
Muhammad Ali Babar
Mansoor Zahedi
Abstract

Context: Global Software Development (GSD) paradigm has gained significant popularity and widespread adoption. There has been a large amount of research efforts dedicated to gain understanding and knowledge about the GSD challenges and benefits in order to devise and evaluate appropriate solutions to address the challenges. It is equally important to provide period review of the published literature on GSD challenges, benefits, and proposed solution to inform practitioners and researchers about the state-of-the-art in this area.

Objectives: This research has been motivated by the need of providing a state-of-the-art review of the GSD research literature published in the main venue of Global Software Engineering in order to identify the main research trends and gaps that needs to be filled by future research. We were also interested placing the findings of our review with respect to a practice-driven GSD research agenda.

Method: We used structured literature review methodology for which we decided to select and review the recently published research papers (i.e., 2007 - 2011) from the International Conference in Global Software Engineering (ICGSE). We used a framework for organizing GSD research challenges and threats and a practice-driven research agenda for extracting and organizing the data from the reviewed papers. We used theoretical reasoning for classifying the reviewed papers under different categories, which were mainly based on the framework, a decision that also enabled us to propose certain extensions to the framework.

Results: Given the resources and time available, we decided to review 50 out of the 97 research papers that had been published in the proceedings of ICGSE 2007-2011 conference over the last five years. This review has helped us to produce interesting findings about the research literature published over the last five years as we have observed that there is an increasing trend of proposing solutions rather than just mentioning GSD problems. Moreover, we also have found that a large majority of the reviewed papers belonged to the software development governance and its sub-categories and many of the papers have focused on the human-aspects of GSD rather than technical aspects.
**Conclusions:** GSD research continues attracting significant attention, however, the most of the published research does not appear to have very tight focus rather its spread quite thin without taking much influence from the previously published research agenda. There is an increasing tendency of providing solutions to the identified problems rather than just reporting problems. However, many of the proposed solutions have not been rigorously assessed. Whilst the output from the GSD research is increasing, there are significant gaps that needs to be filled, for example, studying the challenges and proposing solutions specific to different phases of software development such software design and testing. This review has also enabled us to propose an extension to a previously reported framework for organizing and classifying the GSD challenges, benefits, and solutions. We expect this review to provide useful insights that can be used to draw interesting and challenging research questions to direct the future research in this area.

**Keywords:** Global Software Development, Global Software Engineering, Knowledge management, information acquisition and sharing
# Table of Contents

1 Introduction .......................................................................................................................... 1

2 Characterising Global Software Engineering ................................................................. 4
  2.1 Different Forms and Scenarios of GSE .......................................................................... 5
  2.2 Frameworks for Reviewing the Literature ...................................................................... 7
  2.3 Research Motivation and Objective .............................................................................. 13

3 Research Approach Used .................................................................................................. 14
  3.1 Research Questions ...................................................................................................... 14
  3.2 Selection of the Reviewed Papers .................................................................................. 14

4 Review and Analysis ........................................................................................................... 16
  4.1 Communication in GSD ............................................................................................... 18
  4.2 Coordination in GSD ................................................................................................... 24
  4.3 Collaboration in GSD .................................................................................................... 26
  4.4 Software Development Governance in GSD ............................................................... 30
    4.4.1 GSD Models and Processes .................................................................................. 31
    4.4.2 Project Management in GSD ............................................................................... 39
    4.4.3 General GSD Practices ....................................................................................... 44
    4.4.4 Knowledge Management (KM) in GSD ............................................................... 50

5 Discussion and Conclusions .............................................................................................. 55

Acknowledgement ................................................................................................................. 69

References ............................................................................................................................. 69
List of Figures and Tables

Figure 1: Different models of Global Software Engineering (GSE) ................................................................. 4
Figure 2: Different kinds of engagements in GSE (Source: Gallivan and Oh, 1999) ............................. 5
Figure 3: Tasks allocation to different sites as a source of complexity ......................................................... 7
Table 1: A framework to classify the benefits and threats of Distributed Development (DD)
presented by Ågerfalk and Fitzgerald (Reproduced from [2]) ................................................................. 8
Table 2: Annual categorization of the included papers out of first iteration ........................................... 56
Table 3: Distribution of primary studies based on the themes ................................................................. 58
Figure 4: Distribution of primary studies based on themes ................................................................. 59
1 Introduction

Software development paradigms have dramatically changed over the last two decades during which Global Software Engineering (GSE) (or also called Global Software Development (GSD)) has emerged as one of the most popular models of developing software intensive systems. All types and sizes of businesses are increasingly adopting a geographically distributed software development paradigm, which characterises software development as a multi-site, multi-culture, and global undertaking [52-53]. Apart from the desire to achieve the promised benefits such as extending working hours and/or teams with the required set of skills, taking advantage of low cost destinations, and being closer to a significant market, GSE is also gaining momentum as a result of mergers, acquisitions, alliances, and several kinds of market demands [21-22, 41].

As a result of the increasing popularity of GSE, organizations have distributed their software development across multiple sites, which may be within the same country or even different countries, thus involving themselves in GSE/GSD or off-shoring, e.g., IBM, Bosch, and Siemens [18]. GSE can be characterised by mainly two situations: (1) the software is developed globally by an organization for its own use, or to sell, or to incorporate it into a device that the company sells, e.g., Siemens; (2) a company, the client, outsources the software development to a vendor who then develops the required software globally for a client. GSE projects are often large-scale, and global development leads to significantly increased complexity for project teams, who may have to face several kinds of new challenges characterized by the need of new and novel coordination, collaboration, and communication mechanisms to support different software development activities undertaken by team members located at multiple sites, each with its own time zone, and with different cultural backgrounds [23, 38, 57].

The temporal, geographical, cultural, and linguistic distances, that characterize GSE projects, usually result in several kinds of communication, coordination, and collaboration challenges that may not be encountered by the teams developing software in more traditional way, e.g., in a co-located arrangement. Other kinds of potential challenges of GSE can be delayed feedback, restricted communication, less shared project awareness, difficulty of synchronous communication, inconsistent development and build environments, and lack of trust and confidence between sites. Several studies
have also reported that GSE teams face several kinds of communication challenges such as difficulty of expressions, lack of proficiency in the language of communication, frustration from incomprehensible accents, and delays or bad video bandwidth. These kinds of challenges can exacerbate the feelings of cultural, temporal, linguistic, and geographical distances [42]. That is why it is clear that compared with collocated projects, GSE projects are more likely to be unsuccessful, because of the abovementioned challenges which can have negative impact on communication, coordination, collaboration, and knowledge exchange between distributed software development teams [39]. That is why a relatively large number of outsourced development projects fail [73].

Such a high failure rate of GSE projects can be attributed to several kinds of risks and challenges, for example, those related to client and vendor organizational environments, client and vendor teams and their cultures, unfamiliarity of the development teams with the users’ cultures, and, in addition there may be cross border legal issues. When a software project is developed in multiple countries, the project team must also address several kinds of execution risks, including those related to project distribution, time zone differences, communication, coordination, collaboration, and control issues, project contextual issues, and infrastructure [58]. Hence, it is increasingly becoming apparent that GSE paradigm not only provides several kinds of new opportunities for saving cost and growth through process and product innovation but also presents risks of not achieving desired objectives, including quality, cost, and schedule [31, 47, 102]. Apart from abovementioned challenges, other challenges of GSE projects include requirements elicitation, and specification, functional testing, lack of business knowledge and knowledge of local laws and regulations, non-English communication, client-specific knowledge of business processes, and specific knowledge of legacy systems (for maintenance projects) [20].

From the above discussion, it is clear that the literature on the potential challenges of GSE has grown quite rapidly over the last ten years or so. For example, a fundamental view is that communication across dispersed sites is a major source of problems in GSE [52]. In order to help organize and analyze the GSE challenges, Ågerfalk and Fitzgerald [2] have provided a high level classification of opportunities and threats of Distributed
Development (DD), a term they used for GSD at that time. Their framework had 3X3 matrix in which they presented the opportunities and threats related to communication, coordination and control caused by temporal, geographical and socio-cultural distances that characterize GSE. Whist the literature on GSE challenges and problems has been growing quite fast, there has been slow but steady growth of the literature on reporting the potential solutions to the known GSE challenges. Researchers have also identified very specific areas where GSE researchers and practitioners need to focus in order to devise the solutions [31, 96]. The efforts aimed at devising appropriate solutions and strategies for addressing the GSE challenges has resulted in huge amount of literature which is increasing every year as there is hardly any venue related to different aspects of GSE which is not attracting papers on GSE related topics.

We have decided to review and analyze the literature on GSE from the most relevant and mainstream venue for GSE research and practice, the International Conference on Global Software Engineering Conference (ICGSE)\(^1\), over the last five years. This effort has been motivated by the need of systematically reviewing the literature and reporting the main trends of the challenges and solutions being reported and the areas which appear to be worth future research efforts by researchers and practitioners.

This research has been organized in five main sections. In the following section, we characterize GSE for the reported research and the key motivators of this work. We discuss the research approach followed for conducting the reported research in Section 3. The review and analysis of the literature has been presented in Section 4. We finish this reported by drawing some conclusions and identifying the areas of future research directions in Section 5.

\(^1\) iegse.org
2 Characterising Global Software Engineering

This section describes different characteristic of GSE and how those characteristics may result in different challenges and benefits of GSE that have been used for analyzing literature reported in this report. Moreover, we mention and elaborate the key motivators of the work that aimed at reviewing the recent literature from the most well known venue for reporting research on any aspects of GSE, the International Conference on Global Software Engineering (GSE) over the last five years, 2007 – 2011. The reporting of this section will start with a discussion on different formats of GSE and how this kind of arrangement can get complicated as a result several kinds of contextual factors. After that we discuss different elements of a framework aimed at classifying the challenges and benefits of distributed software development and propose a few more elements that need to be included in the framework in order to extend the framework. We also briefly explain different categories of GSD research agenda presented in 2006 as we have used these categories to draw certain conclusions in terms of the impact of the research agenda paper on the directions of the research over the last five years in this area. Then we conclude this section with a brief description of the key motivators of this research.

![Figure 1: Different models of Global Software Engineering (GSE)](image)
2.1 Different Forms and Scenarios of GSE

It has been stated in this report and several other papers that organizations choose to get engaged in GSE for several reasons such as extending working hours and/or teams with the required set of skills, taking advantage of low cost destinations, being closer to a significant market, as a result of merger or acquisition, forming new alliances or attempting to meet several kinds of market demands [21-22, 41]. Whilst Global Software Engineering (GSE) (or Global Software Development (GSD)) has become a well-known name for getting software developed by globally distributed teams within (i.e., off-shoring) or outside (i.e., outsourcing) an organization, there are different formats and models of software development that share several characteristics with GSE; for example, Open Source Software Development (OOSD) or Inner Source Software Development (ISSD). If we consider the outsourcing type of GSE, it may involve offshore outsourcing or near-shore outsourcing. Then there are situations where an organization may enter into a partnership with its offshore vendor, buy out an offshore vendor, or set up its own offshore development centre. Each of these types of GSE has some common and some unique challenges, which may require similar or completely different solutions to address.

![Figure 2: Different kinds of engagements in GSE (Source: Gallivan and Oh, 1999)](image_url)
Figure 1 shows different models of GSE. Different kinds of organizations depending upon their size and contextual factors may choose one or more kinds of GSE. For example, an organization may decide to be engaged in offshore outsourcing for a particular kind of component for which it does not have in-house expertise and also has its own software development team globally distributed in order to be close to the key markets for customization and maintenance purposes.

There can be different kinds of arrangements in GSE. Figure 2 shows some of the well-known types of global software development sourcing. Each of these kinds of arrangements may have different kinds of complexities, which need to be managed in different manners. For example, an apparently simplistic scenario will be the one in which only one client and one vendor are involved (i.e., top left cell in Figure 2) and the most complex scenario will be the one in which many clients and many vendors are involved (i.e., bottom right cell in Figure 2). If the most complex scenario involves vendors and clients located in different countries and continents, that kind of GSE becomes much more complex and challenging as multiple cultures, languages, time zones, and legal frameworks are involved. Each combination of these factors characterizes different kinds of challenges to the GSE teams in general and GSE project managers in particular. Moreover, most of the GSE projects are large-scale, which may necessitate the need of new and novel coordination, collaboration, and communication mechanisms to support different software development activities undertaken by team members located at multiple sites, each with its own time zone, and with different cultural backgrounds [23, 38, 57]. One of the GSE arrangement scenarios that are increasingly becoming quite common has been shown in Figure 3, which shows that different tasks are assigned to different sites located in different countries, which have different languages, cultures, and time zones.
Figure 3: Tasks allocation to different sites as a source of complexity

Figure 3 can also help appreciate that whilst distributed software development (i.e., a form of GSE) may provide different types of opportunities (e.g., follow the Sun strategy or availing specialized skill set), it can also increase the risk of not achieving the desired objectives of a project, including quality, cost, and schedule [56, 101]. Literature on GSE is awash with the enumeration and descriptions of GSE challenges including requirements elicitation, and specification, functional testing, lack of business knowledge and knowledge of local laws and regulations, non-English communication, client-specific knowledge of business processes, and specific knowledge of legacy systems (for maintenance projects) [19, 40]. Other kinds of potential challenges of GSD can be delayed feedback, restricted communication, less shared project awareness, difficulty of synchronous communication, inconsistent development and build environments, and lack of trust and confidence between sites.

2.2 Frameworks for Reviewing the Literature

By now it should have become very clear that if the technical and/or socio-technical challenges of GSE projects are not fully understood and sufficiently addressed, there is highly likelihood that a GSE project will fail. It has been reported that a relatively large number of outsourced (i.e., a kind of GSE) development projects fail [73]. That is why it is very important to identify and understand the commonly known GSE, benefits, challenges and potential solutions to fully materialize the potential benefits and to
overcome those challenges. One approach of gaining a good understanding of the potential benefits and challenges is to organize them using some categorization scheme or framework. GSE community has reported different attempts to categorize and classify the GSE challenges and benefits. One of the best known classification framework for organizing GSE challenges and benefits has been presented by Ågerfalk and Fitzgerald (DD) [2].

