Journal of Information & Communication Technology
Vol. 5, No. 2, (Fall 2011) 61-73

Agile Software Engineering, a proposed extension for in-house software development

Muhammad Misbahuddin *
Institute of Business & Technology (IBT), Pakistan

M. Ansari *
Muhammad Ali Jinnah University, Pakistan

ABSTRACT

Software maintenance is one of the major part of the software development life cycle and it cannot be detached from other software development life cycle process. One of the major concerns of any software maintenance organization is to analyze, estimate and improve the productivity of maintenance process. In this paper, both qualitative and quantitative analysis was done through incremental experiments to improve the maintenance and evolution process for a particular environment.

The purpose of this research is to present the result of introducing and evaluating agile process based on Extreme Programming (XP) in a software maintenance environment for an in house software development team. Prior to introducing an agile process, an assessment of the organization's software maintenance and evolution processes was done, to understand what should be introduced and why. It was difficult to introduce XP in it original appearance to the case organization, so it is necessary to redesign many of the practices in order for them to fit the needs of the software maintenance team. After, observing and examine the existing maintenance process and related issues of the particular case organization, some agile practices was selected and redesigned to improve the overall quality and performance of the particular environment.

This research successfully proved that adoption and modification of agile practices in particular environment improve the overall quality of maintenance process and overcome the addressed issues of in-house development environment.

INSPEC Classification : C60, C61, C72, D10

Keywords : Agile methods, Extreme programming, Software development, In-House software development, Extension in agile software development, Agile software maintenance.

1. INTRODUCTION

This research presents an easy and simple approach in the extension of agile method for environment software maintenance and evolution process in the context of in-house development, referred to as Extension in Agile Software Engineering (EASE) - A software

*The material presented by the authors does not necessarily portray the viewpoint of the editors and the management of the Institute of Business and Technology (IBT) or Muhammad Ali Jinnah University, Pakistan.

* Muhammad. Misbahuddin : m.misbah@biztekian.com
* M. Ansari: m.ansari.maju@gmail.com.

maintenance framework for in-house development environment. The proposition of EASE has been evidenced from an experimental analysis done in a case organization having a large enterprise system.

It is proved that software maintenance take more cost and time than software development. Software maintenance is generally recognized to consume the majority of resources in many organizations. Software-developing organizations have investigated agile methods as a source for software process improvement. It has been found that agile methods, with most focus on Extreme Programming (XP) (Kent, B, 2000), provide an effective process that can resul in high quality products. There are many different agile process models for software development, in this research the XP method was selected to experiment because researchers proved that XP methods are better suited for software maintenance than a traditional waterfall model. Software maintenance based on non-traditional method promotes customer collaboration to elicit proper software maintenance needs, production of frequent software releases to deliver capability to maintenance customer sooner, collaboration with software maintenance team to leverage contextually rich maintenance communication, and the flexibility to respond to the changing business requirements (David, F, 2007)

There are numerous researches available on software development process but a limited number of researches in software maintenance and evolution process. The maintenance process is all about "Change" and agile methods are effectively welcome changes. This is the fundamental motivation for this research that' way it was analyzed that agile methods should be examined to improve software maintenance process.

2. PROBLEM CONTEXT AND RESEARCH OBJECTIVES

The main domain of the problems for this research was maintenance and evolution process related issues and challenges for in-house development environment. This research examined that how to improve quality, productivity and reliability of particular environment by adopting and reformulating agile methods. We aimed our research to address the problems and set our objectives as 1) understand the process of software maintenance and the benefits of adopting agile practices in software maintenance organization through literature review, 2) asses the issues and challenges of maintenance process for in-house development environment, 3) incrementally experiment and evaluate agile practices, reformulate these practices to get the better results in terms of improvement, analyze the result of implementing agile practices for in-house development organization and proposed our recommendations for improvement of software maintenance activities

In order to complete our research, we set the scope of this research into two parts qualitative study and quantitative experiments. In first part, we reviewed and analyzed the existing literature and observed current maintenance issues and challenges. In second part, we examined the different agile practices in case organization. Second part of the research is also divided into four phases. In Phase-I, assessed the existing maintenance process on the basis of data collected during maintenance releases. Phase-II introduced agile practices as their original behavior into maintenance process of case organization. we also analyzed the result of agile practices implementation. Phase-III proposed some enhancement into original agile practices with respect to the particular environment and try to achieve the desire results also analyzed the result of modified changes in agile practices. Finally "Phase-IV" proposed framework for software maintenance in the context of in-house development environment also includes recommendations and conclusion of our research.

