A Cloud Based Model to Facilitate Software Development Outsourcing to Globally Distributed Locations

Sajid Ibrahim Hashmi, Dr. Ita Richardson
Lero- the Irish Software Engineering Research Centre, University of Limerick, Ireland
Corresponding author name: Sajid Ibrahim Hashmi
Email: sajid.hashmi@lero.ie
Tel: +353-61-233782

This research has received funding by Science Foundation Ireland grant 10/CE/I1855 to Lero (www.lero.ie), a Higher Education Authority grant PRTLI 4 to the Lero Graduate School in Software Engineering, European Union’s FP7/2007-2013 under grant agreement 215483 (S-Cube).
A Cloud Based Model to Facilitate Software Development Outsourcing to Globally Distributed Locations

Abstract - Outsourcing is an essential part of global software development and entails software development distributed across geographical borders. More specifically, it deals with software development teams dispersed across multiple geographical locations to carry out software development activities. By means of this business model, organizations expect to benefit from enhanced corporate value through advantages such as round the clock software development, availability of skills and labour, and a reduction in overall project costs. Outsourcing software development across multiple countries is not an easy task as organizations find it difficult to identify the best strategy because communication among the teams on software development activities remains an issue. The situation gets worse when those teams are in located in different geographical locations with different time zones.

On the other hand, the advent of the cloud computing has supported organizations by bringing new concepts and opportunities resulting in benefits such as scalability, flexibility, independence, reduced cost, resource pools, and usage tracking.

In this research, we aim to make use of cloud services to address the challenges associated with software outsourcing. The scope of our work is three fold: first it will identify different types of software outsourcing models in practice today. Second, it will make a comparison among them in terms of their work practices and will investigate their usefulness to fulfil organizational needs. Third, we will propose a cloud based outsourcing model to facilitate the task by improving communication among software development teams.
Keywords

Software outsourcing, global software development (GSD), cloud services, software organizations, software development teams.
1. Introduction – Global Software Development

GSD is software development incorporating teams spread across the globe in different locations, countries, and even continents. We are motivated by the fact that conducting software projects in multiple geographical locations is likely to result in benefits such as cost reduction and reduced time-to-market (Oshri 2007), access to a larger skill pool, proximity to customer, and twenty-four hour development by following the sun (Nguyen 2008). But, at the same time, GSD brings challenges to distributed software development activities due to geographic, cultural, linguistic, and temporal distance between the project development teams.

Advancement in technology and communication mechanisms has had a positive impact on business growth as the exchange of information has become more timely, accurate and available. Because of this, business organizations are no longer reluctant to outsource software projects and to have software development operations in multiple geographical locations. They urge to make use of customized business models to maximize their benefits. In addition, from the marketing perspective, the goals of globally sourced development (Herbsleb 2001) include making use of globally distributed physical and material resources, reducing time to market, and taking advantage of marketing business opportunities. In the remainder of this introduction section, we highlight the context of this research, the research question, the objective of the research, and the research methodology. Also, we present a summary of the cloud computing, challenges faced by GSD, and our motive for using the cloud paradigm to support GSD.

In order to meet the different challenges posed by GSD, we suggest making use of the cloud computing paradigm and illustrate that it has potential to enhance the usefulness of GSD. We argue that different types of geographic and cultural issues can be addressed by making use of
different cloud computing realizations such as PaaS (Platform as a Service), IaaS (Infrastructure as a Service), and SaaS (Software as a Service). Since data in the cloud is accessed through services (Pires 2010), we study its usefulness in the light of service. Furthermore, we argue that the cloud can facilitate GSD both as a process and as a product. The former one could have implications for the GSD business model in which service providers are organizations and services are parts of a GSD process, for example, requirements, design, coding, and testing. Cloud as a product is developed, run, and distributed globally. The idea is to identify different types and domains of GSD issues and investigate the potential of the cloud to address those.