Table 1: A framework to classify the benefits and threats of Distributed Development (DD) presented by Ågerfalk and Fitzgerald (Reproduced from [2])

<table>
<thead>
<tr>
<th>Process</th>
<th>Dimension</th>
<th>Geographical Distance</th>
<th>Socio-Cultural Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Temporal Distance</td>
<td>Proximity to market/customer</td>
<td>Innovation and shared best practices</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lack of informal communication</td>
<td>Asynchronous communication preferred by non-native speakers</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dependency on ICT</td>
<td>Language differences and misunderstandings</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Increased effort to initiate contact</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Providing technical infrastructure</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cost of travel</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Geographical Distance</td>
<td>Access to large labour pool</td>
<td>Mix of skills and experiences</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Standardisation in work practices</td>
<td>Language and cultural training</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allocation of roles and team structure</td>
<td>Lack of domain knowledge</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reduced trust</td>
<td>Doubtful of others’ capabilities</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lack of awareness/team spirit</td>
<td>Lack of mechanisms for creating shared understanding</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Modularity of work</td>
<td>Lack of awareness/team spirit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lack of mechanisms for creating shared understanding</td>
<td>Coordination complexity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Management of project artefacts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Control</td>
<td>Lack of concurrent engineering principles</td>
<td>Perceived threat from low-cost alternatives</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allocation of roles and team structure</td>
<td>Adapting to local formalized norm structures</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Different perceptions of authority/hierarchy</td>
</tr>
</tbody>
</table>

Table 1 shows their framework reproduced from their paper [2]. This framework consists of 3X3 matrix that has process and distance dimensions. The process dimensions are communication, coordination and control, which are also considered the most commonly known categories of GSE challenges and the distance dimension has been characterized by temporal, geographical and socio-cultural distances that are known to offer most of the benefits and challenge of GSE. In the following paragraphs, we provide a brief description of each of the elements of the distance dimension of their

Temporal distance represents a measure of time difference experienced by two people who need to interact [3] with each other. A temporal distance may be felt as soon as the hours of possible synchronous communication between two people are reduced. One such example is the situation where even one hour difference in time zones can lead to vary few overlapping hours and an appearance of higher than expected temporal distance, but may offer increased temporal coverage [41]. Conversely, a European worker liaising with a counterpart in India who is working a late shift may experience low temporal distance in practice [84]. Temporal distance may create a number of issues including reduced hours of collaboration, synchronous meeting difficulties, and availability of technical infrastructure that may cause response delays [3]. Hence, the elements of the process dimensions of the framework (i.e., communication, coordination and control) are expected to have significant affect as a result of temporal distance [2].

Geographical distance represents a measure of the effort required for one person to travel to another site for meetings [3]. This can involve several factors, such as the need to obtain a visa and the time it takes to complete the journey [84]. In general, low geographical distance offers greater scope for periods of co-located, inter-team working and unplanned informal communications [41]. Geographical distance makes communication, especially informal communication, difficult because of limited opportunity to hold face-to-face meetings [2]. It has been argued that a distributed environment changes the communication context away from an ideal face-to-face setting to a more complex technology-mediated environment [48]. Because of the geographical distance, many of the mechanisms of coordinating work in collocated arrangement become absent or disrupted [50],[63]. The coordination challenges in distributed software development may cause a work to take about two and half times longer to complete than similar activities when work is collocated [49].

Socio-cultural distance represents a measure of understanding that one person may have about another person’s values and normative practices [3]. When we talk about cultural, it can be national or organizational cultures [22]; and socio-cultural distance can be the
“difference” between two persons’ involved in terms of national and organizational culture. Socio-cultural distance is a complex dimension that may involve organizational culture, national culture, language, politics, motivations and work ethics [84]. Socio-cultural distances between two countries can be understood using Hofstede’s definitions of cultural dimensions for different countries based on his well known theory of Power distance [54]. Two persons may share the national culture but they may have quite a lot of distance when it comes to organizational cultures. Hence, it is important to understand the role of socio-cultural distance that is usually considered as a source of challenge, however, more recently researchers have started treating it also as a source of strength in terms of driving innovation and cultural intelligence.

Ågerfalk and Fitzgerald demonstrated the use of the framework to organize the challenges and threats of DD in their paper [2]. The summary of their review was reported by plotting benefits and threats in appropriate cells of Table 1 that represents the 3x3 matrix. One opportunity or threat can appear in more than one cell, as the categorization through this matrix is not exclusive. According to the representation scheme used, smiley face represent a positive effect or opportunity, a long face represents a threats, and a flat face represents no affect of any of the elements of the distance dimension onto the elements of the process dimension. For example, the cell corresponding to temporal distance and communication has time zone effectiveness as an opportunity to increase traceability and accountability through asynchronous communication via email or fax; the same cell has delayed communication as a threat as team members may not have sufficient overlapping time or no overlapping time and hence, things can get slow in terms of response time; the cell corresponding the geographical distance and coordination as an open DD issue.

During our review of the literature reported here, we kept comparing our findings with this framework. We assert that Ågerfalk and Fitzgerald’s framework needs to be extended in order to organize and classifying new types of GSE challenges and threats that have appeared over the last few year. For example, we would propose that the framework to be extended in at least two dimensions of distribution such as organizational/processes and Knowledge.
According to Gumm [43-44], organizational distribution can be referred as distance between different organizational structures and the differences between people who work in those organizations. Differences in commitments, working habits, and visions that are important in development processes can also be placed under this category. Different processes that each organization follows such as the way of informing about changes [24], different testing processes, and producing extensive documents versus relying more on self-explanatory code [46] can be considered as examples of the issues that are stemming from this distribution dimension. Gumm argues that the targeted organizations can be business partners, suppliers, customers, subcontractors of an organization and/or different divisions of the same organization [43].

Moreover, distribution across stakeholders [43] refers to the fact of distributing tasks, artifacts, skills and knowledge across sites. The author indicates that not only the project artifacts are distributed across different locations but also across different stakeholders. For example, requirements specifications are distributed across managers, analysts, developers and users. Furthermore, people with special knowledge or information who are scattered everywhere tend to have certain gaps in their knowledge of the technical and non-technical issues that are usually related to other sites with which they are developing software. We call this dimension as “Knowledge distance”, which should also be taken into consideration when organizing different arrangements of GSD and/or characterizing GSD challenges and solutions.

Another frame of reference that has been used for reviewing the literature reported in this report is the four areas based on which Sengupta et al. [96] outlined an agenda for GSD research. The four areas are: Collaborative software development tools, application of knowledge management, testing in distributed set-up, and process and metric issues. This paper was also one of the reasons for starting reviewing the literature from 2007 as we intended to discover how much the research agenda has been reflected in the research reported in the main venue of the GSE research and practitioner. Moreover, we also decided to use the four areas based on which the research agenda has been proposed for analyzing the reviewed literature in order to identify the areas which have attracted attention and the areas where more attention needs to be paid. Following
paragraph provides a brief description of the four areas identified by Sengupta and his colleagues as the potential areas of GSD research.

**Collaborative software development tools** are considered important in order to facilitate collaboration between distributed teams. The paper identifies the research gap in area of tool support for collaboration in software development phases like requirement engineering and design as compared to implementation related tasks, where a large number of collaboration tools have been development.

**Application knowledge migration and management** covers the difficulties and importance of appropriate mechanisms and tools to support knowledge management in the process of migrating application to remote sites. The identified research gaps are related to the enhancement and maintenance of legacy systems where existing knowledge needs to be transferred to remote teams. Sending selective members to visit client’s premises to acquire the knowledge and transfer to the local members usually does this transfer; however, the speed and accuracy of this process is considered one of the key difficulties. Provision of tool support in order to systematically capture the knowledge and quantifying transmission can increase the effectiveness and efficiency of the process. One of the key questions raised in this area is: *how to automatically create useful linkages between formal artefacts and the knowledge recovered from informal sources?*

**Testing in distributed environment** has also been identified as needing research efforts as there are several kinds of difficulties in distributed testing. Two main aspect of the software testing are 1) data management in terms of providing access to the data without compromising privacy, security, and legal requirements; 2) Minimizing errors and misalignment between different modules developed at different sites during integration.

**Process and Metric** areas need to be explored in the context of GSD as these areas have been extensively studied and well understood in the context of collocated software development [96]. Process and metric areas of research and practice include the development methodologies followed, practices and approaches applied, and metrics collected and analyzed. Researchers are expected to not only extend and validate the
existing process maturity models, like CMMI, but also need to develop and assess new breed of methodologies and metric suite that can incorporate the unique characteristics of GSD projects and their Key Performance Indicators (KPIs). There is also an urgent need to specifying metrics that can help concretizing each partner’s effort in the distributed software development.

2.3 Research Motivation and Objective

It has already been mentioned that Global Software Engineering (GSE) has become an important field of research and practice over the last 20 years. This realization has motivated several significant efforts aimed at identifying and understanding the challenges that characterize GSE as well as developing solutions to address those challenges. As a result of these efforts, the literature on different aspects of GSE has been increasing at a very fast pace. Hence, we assert that it is important to periodically review and report the main research trends and gaps in this area, especially the ones being reported in the main venue of the GSE community. This was the key motivation for this research that aimed at reviewing the research literature reported over the last five years in the International Conference on Global Software Engineering (ICGSE). Hence, the key objective of this research was to identify and report the key research gaps in the reported literature that can help motivate new research questions for others and us. We also intend to explore the impact that previous research agenda types of paper might have had on the research being reported through this venue. That is why we decided to review the research literature published in ICGSE between 2007 and 2011 as the paper [96] outlining the research agenda on GSD was published in 2006 and we wanted to review how the reported research in the area has been influenced since the publication of that paper.
3 Research Approach Used

As mentioned above, this technical report comprises the results of reviewing recent papers from the most relevant and mainstream venue of research and practice in Global Software Engineering namely International Conference in Global Software Engineering (ICGSE).

3.1 Research Questions

1) What are the main research trends of GSE being reported in main venue of this community over the last five years?

2) How much has the reported research on GSE been influenced by the practice-driven GSD research agenda outlining research gaps in the area?

3) What are the main challenges and potential solutions reported in the recent literature on GSE through the main venue of this community? (Solutions can be in the form of GSD reference models, processes, practices, and tools to address the reported challenges).

3.2 Selection of the Reviewed Papers

We started with downloading whole papers published in the mainstream venue of GSD i.e. ICGSE. This conference was started in 2006. We got the citations from 2006 until 2011 during which 275 papers were published in the proceedings of this conference. Since our objective was to provide a State-of-the-Art review of the papers published in GSD, we excluded all the papers except the research papers. The excluded papers included industrial track papers, education track papers, and workshops’ papers. This round of selection resulted in 108 research papers. As previously mentioned, considering the proposed research agenda by Sengupta et al [96] in 2006 and our motivation to investigate how a practice-driven GSD research agenda has influenced the research being published through the main venue on GSD, we decided to exclude the papers published in the ICGSE 2006 and focused on the recent five years (2007-2011). Having applied the exclusion criteria, we came up with 97 full research papers.
However, given the resource constraints (i.e., number of researchers involved and available time), we decided to perform this review in two phases. For the first phase, we randomly chose 10 papers published in each of the ICGSE edition from 2007 to 2011. That means we reviewed 50 research papers from ICGSE 2007 to ICGSE 2011. It is worth mentioning that our review of the first few papers revealed that it was necessary to review each of the selected papers in-depth in order to extract the data that we needed to achieve as the main objective of this review.
4 Review and Analysis

This section presents the summaries of the key points and critical analysis of the papers reviewed for this report. We decided to organize the reviews and analytical commentary on them by using the process categories of the framework proposed by Ågerfalk and Fitzgerald [3] to organize and classify GSD challenges and threats. Their framework has four process elements (i.e., communication, coordination, and control). However, we found their framework’s fourth element, i.e., control, quite restrictive in terms of its meaning in the context of modern software development methods as well as the abstraction level. It seems quite difficult to appropriately and meaningfully categorize the GSD researchable issues and solutions under such a coarse-grained category. Moreover, we also argue that a framework for classifying GSD research should have collaboration as a separate element as there are several issues, which are directly related to collaboration in GSD projects.

Hence, we decided introduce collaboration category in Ågerfalk and Fitzgerald [3]’s framework and also to change the name of the fourth element in their framework from control to software development governance in GSD. The latter category has been further refined into four more categories: models and processes in GSD, project management in GSD, general GSD practices, and knowledge management in GSD. In the following paragraphs, we provide brief descriptions of each of the categories, and relevant sub-categories, which has been used to summarize and analyze the reviewed papers with the expectations of providing a state-of-art trend in GSD literature published through the main venue of the community, ICGSE, from 2007 to 2011 and identify the gaps for further research in the area. We have modified the definitions of the first three categories, i.e., communication, coordination, and collaboration, from a dictionary.

Communication: Exchanging information, knowledge, thoughts, and ideas to support software development activities.

Coordination: Harmonizing the interdependent activities, tasks, artefacts, and interactions in software development.

Collaboration: Working together with mutual understanding of cooperation towards the achievement of a common goal in software development.

---

2 http://www.thefreedictionary.com/
**Software development governance:** Software development governance in the context of GSD means defining and executing certain decisions about certain rights and rules in order to ensure that the processes are followed and risks are managed [36]. We have refined this category into further four sub-categories as described above.

**GSD Models and processes:** This sub-category has been used to discuss the papers that propose a model of performing a particular aspect of GSD, e.g., developing software with geographically distributed software development teams, or describes a series of steps required to perform a particular task, e.g., requirements prioritization. A process’s description usually incorporates activities, constraints, resources, inputs, and outputs for producing a piece of software. We use this category to classify the papers that have studied and/or proposed solutions to the challenges related to any aspects of GSD models and processes.

**Project management in GSD:** This sub-category is related to the topics that are considered to be within the direct responsibilities and mandate of a project manager for a GSD project. The kinds of papers that have been placed under this category can be related to the challenges, solutions, methods, and tools for managing GSD projects, e.g., Scrum, risk management, distributed tasks allocation, and effort estimation for GSD projects.

**General GSD practices:** this sub-category incorporates the papers that study and/or propose concrete activities and work-products required according to a particular model, process, phase, or method used in GSD.

