<http://www.diva-portal.org>

Postprint

This is the accepted version of a paper published in *Information and Software Technology*. This paper has been peer-reviewed but does not include the final publisher proof-corrections or journal pagination.

Citation for the original published paper (version of record):

Jabangwe, R., Šmite, D., Hessbo, E. (2015)

Distributed Software Development in an Offshore Outsourcing Project: A Case Study of Source Code Evolution and Quality: Distributed Software Development in an Offshore Outsourcing Project.

Information and Software Technology

<http://dx.doi.org/10.1016/j.infsof.2015.12.005>

Access to the published version may require subscription.

N.B. When citing this work, cite the original published paper.

Permanent link to this version:

<http://urn.kb.se/resolve?urn=urn:nbn:se:bth-11449>

Distributed Software Development in an Offshore Outsourcing Project: A Case Study of Source Code Evolution and Quality

Ronald Jabangwe · Darja Šmite · Emil Hessbo

the date of receipt and acceptance should be inserted later

Abstract *Context:* Offshore outsourcing collaborations can result in distributed development, which has been linked to quality-related concerns. However, there are few studies that focus on the implication of distributed development on quality, and they report inconsistent findings using different proxies for quality. Thus, there is a need for more studies, as well as to identify useful proxies for certain distributed contexts. The presented empirical study was performed in a context that involved offshore outsourcing vendors in a multisite distributed development setting.

Objective: The aim of the study is to investigate how quality changes during evolution in a distributed development environment that incurs organizational changes in terms of number of companies involved.

Method: A case study approach is followed in the investigation. Only post-release defects are used as a proxy for external quality due to unreliable defect data found pre-release such as those reported during integration. Focus group meetings were also held with practitioners.

Results: The results suggest that practices that can be grouped into product, people, and process categories can help ensure post-release quality. However, post-release defects are insufficient for showing a conclusive impact on quality of the development setting. This is because the development teams worked independently as isolated distributed teams, and integration defects would help to better reflect on the impact on quality of the development setting.

Conclusions: The mitigation practices identified can be useful information to practitioners that are planning to engage in similar globally distributed development projects. Finally, it is important to take into consideration the arrangement of distributed development teams in global projects, and to use the context to iden-

Blekinge Institute of Technology,
SE-371 79 Karlskrona, Sweden
Tel.: +46 455 38 50 00
Fax: +46 455 38 50 57

Ronald Jabangwe E-mail: ronald.jabangwe@bth.se ·
Darja Šmite E-mail: darja.smite@bth.se ·
Emil Hessbo E-mail: emil.hessbo@gmail.com

tify appropriate proxies for quality in order to draw correct conclusions about the implications of the context. This would help with providing practitioners with well-founded findings about the impact on quality of globally distributed development settings.

Keywords Offshore Outsourcing · Distributed Development · Global Software Development · Software Quality · Case Study

1 Introduction

Offshore outsourcing is a sourcing strategy in which a company contracts part or all of the development or maintenance activities to an external company that is located in another country (an offshore vendor) (Šmite et al., 2014; Oshri et al., 2011; Pfannenstien and Tsai, 2004). Such development environments often result in nontrivial, difficult to manage distributed development collaborations (Colomo-Palacios et al., 2014; Šmite et al., 2014), much more demanding than those formed by outsourcing within the same national boundaries (Rottman and Lacity, 2006; Oshri et al., 2011). What is perhaps more worrisome is that distributed development is infamous for failures, including failures of delivering quality software (Moe et al., 2014; Dibbern et al., 2008).

Almost half of the projects in industry are said to fail to realize benefits, like cost savings, because of inadequate planning of, for example, knowledge transfer and communication between onshore and offshore sites, and not taking into consideration the complexities that come with offshore outsourcing ventures (Fabriek et al., 2008; Iacovou and Nakatsu, 2008). Such statistics are troubling given the astronomical amount of money, which range in the billions of US-dollars, that has been associated with offshore outsourcing projects over the last decade (Schaaf, 2004; Lacity et al., 2008; Oshri et al., 2011). There are also indications that distributed software development affects quality (Bird and Nagappan, 2012), though the number of research studies, especially based on in-depth analyses with an empirical focus, is scarce (Marques et al., 2012), and reported results are contradicting. Thus, there is a need for more empirical studies on distributed development (da Silva et al., 2010; Marques et al., 2012) as well as to better understand the possible source of contradicting results reported in literature. This will help with providing better support to practitioners with well-founded evidence that they can use to make informed decisions about current or future globally distributed projects.

The studied case can be characterized as a distributed outsourcing project with development teams located in separate countries (offshore outsourcing). The context consists of multiple locations involved in development activities, which is a characteristic that is uncommon in studies on global projects (Bilal Raza and Clear, 2013). The number of vendors involved in development activities also changes whilst the product evolves. This makes it an interesting case for understanding how quality changes in such environments. An evolutionary view of quality is used to help with the investigation. The study reveals that the case company successfully ensured post-release quality, and the key practices that contributed to the success are presented. However, because of lack of reliable integration defects, and because the development teams at each vendor worked independently, the impact on quality of the distributed setting and organizational changes is unclear. In such a case, integration defects

would be needed to better understand the impact on quality of the development setting. The study, thus, reveals the importance of understanding the nature of the distributed development context, and using appropriate measures as quality proxies in order to derive correct conclusions about the context.