In the global environment, outsourcing software development projects to low cost economies is becoming increasingly popular, especially as there is the expectation that companies who embark on GSD business model will gain and maintain economic advantage through numerous technical and commercial factors (Herbsleb 2007). This increase in GSD implementation is supported by the availability and accessibility of communication tools as they enhance the options to use a remotely located workforce. The business models in low cost countries have provided capable and willing workers who undertake outsourced and offshore software development. This in turn provides cost reduction in software development projects (Carmel 2005). However, outsourcing software development to organizations at various outsourcing destinations is not an easy and straightforward task (Carmel 1999) and organizations very often face difficulties due to global distance and the involvement of the development teams which are geographically distributed.
1.1 Issues with GSD

With the emergence of technologies in a world which has become increasingly globalized, the relationship between culture and management of remote work has become an unavoidable issue which needs to be addressed. Because of distance among the software development teams, GSD encounters certain challenges in terms of collaboration, communication, coordination, culture, management, organizational, outsourcing, development process, development teams, and tools. Global distance comprises of four elements: geographic, cultural, linguistic, and temporal distance. Geographic distance occurs as the teams are dispersed across countries. Cultural distance occurs due to teams being made up of members from different cultures, and the additional expectation that each member will understand and support each other’s culture. When team members speak in different languages, there needs to be one chosen language for work purposes, and as this is everyone’s first language, linguistic distance occurs. As teams are geographically dispersed, there is the additional difficulty of temporal distance – members working across different time zones. Each of these differences individually causes problems within GSD teams, and the culmination of these differences into global distance can and do impede global software development projects. Thus, the management of globally outsourced software development has been accepted as a difficult and complex task. In the Table 3, these four types of GSD challenges are addressed in the light of cloud services.

The collocated software development teams can effectively use communication and coordination mechanisms. There is definitely less need for communication when activities are carried out independently, but the outsourcing model we are considering, requires active communication as teams from different organizations would interact with each other. Moreover, the distributed nature of the
processes makes activities dependent on the geographically distributed sites. Also, the communication needs may also differ depending on the type of the to be developed software application and the size of teams across the globe.

Communication is a process by which information is exchanged between individuals through a common system of symbols, signs, behavior or tools. It is the key to team based success (Battin 2001). Communication among the teams may have multiple facets; i.e. it can be formal, informal, synchronous and asynchronous. Moreover, it can be on technical as well as nontechnical. As stated earlier, GSD involves several issues and one of them is communication. It can be split up into synchronous and asynchronous communication. There are synchronous and asynchronous ways of information exchange and an effective communication mechanism is inevitable to be in place in order to facilitate information exchange. An obvious obstacle to communicating among the development sites is the inability to share the same environment and to see what is going on at the other site. These issues are even reinforced in case of a different time zone because of the selective availability of the remote contact personnel (Palacio 2009). When teams are located in the same time zone, synchronous communication using the existing communication mechanisms may alleviate the distance among them. The situation is not that simple when teams are in different time zones as the communication mechanisms are minimized. Another reason communication is inevitable because it is a mediating factor effecting coordination and control (Al-Ani 2008).

Outsourcing software development is a complex task and must be dealt vigilantly before the implementation. The idea is to reduce time and the associated costs and make use of a global skillset. (Olsson 2008) Has defined multiple facets of outsourcing, i.e. in shore outsourcing and the offshore outsourcing. When the client and the vendor companies are located in the same countries, it is an inshore outsourcing. On the other hand, when they are in different countries, it is an offshore outsourcing. In this research study, the primary focus is on offshore outsourcing
because we feel challenges are apparent in that sort of development model. Whereas when they are in different countries, often a similar term *offshoring* is used. Innovation in outsourcing has been a desired phenomenon. Different frameworks as well as strategies have been proposed for it so far (Oshri 2011).

### 1.2 Issues with Distributed Requirements Engineering

Distributed requirements engineering has been a problem area (Calefato 2012). A typical requirements engineering process 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 distributed requirements engineering process, we consider a multisite GSD scenario in which 3 sites are involved. One of the teams is onsite with the client with team members involved in different roles across each team. These teams make use of the internet as an underlying communication mechanism on top of which they use different means like emails, instant messaging, and video conferencing. The business and technical consultants onsite can communicate using standard communication mechanisms but problems arise when they have to communicate with their colleagues offsite who are in different time zones. Traditional communication mechanisms cannot ensure timely verification and validation of the assumptions made by the development team.
In Table 1, we list the tools that have been identified by (Cheng 2004) and are used by GSD teams. In addition, we present their scope for communication on distributed requirements engineering activities and their limitations with relation to time difference among the teams.

