Journal of Information Engineering and Applications
ISSN 2224-5782 (print) ISSN 2225-0506 (online)
Vol.8, No.9, 2018
www.iiste.org

A Structured Approach to Analyze XP Model and Eliminate Drawbacks for Qualitative Development of Software

Hassan Iftikhar
Department of Computer Science, University of Agriculture Faisalabad, Pakistan

Wajeeha Azmat
Department of Computer Science, University of Agriculture Faisalabad, Pakistan

Abstract
In this modern age each software company choose a software model for software development. But this companies face many problems to use software model because each software model has some drawbacks. These drawbacks effected the software development. In previous research many software methodologies were adopted for development. But it a question yet how to select a correct software model which has no drawbacks. In this research XP (Extreme Programming) will be analyzed. This model has some strengthens and weaknesses. The conclusion of this research will to eliminate drawbacks of XP (Extreme Programming). This can be done through analyze different parameters and converse with software developers for parameters to eliminate drawbacks.
Keywords: Software Development, Extreme Programming (XP), Eliminate Drawbacks.

1. Introduction

Software projects developed in past and also today with modern features. In past due to limitation of technology, software were failed. Today different agile methodologies are used in software development. By using agile methodology, every module of software is authenticated by clients and also meet the requirements of development environment. (Kaur and Sengupta, 2011).

1.1 Software Development Process

1.1.1 Waterfall Model

Waterfall model for software development involves requirement analysis, external design, internal design, implement, test, operation and maintenance. The main problem in waterfall model is that, requirements are gathered in first step of model and during the steps of internal and external design it is necessary to go back to requirement analysis if client wants to change in requirements after testing desired system. But in waterfall model, system is handed over to client after completing the project.

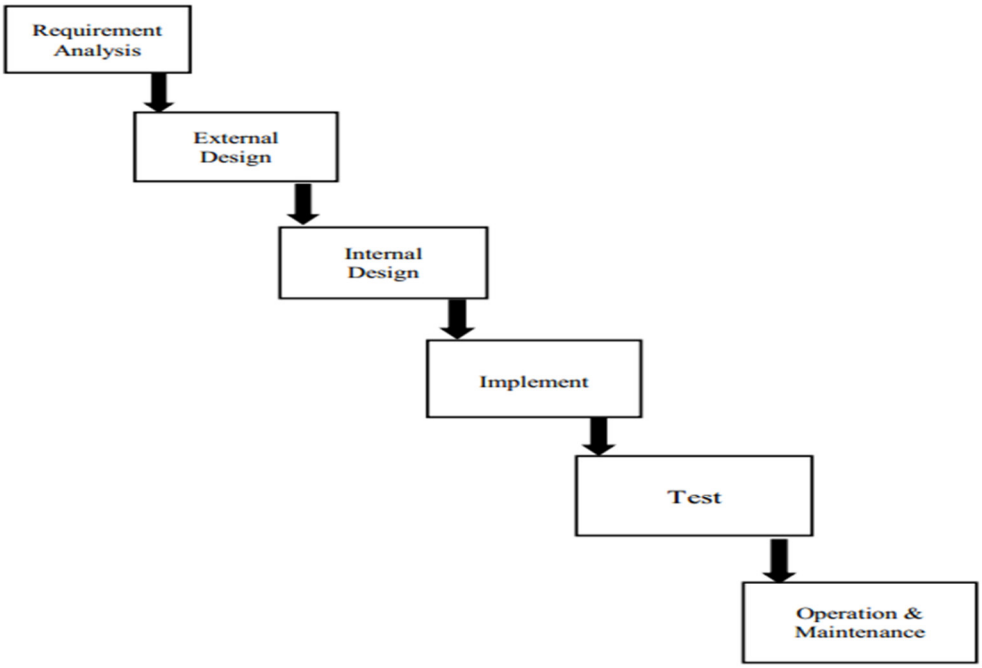

Figure 1. Waterfall Process model

1.1.2 Extreme Programming

Extreme Programming is a light weight process model to develop software in an easiest way. Extreme

Programming converts the complex problems in small modules. It is necessary to create small modules because small modules can easily be developed. Today maximum software companies use Extreme Programming model for better quality of software. (Abrahamsson, 2013). Extreme Programming steps involved are planning, analysis, design, implementation, maintenance. In extreme programming, complex problem is divided into smaller modules. Development of every module involves all steps of extreme programming and at the end every module is validated by client after completion of module. If client wants to any change in requirement of module then these changes can easily be done by programmers.

