EXTREME PROGRAMMING IN GLOBAL SOFTWARE DEVELOPMENT

Yang Xiaohu, Xu Bin, He Zhijun
College of Computer Science & Technology
Zhejiang Univ. 310027 Hangzhou, P. R. China
{yangxh, xb, hezj}@zju.edu.cn
Srinivasa R. Maddineni
SSGM//Securities Trading IT
State Street Corporation, Boston MA 02110, USA
srmaddineni@statestreet.com

Abstract

Reliable communication is essential for the success of global collaborative software development efforts. Software development organizations and methodologies must be tailored to avoid communications issues which may result in misunderstood requirements or missed project milestones.

Extreme Programming (XP) was adopted for the Lattice® Trading System reengineering project - a globally distributed software development effort. Customers in Boston worked with the offshore development in Hangzhou. Business knowledge and requirements were transferred iteratively throughout the project duration.

Steps in the development process were executed at different sites at different times depending on skill set match. An iterative development process reduced communication risk as development for one phase and communication for the next phase could be conducted in parallel. This approach allowed the team to avoid most of the communication delay and improved the quality of the communication as well.

Keywords: *Global collaborative software development; Extreme Programming; software reengineering.*

1. INTRODUCTION

IT development has become globally distributed and time-to-market is critical to the success of businesses introducing products [2, 5, and 7]. Bad communication often results in poorly defined requirements which adversely affect the schedule of global software development projects.

Geographic distribution alone introduces communication risk. Cultural make-up of the teams and significantly different time zones may also become obstacles to the success of distributed teams. Some techniques have been suggested to reduce the cultural and language issues in global projects [3, 4]. The methodology of virtual team management in global development was also introduced [6]. Previous research [2] shows that distributed projects take about two and one-half times longer to complete as similar projects where the project team is centralized. The delays are attributable to communication and coordination issues rather than the size or complexity of the cross-site work [7]. An attempt to minimize risk associated with cross-site communications presented in a study of six software engineering organizations [8]. A study of an automotive engineering group [9] proposed a methodology to reduce dependencies in distributed work projects. For example, one organization may distribute the work over sites [2, 3, and 8] by:

- Developing different subsystems at different sites,
- Executing different process steps at different sites.

To facilitate offshore development, State Street Corporation conducted research on how to achieve good communication for global software development when utilizing Extreme Programming (XP) methodologies [1]. The Lattice® Trading System reengineering project was identified as a candidate for this analysis. Lattice is a legacy equity trading system which was targeted for a reverse-engineering and re-factoring effort. The development team is located in Hangzhou, China while the customer is located in Boston, USA.

2. COMMUNICATING WITH XP

Communication is important both for globally distributed software development and co-located software development. There are several kinds of communication in a software development project:

CCECE 2004- CCGEI 2004, Niagara Falls, May/mai 2004
0-7803-8253-6/04/$17.00 ©2004 IEEE

- Between the development team and the customers,
- Between the developers and the project manager,
- Amongst the customers, and
- Amongst the developers.

XP values communication, simplicity, feedback and courage. XP's recommended practices focuses on improving these different levels of communication practices [Table 1].

Table 1. XP's practices benefit communications

Practice	Benefit
The planning game	Provides systematic communication between the customers, developers and the project manager.
Small releases	Provides a rapid feedback mechanism between the users, the customers and the developers
Metaphor	Provides communication platform for developers, the project manager and the customers.
Simple design	Makes it easy to communicate between the developers.
Testing	Provides rapid feedback between developers and customers.
Re-factoring	Makes it easy to communicate among developers.
Pair programming	Provides a rapid feedback and communication between paired developers.
Continuous integration	Provides developers with rapid feedback on the quality of the code.
On-site customer	Provides timely communication between the customers and the development team.
Coding standards	Makes it easy to communicate between the developers at code level and provides consistency.

Collective ownership requires timely communication between developers when sharing the source code. "40-hour week" suggests developers not to work overtime. Both of them benefit the communication indirectly.

3. ENABLING XP IN GLOBAL SOFTWARE DEVELOPMENT

XP practices achieve good communications in co-located software development projects. However, it is more difficult to implement in global development projects due to delays in customer feedback. The communication between the customers and the developers is expected to be frequent in XP projects. However, when customers and developers are not co-located, the feedback between them is not timely enough, which may become a significant obstacle. In our research, optimal results were obtained when the best practices of XP are adapted and integrated with some existing methodologies resulting in a shortcut communication channel.

3.1 Small releases

When code is delivered in a bulk release, the customer requires large amounts of time to validate all of the new features. Additionally, development may be paused as developers await confirmation from the customer to continue to the next release.

