A REQUIREMENTS ENGINEERING PROCESS MODEL FOR DISTRIBUTED SOFTWARE DEVELOPMENT
Lessons Learned

Leandro Teixeira Lopes and Jorge Luis Nicolas Audy
School of Computer Science, PUCRS, 6681 Ipiranga Avenue, Porto Alegre, Rio Grande do Sul, Brazil

Keywords: Distributed Software Development, Global Software Development, Global Teams, Requirements Engineering, Requirements Analysis, Requirements Specification.

Abstract: In the growing market of global software development (GSD), requirements engineering emerges as a critical process impacted by distribution. The need of a process to address the difficulties caused by team dispersion in requirements engineering is recognized. The objective of this paper is to present lessons learned from a case study conducted to evaluate a requirements engineering process model for distributed software development. Empirical results were obtained in a multinational organization that develops software with teams distributed in a global setting. The main contribution of this paper is providing an insight in the use of a requirements engineering process model for GSD, as well as, new information on current needs for changes or revision in traditional requirements engineering models.

1 INTRODUCTION

In the software development process, requirements engineering (RE) arises as a key point to the success of projects. Studies present that the main challenges to software projects are related to requirements (The Standish Group International, 1995). Problems in requirements engineering can impact all software project, from design to test and maintenance, increasing cost and schedule.

The increasing globalization in business environments has impacted the software development market (Herbsleb and Moitra , 2001). Aiming competitive advantages as low costs, high productivity and quality in systems development, several organizations decided to distribute their development process inside or outside their countries. India, Brazil and Ireland, as well as several other regions offer fiscal incentives and availability of resources in software development.

When teams are dispersed around the globe, several new challenges are introduced to the requirements engineering process. As an activity communication intensive, requirements engineering is highly influenced by team dispersion, for example. Language and cultural differences can introduce ambiguity, misunderstandings, which are negative to requirements process.

The main objective of this paper is to present results of a case study in a global software development setting, aiming to evaluate the effectiveness of a requirements engineering process model for distributed software development environments. Case study was conducted in a multinational organization that develops software with teams distributed globally. Results are analyzed and consolidated in lessons learned, which can be used for further improvements in the process model used, as well as basis for next studies.

This paper is structured as follows. In section 2 is presented the theoretical basis used as reference for this research. In section 3 is presented the research method. The process model used is presented in detail in section 4. Section 5 details the case study and lessons learned. Final considerations are presented in section 6.

2 THEORETICAL BASIS

Distributed software development (DSD) presents some characteristics that differentiate fundamentally from co-located software development (Karolak,

Teixeira Lopes L. and Luis Nicolas Audy J. (2008).
A REQUIREMENTS ENGINEERING PROCESS MODEL FOR DISTRIBUTED SOFTWARE DEVELOPMENT - Lessons Learned.
In *Proceedings of the Tenth International Conference on Enterprise Information Systems - ISAS*, pages 117-122
DOI: 10.5220/0001682901170122

1998). The requirement engineering (RE) process has several activities that need high communication and coordination, what tends to increase difficulties when in distributed environments.

Although several studies recognize the need to increase knowledge about requirements engineering in distributed environments (Zowghi, 2002)(Damian and Zowghi, 2003), after extensive research, a limited number of papers was found in the topic. These are some of the studies that most influenced this research:

Damian and Zowghi (2003) present findings from case study in two global software development organizations. The main result was a model of impact of distance and the affected requirements activities due to problems of cultural diversity, inadequate communication, knowledge management and time differences. It has provided as important insight into the interplay between culture and conflict as well as the impact of distance on the ability to reconcile different viewpoints related to requirements and requirements process.

Lloyd, Rosson and Arthur (2002) report an empirical study of how groupware can be used to aid distributed software requirements engineering. It presents an analysis of factors that affected the quality of the Software Requirements Specification document written at the conclusion of the requirements process and the effectiveness of requirements elicitation techniques which were used in a distributed setting for requirements gathering.

