Enabling to Apply XP Process in Distributed Development Environments with Tool Support

Ali Akbar Ansari [1], Afsaneh Ansari[2]

[1] Islamic Azad University, Najafabad Branch, Department of Computer Science and Engineering Isfahan, Iran

[2] Islamic Azad University, Najafabad Branch, Department of Computer Science and Engineering Isfahan, Iran

Abstract

The evaluation both in academic and industrial areas of the XP methodology has shown very good results if applied to small/medium co-localized working groups. In this paper, we described an approach that overcomes the XP constraint of collocation by introducing a process-support environment (called M.P.D.X.P) that helps software development teams and solves the problems which arise when XP is carried out by distributed teams.

Keywords: *eXtreme Programming (XP), Distributed Software Development (DSD), Project Management, M.P.D.X.P.*

1. Introduction

Globally distributed work is taken up as an alternative to single-site mainly because of the economic and strategic benefits it offers. Distributed software development is becoming the norm by promising potential advantages like global resources, attractive cost structures, round-the-clock development and closeness to local markets[2]. The scattering of team members and functions around the world introduces barriers to productivity, cultural and languages differences can lead to misunderstanding of requirements, time zone differences can delay project schedules[8].

Extreme Programming (XP) [1,5,6,7] is undoubtedly the hottest Agile approach to emerge in recent years. XP processes allow to respond quickly to requirement changes and stress collaboration between software developers and customers (customer-driven) and early product delivery. From a work flow point of view, they deliver software in small, frequent (e.g. weekly) iterations. At the end of each iteration the software is fully built and tested and could be released to the customer. The customer may reprioritize the requirements to benefit from any learning.

XP has in its original form proposed by Beck [3] two severe limitations. First, it does not scale well to larger teams. Second, it requires the XP team to be collocated. Overcoming the collocation requirement while preserving the high productivity and quality of XP processes is one goal of our approach and the focus of this paper.

This paper is organized as follows: The next section, we give an overview on our M.P.D.X.P approach. Section 3 describes about elicitation and managing requirement in XP. Section 4 provides a detailed the proposed solution and usage scenario and finally section 5 concludes the paper and a look on future work.

2. The M.P.D.X.P Approach

In this section, we introduce our solution for the management of Project in Distributed eXtreme Programming called M.P.D.X.P (**M**anaging **P**roject in **D**istributed e**X**treme **P**rogramming).

The overall goal of the M.P.D.X.P approach, a tool supporting the XP practices for requirement gathering and project management – user stories and the Planning Game. M.P.D.X.P has been developed in C# and is accessed through the Internet in an user friendly and agile way. It keeps track of multiple projects, each with its releases, iterations, user stories, acceptance tests and tasks. M.P.D.X.P allows project tracking, automates documentation generation. We now describe requirements that were underlying the development of M.P.D.X.P.

2.1 Defining the Requirements of the Tool

Distributed teams are disadvantaged compared to co-located teams in at least in three ways: Communication within the team is hindered, team members are less aware of each other, and common access to physical objects and

places is difficult. Thus, the tool must have the following characteristics:

- **Agility** – the tool must be easy to use and easy to adaptable and reconfigure.
- **Web-based** – the tool must be accessed through standard Web browsers.
- **Integrate environment with data management** Integrated environment and data sharing enable the project team to focus on development work, while daily data management has been automated. This means that all project-related data is managed at a unified location and integrated tool support eases tedious tasks, such as information retrieval, distribution, consistency checking, archiving, etc.
- **Project coordination** - XP teams are usually much more closely coordinated than distributed projects. Hence, project coordination support is strongly required for DXP. This should allow the team to assign tasks to developers, set deadlines and get an overview on the current state of the project. Team members on the other hand should be able to access their to-do lists and retrieve relevant information for performing their tasks easily.
- **Synchronous communication** - XP replaces documentation by synchronous, face-to-face communication. Face-to-face communication is not feasible in a distributed team and needs to be replaced by technical means. Besides using e-mail for communication, synchronous communication like audio and video calls or text chat may be helpful. If two developers want to do pair programming, application sharing is needed.