**TABLE 1. EXISTING COMMUNICATION MECHANISMS IN GSD**

<table>
<thead>
<tr>
<th>Tools</th>
<th>Usefulness</th>
<th>Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instant Messaging 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 &amp; validations</td>
</tr>
<tr>
<td>Skype</td>
<td>Complements IM environments with oral and video facility.</td>
<td>Usability is limited in different time zones. Use of natural language could also be involved</td>
</tr>
<tr>
<td>Email</td>
<td>Works well for making formal queries or notifications. Documents and files can also be exchanged</td>
<td>Although it has been the most conventional method but does not guarantee a timely communication</td>
</tr>
<tr>
<td>Video</td>
<td>Feasible in case</td>
<td>Availability of team members could be an issue for...</td>
</tr>
</tbody>
</table>
conferencing sites are equipped with the technical facility different time zones. Use of natural language as well as some technical overheads are involved.

<table>
<thead>
<tr>
<th>Wikis</th>
<th>Can be used to assign tasks. Helps to keep everyone updated about the activities</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mainly used as a document repository. There is hardly any mechanism for verification or validation</td>
</tr>
</tbody>
</table>

In addition, there is some research and open source tools but they do not solve the dilemma of the distributed requirements engineering process. 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 requirements engineers. This can result in verification and validation of requirements being delayed.

### 1.3 Research Question

The question this research aims to answer is:

What are the issues involved with software development outsourcing on the whole?

What challenges do the organizations face in terms of communication, specifically on global requirements engineering phase?
Is it possible to make use of the cloud to address the issue and facilitate the teams to cope with the distance issue?

The process used to design and build a software application is called the software process (Humphrey 1989). In other words, we can say that software process is the set of activities to develop a software system. These activities may be adhoc or formally documented. This process may include technological and organizational aspect of software development. A software process model is the representation of a software process (Carmel 2005). In this research we propose a prescriptive model to propose the better management of requirements using the cloud in global software outsourcing domain. An effective process model should effectively communicate the set of activities among all the stakeholders involved in the process. Additionally, as process model must evolve to meet the actual business requirements, it must be evolvable to accommodate any changes for good and should be manageable to track of any changes and cost overruns.

Since cloud computing is the key service delivery platform in the field of services, as a process model, it could allow resource sharing not only for infrastructure and application resources, but also for software resources and business processes. These advantages are likely to support different disciplines, for example, Infrastructure as a Service (IaaS) could help provide different GSD teams with resources such as computing power or storage provisioning to store project related data. Software resources may consist of middleware and development resources like application systems, database servers, and operating systems. The advantage of using first two types of resources as a service is that they are never wasted after the project is over - instead, they can be unsubscribed. Application resources could assist in providing SaaS with necessary interfaces that could facilitate collaboration and sharing of
information among the teams.

In this research, we aim to use the term GSD with an outsourcing perspective. In some of the existing research literature, terms like Global Software Development and Global Software Engineering have been used interchangeably, but this should not make a difference because both development as well as engineering in a global setting may involve outsourcing.

1.4 Objective of the Research

The purpose of this research is to realize the requirements related issues in GSD and propose a set of practices to resolve those problems. The ultimate goal is to develop a cloud based model to facilitate communication among client and vendor teams. The model incorporates strategies to deal with a global requirements engineering process and aims to provide a cloud based solution to cope with the communication challenge of GSD outsourcing. The incorporated practices are taken from research literature, books and established quality assurance models. The model aims to provide process and communication related guidelines to the apart teams.

This work proposes the development of Global Software Development (GSD) processes using the cloud computing paradigm, based on our understanding of current GSD and Cloud Computing from literature, and our overall aim is to propose the re-construction and improvement of the GSD process. This is done through the use of cloud computing. We discuss how the GSD process can be aligned and improved using cloud services. We argue that, for example, some existing tools support GSD communication processes. However, we question
whether these can be streamlined and re-organized by defining how exactly GSD can work better by making use of a service based environment.

