IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 3, Issue 2, April-May, 2015
ISSN: 2320 – 8791 (Impact Factor: 2.317)
www.ijreat.org

Extreme Programming

Rimple Agrawal[1]
[1] Computer Science Department, Satyam College of Engineering,
Hapur, Uttar Pradesh, India

Abstract
Extreme Programming (XP) brought a major change in software development methodologies. It calls for changing practices and patterns that were adaptive in the past with new trends providing better results. Major elements of XP are excellent application of programming techniques, straight forward communication among team members, and teamwork through which better results are achieved that were even unimagined previously. XP provides a path to excel for people working together for developing software.
Keywords: *Waterfall Model, Agile Processes, Automated Testing, Unit Testing, Acceptance Testing, Incremental Planning.*

1. Introduction

Extreme Programming (XP) is a software development methodology designed to bring the whole team together and expedite the process of developing the new software. It is one of the Agile Processes which is already proved to be successful at different companies of different sizes throughout the world.

XP concepts were first implemented by Kent Beck, who is regarded as the creator of this innovative Software development approach.

XP was developed during the C3 project (Chrysler Comprehensive Compensation).This project was to merge a group of different salary systems into single software application. The Waterfall approach used initially for development was failed due to the complex requirements and integration difficulties. In this critical moment, Kent Beck with his XP team was invited to take over the project, and start the development from scratch. After one year, they successfully delivered the first working version of the software.

The XP approach was then tested, improved and set up was available for worldwide usage.

The first edition of Extreme Programming Explained: Embrace Change defines XP as:
"XP is a lightweight methodology for small-to-medium-sized teams developing software in the face of vague or rapidly changing requirements."

Extreme programming basically deals with approach to take to an extreme each of several known good software development practices.

"The first time I was asked to lead a team, I asked them to do a little bit of the things I thought were sensible, like testing and reviews. The second time there was a lot more on the line. I thought, "Damn the torpedoes, at least this will make a good article," [and] asked the team to crank up all the knobs to 10 on the things I thought were essential and leave out everything else." — Kent Beck

2. XP Methodology

2.1 Why XP?

Previously to XP, below were the common problems faced during software development:

- All the requirements are required to be captured before starting design.
- Requirements are required to freeze before starting development.
- Changes in midway of development are resisted as it will lengthen schedule and delay release.
- Change control process is to be properly maintained to ensure that changes requested are inspected carefully and it is ensured that no change is made without intense scrutiny.
- Delivered product is obsolete on release.

XP defines a path for improving software development and it differs from other methodologies in following ways:

- Its development cycle is short, hence providing early and continuous feedback.
- Its incremental planning technique provides the overall plan that is expected to be achieved throughout the life cycle of project.
- Changing business requirements can be flexibly scheduled with ongoing development.
- Automated tests are written by programmers, customers and testers to monitor the development and detect defects early.
- It works in close collaboration of individuals having ordinary talent.

2.2 Rules

Extreme Planning is based on below rules:

Planning:
Under this stage user stories are written. User stories are used to create time estimates for release planning meeting. In this stage developers are frequently releasing small releases of iterative versions of the deliverable system to customers. Project is divided into iterations which are scheduled for completion in one to three weeks.
Managing:
Work spaces are required to be open and physical barriers such as cubicles that divide people are to be dropped. It also requires moving people around and free communication to avoid serious knowledge loss and coding bottle necks. Stand up meeting is required where majority need not to contribute, but attend just to hear the outcome hence a large amount of development time is sacrificed to gain a trivial amount of communication.
Designing:
This rule requires that designing a model must be kept simple. Spike solutions are required for finding answers to tough technical and design problems. A spike solution is a very simple program to explore potential solutions to the issues faced. This rule also restricts early addition of extra functionality since only 10% of the additional functionality (extra stuff) will ever get used.
Coding:
This rule requires that the customers must be constantly available, not only to help the development team but to be a part of it as well. Code must be written as per the agreed standards. It is required to create the test first before creating the code as it will be much easier to create the code later, if test cases are available to developers. Collective ownership encourages everyone to contribute new ideas to all segments of the project.
Testing:
Unit tests are performed on each code segment. Unit tests are considered as the corner stones of extreme Programming. Whenever a bug is found test cases are created to guard against it from coming back. Acceptance tests are created from user stories. The user stories selected during the iteration planning meeting will be translated into acceptance tests for each iteration.