Small releases reduce complexity. As a result, the duration of the validation period is also reduced.

3.2 Small iterations

To reduce the interval of the communication/feedback cycle, the Iteration Planning Game runs weekly. This communication style insures that developers always have enough tasks at hand. This serves to reduce the impact of periodic communications. Additionally, the developers must consider not only their development tasks but also project communications. The project manager should anticipate how communications will impact the schedule in the Planning Game.

3.3 Communicating requirements in pieces

Informal communication is very important in global software development [2]. At the same time, the interval of formal communications should be reduced when communicating globally.

When the customers are not onsite with the development team, story cards may still be used to transfer the requirements between customers and the developers. This allows the requirements to be transferred in pieces in a less formal fashion.

In order to insure goals are clear, the interview should adhere to the process steps outlined below.

3.4 Communication steps

Following the former researchers' insight, the process steps need to be distributed at different sites to reduce requirements for cross-site communication.

In order to do so, the work item of every component is divided into 5 process steps: reverse-engineering, re-specification, re-design, development, and delivery [Figure 1].

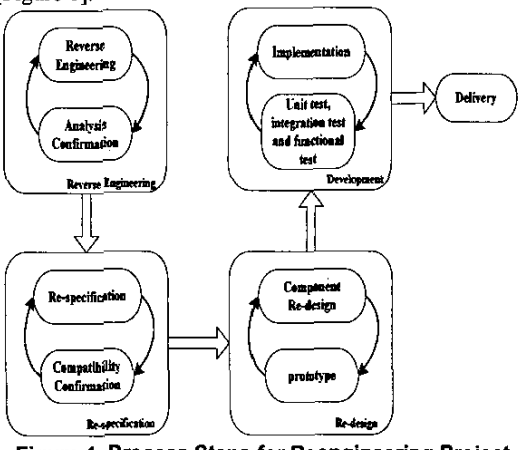

Figure 1. Process Steps for Reengineering Project

In the phase of reverse engineering, the developers comprehend the source code of the original system and deliver the acquired feature-set, specification and design of the old system to the customer. The customer will validate the feature-set and specification based on the business domain knowledge. These two steps in reverse engineering phase are highly iterative and the reverse engineering phase can only be finished when all the results acquired in the code comprehension are validated.

After the customer confirms the feature-set and the specification involved, they start to revise the features according to the newly expanded business scope and the future business support model. The customer sends the new specification to the developers, and the developers will confirm the specification according to the system performance and compatibility issues.

When the re-specification phase is finished, the developers start to re-design the components. In order to accelerate the process, prototyping is suggested to assist the program design. In the re-design phase, the experts of the platform, operation system, and the database will be helpful when some obstacles are encountered.

After the re-design phase, some experts will evaluate the design according to the system performance and scalability, and then make suggestions to other team members. The developers will improve their design following the suggestions from these experts.

When the re-design phase is finished, the developers begin to implement various components. After the components pass unit testing, integration testing and functional testing, the code is delivered to the customer.

To enable the developers exploit their expertise as much as possible, the legacy system analysis and the new system development teams need to be split. This can help reduce the coupling of the tasks.

3.5 Split the analysis and development responsibility

The development team may be divided into two groups. One part of the develop team (reverse engineering group) is responsible for reverse engineering, generating the specification, and send it to the off-site customers. The customers confirm the requirements in sequence and send back the results to second group responsible for forward engineering effort (forward engineering group) [Figure 2].

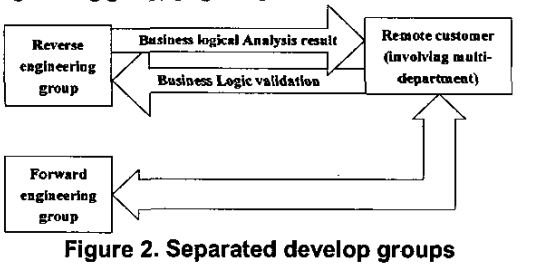

Figure 2. Separated develop groups

As it is highly difficult for the reverse engineering group to comprehend and master the complex business rules in legacy system only from the source code, the analysis result of business rules worked out by reverse engineering group ought to get the confirmation from the remote customer with business rationality in mind.

The face to face communication is by far the most efficient model in communication, besides the periodic visitation between the development team and the customers, the communication between reverse engineering group and forward engineering group may be used to boost the communication efficiency of daily efforts [Figure 3].

