ADAPTING EXTREME PROGRAMMING FOR GLOBAL SOFTWARE

DEVELOPMENT PROJECT

Except where reference is made to the work of others, the work described in this thesis is my own or was done in collaboration with my advisory committee. This thesis does not include proprietary or classified information.

_____

Yuan Tian

Certificate of Approval:

Kai Chang Professor Computer Science and Software Engineering	David Umphress, Chair Associate Professor Computer Science and Software Engineering
Dean Hendrix Associate Professor Computer Science and Software Engineering	George T. Flowers Dean Graduate School

ADAPTING EXTREME PROGRAMMING FOR GLOBAL SOFTWARE DEVELOPMENT PROJECT

Yuan Tian

A Thesis

Submitted to

the Graduate Faculty of

Auburn University

in Partial Fulfillment of the

Requirements for the

Degree of

Master of Science

Auburn, Alabama
May 9, 2009

ADAPTING EXTREME PROGRAMMING FOR GLOBAL SOFTWARE

DEVELOPMENT PROJECT

Yuan Tian

_____

Signature of Author

_____

Date of Graduation

VITA

Yuan Tian, daughter of Liaofa Tian and Xiaoheng Zhou, was born on April 20, 1980 in Sichuan, China. She earned her bachelor's degree from Chengdu University of Technology, Chengdu, China in 2002. She was hired by SCU-NESEC Infosec Co.,Ltd from 2002 to 2003 then joined Beijing Beyondsoft Co.,Ltd in 2004, at where she worked as Software Engineer until she was admitted in to master program of Department of Computer Science and Software Engineering of Auburn University in 2006 Fall.

THESIS ABASTRACT

ADAPTING EXTREME PROGRAMMING FOR GLOBAL SOFTWARE DEVELOPMENT PROJECT

Yuan Tian

Master of Science, May 9, 2009
(B.S., Chengdu University of Technology, China, 2002)

87 Typed Page

Directed by David Umphress

Growth of the global economy has led to remarkable changes in the way software is developed. Global Software Development (GSD) is becoming the norm for many technology companies. Even though organizations are enjoying the benefit brought by GSD, communication has been an issue impeding its further growth. Miscommunication and misunderstanding brought by the distance between development sites happen much more frequently in GSD projects than co-located projects, which eventually influence the software quality and customer satisfaction. Cultural distance also exacerbates these problems. Many studies have been conducted to either find a software process or develop a software application to facilitate the GSD. Because of its flexibility, Agile Methods are considered suitable processes for GSD. In our study, we examine the characteristics of Extreme Programming (XP), the most popular Agile process, and

suggest changes to better support GSD. A prototype Eclipse-based plug-in is designed to facilitate the implementation of this process.

ACKNOWLEDGEMENTS

I sincerely appreciate my advisor Dr. David Umphress not only for the guidance he has provided throughout my study at Auburn, but also his encouragement during my frustration. I would also like to express my gratitude to the advisory committee members, Dr. Kai H. Chang and Dr. Dean Hendrix.

Above all, I would love to express my deeply appreciation to my wonderful families and friends. I feel so blessed to have my parents and brother who have been loving me and supporting me through my life. And I am so grateful for you, Yi Zhang, I would have never accomplished this without you.

Style manual or journal used: Journal of Surface Mount Technology

Computer Software used: Microsoft Word, Microsoft Excel, Microsoft Picture Manager, WinSMITH Weibull

TABLE OF CONTENT

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 The General Research Area

With the growth of the global economy in the past several decades, the software industry has witnessed a steady trend toward the globalization of business. According to Gartner[1], globalization of software development has expanded rapidly in recent years and has brought in its wake changes that impact application development projects. Global Software Development (GSD) is becoming the norm for many technology companies. A software project involving different teams located at multiple sites in different cities is no longer a novelty. A software company could have branches in different cities or in different countries. Companies also collaborate with each other across the globe through software outsourcing. According to statistics collected in 2001, 203 of the US Fortune companies are engaged in offshore outsourcing [2]. Many American organizations are building their development centers outside the country, and many software shops are growing outside traditional centers (such as US, Japan) in India, Ireland, Israel, China, etc. The factors that accelerate this trend include cost savings, proximity to the market, "around-the-clock" development, and survival from competition, etc.

Economic forces are relentlessly turning national markets into global markets and spawning new forms of competition and cooperation that reach across national boundaries. This change is also having a profound impact not only on marketing and

distribution but also on the way products are conceived, designed, constructed, tested, and delivered. Software organizations are required to develop in a high-speed and agile ways to adapt to the current dynamic business environment. Can we still use traditional project management techniques? According to Lindstrom and Jeffries[3], the traditional popular project management techniques focus on developing a plan and sticking to the plan. This improves coordination but reduces the ability of the project to adapt to new information regarding requirements or implementation details. However, traditional project management techniques do not take into account that the customer will be in the US and the development teams will be in India and China. The problems brought by distance are not taken into consideration. Moreover, they cannot meet with the dynamic requirement of the GSD.

The emergence of Agile Methods with their emphasis on flexibility, informal collaboration, and working code brought fresh air to GSD. Software development organizations have been striving to blend the GSD projects with Agile Methods to reap the benefits of both. Among them, Extreme Programming (XP) [31] is the most widely used one which shares the values exposed by the Agile Manifesto[4] for software development but goes further. XP is a set of twelve independent software development practices conceived initially for small development teams working on projects with high degree of change, and later successfully applied to larger teams. However, XP and GSD have significant differences in some of their key tenets. Is XP the best development method for GSD projects? In this study, we examine the nature of XP and GSD projects to find their common interests and the possible areas in which they can be blended. Based

on this study, we propose a new methodology which is XP-based and more adaptive to GSP projects. An Eclipse-based light-weight IDE plug-in is developed to illustrate the application of our new methodology in GSD projects.

1.2 Background

In this chapter we provide some background knowledge of our research. It also explains why among all the Agile methods we chose Extreme Programming for our research.

1.2.1 Global Software Development

First, Global Software Development does not necessarily involve multiple companies. It can be a project involving multiple subsidiaries located in different countries. The most significant difference between one company and multiple companies is that team members of one company share the same organizational culture. This plays a significant part in smooth communication and team management. Also, Global Software Development and Distributed Development are different. Distributed Development is not necessarily global. It can be multiple development sites within one country. This means in most cases that software stakeholders are speaking same language. There is no cultural gap between team members. Since the product is applied within the country, there is no different requirement from the target market. The more stakeholders that are involved, the more complicated project environment will be, which affects the project progress. When the project teams are globally distributed, multiple stakeholders located in different countries with people with different cultural backgrounds, this situation gets much more complicated. In our study, we mainly focus on this kind of Global Software Development.

1.2.2 Agile methods and Extreme Programming

Over time, software development methods have changed with our society. The evolution of development from classic Waterfall to Iterative to Agile Methods illustrates the aim of accommodating the needs of the environment. This evolution is depicted in **Error! Reference source not found.**Figure 1.4 with Extreme Programming (XP) as an example of Agile Methods.

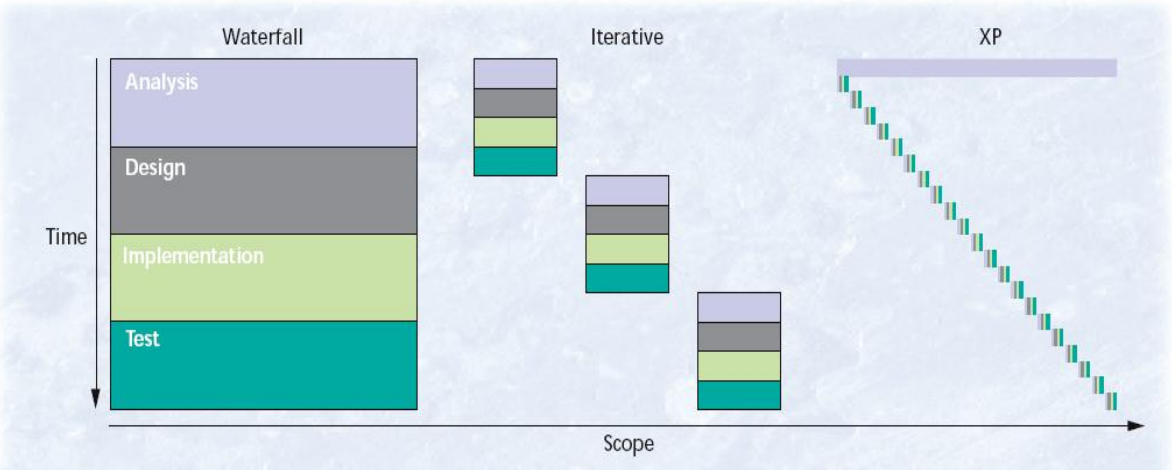

Figure 1.4: The evolution of Software Development Method [31]

Traditional project development processes emphasize the importance of project plans and documentation. They try to identify all the requirements at the beginning of the project and control unexpected changes throughout the project. However, in the current dynamic business environment, major changes in requirements, scope, and technology are often out of the control of the development team. In [21], the authors identify that the question often is not how to minimize changes in a project but how to better handle inevitable changes throughout its life cycle. Agile methods present a possible solution to this dilemma through their strategies. The Agile Manifesto includes different agile methods which have been discussed or practiced for a while such as Dynamic systems

development method (DSDM) [27, 28], Feature Driven development (FDD) [29], Internet-speed development (ISD) [30] , Extreme Programming (XP) [31], SCRUM [32], Crystal [33], Pragmatic programming (PP) [34]. Among these methods, Extreme Programming is the most widely used agile methodology. Part of the reason can be explained in Figure 1.5.

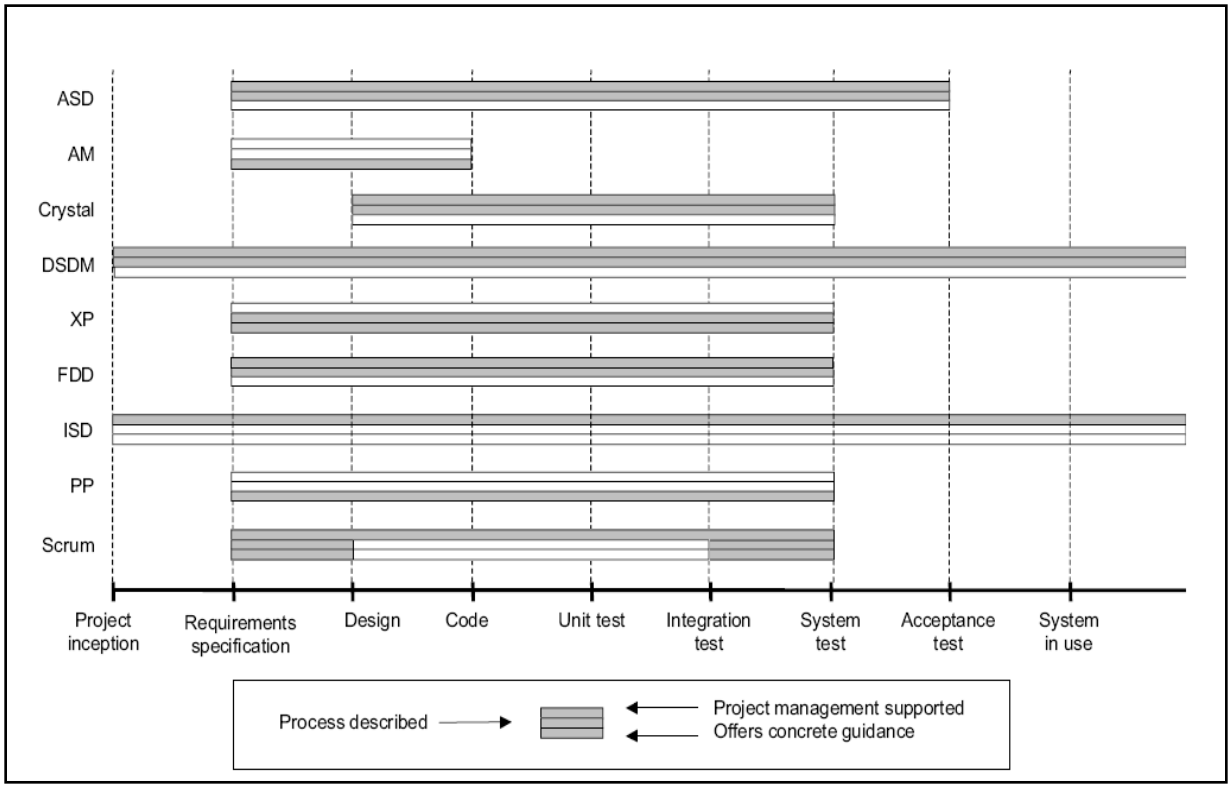

Figure 1.5: Comparing Agile Methods [35]

Figure 1.5 shows the comparison of these Agile Methods. In this figure, each method is divided in three bars which separately indicate its support for project management, a description of whether a process through which the software production proceeds is described pertaining to software development life-cycle analysis, and whether it provides concrete guidance separately from top to the bottom. A shaded bar indicates that the

method covers the perspective while an unshaded bar indicates lack of such support. The length of the bar shows which phases of the life cycle are supported by the method. We can see that each method has both similarities as well as differences. The reason XP is widely adopted can be described in following aspects: First, XP covers most of software development life cycle; second, XP supports situation appropriateness, meaning it can be tailored to suit the needs of individual projects; third, also the most important reason, while most of methods lack of real empirical support, XP is well supported by concrete experiences. Matching these advantages against the characteristics of GSD makes XP a viable approach in our research. We did notice that XP does not fully support project management. This is also a problem we target in our research.

In our context, XP is a set of twelve independent software development practices which include: Planning game, Small release, Metaphor, Simple design, Tests, Refactoring, Pair programming, Continuous integration, Collective ownership, On-site customer, 40-hours weeks, and Open workspace. It is initially designed for small teams working on projects of high degree of change. It is a discipline of software development based on values of simplicity, communication, feedback and courage. In [22] it is clearly illustrated: “The essence [of XP] truly is simple. Be together with your customer and fellow programmers, and talk to each other. Use simple design and programming practices, and simple methods of planning, tracking, and reporting. Test your program and your practices, using feedback to steer the project. Working together this way gives the team courage.” The twelve practices can be described as a cycle of activities as showed in Figure 1.6. The inner circle describes the tight cycle of practices carried out by programmers. The outer loop describes the planning cycle that occurs between customers

and programmers. The middle loop shows practices that help the team communicate and coordinate the delivery of quality software.

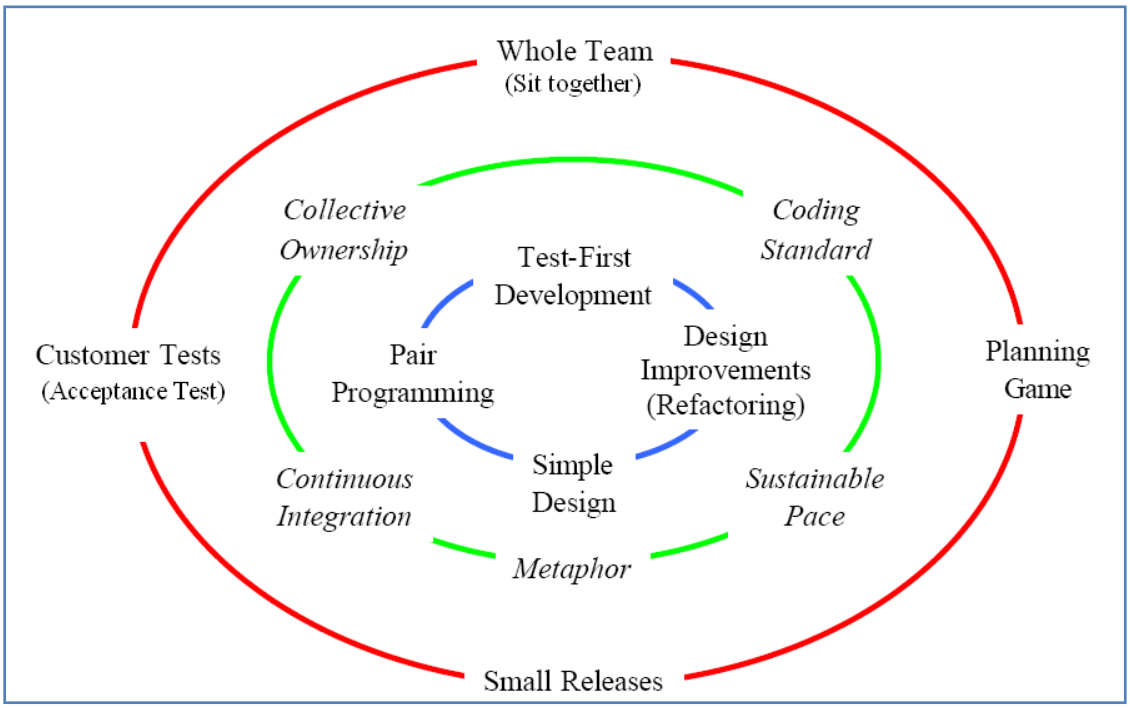

Figure 1.6: XP practices and the Circle of life [3]

Are all twelve practices suitable for distributed development? In the chapter 2, we discuss the possible problems when applying XP on GSD projects. Chapter 3 contains the investigation of related work. In Chapter 4 we examine XP practices, identifying which practices need to be tailored. Chapter 5 describes our methodology. In Chapter 6 we present prototype software, GSDXP, which is used to help applying our methodology. Chapter 7 contains our methodology validation. Conclusions and future work are discussed in Chapter 8.