**Knowledge management in GSD:** Knowledge management can be described as an approach to gathering, managing, and using knowledge. We take a simplistic meaning of knowledge that is information that underpins a person, organisation, or other entity’s awareness of a particular subject and ability to use that information for a specific purpose, in this context developing software in GSD context. We decided to consider Knowledge management related studies as a separate sub-category from project management as it can be seen as a broader role within a company and not limited to the boundaries of a single project. Whilst there is a need to manage project related domain and technical knowledge, documents and competencies, there is also an equally important need of unified strategies for the company to optimally manage and re-use of the knowledge of different projects. Given the increased interest in this topic recently, it seemed appropriate to keep it as a
separate sub-category in order to enumerate the challenges and ideas related to knowledge management in GSD that might have recently been reported.

Having discussed our choice of the categories for reporting the findings from the analytical review of the GSD papers and presented brief descriptions of each of the categories, we report the summaries of and our analytical commentary on the key parts of each of the reviewed papers in the following sub-sections.

4.1 Communication in GSD

Software development is a communication intensive activity (both formal and informal communication). Most of the misunderstandings and problematic situations can be traced back to miscommunication or lack of communication. Hence, it should not be a surprise that communication related challenges have been quite frequently reported in the literature. Communication among GSD team members can be negatively impacted by several kinds of dispersion factors that characterize GSD such as geographical, temporal, cultural, linguistic and organizational distances. One of the main reasons for communication challenges in GSD is lack of informal discussions and face-to-face meetings [41]. A lack of communication can cause coordination breakdowns and lower level of awareness about the activities of group members located at different sites. When people have to communicate, allocate tasks, and coordinate their activities across distances, it becomes quite difficult to know about the status of different tasks assigned to colleagues geographically distributed and decide when to interact with them on ad-hoc and informal basis [3, 55]. Contrary to the long distance, short distance can enable team members to have more and frequent communication. That is why it is perceived that closer distances of culture, geographical, and temporal can make communication easier.

Given the significant role of communication and the potential challenges associated with it in GSD, there has been huge amount of effort dedicated to different aspects of communication in the context of GSD. Like many other challenges of GSD paradigm, researchers and practitioners have also started reporting various solutions to and strategies of addressing the communication challenges in GSD. These solutions and strategies are expected to help address some of the key communication challenges of
GSD teams. Following paragraphs provide summary and critical analysis of the reviewed papers which specifically focus on the communication related challenges and solutions.

Nordio et al [82] conducted a case study to investigate the impact of distance on the number of involved sites (for example, two locations vs. three locations) and time difference scale (varying from small, medium and large temporal distance) on communication. The results from their study revealed that the communication overhead among the teams who were sitting in small time distance is higher but also there is a significant difference in the amount of communication between the teams distributed at two locations compared with the three locations. Moreover, the measurement of the delays in replying the messages at three location sites is considerably higher than two location sites. However, when the authors asked the study’ participants about the impact of temporal distance on communication, it was revealed the team members did not notice any significant difference in communication, productivity and delay due to any of the studied distance dimensions (i.e., number of locations and time difference).

Apart from the studies reporting communication challenges, a number of recent studies show significant improvement in frequency and duration of informal communication within GSD teams due to the provision of supportive tools and processes as well as the increasing tendency of developers to adjust to distributed environments [4]. Researchers have also reported communication patterns in distributed teams from different angles such as communication models, effectiveness, and efficiency of communication and the impact of distance on communication delays and task completion [4, 76]. Al-Ani and Keith report that the patterns of communication found in distributed team members are more “ad hoc” and emerge during project development phases based on the needs; hence, organizations usually do not define formal communication models [4]. Their study describes that there is no significant difference among the frequency and duration of informal communication of distributed teams compared with collocated ones; although face-to-face communication is still more preferred and the use of synchronous media to communicate with collocated members is as likely as using them for communicating with distributed teams. The authors also report that distributed teams follow more decentralized communication patterns because they are perceived more
efficient and effective models. Furthermore, it was also reported that the size of a team also impacts the communication mode as small sized teams (5 or less members) are more likely to follow non-hierarchical approach (everybody talks to everybody), the larger teams might follow a kind of hybrid communication model based on geographical clusters where the communication brokers talk to each other or managers.

Nguyen and his colleagues [76] analyzed the data gathered from Jazz platform development project to investigate the communication patterns. Whilst their findings confirm that the results reported in [4] regarding the emergent networks of communication, the reported structure of the communication network is quite different to the one reported by Al-Ani and Keith. Nguyen et al.’s study shows the formation of a network with strong core-periphery structure rather than a geographically clustered model. One possible reason for different findings can be the different kinds of data collection (self-reported data versus social network analysis of the project data) and analysis methods (i.e., qualitative versus quantitative data analysis). Moreover, the study also shows that the observed delay in communication and task completion of collocated teams do not significantly differ from the ones that of distributed teams. These findings differ to the findings reported by Herbsleb and colleagues whose study found that the completion of a task by a distributed team could take up to two and half days more than by a co-located team [51]. Nguyen et al. argue that their findings of no significant time delays may have been a result of some specific arrangements in the studied project such as applying specific processes and collaborative tools or some unique characteristics of the Jazz development team [76]. We argue that this difference may have been the result of the availability and use of highly sophisticated and advanced technological support available nowadays compared with the days when Herbsleb and his colleagues conducted their study several years ago. Similar assertions can be made to explain the differences in findings from the recent and old studies on the communication related matters in GSD.

The performance of technology supported groups has attracted a large amount of research interest over the years [83, 85]. It has been reported that group interaction and performance is greatly affected by the nature of, and the level of difficulty of the task that a group is performing [74]. Moreover, technology, influences group interaction and
performance and also interacts with task type. Thus, it is important to understand the effects of different kinds of technologies on group tasks. There is a general consensus among researchers that differences in the nature of tasks should be considered when differences in group task performance are studied [74]. From the reviewed literature, we found that Niinimaki et al. [77] investigated the compatibility of media synchronicity theory with communication tool choice in GSD and concluded the application of the theory. Media synchronicity theory enumerates five features for a communication tool:

- Providing the possibilities for immediate feedback which is required for a communication tool in case of tightly coupled tasks with high synchronous interactions (e.g., Instant Messaging (IM) and telephone);
- The capabilities of a tool in expressing various messages and meanings over the medium that in context of software development activities, is quite needed for exchanging technical information; according to authors IM and Desktop sharing can be seen as efficient tools which offer high symbol of variety;
- It has been observed that providing appropriate communication channel (e.g., IM) parallel with programming gives better opportunity to the developers to discuss the issues on the spot and work more productively;
- Rehars ability;
- Reprocess ability are referring to the capabilities of a tool to give the user an opportunity of thinking and formulating the message as well as being able to refer to it later.

Authors observed the existence of language barriers, shortage of vocabularies, hardship in pronunciation of the words, and difficulties with understanding different accents. Whilst text based media (e.g., IM and Email) are expected to offer high rehars ability, email is mainly being used as repository for important information and provided functionalities facilitate the need of finding and sharing information with others even in existence of formal documents. The authors argue that communication media should be technically (e.g. compatible versions of the tools, infrastructure) but as well as socially (e.g., the agreements among team members to reply to the emails on time, logging into the instant messaging and being available for communication) compatible.
Instant Messaging (IM) is one of the most popular communication media used by GSD teams [35, 45, 78]. Niinimaki and his colleagues also explored the experiences of using IM in GSD [78] and showed that the special features of IM can improve awareness of the team members through availability status of their remote colleagues; moreover, the chat sessions can be saved for future reference as a source of knowledge. They also reported that the alerting, interactivity and multi-tasking properties of IM provide the opportunity for quick question-response as a side channel while doing other tasks (e.g. being in meetings). However, they also found that inappropriate use and application of these features could be as negative as they can be useful. For example, the wrongly set status information or repeatedly receiving alerts of new messages can cause frustrations [35].

Dittrich and Giuffrida [35] specifically explore the role of instant messaging in communication between distributed teams. Their study concluded that IM can work as glue to integrate formal and informal communication media due to four main capabilities namely provision of informal channel to discuss the requirements, tests and code, perceived as less disturbing method for initiating contact, provision of high awareness information about availability of the team members and capabilities to improve interpersonal relationships and building trust. The authors also concluded that the chat contents could be classified into four main categories of coordination, collaboration, awareness and socialization dimensions. The study found that people normally use informal chats for clarification and coordination purposes after task allocations. Moreover, the issues, which are easy to resolve, are collaborated over IM. Like many other studies and our own observations of GSD teams, the authors also found that IM was not only heavily used for work related purposes, but the team members also used it for socialization. According to the authors, the use of IM also helped providing real-time awareness of the tasks that are being carried out by different developers in order to avoid the conflicts.

The media synchronicity theory recommends using a combination of communication medium that can match the requirements of the kind of task to be performed. Hence, the choice of communication medium is quite important. Niinimaki et al [79] argue that communicator’s role, level of language skill, communication context, task properties,
availability of communication media, communication performance and satisfaction are the main factors affecting the media choice. Their results show that the technical people would more prefer the use of text-based media due to the format and type of the information (e.g., line of code, bug, and error message).

It is clear that communication and awareness are related to each other. That means improvements in communication increase team awareness; and poor communication can lead to lack of awareness, which can cause other problems from technical issues to coordination and interpersonal relationships among team members. Damian et al [30] studied the impact of awareness gaps on broken builds. Their results demonstrate that organizational culture can affect the level of technical awareness that stems from different strategies of companies and the habits of their employees of sharing specific technical knowledge; following different patterns and working habits and not having agreements and synchronicity about how and where (e.g., repository of a tool or via email) to find specific knowledge can cause serious technical problems as well as a lot of frustration that can impact inter-personal relationship and trust among team members. Their study also revealed that developers were able to adapt different tools such as bug tracking system, as means of communication that enabled them to exchange huge amount of tacit knowledge, which was scattered in emails, chats, phone calls, or any other tool. Dynamic adaptability of tools and exchange of tacit knowledge make GSD teams aware about whom to contact for a piece of information when formal process documentation does not provide any clue. The authors also found that the communication breakdown and information overflow have been observed as one of the major challenges that caused broken builds [30]. The analysis of the in use communication media and the number of the messages exchanged over mailing list, showed that the team members were bombarded with huge amount of irrelevant information that caused the important messages either getting lost or being queued to be read at a later stage. That was why the authors have highlighted the necessity of having a mechanism to prevent awareness overload by filtering the messages and forwarding he information to the relevant people only.

Communication is considered one of the best ways of creating and maintaining awareness about different technical and social aspects in software development team. It
is very important that the members of a software development team are kept aware for a successful project and a productive and motivated team. Given the importance of creating and managing awareness in GSD teams, several efforts have been dedicated to study and address this issue. From the reviewed papers, we did not find many papers, which specifically addressed the awareness aspects; rather we found the papers on communication related issues that also addressed the awareness issues. Aranda et al [10] has reported the features of a tool called “Trusting Social Network” based on the quantitative analysis of surveyed data gathered from software engineers regarding the information they would like to know about their colleagues. Their findings show that the information that is known or useful to be known by stakeholders might differ by having different distance factors. For example, speaking language or the gender might sound necessary to know at level of different countries involved however, not needed to be provided within the same country. The authors have also categorized the identified features into three main profile types: public, project and personal. They argue that categorization of needed information about colleagues in this format enables the tool to provide relevant information to the relevant stakeholders. Their argument is that general information such as name, contact information, working hours and availability status of the member may be useful for everybody while, some other personal information such as qualification, number of years of experience, age etc. might be part of information which people may like to share only with friends. Providing information such as role in the project or current activities under project profile is considered to be useful to share information with the project members who are interested in and increase team level awareness.

4.2 Coordination in GSD

Software development consists of a set of inter-related tasks that are assigned to resources in order to achieve the overall goals. Sooraj and Mohapatra define [100] coordination as the management of dependencies among different tasks that need frequent interactions among team members who are expected to have a common understanding of the status of the tasks being performed by team members. However, the spatial, temporal, cultural and organizational distances found in GSD team can
cause a lot of difficulties in communication and coordination. Cataldo et al. [24] argue that the management of the interdependency among tasks need appropriate combination of organizational structure, processes as well as communication and coordination mechanisms. It has been observed that despite all the efforts to define the processes, there is no guarantee that the defined processes will be followed for several reasons including processes being too prescriptive or lack of time and resources. Cataldo et al. [24] argue that lack of identification and management of technical interdependencies among modules can cause coordination breakdowns resulting in serious integration problems. They also conclude that the gap between documentation and source code can be one reason for coordination breakdown. Whilst agile approaches encourage minimal documentation, documentation is expected to be an important mechanism of communication between sites in order to overcome the cultural and linguistic barriers. That is why outdated documentation can potentially be misleading and can cause serious problems. Moreover, lateral communication can also help overcome the barriers of formal communication implied by organizational structure. Regular visits of the remote sites and having face-to-face meetings enable team members to promote informal communication and facilitate information exchange among them.

Sooraj and Mohapatra [100] have identified the coordination index considering a large number of person, product, and site-related factors. The authors argue that the levels of coupling among tasks, the amount of inter-changing requests between two parties and the context of delay in exchanging messages due to temporal difference are the influential factors in coordination of distributed teams. They also argue that both the needed amount of interactions and the effectiveness index should be taken into consideration in order to counter the negative impact of spatial, cultural and linguistic barriers. The study shows that the factors such as task complexity, the level of experience of requestor in working in distributed environments, team cohesion, availability of documents and error proneness of the requestor can significantly affect the number of requests that can be positively/negatively affected by factors such as media richness, interpersonal relationship and cultural differences. The authors also define the function of task completion time using the proposed coordination index that can facilitate the project management activities by providing the opportunity of trying
different possibilities of allocating tasks to the resources and opting for the best possible choice.