3. RESEARCH METHOD AND PROCEDURES

In this section, research approaches and methods and evolution of the research are defined. This section contains a description of the research problem and research setting including the contexts of the research. In addition, the collection, storage, analysis, and reporting

of the results of the research are described. Finally, the research design is summarized.

3.1. Research Methodology

This study can be classified as both qualitative research and, more pecifically, as constructive or quantitative research. (Jarvinen, P, 2001) defines constructive research as typically involving the building of a new innovation based on existing (research) knowledge and new technical or organizational advancements. Furthermore, it suggests that constructive research also involves an evaluation of the innovation. The qualitative study was carried out through literature reviews and observing case organization. The results of qualitative study were used in quantitative experiments to analyze, evaluate and propose our findings and recommendations.

3.2. Literature Review

The first preparatory stage of this research was started as a literature review on software maintenance and agile software development within the context of in-house software development environment. As suggested by (Cooper, H., Jarvinen, P., 1994) in 1984, the literature review aimed at defining the research topic's current state of knowledge. More specifically, the research aimed to understand the maintenance process and related issues for particular environment, and to identify the fundamentals practices of agile software development which are best suited for particular environment in the improvement of software maintenance process and product. In this stage, existing agile software development practices and maintenance issues were studied through available literatures and researches.

and analyzed. Data was gathered in the form of maintenance request and function point analysis was used to measured different metrics such as effort, productivity and quality. There are numerous measurement metrics are available for effort calculation but function point analysis was best suited in particular environment.

3.3. Case Study

In case study, the particular case organization's maintenance activities, issues and challenges were observed. This research was conducted in the IT department of a large enterprise. For fulfilling organization enterprise need, the IT department of the organization had developed their own ERP system and implemented throughout the organization which is running over different remote sites. Organization had very changing requirements in ERP due to changing needs of daily business activities. The IT Department was continuously engaged in fulfilling changing requirements and enhancement for future needs.

The ERP system had many integrated sub system i.e Human Resource Management (HRM) System, Financial Management (FM), Supply Chain Management (SCM) System, Manufacturing and Production System etc. There were different software teams responsible for each sub system development and maintenance. Each team consist of 3 to 6 professionals those were responsible for their team related activities and producing frequently maintenance releases.

3.3.1. Existing Maintenance Process

The case organization followed traditional methodology for software development and maintenance. The life cycle phases of development were requirement analysis, inspection, design, development, testing and transition. Each maintenance or enhancement request from user was passed through all the phases sequentially and finally the release of the module was delivered to user. Each maintenance or enhancement request was forwarded on standards form provided by IT department and it was recommended and approved from different channels. In the following section we will describe how a bug or maintenance

request is created, approved, inspected, fixed and deliver to the user in the case organization. Figure 2 illustrates the maintenance process of the case organization.

Figure 2
Existing maintenance process of case organization.

3.3.2. Maintenance Issue and Challenges

Software maintenance and continuous evolution remain a big challenge for in-house development environment because there is no cost bored by user to make changes and enhancement to the system. In the case organizations, users frequently demanded to fulfill their needs as soon as possible without considering the work load of IT department. There were different issues, problems and challenges related to maintenance that we analyzed. These issues are categorized into relation with process, product and people.