Zowghi (2002) advocates the development of a different requirements engineering process for global software development outlining some preliminary suggestions on what such process model would include. Research was conducted through field studies where it was concluded that some of the fundamental problems associated with the activities of requirements engineering process are exacerbated when the software development teams are geographically distributed. The study describes briefly the impact of global software development teams in the requirement engineering process and argues that there is a need to investigate and develop requirements engineering process to support global software development

In this sense, (Lopes et al, 2004) present an initial proposal of a process model to this environment. This proposal is based on the adapting the software requirements specification according to the needs of the development team, with a double validation of the requirements artifacts.

3 RESEARCH METHOD

This research is characterized as a study mostly exploratory, since the main research method was the case study. It is possible to justify the use of qualitative methods since it involves the study of the system development process in its real context, with description and the understanding of the state of the art in those situations where practice precedes theory (Yin, 1994).

4 RE PROCESS MODEL FOR DSD

The requirements engineering process model used in this study was based on the proposal presented in (Lopes et al, 2004). The main goal of this process is to reduce the impact of team dispersion in requirements engineering. In this sense it is defined roles and form of evolution of the requirements artifacts with focus on consensus among teams. It is also included in the process a model of natural language specification, with focus on reducing ambiguity and standardizing work among teams.

4.1 Context

Process model considers the existence of physical distance among users, clients and development team. The main groups involved in the process model are the requirements engineering team, the users and clients group, and the development team.

The requirements engineering team is responsible for requirements elicitation, analysis, negotiation, validation and management. In the process model the requirements engineering team has members next to the users and clients, called business analysts and members next to the development team, called application analysts, as presented in Figure 1.

The users and clients group represents the interested parts that requested and contracted the software project, as well as the responsible for using the product built. This team provides information to the software specification.

The development team is composed by the responsible for the development of a specific project, using as input the requirements specified by the requirements engineering team. The development team usually comprehends project managers, testers, developers and support team, among others.

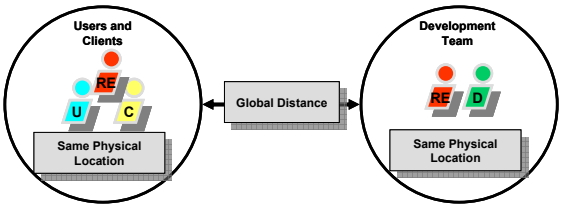

Figure 1: Scenario example - Requirements engineers dispersion.

In the process model, interaction is centered in the requirements engineering team, which is responsible for the requirements artifacts.

According to the process model, the requirements engineering team is the intermediate between the users/clients group and the development team. The requirements engineering team is responsible for creating and maintaining the requirements artifacts, being the only group able to modify these artifacts. Users, clients and developers can evaluate the specification artifacts during all requirements process, asking for changes when necessary.

All changes in the requirements artifacts must be centralized in the requirements engineering team, responsible for controlling these documents. Changes can be requested by any member of the users, clients and development team.

4.2 Process Model

The process model considers the existence of at least one business analyst, and at least one application analyst (see 4.1 Context). Process consists of five steps, as presented in Figure 2 and detailed in sequence.

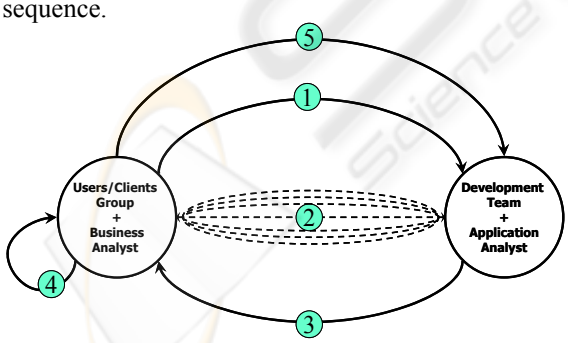

Figure 2: Process model for RE in DSD environments.

Step 1. Initial Requirements Artifacts are sent to Development Team

After creating initial set of requirements artifacts, the business analyst send these artifacts to the application analyst. Initial set of artifacts can comprehend high level documents, as the Vision/Scope document, or even an initial version of a requirements specification document.