Boden, Nett and Wulf [17] have investigated the application of articulation practices within SMEs in distributed arrangements. Their findings demonstrate that the SMEs more rely on improvement of inter-personal relationships to overcome the complexities of work articulation. Regular visits of remote sites as well as making informal communication with team members via formal channels can complement the formal processes and organizational structures to negotiate and discuss the task allocations and sharing common understandings. Given the logistical and economic difficulties involved in frequent visits and the learning difficulties in the use of sophisticated tools, authors have recommended the development of tools as plug-ins that can be easily integrated with users’ working environments. One important observation is that there were not many papers, only 3, that could have been classified in the category of coordination from the set of papers reviewed for this report. Coordination is considered as one of the most important aspects of GSD as it usually results in a several kinds of challenges, which needs to be appropriately understood and addressed with technical and socio-technical solutions. Hence, we can assert that there is a need of allocating a significant amount of effort to provide sufficient and appropriate technological and socio-technological solutions.

4.3 Collaboration in GSD

A simplistic definition of collaboration means working together to achieve shared goals by having a common understanding. Software development is a collaborative activity to be undertaken by multiple stakeholders in order to achieve a common goal, i.e., a quality product. Collaboration requires intensive communication, and sharing of information and knowledge among team members. Like communication and coordination, GSD teams may also encounter several kinds of collaboration challenges because of technical and socio-technical factors. For example, collaboration may be hampered because of lack of information sharing as a result of mistrust among different team members or organization might not have provided appropriate technological support for effective and efficient knowledge sharing required for supporting
collaboration. In the following paragraphs, we review the papers that specifically targeted collaboration in GSE.

Bass et al. [14] define collaboration in GSD from three aspects namely people, engineering and infrastructure/tools. Whilst team building, competency management, and training can be seen as the people-related activities to improve collaboration, software architecting, project management, and software development processes should be taken into consideration as engineering-related parameters. Furthermore, provision of appropriate tools and communication infrastructure also plays an important role to support collaboration in GSD teams. Based on the perspective taken by Bass et al., it can be argued that collaboration is a multi-facet activity which needs to be fully understood from different angles in order to identify the commonly found collaborative challenges in GSD projects and sufficiently address them with appropriate technological and socio-technological solutions.

The GSD collaboration challenges can be classified in different categories. Liukkeunen et al. [71] have identified five main categories of collaboration challenges in distributed teams from literature: loss of communication richness, coordination breakdowns, geographical dispersion, loss of teamness and cultural differences. It has been reported that communication is the most critical issue in having good collaboration as lack of informal chats and face-to-face interactions implied by distance necessitate the need of appropriate technological support from communication media. Moreover, the difficulties in having good understanding and sharing common context due to cultural differences, lack of awareness about the state of activities at remote site would negatively impact the flow of information among team members that may results in technical (such as broken builds) as well as social (such as lack of trust and teamness) problems. However, project management practices such as synchronized milestones, decision making practices, formal commitment methods and decoupling work across sites as well as supporting practices such as frequent builds, integration of heterogeneous tools, practices for building trust and problem solving can help successfully cope with the abovementioned challenges [71]. Bass et al [14] recommend that the use of distributed pair programming and a forum kind of mechanism for responding to urgent request for resolving misunderstandings and conflicts at early
stages and promoting unplanned communication. They argue these practices should be supported by appropriate collaborative tools such as NetMeeting and web-based technologies that can support discussion forums and Wikis. Moreover, the authors also emphasize that the role of training, team building efforts, strategies for improving cross-site relationships in alleviating collaboration challenges. It is argued that sending individuals from central site to remote sites (cross-site delegation) can help achieve better integration of distributed teams consisting of members with divergent cultural backgrounds; management’s site visits not only can help managers to better understand the issues and difficulties at remote sites but can also improve the sense of teamness at remote sites.

Avram [11] specifically focused on the role of knowledge management practices in facilitating collaboration by studying the ways in which people use technology to establish collaborative work. She observed that the issues such as breakdowns in sharing technical knowledge, overloading sources of knowledge, and huge amount of effort needed to make comprehensive documentation to share tacit knowledge can cause serious issues in having smooth collaboration and common understandings during the software testing process that can lead to integration failures. She argues that such challenges can be overcome by adopting certain knowledge management practices to clarify “Who knows what” and “How to” knowledge as well as by increasing awareness by sharing mutual aspects e.g., shared calendars, circulating ReadMe files, sharing statistics and logs regarding the progress of testing. The author also claims that adding more resources to the projects in some circumstances seem to be vital; thus, documentation is needed for sharing context with other people and asking for help; however, documentation is perceived to be time consuming task that usually frustrates developers; Moreover, tacit knowledge can hardly be documented completely. Rather, tacit knowledge can be more effectively and efficiently transferred in face-to-face interactions and observations of doing a set of tasks. Social networks can also considerably improve collaboration among sites by facilitating communication and sharing good will and common sense.

Considering the collaboration challenges in large scale GSD projects, a large number of efforts has been geared to devise appropriate solutions, some of them supported by
tools, to minimize the collaborative complexities [86], [25]. Omoronyia et al. [86] present the Continuum of Relevance Index (CRI) model to monitor the interactions of developers within IDE and generate a list of relevant tasks, project artifacts and contact details of the relevant developers. They also provided a tool to support the CRI model. The special features of the tool included filtering information, reducing overload, and just providing needed information, providing separate perspectives based on project-related, task-related, artifact-related as well as relevance of tasks and artifacts can be seen as powerful aspects in managing knowledge and increasing the level of awareness.

Cataldo and Choi [25] have identified a set of challenges that a distributed design tool should address. The identified challenges include information sharing, availability of sufficient space for drawing, managing focus and shared understanding and storing design relevant information and rationales. They have also developed a tool, CAMEL, to address the identified challenges of collaborative design in the context of GSD. However, these tools are at very initial stages and may be considered as research prototypes that need to be systematically evaluated and fully developed before they can be deployed in a real-life case study of large software development teams.

Serce et al. [97] have reported that task type, culture and Grade Point Average (GPA) can be influential factors in the communication and collaboration patterns followed by different distributed groups of students. The findings from their empirical study demonstrate that task type seems to be one of the most important factors in promoting collaboration among team members; contributing tasks for exchange resources and giving feedback, planning work and group skills are rated among highest proportion of observed collaborative behaviors. Moreover, it has been observed that the performance of the teams who have had heavier amount of communication and used a diverse set of communication media and tools has significantly improved compared with other groups. It is worth mentioning that the collaborative activities of SMEs are not as much complex as of large projects, however, the SMEs are likely to face other kinds of challenges such as poor skills using communication tools, weakness in selection of right communication tool for external use, not being able to have the support of appropriately and sufficiently skilled people because of budgetary constraints, and changing from one tool to another to adapt to customers demands [71]. From the reviewed papers and our
own observations of the collaboration challenges in GSD projects, we can conclude that challenges in collaboration can have significant negative effects on the overall outcome of the GSD projects. Since most of the collaborative challenges are intertwined with other challenges, e.g., lack of communication may have negative impact on team building effort as a result of which there can be lack of knowledge sharing leading to problems in collaboration. Hence, the collaboration challenges not only need to be studied and understood individually, e.g., lack of information flow, but also should be studied and addressed in the context of other challenges such as communication and coordination.

4.4 Software Development Governance in GSD

Governance has recently emerged as an important area of research practices in the context of software development in general and global software development in particular [36]. Governance in the context of software development means defining and executing certain decisions about certain rights and rules in order to ensure that the processes are followed and risks are managed. An increased realization of the importance of this area has resulted in increased research and community building efforts (e.g., workshops collocated with ICSE3). Recently, a team of researchers and practitioners have identified four main governance mechanism based on a study of GSD teams in 6 organizations (4 large and 2 small organizations). The identified governance mechanisms are: governance definition, governance enactment, business awareness, and goal achievement. Through correlation analysis, authors found that a combination of high level of governance definition and high level of goal achievement results in low level of team conflict. The authors have also found a positive correlation between the use of different tools, in particular collaborative tools, and governance mechanisms. According to the reported research, the four mechanisms of governance can be defined as below [36]:

Governance definition means establishing rules, rights, work procedures, measures, and policies.

---

Governance enactment means the actual execution of the governance definition in the software development process to be governed.

Business awareness inspires the alignment of team knowledge with the business vision and goals that can be executed through regular meetings of the team members with the stakeholders to hear their vision and receive their feedback.

Goal achievement means the achieved results against the business goals of both the development process and the governing of its; the project goals are considered to be achieved when delivery is on time and the stakeholders are satisfied.

4.4.1 GSD Models and Processes

It has been mentioned that software development paradigm has significantly changed during last two decades and there an increasing tendency of adopting some form of GSD to benefit from the promised advantages such as having access to the pool of expert labor with lower salary, acquisition and mergers and getting close to the market in specific area. The adoption of GSD model have introduced new challenges in several aspects of software development process such as project management, coordination, and control of a team due to different distribution factors.

It is well-established reality, as mentioned several times in this report already, that GSD has its own challenges. Neither can a company expect to be successful with GSD initiatives because other companies with similar characteristics have succeeded; nor can every project within a company with successful record of GSD be expected to succeed. Companies need to carefully assess the GSD options in order to come up with a set of appropriate strategies and then meticulously execute those strategies with appropriate mechanisms of monitoring and measuring. In order to guide the companies for drawing a suitable set of GSD strategies, Lings and his colleagues have proposed a reference model of ten strategies to support Distributed Development (DD) of software systems [70]. Their reference model has been built based on extensive literature review, field studies, and eight case studies of DD. Whilst authors claim that the reference model purports to guide an organization towards a successful DD, they also acknowledge that it cannot be considered a blueprint for DD in any context. However, they argue that deviations from the reference model’s strategies need to be justified based on the
contextual factors, which may not be the same for all kinds of companies and all kinds of projects. The authors have concretized their reference model not only for DD arrangements across geographical borders but also for intra-national DD, which is supposed to be slightly different to GSD. Their reference model consists of the following ten strategies:

- **Distribution rationale:** Since DD may not be the best solution for all companies and all projects; organizations should take into consideration all the circumstances such as distance factors and available competencies in order to make a wise trade-off.
- **Clarify all understandings:** At the beginning of the project, the managers should make sure to have arrangements in place that everybody would be clear about the goals, targets, and commitments. So documentation of the inter-organizational processes is necessary.
- **Leverage modularity:** It is important to match organizational structure with the system architecture to be distributed (according to Conway’s law).
- **Use Cultural mediation:** It is to have a role of liaison that is familiar with the culture of the remote team to facilitate smooth communication between teams.
- **Facilitate human communication:** It is necessary to provide technological support for rich communication and to facilitate human communication.
- **Manage processes:** Despite having many roles and overlapping between roles, it is necessary to have one identified person as main leader who is taking the full responsibility of a project and regularly communicates with local leaders.
- **Develop a sense of teamness:** The sense of teamness needs to be developed among virtual teams; having an internal website which contains the personal information about team members, their interests, public holidays and so on can be helpful.
- **Encourage temporary collocation:** It is important to team members working all levels at a remote site visit team members the central location.
- **Address heterogeneity:** There is a need of having synchronized tools, methods, and terminologies used at all sites. Having a central site, which can act as a coordinator between all the remote sites and make sure the heterogeneity is being addressed is beneficial.
• **Develop an effective tool base:** It is important to choose appropriate tools that address the real problems e.g. common configuration management tool; as it is always slow to start with a new tool.

When organizations decide to send their software development offshore, they usually adopt different models of off shoring. There are different kinds of practices, which range from offshore outsourcing to offshore internal offshore development according to which an organizational owns the development centre that may have previously been owned by a vendor. The offshore development practices follow different patterns of evolution in organizational strategy of off shoring. It is not recommended to offshore the entire project from the beginning, as it can be a big risk. Hence, organizational internal off shoring needs to evolve over time in terms of distributed work structures and development responsibilities. Hence, organizations need to decide as to how to distribute the work assignments.

Prikladnicki et al. have proposed a maturity model of organizational evolutionary practices of inner off shoring. The model consists of 4 dimensions, which are capability levels, capability attributes, subsidiary interdependency, and type of attributes (i.e., people, project, portfolio, and subsidiary). The capability attribute dimension consists of 28 attributes (such as learning, training, cultural differences, and awareness of activities) that are expected to evolve over time when an organization is engaged in internal offshore software development. The capability levels dimension has three levels: develop basic capabilities, improve basic capability, and at the third level there is usually constant effort to improve the performance of the subsidiary. The last dimension of this model is related to the dependencies among several subsidiaries of an organization, which may have several subsidiaries that are expected to collaborate if all of them are working on the same project or on different aspects of the same component or module. However, it is recommended that organizations avoid having too many interdependencies among several subsidiaries as it usually results in huge coordination responsibilities. Despite being a preliminary model, authors claim that their model can be a guiding tool for organizations to decide how much and when they can outsource software development to their off-shore development teams [93]. We believe that this model needs to be rigorously assessed in different organizational and project contexts.
before being widely recommended. Hence, one of the research challenges in GSD is rigorous and systematic evaluation of the models, tools, and practices proposed for supporting and improving GSD teams.