Given that there are very few empirical studies that investigate GSD contexts and their implication on quality (Nguyen-Duc et al., 2015), this study is a valuable addition within the research area. The contribution of this study can be summarized as follows:

- The study provides an empirical investigation into the links between organizational changes and quality in a distributed development environment. Despite many pessimistic views (often of speculative nature), empirical results from this paper show that it is possible to achieve good post-release software product quality when outsourcing development to two offshore vendors,
- Based on the qualitative analysis, the paper further describes practices that can help ensure post-release quality when working with offshore vendors.
- In addition, the study demonstrates that valid conclusions regarding the implications of distributed development on quality require selection of appropriate measures as proxy for quality and careful attention to the context of the distributed development environments. For example, the current investigation failed to reliably conclude whether distributed development resulted in any quality changes due to the lack of pre-release defect data traceable to the concrete vendors.

The remainder of this paper is structured as follows. Related work is described in Section 2. Section 3 begins with the research question, and then further details the studied case and context, and the research approach followed. Section 4 presents the results, discussed in Section 5. Threats to validity and limitations of the study are described in Section 6. Section 7 concludes the paper with the summary of the major findings and an outline of future work.

2 Related Work

2.1 Offshore Outsourcing and Distributed Development

Offshore outsourcing is one of various sourcing scenarios, often studied under the research umbrella of global software development (GSD) (Verner et al., 2012; Šmite et al., 2014). However, studies on their effect on quality is scarce (Verner et al., 2012). Studies on offshore outsourcing collaborations generally focus on vendor selection, e.g., (Khan et al., 2009), while some studies focus specifically on the relationships between the outsourcing company and the vendor (Ali and Khan, 2014; Alsudairi and Dwivedi, 2010).

Risks and success factors are well documented in GSD literature as shown in various reviews of empirical studies, e.g., (Verner et al., 2012), (Gomes and Marczak, 2012) and (Nurdiani et al., 2011). The areas of concern when engaging in offshore work include communication and coordination breakdown (Damian et al., 2007; Iacovou and Nakatsu, 2008), vendor competency and attrition (Moe et al., 2014), cultural differences (Krishna et al., 2004; Winkler et al., 2008), and building and maintaining trust and rapport between sites (Babar et al., 2007; Moe and Šmite, 2008; Jalali et al., 2010). These areas are also said to pose a great risk in achieving desired quality goals (Kannabiran and Sankaran, 2011).

Offshore outsourcing collaborations can result in distribution of teams or team members (Šmite, 2014). Quality concerns emerge even stronger in such development settings in comparison to single-site or colocated development teams (Carmel and Tjia, 2005). This is because large geographical and temporal distance can complicate collaborations and hinder effective communication (Herbsleb and Mockus, 2003). Because of this issue, and many other challenges faced in distributed development projects (da Silva et al., 2010), it is important to take note of the possible effects on quality of the development settings.

A number of studies link distribution with a decrease in quality, for example, (Bird and Nagappan, 2012), (Ramasubbu and Balan, 2007) and (Cataldo and Herbsleb, 2011). Bird and Nagappan (2012) have studied two open source systems, Eclipse and Firefox, and found that the number of pre-release defects can increase as a consequence of developing components distributedly. Ramasubbu and Balan (2007) have also linked distributed development with a decrease in quality based on an analysis of customer reported issues during acceptance testing at a large multinational corporation working with India and US. Similarly, Cataldo and Herbsleb (2011) found that the possibility of integration failures was higher for geographically distributed feature teams than in a setting with colocated teams.

There are, however, other studies that report on opposing results in terms of the link between distribution and software quality. Spinellis (2006), who assessed coding style and defects, found negligible effect on quality associated with geographical dispersion. Similarly, Bird et al. (2009) analyzed post-release defects and internal source code measures, i.e., cyclomatic complexity and code churn measures, and found little difference in quality when comparing colocated and distributed teams working on Windows Vista. However, the distribution here was mainly within the same country. Sutherland et al. (2009) found no observable negative effect on quality when shifting from colocated to distributed development. In that study by Sutherland et al. the distributed development setting consisted of cross-functional scrum teams, and each team had team members that were dispersed across multiple locations.

The differences in research results can be explored by the variation in the context in which the studies are carried out, and the data used to measure quality. Studies discussed above contain teams distributed in relative proximity - different buildings or cities, as studied by Bird et al. (2009) versus large geographic separation as studied by, e.g., Ramasubbu and Balan (2007). Also, different number of sites involved in distributed development has been studied. Notably, Spinellis (2006) and Bird and Nagappan (2012) studied open source systems, which is a special case of distributed development. The implications of open source research results for commercial development is, however, often debated. Finally, since most of the studies focus on a single proxy for software quality (pre-release defects, post-release defects or customer reported defects), the versatile view of quality changes, and the use of an appropriate quality proxy in certain distributed contexts, is another potential research direction.

2.2 Measuring Distribution and Quality

Related studies are summarized in Table 1. The differences in the proxies for quality are alarming. Although many studies aim at evaluating the impact of distributed development on quality, the differences in the measures used as proxies for software quality, and the differences in the GSD contexts make it hard to relate the cor-

Table 1: Summary of Studies on Distributed Development and Quality

Study	Study aim in relation to distributed development and quality	Proxy for quality	Distributed development context	Findings
Bird and Nagappan (2012)	Level of distribution (in terms of geographical and organizational locations in an OSS system) and its relationship with failures	Pre- and post-release defects	OSS (FIREFOX and ECLIPSE)	Geographically distributed modules have more defects
Ramasubbu and Balan (2007)	The extent to which dispersion affects software quality	Post-release defects (acceptance tests done by customers)	Dispersed team members	Dispersion indirectly affects conformance quality
Cataldo and Herbsleb (2011)	Impact of the organization of feature teams on quality in a GSD context	Defects reported during integration	Dispersed team members	The likelihood of integration failures increases when engineers are geographically distributed
Spinellis (2006)	Effect of geographical dispersion on quality	Code style and defects (unclear if pre- or post-release)	Open-source software (FreeBSD)	Dispersion has negligible effect on quality
Bird et al. (2009)	Comparing distributed development and collocated development	Post-release failures	Dispersed team members (most of the work performed by developers in different buildings, in the same location)	Negligible difference in failures when comparing work performed in a distributed manner versus collocated work
Sutherland et al. (2009)	The effect of transitioning from collocated teams to distributed teams on quality	Pre-release defects	Dispersed team members	No observable negative effect on defect rate as a result of transitioning from collocated to distributed teams

responding conclusions. In addition, our studied case consisted of isolated teams, which is an alternative distributed development context to dispersed teams that are commonly explored in the related studies. However, be it dispersed or distributed teams, there is a clear need to better understand the usefulness of different measures of quality for GSD studies, and the way these measures are obtained and linked to GSD settings.

