Industry's Experiences with Extreme Programming Practices

Dr. Mohamed Basheer.K.P, Asst. Professor, SS College, Areekode, Malappuram- Dt, Kerala
Noushad Rahim M, HSST Dept. of Computer Science DHSE, Kerala

Abstract- Agile methodologies are light weight software development processes. Agile processes focus on developing high quality software by embracing change in software development. In this paper the author discusses the best known Agile methodology called Extreme Programming (XP), focusing on XP practices and its Software Industry Experiences. The industry analysis reveals that XP is a successful software engineering methodology in small projects and some of its practices can even be used in big projects.

Index Terms- Extreme Programming, XP, Agile Software Development, Software Engineering, Pair Programming.

I. INTRODUCTION

Kent Beck who introduced the concept of Extreme Programming describes XP as a light weight efficient, low risk, flexible, predictable, scientific and fun way to develop software[1]. XP contains a set of values, beliefs and practices. The target of XP is to reduce defects, improve design, increase productivity, shorten time to market, easy knowledge transfer, integration of new comers, reduce training cost, responding to changes in the requirements and greater customer satisfaction. This is a replacement for the so-called Big Bang Waterfall model which needs large scale requirement analysis, design plan and requirement freezing. In XP the customer and the development team agrees a series of user stories that concisely define the user requirements. XP addresses the problem of huge development cost by providing early prototypes. XP consists of a number of practices like pair programming, small release, on-site customer input, code refactoring etc. Communication is the key factor of extreme programming and it discourages documentation. Even if XP contains twelve religious like practices, it is basically people oriented rather than process oriented. XP team is cross functional i.e., it includes members with testing skills, business analyst, domain experts and customer himself.

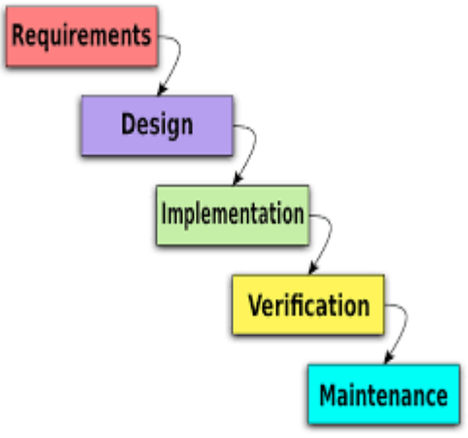

Fig.1: Waterfall Model

II. EXTREME PROGRAMMING PRACTICES

a. Planning Game.
Quickly determine the scope of next release. The customer along with the technical people determines the business priorities and technical estimate is done based on user stories provided by the customer. As reality over takes the plan, update the plan. This will reduce guess work and time wasted on useless features.

b. Small release.
Put a simple system into production quickly then release new versions on a very short cycle. The smaller releases (increments) will reduce development risks, provide frequent feedback and reduce the overall project slippage.

c. Metaphor.
The system will be described using simple shared stories. Customers, programmers and managers use the stories to explain how the system works. It is a naming concept for classes and methods which makes it easy everybody to guess what functionality is. It is a quick and easy way to explain the system and reduce buzz words and jargon.

d. Simple Design.
The System should be designed as simple as possible at any given moment. Extra complexity is removed as soon as it is discovered. The programmer should ask himself is there a simpler way to implement the functionality. Complex design is discouraged.

e. Testing.
Automated unit testing is used to eliminate defects through source code changes. Test first development will ensure that every piece of code is tested before it is shipped.

f. Refactoring
Programmers can rewrite programs whenever he feels to do so to remove duplication, to make the code simple or add flexibility without changing the behavior. This provides the developer the freedom to pro-actively improve the system using design principles and design patterns.

g. Pair Programming.
All production code is written by a group of pair programmers. Each pair will use one machine (One monitor and one keyboard). The pair will work together for designing, coding and testing. One of them is called driver and the other one is called navigator. When one is writing code, the other one will think about refactoring, test cases etc.

h. Collective ownership.
The code belongs to every member of the team and any one can change any code in the system at any time. This gives freedom for any programmer to re-factor the code anybody else has

written. If any member leaves the team it will not affect the development.

i. Continuous Integration.
Whenever a task is completed it should be integrated to the latest version of the system. The system should be built and all the automated test cases should be run. This is done multiple times in a day. This enables small releases.

j. 40 Hour week
XP advocates forty hour week rule for the developers. Overtime work will affect the quality and productivity. XP is a people oriented process and value is placed on the well being of the developers.

k. On-site Customer.
A real customer should be part of the development team all the time. The developers can contact the customer for any business requirement at any time. This will make sure that what is developed is what is really needed by the customer and a lot of guess work can be avoided.

l. Coding standard.
All the programmers should write the code using same coding conventions. It should be such that nobody can identify the author by just looking at the code. This will enable everybody to read and understand anybody else's code. This will ensure the code readability and the code itself become documentation.