Step 2. Requirements Artifacts Analysis and Evolution

Engagement of development team happens, in general, in the beginning of project, what helps the team to be aware of needs and rationales of the software project. After receiving the requirements artifacts from the business analyst, the application analyst try to deep understand artifacts contents and context. Development team also uses these documents to contextualization.

During this phase requirements artifacts are adapted to reduce potential sources of problems. Ambiguity and lack clearness are likely when teams have various cultures and languages. Questions arise and are cleared among teams, with high volume of communication in this step.

Communication among teams, more than helping clarifying the artifacts contents, aims to obtain consensus on the specification being written, aligning multiple visions on requirements.

According to the level of detail of the artifacts received from the business analysts, it may be necessary to complement them, eliciting and negotiating new requirements through contacts with stakeholders.

Requirements artifacts can be rewritten or adapted to standardize inbound documents. Application analysts and development team can apply phrase structures, patterns of document and glossary, use case and requirements formats, for example, to avoid different formats of documents to each project. This need increases when considering metrics application, where these differences can introduce deviations.

Step 3. Requirements Artifacts are sent for all Teams' Approval

Once the requirements artifacts are finished, it must be approved. In this step artifacts are sent back to be verified by specification team.

Step 4. Validation and Approval of the Requirements Artifacts

Key members of each team shall verify the requirements artifacts to assure that after adapted they still reflects the needs and objectives of stakeholders. Communication during the third step keeps teams aware of the adaptation process, what reduces the effort to validate requirements artifacts.

Step 5. Requirements Artifacts Final Version is Defined
After the formal approval of all key team members, the final version of requirements artifacts is defined. Then, development team uses this version as basis for modeling, coding and testing software.

4.3 Model for Natural Language Requirements Specification

The requirements engineering process model for DSD includes a model for natural language requirements specification, aiming to reduce ambiguities and standardizing the work among teams. The model for natural language specification was developed with focus on requirements elicitation and documentation, to capture needs and goals of users and clients.

When the development team is responsible for supplying several projects with various users and clients, it is natural that the initial requirements artifacts provided by the business analyst have different standards and formats. Consequently, the level of detail of requirements, document format, glossary, format of use cases and requirements, for example, can influence the development process and metrics related. Using a model of natural language specification, the input documents are standardized, reducing the impact of the variety of standards and formats of documents provided by the business analyst.

This model can vary among organization, according to the standards chosen. It can be influenced by language used, background knowledge of analysts, development process, etc.

In this study was used a requirements meta-model and a text structure. The requirements meta-model comprehends a set of definitions used to classify and relate the information gathered during elicitation. The text structure defines the main phrase structures to be used in each class of information, with the goal of simplifying the understanding of the information represented.

This approach was built based on the study of the main definitions of requirements (Armour and Miller, 2001), (Goguen, 1996), (Siddiqi and Shekaran, 2006), (Leite and Leonardi, 1998), (Thayer and Dorfman, 2000) and phrase structures in literature (Damian et al, 2002), (Rational Software, 2003), (Kamsties, 2001). Model was tested preliminarily using historical data of two projects of software development.

5 CASE STUDY

Aiming to evaluate the team perception on the process of requirements engineering in DSD and the quality of the requirements artifacts produced, it was conducted a case study in two projects of distributed software development. Initially, it was developed a case study protocol, where the objective, scope, unit of analysis, procedures, dimensions and questions were detailed.

The unit of analysis is composed of software development projects that used the process of RE to DSD environments proposed. In this sense, it was selected an organization that conducts projects of global software development to apply and monitor the process proposal in real projects since the beginning. Organization selected had three software development units around the globe. The software process used in the organization is based on MSF (Microsoft Solutions Framework), and in known methodologies, like RUP (Rational Unified Process) and PMI (Project Management Institute). The unit where the case study was conducted is recognized as a SW-CMM Level 2 organization. In the requirements process, project teams commonly used an "over the wall" approach (Al-Rawas and Easterbrook, 1996), where the specification was built by the business analyst next to users and clients and sent to be developed by a globally distant team.