The organization of the development setting plays a significant role on quality (Nagappan et al., 2008). Changes in the development setting can also negatively impact quality (Jabangwe and Šmite, 2012). In addition to finding a decrease in quality from using distributed development, Mockus (2010) found that changes in the organization of development work had a negative effect on quality. Apart from a change in number of companies involved, a handover of project management responsibilities to an offshore location is another change that occurred in the present studied case. In a previous study Jabangwe et al. (2014) investigated the implications of a handover to an offshore location belonging to the same company after a long period of distributed development, and observed a negligible link of the change with quality. The study piloted comprehensive data analysis and focused on collecting quality measures, defects and internal source code measures (e.g., the complexity measure, (McCabe, 1976)). The present empirical study continues to assess quality through a number of different measures when the offshore location is an external company.

In summary, during the evolution of the product used in this present empirical study there were notable organizational changes. This involved distributed development among a customer company and two offshore vendors, a termination of collaboration with one of the vendors and a final handover of responsibilities to the remaining vendor. The study was motivated by the company’s interest in obtaining empirical evidence pertaining to how the product’s quality varied across releases, as well as understanding the implications on quality of both the distributed development setting and the organizational changes.

In the context of GSD, decisions that result in organizational changes can be traced to five decisions of why, what, how, when and where to source (Šmite et al., 2013). In other words, these organizational changes typically relate to the variations in the sourcing strategies, such as outsourcing through a relocation of development from internal personnel to external vendors (Carmel and Tjia, 2005; Oshri et al., 2011), changes in the number of vendors, vendor changes and back-sourcing (Moe et al., 2014). The organizational changes in the present paper pertain to outsourcing of responsibilities to offshore vendors and the changes in the number of offshore vendors involved in development and maintenance activities during the evolution of a software product.

3 Research Methodology

The research is driven by the following question: ***RQ:** How does software quality vary in relation to organizational changes in a distributed development environment for an evolving product?*

Quality is a versatile concept that can be viewed from different perspectives. In this study we first analyze post-release defects as a reliability indicator that is an important quality characteristic as suggested in the ISO/IEC-25010 (2010) standard. A “defect” is also used, in this study, as a generic term to refer to a fault or failure.

A fault is linked to an error that is caused by human-action, e.g., misunderstanding system requirements, which can in turn result in a system failure (ISO/IEC/IEEE-24765, 2010). In practice, we have collected defect data that includes all issues reported by the customers after the release of the product, from which we extracted only those defects that required fixes/changes in the source code. This means that issues that were not linked to a problem in the source code, such as user documentation mistakes, were excluded. Second, we analyze source code measures that have been linked to reliability indicators (Jabangwe et al., 2015). We also use commit data and the number of features added across releases, which are linked to how quality vary for evolving products, and are analyzed in the study to better understand the changes in quality. The context of the studied case is presented in Section 3.1, and the data collection and analysis procedure is described in Section 3.2.

3.1 Case Description and Context

Details regarding the studied product and the company that originated the development are first presented in this section. This is followed by details on the course of organizational changes that resulted in various forms of multisite distributed development settings.

3.1.1 Company and Product Overview

The name of the case company and the products used throughout this study will remain anonymous at the request of the company. The case company, from hereon, referred to as ZCo, was founded in 1996 and its ownership has evolved over the past few years through mergers and re-branding. ZCo develops online games. Since its creation in 1996, it has been one of the most successful and innovative companies for online/virtual gaming in Sweden. As of end of 2013 the company powered over 100 popular gaming websites. The company headquarters is located in Sweden, and the site is referred to as ZCo-SE.

The product studied is referred to as PLAY. PLAY is one of many online gaming products developed by ZCo. Though it has been developed since the 90s, the history of the product studied is from 2008. This is when the company started systematically collecting data in Subversion¹, a source code version control system, and JIRA², a defect tracking system. PLAY is marketed as a single online gaming solution, but it comes with multiple games integrated within it. The core source code components of PLAY are developed using the object-oriented programming language C++. Implementation of games is mostly done through scripting in XRS files. The latest major release for PLAY contains over two millions lines of code (LOC) in C++ and over 900 KLOC in XRS files.

3.1.2 Evolution of Work Distribution and Organization

Over the years ZCo was involved in two offshore outsourcing collaborations for developing PLAY. Prior to 2008, the headquarters site of ZCo in Sweden, ZCo-SE,

¹ Information about Subversion can be found on: <https://subversion.apache.org/>

² Information about JIRA can be found on <https://www.atlassian.com/software/jira>

contracted work to a Ukrainian site of an external company, referred to as YCo-UKR. In 2008, ZCo hired another external company located in India to take part in development activities, referred to as XCo-IN. In relation to development methodologies, Scrum-of-Scrums was followed, with isolated scrum teams that is similar to the description by Sutherland et al. (2009). Both vendors, YCo-UKR and XCo-IN, had their own scrum teams and scrum masters, who reported to the scrum master at ZCo-SE. The evolution of work distribution for PLAY can be divided into three phases based on the contracts with the vendors, which is described below.