3. Requirement Gathering and Management in XP

Product development in the XP process starts with "planning game." Planning game can be divided into "release planning" and "iteration planning" [5]. During planning game, customer writes user stories, which the developers estimate and customer then subsequently prioritizes. Planning game is a phase in XP development when requirements, that is stories, are elicited, estimated and selected for release. Planning game is performed for each release. A release is divided into iterations. The subset of stories based on priority and size is then selected for each iteration. This is called iteration planning. Developers then divide stories into tasks and give an estimate for each task. Estimating user stories is difficult in other than very coarse level. On the other hand, estimating tasks is much more easy and accurate because tasks are defined in more detailed and concrete level. The next step in the XP process is development when the iterations are produced and released. Then acceptance tests are used to validate the completion of stories. Story cards are one of the important aspects of the XP. They are playing vital role in XP. It describes functionality of system or software to be build that will be valuable to either purchaser or user of software. User stories are composed of three aspects [4]:

• A written description of the story used for planning and as a reminder.
• Conversation about the story that serves to flush out the details of the story.
• Tests that convey and document details and that can be used to determine when a story is complete.

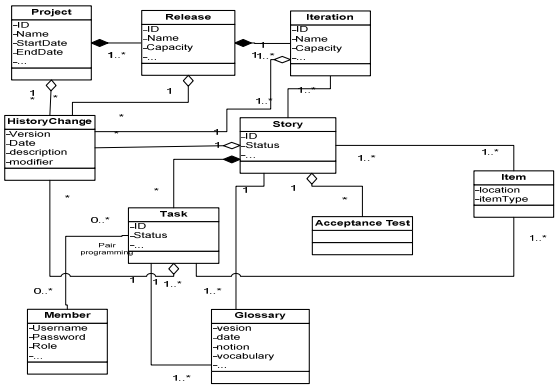

Fig. 1 UML class diagram showing the entities (persons and activities) involved in the Project.

This structure is shown in Fig. 1 in the form of a UML class diagram. In this model, we allow user stories and acceptance tests belonging to more than one iteration, since their implementation could be completed in the next iteration.

4. Using M.P.D.X.P for Distributed eXtreme Programming

The following scenario illustrates the infrastructure provided by the M.P.D.X.P framework to support distributed extreme programming. Team members access

the Internet with a Web browser and connect to the M.P.D.X.P server. First, they login to the M.P.D.X.P system to access their workspace. From their workspace they may retrieve the list of current projects, user stories, currently available tasks, task estimation.

After logging in successfully, they'll want to start a new project by clicking the Add Project button from the Projects list. They can optionally choose one or more users to assign to this project. The users they select will be able to access the project, while other users will not see the project when they log in.
M.P.D.X.P has 4 basic user roles for they to choose from:

- **Viewer**: can read and review stories and in the system, has no edit or delete permissions
- **Editor**: can read, create, and edit stories and tasks, but cannot delete
- **Project Admin**: all Editor permissions, plus can delete stories and tasks
- **System Admin**: all Project Admin permissions, plus can add, edit, and delete Projects and Users

Most developers will need at least Editor access, with perhaps the Team Lead or Project Manager having Project Admin permission. The System Admin permission should be reserved for a select group of users who need to administer projects and users.

4.1 Creating a Story

The left part of the main screen displays a menu that allows accessing all components of M.P.D.X.P**.** The top part shows to user his role, to-do list and projects that has the ability to access. From the main story list, click the Add Story button. Only the story name has to be entered initially, so they can quickly capture a feature request or idea and come back later to flesh out the details.

