Requirements Specification in Distributed Software Development – A Process Proposal

Leandro Lopes, Rafael Prikladnicki, Jorge Audy
School of Computer Science - PUCRS
6681 Ipiranga Av., Porto Alegre, RS, Brazil
{lteixeira, rprik, audy}@inf.pucrs.br

Azriel Majdenbaum
azm57@hotmail.com

Abstract

Distributed software development presents several characteristics that differentiate from co-located software development. Currently, one of the main difficulties imposed by geographically distributed software teams is the requirements engineering process. Traditional approaches to requirements process don't cover the fundamental difficulties of dispersion, like communication and coordination. The objective of this study is to present a proposal to adapt the specification process to address the main challenges found in distributed software development environments.

1. Introduction

In last years, software became a vital component in business. Organizations are depending on software as their competitive advantage. In parallel, economy has converted national markets in global markets, creating new forms of competition and collaboration [1]. However, global software market has experienced several crises. Many software projects have failed while the number of capable professionals could not attend to the growing demand. In this context, the distributed development of software appeared as an alternative to organizations.

Requirements engineering, a critical phase of software development, presents several new challenges and exacerbates its fundamental ones when in distributed environments [2][3]. In this case, the requirements engineering seems to be an even more critical phase. Normally, the requirements' gathering and specification occurs in meetings having all the participants (project team, users and customers) in the same place. This facilitates the communication, becoming easier any negotiation or existing conflict resolution. In the Software Engineering literature, software requirements represent the interests of customers and users and are the heart of any project [4]. The quality and the capacity of analyzing and managing the requirements of software projects not only affect the final product quality, but also the time necessary to satisfy the requirements. A badly identified requirement can compromise the project success, generating delays or project cancellation.

In distributed software development environments the challenges become even more significant. This paper has as objective to present a proposal of specification process to address the main challenges found in distributed software development environments. This paper has the following structure: section 2 and 3 present the theoretical base in distributed software development and requirements engineering; section 4 describes the related studies; section 5 describes the process proposed; section 6 present the conclusions, future studies and the research limitations.

2. Distributed software development

As part of the globalization efforts currently pervading society, software project teams have also become geographically distributed on a worldwide scale. This characterizes the Distributed Software Development – DSD (Global Software Development – GSD, when the distribution becomes global).

Tools and technological environments have been developed over the last few years to help in the control and coordination of the development teams working in distributed environments. Many of these tools are focused in supporting procedures of formal communication such as automated document elaboration, processes and other non-interactive communication channels.

Moreover, [1], [5], [6] and [7] point out that GSD is one of the biggest business-oriented challenges that the current environment presents under the software development process point of view. Many companies are distributing its software development process in countries such as India, China and Brazil. Frequently this process also occurs in only one country, particularly in regions with tax incentives or critical mass in some skill or resource areas.

Organizations search for competitive advantages in terms of cost, quality and flexibility in the area of software development [7], looking for productivity increases as well as risk dilution [8]. Many times the search for these competitive advantages impels organizations to search for external solutions in other countries (offshore outsourcing). This epitomizes the traditional problems and the existing challenges in DSD.

2.1. Centrifugal and Centripetal forces of global software teams

According to [9], software globalization is like a centrifugal force that propels things outwards from the center as it disperses developers to the far corners of the world. A centrifugal force must be balanced by a centripetal force, a counter force that is directed into the center.

The centrifugal forces, the problems that pull the global software team apart and inhibit its performance are: cultural differences, geographic dispersion, loss of communication richness, coordination breakdown and loss of "teamness".

The centripetal forces, the solutions that pull the global software team together and make it more effective are: telecomm infrastructure, product architecture, team building, development methodology, managerial techniques and collaborative technologies.

3. Requirements engineering

Requirements engineering (RE) plays an important role in the software development. Thayer [10] defines requirements as: "1. A capability need by a user to solve a problem or achieve an objective. 2. A capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document …". Therefore, the compliance with requirements determines the success or the failure of a project. The requirements are identified, registered, organized and verified during the project development. Requirements are basis to establish and keep the agreements firmed between the project team, users and customers.

The literature states that the problems related with requirements engineering are one of the main reasons for software projects failures. This means that the final product does not have all the requirements gathering from users and customers. It can be identified that [4]:

- Requirements are not easy to be described in words;
- There are different types of requirements in different levels of details;
- It can be impossible to manage the requirements if they cannot be controlled;
- Most requirements change during the project life cycle.

Therefore, it is not difficult to find errors in the requirement specifications, and they can have a large impact in the project costs. An estimative shows that 40% of the requirements generate rework during the project life cycle. It is evident that the earlier a problem is detected and solved (especially during the requirements phase) the earlier other problems are minimized in the following project phases. But in contrast, it is observed a short time for activities related to requirements, not considering the project type or environment where this phase occurs.