2.3 Values

Extreme Programming is based on following values:

Simplicity:
Only the required functionality is developed with no additional features. It maximizes the value created for investment made till date. Simple and small steps ate taken to mitigate failures as they occur. Maintenance of developed project needs to be done for long term at reasonable cost hence the delivered product is not obsolete.
Communication:
Face to face communication is done daily with each member of team. Team works together on everything from requirements to code and from code to testing. Every team member is equally responsible for each task.
Feedback:
Software being developed is demonstrated early while the developers listen carefully to feedback and incorporate the necessary changes. Each iteration is to be taken seriously to deliver working software with proper commitment.
Respect:
Each and every member of team is treated as a valued team member. Management respects each member's right to accept responsibility and receive authority over our own work.
Courage:
Progress and estimates of the developing software must be taken seriously. Documentation of excuses for failure as we plan to succeed is not justified. Since no task is considered as individual work but team effort so there is no fear of the job at hand. Changes are incorporated with respect whenever they happen.

2.4 Practices

Following practices are to be followed with Extreme Programming approach:

Whole Team:
Each and every person linked with an XP project is considered as part of a single team including a business representative--the 'Customer'. Team members are programmers, testers, analysts and manager. XP teams are considered to have no special talent. Under XP, it is ensured that everyone understands the working of system. Every member equally understands where to look for functionality and where new functionality is to be added.

Planning:
Under XP approach, planning is done for releasing each iteration on time. Planning is divided into:

- XP Release Planning
    - Functionalities required are presented by customers.
    - Estimation for difficulty is done by programmers.
    - Requirements are though imprecise but are revised regularly.
- XP Iteration Planning
    - Iterations are scheduled for around two weeks.
    - Features required in iterations are presented by customers.
    - Features are broken down into tasks by programmers.
    - Tasks are assigned to team members.
    - At the end of iteration, running software with required features is available.

Testing:
Automated acceptance tests are designed by customer for a feature. These tests are designed to verify that a feature is implemented correctly as per requirements. Once the test is executed, the team is confident that the tested functionality will run accurately thereafter. System under test always improves, never backslides.

Release:
Small and functional releases are required frequently in XP approach. For every iteration running and tested software is released. The Customer evaluates the software, provide feedback and releases it to end users. Since the software is visible and given to the Customer at the end of every iteration, it develops confidence in developers and continuous feedback help to incorporate necessary changes.

Pair Programming:
Under pair programming, production software is developed by two programmers, sitting side by side, at the same machine, one programmer writing and other reviewing and providing necessary comments. Hence code is reviewed by at least one other programmer. Research into pair programming shows that pairing produces better code in the same time as compared to the programmers working singly.
Pairing also improves team technically since it helps in communicating knowledge throughout the team.

Test driven development:
Under XP approach, software development is Test Driven Development. Software development work is done in short cycles of adding a test, and then making it work. This approach makes easy to produce code with 100 percent test coverage. Under this approach, all unit tests are collected together and during each release of code each test must execute properly.

Continuous Integration:
Since XP works in multiple iterations, integration is a major part in this environment. System must be fully integrated at all times. This requires an additional activity by the development team. Daily, or even multiple times a day builds are created. It avoids 'Integration Hell' and code freezing.

Collective Code Ownership:
Since there is team involvement in every activity hence the code ownership is not individual. Code is considered to be owned by complete team in a manner that any pair of programmers can improve any code at any time. It results in no 'secure workspaces'.

Coding Standard:
Common coding standard is to be followed by developers in XP approach. All code in the system must look as though written by an individual. Code must look familiar, to support collective code ownership.

2.5 Lifecycle

The lifecycle of XP consists of five phases:

Exploration :
In the exploration phase, customer prepares the story cards. These story cards describe the features that are to be included in the first release. Each story card describes a feature to be added into the program. At the same time the project team familiarize themselves with the technology and practices that will be used in the project. The technology to be used will be tested and the architecture possibilities for the system are explored by developing a prototype of the system. This phase takes between a few weeks to a few months, depending largely on how familiarity of the programmers with the technology.