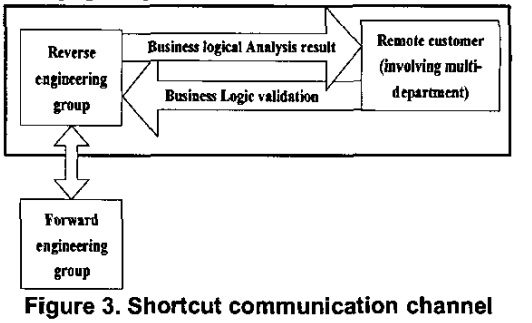

Figure 3. Shortcut communication channel

The reverse engineering group analyzes the legacy system and delivers the business logical analysis result to the remote customer periodically. When all the parts of a

component are analyzed and confirmed, the reverse engineering group creates it into a story.

The reverse engineering group allocates the stories to the forward engineering group when they are available. During the implementation of a story, the forward engineering group may consult the reverse engineering group, and when the content of consultation exceeds its current knowledge, the reverse engineering group will expand the scope of analysis and consult the remote customer as needed.

Therefore a combination of on-site reverse engineering group and off-site customers was set up to play the role of customer in XP parlance and the forward engineering group serves as the original programmer in the project, in order to implement the "On-site customer" XP practice.

4. RESULTS

For Lattice® Trading System reengineering project, there was a 12 member development team in Hangzhou, China, which included 6 reverse-engineering members and 6 forward-reengineering members. In Boston, USA, 4 members assumed off-site customer role for the requirement confirmation and code acceptance.

Besides communication tools such as email, instant message meeting, and teleconference which were used in the communication between the customers and the off-shore develop team, XP practices were implemented to reduce the communication delay and improve communication quality as described in the previous sections. The requirements were communicated in pieces, the development was conducted in small iteration, an organized knowledge database was set up on a website, accessible to developers and customers, to record all the knowledge generated, and a problem tracking tool were developed to record the communication as well. Only one person from Boston paid a two-week visit to kick off the project. The project was completed without major communication issues within 6 months as originally estimated, and with a cost-saving of at least 60% compared with doing the project entirely onshore.

5. CONCLUSIONS

While there are so many methods suggested to be used in global development environment, the customers still have the problems when they are working with off-shore development team.

Extreme Programming is introduced to reduce the communication issues for global software development efforts. The iterative developments reduced the impact of the communication delay since development of iteration can be done in parallel with the confirmation of the requirement for the next iteration. The shortcut communication channel removed most of the communication delay in development and improved the communication quality at the same time. The only delay in the project happens with the release confirmation, where small release cycle is helpful to reduce the duration. Setting up a combination of on-site reverse engineering group and off-site customer for requirement confirmation, we also avoid the expensive on-site customer cost and keep the communication efficiency.

Acknowledgements

This research was sponsored by State Street Corporation and conducted at State Street Zhejiang University Technology Center. Lattice® is a registered trademark of State Street Corporation. All other company and product names are trademarks or registered trademarks of their respective owners.

References

[1] K. Beck, *Extreme Programming Explained: Embrace Change*, Addison-Wesley, Reading, Mass., 1999.

[2] James D. Herbsleb and Audris Mockus, "An Empirical Study of Speed and Communication in Globally Distributed Software Development", *IEEE Transactions on Software Engineering*, Vol. 29, No. 6, June 2003.

[3] E. Carmel, *"Global Software Teams: Collaborating, Across Borders and Time Zones"*, Prentice Hall, Upper Saddle River, N.J., 1999.

[4] Erran Carmel, Ritu Agarwal, "Tactical Approaches for Alleviating Distance in Global Software Development", *IEEE Software*, Vol. 18, No. 2, Mar./Apr. 2001, pp. 22-29

[5] Richard Heeks, S. Krishna, Brian Nicholson, Sundeep Sahay, "Synching or Sinking: Global Software Outsourcing Relationships", *IEEE Software*, Vol. 18, No. 2, Mar./Apr. 2001, pp. 54-60

[6] D.W. Karolak, *Global Software Development*, IEEE CS Press, Los Alamitos, Calif., 1998.

[7] J.D. Herbsleb and D. Moitra, eds., "Global Software Development", *IEEE Software*, vol. 18, no. 2, Mar./Apr. 2001.

[8] R.E. Grinter, J.D. Herbsleb, and D.E. Perry, "The Geography of Coordination: Dealing with Distance in R&D Work", *Proc. Int' ACM SIGROUP Conf. Supporting Group Work*, pp. 306-315, 1999.

[9] J.S. Olson and S. Teasley, "Groupware in the Wild: Lessons Learned from a Year of Virtual Collocation", *Proc. ACM Conf, Computer Supported Cooperative Work*, pp. 419-427, 1996.