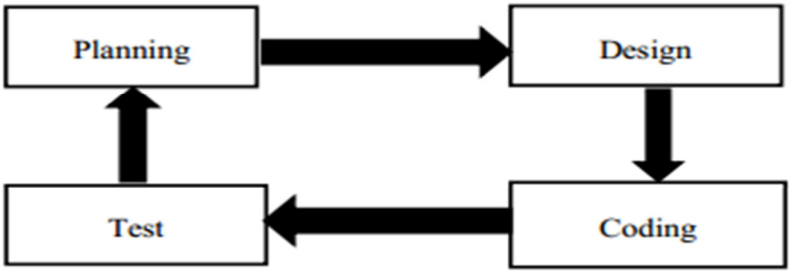

Figure 2. Extreme Programming (XP) Process Model

1.2 Problem Statement

Extreme Programming (XP) model is widely used in software development. But this model has some drawbacks as well as strengthens. These drawbacks effect the progress of software development.Extreme Programming (XP) is an agile methodology so it is necessary to deliver the software to client at time and client should be satisfied from development. But drawbacks of Extreme Programming (XP) model are hurdles in efficient software development.

1.3 Objectives

The main objective of this research to eliminate drawbacks of Extreme Programming (XP) model because it is necessary for qualitative development of software. By eliminating drawbacks, it will easy to create software in effective software development.

2. Literature Review

Waterfall model was used in large projects but software was developed using this model validated by client at the end of project. Extreme Programming was used for small projects. Because in extreme programming small modules were validated by client for any change in requirements. Software were developed in waterfall model were not quality wise good. Therefore extreme programming was used. A method EPISODE (Extreme Programming method for Innovative Software based on systems Design) was used to grouping the different parts to make an effective software (Takaaki *et al*., 2014). In past years, projects were developed using waterfall model. But it was a big challenge to develop a complex project. Because when complex project was developed using waterfall model then clients were not involved in project. The project was handed over to client at the end of phase. If there were errors in project at the end then it was difficult to remove errors. So Extreme Programming (XP) was used to handle complex projects. Extreme Programming was an agile methodology. A complex problem was divided into several modules. Client was completely involved in development and every module was validated by client after completion (Sharma, 2016). Indonesia wanted to develop financial management system for public sector. Government had three levels. Local government was called high level. Local level had many departments so these departments were called middle level and users were called low level. Local government and departments were satisfied by system because extreme programming was used in this development. The main issue was users had no awareness about this system. So all users were involved in this development and system was also validated by users. So they could easily understand the system using extreme programming. (Haryono, 2015). XP (Extreme Programming) was used with existing methods to develop an effective software. In Taiwan, 4 students were developed object oriented grade system. The development of this project was complex. This project used java, data structure and algorithm. So extreme programming was integrated with these steps. After completing each module this software was validated with all students who were using this software. By using extreme programming, object oriented grade system was working in an effective manner. (Yen-chen, 2015).

3. Research Methodology

There were some following parameters which were drawbacks effected the working of XP (Extreme Programming) model.

3.1 Requirement Analysis

Requirement analysis was an important factor in XP (Extreme Programming) model. In XP (Extreme Programming). When client gave all requirements then it was not necessary that there was no need of extra meeting. The main issue was if client wanted to change in requirements of project then how to deal these changes. Because in agile methodology it was necessary to satisfy the client. So if client wanted changes then these changes should be done in requirements. It was necessary to analyze all changes in requirements to integrate in project.

3.2 Reliance on Refracting and unit testing

Refracting was to restructure the existing code to improve the quality of software. But in XP model, it took a lot of time to restructure the code. In large projects, programmers had no extra time. So if they restructured the code then it took a lot of time.

3.3 Elimination of bugs

Elimination of bugs were flaws, errors in coding or designing of project. It was necessary to eliminate the bugs of project for effective development. In XP (Extreme Programming) small modules were handed over to client after completion. So if there were errors in module then how to resolve it. Were these errors should be corrected at the end of project or after completing each module. This was a major issue in XP (Extreme Programming).

3.4 Define Process

In XP (Extreme Programming), there was continuous developing of software. Complex problem was divided into several smaller modules and all modules were assigned to different developers. For this purpose it was necessary to define process which included how many developers should be involved in project? Which module should be assigned to which developer? Was it necessary to calculate the complexity of each module?

3.5 Time and Cost

Time and cost were important factors in XP (Extreme Programming). It was necessary to calculate time to develop each module and also cost for each module. The main factor was, was it important to consider time and cost factor in XP.

