A Communication Process for Global Requirements Engineering

Sajid Ibrahim Hashmi  
Lero - the Irish Software Engineering Research Centre  
University of Limerick, Ireland  
Sajid.Hashmi@lero.ie

Fuyuki Ishikawa  
National Institute of Informatics  
Tokyo, Japan  
f-ishikawa@nii.ac.jp

Ita Richardson  
Lero - the Irish Software Engineering Research Centre  
University of Limerick, Ireland  
Ita.Richardson@lero.ie

ABSTRACT
Globally distributed software development teams face problems with software development life cycle phases, as the distributed nature of each of these phases make it even more challenging to communicate between the stakeholders. Global distance can give rise to incomplete requirements handovers which make the situation more difficult. It is important to address this issue as the end product is likely to deliver less business value when such problems arise. In this research, we propose a process to facilitate non-verbal communication among globally distributed requirements engineering teams. The focus of this research is the situation that occurs after requirements are handed to another site. Our proposed process endeavors to ensure that incomplete and conflicting requirements are identified and mitigated.

Categories and Subject Descriptors
D.2.9 [Software Engineering]: Management – life cycle, programming teams

General Terms
Management, Documentation, Verification

Keywords
Global software development (GSD), software development teams, requirements engineering (RE), global requirements engineering (GRE), communication

1. INTRODUCTION
Advancement in communication channels has had significant influence on the ways companies have been doing their business. Software industry has been one of the major beneficiaries of this technological revolution as computers and the associated channels have had a major contribution to this technological age. This has resulted in increased motivation for software organizations to not only look for clients in different geographical locations but also to outsource software life cycle activities to distance organizations.

Global Software Development (GSD) entails software development distributed across geographical borders. Teams dispersed across multiple geographical locations carry out and accomplish software development tasks [1]. Organizations expect to benefit from enhanced business value through advantages such as round the clock software development, availability of skills and labour, and a reduction in overall project costs. This kind of development model emphasizes on the need for communication among the development teams [2][4].

Software organizations involved in conducting GSD projects normally split up the software development life cycle activities among their teams depending on their geographical location and their nearness with the client. Terms like “onsite” and “offsite” are used to classify the software development activities in which they are involved. Onsite is referred to a team that is located close to a client and the term offsite is used for the one which is remotely located and could possibly be the actual contracting company with a team of software developers and testers. The onsite team works closely with the client in order to elicit their business requirements. The requirements gathered and managed by the onsite team are handed over to the offsite team so that software development can be carried out. This practice is termed as requirements handovers.

Since requirements engineering resides in the problem space [3], it has to overcome several challenges. Especially in case of GSD, an RE process has to incorporate particular strategies in order to deal with multifaceted challenges like global distance and communication barriers. Therefore, a key challenge when carrying out GSD projects is how to effectively communicate when requirements are handed over. This requires an effective communication process especially when the teams are in different time zones. The teams involved in GSD projects often appoint one of their colleagues to work beyond the normal working hours to answer queries from the other team. The strategy may work well for certain situations but the use of natural language to discuss clarity on requirements is likely to create ambiguity [8].

The scope of this research is twofold: first, it focuses on the need for communication after requirements handovers while addressing deficiencies with the existing communication mechanisms being used for GSD. Second, it highlights situations where incomplete requirements could be handed over and proposes a process to facilitate nonverbal communication among the teams. While doing so, the process facilitates verification and validation activities for the conflicting situations which arise after the requirements handovers. The remainder of the paper is organized as follows: Section 2 highlights the challenge associated with global requirements engineering and communication issues faced...
by the geographically distributed software development teams. Section 3 describes our literature review followed by research methodology in Section 4. Section 5 details the proposed process and its preliminary validation using a case study in Section 6, and finally Section 7 concludes the paper.