Planning:
The Planning Phase sets the priority order for the stories and develops an agreement of the features that are part of

the first small release. The Programmers estimates the efforts required for each story and the schedule is then planned with proper agreed customers. Schedule of the first release does not normally exceed two months. The planning phase itself takes around 2 to 3 days.

Iteration to Release:
The Iteration to release phase includes several iteration of the systems before the first release. The schedule developed in the planning phase is broken down into a number of iterations, each of which takes one to four weeks for implementation. The first iteration creates the architecture of the whole system. This is achieved by selecting the stories that requires building the structure for the whole system. Selection of stories for each iteration is decided by the customer. Customers are required to create functional tests that are required to be executed at the end of each iteration. After completion of the last iteration the system is ready for production.

Productionizing:
This phase basically requires additional testing and performance check of the deliverable Additional changes can still be found at this phase and the decision is taken if they are to be included in the current release or delayed for future releases. At this stage, the iterations may be required to be quickened from three weeks to one week. Documentation of postponed ideas and suggestions is done for future implementation during, e.g., the maintenance phase.

Maintenance:
After the first release is productionized for customer, the XP project team must both keep the system in the production running along with developing new iterations. Thus under maintenance phase team requires an additional effort for customer tasks as well. Thus, the development velocity may appear to decelerate after the system is in production state. The maintenance phase may require incorporating new people into the team and may call for changing the team structure.

Death:
The Death phase is near when the customer have no more stories for implementation. This requires that the system is completely developed satisfying the customer requirements along with issues regarding performance and reliability. Under XP process, this is the appropriate time for creating the necessary documentation of the system.

3. Figures

3.1 Lifecycle of Extreme Programming

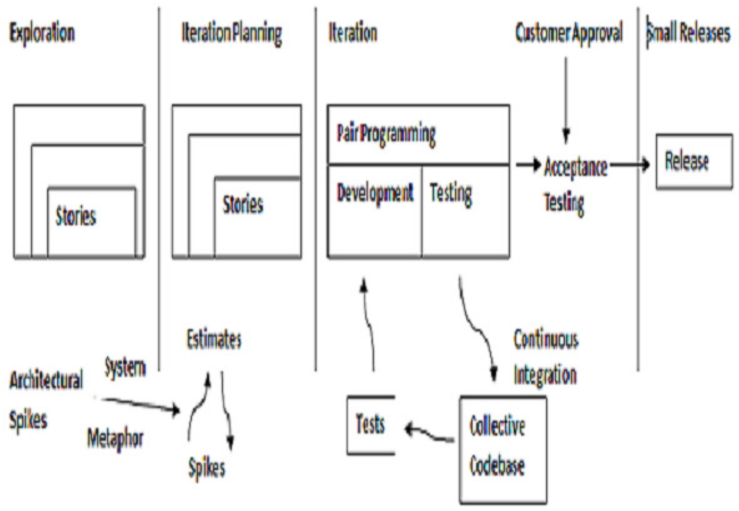

Fig. 1 Lifecycle of XP Approach.

4. Conclusions

Extreme Programming is a agile software development methodology. Under XP each individual is considered equally as part of a team, with clear goals and a defined plan of execution. XP assumes that each member want to work together in a team to achieve common goals. Each team member is considered to have ordinary skills with no special talent. XP assumes that each member want to grow by improving their skills and relationships. XP development model tries best to incorporate necessary changes if demanded by customer and make sure that no extra functionality is developed implicitly. XP approach may appear to be expensive for beginners but it is proved to provide improved in long run. This approach is tested and now a days it is implemented worldwide in IT industries.

References

[1] Kent Back, and Cyntia Andres, Extreme Programming Explained: Embrace Change, Boston: Addison-Wesley, 2012.
[2] A. Name, and B. Name, "Challenges of users Involved in Extreme Programming Projects", International Journal of

Software Engineering and Its Applications , Vol. 3, No. 1, 2009, pp. 019-029.
[3] Christian H. Becker, "Using eXtreme Programming in a Student Environment: A Case Study ", M.S. thesis, Computer Science, Karlstad University, Karlstad, Sweden, 2010.