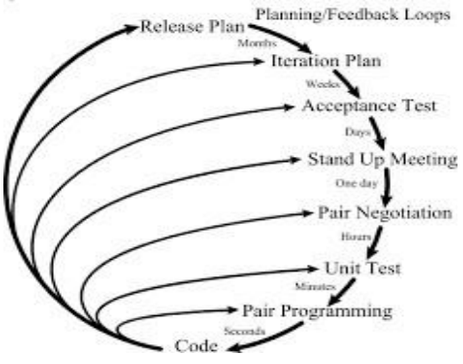

Fig.2: Extreme Programming Practices

III. INDUSTRY'S EXPERIENCES WITH EXTREME PROGRAMMING PRACTICES

Some of the practices of Extreme Programming were part of the software industry before XP was introduced as a software development process. But XP was invented 1n 1996 when automaker Chrysler invited Kent Beck to save their project Chrysler Comprehensive Compensation (C3). In C3 Beck started using some of Extreme Programming practices like test first development. But XP started using in software industry after Kent Beck wrote the book "Extreme Programming Explained" based on his experiences in C3. By 2000 onwards many companies have started using XP. In this paper the author tries to discuss experiences of some of the software companies like IBM, Connextra (www.Connextra.com), Avaya, Subex Systems with Extreme Programming.

At IBM a team of ten developers used XP for the development of an internal component for WebSphere Application Server, one of IBM's large middle ware products [4] . The team recognized that they will have to face too many changes in the requirement and thus they decided to try Extreme Programming. They followed XP practices religiously and delivered the component on time [4].

Connextra (www.connextra.com) has used XP for the development of a context-sensitive Advert for web, ActiveAd. Elena, a graphic designer at Connextra says that XP in Connextra worked well [5].

Avaya Labs also had positive results on their experience with Extreme Programming. All projects at Avaya was large and they used XP in some of the teams those were part of a larger project [6].

Subex Sytems,a Bangalore based Indian company, where the author had worked for three years, has used Extreme programming from the beginning itself and they were religiously following the XP practices.

The team at IBM says sitting developers and testers in the same room developed a great team spirit and a new working relationship was established each offering their own expertise to advance the delivery of the solution, and productivity noticeably improved [4]. Elena from Connextra observes that pair programming is some times difficult to follow, but being a graphic designer she was able to pair with Java programmers to work on java code even if she did not know java. She says that this helped her to understand the wider system context. Mark Windhottz who followed solo programming for twelve years and later moved to XP says that pair programming will correct the team to right direction if one of them is going in wrong direction. At Avaya the team used pair programming only upon difficult code. In normal cases they followed solo programming. Avaya had to do this because their project was a large one and only in small teams they were allowed to follow XP practices. One interesting thing noted at Avaya is that two developers who were located 2000 miles apart paired for debugging using a shared desk. Pair Programming is good as everybody share product knowledge and if any developer quits the organization it will not affect the development. In a study conducted by Prashant Behti, Dr. Laurie Williams and Dr. David Stotts, Dept. of Computer Science, North Carolina State University on pair programming on distributed Object Oriented projects[8], finds out that Pair programming can be used in distributed environments and is comparable with collocated software development in terms of quality and productivity. The author's experience at Subex System is that pair programming helped to resolve complex problems more easily than solo programming. Using pair programming for simple code is also good since it will help the new comers to be trained automatically on the product and the process. The only problem with pair programming is that it will not allow you to sit and silently analyze the code oneself.

The practice of short iterations has good result from the software industry. At IBM the team used fortnightly iterations with fifteen minute daily standup meeting. It took a few iterations for them to find the best way of doing iterations. But the team says the result was impressive [4] .The author's experience at Subex Systems of iterations based on user stories is that it is an excellent way of identifying the requirements of the system.

Since it is a cross functional activity including customer, developer and tester, time wasted on low value features can be reduced much. At Avaya Labs, projects used small releases in some form. The size of

the interval varied from two to four weeks. Some of the teams punctuated each interval with demo. Some other teams failed but they admits that they did not re-estimate after each iteration [6]. John Giblin who was a senior vice president of engineering at Dublin an Ireland based Software Company, Iona observes that XP embraces changes and that is a direct consequence of small releases. He says that because of the lengthy development cycles in traditional approach the original set of requirements only be partially relevant at the end of the development and XP is the solution. Also that XP can reduce time to market [7].

