Extreme Programming and Its Positive Affect on Software Engineering Teams

Sharifah Lailee Syed-Abdullah[a], Mazni Omar[b]

Department of Computer Science
Universiti Teknologi MARA, Arau Campus, 02600 Arau, Perlis, MALAYSIA
Tel : 04-9875076, Fax : 04-9874225
E-mail : [a]shlailee@perlis.uitm.edu.my, [b]mazni@isiswa.uitm.edu.my

ABSTRACT

This paper presents an early empirical study on Extreme Programming (XP) practices employing Positive Affect metric. The study was conducted on university students doing development projects to gain an insight understanding of the effect of using agile practices on software engineering (SE) teams. The finding indicates that XP practices do have positive affectivity on the SE teams. This is to be expected because of the existence of the practices such as simple design, pair programming, continuous testing, continuous integration and frequent review (release) that command feedback. This finding helps to provide early empirical evidences on the impact of XP methodology on the positive affectivity of the developers.

Keywords
Agile methodology, empirical study, XP, positive affect, SE team

1.0 INTRODUCTION

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

As a reaction to these problems, a new group of methodologies evolved, these are known as **agile methodologies.** Agile methodologies welcome change and unpredictability. These new methodologies are more adaptive than predictive, and more people-oriented than process-oriented. Adaptive approaches are better when the requirements are uncertain or volatile as in the new software being developed nowadays. If the user requirements are not stable, it is difficult to develop stable designs and follow a planned process as practised in the formal methodologies. When faced with unpredictable user requirements and changes that must be accommodated during software-in-progress development, developers often experienced stressful emotions such as anxiety and depression (Syed-Abdullah, Holcombe, & Gheorge, 2006a, 2006b).

It is the intention of this paper to introduce the most prevalent agile methodology: Extreme Programming (XP) as the answer to the existing problems faced by developers when designing and creating dynamic software applications. The second part of this paper will explore the possibility of using XP methodology as a positive affect inducer and to discuss the findings of a study on the possible impact of the selected XP practices on the positive affectivity of the SE teams. To achieve this, a comparison study was conducted on the SE teams consisted of third year students at University Utara Malaysia. Findings revealed that the XP methodology does have an impact on the positive affectivity when most of the practices were implemented.

2.0 EXTREME PROGRAMMING (XP)

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

In XP, features that provide the most business value to the customer must be developed first because the real goal of this approach is to deliver the software that is needed when it is needed. Requirements are written as user stories, which are chunks of functionality that are valuable to the customers. Chunking is a technique in cognitive learning strategy that allows information to be broken down into smaller and meaningful collection of knowledge. Through the use of story cards, it is easier for developers to group different stories according to main functions. Chunking assist developers because human has limited memory capacity and often have difficulty to memorize a large amount of functions or information (Mazni, Syed-Abdullah, & Holcombe, 2009).

The **communication** between developer and manager, which can be lacking in other methods, is highlighted as one of the main values, which must be emphasized. XP encourages communication by having the developers collectively owning all of the code and work in pairs. **Collective code ownership** considers that the code belongs to the team and not to the individual developers. It encourages every developer to contribute new ideas to all segments of the project and allows any developer in the team to add functionality, correct errors or refactor the code. Pair programming is a practice that requires two developers to sit side by side in front of a computer. One person types and thinks tactically about the methods being created, while the other thinks strategically about how the methods fit into the class. Each partner must explain what they are doing and this encourages the development of new ideas and an improvement on the old approaches. To facilitate a collective code ownership and continuous integration, pair programmers must swap partners amongst the team. Pair programming changes the environment from criticism and competition to learning and cooperation thus improving group cohesiveness and communication.

The next value is **simplicity**. Complex requirements must be simplified to enhance understanding between team members. XP simple design evolves through constant refactoring, which is guided by suitable metaphor and implemented in accordance to common coding standards. Fowler, Beck, Brant, Opdyke, and Roberts, 1999 defined refactoring as the removal of redundant or unused functionality and the restructuring of obsolete designs in order to improve smelly codes. Tong (2004) considered too many comments as useless because it takes too many visual spaces and further suggested refactoring of these comments by converting them into codes. System metaphor is a narrative that everyone (customer, programmer and managers) can associate with when discussing new functionality. The reason for using a metaphor is to achieve a common vision and shared vocabulary.