Two projects were selected to apply the process proposal, called Project 1 and Project 2 from now on. These projects followed the process proposal presented in Chapter 4. When team had the final requirements artifacts, it was applied a survey.

A semi-structured survey was used as a data collection instrument, mainly with questions in Lickert scale of five levels. The data collection instrument was validated by two senior researchers and a project manager from the organization, being refined based on their suggestions.

A pretest was conducted in the preliminary version of the instrument with two technical leaders of the organization selected, aiming to identify problems, ambiguities and improve the question statements. Final version of instrument was sent to respondents through e-mail.

In Project 1 survey had seven respondents, including the project manager, the application analyst, the technical leader, developer and testers.

Project 2 had six respondents including the project manager, the application analyst, the system architect and developers. There was no answer fro test team which was located in other physical site.

After receiving answers, results were analyzed with content analysis techniques and the statistical module of Excel. Project documents were used to triangulate data and increase results reliability.

5.1 Case Study Results

Based on the survey results, as well as in observations made during the projects development, several highlights and problems were identified.

5.1.1 Highlights

The highlight points identified in the process model are mainly related to communication, level of detail of information gathered, trust, clearness and benefits to subsequent phases of software development. When requirements are easier to understand, validation is simplified. Feedback channels are more efficient, with a better evolution on requirements artifacts. Besides, teams trust that requirements are understood, improving their relationship.

The standard structure of requirements allowed a better communication among teams, once they had improvements in the form of expressing needs and goals. A better communication promoted by several interactions among teams was a consensus among interviewed. Besides, requirements were considered more clear and specification richer in detail.

With requirements more clear and deep in detail, documentation and testing activities, as well as MSF phases of planning and developing are improved. Respondents also pointed that the requirements specification was more verifiable using the preliminary process.

5.1.2 Problems

When evaluating the process model used, the need of capturing a wider range of information was pointed, mainly by the project manager of the Project 1. This limitation is most linked to the model for natural language specification.

Contributions to estimation process were expected, with requirements more clear, correct and detailed. However, it was not pointed in the survey. As the estimation process used in the organization is based on historical data, it may be necessary a higher number of projects using the process model to adequate estimative.

When considering the characteristics of a good SRS, as defined in IEEE Std 830-1998, consistency and ranking of requirements had the worst evaluation in survey. The former had a high standard deviation, being not a consensus among respondents.

5.2 Lessons Learned

Based on consolidated results, were identified the case study lessons learned, as presented below:

Lesson 1 - Interaction among teams to evolve the requirements artifacts allows a better understanding of the rationale of requirements.
Interaction among development team, users and clients aiming to evolve requirements artifacts increase comprehension of the rationale that guides the software development. Without interaction, software design and codification can start without aligning team vision on requirements. Also, this way, some usual problems like “over the wall” development (Al-Rawas and Easterbrook, 1996), is avoided once teams must ensure a complete understanding of the requirements specification contents instead of just throwing the specification “over the wall” to the next team in each software development phase.

Lesson 2 - The use of phrase structures to natural language requirements contributes to improve communication and understanding among teams.
When evaluating the case study results, a key point to improve communication among teams, as well as clearness in requirements was the phrase structures used to specify requirements in natural language. As there was native language difference among teams, the use of simple phrase structures allowed a better comprehension of written requirements. Communication was also influenced by phrase structures, once teams started using it when interacting. Clearness by the use of phrase structures also promoted an improved communication.

Lesson 3: It is necessary improvements in process used to better understand and capture teams’ context information.
There was no clear improvement in capturing context information with the process used. Although team interaction to evolve requirements artifacts could help teams to share context information, there was no specific activity with this goal in the process used. User and clients context information is important to understand the location where the software being developed will be used and how it can affect the software specification. A possible alternative is presented by Mahemoff (1998), through the use of a location specific information database.

Lesson 4: The preliminary process needs new activities to reduce inconsistencies and improve requirements ranking.

In the case study, respondents pointed that the main characteristics of the requirements specification that needed improvements were requirements consistency and ranking. In this sense, it is necessary new activities in the process used aiming to reduce inconsistence and improve requirements ranking.