To analyze these requirements, a survey was conducted from software practitioners. For this purpose a questionnaire was design and interviewed the software practitioners. Then anova test was applied to get statistical result.

4 Results

4.1 Analysis of Requirements

Total 50 responses were recorded and 30 people show that analysis of requirements are important at each phase of XP model. 20 People show that analysis of requirements are not important at each phase of XP model.

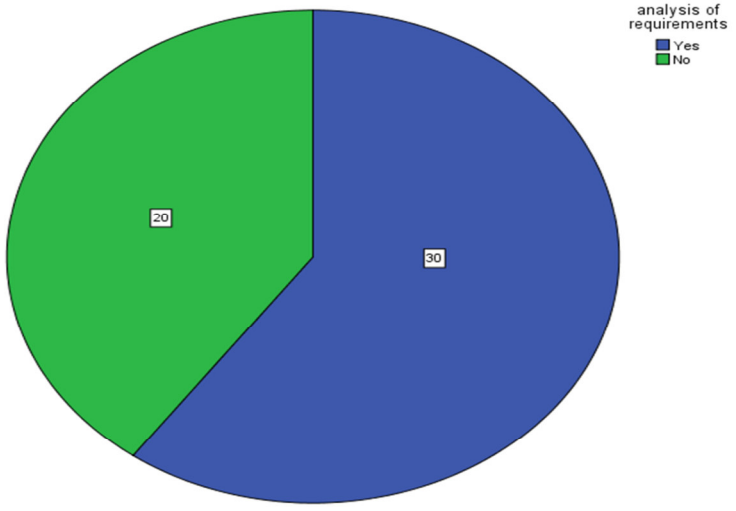

Figure 3. Analysis of Requirements in Extreme Programming

H0: Analysis of requirements at each phase

H1: Analysis of requirements is not important

Table 1. Anova Test for Analysis of Requirements in Extreme Programming

ANOVA					
analysis of requirements					
	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	.798	5	.160	.627	.680
Within Groups	11.202	44	.255		
Total	12.000	49			

$F_{cal} < F_{sig}$

$0.627 < 0.680$

So null hypothesis is not rejected. These results show that analysis of requirements are important at each phase of XP (Extreme Programming) for qualitative development of software.

4.2 Reliance on refracting and unit testing

Total 50 responses were recorded and 36 people showed that it was necessary to restructure the code and unit testing in each module of software and about 14 people showed that it was not necessary to restructure and unit testing in development of each module.

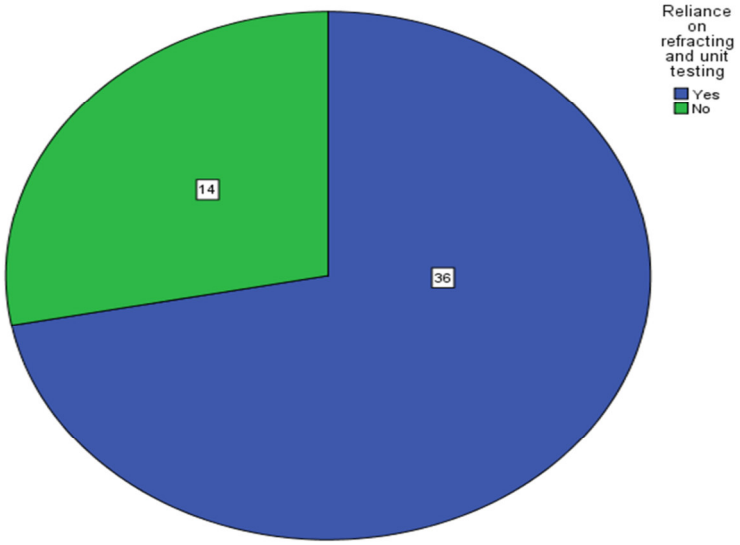

Figure 4. Reliance on Refracting and Unit Testing in Extreme Programming

H0: Extreme programming should reliance on refracting and unit testing.

H1: Extreme programming should not reliance on refracting and unit testing.

Table 2. Anova Test for Reliance on Refracting and Unit testing in Extreme Programming

ANOVA					
Reliance on refracting and unit testing					
	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	.859	5	.172	.819	.543
Within Groups	9.221	44	.210		
Total	10.080	49			

$F_{cal} > F_{sig}$

$0.819 > 0.543$

So null hypothesis is not accepted. These results show that it is not necessary to restructure the existing code because it is time a lot of time consuming and there may errors in restructuring the existing code.