2. CHALLENGES WITH A GRE PROCESS

Distributed requirements engineering has been a problem area [6][7][9][14][16]. A typical requirements engineering process [10] involves communication on negotiation and validation of requirements which is easier when teams are collocated, as they can communicate face to face. But things are not that easy when teams are in different time zones. Moreover, effective communication and coordination are always required. In order to understand the challenges associated with a global requirements engineering (GRE) process, we consider a multisite GSD scenario in which 3 sites are involved (Figure 1). One of the teams (A) is onsite with the client C with team members involved in different roles across each team. The business and technical consultants (or technical leads) onsite can communicate using standard communication mechanisms but problems arise when they have to communicate with their colleagues offsite (B) who are in different time zones. These teams (A & B) make use of internet as an underlying communication mechanism on top of which they use communication tools like emails, instant messaging, and video conferencing. However, since the time zone difference results in delays in communication between these sites, the team members from the offsite software development team (B) are likely to make assumptions in cases where incomplete requirements were handed over to them. Traditional communication mechanisms cannot ensure timely verification and validation of the assumptions made by the development team, in the aforementioned scenario. Our conclusion from this is that most of the existing mechanisms or tools involve natural language communication which itself minimizes the chances of asynchronous communication for distributed requirements engineers. This can result in verification and validation of requirements being delayed. In this research, we focus on the GRE process, particularly at handover stage. We facilitate non-verbal communication through requirements documentation.

In Table 1, we list the tools that have been identified by [5][11][12] and are used by GSD teams. In addition, we present their scope for communication on global requirements engineering activities and their limitations with relation to time difference among the teams. In addition, there is some research [13] and open source tools [11] but they do not solve the dilemma in which we are interested – developers making assumptions about requirements which are not clearly stated. Our conclusion from this is that most of the existing mechanisms or tools involve natural language communication which itself minimizes the chances of synchronous communication for distributed teams. This can result in verification and validation of requirements being delayed. In this research, we aim to focus on the requirements engineering process, particularly at handover stage.

![Figure 1. A global RE process workflow - issues are highlighted](image)

<table>
<thead>
<tr>
<th>Tools</th>
<th>Usefulness</th>
<th>Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>IM Environments</td>
<td>Provides an instant way to communicate with distance members</td>
<td>Usability is limited in different time zones. Cannot be used for formal verification and validations</td>
</tr>
<tr>
<td>Email</td>
<td>Works well for making formal queries, notifications, or document exchanges</td>
<td>Although it has been the most conventional method but does not guarantee a timely communication</td>
</tr>
<tr>
<td>Video conferencing</td>
<td>Complements IM environments with oral and video facility</td>
<td>Availability of team members could be an issue for different time zones. Use of natural language as well as some technical overheads are involved</td>
</tr>
<tr>
<td>Wikis</td>
<td>Can be used to assign tasks. Keep everyone updated about the activities</td>
<td>Mainly used as a document repository. There is hardly any mechanism for verification or validation</td>
</tr>
<tr>
<td>Requirements management</td>
<td>Provide support for grouping, and structuring of requirements</td>
<td>Asynchronous communication is facilitated through the use of comments only. Moreover, not optimized for nontechnical users, high license costs</td>
</tr>
</tbody>
</table>
3. RELATED WORK

Our literature review targets inter-related topics: communication tools in GSD, software engineering processes with emphasis on GRE, and how distributed teams communicate. We also had to investigate existing synchronous communication mechanisms and their deficiencies.

Research on requirements engineering of GSD projects have mainly been through empirical studies justifying the significance of the domain [14][15][16]. Their results confirm the problems related to communication in distributed requirements engineering but they do not provide solutions to those issues that cover technical as well as non-technical facets. Moreover, the existing work on the domain mainly focuses on early phase activities such as requirements planning and elicitation. We have not seen the facilitation of requirements handovers for GSD team members who deal with consequent communication challenges.

Cheng et al. [17] have listed some measures which enhance collaboration, for example, Email, IM (instant messaging), and screen sharing. Email and IM can be used for communication but their context and scope is quite limited in terms of the way they can facilitate handover. They suit best only when synchronous communication is involved. For example, the screen sharing option would not be useful when users are in different time zones. While this practice could be helpful at coding stage of the development where a developer could ask a colleague to help him/her figure out any problems with his/her code, we have not established its usefulness during requirements handover.

Some industry based tools [18][19] have also been proposed to facilitate collaboration among software development teams. Both tools support shared document reviews, distribution of documents among teams, and some mechanisms for performing common collaborative tasks. However, they facilitate task level collaboration only and are mainly focused on the software development phase of the life cycle, i.e. keeping track of the task level audit trail in addition to a basic collaboration environment. In addition, some research prototypes [13] have also been developed but their usefulness across different time zones remains questionable. Moreover, they have been found useful for only capturing and tracking requirements changes.