It has been stated that there can be different formats of GSD, which initially started from software development outsource for particular tasks, e.g., testing and maintenance. Over the years, the forms and nature of GSD has changed in many ways with several different models of GSD devised and adopted. Prikladnicki and his colleagues [92] have also identified four main business models of GSD: “Onshore in-sourcing/shared services”, “onshore outsourcing”, “offshore in-sourcing/internal off-shoring” and “Offshore outsourcing”. These four business models of GSD have been presented in a 2x2 matrix based on two main distribution factors: geographically (within or out of country border; offshore vs. onshore) and organizationally (within or out of organizational border; outsourcing vs. in-sourcing). Their study aimed to investigate the differences and similarities in challenges faced by the projects running within different business models and the relationships between development processes, project size and complexity. The results of their study show that defining a common infrastructure is necessary for applying out-sourcing strategy, thus, providing a formal change management definition, knowledge management, and documentation become critical for the projects running in this arrangement. Moreover, it has also been observed that requirement engineering can be a major challenge in off-shoring strategy. The activities within the requirement engineering process such as requirement elicitation needs a lot of interactions within team members and context sharing that can be quite challenging due to cultural and geographical distances. Communication problems, and language and cultural differences can have negative impact on building trust and providing appropriate level of awareness. The suggested solutions for coping with these challenges in all business models include standardization of software development activities, coding styles, and tools. The study reports that the onshore in-sourcing model represents the simplest level of process definition, the offshore outsourcing is the most critical level for process definition for all types of companies despite the fact that small companies usually apply light processes and large companies usually follow standard processes and even apply heavy ones.
One of the key characteristics of GSD is geographical distance, which usually results in other kinds of distances such as temporal and cultural. Apart from real distance, software development teams can also be challenged by perceived distance despite being physically not far apart from each other. It has been argued that a lack of perceived distance can cause several kinds of challenges, as managers may not be fully aware of the distance perceived by GSD team. Since quantification of the perceived distance is not an easy task, this aspect may not be taken into consideration while designing strategies for addressing communication and cultural issues in a GSD team. Drawing upon the definitions of perceived distance, perceived proximity, and the five centrifugal forces proposed by Carmel, Prikladnicki has proposed a model called Perceived Distance Index (PDI) that is designed to quantitatively measure the perceived distance in a GSD team [91]. This model purports to help a project team to use numbers to represent subjective feelings and interpret those numbers in the context of a particular project, which may have team members geographically distributed. Apart from the well known concepts from psychology (i.e., propinquity) and GSD (i.e., centrifugal), the model is also based on other assumptions about GSD teams, for example, it is argued that a well composed team may not feel the need of as much of physical contact as is usually perceived. Whilst the model appears to be contributing to the growing body of knowledge about dealing with human-centric factors in GSD, there needs to be further research required before the model can be considered a fully validated tool that can help project managers to quantify the perception of distance in their GSD teams. The author has also acknowledged that the presented form of the model is quite preliminary and they plan to fully validate the model. We believe that such models can be quite valuable as they can help managers to objectively study the human factors involved in GSD project in order to devise appropriate strategies; however, their use needs to take into consideration of the underlying assumptions, contextual variables, and validation mechanisms.

One of the potential challenges associated with GSD is ambiguity in responsibilities, roles, and assigned tasks and agreed upon processes to be followed by different sites, which can be referred as process distance. Non-uniformities of processes such as variation in definitions and mismatch expectations would cause coordination issues.
According to Heijstek et al. [46], lack of the definition of the work units such as design documents, lacking prescriptions on methods of building work units and lacking prescriptions on procedures for knowledge management can be enumerated as examples of frustrating issues stemming from absence of structured processes. The authors argue that provision of tailoring process description for GSD-specific issues by organizations might help to address abovementioned difficulties. The findings of their study demonstrate that defining process description for GSD is desired by organizations; not only it can be used as “course-material” for new team members, but also due to “repeatability of the approach” it provides the opportunity to predict the development process; moreover, organizational maturity and obtaining certain level of CMMI can be also another motivation for preparing process description.

However, the study shows that the process description approaches adopted by the companies vary in their structure (e.g. using lists and tables vs. text supported by various UML diagrams and free from diagrams), the level of details (e.g. outlines, less details or only GSD pitfalls), and the levels of integration with project management activities. Furthermore, definition and application of processes adopted by different companies are also influenced by the size of the organizations; while larger organizations tend to adopt formal processes, smaller companies shift more responsibilities to individual project managers.

It has been stated that as a result of growing popularity of GSD, like other traditional software development processes, software requirement engineering, architecture design and evaluation processes also face new challenges, which need to be addressed by designing and evaluating new processes to tailor and/or scale traditional processes to support their practices in GSD projects [6]. That is why GSD researchers and practitioners have been allocating significant amount of effort to adapt and/or scale existing methods, approaches, and tools to support GSD teams. Like many other researchers [32], Prause et al. [90] have highlighted the need of scaling the existing requirements engineering processes for GSD by reporting difficulties and challenges of conducting user-centric design in a large GSD project with a number of distributed stakeholders. In the reported case study, they also studied the suitability of web-based tools and sophisticated requirements schema in alleviating the observed challenges. The
reported case project required all the stakeholders, who were distributed at various sites, to fully analyze and understand the functional and non-functional requirements and their potential effects on each other by having close interaction and face-to-face meetings. However, close interactions in face-to-face meetings became quite challenging because of cultural, temporal and geographical distances. To address these challenges, the authors applied the Volere requirement schema with bug tracking system Jira that provides some special features that can be used to minimize the need of face-to-face meeting. Their results show that not only the Volere process ensures that all the important aspects of the requirements are addressed, the special features of Jira (such as sending notification emails to the relevant stakeholders about the changes and enriching the requirement description with comments), can significantly fulfill the communication needs. However, authors warns that a sophisticated tool like Jira may have need significant amount of time and energy to overcome the steep learning curve before the potential benefits can be realized.

Laurent and colleagues [69] have presented a modeling language associated visual notation for planning RE activities in GSD projects. Their approach distinguishes between three general entities (i.e., roles, sites and artifacts) that are involved in the RE process and describe the relationships between them within the proposed model. Their approach organizes information in RE process by storing site-related information such as location, language, time zone as attributes of the entity as well as classifying the artifacts as stationary and traveling and provides master role types (i.e., subject matter expert, requirement analyst, location spokesperson) and requires. The authors argue that the simplicity of their visual modeling language makes it easy to understand and use for all kinds of stakeholders. Moreover, the tool supporting the modeling language provides the features to generate underlying graph of a model that can potentially help project managers to proactively recognize the needed communication and collaboration processes and the need infrastructure.

Ali Babar has argued that software architecture evaluation methods need to be scaled in order to meet the demands of GSD [12]. Author has identified several aspects of the existing architecture evaluation methods (such Architecture Trade-off Analysis Method (ATAM) [26], Performance Assessment of Software Architecture (PASA) [103] and
Architecture-Level Maintainability Analysis (ALMA) [68]) that needs to be modified in order to deploy them for GSD teams. The author has proposed a framework aimed at providing process and technical guidance to support the software architecture evaluation process using electronic workspaces provided by groupware systems. This framework also identifies some of the unique tooling features that are required to support the proposed process. This paper also provides a details description of the framework and discuss the potential benefits and socio-technical challenges of introducing groupware support for the software architecture evaluation process.

Al-Ani and Redmiles [5] investigated decision making process (DMP) in distributed development teams specifically based on the role of leadership, culture, and distribution of the team. They argue that reviewing the literature show two main approaches of leadership; while participative (relationship-oriented) leadership is more based on equalization of power, creating collaborative environment, sharing problem solving by consulting with others, directive (task/goal-oriented) leadership is more focused on achieving goals and accomplishing problem solving tasks. Authors have defined three main types of DMP as consultation where the leader makes decision with consulting with other members who are either known as experts or being affected by the decisions, consensus where the issue is being discussed by selected members or everybody and being decided until everybody agrees and autocratic when leader makes decision without considering the opinion of any of the team members. Their study shows that participative leadership is dominant and effective approach in distributed development due to the involvement of diverse cultures, unstructured tasks and lack of teamness characterized by distribution factors. Whilst decisions are usually made in consulting with team members, leaders are still the main decision makers regardless of the size and distribution of the teams. The authors argue that these findings outline the significant role of the leaders and importance of assigning this role wisely. Al-Ani and Redmiles also noted the important role of culture in DMP and reported that organizational compared with regional culture influenced the decision making approaches adopted in an organization. They also noted that most of the decision-making processes were ad-hoc and emergent that made it hard for employees to understand the reasons and reflect upon them. The authors claimed that providing a decision-making support tool with the
ability of documenting the DMP and allowing the remote members to contribute to it considering the time zones would improve the efficiency and effectiveness of the decision making process.[5],

Regarding the importance and complexity of decision making in distributed development teams, Barney et al [13] explored the levels of alignment between different stakeholders on software quality aspects. The authors argued that this misalignment results from three main factors: the ambiguous and unclear quality requirements make the gap between what developers and testers assume as required quality with customer and management expectations. Moreover, it has been observed that cultural differences significantly influence this misalignment; whilst the European developers proactively participate in effort estimation and openly negotiate the possibilities with the managers, their Asian counterparts mainly rely on what managers know about their needs and capabilities and try their best to fulfill the expectations. Though in many cases they even produce more than their European peers in the same period of time failing to reach the expectation of managers always bring the filling of dissatisfactions and frustrations. It has been observed that another contributing factor is the different temporal perspectives of the roles to the software quality. The product support team are more aware of customers’ needs based on their past experiences, the architects opt for more sustainable long term solutions; in other word, the technical people usually aim for technically perfect solutions which sometimes require excessive investment, however, that might not be commercially justifiable.

4.4.2 Project Management in GSD

Due to the complexity and the importance of project governance in distributed arrangement, this topic has been widely studied by researchers who have proposed different models and frameworks for task allocation, effort estimation, and evaluation of cost overhead in GSD. They have also reported several practices and lessons learned about the overall challenges and difficulties of project management, decision making, dealing with software quality aspects and role compositions at large scale projects as well as SMEs.
One of the most critical decisions about a software development project is selecting an appropriate development methodology, which matches with the available resources, capabilities, and a company’s circumstances. It is also equally critical to ensure that the chosen methodology and its practices are aligned with an organization’s overall goals. Software development community has witnessed a huge popularity and adoption of agile development method such as Scrum, Dynamic systems development method and XP. Agile methods heavily rely on close interactions among team members and stakeholders. However, the requirement of close interactions in GSD may not be viable as dispersion of team members usually results in a lack of face-to-face interaction and infrequent synchronous communication. These characteristics of GSD are likely to be quite risky and present several challenges. Given the popularity and need of both GSD and agile methodologies, there have been significant amount of efforts geared towards identifying, understanding, and resolving the problems in combining GSD and agile methodologies, for example, the use of Scrum in GSD projects has attracted significant amount of attention [57, 59]. Agile and GSD researchers and practitioners have not only focused on the strategies of adapting agile practices to distributed arrangement, but they have also been paying attention to study the means and effects of using GSD practices (such as frequent visits, having multiple communication modes, decreasing the inter-dependency across sites by mirroring the roles, rotating ambassadors who carry knowledge across sites and synchronization of working hours) to facilitate distributed agile development [89]. Paasivaara et al. [88-89] have reported findings from a multiple case study that aimed at investigating how the studied projects applied agile practices, i.e., Scrum, in distributed arrangement, which GSD practices were used and what benefits and challenges were observed from applying agile practices to GSD projects. The authors observed challenges in applying agile to GSD at the beginning, however, the overall experiences of the teams of all projects were positive as a result of using certain GSD practices to overcome the limitations of agile which have been designed for collocated software development. Some of the reported good practices for applying Scrum in GSD include well-schedule daily scrum meetings by taking temporal differences and unavailability of key members at certain time into consideration, conducting weekly scrum of scrums, arranging distributed sprint planning meetings in different phases by taking into consideration of different working hours and the needed
inputs to attend the meetings, ensuring nightly builds, and performing automated testing. They observed that following Scrum practices had positive impact on other issues of distributed development for example initiating informal communication or ad-hoc meetings. The study revealed that attending the official daily scrum meetings and scrum of scrums made it easier for the project members to start talking to each other, overcoming the silence caused by distance, and initiating ad-hoc meetings. The authors also observed that incremental builds and frequent communication helps quickly resolve misunderstandings that can help improve the quality of the final product and increase motivation that usually leads to less frustration. However, the issues like poor network connections, limited opportunity of videoconferencing, poor sound quality, and difficulties in recognizing speakers without seeing them is still an issue to be resolved.

Del Nuevo and colleagues [33] have proposed a scrum-based distributed software development method called scRumUP. They argue that while agile practices (such as daily stand up meetings, and scrum of scrums) can help alleviate coordination and control challenges of GSD projects by improving communication, the emphasis of RUP on establishing development processes and documentation can help scale the agile approaches to the requirements of globally distributed projects. The process of scRumUP leverages RUP principles and incorporates scrum practices for the project management activities. However, scRumUP relies on Test Driven Development (TDD) and Continuous Integration (CI) for software testing instead of the testing practices of RUP, which are based on implementation of components with well-established interfaces. The TTD and CI are expected to enable distributed software development teams to clarify the issues and misunderstandings at earlier stages [33].

Almeida et al [9] argue that despite a large number of studies about the application of Scrum in GSD, there is a lack of literature on project planning methods. A majority of the studies are focusing on the benefits and challenges of Scrum practices in distributed environment whilst there is no established model targeting the complexity and criticality of combining Scrum and GSD. Thus, the authors propose a multi-criteria model for planning and fine-tuning distributed scrum projects by refining a Multi-Criteria Decision Analysis (MCDA) approach for GSD. Compared with the previous approaches that apply MCDA for tasks allocation and cost reduction in GSD projects,
their approach applies multiple criteria for improving decision making for GSD project planning. The proposed decision making model applies cognitive mapping technique to cover three different types of criteria, namely general projects, GSD projects, and Scrum projects, and facilitate project planning and tuning by providing a wide range of considerations.

Risk management has been always a critical task for project managers as it can significantly influence the success of a project and productivity of software development teams. The dynamic nature of GSD projects implies more complexity and at the same time criticality to the process of identifying and mitigating risks. Researchers have looked into the possibilities of dynamic identification of risks in distributed arrangements [75],[64] in form of models and argue that their model can effectively facilitate project management in case of planning, task allocation and making decisions.

Mudumba and Lee [75] have proposed a framework for dynamic identification of risks based on the analysis of dynamic interactions among internal GSD elements (processes, people and technology) and interactions with external elements (e.g., environment, organization, and project specific characteristics); they apply multiplicity factors viewpoint stem from distribution circumstances such as involvement of multiple-culture, multiple-stakeholders, multiple-language, multiple-locations to describe the risks and provide agile mitigation practices. For example, the people-to-people interactions can cause challenges in knowledge sharing and due to language barriers, hierarchies, and different perceptions of stakeholders; however, these kinds of risks can be addressed by using agile people management skills such as localization of the skills, visiting remote sites, and cultural trainings.