But implementing XP practices in a non XP environment is found to be difficult. Usually an upfront design is required for high level architecture design. This really contradicted XP's *just in time* design approach. But once a high level design is obtained, development of each module can be done using XP practices [4].The team at IBM observes that short release planning and iterations makes estimation of the development more realistic and project velocity can be easily tracked. Daily stand-up meeting allowed tracking progress concerns of the development. At Avaya XP's simple design is considered an important part of the practice.

The team at IBM observes that the test first development helped them to reduce much of the defects much earlier than they would have had if they had followed traditional approach. Test driven development, continuous integration with regression testing, refactoring and pair programming certainly improved code quality. Kyle Larson, who was a senior consultant at Minnesota based Advanced Technologies Integration says XP's continuous integration eliminates the nightmare of last minute integration of the product [8]. At Avaya, they tried test first design, but at some occasions schedule pressure created problems to implement it rigorously. When XP practices are tried in large projects they had to adapt some of the practices for the organizations. At Avaya, where XP is used for large projects code integration occurred weekly which is against XP practice of continuous integration. And code ownership varied from project to project; some occasions there were people who had gained natural expertise in certain part of the code. At the same time 40 hours/week was successfully practiced at Avaya.

Considering code refactoring, it helped IBM team to analyze complex problems and extensive unit testing gave people confidence to refactor code. At Avaya since they had an upfront design code refactoring did not figure prominently.

In all the companies, considered in this study, coding standards were successfully implemented. The author's experience at Subex Systems, it is found that following coding standard religiously will make the code easily readable for any developer and the code itself become documentation of the product. The only constraint is that every developer should strictly follow coding standard. The good side is that pair programming will guide the programmers in this direction. The most difficult practices of XP are onsite customer and metaphor. At Avaya no project had a onsite customer. This was not practicing at Avaya since a project itself had multiple customers. Also that use of metaphor found difficult because of existing practices in the organization. This has been reported by Kent Beck, that people tell him that they do XP except metaphor, of-course. The team at IBM observes that in XP customer is the driving force and this helps to reduce wastage of time on working on low priority features. But the thing is that the customer of their product was a framework itself.

From quantitative study of the XP practices the following success factors are identified:

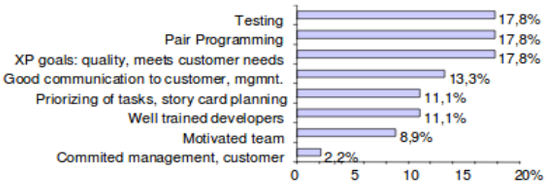

Fig.3: Success factors of XP

The study also reveals the risk factors of XP[10]. The most critical aspects are Metaphor and On-site Customer. The chart given below explains it in detail.

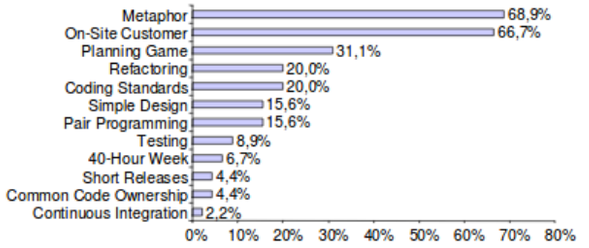

Fig.4: Risk factors of XP

IV. CONCLUSION

Extreme Programming has been successful in software industry. All developers who tried XP says that they would reuse XP in next project if allowed. There are software companies practicing XP in their entire development team. Most critical aspects are Metaphor and on-site customer. Pair programming, small release, test first design, collective ownership, refactoring and continuous integration are welcomed [10].Implementing XP practices in large companies faces difficulties, but practices like test first design can be adapted even in large projects.

V. REFERENCES

[1]. Beck Kent, Extreme Programming Explained. ISBN 0-201-61641-6, Addison Wesly, 2000.www.ibm.com/

[2]. Beck, Kent et al. (2001). Manifesto for Agile Software Development. Agile Alliance. Retrieved 14 June 2010.

[3]. International Journal of Computer Trends and Technology volumn2 Issue2 – 2011.

[4]. http:// developerworks/ websphere/ techjournal/ 0408_mitchell/ 0408_mitchell.html. IBM WebSphere Developer Technical Journal.

[5]. Using XP to develop Context-Sensitive Adverts for the Web. http://www.id-book.com/firstedition/casestudy_xp.htm

[6]. A Study of Extreme Programming in a Large Company. Niel B Harison Avaya Labs.

[7]. More Programmers going Extreme. http://news.cnet.com/2100-1040-255167.html

[8]. Exploring Pair Programming in Distributed Object Oriented Team Projects. Preshant Behti, Dr. Laurie Williams,Dr. David Stotts. Dept. of omputer Science, North arolina.

[9]. Jeffries.R, Anderson A Hendrickson , Extreme Programming Installed. Addison Wesly 2001

[10]. Bernhard Rumpe, Astrid Schroder. Quantitative Survey on Extreme Programming Projects.