4.3 Elimination of bugs

Total 50 responses were recorded and 23 people showed that elimination of bugs must be necessary at each phase of extreme programming (XP). About 9 people showed that elimination of bugs should be done in small projects. But in complex projects, it was not necessary to eliminate bugs at each phase. About 18 people showed that it was not necessary to eliminate bugs at each phase.

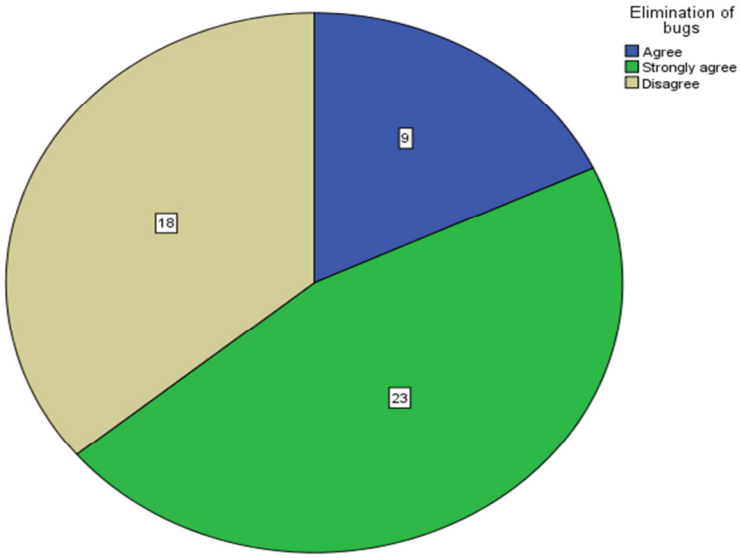

Figure 6. Elimination of Bugs in Extreme Programming

H0: Elimination of bugs are important in extreme programming.
H1: Elimination of bugs are not important in extreme programming.

Table 3. Anova Test for Elimination of Bugs in Extreme Programming

ANOVA					
Elimination of bugs					
	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	1.725	5	.345	.642	.669
Within Groups	23.655	44	.538		
Total	25.380	49			

$F_{cal} < F_{sig}$

$0.642 < 0.669$

So null hypothesis is not rejected. These results show that it is necessary to eliminate all bugs at each phase and in each module for effective development of software.

4.4 Definition of process

Total 50 responses were recorded and 40 people showed that definition of process was important at start of project and 10 people showed that definition of process was not important at start of project.

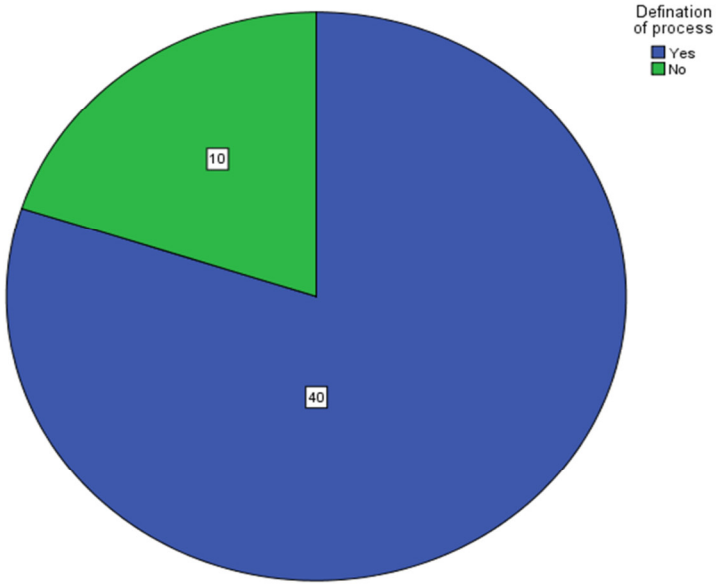

Figure 7. Definition of Process in Extreme Programming

H0: Definition of process has importance in extreme programming.
H1: Definition of process has not importance in extreme programming.

Table 4. Anova Test for Definition of Process in Extreme Programming

ANOVA					
Defination of process					
	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	.364	5	.073	.420	.832
Within Groups	7.636	44	.174		
Total	8.000	49			

$F_{cal} < F_{sig}$

$0.420 < 0.832$

So null hypothesis is not rejected. These results show that it is necessary to define process before start developing a module of software to analyze the time and cost in which module would be completed.

4.5 Time and Cost

Total 50 responses were recorded and 32 people showed that it was necessary to consider time and cost in using XP model and 18 people showed that it was not necessary to consider time and cost in XP model.