Lamersdorf et al [64] have contributed to the area by defining a model for identifying risks of GSD projects at early stages using the lessons learnt from the history. They propose a model consisting of three dimensions namely risks, influential factors, and the logical rules, which describe the relation between the two elements. The authors define the model based on qualitative analysis of 19 interviews and have come up with a set of logical rules which define the causal relation of reported issues and observed risks. Their approach is based on categorization of factors associated with project
characteristics (e.g. process maturity, product size), tasks characteristics (e.g. criticality, complexity, coupling factor), site characteristics (e.g. transparency, technical knowledge, process knowledge, staff motivation) and the relationship between sites (time zone differences, cultural differences, interpersonal relationships). The researchers argue that the provision of such model can significantly help project managers to carry out their daily tasks more efficiently and effectively and can facilitate other activities such as tasks allocation process. They prove their claim in their next study [65] by integrating the risk-based model with their previous work on defining causal model and work distribution [63],[66] to introduce a risk-driven model for work allocation in GSD projects. They also discuss the dynamicity of the model from two angles; not only the risk identification model can evaluate the suggestions of task allocations but also the lessons learnt from the risk model can be formalized and gives as input for task assignment suggestions.

Lamersdorf et al. [66] have presented a decision model for distribution of software development tasks. According to them [66], a Distribution Model (DM) would consist of four main elements: “Goal”, “Tasks”, “Resources” and “Mapping”; such a DM can be seen as a function which describes mapping of tasks to resources in order to achieve multiple goals (e.g. cost saving, increased productivity and quality). Lamersdorf and his colleagues also define the six main requirements of a DM that they use to evaluate a number of existing decision models from different domains including distributed software development, distributed production and distributed systems. They conclude that none of the existing models is appropriate for distributed task allocation due to the shortcomings in fulfillment of the requirements such as adaptability, multi-objective goal, and empirical-based criteria. However, the algorithm defined for assigning tasks to processors in distributed systems domain seems to be closer to the defined requirements. Nevertheless, this model is based on a set of sequential tasks; that means it is only applicable to traditional software development methodologies and does not support agile and/or iterative development methods.

Another study by Lamersdorf et al [62] demonstrates that number of different factors such as task criticality, task complexity, formality of task description, process maturity, technical knowledge, requirement stability, and coupling between tasks can impact the
effort overhead and productivity of software development teams. The effort estimation model proposed in [62] has been based on CoBRA approach. It identifies a set of effort drivers and the causal model that describes the relationship between these factors and the estimation of quantitative impact of each factor on effort overhead based on a set of interviews of the personnel of completed projects. Authors argue that the development of such a model can significantly help practitioners to save project costs, improve productivity in GSD, and to predict project plan and systematically allocate tasks.

Another category of studies of project control and governance in GSD focuses on analyzing qualitative data to provide lessons learned from past projects, challenges and practices. Richardson et al [94] report findings from case studies of SMEs to investigate the applicability of the strategies defined as reference model for distributed software development by Ling [70] in small-medium sized companies. The results show that the success of SMEs mainly relies on their potential in creating social capital based on forming good personal relationships between virtual teams by using cultural mediation, applying the practices of agile methods (e.g., scrum meetings) to encourage developers to talk to each other, aligning working hours to maximize work overlaps and arranging regular visits of managers to remote sites. These are considered good practices that can play an important role in making GSD successful.

4.4.3 General GSD Practices

In this section, we provide the summaries of the reviewed papers which have been categorized in general GSD practices, which can be used with different models and processes as well as project management approaches.

Smite and Wohlin [99] report a set of lessons learned from a case study in order to transfer product to remote sites; the findings show that organizations not only should have clear idea about finding the right people at the remote sites who can accept the knowledge transfer and continue running the project smoothly, but also have perform appropriate planning and strategy for transferring the competencies including explicit and implicit knowledge. It has been reported that though transferring people with the product can be seen as a quick and effective solution to start with the transfer, it cannot be a long term solution and there is a need to plan for co-located hands-on training.
According to the recommendations, the staff at the new site should gradually take the responsibilities of a project after being sufficiently trained and coached. It has been observed that transferring a product with poor documentation by relying on the source code or people interactions to understand the design decisions and logic behind them would lead a project to big problems due to misunderstandings. On the other hand, agile software development methodologies tend to have minimal documentation and focus on delivery. The authors argue that there is a need to have a balance in that respect. Project managers should ensure product documentation. Transferring knowledge to people who are coming with different cultural background can be easier if managers arrange training sessions to teach about the culture of the colleagues who are sitting in the other sites.

Software development teams consist of different roles each of which is supposed to have different kinds of responsibilities and skills, e.g., business analyst is responsible for requirements elicitation and specification and needs to have skills in interviews, documents inspection, and technical writing. It can be assert that GSD may have different kinds of (e.g., positive, negative, no affect) impacts on different roles involved in GSD project. Lane and Ågerfalk [67] studied the role compositions of distributed sites and investigated the suitability of particular roles (i.e., business analysts, designers, developers and development-support teams) to GSD by comparing each kind of role located at different development sites. The framework reported to categorize the GSD challenges and benefits using critical incident technique of interviewing guided their study. Their findings revealed that GSD challenges outweigh its benefits for the role of developer. One of the challenges reported by a developer was caused by coordination complexity related to both temporal and geographical distance. There was also a lack of trust between remote developers as a result of difficulties in understanding poorly documented framework patterns and extension mechanisms. The authors observed only one benefit reported as the team experienced some efficiency as a result of time zone differences. The studied design team was distributed at four locations. A general finding was that there were many obstacles to effective design between remote parties such as lack of informal communication that eroded the trust. Coordination issues reduced team spirit and increased frustration caused by the short overlap in collaborative time. It was
observed that the coordination across geographical distance had the largest effect, as it was difficult to effectively communicate abstractions or draw conclusive agreements. However, when the designer spent some time together, they developed strong relationship and appreciation of each other’s approach that helped them to create a collaboration process, which they followed after returning to their respective places. In case of the development-support role, the benefit outweighed the challenges. The development-support teams used clearly traceable processes that promoted the infrastructure and culture of a learning environment. Moreover, the time zone effectiveness was also of great benefit to this role. There were also some challenges as a result of redeployment of certain people but these challenges did not have a significant impact on the role.

The challenges and benefits of GSD on the suitability of the business analyst’s role appear to balance one another. It was found that certain aspects of the role could be performed in a collocated fashion than in a distributed one. However, several drawbacks to the role of business analyst in GSD were noted; for example, cost of preparation for communication to prevent misunderstanding, delays caused due to misunderstanding, and travel cost dealing with certain complex issues that required richer context.

While there are many studies discussing the challenges of distribution factors in GSD, Gumm investigates the mutual dependency of distribution benefits and challenges by focusing on requirements engineering [44]. Her findings demonstrate that GSD benefits depend on project settings, adopted processes, tools and social actions with software development and there are tight interdependencies between observed issues. She argues that GSD benefits are discussed controversially; while one factor can be seen as benefit from one perspective, it may be perceived as a disadvantage by some others; she proposes the use of causal chains and loops for representation and better understanding of these interdependencies. Her work has realized five main benefits of distribution as process definition, documentation, requirements, autonomy, and working situations. According to the author, working in distributed environment make a team realize the necessity of having a defined process as clearly defined processes can make the establishment and maintenance of a process easier. Furthermore, defined processes imply having richer documentations of the requirements change requests, the analysis
results and the key decisions. Such kind of documentation not only decreases the loss of information, but also provides a good support for knowledge transfer and communication based on documents. Moreover, development team gets an opportunity to learn about specific aspects of requirements as a result of the involvement of a variety of customers. Furthermore, Moreover, physical distribution also gives the software development units an opportunity to work with the local working culture and processes without being too much dependent upon the firm’s centralized processes. Such autonomy is expected to result in higher process pace and better formulation of requirements. Despite all the well-known challenges of temporal and physical distances in GSD, the findings report the beneficial aspect of distribution in terms of improved quality of an individual’s work. The author concludes that having rare meetings to attend, calm and unhurried environment and fewer interruptions make it possible for an individual to work on the problems more intensively. Moreover, an individual can experience more freedom. Author also concludes that temporal and geographical distance decreases the pressure and guilt feelings upon delays, as it is quite stressful when the manager or the one who needs the work is around and come to the developers quite frequently.

Due to the criticality and difficulties of using agile in distributed development, it is important to benefit from the experiences of knowledgeable people to provide training and/or coaching to project teams before they are asked to start using agile practice in GSD projects. For this purpose, researchers have coined the word of “Agile Coach” that refers to either internal agile experts or external consultants who can take up different roles such as teachers, facilitators or problem-solver in order to help distributed software development teams to successfully apply agile practices [1, 87]. Paasivaara [87] has investigated the role of agile coaches in successfully implementing agile approaches in GSD projects. She has reported a set of lessons learnt based on the observed challenges and experiences from three different distributed projects that were transitioning to agile processes from traditional software development methodologies. The findings from her study vouch the significant role of agile coaching in successful transition to agile in GSD. However, it has been observed that getting all the sites involved in the process change is much more effective than just changing one of the
sites. The author concludes that agile coaches should be knowledgeable about the successful GSD practices in order to have an effective and successful transition. However, like any other organizational change, agile coaches also need complete support of management for a successful transition to agile in GSD. Author concludes that the process of transition to agile can be considerably improved by following certain practices such as involving the coaches in the planning phases of new projects, arranging workshops with selective members of all sites together with coaches to share common vision, conducting regular meetings with coaches, and hiring agile coaches on long term basis.

Apart from the availability of experts who know how best to apply the agile practices in GSD, appropriate technological support is also required because agile approaches may not be able to provide the potential benefits in the context of GSD as these approaches have not been designed for distributed arrangements. Hence, it is important not only to tailor the agile approaches according to the GSD context but also provide supportive tools. Dullemond et al. [37] have identified the potential benefits of agile approaches in GSD and associated challenges that need to be addressed by providing suitable technological support. To help project managers to decide the kind of technological support required for different agile approaches in the context of GSD projects, they have specifically distinguished five types of tools that can effectively be used to support agile practices in GSD. The types of tools are:

- Tools for facilitating communication;
- Technical knowledge sharing tools;
- Transparency tools to share organizational knowledge and project status,
- Quality assurance facilitators and;
- Tools for frequent builds and integration.

The authors argue that the benefits and challenges of applying agile practices to GSD projects by proposing a model demonstrates the complete overview of the relationships between the different aspects of agile manifesto and distance factors. Like many other researchers [57], their proposed model show that geographical, temporal and socio-cultural distances in GSD projects imply a number of challenges in using distributed agile developments such as difficulties in smooth communication, frequently accessing
and interacting with customers, accessing the information required to make decisions, and arranging regular meetings. However, different aspects of agile manifesto can help alleviate the gravity of the challenges of using agile in GSD; for example, having close collaboration among stakeholders, decentralized decision making, having collective sense of ownership of the work, and keeping customer involved and satisfied can significantly decrease the perceived distance factors among team members. Moreover, the authors also provide a framework which shows how to use each types of the identified tool that can help not only exploit the promised benefits of using agile in GSD arrangements, but can also alleviate the commonly reported challenges of agile in GSD projects.

Software testing is one of the key software development activities, which has been extensively and continuously researched. It is a common practice to outsource software testing to specialized vendors and teams. Outsourced Offshore Software Testing (OOST) in the context of GSD has gained an increased adoption. However, there is relatively little knowledge about the ways test engineers perform their tasks under deadline pressures, the challenges they face, and the strategies they apply to cope with those challenges. Shah and her colleagues have reported an ethnographically inspired study of three vendor testing teams [98]. Their findings provide important insights about the test engineers’ activities while working under the pressure of deadlines. According to the findings, test engineers gain knowledge about the functional requirements, assess tasks, and duration feasibility and negotiate the allocated time as a part of the test preparation phase. Authors report that the test engineers reported more pressure during the test execution phase as it was difficult to negotiate for more time and the deadlines could not be pushed. After the test execution phase, there was a client management phase, which was also called warranty period. The study reports that the test engineers felt pressure when they were frequently queried and were unable to complete the assigned tasks on time; they also felt pressure after signing off as a result of being fearful of more bugs to be detected. The factors contributing to the pressure situation were delay in code deployment from developers, sudden shortage of resources, technical issues, and delays in defect fixing. The participants reported that the factors
impeding high quality testing were information gaps, visibility issues, long and rigid communication channels, and clashing or misunderstood priorities.

The researchers also identified the strategies used for mitigating the observed challenges. The observed strategies were good communication and rapport building with clients, announcing risks in advance, and providing support to speed up defect resolution. The authors also compared the pressure situations between in-house testing and outsourced testing. Based on their findings, they presented four hypotheses, which are:

1. In-house and offshore testing differ based on degree of authority and impact of organizational settings
2. Team configurations influence degree of pressure experienced
3. Offshore quality concerns arise due to situations not people, that work offshore
4. Many models are hidden, exposing, and analyzing their impacts can help improve the off-shoring practice.

The presented hypotheses can direct further research on this topic. The authors have also described the situation of unproductive productivity according to which the test engineer produced the results (e.g., performing 500 test cases) but felt unproductive because the work performed was unnecessary and was performed just to invoice the clients who did not know whether or not that many test cases were required. We believe that this aspect of the findings should also be further explored as the findings can bring more efficiency and effectiveness to GSD in general and offshore outsourced software testing in particular.

4.4.4 Knowledge Management (KM) in GSD

Since software development is a knowledge-intensive process, having appropriate support for managing technical and non-technical knowledge can be an effective mechanism of software development governance [72]. However, lack of face-to-face communication and informal chats caused by geographical dispersion can make it difficult to have a smooth flow of information to share the context and common understanding. Furthermore, the process of capturing and transferring knowledge generated at different sites in a GDS project can be quite difficult because of the
involvement of multi-culture, multi-stakeholders, multi-languages and multi-processes. Knowledge management has been a well-researched topic in software engineering. Knowledge management refers to the process of creation, capture, transform, deploy and applying knowledge [95]. Knowledge sharing is at the heart of knowledge management process, which can be assumed as part of knowledge management life cycle from creation to transformation. In the context of software engineering, one of the noticeable discussions in KM (Knowledge Management) is about Software Architecture Knowledge which is defined as Architectural Design + Architectural Design Decisions [60] generated during the process of software architecture life cycle. Nonaka states that knowledge is created during interactions between individuals and converted between tacit and explicit [81]. Considering this definition, we can distinguish between tacit architectural knowledge as experiences, skills, goals, constraints, assumptions, and concerns, and explicit architectural knowledge as patterns, reference architectures, standards, requirements, design decisions, design rationales, views, and models [8] that are complementary to each other.