Phase I (2008 until end of 2009): ZCo maintained two separate contracts with two outsourcing vendors. ZCo-SE was responsible for developing and maintaining the core system of PLAY, i.e., architecture for games integrated within PLAY and the platform for PLAY. YCo-UKR and XCo-IN were each primarily involved in developing and maintaining separate sets of games that were integrated into PLAY. In terms of the approximate number of employees involved in the development activities, ZCo-SE had 14, XCo-IN had 10, and YCo-UKR had approximately five. Each company had its own local team lead that reported to the team lead at ZCo-SE. Involvement of XCo-IN was terminated towards the end of 2009.

Phase II (2010): In this period, YCo-UKR was the only remaining offshore outsourcing vendor. ZCo-SE involvement was only focused on the core system of PLAY, i.e., architecture for games integrated within PLAY and the platform for PLAY. As YCo-UKR developers gained more product knowledge, their involvement spread to the core components, architecture and platform of PLAY, and full responsibility over all game development. ZCo-SE and YCo-UKR each only had one team involved in development and maintenance activities, with approximately seven and eight employees, respectively. The team lead in YCo-UKR reported to the team lead in ZCo-SE.

Phase III (2011 onwards): Project responsibilities were gradually relocated to YCo-UKR in 2011 in accordance with a contract agreement. This is referred to as a handover. YCo-UKR had one team with approximately 10 employees involved in developing and maintaining PLAY. They had their own team lead that reported to ZCo-SE. Product responsibilities, i.e., requirements analysis and release planning, remained at ZCo-SE. In relation to development activities, involvement of ZCo-SE was reduced to supporting YCo-UKR. Thus, the third phase can be characterized as single-site or collocated development.

The organizational changes during the evolution of PLAY are summarized in Figure 1.

3.2 Data Collection and Analysis

Using definitions provided by Runeson et al. (2012), the present empirical investigation is characterized as a single case study. This research approach was used because the study is done retrospectively and requires recollection of events from many years back. There is also a need to use more than one source of data so as to increase validity of the study. Collection and analysis of quantitative data are presented in this section.

Fig. 1: PLAY Evolution and Organizational Changes

3.2.1 Quantitative Data

Between 2008 and 2013 there were 132 versions of PLAY released, and this included 14 evolutionary releases and 118 maintenance releases. Evolutionary releases contained new features or new games. Maintenance releases contained primarily fixes for issues found in the evolutionary releases during both internal testing activities and those reported by customers. Quantitative data collected from the releases included: source code measures for C++ and XRS files, defect data, features added, and commit activity data. Defect data is used as an indicator of external quality. Data about features added and commit activity are also included in the quantitative data analysis because they are critical to product modification activities that result in the evolution of a product. Source code measures are included because they quantify internal properties that are linked to external quality (ISO/IEC/IEEE-24765, 2010; Jabangwe et al., 2015).

Source code measures: Two measures that quantify size and complexity properties were extracted from all releases because they have been evaluated and found to be good indicators of quality-related concerns (Jabangwe et al., 2015). LOC (lines of code), a size measure, was extracted from C++ components and XRS files. LOC is a simple count of all the executable lines of code. This measure was selected because of its simplicity and understandability. Herraiz and Hassan (2012) argues that simple size measures can reveal sufficient size related information from source code, and the information may not necessarily differ from that revealed by more complex size measures. RFC (response for a class), a complexity measure was ex-

tracted from C++ components. It is from a set of measures proposed by Chidamber and Kemerer (1994). RFC is the set of methods in a class and the set of methods that can be invoked by a method within that class (Chidamber and Kemerer, 1994; El-Emam, 2002). The complexity measure, churned LOC were computed for XRS files by summing total changed LOC and inserted total LOC for each release. High values for RFC and churned LOC are indicators of fault-prone releases (Chidamber and Kemerer, 1994; Jabangwe et al., 2015). Understand tool³, a commercial source code analysis tool, was used to extract source code measures from C++ components. Measures for XRS files were extracted using the text analysis tool Araxis Merge⁴.

Defect data: Defects collected in this study are those reported post-release. For the purpose of this study a defect can be defined as any issue that caused the system performance to deviate from how it is expected to perform according to system requirements (ISO/IEC/IEEE-24765, 2010), with the prerequisite that they required changes to the source code artifacts as a solution. This meant that non-source code related defects, such as documentation defects, were excluded. The purpose was to make sure that all data was linked to the evolution of the source code artifacts, and thus maintaining consistency of the type of data analyzed in the study.

Feature added: The number of features added to each release were collected from JIRA. The number of features added is a counting measure. In the context of this study, a feature is defined according to ISO/IEC/IEEE-24765 (2010).

Commit data: Commit activity for source code releases were extracted from Subversion. Each commit was mapped to the appropriate release. The developer that made the commit was mapped to their respective companies. Essentially, the number of commits were counted per company.

3.2.2 Analysis of Quantitative Data

The first two authors who were not involved in the development of PLAY independently performed the analysis. The goal of this analysis was to identify patterns and unusual trends in the data. These were then discussed in a joint meeting and a final list of changes and time frames of interest were noted down for discussion at the case company in focus group meetings. Statistical analysis methods and graphical representations were used in the analysis of the quantitative data.

Visualization: Analysis of commit data, source code measures, defect data and number of features added across releases was done using descriptive statistics and populating the data into graphical representations of line graphs and bar charts. Line graphs are used for source code measures, which are aggregated per release. Bar charts are used for defect data, features added data, and commit activity data, and they are both aggregated at the release level.