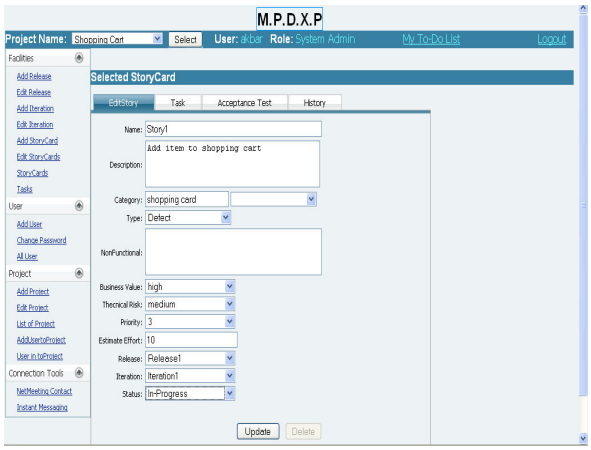

Fig. 2 Creating a Story

4.2 Adding Tasks to a Story

They can add tasks from the Edit Story screen. Click the Tasks tab from the story edit, and click the Add Task button. Once a task is assigned the task owner can edit the task to update the completed hours and/or the estimated hours to track progress. When a task is finished, the task owner can set the Status to Completed to communicate that to the rest of the team.

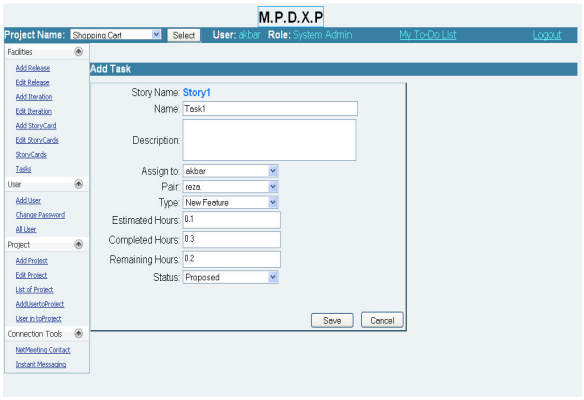

Fig. 3 Adding Tasks to a Story

4.3 Viewing History

M.P.D.X.P automatically tracks changes to Stories, Tasks, Releases, Iterations, and Projects. By editing one of these items, and clicking the History tab, they can see a log of what changes were made when, and by whom.

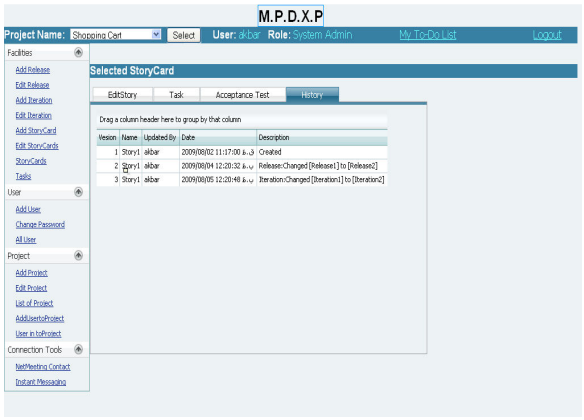

Fig. 4 Viewing History

4.4 Planning an Iteration

M.P.D.X.P supports an Iteration Planning tool which they can access by editing a iteration and selecting the "Iteration Plan" menu.
The Iteration Planner lets they move and delete stories from a backlog. By default, they can assign stories that are not yet assigned to a release, or limit the selection to stories in a particular release.
The Iteration Plan is updated in real time with the total estimated effort, and will warn they if the effort estimate exceeds the capacity of the iteration (that they can specify from the Iteration Edit screen).

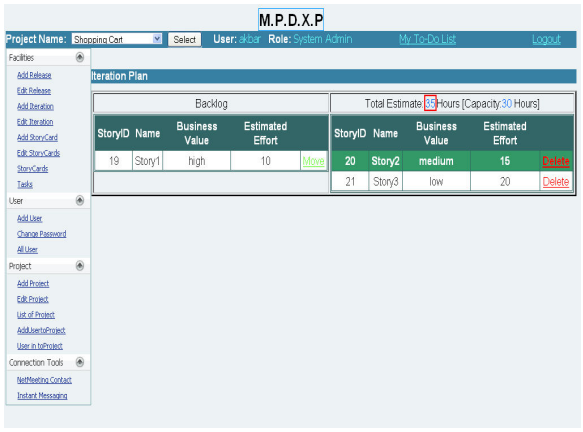

Fig. 5 Planning an Iteration

4.5 Tracking Iteration Progress

The Iteration Status View is the view they will use most often when tracking an iteration in progress. They can access it from the Iteration List using the Status View link for an iteration.
The Iteration Status view lists all stories scheduled for the iteration, and for each one, shows all tasks as graphical cards. The tasks are organized into four columns based on their status:

- **Defined:** Defined state is initial state which is when story or task has been written.
- **In Progress**: In Progress state is activated when task has been assigned to some developer and story is in implementing state when some of its tasks are implementing state.
- **Completed:** Completed state is activated for task when task has been completed. Story is in completed state when all its tasks are also in completed state, But it is still not passed the Acceptance test.
- **Accepted**: Accepted state is activated for story when it is passed the Acceptance test.

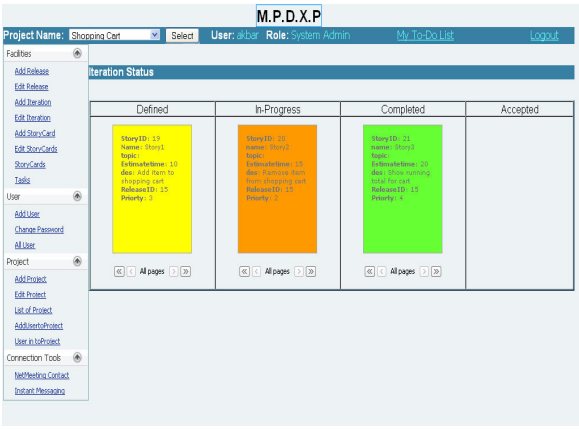

Fig. 6 Tracking Iteration Progress

5. Conclusions and Future Work

In this paper, we have shown that awareness and communication are key issues when carrying out XP in a distributed setting. The M.P.D.X.P environment can support distributed XP teams by integrating these key issues into the programming environment. We argued that this solution provides enough support to make XP work in a distributed setting.

M.P.D.X.P combines many of the techniques described in Section 4:

- M.P.D.X.P provides basic communication support (chat and audio).
- M.P.D.X.P is a coordination system since it helps to plan and co-ordinate activities in the planning game.
- M.P.D.X.P integrates a version management system
- In supporting the planning game, M.P.D.X.P is also a group decision support system.
- M.P.D.X.P includes multi-user editors for annotating story cards in the planning game.

Other future work on the tool will be entirely driven by our partners' needs. They include:

- Extending the tool to support other well-defined agile process, such as SCRUM and Feature Driven Development.

References

[1] Canfora, G., Cimitile, A., Di Lucca, G. A. & Visaggio, C. A., "How distribution affects the success of pair programming", International Journal of Software Engineering and Knowledge Engineering, 2006, 16**,** 293-313.
[2] Damian, D. & Moitra, D., "Global software development: How far have we come?", IEEE Software, 23**,** 17-19, 2006.
[3] Beck, K., Extreme Programming explained: Embrace Change, Addison-Wesley (September 1999), ISBN: 201-61641-6.
[4] Cohn, M., User stories applied for Agile Software Development. Reading, MA: Addison-Wesley, 2003.
[5] K. Beck and M. Fowler, Planning extreme programming. New York: Addison-Wesley, 2001.
[6] Jeffries, R., Anderson, A., Hendrickson, C.: Extreme Programming Installed, Addison-Wesley Pub Co, 2000, ISBN: 0201708426
[7] Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J., "Agile Software Development Methods: Review and Analysis". VTT Publications, 2002.
[8] R. Prikladnicki,J. Audy,R. Evaristo." Distributed software development: toward an understanding of the relationship between project team, users and customers", ICEIS conference, 2003.

A.A. Ansari was born in 1984. He received the B.Sc. degree in computer engineering from Azad University, Shiraz, Iran, in 2006. the M.Sc. degree in Software Engineering Azad University, Najafabad in 2010. His research interests are Software Engineering, data mining and computer networks.

A. Ansari was born in Iran. She receives B.Sc degree in computer engineering from payam noor univercity in 2010. She is currently a M.Sc. student in Information Technology at qeshm University. Her research interests include Information Technology, Software Engineering, telecommunications and computer networks.