On-site customer is a practice which requires the customer to sit with the development team on a full-time basis. It is the customer's duty to assist in the writing of stories, to answer questions and to set priorities to the project. Holcombe (2002) is more realistic in this practice, by balancing between having customer on site full time, with one hardly there at all; he proposes regular visits and meetings at both the development site and the business site. In reality, not all customers could afford to adhere to on-site customer practice. This is not because they are not serious but it is due to other managerial issues.

The humanistic aspect of the communication and the simplicity aspect promote good teamwork because it is an important ingredient towards developing quality software. A stable teamwork will facilitate continuous testing, which will enhance **courage** because members are confident of producing better and well-tested software. The satisfaction of producing quality software is very important because it will boost the confidence of the team to produce more challenging software.

3.0 THE POSITIVE AFFECT OF THE XP PRACTICES

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

In this study, XP methodology was chosen as a positive inducer because of the existence of several XP practices that warrant feedback to the developers. Positive feedback about one's performance has been known as a positive affect inducer (Estrada, Isen, & Young, 1997; Syed-Abdullah, Holcombe, & Gheorge. 2006a). The accumulating evidences suggest positive affect may predict increased attention to information and a more careful, thorough processing of any information when the information is self-relevant or important. The XP practices associated with feedback seeking are simple design, pair programming, continuous testing, continuous integration and frequent review (release).

Studies by Aspinwall and Muraven et al noted that people must have a surplus of resources such as time, energy and attention to engage in a proactive behaviour. Using XP approach, the developers experienced a surplus of time during coding because less time was engaged in the designing phase. Using simplified design as an alternative to cumbersome steps in formal design, XP developers are actually releasing the stressful task of creation, thus liberating the mind to be more creative and innovative. By reducing the technical aspect of the design, the mind was able to approach the

problem solving task through a breadth first approach. Design is only an early manifestation of ideas, whereas the coding process allows the developers to realize their idea in a more concrete way. This approach is considered as a positive affect inducer because it allows feedback on the design through the programming code. The ability to see the advantages and identify the flaws in the design allows the developers to be more creative in the next part of the system. This is the reason why simple design can accommodate a flexible requirement because the process of creating part of the system in this manner allows the developers to be more innovative in the problem solving process.

It was observed that the practice of pair programming started with the initial socializing amongst the pair thus creating a positive mood amongst them before any formal programming commenced. The positive mood which is experienced and the attention of the two developers allow the pair to engage in a more proactive behaviour. The ability to discuss the advantages and disadvantages of certain coding ideas enables the pair to seek improvements and to avoid specific weaknesses. Even though pair programming was not a favourite practice, because it was perceived as difficult due to being time consuming and at a different level of programming experience, nevertheless, at the end of the project, the members often acknowledge that their creative ideas were explored much more during this process. Studies on pair programming have provided the evidence about the benefits of pair programming . With pair programming practice, positive affect is induced through early socializing, more attention and immediate feedback amongst the pair.

Continuous testing allows feedback on the developed code. In the normal software testing domain, testing is usually left at the end of the development cycle thus leaving a very short time for complete testing. In this situation, often the developers were faced with products that have too many defects, as the bugs were discovered too late. The benefit of testing as the software is developed is that the developers are always certain that the software developed is always test compliant. Continuous testing is a practice that is structured so that different levels of testing can be conducted as the solution is being built. In the study by Trope and Pomerantz (1998), participants in whom positive affect has been induced showed greater interest in the part of the test they had failed than did neutral mood participants. The emphasis of the continuous testing enables the developers to feel more confident about the correctness of the code and therefore bolster their confidence and self-esteem.

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

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

The above theoretical study of XP practices identified these practices as being a positive affect inducer. A replicated study in UUM were carried out to determine empirically, whether teams using these practices would experience higher positive affectivity than the teams using the design-based Rational Unified Process (RUP) approach.

H_1: The Agile (XP) team will experience a higher level of positive affectivity than the Formal (RUP) team at the end of the project.

To test this hypothesis, a comparison study was carried out on the third year software engineering students in 2008.