Defects and features were separated and visualized by their criticality levels. For defects, the range from critical to low is determined by its effect on the system performance and reliability, e.g., whether it causes system failure or it is a trivial issue with a current workaround. For features the criticality is linked to business values and it is determined by its importance to, and demand from, customers.

³ Information on the Understand tool can be found on the website: <http://www.scitools.com/>

⁴ Information on the Araxis Merge tool can be found on the website: <http://www.araxis.com/merge/index.en>

Statistical analysis: A correlation analysis was performed to investigate the relation between the defect data (the proxy for quality) and the other measures collected: size measures, complexity measures, features added, and commit data. For this analysis a simple count of number of features add per release was used. Commits were aggregated at the company level to get the number of companies involved per release as a proxy for capturing the organizational changes. The measure of the number of people that did commits was excluded from the analysis because at one of the vendors it was not representative of the individuals that actually were involved in implementation activities.

A test for normality in the quantitative data was performed using the Shapiro-Wilk test, which is a powerful normality test (Razali and Wah, 2011). This was done to narrow down the choice of appropriate statistical methods, i.e., parametric or nonparametric statistical methods. For large datasets, Razali and Wah (2011) found it to be more powerful than some of the common normality tests like Kolmogorov-Smirnov and Anderson-Darling. All statistical analyses were performed using the IBM SPSS Statistics tool version 22⁵.

3.2.3 Qualitative Data

Results of the quantitative data analysis were reviewed and discussed at the case company (ZCo-SE) in focus group meetings. A focus group meeting is an efficient method for collecting rich data because it is an open-ended discussion with a group that consists of experts or knowledgeable individuals on a particular topic (Robson, 2011). Since it is a retrospective study, focus group meetings enabled the participants to help each other recollect events and practices from the past. This in turn triggered and stimulated deeper discussions.

There were two focus group meetings held at the case company. In the first focus group meeting, there was one employee from YCo-UKR and four were from ZCo-SE. The second focus group meeting had three employees from ZCo-SE only.

In the first focus group meeting, the quantitative data analysis results were presented and discussed. The aim was to get insights and explanations pertaining to factors that contributed to changes in data over time. The visualizations of the quantitative data were presented during the focus group meetings. Each visualization contained markings of periods with unchanging data values between releases as well as increases and decreases between releases. The aim was to discuss the possible reasons for the changes and the lack of changes from the point of view of the participants of the focus group. Thus, opinions and additional context information pertaining to the history of the development were collected. Two researchers led the first focus group meeting. The first author guided the discussions, whilst the second author took notes. The second meeting was led solely by the first author, and it was mainly used to gather factors that had an influence on patterns in the quantitative data. The third author (a team lead at ZCo at the time of this study) participated in both meetings as the lead company representative.

Each meeting lasted for approximately three hours. Both meetings were recorded with the permission from the participants, as recommended by Robson (2011). The

⁵ Details about the SPSS tool can be found on: <http://www-01.ibm.com/software/analytics/spss/>

recordings enabled the authors to crosscheck the meeting notes and to verify the context in which remarks by the focus group participants were made. The participants expressed generally similar reflections and explanations for changes in quantitative data across releases.

During the focus group meetings, release notes, defect reports, and other product documentation were used to get deeper insights and explanations about the trends in the quantitative data and the evolution of the product. There were also email exchanges with the third author before and after the meetings to discuss and clarify details about the product and the quantitative data.

3.2.4 Analysis of Qualitative Data

Reflections from the focus group meetings: The approach followed in the analysis of reflections from the focus group meetings can be compared to content analysis (Robson, 2011). The reflections were examined to identify practices in the development context that were linked to patterns and trends in the quantitative data, and were implemented to alleviate quality issues during evolution. The practices were then characterized into three critical factors, product, people and process, that were proposed by Šmite and Wohlin (2011).

4 Results and Analysis

A presentation of the results that addresses the research question is provided in this section. Data from the 14 evolutionary releases is shown in the visualizations, and all 132 releases are used in the statistical correlation analysis.

4.1 Visualizations and Correlation analysis

Figure 2 shows visualizations of data per evolutionary release in the following order: commit activity data, defect data, number of features added, source code measures for C++ components, and source code measures for the XRS scripting files. The actual values are not shown in the figure for confidentiality reasons. Phases described in Section 3.1.2 are highlighted in the figure.

An important observation is that the period of involvement of outsourcing companies according to the contracting agreements does not directly coincide with the time frame of the commit activity across releases. For example, given the contractual agreements at the company as described for Phases I in Section 3.1.2, the expectation was to find employees from three companies for Phase I making commits between R1 and R7. However, Figure 2 shows that the releases with three companies involved are R6 and R12. Company representatives explained that this was because development work from certain locations was often integrated in later releases. However, this observation meant that, commit activity data captures organizational changes that may influence quality that is not evident from Phase information, which is based on period of contractual agreements.

As part of the study we had planned to also compare the differences in how quality changes between the period with distributed development and the period with collocated development, which is the more traditional setting. However, from

Fig. 2: Visualizations of Changes in Measures

analyzing the commit data shown in Figure 2 it can be observed that the period with distributed development is much longer and there are more releases, i.e., 13 evolutionary releases, than when the period with collocated development, which only has one evolutionary release. Therefore this huge difference in number of releases would make for an unfair comparison.

The source code measures shown in Figure 2 are normalized for both C++ and XRS files so as to show the trend in size and complexity measures across release. Complexity of core components (RFC from C++ components) and XRS files has a unique pattern (immediate ups and downs) across releases. Release notes suggest refactoring activities were often done to manage complexity, which could be linked to the unusual patterns. However, there are no discernible differences between patterns across releases with changes in number of companies making commits.