Initially, we identify communication related problem areas in GSD and subsequently, propose the support of GSD development activities through services.

The emphasis is on facilitating communication and collaboration activities among GSD teams by structuring those activities. Our rationale is that we can parallel the GSD situation with manufacturing supply-chain management where systems used are composed of ready-to use service-oriented systems. The reason services are widely adopted in industry is because they can be integrated seamlessly. This results in benefits to industry such as increased return on investment and reduced information technology costs (Watson 1994). We argue that processes to support GSD activities could be defined in the form of cloud service based tasks and that what we need are heterogeneous services which could support different process activities. Moreover, output from one service could be taken as input to the next, in cases, where those services supported interrelated activities. In this article, terms like services and the cloud services have been used interchangeably as different representations of the cloud are being accessed using services.

1.5 Research Methodology

In order to conduct this research, our literature review studied characteristics of services and the cloud. We also identified challenges faced by GSD. Following this step, we held a workshop, each of the participants had research and/or industrial expertise in GSD and/or services. During this workshop, through interactive discussion and brainstorming, we developed the concepts
presented in this paper. To do this, we summarized the GSD challenges and requirements and investigated the potential of cloud services to address these. We are embarking on further research to understand whether these indeed can be of value to both the industrial and research communities.

2. Cloud Computing

Cloud computing is an internet based computing paradigm in which shared resources like software, hardware, and information are provided to the subscribers on demand (Turner 2003). NIST (Badger 2011) defines cloud computing as a model for enabling convenient and on demand network access to shared computing resources that can be managed and provided rapidly with minimal effort. The aim is to construct a low cost computing system by using certain entities without compromising on computing capabilities. Depending on the type of shared resources, the cloud paradigm can have different implementations like IaaS (Infrastructure as a Service), PaaS (Platform as a Service), and SaaS (Software as a Service), to dispense computing capacity to end users. Infrastructure as a Service (IaaS) includes the delivery of hardware such as processors and storage as a service, e.g., Amazon Elastic Cloud (EC2) and Simple Storage Service (S3). In other words we can say that it delivers a platform utilization environment as a service. Instead of physically purchasing hardware and software infrastructure, clients buy such resources as a fully outsourced service.

In addition to the infrastructure, Platform as a Service (PasS) occurs when a software platform is provided on which systems can be run. This includes the delivery of programming platforms and tools as a service. This kind of cloud computing provides a development environment and the
infrastructure provider’s equipment can be used to develop programs which are delivered to end users through internet and servers.

Software as a Service (SaaS) occurs when applications are delivered as services using IaaS and PaaS. This implementation of the cloud focuses on separating the ownership and possession of software from its use (Turner 2003). It is based on the idea that software functionality could be provided as set of distributed services that could be configured and bound at delivery time, to avoid the current limitations with software use, deployment, and evolution (Turner 2003). Since cloud computing stimulates the provision of online services via the World Wide Web, software can be hosted on web servers as services (Turner 2003). Thus, the advent of SaaS within the cloud computing paradigm has created new opportunities for organizations to communicate and coordinate among themselves. In Table 2, we list down some supporting characteristics of cloud computing that can be useful for GSD.

<table>
<thead>
<tr>
<th>Virtualization</th>
<th>Courtesy of this privilege, cloud providers can enhance their infrastructure to accommodate in case there is growing demand for services. Usually, a combination of hardware and software are used on the provider side to meet with the scaling requirements.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reduced Cost</td>
<td>Costs in the cloud do not include server side infrastructure and equipment costs. Moreover, pay as you go model ensures that subscribers are bound to pay for only those resources which they use. In short, the distribution costs of software are reduced.</td>
</tr>
</tbody>
</table>
### Scalability
On-demand provision of application software provides scalability, which results in greater efficiency. Whereas cloud-based application development platforms provide with high level of scalability thus making the developed application to cope with the fluctuation demands.