CHAPTER 2

STATEMENT OF PROBLEM

Compared to traditional co-located projects, what is different about GSD? Herbsled writes, *"The fundamental problem of GSD is that many of the mechanisms that function to coordinate the work in a co-located setting are absent or disrupted in a distributed project.*"[5] In traditional co-located projects, team members working together have already built a common, recognized environment and a number of ways of coordinating work. They share the same view of the project by using a common vocabulary and process. The frequent informal and formal communication among team members ensures everyone has a clear picture of project. Also, misunderstanding is minimized when people share a common native language and cultural background. All these benefits of co-located projects diminish in GSD. Physical separation among project teams and members has diverse effects on many aspects. Among them, the most critical issue is the communication and coordination between the development sites which includes:

- **Decreased frequency of communication**. Instead of immediate face-to-face communication in a co-located project, people in a GSD project have to rely on communication media that are not always dependable. People are more reluctant to initiate the communication. According to a study by Tom Allen [6], people 30 meters away do not communicate more often than those are miles away.
- **Difficult to initiate communication.** When communication is infrequent, team

members often lose the vision of the project. The situation is worsened when more than one development site or organization is involved. “Who to contact about what” is the common question among GSD projects.

- **Miscommunication**. Although miscommunication results from communication media itself, the primary cause of miscommunication in a GSD project is cultural differences. When team members do not share a common native language, miscommunication happens much more frequently. Cultures differ on many critical dimensions, namely the need for structure, attitudes toward hierarchy, sense of time, and communication styles.
- **Increased communication cost- time, money, and staff.** Communication among remote sites incurs a cost not only in financial terms, but also in human terms. This needs to be considered in project budgets. Even though telecommunication is cheap, time to initiate the communication should also be considered. Sometimes each project site needs a special person in charge of coordinating with other sites.
- **Time difference**. When a project site is located in a different time zone, especially one more than eight hours away, person-to-person communication becomes logistically difficult. This problem also increases the possibility of miscommunication and slows down the project progress.

GSD requires a prompt response to changes, which is hard to fulfill because of the communication gap brought by the reasons listed above. XP is reported as one of the best suitable development method for GSD projects because it is a discipline of software development based on values of simplicity, communication, feedback, and courage. The simple and agile nature of XP enables it meet the dynamic requirement of GSD projects.

But, XP also emphasizes frequent customer-centered communication that GSD can not promise. Moreover, in a GSD project in which more than one organization is involved, across-site coordination is another issue that needs to be considered because it is also impacted by communication deficiency. How to blend the XP and GSD projects together, while at the same time maintaining agility and alleviating the communication impedance to improve the project success is a vital issue, and is the focus of our study.

As stated earlier, XP does not provide concrete guidelines for project management. The practices provide the guidance for specific activities. There is no method to glue them together as a whole. How are user stories well managed as they grow in number? How are iterations and releases managed when the project is growing? How are project resources and human resources managed so that programmers know where to find what? Our research is focused on solving above problems.

CHAPTER 3

LITERATURE SURVEY

Many studies have been conducted about Global Software Development, focusing primarily on distance, which is a major factor in communication and coordination problems. Scientific research and empirical studies of Extreme Programming and its applicability in different environments are also available. Both successful experience and lessons are reported from the GSD projects practicing XP. The problems and challenges presented by this research provide the theory basis for our study.

3.1 Global Software Development Challenge And Approaches

There is a wealth of literature that notes the challenges in Global Software Development. According to *Challenges of Global Software Development* [7], difficulties include interdependencies among distributed work items, difficulties in coordination, difficulties in dividing the work into modules that could be assigned to different locations, conflicting implicit assumptions that are not noticed as fast as in collocated work, and communication challenges.

Erran Carmel and Ritu Agarwal propose three tactical approaches to alleviate the distance influence in *Tactical Approaches for Alleviating Distance in Global Software Development* [8] including reducing intensive collaboration, reducing the culture distance, and reducing the temporal distance. This research suggests that collaboration intensity decreases when a foreign entity (an organization that is in a different nation from its parent) assumes the low complexity task or full responsibility for a product. There are

four ways to reduce the culture distance. The first one is called the 75/25 rule which means that 75 percent of project work occurs offshore while the remaining 25 percent occurs onshore in order to maintain the closeness to the customer through face-to-face communication. Secondly, open internal-to-the-firm foreign software centers can reduce the organizational culture distance because these centers are trained in the corporate methodologies and policies and have the access to all the organization resources. The third method is that of a project manager or key executive acting as a culture liaison to travel back and forth between the key stakeholder sites. In doing so, they facilitate the cultural, linguistic and organizational flow of communication and bridge cultures, mediate conflicts and resolve cultural miscommunications. The last one includes such things as giving a language course to employees to reduce the impact of cultural distance brought by language. Carmel and Agarwal also suggest using synchronous communication to reduce the temporal distance. While this study makes sense generally, there are some situations in which these four approaches are hard to implement. For instance, letting a foreign entity do the full project development may be too risky because of its distance from the target market. A frequently traveling project manager is not efficient because all the communication relies on one person. Language training can be time-consuming. Also synchronous communication eliminates the advantage of follow-the-sun type work that requires large difference in time zones.

In *Stakeholders in Global Requirements Engineering: Lessons learned from Practice* [9], Daniela Damian suggests a relationship of organizations in Global Software Development as Figure 3.2.

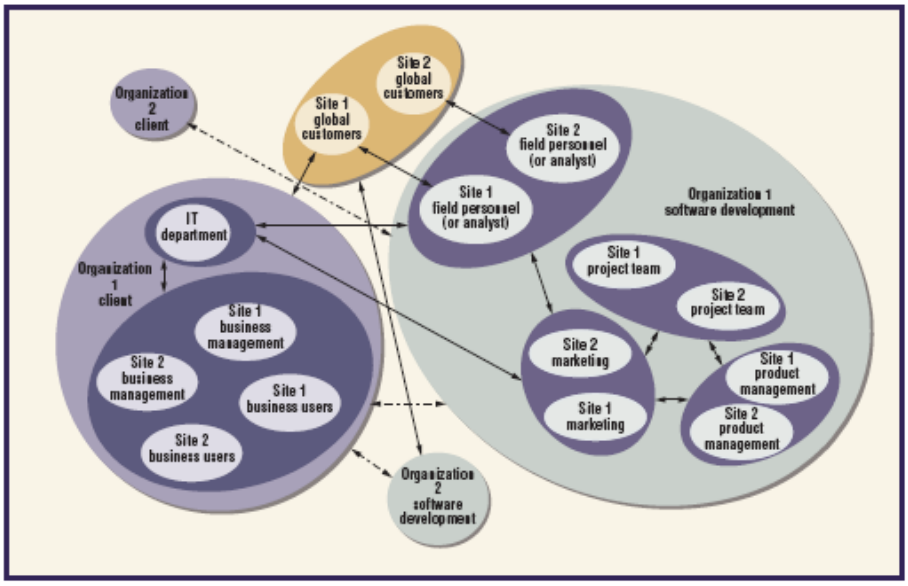

Figure 3.2: Global Requirement Engineering main stakeholder categories [9]

The stakeholders' ability to communicate globally is challenged by GSD in three ways. First, designers have less opportunity to seek out relevant knowledge from the multiple stakeholders, making knowledge sharing and integration across sites and functional groups problematic. Second, process differences inherent in inter-organizational partnerships lead to difficulties in aligning requirement engineering processes and supporting tools, preventing management practices from being effectively implemented across sites. Third, lack of informal communication in global teams negatively impacts relationship building and inadequate channeling of changes to requirements across sites leads to difficulties in coordination.

Damian also suggests two sets of strategies to alleviate these challenges. The first is to support interorganizational structures by defining a clear organization structure with communicating responsibilities for the distributed projects; establishing peer-to-peer links at all levels across distributed sites; partially synchronizing interorganizational process by performing frequent iterations and deliveries; and establishing culture liaisons. The

second is to support communication practices by maintaining open communication lines between well-defined stakeholder roles and frequently informing and monitoring progress on commonly defined artifacts. Although her study is mainly focused on requirement engineering because it has the highest communication density, her findings can be extended to global software development in general.

3.2 Agile Methods And Extreme Programming

Since the emergence of Agile Methods, both theoretical and empirical researches have been conducted within this field. Kahkonen and Abrahamsson build the theoretical base for Extreme Programming in their paper, *Digging into the Fundamentals of Extreme Programming* [25]. They discuss the rational of practicing XP using an acknowledged scientific framework designed to explain how knowledge is created when several communities are present. Their 5-A model [26] defines three modes of knowledge creation: articulation, appropriation, and anticipation; and two processes: accumulation and acting. They observe that when XP is analyzed using the 5-A model, most XP practices are enhancing knowledge creation through immediate (or frequent) and mutual articulation and appropriation. The practices help to accumulate knowledge by utilizing external cognitive tools, such as concepts, words, language, signs, tools, documents or social practices. Anticipation is done for short intervals only and XP practices are action-oriented. While not all the XP practices fit into the model, this analysis gives a good initial understanding and a more solid scientific basis for further research.

In *Extreme Programming: A Survey of Empirical Data from a Controlled Case Study* [37], Abrahamsson and Koskela report on a survey of the empirical data obtained from a controlled case study on Extreme Programming in practical settings. According to

their data, the XP practice of "user involvement" in the system development process has a positive impact on subsequent system adoption and use. The majority of the customer's involvement is required on the planning game and acceptance testing during the project. Even though the customer does not develop automated acceptance tests, the mere presence of the customer is highly valued by the development team. They also found that user involvement is one of the reasons for low defect density because customer representatives collected suggestions and bugs report frequently, thus generating the feedback for development team. Customer involvement also plays a positive role on customer satisfaction even when delay happens because involvement minimizes surprise. This argument is supported from another perspective as well. *In Recognizing and Responding to "Bad Smell" in Extreme Programming* [36], Elssamadisy and Schalliol note that they "failed to push the customer hard enough early in the process to be an actual partner in the planning and acceptance of the development". They argue that the customer must provide honest and substantial feedback from the very beginning of the development process. Abrahamsson and Koskela follow up on this by noting that a one-week release cycle to end-user testing is seen as disturbing to users and is not appreciated.

Delivering a system that satisfies customer requirements and which is on time and within budget with few defects is the ultimate goal of any software development activity. When customers are not satisfied there is a gap between customer expectation and experiences. The principles behind the Agile Methods include specific strategies for satisfying the customer through involving the customer regularly, relying on face-to-face communication, responding to evolving requirements and providing early and regular feedback. In XP, there is an explicitly-defined role for the customer in development team

so customer can work with developer closely. Communication impacts the customer satisfaction as well. In *Customer Relationships and Extreme Programming* [39], Grisham and Perry examines XP from the perspective of customer satisfaction. They point out that with its high degree of communication, rapid feedback, and constant adjustments, XP should prevent expectations gaps from becoming unmanageable; but this depends on the quality of the communication between the customer and the development team. They also mention that there is a risk of high transparency that the customer could perceive daily chaos of the development process. Risky situations such as schedule slippages and technical difficulties are more difficult to hide from customer. More research on overcoming communication obstacles when applying XP on GSD projects and the level of customer involvement is needed.

Even though XP practices are designed for small development teams, does it fit large-scale projects? Proponents of XP claim that using this method has advantages over traditional approaches including higher team productivity, lower management overhead, and better customer satisfaction. However, the applicability of agile approaches is constrained by several factors such as project size and type, experience level of project personnel, and access to committed customers. In *Get Ready for Agile Methods, with Care*[39], Boehm argues that agile methods are difficult to scale up to large projects because of the lack of sufficient architecture planning, over-focusing on early results, and low levels of test coverage. He also recommends that agile methods not be used in mission-critical software development. However, large projects also face constantly changing business environments that can be addressed by agile methods. In *How extreme does extreme programming have to be? Adapting XP practices to large-scale projects*

[40], Cao et al argue that Agile Methods such as XP can be adapted to large-scale projects. They propose some general guidelines on tailoring agile development methodologies to make them suitable for the development of large, complex software system. The guidelines include seven practices: Designing upfront which combines design upfront in traditional approaches with agile practices such as short release, pair programming and refactoring; Short Release cycles with a layered approach; Surrogate customer engagement; Flexible pair programming that applies pair programming only in the analysis, design, and test phases; Identifying and managing developers; Reuse with forward refactoring to yield reuseable systems; and Flatter hierarchies with controlled empowerment to improve communication between stakeholders and increase productivity. Cao et al also point out that organizations need to be very careful at tailoring lightweight methodologies like XP to ensure their suitability.

XP approaches have been successfully applied on various software development projects, but should we follow all the practices exactly as they are described? *In Recognizing and Responding to "Bad Smell" in Extreme Programming* [36], Elssamadisy and Schalliol note "The software development process (XP) that purportedly 'embraces change' must itself embrace changes to its own specific implementation as needed it is to succeed." In this article, the authors describe a large software development project that used a modified XP approach after a more traditional approach proved ineffective. They identify poor XP practices and discuss the solution implemented to correct them. They conclude that XP is a valuable and effective approach to software development so long as one recognizes that 1) it cannot succeed without conscientious

participants, and 2) it must be adapted as necessary for projects that do not fit the "small team" limits recommended by its founders.

3.3 Applying Agile Methods On Global Software Development

In traditional software development, many projects are divided into different modules, integrating them in "a big bang" at the end. This approach is challenging in GSD because the integration may cause serious problems that may not be expected. In *Leveraging Resources in Global Software Development* [10], Battinet al suggest an incremental integration solution that is based on clusters and shared incremental milestones to avoid "big bang" integration. This strategy was tested successfully in Motorola successfully. In *Surviving Global Software Development*[11], Ebert and De Neve also report successfully using incremental development in Alcatel. Each increment is developed within one dedicated team and the project progress is based on tracking successfully integrated and tested customer requirements. The study reports that increments toward a stable build are proven to be one of the key success factors. Globally-applied continuous builds improve the project cycle time as well. There is also evidence that even very frequent builds are possible in distributed development. In *Daily Build and Feature Development in Large Distributed Projects*[11], Karlson et al report their successful experience of using very frequent builds and feature-based development in Distributed projects.

In *Internationally Agile*[13], Simon suggests that an iterative model may fit into internationally distributed projects to help alleviate some of the problems brought by distribution. Frequent integration and test phases enable problems to be solved early, thus avoiding serious problems at the end of the project. Iterative development with frequent deliveries also provides good vision to the project, giving team members and customers

an accurate sense of the project progress. In *Using Agile Software Process with Offshore Development*[14], Fowler also points out that a continuous integration and test process flushes out many integration problems quickly so they can be fixed before they become hard to find. He discusses suitable iteration lengths for GSD projects and concludes that a two-week interval is the minimum because of the communication overhead. Incremental integration and frequent deliveries are the core practices in agile methods. Both Fowler and Simons point out that the major benefit of using an agile method in their project has been the fast response to changes and fast delivery of business value, benefits which have outweighed the challenges of global distributed development. Thus, it seems that at least agile method principles are suitable for GSD projects.

A few studies have presented the use of Extreme Programming in distributed software development. In his article Fowler [14] gives a detailed discussion of his experience in using XP in projects distributed in US and India. Both onshore and offshore teams using Agile/XP practices and agile communication principles were applied in these highly distributed projects. The successfully-used practices include continuous integration, sending business-oriented ambassador to the offshore team, using test scripts to help understand requirement, using regular builds to get feedback on functionality, using regular short status meetings, using user short iterations, using an iteration planning meeting that is tailored for remote sites and separate teams by functionality not activity. In two articles *Extreme Programming In Global Software Development*[15] and *Agile Methods handling Offshore Software Development Issues*[16], Yang et al and Nisar and Hameed, respectively, discuss their experience in using XP in GSD projects in which offshore teams collaborate with onshore customers. The projects mentioned in both

papers describe development teams located in China and Pakistan, with customers located in the US, UK, etc. The reason why Yang's project adopted XP was to reduce the communication delay and improve communication quality, which he identified as the major obstacles in GSD projects. Nisar and Hameed report eight XP principles they followed and benefited from. These principles include: client satisfaction should be on the top most priority; always welcome the change and incorporate the change usually in next iteration; frequent development iterations (maximum 2 weeks); "working software" is the primary measure of progress; frequent communication with offshore clients (minimum once in two days); continuous attention to the technical excellence; user of pair programming for critical project modules and sections; iteration planning. Both Yang et al and Nisar and Hameed conclude that the principles of XP have been proven successful in their projects. Yang et al reported that their project was completed on time and with a cost-saving of at least 60% compared with doing the project entirely onshore. Nisar and Hameed's projects gained a 100% rate of client satisfaction. The application of XP on large distributed projects also has been reported in Karlsson et all's paper, *Daily Build and Feature Development in Large Distributed Projects*[17], and Farmer's *Agile Development in a Large, Distributed Team*[18], although they restricted their use of XP to continuous builds and unit tests, small releases, continuous integrations, and automatic testing. Both papers found that applying an agile process is useful but hard to implement due to the problems of GSD introduced by distance discussed above.