Lesson 5: Improvements in preliminary process must be conducted to capture a wider range of information as well as discover hidden requirements.

Other improvement point in the process used is the definition of forms to capture a wider variety of information, as well as discover hidden requirements. The use of natural language to specify requirements was not enough to represent all information needed. It is clearly necessary to use other forms of representation according to the project characteristics.

6 FINAL CONSIDERATIONS

Requirements engineering has a critical role in software development process. RE artifacts are used in all subsequent phases of development. Estimative, modeling, development and test are made based on requirements.

There are several difficulties in requirements engineering process. Most of those difficulties are increased when software development teams are distributed. Some new difficulties appear.

Considering the growing adoption of distributed software development, there are few studies about the impact it has in requirements engineering. In these studies, the technical aspects aren't considered in detail. It is clearly necessary processes, patterns and tools to address difficulties cause by team distribution in requirements engineering.

This study presents results from a case study in two projects using a process model for requirements engineering in DSD environments. It contributes presenting an insight towards improvements for the model used as well as new empirical information on the theme.

REFERENCES

The Standish Group International (1995). *Chaos Report*. [Online], Available: http://www.standishgroup.com/.

Herbsleb, J.; Moitra, D. *Global Software Development*. IEEE Software. 2001.

Karolak, D. *Global Software Development – Managing Virtual Teams and Environments*. IEEE Computer Society. Los Alamitos, EUA. 1998. 159p.

Thayer, R.; Dorfman, M. *System and Software Requirements Engineering – Second Edition*. IEEE Computer Society Press Tutorial. 2000. 528p.

Zowghi, D. *"Does Global Software Development Need a Different Requirements Engineering Process?" Proceedings of International Workshop on Global Software Development* (ICSE 2002). 2002.

Damian, D.; Zowghi, D. "An insight into the interplay between culture, conflict and distance in globally distributed requirements negotiations". *Proceedings of the 36th Hawaii International Conference on Systems Sciences* (HICSS'03). IEEE. 2003.

Lloyd, J.; Rosson, M..; Arthur, J.. Effectiveness of Elicitation Techniques in Distributed Requirements Engineering. *Proceedings of the IEEE Joint International Conference on Requirements Engineering* (RE'02). IEEE. 2002. 8p.

Yin, R. K, *Case study research: design and methods*, Sage, 1994.

Lopes, L.; Prikladnicki, R.; Majdenbaum, A.; Audy, J. "Distributed Requirement Specification: Minimizing the effect of geographic dispersion". *Proceedings of the 6th International Conference on Enterprise Information Systems*, Porto, Portugal, 2004.

Leite, J.; Leonardi, M. "Business Rules as Organizational Policies". *Proceedings of the Ninth International Workshop on Software Specification and design*. IEEE Computer Society. 1998.

Armour, F.; Miller, G. *Advanced Use Case Modeling*. Addison Wesley. 2001

Goguen, J.. "Formality and Informality in Requirements Engineering". *Proceedings of the 2nd International Conference on Requirements Engineering* (ICRE '96). IEEE. 1996.

Siddiqi, J.; Shekaran, M. *Requirements Engineering: the emerging wisdom*. IEEE Software. 1996.

Damian, D. et alli. "An industrial experience in process improvement: An early assessment at the Australian Center for Unisys Software". *Proceedings of the 2002 International Symposium on Empirical Software Engineering* (ISESE'02). IEEE. 2002.

Rational Software. *The principles behind good requirements*. Webinar. [Online] http:// www.rational.com/September 2003.

Kamsties, E. 2001. *Surfacing Ambiguity in Natural Language Requirements*. PhD Thesis. Fraunhofer-Institue für Experimentelles Software Engineering. 2001.

Al-Rawas, A.; Easterbrook, S. Communication problem in requirements engineering: A field study. *Proceedings of the Westminster Conference on Professional Awareness in Software Engineering*, 1996, London.

Mahemoff, M. J.; Johnston, L. Software Internationalisation: Implications for Requirements Engineering. *Proceedings of the Australian Workshop on Requirements Engineering*, 1998, Geelong, Australia. p. 83-90.