3.1. RE Process

According to [11] a requirements engineering process is a structured set of activities which are followed to derive, validate and maintain a systems requirements document. Process activities include requirements elicitation, requirements analysis and negotiation and requirements validation [12]. Figure 1 presents a spiral model of the requirements engineering process.

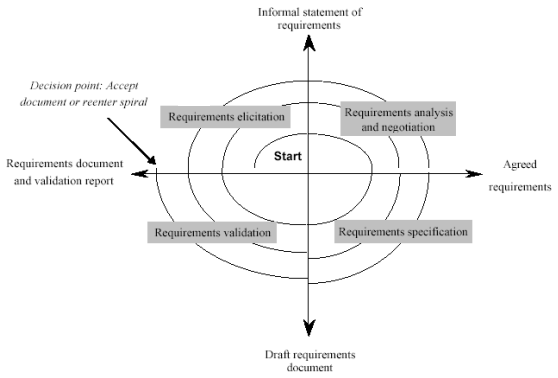

Figure 1 - A spiral model of the requirements engineering process [12].

4. Related Studies

Damian [3] presents 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's research [13] reports 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.

Mahemoff [14] presents a study about how cultural factors affect requirements in software internationalization. The main contribution is a form of classifying cultural factors that impact in requirements, based on overt and covert factors. It also proposes that the classification could form the basis for a repository of cultural information accessible by developers.

Zowghi [2] 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.

4.1 Research group results

During research development, it was proposed a reference model for requirements engineering in distributed software development [17].

The categories identified in this study are culture, communication, knowledge management and technical aspects, as shown in Figure 2. The main difference comparing to similar researches is the category of technical aspects once it brings a new perspective of view.

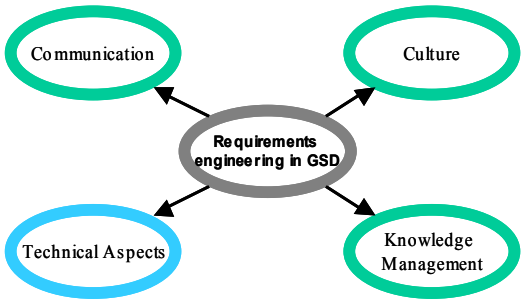

Figure 2. Categories related to requirements engineering in GSD

Each of these categories has several factors involved and the relationship between them is close, what makes it difficult to define the limits of each one.

Communication: the requirements engineering process depends largely on communication. Clear communication is critical to avoid misunderstandings and conflicts. The core factors related to communication found in this study are language, time zone and communication medium.

Culture: culture of the teams members influence the requirements engineering process. Both, organizational and national culture can affect requirements engineering. The main factors related to culture are context, attitude and values.

Knowledge Management: The requirements engineering process deals with large volume of information. Collect, process, store and make available the knowledge related to the requirements process, as well as unify the organizational vision are needs that should be addressed with knowledge management. The

main factors identified related to knowledge management are expectations, awareness and management of cultural information.

Technical aspects: Several technical aspects affect requirements engineering in distributed environments. The requirements process depends on coordination and control mechanisms, for example, that can help to reduce the impact caused by team distribution. The main factors found are patterns, process and configuration management.

The need for a requirements process that address distributed development issues was already identified in [2]. This study conducted in the same direction. After identifying several factors that impact requirements engineering in distributed environments, we worked towards proposing a process aiming to minimize this impact. This proposal is presented in sequence.

5. Process proposal

This process proposal is based on one common distribution and division of roles of offshore outsourcing, as described in sequence.

Offshore outsourcing has several roles related to requirements engineering. According to the software process used and the organization structure, these roles can have different names and tasks. Aiming to simplify the understanding, in this study it will be considered mainly the following roles in project team:

Business Analyst: Conducts elicitation, analysis, negotiation, specification and validation of requirements with stakeholders.

Application Analyst: Responsible for modeling software based on the specification. This role is also responsible for maintain requirements traceability with code and test.

In co-located development one person or group can, sometimes, perform these roles. In offshore development these roles tend to be performed by different groups, once the development team is usually distant from the user and client group.

Therefore, it's possible to identify two teams with different tasks in requirements engineering in offshore environments. In this study, the business analyst team is called specification team and the application analyst team is called development team.

The team's distribution can be represented in a similar form of [16], as shown in Figure 3. This figure represents the specification team in United States and the development team distributed in Brazil, China and India.

It is important to recognize that beyond distribution between specification and development teams, there is possible to have distribution between the specification team and stakeholders, what affects the requirements engineering phases related to those groups as shown in [13] and [14].