In general, all these reported experiences about the use of XP in global distributed projects are successful according to the respective authors. This leads us to conclude that XP process can benefit the GSD projects either in communication or client satisfaction to

improve the software quality, even though we need to define carefully how we implement these principles in practice when handling offshore projects.

3.4 Supporting Tools

Besides theoretical research, there have been some tools developed to help application of Agile Methods. Some of them come from project management perspective. Some are focused on one or some of practices of Agile Methods.

For the tools targeting a single agile technique, we found a prototype user story software tool called DotStories purposed by Rees in his paper *A Feasible User Story Tool for Agile Software Development* [24]. The author introduces DotStories, a web-based tool that can be applied in distributed team. DotStories offers any number of web sites with the intention that each web site corresponds to a single software development project. Each web site contains a collection of user story groups or web pages. A web page contains any number of user stories. Each user story page contains the basic information required by XP practices. User stories are categorized as Complete User Stories, Future Stories, and Archived Stories. They can be browsed in different mode for user-friendly purposes. DotStories is mainly implemented as a website containing a large body of Jscript functions embedded in a series of HTML pages with some.asp pages to manage the XML files on the server. They are accessible from anywhere using Internet Explorer 5.5 or a later version. We found DotStories to be limited in its applicability on big projects. For example, the categorization schema makes finding user stories difficult when there is a large inventory of user stories. Also it does not support multi-site development.

Another user story tool with much wider capabilities is called Storm User Story Tool. It is under development as an Open-source project at Sourceforge.net [23], and provides features such as access control within and between accounts; release management with user stories sorted by various properties; linking remote files to a user story; and user story version control. These features are very impressive but the tool itself suffers from a complex interface. This could drive users away from the simple, lightweight nature of XP. As with DotStories, it does not support multi-site development.

There are some tools designed to solve project management difficulty. In *Enabling Collaboration In Distributed Requirements Management*[19], VibhaSinha, BikramSengupta, and Satish Chandra introduce a tool by IBM called EGRET, which is an Eclipse-based global requirements tool for distributed requirements management. EGRET aims to support change management, knowledge management, awareness and informal collaboration in teams that subscribe to the communication about a particular requirement. The potential users include business analysts and architects who interface with the customer and elicit high-level requirements, as well as system engineers; testers and other members of distributed development teams who help refine these requirements and define validation criteria for them. EGRET is based on an Eclipse client communicating with back-end repositories. It uses MySQL as the repository for data, CVS as the version-controlled repository, and an experimental collaboration server developed by IBM for synchronous communication. EGRET interface consists of a set of views: Artifact Explorer shows the hierarchical structure of project requirements; Communication Record lets user initiate conversation or accesses all the conversations they participated in; Project Stakeholders lists all stakeholders along with their roles and

status; and Traceability shows the requirement's traceability. EGRET was tested on 12 practitioners from three projects and was proven to aid distributed teams in supporting informal collaboration, managing changes, promoting awareness and managing knowledge. The authors also suggest some guidelines when building such tools:

- The tools should be able to plug into existing collaboration mechanisms.
- User authentication and access to various collaboration services should be uniform.
- The tool should be interoperable with other tools belonging to subsequent parts of life-cycle
- A web interface is essential for the tool to be widely accessible.

This research validates that a deep integration of appropriate collaboration support with a requirement management tool can greatly aid distributed teams. The authors also point out that while the preliminary evaluation of EGRET is encouraging, it involved only a few practitioners. There is no proof that EGRET fits other projects outside of IBM. Moreover, this tool is focused on requirement engineering instead of the whole life cycle. Interoperability with other tools in different parts of life cycle needs to be considered when choosing it for requirement engineering management. Tool that supports the whole software development life cycle is more desirable.

Another project, which is still under development by IBM, is called Jazz. According to the IBM website [20], "Jazz is an IBM Rational project to build a scalable, extensible team collaboration platform for integrating work across the phases of the development lifecycle." As a provider of collaborative capabilities to development teams, Jazz breaks

new ground by in incorporating collaborative tools into the IDE instead of using stand-alone collaboration tools such as instant messenger. Jazz also can:

- Handle connections to the server infrastructure to support messaging and source control
- Place hooks in Eclipse to track developers' interactions with source code and source control
- Integrate the user interfaces that developer use to communicate with each other

The goal of Jazz is to find a way to address the needs of a broad spectrum of end users using different processes, which makes it very powerful. But it is not customized for any specific software process such as XP, even though some XP principles are included such as iterations; consequently some of functionality is not necessary for XP users. Our Eclipse plug-in is built specifically for XP. Part of it will be patterned after Jazz, but will be lighter in weight.

CHAPTER 4

EXTREME PROGRAMMING EXAMINATION

4.1 Introduction

In this chapter we examine the XP practices within Global Software Development context and discuss which practices can fit into GSD and which cannot. We identify aspects of XP that are necessary for adapting it to GSD projects. We propose a methodology for tailoring XP to fit into GSD.

4.2 Xp'S Practices Benefit Communication

Communication is important throughout the entire software development lifecycle. There are several kinds of communication we needed:

- Communication between project manager and customers;
- Communication between developers and customers;
- Communication between developers and project manager;
- Communication between developers;
- Communication between customers.

Communication is also one of the core values of XP discipline. XP's practices focus on improving these kinds of communication. Table 4.1 shows a collective generalization of what XP practices benefit what kind of communication (without considering GSD). As we can see most of XP practices can benefit from communication. When it comes to the globally distributed software development, some of the benefits become hard to achieve

due to the kinds of reasons stated earlier.

Practices	Benefit
Planning game	Benefit communication between project manager, developer and customers.
Small release	Benefits rapid feedback between developer and customers.
Metaphor	Provides easy understandable communication platform for developers, project manager and customers.
Simple design	Facilitates communication within developers, and between developers and project manager.
Tests	Provide rapid feedback between customers and developers.
Refactoring	Makes it easy to communicate between customers and developers.
Pair programming	Provides instant communication between paired developers.
Continuous integration	Provides developers with rapid feedback on the quality of the code.
Collective ownership	Benefits communication between developers
On-site customer	Benefits communication between project manager, developer and customers.
40-hour weeks	Not identified
Open workspace	Benefits communication between developers, and developers and project manager.

Table 4.1: XP practices benefit communication

We exam the practices which may be problematic to practice under GSD environment. We identify which aspect needs to be tailored to fit in GSD background as the theory basis for the methodology we propose in the next chapter.

4.3 Xp Practices Examination

4.3.1 On-Site Customers

We first examine on-site customer practices because it provides the major premise for our following discussion. XP recommends that a customer sit with the team full time

during the entire life cycle. This practice requires that the customer have a thorough understanding of the desired software. In [36], the authors identified this "bad smell" when practicing XP as well. They noted that the customer must be coached sufficiently to provide honest and substantial feedback from the very beginning of development process. In most cases, one customer cannot fully provide requirements to the development team; there will be multiple customers involved. When the project is globally distributed, it is costly to set up on-site customers. Plus, there are other issues involved, such as international visa, travel time, etc. A timely customer presence, especially when an emergency happens, can hardly be guaranteed. When the project is distributed across multi-sites, customers have to travel between the sites, which decreases productivity and increases costs. As a consequence, a tool that can provide the customer's virtual on-site presence is needed to apply this practice to GSD projects. E-mail, Instant Messenger, and conference calls are good options to help facilitate communication among customers and development teams that are globally distributed. For teams more than five time zones away, telecommunication can be held at the beginning and the end of each release and iteration, or as necessary. Even though e-mail is less efficient than face-to-face communication, it can be responded to within 24 hours. Moreover, when there is a language difference, people tend to feel more comfortable communicating in writing than through oral means.

4.3.2 Planning Game

In the planning game, customers decide the scope and timing of the release based on estimates provided by the developers. Developers implement in any one iteration only the functionality demanded by the stories. In XP, there are two kinds of planning games:

Release Planning and Iteration Planning. After customers finish editing the user stories, a meeting is set up to create the release plan which lays out the overall project. The idea of this meeting is for developers to estimate how long in programming days it will take to finish each user story, and for customers to then decide which user stories to complete. The release plan is then used to create iteration plans for each individual iteration. In this way, every team member has clear goal of project progress. GSD throws a wrinkle into this process. First, as we stated before, when there is no on-site customer present it is difficult to organize a release plan meeting. A conference call may be the best option, but still has its limits. Especially at the beginning of project, a meeting is inevitably going to last longer than it is in the latter part of project because of negotiation between development team and customers, such as requirement clarification. Another problem is the more stakeholders involved, the more unorganized the negotiation tends to be. A role such as a project manager for each development team is necessary to ensure the meeting goes smoothly. The project manager collects the estimate from developers. Discussions or meetings can be conducted before release plan meeting if the project manager feels the developers' estimates are problematic. After gathering all information, the project manager attends the meeting as the representative of team to negotiate with customer. When necessary, the project manager brings back the customer's feedback to discuss with specific developers. This systematic communication enables release plan meeting to be more organized and efficient.

Another communication issue that needs to be considered is the format of the user story. In traditional XP practice, the user story is written on a physical card. Together, developers and customers move the cards around on a large table to create a set of stories

to be implemented as the first (or next) release. In the GSD environment where there is no on-site customer, a virtual large table that gives the customer and the development team synchronized access to virtual user story card is needed. A tool support this functionality will benefit both communication and project management.

4.3.3 Small Release

The development team needs to release frequent iterative versions of the system to the customer. This is critical in getting valuable feedback in time to have an impact on the system's development. Declaring the introduction of important features shortens the implementation time. How to control the releases in GSD needs to be further considered. When a release plan includes user stories from multiple development sites, a methodology that makes the plan easily understandable for all development teams and customers helps relieve the misunderstanding problem we stated before.

4.3.4 Simple Design

XP emphasizes keeping things as simple as possible. It frees the developers' from the requirement of heavy documentation in an effort to focus more attention on the design itself. Simple design eases communication among developers, and between developers and the project manager. We suggest using a standard design language such as UML diagrams for better communication in GSD projects especially in cross-site communication. As we stated before, developers tend to have miscommunication problems when they come from different cultural backgrounds. The frequency of cross-sites communication drops for the same reason as well. A standard design methodology such as UML remains a design simplicity that all developers understand. "Where to find who" is also a common problem in GSD. A methodology that helps development team

members to easily locate the partner they want to communicate with should also be applied in GSD project.

4.3.5 Testing

The acceptance test for each user story is conducted by the customers and a test score is given after that. This practice provides fast feedback between the development team and the customers. This convenience should not be jeopardized by the communication gap in GSD projects. What is the convenient way to transmit the feedback from customer to development team without introducing too much overhead? How to make sure developer will be notified timely without interference from the time and special distance? A methodology that provides timely notification of test feedback is required here.

4.3.6 Collective Ownership

In XP, the practice of collective ownership means that every programmer improves any code anywhere in the system at any time. It benefits communication between developers because, in this way, everybody can learn from each other. For co-located development teams, it is easier to trace back to the author who made the change when the defect is introduced by such change. Co-owners can initiate the communication easily. There is also no copyright issue involved. For globally distributed development teams, it may cause legal issue if one site modified the source code of other side that belongs to other company. Even there is a mutual agreement on source code ownership. It is hard for developers to accept the fact that a stranger from other company can make changes to software they authored. The traceability of changes across sites decreases significantly as

well. Therefore, we suggest collective ownership to be practiced within each development site, but not practiced among development sites.

CHAPTER 5

GENERAL APPROACH

We propose a methodology for adapting XP practices to a GSD project by using a project management tool. We identify the aspects of developing this kind of software that are necessary for global distributed development. We are mainly targeting solving the communication problem and weak project management support of XP as stated in Chapter 2. The methodology is illustrated in a project management perspective because we believe a good management can facilitate systematic communication.

5.1 Introduction

In a GSD project, an important thing is to make sure globally-distributed teams are on the same page during the project. A project management tool that supports information sharing and promptly information update is required.

5.2 Project Information Overview

5.2.1 Description

Awareness is a problem we need to tackle in GSD. Customers normally do not involve themselves in development but they always want to know the project status so that they can adjust their plans. If they observe that the project is going faster than scheduled, they can add more user stories in the next release. A delay also can be detected at an early stage so customers will not be surprised. This helps to improve customer satisfaction. The project manager wants to monitor the site's progress as well

concrete project progress data helps in negotiating reliable goals with customers. Team members normally focus on their own module and, as a consequence, lose the big picture of the whole project. The system shall provide users with overall project information and the access to detailed information. The system shall provide visibility to the project progress of each project site. The detail project information shall be organized in the site manner.

5.2.2 Functional Requirement

REQ1-1: The system shall display the project name the user is working on and current available information.

REQ1-2: Project abstract information shall be displayed appropriately.

REQ1-3: The access point to detailed project information shall be displayed.

REQ1-4: The system shall provide appropriate information about Team Members, User Stories, Release Plans and Iteration Plans for each site.

REQ1-5: Both graphic and text project release plan, if available, shall be displayed

REQ1-6: Users shall be able to view team member information grouped by team properties.

REQ1-7: Users shall be able to view user story information grouped by its properties.

REQ1-8: Users shall be able to view release plan information grouped by its properties.

REQ1-9: Users shall be able to view iteration information grouped by its properties.

REQ1-10: Project information can only be modified by the user who has Edit privilege, otherwise it is only viewable.

5.3 Project Team Member Management

5.3.1 Description

A team member management function can help all project members easily locate who is where and who is working on what, which caused the "hard to initiate communication" problem of GSD. Besides distance, people normally find it difficult to communicate with somebody they do not know. This barrier especially exists in cross-site communication among different companies. A personal picture will help remove this barrier. The system shall record every team member's information including:

- First, Last Name
- Personal image
- Site
- Role
- Assigned User Stories

The system shall let project team member information be accessible to the entire project team. Only those who have Add/Update/Remove privilege may update the list.

As we stated before, instant messenger tools can help solve the "Decrease frequency of communication" problem between distributed team members. An embedded instant messenger in the system will be convenient for users so they do not need to rely on third-party software. Therefore, the system shall allow users to initiate conversations whenever they want. A copy of the conversation's content shall be saved automatically as a record for future use.

When distributed teams are more than five time zones away, team members from different sites have small time overlaps. A messenger that can display messages received

received when the user is absent is very helpful to speed up project progress.

5.3.2 Functional Requirements

REQ2-1: The system shall display the Project Team member information for all team members.

REQ2-2: The system shall let the team member who has update(add/remove/update) privilege to update project team member information

REQ2-3: The system shall notify all project managers when team member information is updated

REQ2-4: The system shall let a team member to initiate a conversation with one or more other team members.

REQ2-5: The system shall let a conversation participant save a copy of the conversation content.

REQ2-6: The system shall let the conversation participant choose the person they want to inform about the conversation content.

5.4 Project Sites Management

5.4.1 Description

How project information is organized influences project management and communication significantly. Imagine if there is only one database table with hundreds of user stories how difficult it is to dig out who is working on which story and what is its status. When a project spans multiple development sites, it is more complicated to maintain a big picture of project progress for customers and project manager. In our approach, besides project information overview, we suggest organizing project resources in a site fashion. Each site contains its own people, user story, and release/iteration plan.

It is editable for each team member of the site, and viewable for other sites. Development site information not only enables customer to observe each site's progress without being physically present, but also facilitates team members' communication and cross-site communication by providing a big picture perspective of project. It contributes to solving almost every communication issues in GSD. To implement this functionality, the system shall record information include at a minimum:

- Site basic information (Location, Development team size, Development velocity)
- Team members
- User stories assigned to
- Project schedule (release plan)
- Iterations
- Releases

By browsing the above information, customers can monitor project progress remotely so as to save communication cost, time, money and staff. Because information is available on the server, accessibility is not limited by time zone differences. Project managers and developers also can get an idea where they are and who is doing what during the project.