3.1 Comparison Study (UUM 2008)

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

The validity and reliability of PANAS scale has been demonstrated by other studies (Watson & Tellegen, 1985; Watson, Pennebaker, & Folger, 1987; Watson & Clerk, 1997).The Positive Affect scale showed a satisfactory internal consistency coefficient, Cronbach alpha = 0.78 during the first reading (Week 2), Cronbach alpha = 0.89 during the second reading (Week 6) and Cronbach alpha = 0.87 during the third reading (Week 15). At the beginning of the study, **Independent sample t-test** was used to compare the total mean score for Positive Affect variables and the result showed **no significant difference** between Formal teams

[N=28 , Mean Score (M) =34.12, Standard Deviation (SD)=4.77] and Agile (XP) teams [N=30, Mean Score (M) =34.53, Standard Deviation (SD)=4.41].

In order to test the hypothesis, **Mixed between-within ANOVA** was conducted. It was indicated that there is **no significant difference** between the three intervals; Reading 1 (Week 2), Reading 2 (Week 6) and Reading 3 (Week 15) for both methodologies; Formal (N=28, M_1= 34.12 SD_1 = 4.77; N=28 M_2 =33.61, SD_2 = 5.07; N=28 M_3 =33.86, SD_3 = 4.57) and Agile (N=30, M_1=34.53, SD_1 = 4.41; N=30 M_2 = 34.17, SD_2 =5.90; N=30 M_3 =35.37, SD_3 = 6.37) (see Figure 1 and Table 1). This may due to the small effect size (eta squared =0.010). Besides, the results may be moderated by others factors such as partial adoption of Agile (XP) practices during this study. This finding supported earlier finding on the positive effect of Extreme Programming on SE teams (Syed-Abdullah et al. 2005).

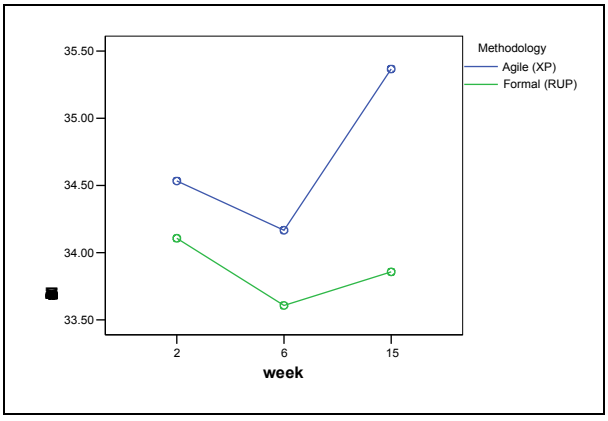

Figure 1: Line graph of the positive affect of the two teams before treatment (week 2) and after treatments (week 6 and week15) [UUM 2008]

Table 1: Descriptive Statistics of Positive Affectivity of Both Teams (UUM 2008)

Method	N	M_1	SD_1	M_2	SD_2	M_3	SD_3
Formal (RUP)	28	34.12	4.77	33.61	5.07	33.86	4.57
Agile(XP)	30	34.53	4.41	34.17	5.90	35.37	6.37

Note: $p < 0.05$

At the end of Week 15, the systems were graded by teams of evaluators consisted of the project client and one lecturer for each team. The mean scores awarded by both evaluators were assessed. The following graph shows marks achieved by both teams according to the projects.

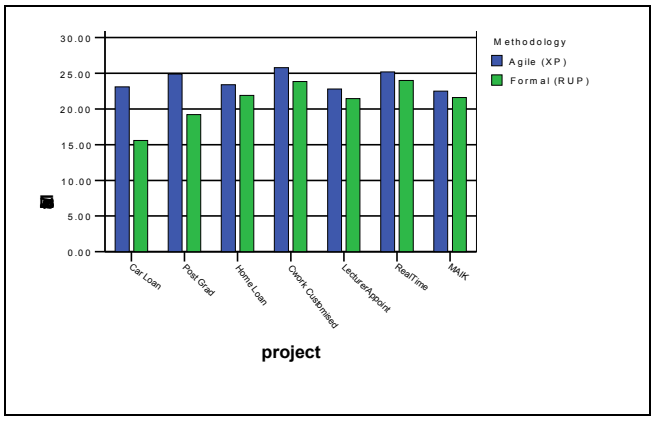

Figure 2: Bar graph showing Teams Performance according To Project (UUM 2008)

The Mann-Whitney non parametric statistical test was used to compare the mean scores and the results showed **significant difference** in the mean scores for the Formal teams [M=21.09, SD=2.91] and the Agile (XP) teams [M=23.96, SD=1.31] The graph indicates that teams using Agile (XP) approach were awarded higher score than Formal (RUP) teams.