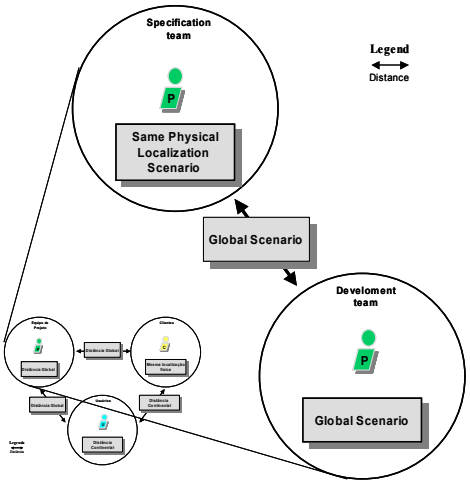

Figure 3. Requirements engineering team distribution

Another important problem commonly faced by offshore organizations that develop software to more than one (internal or external) client is the diversity of structures, patterns and level of detail used in requirements documentation. Once there are several different clients, the diversity leads to difficulties in obtaining metrics, estimative and organizational standards.

5.1. Process

The proposed process intends to reduce distribution difficulties in requirements process by introducing tasks of adaptation and understanding the SRS in development team. It is composed of five steps, as presented in Figure 4 and described below.

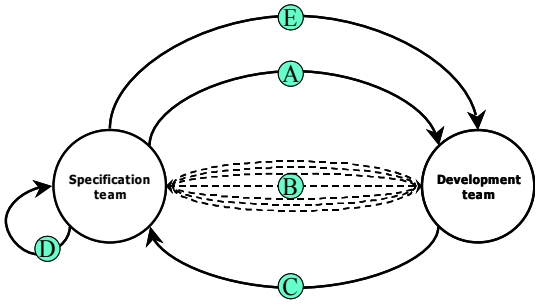

Figure 4. Proposed process

A. SRS first version is concluded and sent to development team.

While executing tasks of elicitation, negotiation and specification, the specification team writes the first versions of SRS. After concluding these tasks the document is sent to development team for adaptations.

In co-located environments, at this moment, modeling and coding activities could begin. However, considering the context in study, difficulties due distribution can reduce the SRS understanding by development team. Cultural and language differences, for instance, can cause misunderstandings, and result in a product that doesn't attend to costumer needs.

B. SRS analysis and adaptation by development team.

According to the moment of engagement of development team, this can be the first contact of this with the project.

Engagement of development team occurs, in general, in the beginning of project, what helps the team to be aware of needs and rationales existent. The context can be understood through documents like Vision/Scope. However, it's hard to be aware of all requirements evolution. Some new needs can appear and several contacts with stakeholders are made.

After receiving SRS from specification team, the first action of development team is try to deep understand the requirements and its context.

During this phase the SRS is adapted to reduce potential sources of problems. Ambiguity and lack clearness are likely between cultures and languages. In cases of great difficulties, SRS can be completely rewritten.

The information obtained while writing SRS tends to be deeper than the information it contains. The process of adapt or rewrite the SRS permits the development team to get the information not explicitly written in the document. Several questions arise and shall be cleared with specification team.

Communication between team is heavy during this phase. Beyond clearing information, communication occurs to build agreements, simplifying step D.

Sometimes, clarifications asked to specification team can lead to new contacts with client or users, if teams don't know the information. Therefore, specification quality tends to increase.

To rewrite or adapt the SRS can also be used to standardize inbound documents. Commonly, one development team can attend to several specification teams, which tend to write the SRS differently. 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 necessity increases when considering metrics application.

C. SRS adaptation is concluded. SRS is sent to specification team approval.

Once the SRS is completely adapted, the new document must be approved. In this step the document is sent back to be verified by specification team.

D. SRS validation and approval by specification team.

Specification team shall verify the SRS to assure that after adaptation it still reflects the needs and objectives of stakeholders.

An apparent difficulty in this step would be the effort need to validate the document and assure it still reflect client needs. However, communication occurred in step C maintains specification team aware of the adaptation process, what reduces the effort need to validate SRS.

E. SRS Final version is defined.

After the approval by specification team, the final version of SRS is defined as approved SRS. Then, development team uses this version as basis for modeling, coding and testing software.

6. Conclusion

Requirements engineering has been considered a critical area in software development. Its study in distributed software environments points an exacerbation of fundamental difficulties and the raise of new challenges. To address these difficulties, new

techniques and processes are clearly needed. This research presents first steps towards a process to address the differences related to distribution.

Considering the proposed process, communication issues due to language are partially addressed during SRS adaptation. While the SRS is adapted or rewritten, ambiguity and lack of clearness in requirements, caused by language differences, is reduced. Problems with time zone tends to arise during the process, but as development team increases its comprehension of requirements, the frequency of communications between teams decreases, reducing this difficulty. Tools to support requirements engineering process can aid in communication, providing discussion areas and a common repository of requirements information.