Process Related Issues

- Sequential development process took longer time to deliver product releases
- Maintenance task was assigned to individual team member who was responsible for

particular software module
- No effective testing was done due to shorter teams and early demand of release

Product Related Issues
- New product releases always remained immature because of lack of user awareness and improper planning, It took time to further refine the release after delivery and continuous bug fixing
- Productivity was lower due to over loaded team work schedule

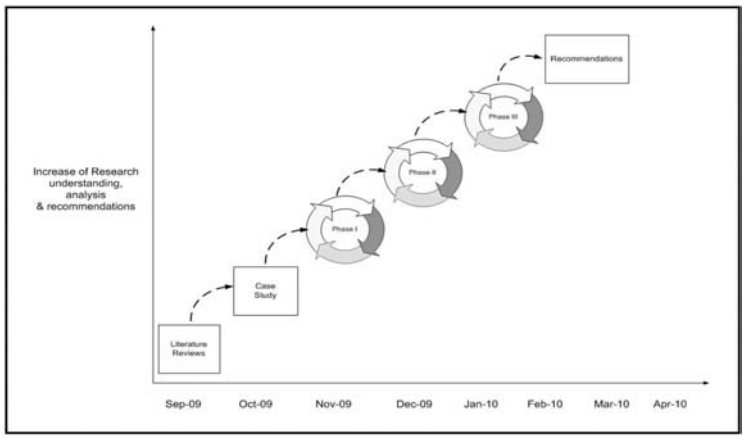
People Related Issues
- Users were not satisfied with product release schedule and demanded that every request should be fulfill as soon as possible
- User did not prioritize their needs
- Management always complaint about task delays and not satisfied with schedules
- IT team members were not motivated due to routine daily task
- Individual team members knew their work independently, code written by one programmer was difficult to understand by other programmer due to lake of standardization and independent working.

All the above mentioned issues and problems were used in our next phases to extend our experimental research and to get some useful results.

3.4. Experimental Research

Second part of this research contains some incremental experiments. Continuous examination of maintenance practices progressed thru different phases to analyze and evaluate the result and to identify best agile practices. During experiments, some extensions were made into agile practices and their usages were also examined in the case organization. Finally, on the basis of results a maintenance framework was proposed. Figure 3.2.3 illustrates the research progress graph.

Figure 3.2.3
Research Life cycle Progress graph.



3.4.1. Data Collection

Different approaches of data collection were used in this study. For quantitative analysis, data were collected in the following forms of maintenance requests, test cases and effort

reports. In the case organization, each maintenance or enhancement activity is started with the initiation of maintenance request. A maintenance request can be raised by departmental users, operators, testers or higher managers due to error or change in functionality. There were three types of requests i.e Bug Report (BR), Change Request (CR) and Enhancement Request (ER).

3.4.2. Estimation Method and Measurement Metrics

Function Point (FP) Analysis method was used to estimate the size of the maintenance releases which include all types of maintenance requests. The size of the software maintenance release is measured as the sum of number of FP added, changed or deleted. Effort is measured in person hours to produce the maintenance release. We use the following metrics to measure the quality, reliability and productivity of the releases

Table 4.4
Selected maintenance metrics and measurements.

Metrics	Description
Productivity	Effort / size in person-hour/FP
Quality	Defects per release (Defects / Total FP x 100)
Reliability	Number of test cases passed / all test cases x 100
Customer Satisfaction	Measured from Feedback Reports

4. RESEARCH DESIGN

The main goal of the research was set to improve the overall quality, reliability and productivity of maintenance process for in house development organizations. This research was progressed thru to resolve the addressed issues of the maintenance environment. The research was designed to experiment agile practices in maintenance development process and evaluate the result of implementation into phases as mentioned in figure 3.2.3.

In the case organization four teams were engaged in producing maintenance release for different sub systems of ERP. We selected Supply Chain Management (SCM) system team to experiment XP practices in their maintenance workflow. SCM team consisted of four members and they produced 1 or 2 releases in one month.

4.1. Research Phase I

In this phase, SCM releases processes were observed, they followed traditional software development life cycle method to produce releases. Initially, three previously produced releases data were collected to estimate the size and effort. During this phase, current maintenance practices were also compared with agile practices (XP's practices) and issues of maintenance release process were identified. Table 5.4 (a) represents the size and efforts of three releases

Table 5.4(a)
Size and effort data for SCM R1, R2 and R3.