### Infrastructure
Providers’ applications are run on a cloud infrastructure from where a consumer can access those. Similarly, consumer-modified information or application can be deployed on the same infrastructure as well. The privilege is that the consumer does not have to deal with the underlying infrastructure.

### Performance
The cloud paradigm can support various levels of performance requirements like service scaling, response time, and availability of the application based on the needs of the consumers. In addition indirect performance measures may also be achieved by eliminating the overheads involved with installation procedures and reduction in unnecessary reduction among the applications running on the cloud.

### Multi Tenancy Support
Public clouds are elastic in nature as their consumers are not limited. More importantly, consumers’ workloads are isolated to provide privacy. However, the number of consumers can be restricted by opting out a specific deployment model.

#### 2.1 Motive for Using the Cloud for Supporting GSD

One of the missions (Behrendt 2011) of the cloud architecture is to provide services to customers by not only managing them but optimizing them by taking into consideration economies of scale.
The cloud model is composed of three service models (IaaS, PaaS, and SaaS), five essential characteristics, and four deployment models (Behrendt 2011). The cloud deployment models—Private, Community, Public, and Hybrid—define the scope of the cloud solution. The cloud models are discussed in terms of creation and provision of services which means that it supports services. Since services run a mechanism for development and management of distributed dynamic systems and it evolved from the distributed component based approach, we argue that it has potential to cater the challenges of GSD where a project is developed across different geographical locations. Our thesis is that GSD challenges can be overcome through the support of services. This will contribute increased interoperability, diversification, and business and technology alignment. Moreover, the vision behind the cloud paradigm is to set up common goals and objectives to improve the collective effectiveness of the enterprises participating in globally distributed projects. Since software processes are software too (Osterweil 1997), we argue that the cloud has potential to reinforce GSD as a process. Initially, we considered the use of standard procedures to meet the communication and collaboration challenges posed by GSD. But, since organizations have to interact dynamically in global environments, these standard procedures cannot scale up to support dynamism. Moreover, the ideology posed by both services and GSD is somehow similar, for example, coordination, transaction, context, execution monitoring, and infrastructure. In addition, services are one of the main technical foundations of the cloud (Dikaiakos 2009).

For GSD, the use of collaboration tools among teams is not new. Existing research has already proposed further work in this regard (Cheng 2004). We adopt the idea of SaaS for GSD to make use of properties of both cloud and SaaS, such as reusability, reliability, extendibility and
inexpensiveness. Teams with frequent communications among their members are likely to collaborate better. Thus, this frequent communication is important to make full use of GSD advantages, e.g. improved productivity, reduced time to the market, and reduced cost. However, oral communication is prone to confusion and misunderstanding. One way could be to minimize the need for communication but such strategy would emphasize on the involvement of more dedicated personnel from each development site which could not be feasible either. At the same time it is important for the communication media to be formal, flexible, and evolvable to ensure the collaboration mechanisms work effectively.

GSD teams also need to collaborate effectively and the attributes of the cloud paradigm, especially SaaS, can be used to facilitate efficient collaboration between geographically distributed teams during software development phases such as requirements, design, coding, and testing. The characteristics and the architecture of the cloud model itself has the potential to fulfill the GSD task requirements. For example, cloud deployment models allow certain trusted partners (which could be GSD team members) to share resources among themselves. Service models may not only provide access to collaboration and productivity tools but also allow network access to computing resources, and the “use as you go” feature is likely to reduce the overall project costs across multiple development sites as computing resources and infrastructure is not required upfront.

We investigate the impacts of the aforementioned collaboration challenges and suggest the likelihood of using the cloud to address them. We expect to achieve efficiency in communication through using the cloud in different implementations. The essence of using this paradigm to
facilitate GSD is that instead of acquiring and owning the software and project data, GSD team members can access and subscribe to some of the software at a time (according to the need) in the form of services. In addition, we will be able to take advantage of the service characteristics [16] like loose coupling, service composition and negotiation to facilitate a similar level of development practices across multiple sites. Moreover, the service provider and user are important to the technical and economic changes made possible by cloud computing. In our model, this concept of provider and consumer is similar to the cloud paradigm.