Cultural issues in requirements like context and values tend to be reduced when SRS is adapted. However, these issues associated with the attitude of team members can lead to conflicts. It may be addressed by training teams in soft skills like trust and cultural differences [16].

Knowledge management is partially addressed through the use of common documents and process, once teams have common expectations and are aware of each other roles. Management of cultural information can be addressed through some techniques as in [14].

Technical aspects are addressed through the process definition. Patterns like phrase and document structure can be applied during adaptation.

Moreover, another important contribution of the process is helping to achieve SRS standardization in the beginning of development process, concerning to phrase structure, patterns, and level of detail, for instance. Through the process, it is possible to adequate the SRS to reflect organizational characteristics, what improves, among others, metrics and estimative.

The next research step involves empirical studies on the proposed process. We are applying the process in two global projects to investigate its effectiveness in distributed requirements engineering.

Initially, it was developed a RE model associated to the proposed process. This model is composed of a requirements consensus, a requirements structure and requirements phrase structure.

The requirements consensus comprehends of a common vision of requirements to the development organization. It captures the vision of the groups involved, as application analyst and technical leaders. The requirements consensus helps assuring the use of a standard description of requirement through the organization.

The requirements structure defines the types of requirements and how they can be related. The structure also defines standards to fill the SRS.

The requirements phrase structure defines standard phrase structures linked to the requirements types. These structures are language specific and help reducing ambiguity and improving requirements clearness.

Based on the process proposed and in the RE model, we are beginning empirical studies in two organizations of information technology that develop software globally. It will be conducted a case study in two projects distributed globally. The objective of the case study is to evaluate the perception of the team members in relation with the process, the RE model and the final SRS quality.

7. References

[1] Herbsleb, J. D., and Moitra, D., *Global Software Development*, IEEE Software, pp. 16-20. March/April/2001.

[2] Zowghi, D., Does Global Software Development Need a Different Requirements Engineering Process?, *Proceedings of International Workshop on Global Software Development – ICSE 2002*, Orlando, Florida, USA, 2002, 53-55.

[3] 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 Science – HICSS'03.* Hawaii, USA, 2003

[4] Oberg, R., Probasco, L., and Ericsson, M., "Applying Requirements Management with Use Cases", *Rational Software White Paper*, Cupertino, CA, 2000, pp. 3-5.

[5] Grinter, R. E., Herbsleb, J. D., and Perry, D. E, *The Geography of Coordination: Dealing with Distance in R&D Work.* ACM, 1999.

[6] Herbsleb, J. D., Mockus, A. Finholt, T. A., and Grinter, R. E, *An Empirical Study of Global Software Development: Distance and Speed*, IEEE, 2001.

[7] Prikladnicki, R., Peres, F., Audy, J., Móra, M. C., and Perdigoto, A., Requirements specification model in a software development process inside a physically distributed environment, *Proceedings of ICEIS 2002,* Ciudad Real, Spain, 2002.

[8] McConnel, S., *Rapid Development***.** Microsoft Press, 1996.

[9] Carmel, Erran. Global Software Teams – Collaborating Across Borders and Time Zones. *Prentice Hall*. 1999. 269p.

[10] Thayer, Richard; Dorfman, Merlin. System and Software Requirements Engineering – Second Edition. *IEEE Computer Society Press Tutorial*. 2000. 532p

[11] Sommerville, Ian; Sawyer, Peter. Requirements Engineering – a good practice guide. *Wiley*, 1997.

[12] Kotonya, G. Sommerville, I. "Requirements Engineering – Processes and Techniques". *Wiley*. 1998.

[13] Lloyd, W., Rosson, M, Arthur, J. Effectiveness of elicitation techniques in distributed requirements engineering. *Proceedings of the IEEE Joint International Conference on Requirements Engineering (RE'02)*. 2002.

[14] Mahemoff, M, Johnston, L. Software internationalization: Implications for requirements engineering. *Proceedings of the Third Australian Workshop on Requirements Engineering*. Geelong, Australia, 1998, 83-90.

[15] Damian, D., The study of requirements engineering in global software development: as challenging as important, *Proceedings of International Workshop on Global Software Development – ICSE 2002*, Florida, USA, 2002.

[16] Prikladnicki, R., Audy, J., and Evaristo, R., Distributed Software Development: Toward an understanding of the relationship between project team, users and customers, *Proceedings of ICEIS 2003,* Angers, France, 2003.

[17] Lopes, L.; Audy, J. "Towards a reference model for requirements engineering in distributed software development". Proceedings of 16th International Conference on Advanced Information Systems Engineering Forum (CAiSE Forum). 2004.