Releases	Total Requests	Size in FPs	Effort In Days	Total Test Cases	Passed Test Cases	Total Defects
SCM R1	23	76	20	125	80	17
SCM R2	32	106	29	142	89	31
SCM R3	22	69	17	104	73	18

4.1.1. Results

Analysis based on measurement metrics identified in section 2.3. Releases size and effort data were applied on metrics to analyze the results. Table 5.4(b) represents the metrics statistics for three releases.

Table 5.4(b)
Measurement results of research Phase-I for SCM R1, R2 and R3.

Metrics	Measure	SCM R1	SCM R2	SCM R3
Productivity	FP/person-hour	2.1	2.18	1.9
Quality	Defects/FP per Release	22%	29%	26%
Reliability	Passed TC/ALL TC	64%	63%	70%
Satisfaction	Feedback	Average	Average	Above Avg.

It is analyzed that maintenance activities were prone to longer development time and lack of coordination between team and departmental users. From the observation and analysis of first phase, some results are presented below

Development Effort
- Overall productivity was low, task assigned to individual team member ended up with lots of bugs, bugs removal took longer time to complete the task
- Future changes in the code written by one member was difficult for others members to understand and modify
- There were no effective planning, a large part of effort was consumed on development and coding rather than planning and analyzing.

Product Quality
- Shorter test team was not able to perform maximum testing so reliability of delivered product was not high.
- Overall quality of product was not satisfactory due to lack of user involvement in planning and appropriate response of change.
- Poor design quality harder to modify for future changes

Collaboration
- User was not truly satisfied with the services provided in terms of changing requirements
- Team member was not collaborated within team due to responsibility of individual task assignment
- Management was not confident about the release schedule and demanded early production.

4.2. Research Phase II

Purpose of this phase was to experiment the effect of agile practices in the particular environment. An agile method XP was selected and its practices were implemented in the case organization. As mentioned in (Poole, C. J., Murphy, T., Huisman, J. W., & Higgins, A, 2001) that Extreme programming is a viable and very successful model for teams involved in pure maintenance and enhancement of a legacy code base.

In the first step toward implementing XP in maintenance environment, an analysis of comparison was done between existing maintenance practices and current XP practices. After comparison, it was observed that some practices of the case organization were similar to XP practices and already in use. It was also analyzed that some XP practices could considerably enhance quality and productivity of the case organization's maintenance process if used properly. The selected XP practices (Planning Game, Pair Programming,

Coding Standards & Refactoring and Test Driven Development) were used in next maintenance release. Data were collected and effort were calculated as defined above. Table 4.2 represents the size and efforts of SCM R4 release.

Table 4.2
Size and effort data for SCM R4.

Releases	Total Requests	Size in FPs	Effort In Days	Total Test Cases	Passed Test Cases	Total Defects
SCM R1	23	65	15	92	73	12

4.2.1. Results

Results of experiment of Phase-II and comparison with Phase I are presented in Table 4.2.1. These results were quite successful, when compared with Phase-I, in terms of overall improvement in maintenance process.

Table 4.2.1
Measurement results for research Phase-II for SCM R4.

Metrics	Measure	Existing Traditional Practices			XP Practices
		SCM R1	SCM R2	SCM R3	SCM R4
Productivity	FP/person-hour	2.1	2.18	1.9	1.93
Quality	Defects/Release	22%	29%	26%	19%
Reliability	Passed TC/ALL TC	64%	63%	70%	79%
Satisfaction	Feedback	Average	Average	Average	Satisfied

The adoption of original behavior of XP practices in maintenance environment was satisfactory but it was analyzed that there were some issues related to adoption and overall results.

Productivity
The result shows that productivity of team was decreased when pair programming and code refactoring was adopted

Quality
Result shows that a slight improvement in quality of production increased with respect to defect rate due to adoption of refactoring and pair programming practices but the result of quality was not significantly achieved as desired in this research.