**TABLE 3. GSD CHALLENGES POTENTIALLY FACILITATED BY THE USE OF SERVICES**

<table>
<thead>
<tr>
<th>Collaboration Challenges</th>
<th>Issues</th>
<th>Negative Impact on Software Project</th>
<th>Facilitating GSD Using Services (SOA/Cloud)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geographic</td>
<td>Distance, Time, Knowledge transfer, Tools</td>
<td>Communication gaps, Project Delays, Ambiguity on technical aspects, Unequal quality levels across the software development sites</td>
<td>Dynamic binding, runtime adaptation, and timely availability of required services could help dealing with geographic issues. Also, availability of SaaS could diminish installation overheads at each development location.</td>
</tr>
<tr>
<td>Cultural</td>
<td>Unequal distribution of work</td>
<td>Increase in cost</td>
<td>Service could maintain a fair distribution of work between the teams. Only a specific person will be responsible for the task assigned to thus skill management would be easier too.</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------------------------</td>
<td>------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>Lack of Trust, Fear</td>
<td>Poor skill management</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reporting problems</td>
<td></td>
</tr>
<tr>
<td>Linguistic</td>
<td>Frequency of communication</td>
<td>Loss in project quality</td>
<td>Run time evolution of services can meet with the linguistic issues. Also, isolation of each task and related information as a service can ensure right level of knowledge transfer.</td>
</tr>
<tr>
<td></td>
<td>Knowledge transfer</td>
<td>Invisibility on project development</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ineffective project management</td>
<td></td>
</tr>
<tr>
<td>Temporal</td>
<td>Lack of Motivation</td>
<td>Loss in project quality</td>
<td>The cloud service models imply that the data resides on a centralized location where inventory of services is maintained. Services maintain a registry where all of them are</td>
</tr>
<tr>
<td></td>
<td>Less visibility</td>
<td>Poor management of configuration</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Risk</td>
<td>Chances of project artifact loss</td>
<td></td>
</tr>
</tbody>
</table>
stored. This attribute could be used to store and retrieve configurations.

3. Related Work

Our literature review focuses on interrelated topics: means of communication in GSD, software engineering processes involved in GSD with emphasis on requirements engineering, 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. Their results confirm the problems related to communication in distributed requirements engineering but they do not solutions to those issues that cover technical as well as non-technical facets. Moreover, the existing work on the domain mainly focuses on early activities of the phase like requirements planning and elicitation. We have not seen the facilitation of GSD team members who deal with communication challenges associated with requirement handover.

Some measures to enhance collaboration; for example, Email, IM (instant messaging), and screen sharing have been reported. 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. And, while this practice could be helpful at coding stage of the development where a developer could ask a colleague to help him
figure out any problems with his code, we have not established its usefulness during requirements handover.

Use of wikis has also been instrumental in software engineering. Development teams use them to organize, track, and publish their work, but 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 and survey of existing industrial practices reveal that existing research on the domain has been unable to fully address the issue. There hardly exists any proven methodology or automated techniques that could facilitate a communication mechanism throughout the requirements engineering process by alleviating the problems with communication caused by team members being in different time zones.

4. The Proposed Process Model

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, consequently, 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 to assume that the offsite development team understands the requirements. Third, with development teams in different geographical locations who have to cope with 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 gets time consuming especially when teams are located in different time zones as delays become quite probable in getting feedback or priorities from one team to another.

![Figure 1. The Proposed Process Model](image-url)
Figure 1 shows our proposed process. The process workflow is built on top of a SaaS (Software as a Service). We propose this rather than a simple web based architecture as its use will ensure that a collaborative as you go space is provided to the software development team members, the scalability feature ensures that it can handle variable number of communication work flows and it can also provide as you go storage of requirements artifacts. We propose making use of goal modeling to model the requirements handed over to team B by the onsite team A. Goal modeling is quite useful in understanding requirements and to identify the missing links. Since requirements are transferred from onsite to offsite,

Goal model representation would allow the teams to identify the missing as well as the conflicting requirements. The conflicts are likely to be established once the offsite team input to the system requirements. Therefore, we propose that requirements should be represented in the form of goals before the handover is made. It will allow that team more visibility into the system and they can append the goal models with the requirements which went missing earlier. The onsite team has to verify and validate the additions before the offsite team could proceed with the implementation.

