clients

As it is well know, even the customer doesn't know what they need for their system. Clients often are looking for the principal objective of the system, but they don't cover all the aspects that are need. As an example, a new task requires members, but who is the responsible of assigning these members?, which is the hierarchy of the different members?, from where do they take these members?, can all members perform changes to tasks?

In this case, the limitations are because the current Excel system doesn't take care of permissions. Consequently, in the new system, these modules are minimized or ignored. Even students didn't take care of these details. Having these modules out of control and scope, release time could be compromised; therefore, professors with knowledge on the development process are required. Some of the principal details that where forgotten until now are:

- Hierarchy issues
- Access issues
- Platform issues.

5.3 Evaluation

Project evaluation is based in two great axis: customer satisfaction (through feedback) and student's knowledge (through feedback and inspection). The actual evaluation metrics for the customer side are:

- Correctness – how released modules are working
- Deliver time – are the modules that are released on schedule?
- Quality – does there exist or have been detected errors (bugs) on the released modules?

The evaluation metrics on the student's side are:

- Learning path – how many errors were detected during the coding phases?
- Do these errors have been minimized?
- Have somebody optimized code from another team?
- Personal experiences about pair programming.
- Measure performance when working on modules that have the same grade of complexity

6. Conclusions and future work

XP is a methodology that can be applied to projects with no more than 15 programmers. Among his advantages is the adaptation to priority changes and functional requirements. Hence the client sees the working product, in consequence he can change some needs without the complete reworking of the analysis and design. The other side is that to implement these techniques, special work is needed. Even more, is completely necessary that the client knows all about the project. If is no that the situation, the project could fail.

The first part of the SET Tool project was the understanding of the whole system, and the development of specific code. In these stages of the development, the students learned and worked with the modified XP practices. Hence, some technical aspects of the system, such as development platforms, database configurations, operating systems among others, were investigated by the sophomore students. This research gives them a better vision in the developing industry. At this stage of the project, XP has aid the students to acquire self study skills, and to develop efficient code.

The SET Tool is planned to be released late on this year. During the next months, more *stories* will be released and assigned. This project will be evaluated with the proposed metrics described above. A beta version also will be evaluated by the client, and with his feedback, improves will be made (management and academic).

References

Beck, Kent. "Embracing Change with Extreme Programming", *IEEE Computer*, 1999.

Class, Robert L. "Extreme Programming: the good, the bad, and the bottom line", IEEE *Software*. 2001 IEEE.

Duncan, Richard. "The Quality of Requirements in Extreme Programming". Available from http://www.isip.msstate.edu/publications/journals/crosstalk/2001/extreme_programming/crosstalk_final.pdf; Internet; accessed on December 10, 2003.

Jensen, Randall W. Software Technology Support Center, "A Pair Programming Experience". Disponible en **http://www.stsc.hill.af.mil/crosstalk/2003/03/jensen.html** , accessed 16 December 2003.

Jovanovic, Vladan,. Murphy, Thomas,. and Greca, Ardian. “Use of extreme programming (XP) in teaching introductory programming”, *32nd ASEE/IEEE Frontiers in Education Conference(2002).*

Kobayashi, Atsushi and Maekawa, Mamoru. “Need-Based Requirements Change Management” *IEEE Computer*, 2001 IEEE.

Lam, W. and Shankararaman,V. “Requirements Change: A Dissection of Management Issues” Department of Computer Science, University of Hertfordshire 1999.

Maurer, Frank and Martel, Sebastien. “Extreme Programming – Rapid Development for Web-Based Applications”, *IEEE Internet Computing*, January – February 2002.

NawRocki, Jerzy,. Jasinksi,Michal,. Walter, Bartosz,. and Wojciechowski, Adam. “Extreme Programming Modified, Embrace Requirements Engineering Practices”, *Proceedings of the IEEE Joint International Conference on Requirements Engineering (RE’ 02).*

O’Neal, James S. and Carver, Doris L.. “Analyzing the Impact of Changing Requirements” Nov. 2001.

Paul, Mark C. “Extreme Programming from CMM Perspective”, *Paper for XP Universe, Raleigh*, NC, 23 – 25 July 2001.

Schneider, Jean-Guy and Johnston ,Lorraine. “eXtreme Programming at Universities – An Educational Perspective” *Proceedings of the 25th International Conference on Software Engineering (ICSE’03).*

Schneidewind, Norman F. “Investigation of the Risk to Software Reliability and Maintainability of Requirements Changes” Nov 2001.

Vicerrectoría Académica del ITESM, “A El Modelo de Enseñanza-Aprendizaje del Tec de Monterrey”. Disponible en **http://www.sistema.itesm.mx/va/dide/modelo/home.htm**, Internet, accessed 16 December 2003.

Wagner, Larry. “Extreme Requirements Engineering”, *Cutter IT Journal* Vol.14, No. 12, December 2001.

Williams, Laure and UpChurch, Richard. “Extreme Programming for software engineering education?” *31st ASEE/IEEE Frontiers in Education Conference(2001).*