Reliability
Results show that reliability of product improved in terms of passed test cases. The ratio of failed test cases decreased by adopting test driven approach and pair reviews. Test driven approach was not utilized accordingly due to shorter testing staff. XP's continuous testing approach was not used and testing was not covered fully due to limited staff.

Satisfaction
User satisfactions increased by involving them in prioritize their request. Minimum bugs reported after product release and user were satisfied.

4.3. Phase III

On the basis of the results analyzed in the second phase, it was observed that original appearance of agile practices were not effective for the particular environment, therefore

following extensions were made into the original agile practices to get the better results and improve overall quality of the process.

Planning Game with Feedback
We involved departmental user in release planning phase. User and technical team members set their priorities mutually. In the research phase, it was analyzed that a recorded feedback and lesson learned for every release would be helpful for future releases and their planning. It was also proposed that at the end of release completion, user should submit their feedback and all feedback should be stored in a repository for planning of future releases.

Dynamic Pair Programming
Pair programming was effective in terms of quality but lower productivity. In this phase, we proposed a dynamic pair programming practice where pair planning is more needed than pair coding. Pair should decide first which task are complex and need mentoring and reviews, and which tasks are simple those can be completed individually. Simple task should be completed first individually and there should be a cross review. Complex tasks should be completed in pairs. It would encourage pair learning, knowledge sharing and definitely the productivity of pair would increase significantly.

Test at End
Test driven development and continuous testing approach were not fully adopted in the experiment due to limited staff in testing team. Continuous testing approach was considered as an extra over load. It was analyzed that there was no need of continuous testing while pair reviews and refactoring were already in practice. Preparing test cases before construction phase was also not feasible in demanding situations because in test driven approach development team has to wait till the preparation of all unit test cases which was also wastage of development time. Due to shorter team and pair planning, we proposed the release testing at the completion of all development task.

Problem Patterns
We replaced refactoring with patterns. It was analyzed that for similar kind of problems or complex problem, team should have some patterns over process and practices such as design, code and interface. These patterns should be developed and continuously refined by team. Each pattern describes a problem which occurs over and over again in the environment and then describes the core of the solution to that problem in such a way that team can use this solution a million times over, without ever doing it the same way twice. Therefore, it was proposed that team should develop their process and practice patterns and stored them in repository for future uses.

After successfully adoption of above extended practices in the case organization's maintenance process, data was collected for the two releases. The same procedure was used to collect data and effort estimation. Table 4.3 represents the statistics of release R5 and R6 of SCM team.

Table 4.3
Size and effort data for SCM R5 and SCM R6.

Releases	Total Requests	Size in FPs	Effort In Days	Total Test Cases	Passed Test Cases	Total Defects
SCM R5	18	57	11	70	57	9
SCM R6	16	52	9	68	57	6

4.3.1. Results

Results of examination of extended XP practices are presented in Table 4.3.1. The results were quite successful in terms of productivity, reliability and user satisfaction, when

compared with Phase-I and Phase-II. The overall experiment was evaluated successful in improvement of maintenance process for particular environment.

Table 4.3.1
Measurement results for all research phases.

Metrics	Traditional Practices			Original XP Practices	Extended XP Practices	
	SCM R1	SCM R2	SCM R3	SCM R4	SCM R5	SCM R6
Productivity	2.1	2.18	1.9	1.93	1.54	1.52
Quality	22%	29%	26%	19%	15%	12%
Reliability	64%	63%	70%	79%	81%	84%
Satisfaction	Average	Average	Average	Satisfied	Good	Good

The above results show that the overall quality, reliability, productivity and satisfaction were increased when enhancement made in original behavior of XP practices for maintenance and evolution process. We got better results after implementing modified XP practices.