Apart from the source code measures shown in Figure 2, there were four other measures that were collected but were excluded from the analysis: number of classes, weighted method per class (WMC) (Chidamber and Kemerer, 1994), depth of inheritance (DIT) (Chidamber and Kemerer, 1994) and ratio-comment-to-code measure. The complexity measures, number of classes and WMC, correlated with the size measure LOC, which indicates that the measures captured redundant trends. Related studies have made similar observations regarding the redundancy of size and complexity measures, e.g., (Gyimóthy et al., 2005; Singh et al., 2010). Therefore, number of classes and WMC were excluded because they did not provide any added value for the analysis or conclusions. The inheritance measure, DIT, and the ratio-comment-to-code measure were excluded because they were relatively stable and did not significantly deviate across releases. A review of related studies that investigate the links between source code measures and quality also highlighted that DIT does not show any significant relation to quality characteristics such as reliability (Jabangwe et al., 2015).

Company representatives mentioned that the increase and decrease of defects across releases under each phase could be better explained by the changes in features added rather than the development setting or the organizational changes between Phase I-III. Figure 2 appears to support their notion as there appears to be a similar trend between the number of defects and features added across releases. For example, the figure shows that the largest number of defects reported is for release R8, and a manual analysis of release notes that was performed together with an expert at the company confirmed that release R8 came with some of the most complex games.

Correlation analysis was then performed to assess the link between the changes in defect data and the internal source code measures, number of companies involved, and features added during the evolution of PLAY. The number of companies is based on the activity, which captures the organizational changes in terms of development activities across releases. According to the results from the Shapiro-Wilk normality test, shown in Table 5 in Appendix A.1, all data variables are not normally distributed, i.e., total defects, RFC, LOC (for both C++ and XRS files), churned LOC for XRS files, number of companies, and features added. This suggests that nonparametric statistical methods are appropriate for these data variables.

Table 2 shows the results of the Spearman correlation analysis, a nonparametric statistical method. It is important to note that the data used in the correlation analysis is not grouped by company, location or teams. Number of companies has a very low correlation with defect data and it is not significant at either 0.01 or 0.05 level. Similar observations can be made for LOC (for both XRS and C++),

and RFC. Features added has the highest correlation with total defects (with a correlation of 0.648), and significant at 0.01. In essence, features added explains more of the variability in the post-release defect data (0.648^2 approximately 42%) than the other measures, in particular, the change in number of companies. This led us to investigate the practices employed to mitigate the influence of the development setting on the patterns in post-release defects.

Table 2: Spearman Correlation Results (** *Correlation is significant at the 0.01 level*; * *Correlation is significant at the 0.05 level*)

	Total Defects	RFC	LOC	LOC XRS	Churned LOC XRS	Num of Companies	Features Added
Total Defects	1.000	-.052	-.004	.050	.532**	.037	.648**
RFC	-.052	1.000	.244**	.356**	-.058	-.32**	-.166
LOC	-.004	.244	1.000	.885**	.092	-.43**	-.007
LOC XRS	.050	.356	.885	1.000	.160**	-.62**	.016
Churned LOC XRS	.53**	-.058	.092	.160*	1.000	.072	.592**
Num of Companies	.037	-.321	-.432	-.624	.072	1.000	.032
Features Added	.65**	-.116	-.007	.016	.592	.032	1.000

4.2 Mitigation Practices

An important finding is that independent and well defined development tasks were partitioned across the geographically distributed companies (e.g., the development of individual games that would be integrated into each release for PLAY). Essentially, each development location worked independently, in separate development branches in Subversion that would later be merged. This formed a distributed context characterized as a multisite distributed development setting with colocated team members. This decreased dependencies between the companies, and the need for complex communication and coordination for task completion between the development locations. In Phase I and Phase II, a configuration manager that was also a developer at ZCo-SE, performed all merging and integration activities, while in Phase III this responsibility was relocated to YCo-UKR as part of the project management responsibility. Company representatives explained that the partitioning was made possible by the modularity of the system. They explained further that, because of the modular system structure it was possible for vendors to complete their development tasks with limited extensive knowledge about the core system components.

Company representatives revealed that sufficient knowledge and competency of the offshore companies was key during the evolution of the product PLAY. Knowledge transfer activities were carried out at both offshore vendor locations. The strategy applied for YCo-UKR can be characterized as learning by doing. Prior to 2007, YCo-UKR developed their knowledge about PLAY by taking part in maintenance activities. Eventually, the company received more complex tasks and started developing new features, i.e. implementing new games and developing the core components of PLAY. For XCo-IN knowledge transfer was handled through onsite training and exchange visits. First, two developers, two testers and a manager from ZCo-SE stayed

at XCo-IN for about two weeks during which they conducted workshops and other knowledge transfer activities. Then, a few employees from XCo-IN were temporarily relocated to ZCo-SE for approximately a month to continue acquiring knowledge about the product. On their return to XCo-IN they disseminated the knowledge to the other employees.

Because of YCo-UKR's long involvement in product development activities, by the time the handover of project management responsibilities were carried-out, the offshore location had built-up necessary competencies. In addition, during the handover activities, the sending company ensured that there were sufficient resources to support the receiving company (i.e., YCo-UKR) during development and maintenance activities.

The case company strategically monitored the evolution and quality of the components onshore prior to handing over responsibilities offshore in Phase III. It was uncovered from the focus group meetings that refactoring activities were also often done in order to manage complexity of the source code. This resulted in the unusual patterns in the complexity measure, RFC, for the core source code components that can be observed in Figure 2.