5.4.2 Functional Requirements

REQ5-1: The system shall organize project information by site

REQ5-2: The system shall let site information be visible to the entire project team

REQ5-3: The system shall be able to distinguish site member and site visitor

REQ5-4: The system shall be able to distinguish which site information is editable based on the user's role

REQ5-5: The system shall inform site members of any project updates and notification of the site they belong to

REQ5-6: The system shall allow communication between team members and different sites

5.5 User Story Management

5.5.1 Description

When practicing XP, the user story is a part that has high communication density. From the planning game to implementation to test, user story information is needed in most XP practices. In the planning games, developers need to estimate the programming time for each user story. At the planning meeting, customers and project managers discuss them to set up the release plan and the iteration plan. The clear images of which user stories have been completed and which are incomplete at what priority is crucial. In design, developers may need to know who is working on a user story that is similar to theirs, who is the author of the user story when the requirement is not clear. In a small release, project members need to be aware which user stories are included in each release. In testing, the developer needs to get feedback from customers if the user story he is in charge of is accepted or has failed a test. There is a lot of communication required in a background full of communication obstacles. The crucial problem is when an on-site customer is not available. We need a methodology that can bring customers, project managers and developers into one virtual room, thus efficiently accomplishing the XP practices with as little face-to-face communication as possible. Information can be transmitted timely and accurately.

We propose two key approaches here: “User Story Inventory” and “User Story Status”. There are two kinds of “User Story Inventory”. A “Project User Story Inventory” consists of all the user stories created by customers. Since project information is organized by sites in our approach, a “Site User Story Inventory” lists all the user stories assigned to this development site. Generally, every user story has following properties:

- Unique ID
- Name
- Description
- Author
- Estimate Programming Day
- Actual Programming Day
- Developer
- Priority
- Iteration ID (which is assigned to)
- Release Version (which is assigned to)
- Status

“User Story Status” indicates the status of a user story during project. It can be one of the following:

- Not Started – User story is created by customer
- Assigned – User Story has been assigned to specific developer
- In Progress – User story is in the implementation phase
- Complete – User story implementation is complete

- Accepted – User story is accepted by customer in acceptance test
- Test Failed - User story is not accepted by customer in acceptance test
- Released – User Story has been released

Events	User Story Status
Customer creates an user story	Not started
Developer picks up user story	Assigned
Developer starts implementation	In progress
Developer finishes the unit test	Complete
Passes customer acceptance test	Accepted
Failed at acceptance test	Test failed
Release	Release

Table 5.1: Change of User Story Status

"User Story Inventory" and "User Status" help customers and project managers easily track every user story during projects. Table 5.1 shows the changing of properties for one user story along the project progress. The customer first creates a user story, the Name, Description, Author fields are filled out and the status is set to "Not started". The user story is saved in the "Project User Story Inventory". The customer assigns user stories to each development sites. This information is saved in the "Site User Story Inventory". The project manager then checks the inventory and lets developers pick user stories. The project manager fills out the "Developer" field and changes the User Story Status to "Assigned". Each developer gives an estimate in programming days of how long the story will take to implement. The user and project manager pick stories that will go into the release plan and the iteration plan. The developer sets the Status to "In progress" at the beginning of implementation and sets to "Complete" after unit test. Customers test the user stories and mark them as "Accepted" if each passes its respective acceptance test, or

"Test Failed" if not. The "Release" status is used if a user story is included in a release version. The "actual programming day" gives the project manager a means of calculating the project velocity so as to help set up the next release/iteration plan. In this way, heavy communication becomes a series of systematic processes that are not impacted by regional and cultural difference. Simple activities and standard description give less opportunity for misunderstanding. Customers and development team members can track every user story at any time, thereby decreasing the time zone distance and increasing awareness.

5.5.2 Functional Requirement

REQ3-1: User Story shall contain all the properties listed above

REQ3-2: The system shall display the "Project User Story Inventory" for the whole project team

REQ3-3: The system shall display the "Site User Story Inventory" for each site

REQ3-4: Only the customer is allowed to add/remove a user story

REQ3-5: Only the project manager are allowed to assign a user story

REQ3-6: The developer can only update the estimate and actual programming day and status of task assigned to him

REQ3-7: User stories can only be viewed by users from other development sites

REQ3-8: The system shall display stats on user stories for the whole project

REQ3-9: The system shall display stats on user stories for the whole site

REQ3-10: The system shall save the discussion under specific user story

REQ3-11: The system shall allow user stories to be assigned to only one iteration

REQ3-12: The system shall allow users to update user story status.

REQ3-13: User Story Status options shall be customerized based on user role.

REQ3-14: The system shall allow users to break user stories into tasks when needed

REQ3-15: The system shall record UML design diagrams and test cases for each user story.

5.6 Project Release Management

5.6.1 Description

Unlike traditional development approaches where software modules are combined together at the end of development, XP requires iterative development and frequent releases to deliver a runnable system to customer as early as possible. Customers pick user stories for each release at release planning meeting. In a globally distributed development environment, release management is more complicated than single sites, because a project release may consist of user stories developed by different sites. Without a systematic management approach, the release plan is influenced by conflicting site development schedules. To control this chaos, we propose using two kinds of release plans: Project Release Plan and Site Release Plan. The project release plan is for the whole project. It consists of release plans for each site. The approach is described in Figure 5.1.

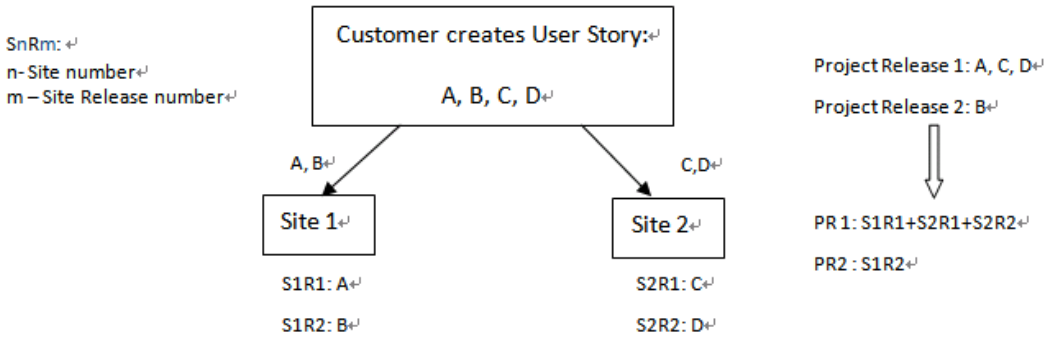

Figure 5.1: Project Release Plan and Site Release Plan example

In this example, the customer first creates four user stories and assigns them to Site 1 and Site 2 separately. The customer and project manager settle the Site Release Plan for each site based on business needs and the user stories' priorities. The customer decides which site releases they want to put into a Project Release. From the example above, the customer wants user stories A, C, D as first project release. It includes Site Release S1R1, S2R1 and S2R2. Using this approach avoids schedule conflicts between sites 1 and 2. Development tasks are clear to customers and development is easy to manage.

The system supports above approach with two kinds of release plan – the Project Release Plan and the Site Release Plan. The system shall provide customers an easy-to-user platform for creating release plans. The system shall also make the project schedule accessible to the entire team, editable to those who have privilege.

5.6.2 Functional Requirements

REQ5-1: The system shall record both the Project Release plan and the Site Release plan

REQ5-2: Project Release Plan and Site Release plan shall be visible to all team members

REQ5-3: The system shall allow the customer to create the Project Release Plan based on the Site Release Plan

REQ5-4: The system shall allow the customer to create the Site Release Plan based on user stories assignment.

REQ5-5: The system shall prompt the user when a Release Plan conflict happens.

REQ5-6: The system shall let team members with update privilege to change the schedule

REQ5-7: The system shall record the update history for the Project Schedule.

REQ5-8: The system shall notify all team members when a release plan is updated

5.7 Project Iteration Management

5.7.1 Description

After the Release Plan is set up, the customer and project manager will pick which user stories from the Release Plan will go into the Iteration plan. An iteration plan should not contain the user story from later release plan when the user stories of earlier release plan are still available. When the iteration is complete, a project velocity should be calculated. Only customers and the project manager have the update privileges to edit (Add/Remove/Update) an Iteration Plan. Iteration information is viewable to all team members so as to make sure each developer is aware of his tasks and schedule. We also suggest displaying a stat of user stories under seven statuses separately during the project, together with project velocity calculated at the end of iteration. This way customers and project managers can easily measure project progress and make adjustments if necessary.

5.7.2 Functional requirement

REQ6-1: The system shall display iteration information for the entire development team

REQ6-2: Only customers and project managers are allowed to edit iteration properties

REQ6-3: The system shall not allow users to pick a user story from a later release version when there are ones remaining in a earlier version

REQ6-4: The system shall display the stat of user stories under each status for each iteration

REQ6-5: The system shall display project velocity at the end of each iteration

5.8 Project Event Notification

5.8.1 Description

Event notification functionality is crucial to fill the communication gap between distributed teams. It helps team members be aware of what is happening in the project without having to ask. This avoids the necessity of certain communication which is hard to initiate in a GSD environment. Customers and project managers can monitor the project in a real-time fashion. It is important for the system to notify appropriate project stakeholders when significant events happen.

5.8.2 Functional Requirements

REQ7-1: The system shall notify customers of the following Planning and Release phase events:

a. An user story is created/updated/removed
b. A Project Release Plan is created/updated/removed
c. A Site Release Plan is created/updated/removed
d. An iteration is created/updated/removed
e. An iteration is completed
f. A Release is ready/released
g. A developer is added/removed from site

REQ7-2: The system shall notify project managers of all Planning, Implementation, Test and Release phase events

a. A user story is assigned to site
b. A user story assigned to site is updated/removed
c. A Project Release Plan is created/updated/removed

d. A Site Release Plan is created/updated/removed

e. An iteration is created/updated/Removed

f. An iteration is complete

g. A iteration is accepted by the customer

h. A Site Release is ready/released

i. A Project Release is ready/released

REQ7-3: The system shall notify developers about the following Planning, Implementation, Release phase events:

a. A user story is assigned

b. A user story assigned is updated/removed

c. A user story acceptance is accepted/failed

d. An iteration is created/updated/removed

e. A Site (to which the developer belongs to) Release Plan is created/updated/removed

f. A Project Release Plan is created/updated/removed

g. A Site Release is ready/released

h. A Project Release is ready/released

i. A developer is added/removed from site

REQ7-4: The system shall notify users when a conversation invitation arrives

REQ7-5: The system shall notify users when an instant message arrives

CHAPTER 6

IMPLEMENTATION AND VALIDATION

In this chapter we present prototype software, GSDXP, which implements the methodology we introduced in previous chapter. This prototype is developed as an Eclipse plug-in. This prototype does not fully implement all the requirements of our approach, only the ones considered necessary to demonstrate proof of feasibility.

6.1 Prototype Scenario

Our prototype is developed based on the scenario shown in Figure 6.1.

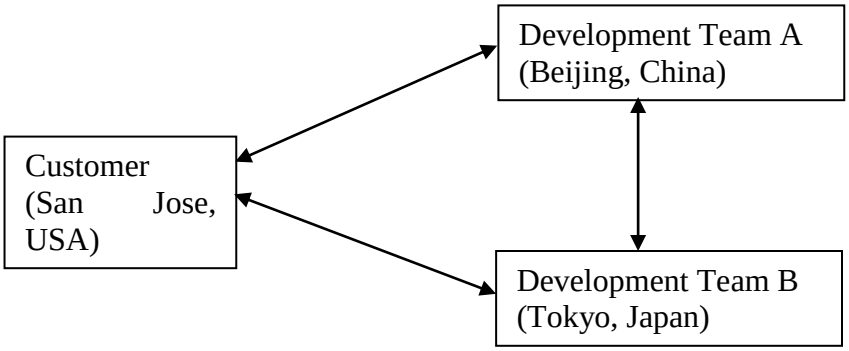

Figure 6.1: Prototype scenario

In this scenario, there are three sites located in three different countries. The customer's company is in San Jose, CA, USA. Two development sites are located respectively, at Beijing, China and Tokyo Japan. Each development site has a project manager who is in charge of coordinating and managing the development site.

6.2 Project Information Overview

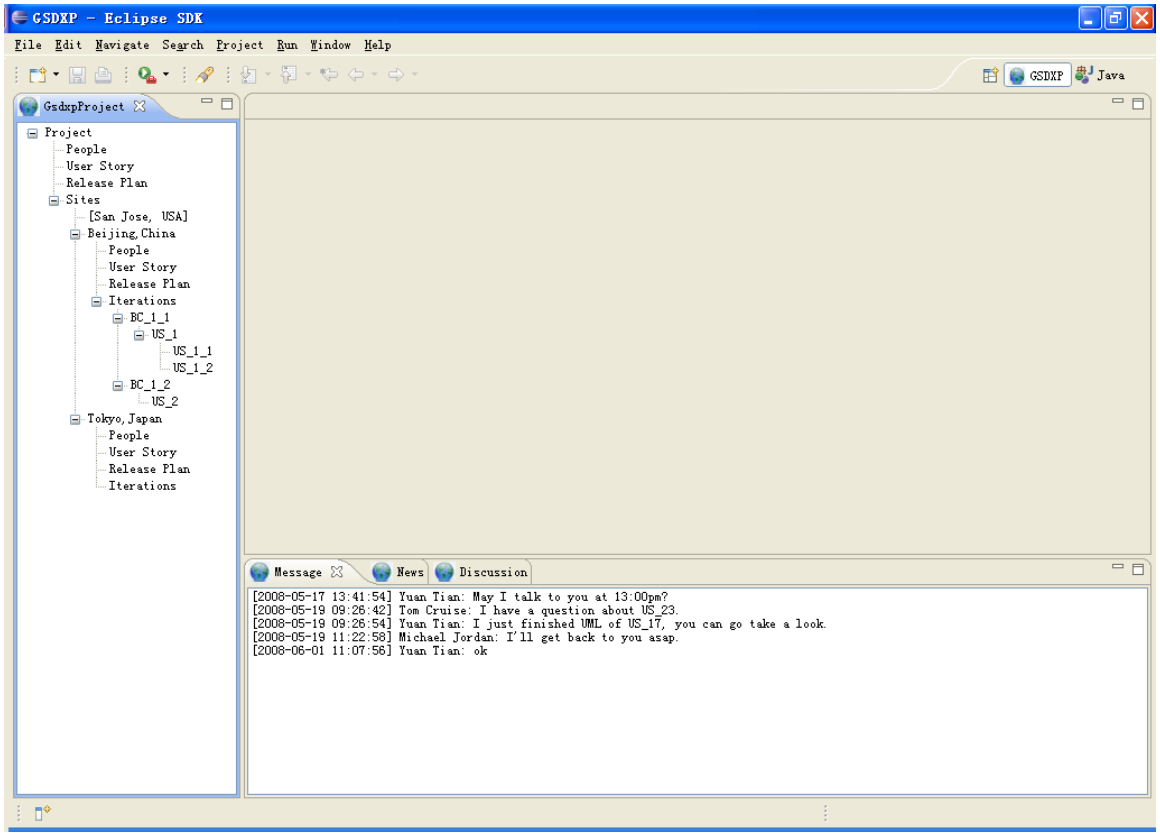

Figure 6.2: Project Information Overview

Figure 6.2 shows the project information overview of the plug-in. In the left view, project information and structure are displayed in a tree fashion. The "People" node contains information on the entire team member (customer, project manager, developer). The "User Story" node and "Release Plan" node contain all the user story information and the project release plan. The three sites information are saved under the "Site" node. The customer site at San Jose is listed at the top. The following two development sites are located at Beijing and Tokyo separately. Each site has its own team members, user stories, release plans, and iteration lists. At the right bottom of window, there are three tabbed views: Message, News and Discussion. The Message view lists messages received

whether user is online or not. The News view lists event notifications. The Discussion view is an embedded instant messenger which supports real-time conversation. By double clicking on the “Project” root node, a window like Figure 6.3 presents project outlines.

Figure 6.3: Project outline

Double clicking on a site name results in a window like Figure 6.4: Site outline. This window presents an outline of the site.

Figure 6.4: Site outline

Using this feature, every team member can easily obtain a big picture of the project and locate the project resources he needs no matter where he is. This is especially helpful if new team members are introduced in the middle of project.

6.3 Team Member Management

In the previous chapter, we proposed that a team member management approach to alleviate communication difficulties. In GSDXP, a team member can check out the full member list from “People” node under “Project” tree, as illustrated in Figure 6.5.