4.3.2. Phase IV

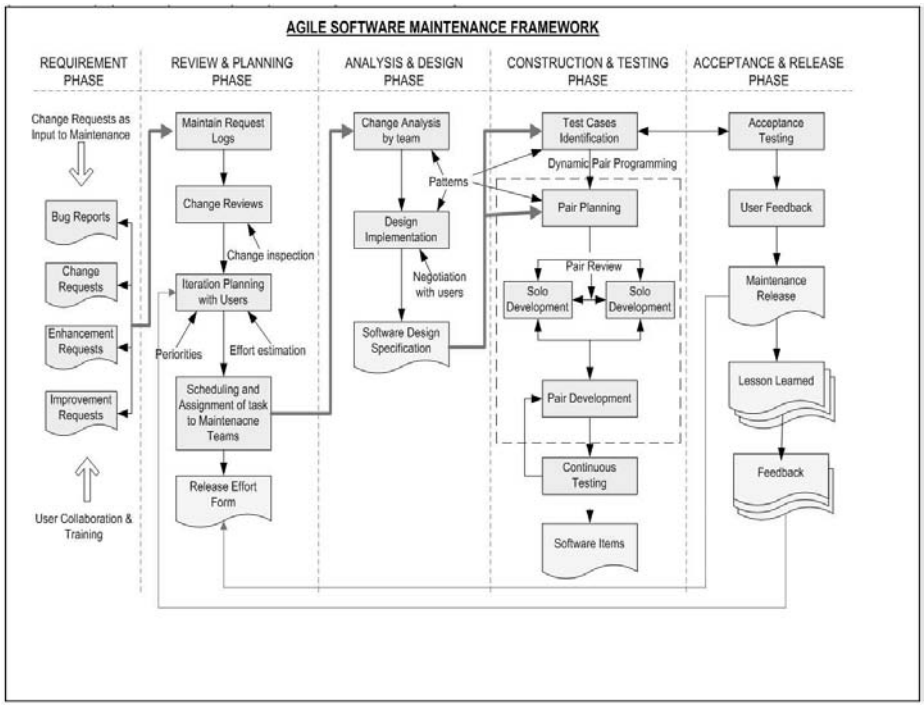
In the last phases of the research, results of these experiments were summarized into proposed framework. Finally, the results bringing it together to establish an agile maintenance work flow and we proposed an EASE - Software maintenance framework for in-house development organization and some recommendations in agile manifesto were provided.

5. PROPOSED FRAMEWORK

5.1. Proposed Agile Maintenance Framework

The framework was designed as it became evident that software development organizations were increasingly interested in adopting agile processes and practices, while lacking procedures and methods for supporting a systematic selection and deployment of new agile practices, and tailoring them to suit the organizational context. It illustrates how the method can be used in an organization' maintenance process when adopting and adapting agile. The proposed framework will serve as a guideline of implementing and tailoring agile practices for particular type of environments. Figure 4.1 illustrates the proposed software maintenance framework for in-house development environment. Figure 4.1 illustrates the proposed agile software maintenance framework for in-house development environment.

Figure 4.1
Proposed agile software maintenance framework.



The proposed framework was designed using IEEE Std 1219-1998 and agile's XP life cycle process. This framework provides the complete life cycle of maintenance and evolution release generation process. A brief description of maintenance activities of each life cycle phase is presented below

- **Requirement Phase**

Software modification or enhancement requests are collected and identified as input for maintenance iterations. Each modification request is classified into four categories as mentioned in Case Study earlier

- **Review & Planning Phase**

All requests are logged into repository. Changes are reviewed and inspected. All stakeholders altogether sets their business and technical priorities. Release iteration is estimated and planned with the collaboration of all stakeholders. Estimated schedule is recorded in REF and tasks are assigned to team accordingly.

- **Analysis & Design Phase**

Detail analysis of change or enhancement is carried out by team and design is prepared. Problems and requirements are negotiated with users and release schedule is updated accordingly. Appropriate patterns are applied to design if available. Finally, Software Design Specification (SDS) is prepared for construction phase.

- **Construction & Test Phase**

Software is constructed with respect to the design specification. Dynamic Pair Programmer approach is used in construction. Process and activities patters are applied where available.