Focus group participants described the development teams at ZCo-SE and YCo-UKR as relatively stable. However, one of the offshore outsourcing companies, XCo-IN, faced issues with high employee turnover, which destabilized the competency and experience level of the team at the company. This is not a peculiar issue in offshoring projects and it is difficult to eliminate as also discussed by Moe et al. (2014). As a quality assurance practice to mitigate the effect of attrition on quality, XCo-IN had dedicated senior company representatives that were responsible for monitoring development work and evaluating final source code artifacts developed at the company, as well as making all commits into Subversion development branches. This served as a solution to avoid issues that could be attributed to new or novice employees that continuously joined the team. This also explains the few number of contributors from the company, XCo-IN, that appears in the commit activity data visualization shown in Figure 2.

4.3 Overall Observation (RQ1)

Essentially, the visualizations and the correlation analysis results are consistent with the perspectives from company representatives, with regards to the link in patterns in the defect data and features added. In general, the expectation was that changes in the development setting in Phase I, II and III would have a stronger link to changes in post-release defect data. This is based on the challenging nature of global projects (Gomes and Marczak, 2012; Kannabiran and Sankaran, 2011), and the large geographical distance with one of the development locations, which adds to the difficulties (Herbsleb and Mockus, 2003). This suggests that post-release quality was ensured.

The practices revealed in Section 4.2 are summarized in Table 3 and they have been categorized on their relation to people, process and product factors. The phase in which the practices were mostly implemented is indicated in the table under "Phase". It is important to note that these were factors pointed out by company representatives that attended the focus group meetings, therefore this may not be an exhaustive list of all practices that were implemented at the case company.

Table 3: Mitigation Practices

Category	Practices	Phase
People	Low attrition thus stabilizing competency levels at one of the offshore location.	Phase I,II and III
People	Senior company representatives managing commit activity at offshore location that had attrition issues.	Phase I and II
People	Support from original developers to offshore developers when relocating project management responsibilities.	Phase III
Process	Knowledge transfer to improve competencies at offshore locations.	Before and during Phase I
Process	Monitoring the evolution and complexity of the core source code components onshore prior to relocating project management responsibilities offshore.	Phase I and II
Product	Modular architecture reduces the need to have extensive knowledge about each system component for independent task completion for isolated teams.	Phase I, II and III
Product	Refactoring activities to manage source code complexity.	Phase I, II and III
Product	Strategic task allocation of independent development tasks to reduce dependencies between locations by forming isolated and loosely coupled development teams. This formed a multisite development setting.	Phase I, II and III

A key finding from the reflections in the focus groups pertains to the nature of distributed development. The development teams at each location worked independently across releases. This suggests that team members were collocated rather than distributed, even though the development locations were distributed. This essentially means that development teams at each location were working on separate source code branches. Šmite (2014) refers to this kind of project arrangement as a “distributed outsourcing project”. In such a case, defects reported during integration are vital to fully capture the impact on quality of either the distributed setting or the organizational changes. Unfortunately, they were inconsistently maintained across releases, and hence were unreliable. Because, post-release defects were analyzed (as the only reliable quality proxy), the only conclusion that can be made is about the post-release quality. That is, the studied case appear to have ensured post-release quality given the challenges posed by offshore outsourcing ventures. But, the

“impact” on quality of either the distributed setting or the organizational changes on quality is inconclusive.

5 Discussion

The studied case is an offshore outsourcing software development project, which consisted of changes in the number of vendors involved whilst the product evolved. The findings suggests that the onshore company ensured post-release quality despite the complex multi-site development environment with several organizational changes. The context and findings of our study are summarized in Table 4. The contents of the table can be used to compare and contrast our study and related studies that are summarized in Table 1 from Section 2.

Table 4: Summary of the presented study (compare with Table 1)

Study aim in relation to distributed development and quality	Proxy for quality	Distributed development context	Findings
To investigate how quality changes during evolution in a distributed development environment that incurs organizational changes in terms of number of companies involved, across releases.	Post-release defects	Distributed outsourcing project, which incurred organizational changes in terms of number of outsourcing companies involved, across releases.	Though the studied case appears to have ensured post-release quality, the “impact” on quality of either the distributed setting or the organizational changes on quality is inconclusive.

In summary, our case complements the findings from Ramasubbu and Balan (2007) and Bird et al. (2009) who look at post-release defects, although in contrast to the focus on dispersed teams versus co-located teams in an insourcing context we rather explore a distributed setting with multiple outsourcing vendors. As such, our study appears to be the only case among the outlined related studies that covers distributed development modularized by design (as opposed to dispersed teams), and the only outsourcing case (as opposed to insourcing and open source projects). The findings in terms of quality indicate that the studied case appear to have ensured post-release quality. Although our findings come from a different context than those

by Spinellis (2006), Bird et al. (2009) and Sutherland et al. (2009), since we study distributed teams and not dispersed teams, all of the mentioned cases including ours can be yet claimed to represent successful global software development projects. And finally, the results of our qualitative analysis provide an explanation for the achieved success in terms of post-release quality. We found that, among other reasons that are summarized in Table 3, strategic task allocation of independent development tasks across distributed locations and dedicated integration effort helps to address the possible risks of poor quality.

At first glance it seems as if the findings in the present study is in line with studies that report a negligible link between distributed development and quality (e.g., by Spinellis (2006), Sutherland et al. (2009) and by Bird et al. (2009)), and contradicting those that report otherwise (e.g., by Ramasubbu and Balan (2007), Cataldo and Herbsleb (2011) and by Bird and Nagappan (2012)). But on closer examination that would not be an accurate inference, and this is because of differences in GSD contexts or data used in the analysis.