The use of wikis has also been instrumental in software engineering. Development teams use them to organize, track, and publish their work [20], but wikis have their own challenges and limitations. The good thing is that wikis can be used to share information among the stakeholders and can update them at certain level of details. But they have mainly been used to store and retrieve documents, to allocate tasks, or to keep track of what has been done. The problem domain we are trying to address does not have to do with document repositories.

In short, our literature review to date reveals that existing research on the domain has been unable to fully address the issue. We have not found any proven methodology or automated technique that could facilitate a communication and collaboration mechanism throughout the requirements engineering process by alleviating the problems with communication caused by team members being in different time zones.

4. RESEARCH METHODOLOGY

Our research question is how we can facilitate non-verbal communication on a GRE process so that dependency on oral communication can be minimized, due to presence of the teams in different time zones, which limits frequency of the synchronous interaction.

As part of the solution domain, we first examined the alleviation of the need for oral communication so that the communication challenge associated with teams in different time zones can be minimized. Second, the proposed process should facilitate a requirements validation and verification mechanism which accounts for missing requirements and caters for conflicting situations. Based on the deficiencies with the existing communication mechanisms for GSD, we propose a non-verbal communication process through requirements documentation that we argue can deal with the problems associated with a GRE process.

Moreover, in order to perform this research, we carried out a literature review. We investigated the existing communication mechanisms and their limitations of coping with requirements engineering phase of GSD. We argue that in order to improve communication among the teams that are in different geographical locations, representation of actual requirements must be simple and easy to comprehend. For that purpose, we opted for a goal-based methodology [21]. This entails illustrating requirements in the form of goals that are easy to realize and manage. In addition, we consider a real time example in which distributed teams are involved in development of a financial system. We look into the possibilities of incomplete requirements handovers that are likely to occur due to tight schedules and geographical distance, and finally we demonstrate how the proposed process can deal with those situations.

5. THE PROPOSED PROCESS

Since one of the main goals of GSD is to decrease project development costs by reducing development time, the organizations involved in GSD usually work under tight schedules to deliver business functionality. This phenomenon can result in incomplete requirements while analysis and design documents or to be developed requirements are passed on from the onsite team to the offsite team. There could be multiple reasons for this phenomenon. First, the teams could be working under a tight schedule and they are likely to rush the requirements, analysis and design phases. Second, the onsite team is collocated with customer which means that more project context is available. Therefore, it is likely that the onsite team will assume that the offsite development team understands the requirements. Third, having development teams in different geographical locations which have different regulations can also make complete requirements hard to transfer. Overall, this demands an efficient communication mechanism where teams can formally communicate and negotiate on different activities during the distributed requirements engineering phase. But this sort of communication becomes time consuming especially when teams are located in different time zones as delays in feedback and priorities become quite probable.

Figure 2a shows our proposed process.

(1) Total system requirements are represented as a goal model. The left hand side denotes requirements handed over by the onsite team (X) to the offsite team (Y) and the arrow signs denote the flow of the process.

(2) The goal model is under populated indicating that incomplete requirements were handed over by team X.
(3) Team Y identifies this and adds more requirements in the form of an extension to the goal model. This extension could give rise to conflicts which implies that at least one goal at that level is mutually exclusive and cannot be satisfied in conjunction with others. In other words, a set of related requirements at the same level cannot be implemented. The arcs labeled as Append denote changes made by team Y whereas the ones in Conflicts represent the goals which have conflicts. A conflict at the preceding level is an indication that there has been priority assigned to certain goals that Team Y believes are technically more feasible to implement (labelled as Priority). Hence Team Y prioritizes the conflicting goals.

(4) Team X provides feedback, as those changes as well as prioritizations have to be verified and validated by the onsite team who is in close proximity with the client. Team X not only has to validate those goal changes made by team Y but also has to consider the priority proposed by them in order to resolve the conflicting situation. Moreover, team X could also suggest replacing the conflicting goal with something new that could fit well into the existing requirements and could also eliminate incompatibility among the goals, as denoted by an edge which is labeled as Append in a model on the left hand side.

(5) The verified and validated goals are returned to team Y.