Software architecture results from a set of interactions among stakeholders, negotiations and knowledge building activities. The necessity of managing the associated knowledge with the architecture has become crucial topic in context of global software engineering due to the barriers and challenges caused by distance factors. Hence, there have been several efforts to identify the Architectural Knowledge Management (AKM) practices and their relationship with GSD challenges [29],[28],[15]. Manteli et al. [72] have studied the impact of different governance strategies on architectural knowledge management (e.g. knowledge about architecture design decision) of multi-site development.

They have identified three main categories of knowledge management challenges as knowledge creation, knowledge transfer and communication and distinguished between system generic knowledge as information regarding entire system and the final product versus unit specific knowledge as information regarding individuals, available competencies and resources. Their work aimed at studying the impact of governance strategies on specified knowledge management categories from three angles: business strategy, the relationship between sites, team structure and role compositions as well as
work distribution and task allocation strategies. The study revealed strategic barriers between sites due to organizational policies and limitations in having access to the sources of information and sharing the filtered information with remote sites would increase the need of frequent communication among sites for seeking information and clarifying the ambiguities. Moreover, hierarchical structures, variety of role descriptions and unbalanced team sizes have been observed as issues that can cause KM challenges in form of bottlenecks, difficulties in locating source of knowledge as well as “sticky” knowledge to the locations where the majority of competences are located. The strategies in task allocations and separating tightly coupled activities across sites, not only increase the need of frequent communication, but also imply the need of having richer documentation to share knowledge within sites. On the other hand, allocating loosely coupled tasks to independent sites can also cause the phenomenon of “sticky” knowledge in absence of appropriate strategies for smooth communication and sharing knowledge.

Clerc et al [28] define a framework for GSD challenges based on literature review and empirical investigations in two organizations. They specifically focused on the use of architectural rules in comparison with joint team efforts for that purpose. The authors define architectural rules as the principles and statements on software architecture that must be applied within an organization. The study demonstrates that architectural rules such as providing Yellow-pages, dedicated websites, alignment of architecture with organizational structure and organizing configuration management tools can be useful to overcome the issues such as software development processes, communication, information exchange, team building and collective ownership. However, authors acknowledge that though the architectural rules are expected to be valuable to overcome certain GSD challenges, they cannot be help overcome certain kinds of GSD challenges such as cultural issues and human factors. That is why authors recommend combining the rules with a set of lightweight practices based on joint team efforts. The authors carried out another study [29] to investigate the perceived usefulness of the AKM practices in GSD in order to validate the practices reported in [27] and to find out any logical relationship between the number of sites and the popularity of the adopted practices. Their study demonstrates the general perceived usefulness of all the AKM
practices; however, the personalization strategies, promoting interactions among knowledge workers are perceived to be more useful compared with codification strategies that focus on codifying and storing knowledge in repositories. Furthermore, it was found that some practices are more popular in 3 sites projects such as visiting remote locations or sending representatives to remote sites. According to authors, this difference might be a result of evolving the two-site projects to three-sites ones.

Following the efforts for defining the usefulness of AKM practices in the context of GSD, Beecham et al [15] have identified a set of key AKM practices and their implementation within a proposed “Global Teaming Model” (GTM). The authors argue that GTM addresses the GSD related goals and sub practices which are not covered in CMMI model and can affect management of globally distributed teams. They have realized three main categories of AKM challenges addressed within literature as the challenges regarding alignment of organizational structure and architecture, the issues regarding creation and transmission of AK [27] as well as difficulties and importance of providing appropriate infrastructure for AKM. According to them, AKM would fit into one specific sub practices of GTM, which describes collaboration between locations; they have supported their claim with relevant literature.

Boden et al. [16] have investigated the concept of KM in small distributed software development teams from the angle of culture. They argue that national, organizational and professional cultures can impact the process of knowledge creation and sharing among team members. Their analysis of the KM practices (such as status meetings to maintain awareness, sharing artifacts and repositories, spending time at other sites, and mediation between people and culture) at the studied SMEs revealed that cultural and social issues could be influence the way knowledge is exchanged. Weekly meetings and circulating minutes of meetings versus daily Skype sessions for status review and coordination of activities between managers; different approaches to share artifacts and writing specifications, relying on extensive documentation as means of communication versus tendency to have more self-explanatory code and frequent communication with the customer can be seen as examples of diverse approaches that distributed teams choose due to social as well as organizational culture for exchanging information. Their findings demonstrate that high social capital - having network ties of goodwill, shared
norms, and shared language - facilitate knowledge sharing, however, it can also negatively impact cooperation among distributed teams as people desire to communicate with the ones who are coming with closer culture and more common understandings. Like other studies’ findings, this study also concludes that visiting other sites and spending time with remote colleagues can decrease the cultural gap and make communication easy. Knowledge brokers who are having notable technical and domain knowledge can significantly improve the process of knowledge sharing; however, it should be taken into consideration that only relying on the performance of these key roles and overloading them may make them bottlenecks. These findings demonstrate that SMEs are more likely to know (who does what, who works with whom etc.) in practice rather than following standardized procedures, which shows the phenomenon of second wave of knowledge management (transition to more tacit and emergent aspects of knowledge and supporting informal knowledge sharing).
5 Discussion and Conclusions

Global software development paradigm has gained significant popularity over the last two decades as it promises several benefits such as extending working hours and/or teams with the required set of skills, taking advantage of low cost destinations, and being closer to a significant market. However, the temporal, geographical, cultural, organizational, and knowledge distances that characterize GSD projects usually result in several kinds of challenges that need to be fully understood and addressed in order to materialize the promised benefits of GSD [23, 38, 57]. GSD challenges can commonly be categorized into different research areas such as communication, coordination, collaboration, awareness, trust, and linguistic, culture, and domain knowledge. Researchers have also suggested framework [3] to organize and understand GSD challenges and benefits, and proposed research agenda [96] that highlights the most important areas to be researched by the GSE community. Since the literature on GSD has been growing quite fast, it is important to regularly take a stock of the published literature. The main motivation of this work was to provide an up-to-date review of the literature on the key research trends, GSD challenges, and potential solutions that have been reported during the last five years in the main conference on GSD, International Conference on Global Software Engineering (ICGSE). We also expect this review to identify the research gaps that need to be filled by the future research efforts. The motivation and objectives of this work were transformed into three key research questions to be answered through this review.

1. What are the main research trends of GSE reported in the main venue of this community over the last five years?

2. How much has the reported research on GSE been influenced by the practice-driven GSD research agenda outlining research gaps in the area?

3. What are the main challenges and potential solutions reported in the recent literature on GSE through the main venue of this community? (Solutions can be in the form of GSD reference models, processes, practices, and tools to address the reported challenges).
In this section, we summarize and discuss the findings from the review with respect to the research questions to be answered and draw a few conclusions that can direct a reader to the areas that are expected to provide interesting questions for the future GSD research. We start the summary of the review’s findings by providing the key statistics about the reviewed papers for each of the five years from which the papers were selected and for each of the categories that have been described and justified for appropriateness in Section 4 (i.e., communication, coordination, collaboration, and governance).

Table 2 presents the distribution of the number of published research papers over the number of years, i.e., 2007 – 2011 from which the reviewed papers were selected. Table 2 shows that the total number of research papers reported in the 5 years has been 97. For the report, we decided to select and review 50 out of 97 research papers by choosing 10 random research papers from each year. The statistics showed in Table 2 indicate our intention to provide an overview of the number of the contributions for each of the reviewed year while covering reasonable number of published papers.

<table>
<thead>
<tr>
<th>Year</th>
<th>Total number of Research papers</th>
</tr>
</thead>
<tbody>
<tr>
<td>2007</td>
<td>22</td>
</tr>
<tr>
<td>2008</td>
<td>17</td>
</tr>
<tr>
<td>2009</td>
<td>20</td>
</tr>
<tr>
<td>2010</td>
<td>22</td>
</tr>
<tr>
<td>2011</td>
<td>16</td>
</tr>
<tr>
<td>Total</td>
<td>97</td>
</tr>
</tbody>
</table>

In order to answer our first research question, “What are the main research trends of GSE reported in the main venue of this community over the last five years?” we have provided an overview of the papers that are placed in different categories, i.e. communication, coordination, collaboration, and software development governance and its sub-categories in Table 3.
We should make it clear that there were many papers, which addressed the themes, or challenges that overlapped more than one or two categories. However, we decided to place one paper in only one category to minimize ambiguity and manage the cognitive load of a reader of this report. For this reason, we decided to place a paper in a category on which that particular paper appeared to have mainly focused. For this classification, both of the researchers sought agreement about the main focus of each of the reviewed papers before placing the papers in different categories. Whilst the placement of the papers targeting communication, coordination, and collaboration was relatively easy, it was quite challenging to decide about the placement of the papers in the sub-categories of the software development governance in GSD. The following paragraphs indicate which papers fall under which categories in order to help easily read Table 3.

**Communication:** We have placed those papers in this category, which are specifically investigating communication matters in GSD from the perspective of the role and choice of communication media [35, 78-80] to browsing into the communication patterns [4, 76], and associated challenges and solutions [10, 30, 82]. Hence, there were 9 papers which mainly focused on communication related issues and solutions.

**Coordination:** There are only a few studies, which are mainly targeting coordination related issues. The three papers that mainly focused on coordination covered the research issues such as challenges and roots of coordination breakdown in GSD [24], provision of solutions in form of practices [17] and coordination index [100].

**Collaboration:** It has been already said software engineering is a collaboration-centric activity, which gets negative impact as a result of distribution factors. GSD literature reports several kinds of challenges, which are directly or indirectly, related to collaboration among GSD project team members. Our review included six papers (i.e., [86], [14], [11], [71], [97], [25]), which mainly focus on different aspects of the collaborative challenges and potential solutions in GSD.

**GSD Models and Processes:** This sub-category contains all the papers proposing reference models for successfully conducting GSD [70, 93], business strategies [92], process descriptions [46] and the issues associated with software development processes such as requirement engineering [69], [90], design [25] and software architecture evaluation [12].
**Project Management in GSD:** This sub-category includes all the papers discussing the topics for managing GSD projects such as conducting Scrum practices [9, 33, 88-89], risk management strategies [64, 75] and relevant models provided for task allocation [65-66], effort estimation [62] and process description.

Table 3: Distribution of primary studies based on the themes

<table>
<thead>
<tr>
<th>Themes</th>
<th>Number of primary studies</th>
<th>Primary studies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication</td>
<td>9</td>
<td>[78], [30], [79], [82], [35], [80],[10],[76], [4]</td>
</tr>
<tr>
<td>Coordination</td>
<td>3</td>
<td>[100], [24], [17]</td>
</tr>
<tr>
<td>Collaboration</td>
<td>6</td>
<td>[86], [14], [11], [71], [97], [25]</td>
</tr>
<tr>
<td>Software Development Governance in GSD [36]</td>
<td>GSD Models and Processes</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Project Management in GSD</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>General GSD Practices</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Knowledge Management (KM) in GSD</td>
<td>5</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>50</td>
<td></td>
</tr>
</tbody>
</table>

**General GSD Practices:** Under this sub-category, we have covered general GSD practices reported in the reviewed literature including the challenges and benefits of applying agile approaches in GSD [37, 87], the guidance for successful product transfer [99], and more generic topics such as the suitability of the roles in distributed arrangements [67], the interdependency of distribution dimensions [44], and the challenges and successful practices associated with distributed testing [98].

**Knowledge Management (KM) in GSD:** According to our review, the papers that are classified under this category either target architectural knowledge management
specifically [29], [15], [28] or discuss the impact of different factors and circumstances associated with distributed development such as organizational decisions [72] and culture [16] on managing knowledge.

![Distribution of primary studies](image)

**Figure 4: Distribution of primary studies based on themes**

Figure 4 shows that distribution of the reviewed papers according to the four high level categories that we have used for analyzing the work reported in this paper. Figure 4 shows that Software Development Governance in GSD (and its sub-categories: GSD Models and Practices, Project Management in GSD, General GSD Practices and Knowledge Management in GSD) have been extensively researched during the last 5 years as more than half of the papers were placed in this category. Other categories have considerably less number of papers. For example, the coordination category has only 3 papers.

Although we have done the classification of the primary studies based on the main focus of each paper and the acknowledge that there may be significant overlaps between the themes, the distribution shows that the main focus of the studies in this community has been on the fourth category which we refer as Software Development Governance in GSD.

We have addressed our second research question, “How much has the reported research on GSE been influenced by the practice-driven GSD research agenda identifying
**research gaps in the area?**” by discovering the research that might have focused on the areas which have previously been identified as important research area by Sengupta et al. [96] who have proposed a practice-driven GSD research agenda. In the following paragraphs, we discuss one review’s findings with respect to the four research areas identified in [96].

One of the GSD areas identified for future research was about collaborative software development tools. Our review has revealed that there has not been much research reported in this area in ICGSE between 2006 and 2011. One of the mentionable work has been reported by Cataldo et al. [25] in the form of a tool for collaborative design in GSD. Their tool is called “CAMEL” that provides several kinds of features to support distributed design teams as has been reported in the findings from the review under the *collaboration* category. Apart from the features to support collaborative design, the tool provides several other features to support collaboration (such as information sharing, conflict resolution, developing shared understanding, and managing focus during discussions) for highly synchronous interactions among the involved parties and improve the effectiveness of design meetings with geographically dispersed stakeholders.

Collaborative planning for requirements engineering activities has been supported by Laurent and colleagues [69], who have provided a modeling language and visual notation along with an associated tool to support distributed stakeholders in the RE process. This kind of tool can help to support knowledge sharing in the process by classifying artifacts as stationary/ traveling, defining master role types, defining and storing the relationships between sites, artifacts and roles. Their tool can also generate the underlying graph of the identified interconnections among stakeholders to enable managers to make proactive decisions about needed communication infrastructure and collaborative processes between sites as well as the possibility of detecting bottle-necks by analyzing the graph and monitoring the state and movements of traveling artifacts. In a similar line of research, Prause et al. [90] have shown how distributed requirement engineering activities can be supported by combining a bug tracking system, Jira, and a requirement management methodology called Volere. However, none of the collaborative support for RE specifically highlights how it addresses the RE challenges in GSD outlined by Damian and Zowghi in [32].
Hence, we can conclude that though there have been a few efforts to provide collaborative tools as per suggested by Sengupta et al. [96], there needs to be significant amount of effort required to identify, understand, and support the collaborative activities in different phases of software development in the distributed arrangements. For example, from collaborative support point of view, there is an important need of revisiting the RE challenges in GSD that were identified almost 10 years ago in [32]. Researchers also need to analyze the literature to help reveal as to which challenges have been addressed by the solutions proposed over the last 10 years and which challenges have yet to be addressed and/or which new challenges have been surfaced during this period.