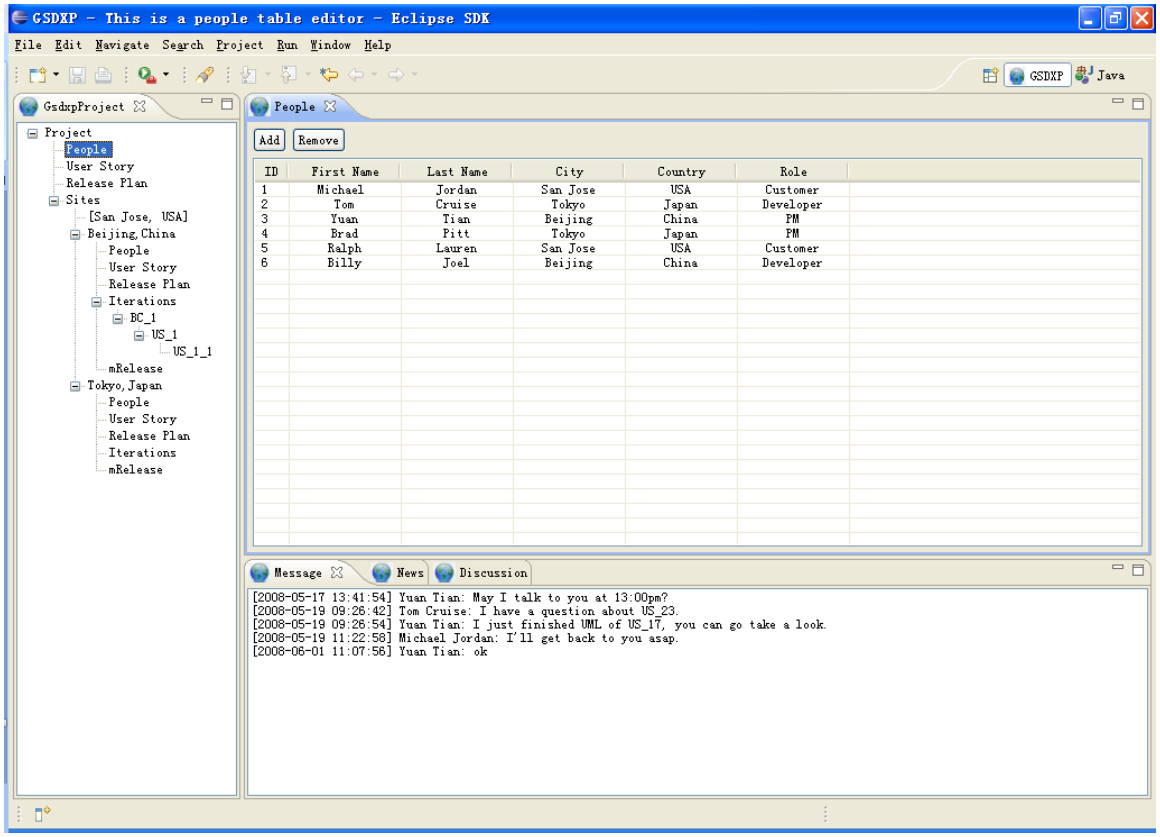

Figure 6.5: Full team member list

From this list, users can easily locate any team member. This list also can be grouped by team member property to facilitate easy reference to information. When each member is double clicked, a pop-up window displays the member's detail information (see Figure 6.6). Adding or Removing a team member can be done by clicking "Add" and "Remove" button at the upper-right corner.

Figure 6.6: Team member detail information

Besides basic information, the "User Story List" lists the user stories assigned to this developer. This list is very useful to answer "who is working on what" question.

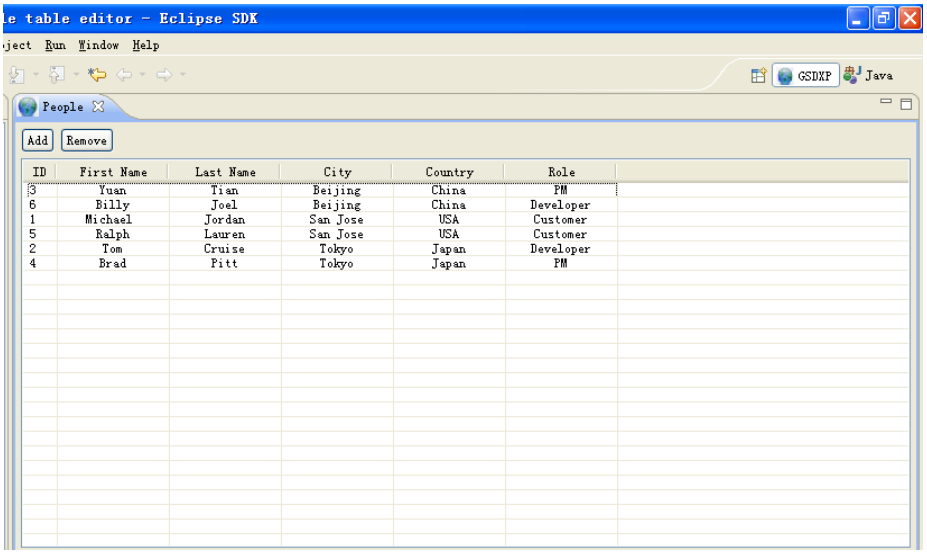

Figure 6.7: Team member list group by "City"

The “People” node under each development site displays a team member list of the site, as illustrated in Figure 6.8.

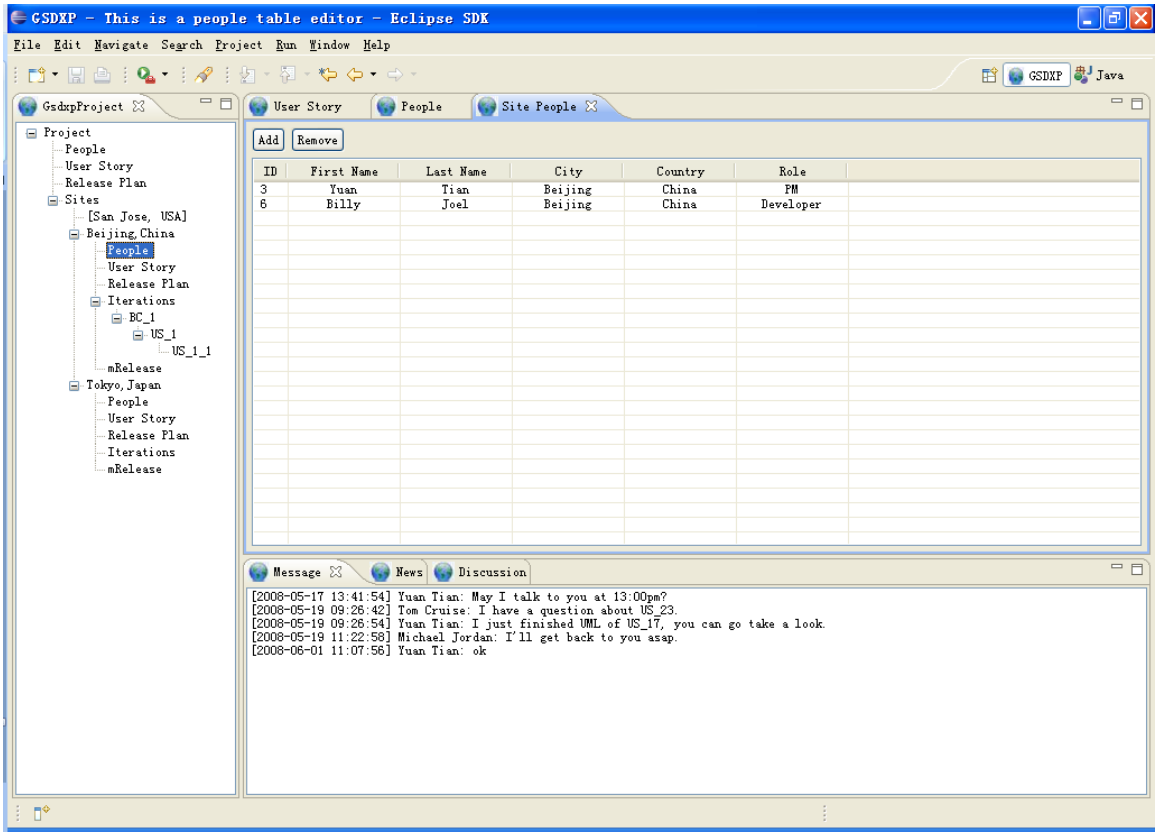

Figure 6.8: Team member list for site “Beijing, China”

6.4 User Story Management

Since most of communication is highly concentrated in XP through user stories, more detail and systematic information is helpful in solving communication problems. We propose in Chapter 5 using the “Project User Story Inventory” and the “Site User Story Inventory” for better management and communication. This approach is implemented through the “User Story” node under the “Project” tree root and the “Site” node separately. Each “User Story” node under the tree root saves all the user stories created for this project.

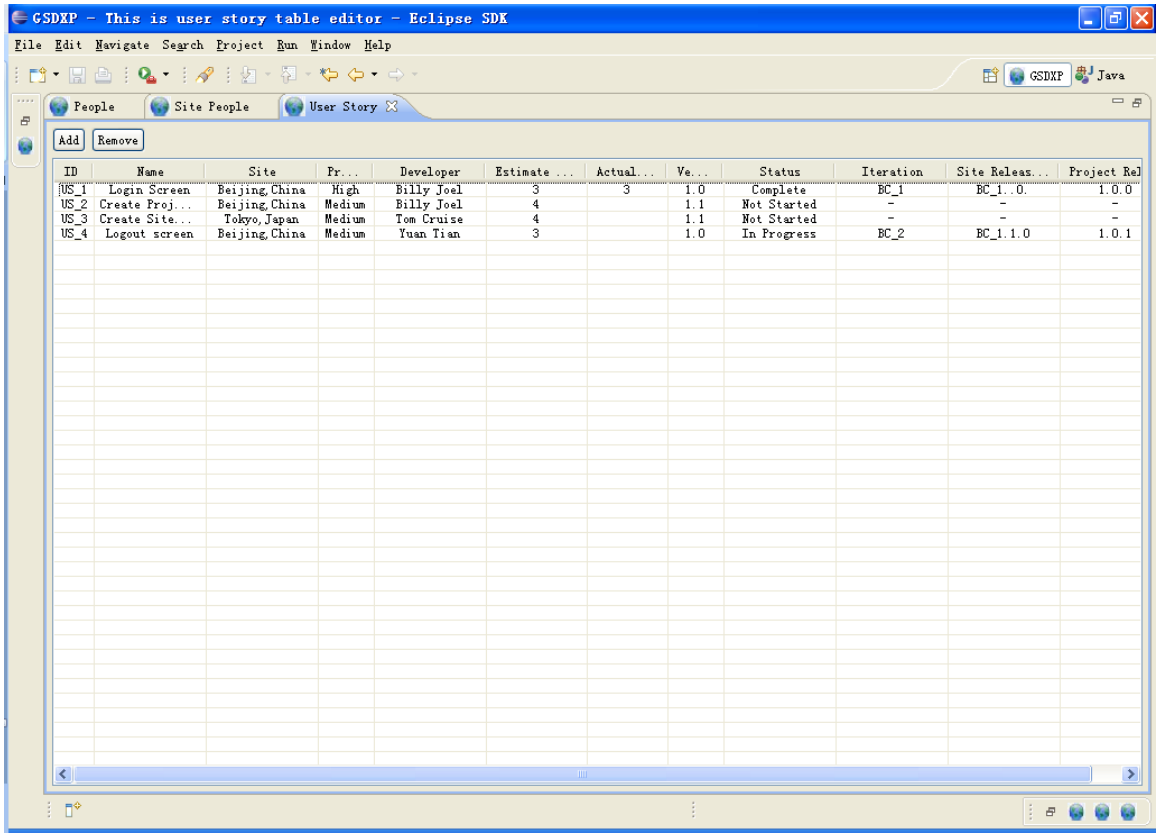

Figure 6.9: Project User Story Inventory

Figure 6.9 shows the “Project User Story Inventory”. From this inventory, we can see there are four user stories. Each user story’s progress can be observed from this list. For example, user story “US_1”, “Login screen”, is set to have High priority with a status of “Complete”. It was estimated as requiring 3 programming days and actually took 3. It is included in the iteration “BC_1”, which is belong to site release “BC_1.0.0” and project release “1.0.0”.

Double clicking on “US_1” will pop a “User Story Detail Information” window shown in Figure 6.10.

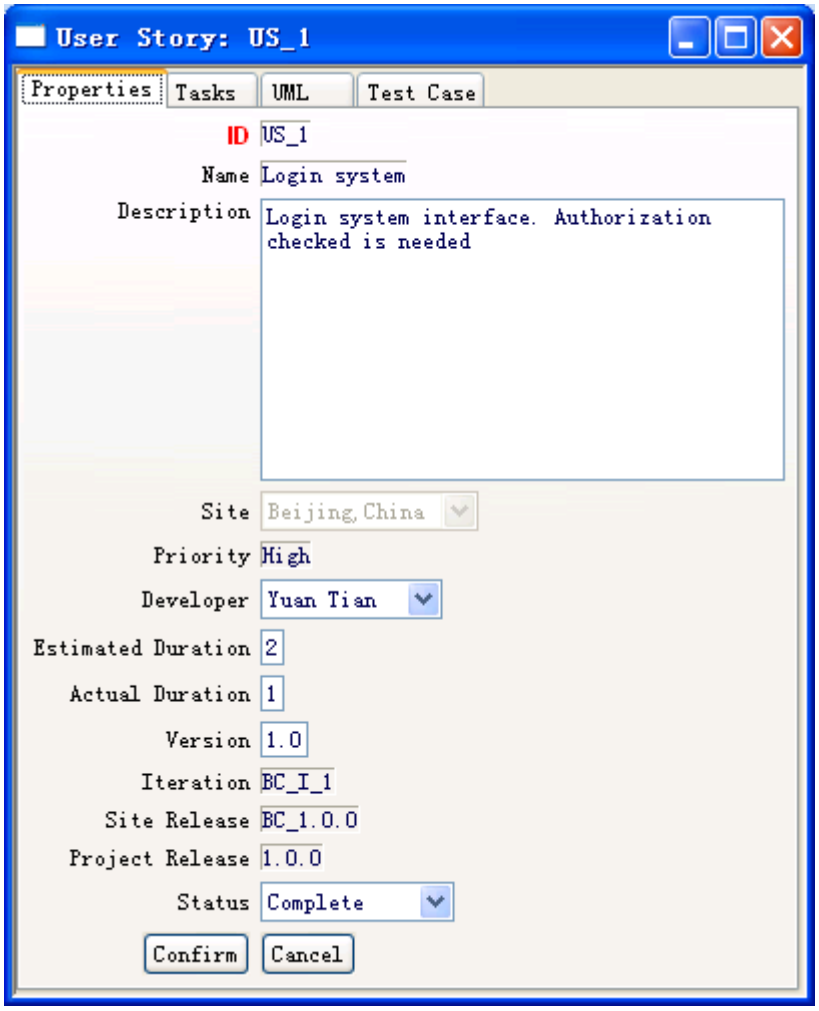

Figure 6.10: User Story Detail Information Window

The “Properties” tab displays the general information of user story.

The “Tasks” tab shows the tasks list if the user story has been broken down to tasks. Figure 6.11 shows that US_1 was divided into two tasks.

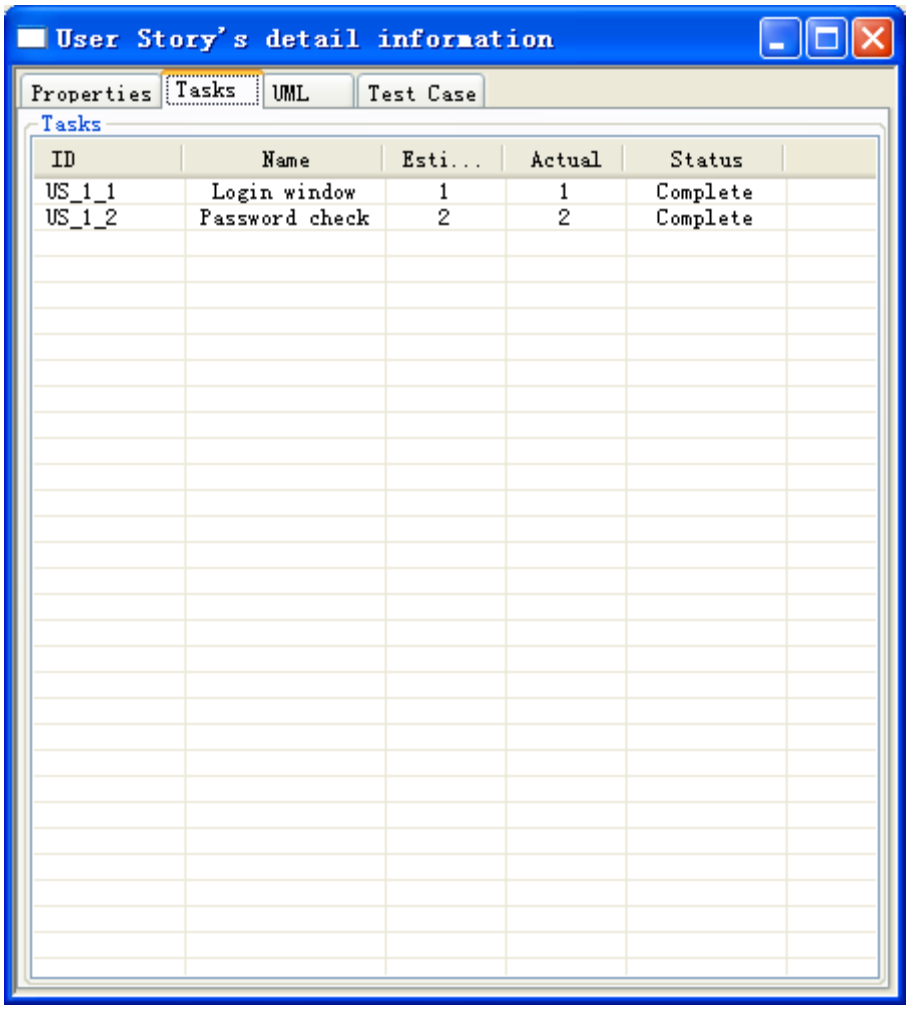

Figure 6.11: User Story Detail Information Window

The "UML" tab displays the UML diagram path for this user story as illustrated in Figure 6.12. Add, Remove and View functionalities can be performed by clicking specific buttons on the right.