6. EVALUATION

In order to fully understand the applicability of the proposed process, we consider a real time case study in which a software development company X in one of the capital cities in Europe carries out GSD projects. Two teams are involved in a financial system development having n number of components. In order to deliver the complete business functionality, the main goals, those associated with sub system development, must be satisfied. With the onsite team sending requirements to the offsite team, the system requirements can be classified into two groups - functional and non-functional requirements. Correctness of requirements in the financial industry is very important. Otherwise, great financial losses can be encountered. A financial system may consist of many individual subsystems, but as part of this research, we consider the Insurance and Claims sector only.

According to a financial system’s basic functionality, a user must be facilitated to interact with the system to apply for a claim. The company should process the application and should request a financial intermediary for a payment in case the application is successfully processed. Finally the applicant gets paid by the bank. In Figure 2b we show a goal model that is based on Provide Application on Multiple Platforms. The edges in black represent the requirements handed over by the onsite team to the offsite team. The ones labeled Append represent the append made by the offsite team to be validated by the onsite team, whereas the ones labeled Conflicts denote the conflicts with certain priority levels suggested by the offsite development team. We opted for this goal in our example because this is the one that is likely to require more explanation as well as adjustment. This is due to the different geographical locations of the teams, distance, and consequently by the context in which they opt for a specific platform. Moreover, each location has its own preference and culture for using a new technology.

In Figure 2b, we first try to goal model the possible requirements for the domain Provide Application on Multiple Platforms and then try to figure out the missing requirements, using the perspective of the offsite team. It is understandable that the onsite team would drill down to a certain level of detail. For example, its detailed requirements could be that the system should be made available on different types of devices. But the offsite team will have to make sure that they can provide a consistent user interface and functionality in case the system has to go on different devices. It would require an extra bit of work to do and some of those requirements could be conflicting not only among themselves but with some of the other business requirements as well. In the figure 2b, it is noticeable that conflicts start appearing as soon as the offsite team start appending the goals. For example, different types of mobile platforms such as Android and IPhone exist. But it is not that easy to provide the same functionality on those platforms because of the underlying limitations. Moreover, the service provider must give permission to run certain types of application on the phone.

Requirements and structure of a financial message could also make it difficult to be broadcasted on multiple platforms. In addition, provision of a common user interface may encounter a tricky situation across smart phones and tablets as they demonstrate different display properties. For example, Android and IPhone use different underlying platforms and also differ in terms of their screen dimensions. This variation increases even more within the Android class as different manufacturers make Android compliant smart phones that vary in terms of their

![Figure 2. (a) The proposed non-verbal GRE communication process (b) An example scenario](image-url)
hardware dimensions. In this situation, it could be difficult to confirm with the similar business rules and program logic. Finally, it is up to the onsite team to decide on the trade-offs or alternative requirements based on the priority suggested by the offsite team. In this case, the priority might be set to Confirm to Similar Business Rules as this goal also has an impact on a higher level goal Provide on Desktop.

7. CONCLUSIONS AND FUTURE WORK
To implement a good communication mechanism for a globally distributed requirements engineering process is important. We have proposed a process to facilitate communication on validation and verification of requirements through improving the way requirements are documented. This can be used in situations that are likely to happen once requirements handovers occur between GSD teams. We have demonstrated a preliminary validation by elaborating the proposed process using a case study to demonstrate its usefulness.

This facilitates communications by representing requirements as goal models that are to be implemented by the development team. In addition, it provides more visibility into the globally distributed requirements engineering process. This will not only facilitate requirements’ comprehension but will also alleviate ambiguities associated with natural language communication. The software development team will be able to append those goal models if they find any missing links in the to be developed system requirements. Such changes can be easily validated by the other team. Another advantage of the proposed methodology is that unnecessary delays can be avoided when synchronous communication is not possible as the development team can proceed with the coding tasks for the requirements that have already been validated by the onsite team. As part of the future work, we want to automate the proposed methodology on Software as a Service (SaaS). This would make the proposed solution scalable enough to adapt for any number of communication workflows.

8. ACKNOWLEDGEMENTS
We would like to thank Professor Shinichi Honiden for his support of this project. This research has received funding by Science Foundation Ireland grant 10/CE/11855 to Lero (www.lero.ie), a Higher Education Authority grant PRTL14 to the Lero Graduate School in Software Engineering, and from the European Community’s Seventh Framework Programme FP7/2007-2013 under grant agreement 215483 (S-Cube).

9. REFERENCES