Software engineering literature is awash with papers on the role of and approaches to managing knowledge for supporting different activities [7, 34]. GSD brings new challenges to sharing and managing knowledge as a result of distribution factors. Sengupta and colleagues also have identified the need of significant research in the area of application knowledge migration and management [96]. Our review has shown that there has been an increased interest in knowledge management since the presentation of the Sengupta et al.’s work. Whilst there have been only a few papers (such as [29], [16], [72], [15], [28]) that have directly investigated knowledge management in GSD from different angles (such as architectural knowledge management, role of governance in knowledge management, and organizational decisions and culture on managing associated knowledge), the topic has also been partially covered in other studies. We have observed that knowledge sharing (managing who knows what, whom to contact, how to contact) and filtering information (provision of only relevant information to whom it may concern) have been addressed through the provision of collaborative tools [25], [11], [86]. There has also been an increased emphasis on the role of social media based communication technologies such as IM, Wikis, blogs, and discussion forums in sharing common vision as well as creating and developing tacit knowledge through ad-hoc and informal discussions. IM provides immediate means of synchronous text-based communication and the possibility of storing and referring back to the chat logs as informal sources of knowledge. Cataldo et al. [25] make use of these facilities and provide the chatting option in their collaborative design tool, CAMEL, to integrate explicit knowledge (e.g., design models) with tacit knowledge (e.g., discussions between distributed stakeholders to make design decisions). Researchers have also made some
recommendations about the ways to address the issue of knowledge transfer and management in the context of transferring product development to remote sites [34] where domain and technical knowledge need to be transferred along with the product development. However, there has been no sophisticated tool reported for systematically acquiring and transferring knowledge from one site to another. The most commonly used means of transferring domain knowledge include transferring people to remote sites for some time, ensuring updated product documentation, finding right resources with the required competencies for receiving knowledge [99], arranging tailored training, co-located analysis phase and workshops [14]. From these findings, we can conclude that there has been an increased attention paid to knowledge transfer and management, however, there are still several GSD research challenges to be addressed related to knowledge sharing and management in GSD projects. For example, automating the identification and synthesis of knowledge dispersed in different tools and applications, and contextual provision of such knowledge is an important area to be researched. There is also an important need of a metric suite and associated tool for justifying the resources required for knowledge acquisition and utilization in GSD.

Testing has been one of the most outsourced activities, however, there has been not much attention paid to the challenges faced by globally distributed software-testing teams. Sengupta et al. also identified the distributed software testing as an important research area. The findings from this review have revealed that software testing process has not been appropriately addressed as out of 50 reviewed papers, there is only one paper [98] that has reported research on software testing practices. However, the reported research focuses on the socio-cultural aspects of software testing in distributed environments and working under pressure rather than focusing on the process of software testing and more technical issues such as the potential challenges in unit and integration testing with globally distributed teams. There were also a few papers, which were not specifically targeted at reporting challenge or solutions in the area of distributed software testing, but they covered some aspects of software testing in distributed teams. For example, Avram studied knowledge management to support collaboration in the context of software testing with distributed teams [11]. Her study also shows that improving interpersonal relationship between teams and building trust can increase the level of awareness and acceptance of the changes that can reduce frustrations. Cataldo et al. [24] highlight the
importance of identifying and managing dependencies between different modules developed at different sites in order to avoid coordination breakdowns and integration errors. Damian et al. explore the impact of communication breakdowns and lack of awareness on broken builds [30]. The study shows that various working habits in communicating the changes (e.g. via email and build reports) due to different organizational culture can cause misalignments and misunderstanding about the changes and lead to serious errors at integration stage. Hence, it can be concluded that this area has not attracted any mentionable attention since the publication of the research agenda paper. And it is also clear that none of the reviewed papers addresses the critical issues of software testing in GSD of data privacy and its impact on unit testing. Therefore, there appears to be an urgent need of studying and addressing the challenges of software testing with distributed teams, especially the issues related to privacy and security of the data. We also assert that researchers should also look into the viability of cloud-computing based solutions for supporting the software testing in the context of GSD projects.

The fourth area of potential GSD research identified was process and metric issues. Software development processes play an important role and the well-defined processes in GSD have significant importance. Project management and process improvement efforts require appropriate metric suites that can help decision making with quantifiable data. That was why Sengupta et al. [96] emphasized the need of scaling the conventional process frameworks like CMMI and developing appropriate metric suites for GSD situations. We found that GSD researchers have paid significant attention to this area, as there were several papers on different aspects of reference models for processes, project management, and GSD practices. For example, Prikladniki et al. [93] have helped to understand the evolution through which a distributed software development initiative can be expected to go through by providing a capability maturity model for distributed software development. Whilst the suitability of the proposed model has yet to be fully evaluated, it can provide an initial foundation for addressing the lack of GSD specific CMM model. Along the similar lines, Lings et al. [70] proposed a reference model for successful distributed software development. Their reference model is expected to help a project manager to successfully manage GSD projects. However, the reference model is quite descriptive in terms of a set of practices to be followed in order to be successful in GSD projects and it lacks quantifiable metrics.
Apart from the reference models for GSD processes and practices, this review has revealed that one of the most notable areas of research is the application of agile approaches to GSD environments. Since agile approaches have been designed for collocated arrangements, there can be several potential challenges in applying agile approaches to GSD projects. Many papers have indicated the potential challenges and proposed solutions. For example, Dullemond et al. [37] have highlighted the benefits of using agile approaches but have also identified the potential challenges for which they propose to use technological support. We have also noticed that there has been quite high amount of effort allocated to support different aspects of GSD projects such as decision-making processes in GSD, task allocation, and effort estimation in GSD projects. For example, Lamersdorf et al. [61] have provided a decision support system for task allocation in distributed arrangements based on four elements (i.e., Goal, Task, Resource and Mapping function); Al-Ani and Redmiles [5] has empirically investigated decision making processes (DMP) in distributed software development teams based on the role of leadership, culture and level of team distribution. They have provided an understanding of the types of DMPs in GSD: consultation, consensus and autocratic. Whilst each of these papers addresses different decision making problem in GSD, both of them confirm the need of in-depth studies and tool support for decision-making in distributed arrangements. Hence, we can conclude that there is a research gap to be filled in terms of providing in-depth understanding about the decision making in distributed arrangements and supporting it with appropriate tools which have been built upon solid theoretical basis.

Along the similar lines, Lamersdorf et al. [62] have also proposed an effort estimation model based on the impact of various factors such as task criticality, task complexity, formality of task description, process maturity, requirement stability, level of task coupling on productivity of the teams and quantitatively estimating the effort overhead based on the identified drivers. The authors have also proposed a risk-driven model for work allocation in GSD [65] based on their effort estimation model. Their risk-driven work allocation is characterized based on three dimensions namely risks, influential factors and logical rules. Mudumba and Lee [75] also provide a framework for risk mitigation when applying agile in GSD.

From these contributions and given the fact that there were 22 papers out of 50 that fall under the software development governance category that represents the process and
metric issues area, it is clear that this area has attracted significant amount of research efforts. Considering the three main approaches indicated by Sengupta et al. [96] for providing metrics for GSD projects and the findings from our review, we can conclude that despite a reasonably good amount of work reported on defining effort estimation, risk factors, and risk mitigation strategies in the context of GSD, there is still a lot of room to provide sufficient support for clarifying the processes, roles and responsibilities more systematically and appropriate process descriptions. However, none of the reviewed papers addresses the issues surrounding the development of indicators that reflect client expectations and provide information to show the impact of each participating organization on a project value chain. Regarding the learning from the history of the past projects in order to provide estimation for pricing and effective work distribution, this review has revealed that Lamersdorf et al. [65], [62] have tried to fill the identified research gap in terms of risk-driven work allocation model integrated with an effort estimation model based on logical rules. We assert that GSD researchers should join these efforts along with the researchers with expertise in mining software repositories in order to define new metrics and gather data to provide interesting test-bed for experimenting with different kinds of scenarios that characterize GSD.

Our third research question, “What are the main challenges and potential solutions reported in the recent literature on GSE through the main venue of this community? (Solutions can be in the form of GSD reference models, processes, practices, and tools to address the reported challenges)”, has mainly been addressed in Section 4 where we have provided an elaborative summaries from the analytical reviews of the studied papers. In addition to the main challenges and potential solutions such as the impact of distance factors on coordination breakdowns, knowledge management issues, and the difficulties in collaborative activities among team members, one of the most frequently reported challenges and solutions are related to communication in GSD teams. The so-called distances of GSD (i.e., geographical, cultural, and temporal) make it quite challenging to have effective and efficient communication (formal or informal) in GSD teams compared to co-located teams. To overcome the communication related challenges, software development teams have started using web-based technologies both relatively traditional (such as Instant messaging) as well as contemporary social networking (such as Skype and Google+) in new and innovative ways [11, 35, 78-79]. For example, IM has
become one of the most commonly used media of initiating and having informal discussions to resolve non-complex issues between remote co-workers. The use of text-based communication media is also preferred to voice-based one as it provides users with time and opportunity to formulate a message in more unambiguous manner. IM has also been reported to play a significant role in creating and maintaining contextual and technical awareness among GSD teams. Chat logs saved from IM also serve as means of explicating and sharing tacit knowledge as previous mentioned.

Despite all beneficial uses, IM has also been reported to cause frustration as a result of inappropriate use of it and/or lack of protocols about its use at workplaces. For example, incorrect status [35] and over-use of chatting and notifications of new requests for unnecessary issues can cause distraction and frustration with negative impact on productivity as a result of information overload. Similarly, the use of email has increased many folds as a reliable medium of asynchronous and unambiguous communication that can help minimize misunderstandings as it provides several features to follow different threads of communication on the same or similar topics. However, an increased number of emails can overload a receiver who may start overlooking even the emails which need urgent actions that can lead to serious coordination issues [30]. One area where research needs to focus is the ways of promoting effective and efficient use of IM and emails based on quantifiable mechanisms of measuring the benefits and negative effects of this medium with different levels of frequency of their use and the categorization of the issues to be addressed through these media. Moreover, there also needs of systems that can provide contextual information about co-workers on IM without impacting users’ privacy and security.

Given the increased popularity and use of social networking applications (such as discussion forums, blogs, and micro blogs), organizations and individuals have also started using these applications for communication and collaboration to share knowledge and support contextual and technical awareness [11]. Our review has revealed that social networking applications are being used to support software development activities of dispersed team members and address different kinds of GSD challenges. However, these are very early days of using social networking applications to address the challenges caused by GSD distances. There are several kinds of technical and socio-technical issues that need
to be fully understood and addressed in order to exploit the full potential of these applications to support GSD teams by improving interpersonal relationship between team members, developing the sense of team-ness, and increasing the overall team spirit and work satisfaction.

The presented discussion and drawn conclusions from analytically reviewing a selected set of research papers have revealed that the GSD researchers have been quite active in providing knowledge and understanding about the challenges and potential solutions to help organizations to successfully adopt and evolve GSD paradigm for their projects of all sizes and nature. There is an increasing trend of proposing appropriate solutions along with empirical evidence to the challenges previously reported or observed by the solution providers themselves. That means researchers are not only merely focusing on studying and reporting challenges, benefits, and threats of GSD projects, rather, they are predominantly focusing on studying the GSD projects and arrangements with the aim of identifying interesting and challenging research problems and devising and trying new and innovative solutions to address the identified challenges. This two-pronged approach to directing the GSD research is resulting in not only improved understanding about the challenges and issues involved in adopting and sustaining GSD paradigm for building a body of knowledge but is also providing new solutions and tools to the repertoire of different stakeholders to be more skilled and knowledgeable to deal with the challenges in different process areas (i.e., communication, coordination, collaboration, and software development governance) caused by the GSD distances, i.e., geographical, temporal, cultural, organizational/processes, and knowledge. This review has also revealed, as discussed in this section, that there has not much work done in the areas considered most important according to a practice-driven research agenda for GSD by Sengupta et al. [95]. We have reviewed a relatively small number of papers from only one potential, albeit the main one, venue for publishing research on GSD. Hence, our findings should be read and interpreted by keeping the limitations and scope of the reported review in mind.

This review has also enabled us to critically analyze the Ågerfalk and Fitzgerald [3]’s framework for organizing the GSD opportunities and threats. Based on the reviewed papers and our observations from the general GSD literature, we assert that the framework needs to be extended, as it may not sufficiently serve the needs of the current trends of GSD and the recently reported opportunities, challenges, and solutions of GSD. Hence, we have
proposed that the framework needs to be extended not only in terms of distance factors but also in terms of process elements. Our proposal is to extend the framework with two distance factors, i.e., organizational/processes and knowledge and with two process elements, collaboration and software development governance, which can be further refined into four sub-categories, i.e., GSD models and processes, project management in GSD, general GSD practices, and knowledge management in GSD. We also suggest that the future GSD research efforts be also dedicated to provide solid theoretical foundations for extending the framework in the ways proposed in this report and systematically justifying the proposed extension dimensions. We assert that an extended framework will enable GSD researchers to more easily organize and classify GSD opportunities, challenges, and solutions by extracting and organizing the data from a large set of research papers on GSD in order to provide guidance about the areas where more research is needed to support the GSD practice. We expect that this report will contribute to the efforts geared towards helping practitioners to become aware of the state-of-the-art in GSD and researchers to identify the potential areas of GSD research.
Acknowledgement

This research has been funded by the Danish Agency for Science, Technology and Innovation under the project “Next Generation Technology for Global Software Development”, #10-092313.

References