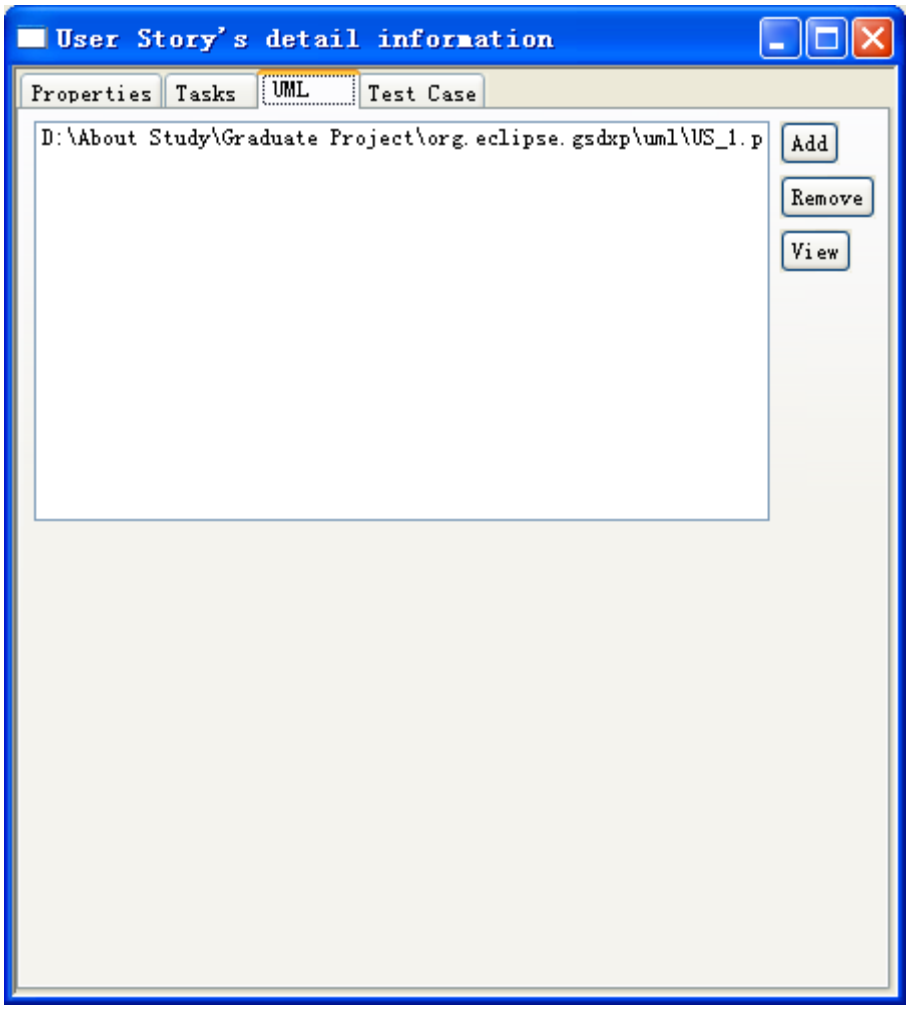

Figure 6.12: UML tab shows the UML diagram file

The "Test Case" tab displays the unit test case file for this user story. User can Add, Remove or View the file by clicking specific buttons on the right.

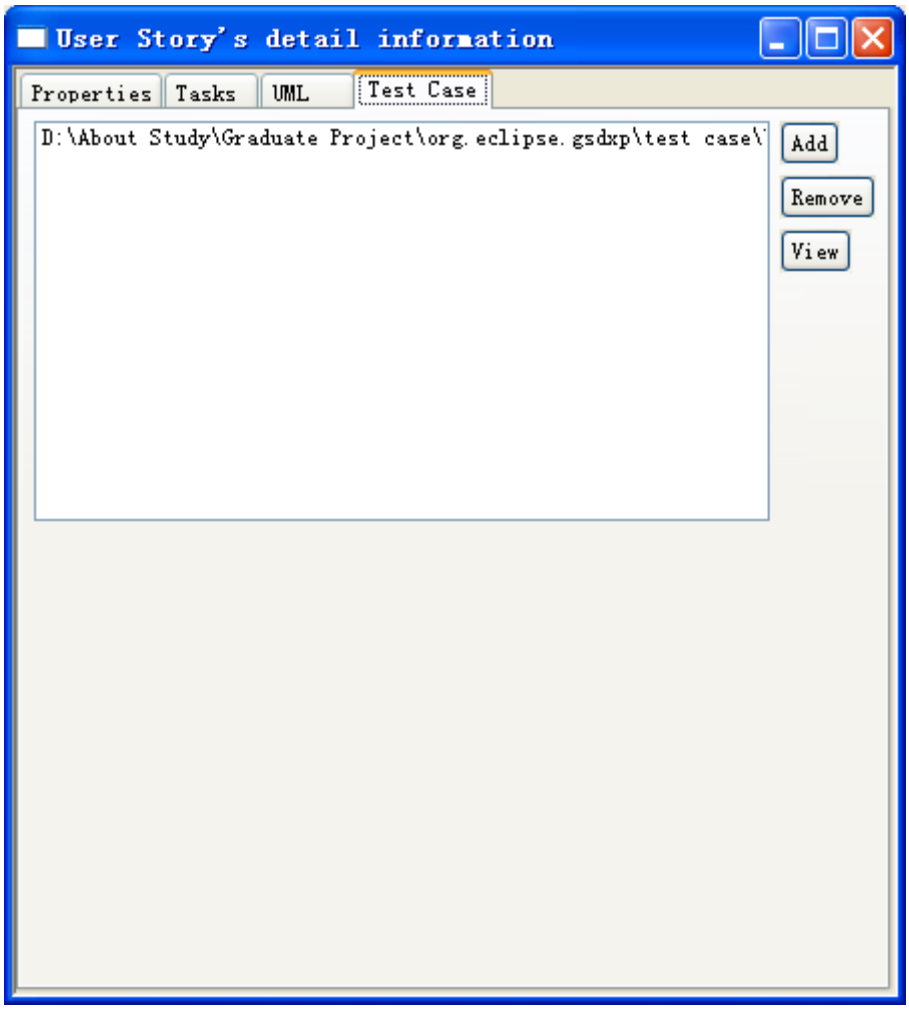

Figure 6.13: Test Case tab shows the unit test case file

The "Project User Story Inventory" provides a way to quickly acertain a user story status and its related work for globally distributed team members. It facilitates the communication from many aspects.

- **Difficult to initiate communication –** a team member can find out "who to contact what" from here
- **Miscommunication –** User story properties give users a direct way to understand content and progress. Standard UML diagram and test case decreases the possibility of misunderstanding.

- **Increased communication cost- time, money, and staff –** The User Story Inventory decreases the necessity of initiating communication because a team member can obtain all the user story information from the inventory without asking. Communication is only needed when user has a question about a certain property.
- **Time difference –** The Inventory is saved on the server, which is accessible any time.

Above all, The "User Story Inventory" can facilitate communication during project. When customers create a lot user stories, it may be inconvenient to locate a specific user story from the "Project User Story Inventory". A "Site User Story Inventory" contains the user stories which have been assigned to the site, thereby giving project managers a big-picture perspective of their site. Figure 6.14 shows the "Site User Story Inventory" for Beijing development site.

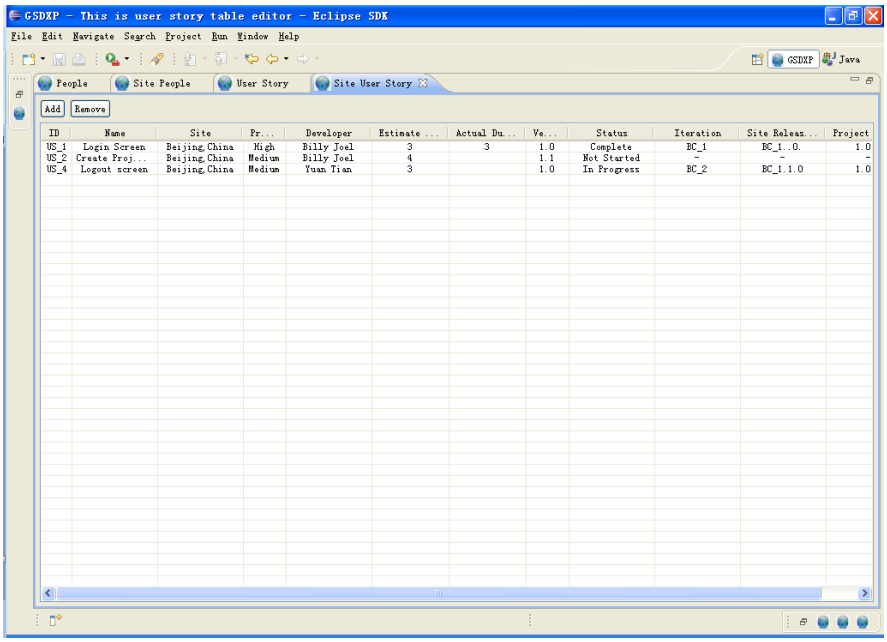

Figure 6.14: "Site User Story Inventory" for Beijing development site

The User Stories can be broken down into tasks. Figure 6.15 shows a task added to User Story US_3.

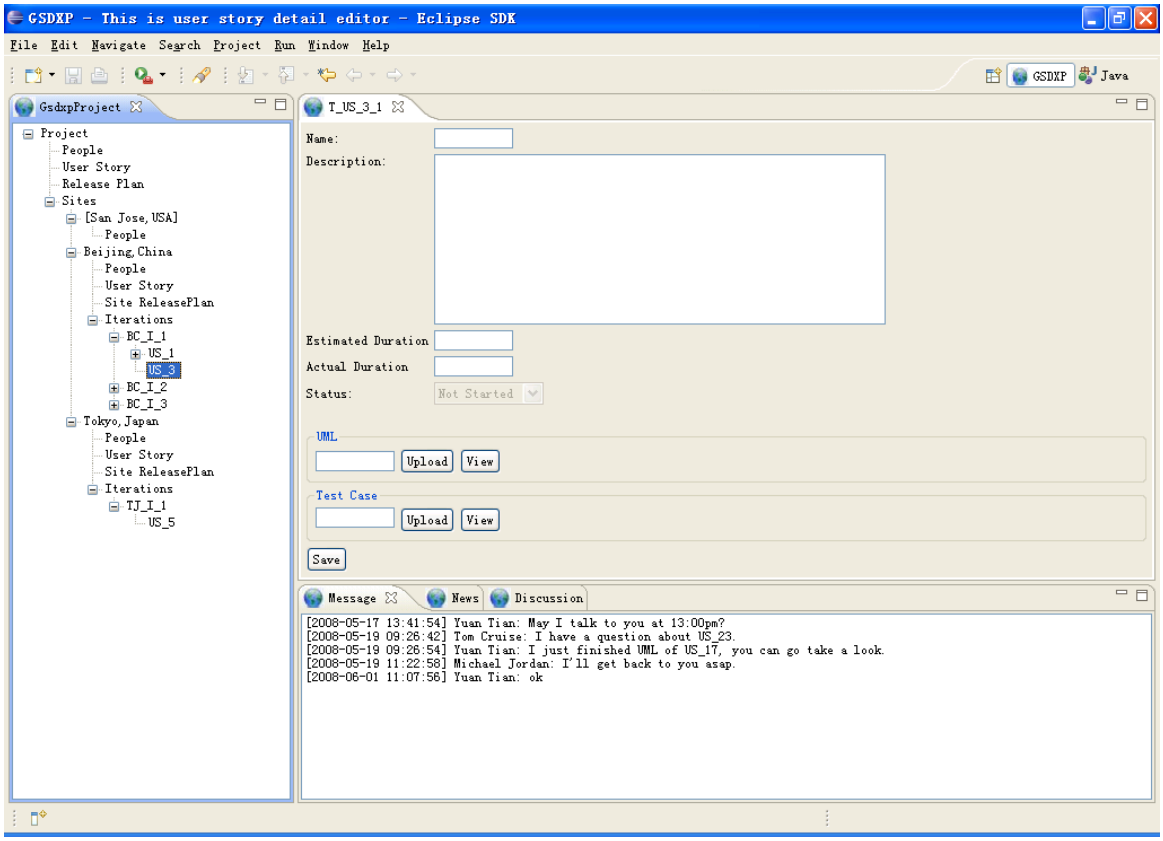

Figure 6.16: Create a Task for US_3

6.5 Project Release Management

In Chapter 5.6 we describe our approach of dividing the release plan into the Project Release Plan and the Site Release Plan. In our prototype plug-in, they are represented by the "Release Plan" under Project tree root and the "Release Plan" under the site branch separately. The Site Release Plan for Beijing is shown in Figure 6.17.

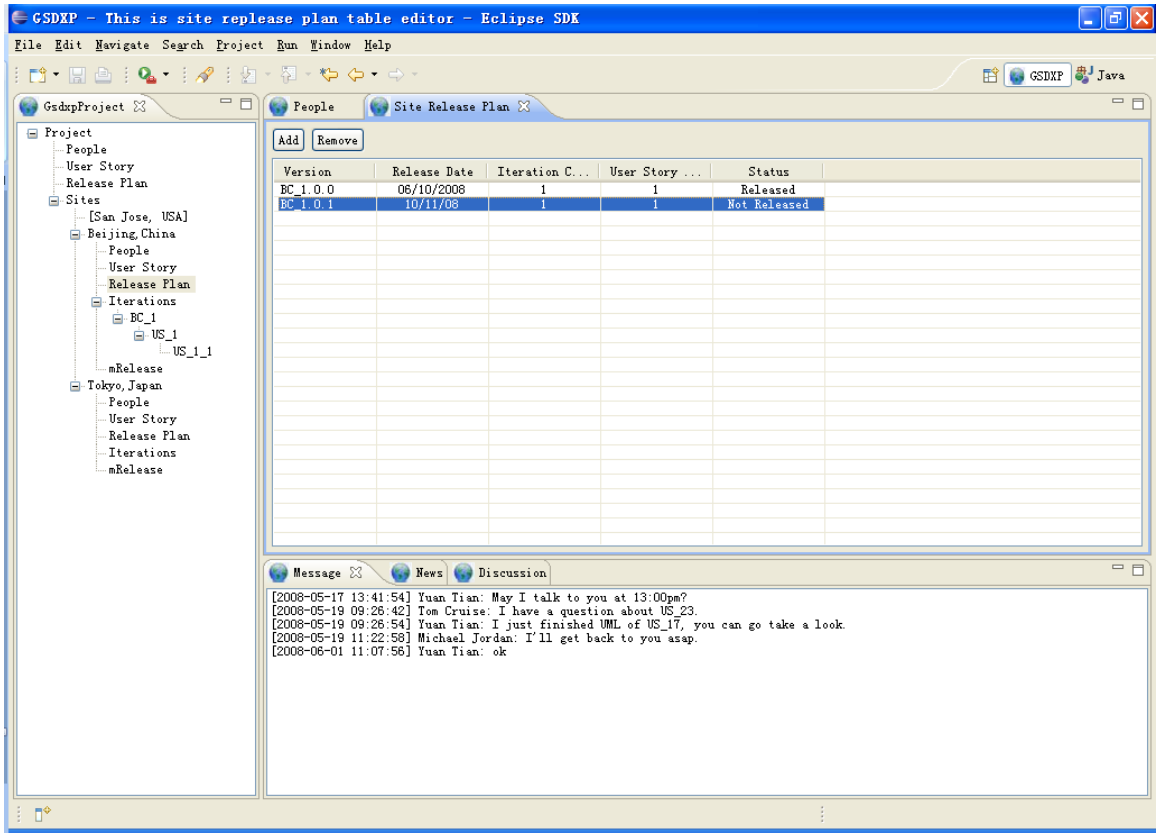

Figure 6.18: "Site Release Plan" for Beijing development site

There are two site releases created, BC_1.0.0 and BC_1.0.1. Each release contains one user story that is developed in one iteration. BC_1.0.0 is already released. BC_1.0.1 is still in progress. To create a new Site Release Plan, we can click on "Add" button. A pop-up window is looked as Figure 6.19.