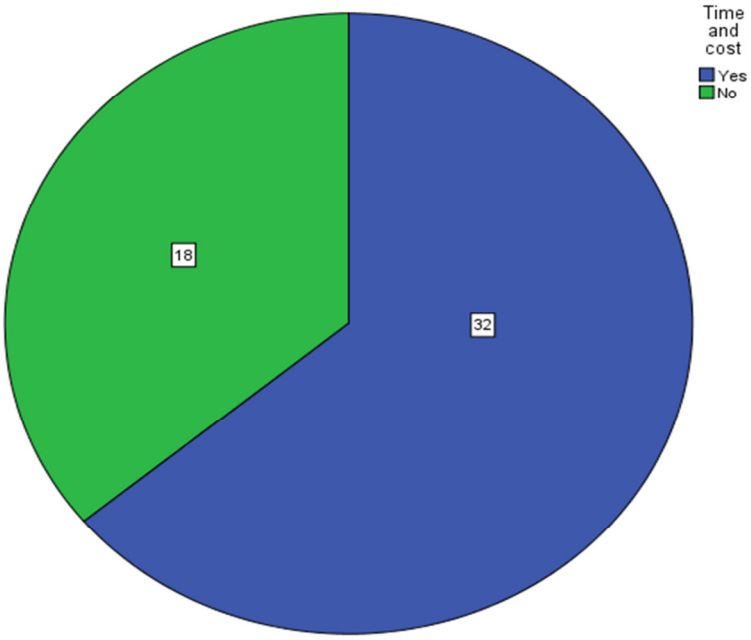

Figure 8. Time and Cost in Extreme Programming

H0: Time and cost are important in extreme programming.

H1: Time and cost are not important in extreme programming.

Table 5. Anova Test for time and Cost in Extreme Programming

ANOVA					
Time and cost					
	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	.453	5	.091	.360	.873
Within Groups	11.067	44	.252		
Total	11.520	49			

$F_{cal} < F_{sig}$

$0.360 < 0.873$

So null hypothesis is not rejected. These results show that it is necessary to calculate time and cost for developing each module to complete the project at time and budget.

5. Conclusion

The agile methodology, XP (Extreme Programming) has strengthens therefore this model is widely used in software development. There are some parameters discus in methodology, these parameters are drawbacks which effect the XP (Extreme Programming) model. So it is necessary to analyze these parameters for qualitative development of software. Because it is necessary to satisfy the client by completing the project at a time with good quality. So parameters should be analyzed and apply changes in XP (Extreme Programming) according to these parameters for effective development of software.

References

Carvalho. W., P. Rosa, and M. Soares. 2011. A comparative Analysis of the Agile and Traditional Software Development Processes Productivity.*30th International Conference of the Chilean Computer Science Society*: 74-82.

Abrahamsson. D. 2013. Extreme programming: first results from a controlled case study. *In Proceedings of the 20th IEEE Instrumentation Technology Conference, EURMIC-03*: 259–266.

Rao. V., V. Krishna, and K. Rao. 2013. Rational Unified Process for Service Oriented Application in Extreme Programming. *4th ICCNT Institute of Electrical and Electronic Engineers.*

Alshehri. S., L. Benedicenti. 2013. Prioritizing CRC Cards as a Simple Design Tool in Extreme Programming. *26th IEEE Canadian Conference of Electrical and Computer Engineering (CCECE).*

Matharu. G. S., A. Mishra, and P. Upadhyay. 2013. Empirical Study of Agile Software Development Methodologies. *ACM SIGSOFT Software Engineering Notes,* 40(1):pp. 1-6.

+Rao. G., V. Krishna, and K. Rao. 2014. Extreme Programming for Service-Based Application Development Architecture. *Institute of Electrical and Electronic Engineers.*

Goto. T., K. Tsuchida, and T. Nishino. 2014. EPISODE: An Extreme Programming Method for Innovative Software Based on Systems Design. *IIAI 3rd International Conference on Advanced Applied Informatics*: 780-784.

Akpolat. B., W. Slany. 2014. EPISODE: Enhancing Software Engineering Student Team Engagement in aHigh-Intensity Extreme Programming Course using Gamification. *Institute of Electrical and Electronic Engineers*: 149-153.

Chen. J., M. Wu. 2015. Integrating Extreme Programming with Software Engineering Education. *MIPRO, Opatija, Croatia*: 577-582.

Sharma. P., N. Hasteer. 2016. Analysis of Linear Sequential and Extreme Programming Development Methodology for a Gaming Application. *International Conference on Communication and Signal Processin*g: 1916-1920