4.1 Case Study

4.1.1 Problem Description

In order to fully understand the applicability of the proposed process, we consider a real time situation in which a software development company X in one of the capital cities in Europe carries out GSD projects. In order to explain the problem description, we consider a situation where 2 teams are involved in a financial system development having n number of components. In order to deliver the complete business functionality, it is inevitable to satisfy the main goals
which are the goals associated with sub system development. With the onsite team sending on requirements, the system requirements can be classified into two groups, functional and non-functional requirements. Correctness of requirements in the financial industry is very important as the industry involves huge sum of money transfers, otherwise great financial losses can be encountered. A financial system may consist of many individual subsystems, but as part of this research, we consider *Insurance and Claims* sector only.

A financial system like Insurance & Claims would mainly consist of three sub systems: Management of personal details, Document Management, Management of Payments. In other words, we can say that those three high level goals should be satisfied in order to achieve the overall system functionality goal.

### 4.1.1.2 Example Scenario

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 his application and should request a financial intermediary for a payment in case the application is successfully processed, and finally the applicant gets paid by the bank. In Figure 2 we show a goal model that is based on *Make Payments* requirement. The edges in black represent the requirements handed over by the onsite
Figure 2. A Goal Model Based Exchange of the *Make Payments* Requirement Among the Teams

team to the one offsite, the ones labeled as $A$ represent the appends made by the offsite team to be validated by team A, whereas the ones labeled as $C$ denote the conflicts with certain priority levels suggested by the offsite development team.
We opted for this goal in our example because they are the ones that are likely to require more explanation as well as adjustment because of different geographical locations of the teams, distance, and consequently by the context to which the payment may refer. Each location has got its own preference and culture of using a new technology. Moreover, electronic payments have got different standards to cover different conditions and situations.

In Figure 2, we first try to goal model the possible requirements for the domain Make Payment and then try to figure out the missing requirements, using the perspective of the offsite team. Payment function constitutes an important part of any financial system so it is important to identify any incomplete requirements and then validate those for completeness and conflicts. In Figure 2, the offsite team identifies the potential incomplete requirements that are labeled as A and could further lead to conflicting situations. The payment component must ensure that the system should distinguish between the two different types of customers - employees and corporates. Then, the system must be able to support manual as well as electronic fund transfers. An important part of electronic fund transfer is to capture the customer location in case they are relocated. Moreover, different standards exist for financial messages; for example SWIFT and MT100 messages are global standards whereas SEPA, PSD, and ISO 20022 are European standards. Also, there needs to be checks for the case where payment has been split up into more than one installment that have to be paid by different financial institutions in different parts of the world. Different customer locations for a financial transaction would bring in regulation issues and an onsite team is likely to miss out this following if not alerted to its existence. In this case although the goal model in Figure 4b identifies a conflicting situation but it is unlikely for the team A to make a trade-off between the two as both types of goal are inevitable. So the best
possible way would be to improve those conditions and scenarios that result in this type of a conflict.

5. Conclusions & Future Work

To implement a good communication mechanism for globally distributed requirements engineering process is important. Based on the problem description and the characteristics of SaaS (Software as a Service), we have proposed a communication process that is built on top of the SaaS cloud to facilitate communication on validation and verification of requirements. This can be used in situations that are likely to happen once requirements handovers are made among the GSD teams. We have elaborated the proposed process using a case study to demonstrate situations where GSD problems can occur. We have demonstrated the applicability and usefulness of the proposed process by means of a real time example scenario.

The proposed process 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. The provision of this approach on SaaS cloud means that the system is scalable to adapt for any number of communication workflows. The requirements we have presented as
goal models are at a high level of abstraction. As part of our future work, we plan to drill down to more fine grained requirements which are at lower abstraction levels. This will enhance the usefulness of the process.
6. References