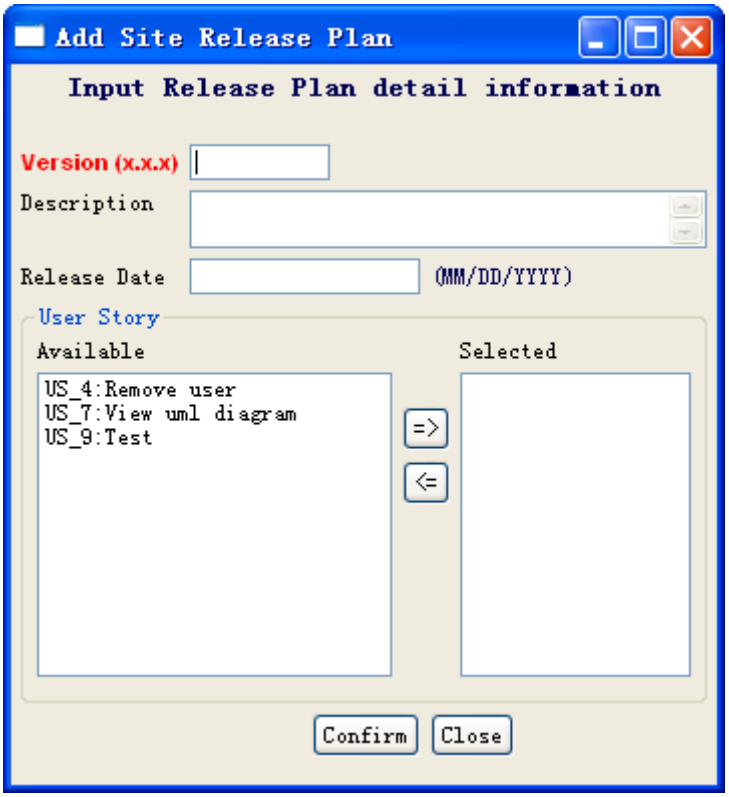

Figure 6.20: Create new Site Release Plan

In the “User Story” area, the “Available” box contains all the user stories assigned to the Beijing site but which haven’t yet been assigned to any Site Release. User Stories assigned to a release plan will not be shown here. In this way, we can avoid assigning a user story to multiple sites. To view/edit a release plan, double click on the Site Release Plan.

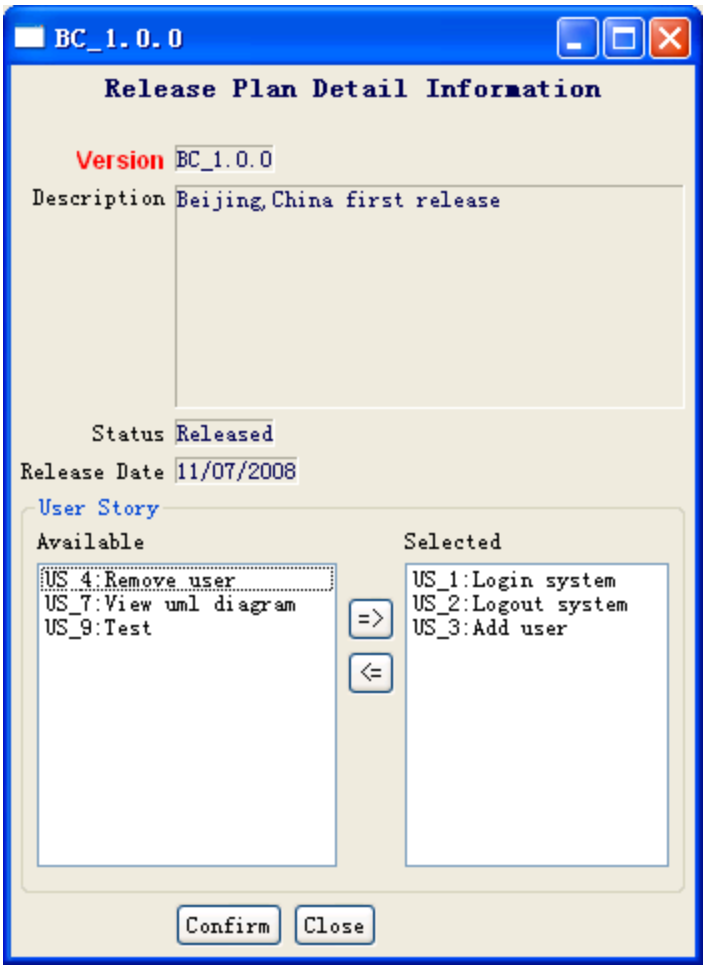

Figure 6.21: View/Edit Site Release Plan

Moving a user story from the “Available” box to the “Selected” box will add a user story into this site release; moving stories from “Selected” to “Available” remove them from the site release.

By double clicking on “Release Plan” under the project tree, the Project Release Plan of GSDXP is shown, as in Figure 6.22.

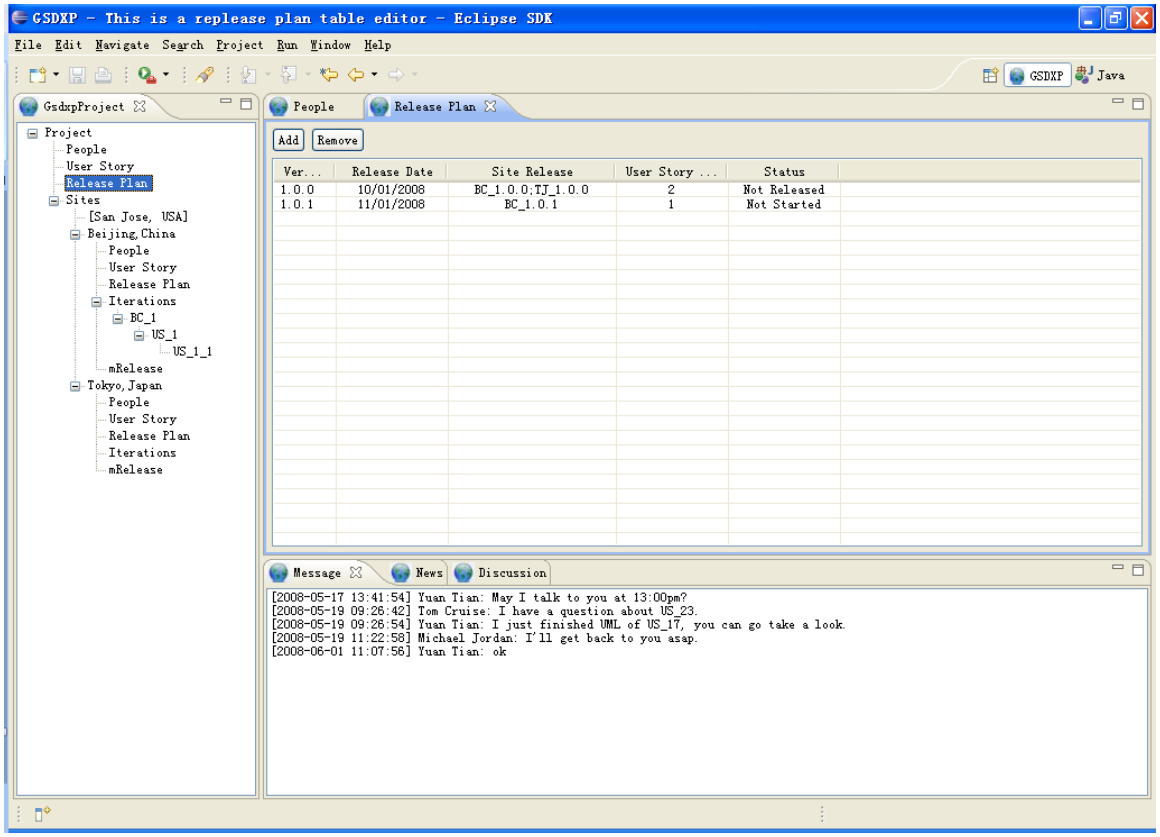

Figure 6.23: Project Release Plan of GSDXP

From the release plan list, we can see there are two Project Release Plans. Version 1.0.0 contains two Site Release Plans: BC_1.0.0 from site Beijing and TJ_1.0.0 from site Tokyo. Each of the site release plans has one user story, the total user stories that will be delivered is two. The scheduled release date is 10/01/2008, which is "Not Released" yet. By double clicking on each release plan, a Release Plan Detail Information window appears, as in Figure 6.20.

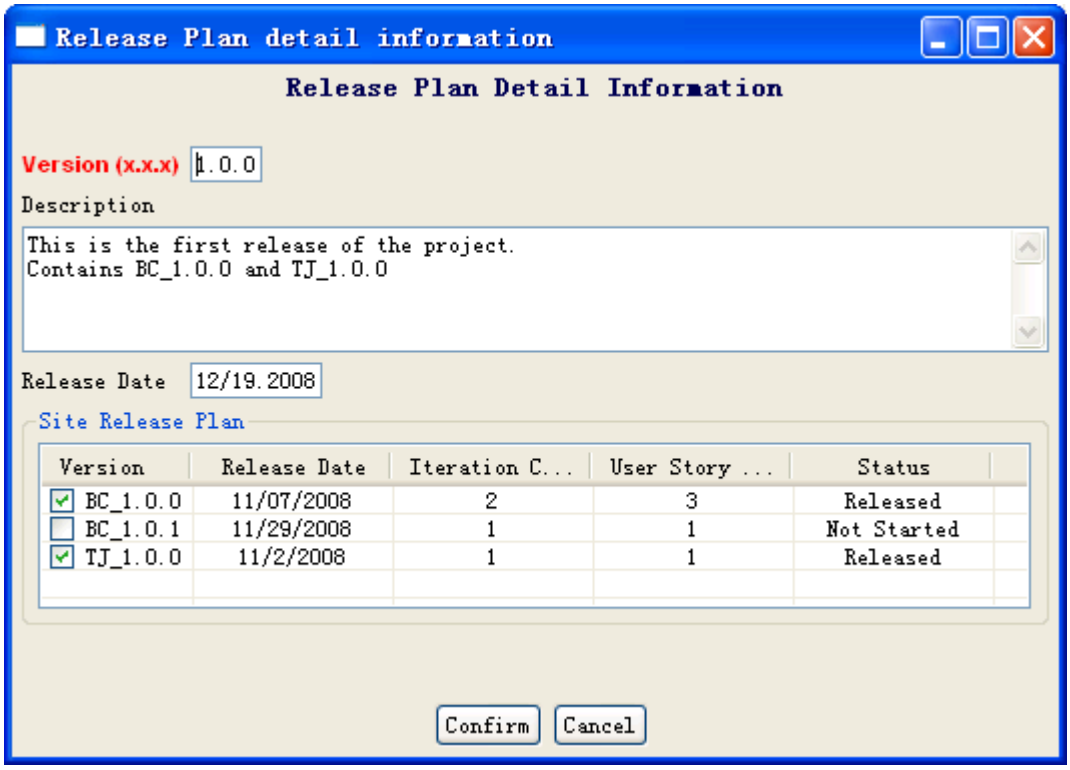

Figure 6.24: Project Release Plan Detail View

The "Site Release Plan" lists all the Site Release Plans included in this Project Release Plan –BC_1.0.0, TJ_1.0.0 and unassigned Site Release Plan TJ_1.0.1. From this window, the customer can add or remove a Site Release Plan.

6.6 Project Iteration Management

On the project tree, the "Iteration" node contains the iterations the site is working on.

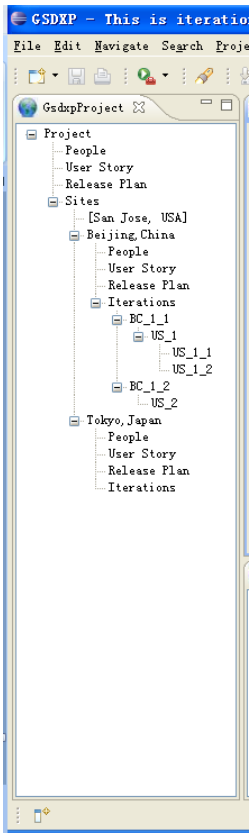

Figure 6.25: Iteration tree

As showed in Figure 6.21, the site Beijing has two iterations so far. In BC_1_1, one user story “US_1” was developed, which was further divided into two tasks “US_1_1” and “US_1_2”. The second iteration “BC_1_2” developed user story “US_2” which was not divided into tasks. By double clicking on the “Iteration” node, a Site Iteration list is shown as Figure 6.26.

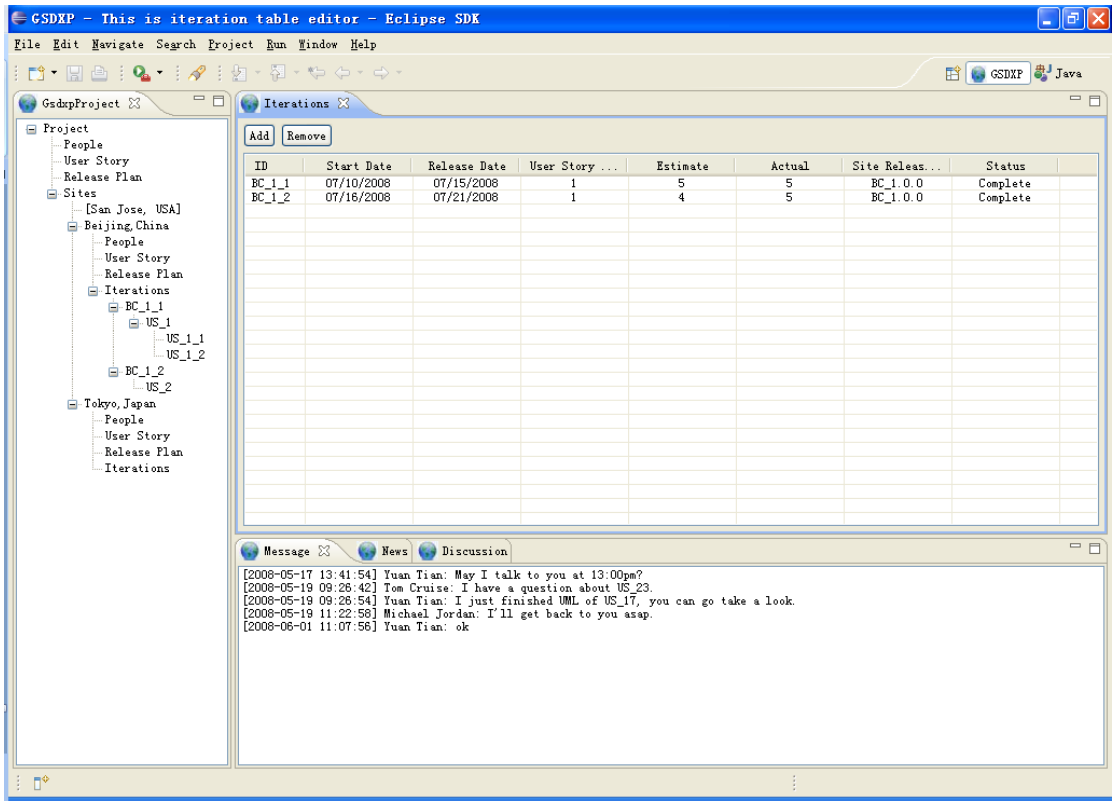

Figure 6.26: Site Iteration List

The iteration detail view is shown in Figure 6.27.

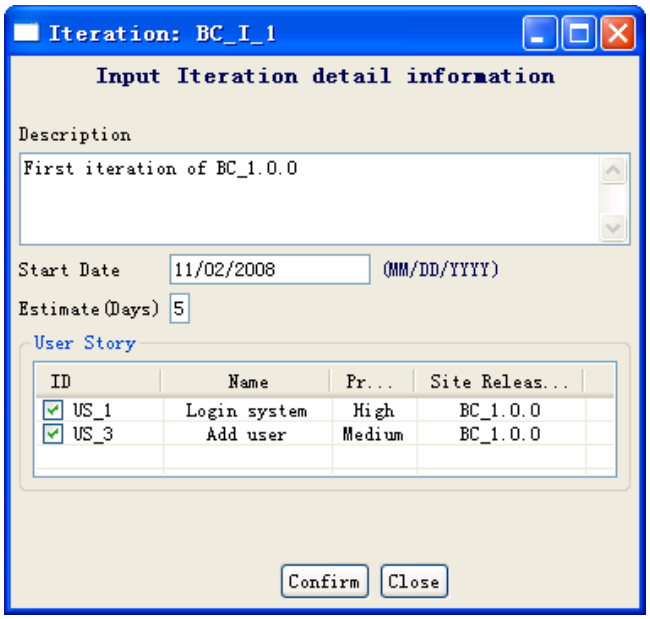

Figure 6.27: Iteration detail view

The checked user story in User Story list is the story currently included in iteration. Because iteration BC_1_1 is already complete, no other user stories are listed.