Test cases are also prepared according to requirements. At the completion of development artifacts, tests are executed. A development and test cycle is run until all test cases passed. Finally, the product is delivered to user for acceptance testing.

- **Acceptance & Release Phase**

In the last phase, software is tested in user environment as per acceptance criteria. Defects are reported by users and resolved by pairs. User provides their feedback and lesson learned comments. Finally, the product release is implemented in production environment. Team also record their lesson learned remarks regarding the release generation. Iteration is completed here and next iteration started from planning phase.

5.2. EASE Recommendation

The EASE (Extension in Agile Software Engineering) will be an easy and adaptable framework for any particular environment. It suggests some great amendments in current agile practices which can be easily implemented to get the overall improvement in maintenance and evolution process. The aim of this research was to enhance the agile maintenance framework for in-house development environment. In this regard an incremental experiment approach was used to analyze and evaluate agile practices. During this research we observed some limitation in agile manifesto values. Finally, on the basis of results this manifesto values is reformulated and a recommendation is provided in agile software engineering manifesto. As already mentioned in literature review section that agile has four manifesto values. These values are

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

The agile software engineering is around the all four values. All these values have great influence on software quality and productivity. In this research, it is observed that there is lacking of evolution of patterns and lesson learned. Agile values don't focus on refinement of similar kind of problems through previous experiences. It suggests continuous improvement but not specifies any method to record and refine. In the conclusion of this research, we recommend an inclusion of a new value to agile manifesto which focus on creation and refinement of patterns for similar kind of problems and activities such as interface patterns, code patterns, design patterns and query patterns through experiences and lesson learned. The new value of agile manifesto is

- **Creation and refinement of patterns over similar activities and problem**

The above recommendation is for improvement in agile software engineering by getting advantages of previous experiences and making patterns for general problems which has refine solution. This value definitely improves future productivity.

6. CONCLUSION

This research presents the results of adoption and reformulation of agile practices in maintenance process for particular environment. In this research, both qualitative and quantitative researches, through incremental experiments, have been done to analyze the benefits of agile methods for software maintenance process. In this qualitative study, an assessment was done to identify the maintenance related issues and challenges for particular type environment. It was also identified that which agile practices could be beneficial to address the issues and problems related to maintenance process. In the quantitative research, an incremental approach was used to experiment and evaluate these practices. On the basis of results some extension in original agile methods was proposed to get the better results in quality and productivity. Finally, a software maintenance framework and recommendation

in agile manifesto is proposed. This research successfully proved that adoption and modification of agile practices in particular environment improve the overall quality of maintenance process and overcome the addressed issues.

The results from this study add to the empirical body of knowledge concerning introduction and use of processes based on XP in various contexts, which is needed as stated in (Layman, D. and Williams, L, 2004). Further, as reported in (Sharp, H. and Robinson, H., 2003) it is not necessary to strictly follow all practices without modifications in order to create an appropriate culture for developing software in an agile way. Instead, a modified, different set of practices might produce the necessary requirements. These results support the results presented in this research. That is, the introduction of an agile process based on XP can still be successful even though not all of the twelve practices are introduced and those introduced have been adapted to the team's development environment.

7. REFERENCES

IEEE Std 1219-1998, 1998. *IEEE Standard for Software Maintenance*, Software Engineering Standards Committee of the IEEE Computer Society, USA.

David, F., 2007. *Agile Methods and Software Maintenance*, Addison-Wesley.

Kent, B., 2000. *Extreme Programming Explained - Embrace Change*, Addison-Wesley.

Poole, C. J., Murphy, T., Huisman, J. W., & Higgins, A., 2001. *Extreme Maintenance*, Proceedings of the 17th IEEE International Conference on Software Maintenance (ICSM 2001), Florence, Italy, 301-309.

Jarvinen, P., 2001. *On Research Methods*, Juvenes-Print. Tampere.

Cooper, H., Jarvinen, P., 1984. *The Integrative Research Revise: A Systematic Approach*, SAGE Publications, Inc. Beverly Hills. p.144.