4.0 DISCUSSION

In this study, even though there was no significant difference in positive affect between the methodologies, it is interesting to observe the impact of the XP on the positive affectivity of the developers which result in higher score being awarded to the agile teams. The teams using a more flexible approach, such as the XP methodology, were able to incorporate the constant changes made by the clients and thus able increase their positive mood.

When a person experiences a positive affect, they show a greater preference for a larger variety of actions and are able to see and think of more possibilities and options to solve whatever problem is faced. People with a positive affect are more likely to take action because they are proactive. This study suggests that when people experience joy and mild contentment, they are more likely to think of a wider range of actions, become more resilient over time and are more likely to develop long-term plans and goals.

ACKNOWLEDGEMENT

The authors would like to express our appreciation to Software Engineering Project 1 course lecturers, from Universiti Utara Malaysia, Dr Azman Yasin and Dr Haslina Mohamad who have given their support towards the completion of this research. In addition, we would like to thanks all clients, supervisors and students in this study.

REFERENCES

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

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

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

Beck, K. (2000). *Extreme Programming Explained: Embrace Change*. USA: Addison-Wesley.

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

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

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

M. Fowler, K. Beck, J. Brant, W. Opdyke, & D. Roberts. (1999). *Refactoring: Improving the Design of Existing Code.* Addison Wesley.

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

Isen, A. M. (2001). An influence of positive affect on decision making in complex situations: Theoretical Issues with practical implications. *Journal of Consumer Psychology*, *11*(2), 75-85.

Mazni, O., Syed-Abdullah, S. L., & Holcombe, M. (2009). Being Agile in Classroom: An Improvement to Learning Programming. In *CD Proceedings of National ICT in Education Seminar* (February 3-4, 2009). Ipoh, Malaysia.

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

Succi, G., Marchesi, M., Pedrycz, W., & Williams, L. (2002). *Preliminary Analysis of the Effect of Pair Programming on Job Satisfaction*. 3rd International Conference on Extreme Programming and Flexible Processes in Software Engineering (XP2002), Sardinia, Italy, 212-214.

Syed-Abdullah, S.L., Holcombe, M., Karn, J., Cowling, T., & Gheorge, M. (2005). The Positive Affect of the XP methodology. *Lecture Notes in Computer Science, LNCS, 3556*. (pp. 218-221). Berlin Heidelberg: Springer-Verlag.

Syed-Abdullah, S. L., Holcombe, M., & Gheorge, M. (2006a). *Extreme Programming and its ability to improve the creativity and innovative of SE teams*. Paper presented at the Proceedings of National ICT Conference, UiTM Perlis, Malaysia.

Syed-Abdullah, S., Holcombe, M., & Gheorge, M. (2006b). The Impact of an Agile Methodology on the Well Being of Development Teams. *Empirical Software Engineering, 11*(1), 143-167. doi: 10.1007/s10664-006-5968-5

Tong, K. L. (2004). *Essentials Skills for Agile Development*. Macau Productivity & Tech. Retrieved from http://www.agileskills.org/pdf/ESAD.pdf

Trope, Y., & Neter, E. (1994). Reconciling competing motives in self-evaluation: The role of self-control in feedback seeking. *Journal of Personality and Social Psychology, 66*, 646-657.

Trope, Y., & Pomerantz, E.M. (1998). Resolving Conflicts Among Self-Evaluative Motives: Positive Experiences as a Resource for Overcoming Defensiveness. *Motivation and Emotion, 22(1)*, 53-72.

Watson, D., & Tellegen, A. (1985). Towards a consensual structure of mood, *Psychological Bulletin, 98*, 219-235.

Watson, D., Pennebaker, J. W., & Folger, R. (1987). Beyond negative affectivity: Measuring stress and satisfaction in the workforce. *Journal of Organizational Behavior Management, 8*(2), 141-157

Watson, D., & Clark, L. A. (1997). Extraversion and its positive emotional core. In R. Hogan and J.A. Johnson (Eds.), *Handbook of Personality Psychology,* Academic Press, San Diego, Ca., 767-793

Williams, L. (2002). Pair Programming: Why Have Two Do the Work of One? In *Extreme Programming Perspectives,* M. Marchesi, G. Succi, D. Wells and L. Willaims (eds.), Addison-Wesley, Boston, pp. 23-33.