6.7 Project Event Notification

Event notification can help improve communication by making team members aware of significant occurrences that have taken place.

6.7.1 Message View

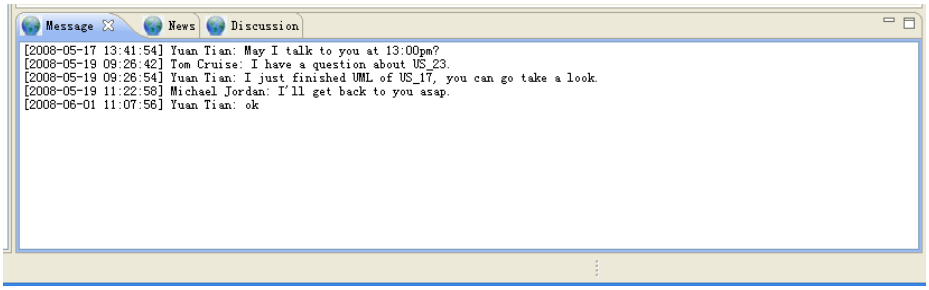

Figure 6.28: Message view

The message view is used to display messages in the order in which they were received. This functionality does not require the user to be online. Messages are saved in database. When the user logs in, he can browse all the messages he received. Users can reply to messages by double clicking on the message, the result being a dialog box illustrated in Figure 6.29.

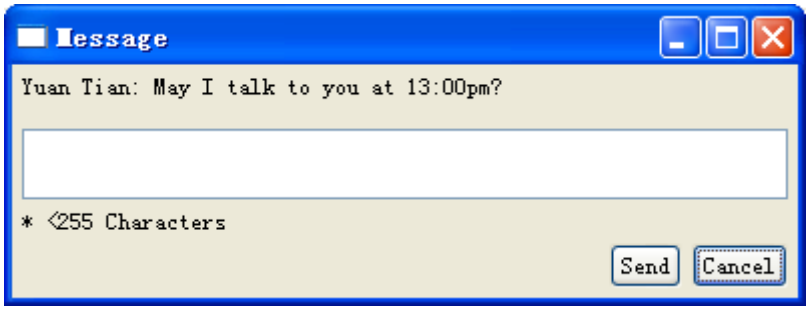

Figure 6.29: Reply message

6.7.2 News View

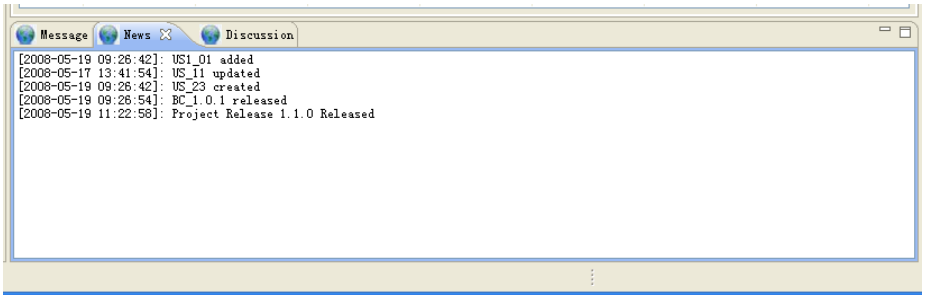

Figure 6.30: News view

The News view displays the news regarding events that have happened in project. This functionality is designed to help improve the "Awareness" problem. From this window, team members can maintain a vision of the project. The "News" tab blinks to prompt user when an event message has arrived.

6.7.3 Discussion View

The Discussion view is used to display the real-time conversation between communication peers. Due to time constraints, this functionality was not implemented. The idea of Discussion is to allow user to initiate communication from the GSDXP plug-in without relying on third-party software. Users can choose to save the content of conversation as a record. Instant messenger functionality is very handy for team members to communicate with other members at other sites without regard to time and spatial limits.

CHAPTER 7

METHODOLOGY VALIDATION

To validate approach, GSDXP was sent to managers and software engineers involved in Global Software Development for usability evaluation. We invited two department managers and six software engineers to be test users. They came from five different companies located at Beijing, China and Tokyo, Japan. All of them had been involved in software outsourcing industry for more than four years. The longest was more than ten years. Among eight trial users, three had practiced XP on real projects, and five knew the basic idea of XP. The software development process they were using at the time of the evaluation was a combination of waterfall and iterative.

Before the trial run, the virtual users were asked to name the most common problem they have in software outsourcing. Six users answered communication problems and the other two gave understanding user needs as the largest problem, which is partly caused by communication. All of them mentioned that project progress is significantly slowed down without a on-site customer. A project manager said "Sometimes it takes more than two days to confirm a question which should only take twenty minutes if there were an on-site customer. We have to rely on back-and-forth emails, conference calls. That slows down project progress."

After the trial run, a survey was conducted regarding the usability of plug-in. The survey result is showed in Figure 7.1.

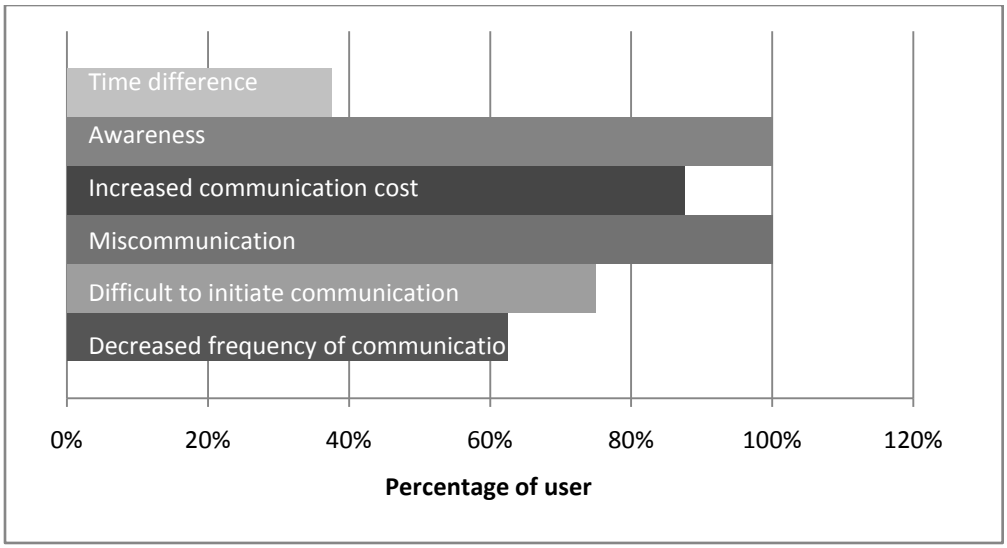

Figure 7.1: User's feedback on plug-in usability

All eight users agreed that our approach helps solve "Awareness" and "Miscommunication" problems. Seven out of eight agreed it decreases communication cost, time, money and staff. Six think it facilitates communication initiation. Three thought it helps to solve time difference problem, although not all the respondents were able to reply definitive to this issue.

All of users thought this plug-in is a light-weight and handy tool for project management. The user interface is straightforward to use, although general XP knowledge is required. Five of them thought the approach is generally practical in GSD projects and helps solve communication problems. Three of the eight virtual users had concerns regarding project size, server reliability, and network quality etc. Two managers were most interested in the User Story Management functionality, while four of engineers were pleased to see the Event Notification functionality. In the end, three of users suggested the further improvement of Release Plan Management.

CHAPTER 8

SUMMARY AND FUTURE WORK

8.1 Summary

XP cannot be fully implemented in a global distributed development setting. We have proposed, partially implemented, and evaluated an XP-based methodology to solve the communication problems in GSD projects from many perspectives. It also provides an easy and systematic way to address the weak support of project management. Based on our survey results, we find our approach is generally welcomed by GSD practitioners. Two of tool evaluators who barely knew XP before showed interest in practicing it in future projects. Our study shows that after carefully tailoring, together with its dynamic characteristic and complimented by a project management methodology, XP is a suitable practice for Global Software Development.

8.2 Conclusion

Based on our study, it shows that according to appropriate adaption, XP and GSD can be combined together to reap the benefits of both even though former emphasizes on communication, and latter is inherited with communication gap. We identified that the XP practices of on-site customers, planning game, small release, simple design, testing and collective ownership as the ones can be customized to alleviate communication problems in GSD. Since XP does not provide support for project management, management approaches are needed when practicing it. We found that management of

key elements in the project facilitates the deployment of XP. Those key elements include project information, project site information, project team member information, user story information, project release plan information, project iteration information, and project events information. The goal of managing this information is to provide every stakeholder a clear vision of the project, in such a way to decrease the communication necessity and communication difficulty.

8.3 Future Work

As with any significant endeavor, work still needs to be done. One of the problems is that our approach requires that users have a relatively high understanding of their business in order to make the right decision -- such as user stories assignment and priorities setting -- at critical points so site releases can finish at right time to be combined to project release. Server stability is another problem we need to consider. When this approach is applied to a big project, server performance under heavy loads is crucial to ensure project progress. A mechanism of processing concurrent access is definitely necessary because project information is shared with all stakeholders. For the reason of time, we only developed an outline of prototype software. There are few other key functionalities we have not implemented, such as user privilege management, embedded instant messenger, etc. As suggested by our trial users, the release plan management part may be the part needs further study because in our study, release planning mainly relies on customer understands the business and big picture of project. How to handle release conflicts and delay are left for the following study.

BIBLIOGRAPHY

[1] Iyengar,P.Application Development Is More Global than Ever, Publication G00124025, Gartner, 2004; www.gartnre.com/resources/124000/124025/application_dev.pdf

[2] "Offshore's New Horizons," Global Technology Business, v.3, n.3 Mar 2000, pp.12-15

[3] Lindstrom,L and Jeffries, R.Extreme Programming And Agile Software Development Methodologies, *Information Systems Management*, 24:3 , pp. 41-60, 2004

[4] Manifesto for Agile Software Development, http://agilemanifesto.org/

[5] Herbsleb, J. Global Software Engineering: The Future of Socio-technical Coordination. *Future of Software Engineering 2007* Briand, L. and Wolf, A. eds. IEEECS Press, 2007.

[6] Allen, T.J., Managing the Flow of Technology, 1977, Cambridge, MA: MIT Press

[7] Mockus, A, and Herbsleb, J, Challenges of Global Software Development, *Proceedings of the Seventh International Soft-ware Metrics Symposium*, METRICS 2001. IEEE. pp. 182-184.

[8] ErranCarmel ,RituAgarwal, Tactical Approaches for Alleviating Distance in Global Software Development, *IEEE Software*, v.18 n.2, p.22-29, Mar 2001

[9] Damian, D., Stakeholders in Global Requirements Engineering: Lessons Learned From Practice. *IEEE Software*, v.24 n.2, p.21-27, Mar./Apr. 2007

[10] R.D. Battin, R. Crocker, J. Kreidler, and K. Subramanian, Leveraging Resources in Global Software Development, *IEEE Software*, vol. 18, no. 2, pp. 70-77, Mar./Apr. 2001

[11] C. Ebert, and P. De Neve, "Surviving Global Software De-velopment", *IEEE Software*, pp. 62-69, Mar./Apr. 2001

[12] E.A. Karlsson et al., "Daily Build and Feature Development in Large Distributed Projects, *Proc. Int"l Conf. Software Eng*., *IEEE CS Press*, Los Alamitos, Calif., 2000, pp. 649-658.

[13] Simons, M. Internationally Agile. InformIT, Mar 15th, 2002

[14] Fowler, M. Using Agile Software Process with Offshore Development. http://www.it.uu.se/edu/course/homepage/acsd/ht03/Fowler.pdf, Sept. 2003

[15] Xiaohu, Y., Bin, X., Zhijun, H. Maddineni, S. Extreme Programming in Global Software Development. *Proceedings of the Canadian Conference on Electrical and Computer Engineering*, pp.1845-1848. Vol. 4, 2-5 May 2004

[16] Nisar, M. and Hameed, T. Agile Methods Handling Offshore Software Development Issues. *Proceedings of INMIC 2004*, *8th International Multitopic Conference*, pp. 417-422, Dec.2004

[17] Karlsson, E., Andersson, L. and Leion, P. Daily Build and Feature Development in Large Distributed Projects. *Proceedings of the International Conference on Software Engineering*, pp.649-658, 2000

[18] Farmer, M. DecisionSpace Infrastructure: Agile Development in a Large, Distributed Team. *Proceedings of the Agile Development Conference*, 2004

[19] Sinha, V.; Sengupta, B.; Chandra, S. Enabling Collaboration in Distributed Requirements Management, *IEEE Software*, V. 23, No.5, pp.52-61, Sept./Oct. 2006

[20] Jazz Community, https://jazz.net/index.jsp

[21] Highsmith, J. ; Cockburn, A.; Agile software development: the business of innovation, *Computer*, V.34, Issue 9, pp.120 – 127, Sept. 2001

[22] Jeffries, R. et all., Extreme Programming Installed, Addison Wesley Longman, 2001, 172.

[23] Storm User Story Tool, http://xpstorm.sourceforge.net/.

[24] Rees, M.J.; A feasible user story tool for agile software development, Software Engineering Conference, 2002. Ninth Asia-Pacific, pp.22 – 30, 4-6 Dec. 2002

[25] Kahkonen, T.; Abrahamsson, P., Digging into the fundamentals of extreme programming building the theoretical base for agile methods, *Proceedings of 29th Euromicro Conference*, pp.273 – 280, 1-6 Sept. 2003

[26] Tuomi, I., Corporate knowledge: Theory and Practice of Intelligent organizations. 1999, Helsinki: Metaxis.

[27] DSDMConsortium, Dynamic Systems Development Method, version 3. Ashford, Eng.: DSDM Consortium, 1997.

[28] J. Stapleton, Dynamic systems development method -The method in practice: Addison Wesley, 1997

[29] S. R. Palmer and J. M. Felsing, A Practical Guide to Feature-Driven Development, 2002.

[30] Richard Baskerville , Jan Pries-Heje, Racing the E-Bomb: How the Internet Is Redefining Information Systems Development Methodology, *Proceedings of the IFIP TC8/WG8.2 Working Conference on Realigning Research and Practice in Information Systems Development: The Social and Organizational Perspective*, p.49-68, July 27-29, 2001

[31] K. Beck, Embracing Change With Extreme Programming, *IEEE Computer*, vol. 32, pp. 70-77, 1999.

[32] K. Schwaber, Scrum Development Process, In *OOPSLA'95 Workshop on Business Object Design and Implementation*, 1995.

[33] A. Cockburn, Writing Effective Use Cases, The Crystal Collection for Software Professionals: Addison-Wesley Professional, 2000.

[34] A. Hunt, Thomas, D., *The Pragmatic Programmer*: Addison Wesley, 2000.

[35] Abrahamsson, P.; Warsta, J.; Siponen, M.T.; Ronkainen, J., New directions on agile methods: a comparative analysis, In *Proceedings of 25th Internationa Conference on Software Engineering*, pp.244 - 254 , 3-10 May 2003

[36] Elssamadisy, A.; Schalliol, G.; Recognizing and Responding To "Bad Smells" In Extreme Programming, *Proceedings of the 24rd International Conference on Software Engineering, 2002. ICSE 2002*. pp.617 – 622 2002

[37] Abrahamsson, P.; Koskela, J.; Extreme programming: a survey of empirical data from a controlled case study, *Proceedings of International Symposium on Empirical Software Engineering*, 2004. ISESE '04, pp73 – 82, 19-20 Aug 2004

[38] Paul S Grisham,; Dewayne E. Perry,; Customer Relationships and Extreme Programming, *Proceedings of the 2005 workshop on Human and social factors of software engineering*, HSSE '05, May 2005

[39] Boehm, B,; Get ready for agile methods, with care, *IEEE Computer*, V. 35, Issue: 1, pp. 64-49, Jan 2002

[40] Lan Cao,; Kannan Mohan,; Peng Xu,; Balasubramaniam Ramesh,; *How Extreme does Extreme Programming Have to be? Adapting XP Practices to Large-scale Projects, System Sciences, Proceedings of the 37th Annual Hawaii International Conference on 2004*, pp 10, 5-8 Jan. 2004.