It can be argued that the contexts in the related work differ considerably. Where as Sutherland et al. (2009) studied a product developed in a fully distributed scrum team, the product in the present study can be compared to isolated scrum teams. In addition, the proxy for quality used by Sutherland et al. (2009) seems to be those reported pre-release as they are issues reported during iteration, which differs from the post-release defects used in the present study. Spinellis (2006) also studies an open source project, which has unique characteristics in comparison to software development activities performed in traditional office space buildings.

Cataldo and Herbsleb (2011) has shown that integration problems are higher for dispersed feature teams. Ramasubbu and Balan (2007) also found that distributed development decreases quality when analyzing pre-release defects. Bird and Nagappan (2012) found similar findings when analyzing both pre-release and post-release defects. Given, that the present study uses post-release defects, and there are reports of issues occurring pre-release in distributed projects, it is possible that defects could have been found during integration and fixed before releasing each version. But this is not observable from post-release defects. Similar issues can be raised for studies that do not find a link between distributed development quality when using only post-release defects in the analysis, e.g., Bird et al. (2009).

Given that in the present study development teams worked independently and only post-release defects are analyzed, then it can be postulated that the actual impact on quality of the distributed setting or the organizational changes is unclear. Unfortunately, pre-release and integration defects at the case company were unreliable and it was not possible to accurately trace them to source code components from each development location for each release. As Posnett et al. (2011) points out, conclusions and inferences drawn made using data that is aggregated at one level are not directly transferable to other levels. Thus, measures for post-release quality are not indicative of quality concerns, or lack thereof, pre-release or during integration. In essence the study findings presented do not offer a conclusive indicator of the lack of impact of either distributed development or organizational changes on quality, but its the lack of evidence of the impact.

5.1 Implications for Research

The importance of verifying the accuracy of data with practitioners cannot be over emphasized. It helps strengthen the validity of study findings. Failure to verify the accuracy and taking appropriate corrective measures may result in flawed conclusions. At one of the vendor locations in the present study, there were a few selected employees that were permitted to make commits - this was done for quality assurance purposes. This issue was only found after discussing the data analysis results with practitioners. The practice may not be uncommon beyond the presented context. The implication is on the correctness of measures extracted from the commit data that are intended to capture the development activities of source code artifacts, such as, number of developers, number of contributors, and ownership measures for source code components. These measures are commonly used in studies that involve analysis of commit activity, e.g., the study by Bird et al. (2011) and by Kocaguneli et al. (2013). Thus, there should be emphasis to avoid blindly using measures obtained from data, but to understand the contexts, and to involve practitioners in the analysis to help verify accuracy of the data.

Because of the different ways in which development teams can collaborate in GSD, caution should be made when drawing inferences from particular type of data. In order to determine the effect on quality of distributed development settings, the data used as a proxy for quality should be aggregated at an appropriate level. As observed in the present study, development teams can work independently with collocated team members, and integration defects would have been key for better understanding the impact on quality. If development team members were distributed then, in order to better understand the implications on quality, the proxy for quality measured should be traced to the source code modules that each team develops. This highlights the importance of using an appropriate quality proxy for certain distributed development contexts in order to avoid making erroneous inferences, and to better support practitioners with well-founded evidence.

5.2 Implications for Practice

Practitioners should maintain defect data consistently across releases, especially defects found during key steps in the development activities. This will give a better reflection of the quality of the software pre- and post-release. The information is key for getting useful research results that can be used to build a richer repository of evidence, which can then be used for making better-informed decisions in future projects.

The importance and usefulness of the mitigation practices that are presented in this present study have been noted in studies on other GSD contexts as success or critical factors (e.g., the studies by Moe et al. (2014), Jabangwe et al. (2013), and by da Silva et al. (2010)). Thus, the findings of this study shows the relevance of the factors in a distributed development context with offshore outsourcing vendors. The findings can be valuable input to decision-making structures for GSD projects similar to the one proposed by Šmite et al. (2013). It is, however, important for practitioners to compare their own context with the one in this study (described in Section 3.1) so as to determine the applicability and usefulness of the mitigation practices in their own specific contexts.

6 Validity Threats

Empirical studies, such as the one presented, are not impervious to threats to validity. Validity threats pertain to factors that may affect the outcome of a study (Robson, 2011).

Construct Validity: This is concerned with measuring what is intended to be measured for the purpose of the study (Robson, 2011). The size measures, LOC and churned LOC, and the complexity measure, RFC, used in the study have been empirically evaluated on their link with quality (Jabangwe et al., 2015). LOC and churned LOC are common and simple size measures, and the theoretical base of RFC is well described by Chidamber and Kemerer (1994). To ensure consistency, extraction of these measures was performed using the same static analysis tools, for all releases. Experts that had knowledge about the evolution of the product at the case company verified the accuracy of the values for the measures, as well as other data used throughout this study.

Internal Validity: The retrospective analysis nature of this study makes it susceptible to historical-types of internal validity threat (Robson, 2011). This pertains to historical events (e.g., that occurred during product evolution) that may influence the validity of cause-effect relationships (Robson, 2011). A company representative with extensive knowledge about the product helped with constructing a time-line of the products evolution, identifying appropriate releases, as well as isolating and collecting relevant data. Company documentations and focus group meetings with those that had extensive knowledge of software development activities or the evolution of the product were used to capture detailed events and practices that occurred whilst the product was evolving, and to determine their possible link with changes in internal and external product measures.

Apart from the historical-types of internal validity threats that this retrospective study is subject to, another issue is related to the selection of participants of the focus group meetings. Due to geographic distribution and organizational changes it was not possible to involve representatives from all sites. Participants included key employees from ZCo-SE that at the time of investigation were still employed, but not all employees involved in the evolution could participate. Only the key developer/project manager from YCo-UKR was involved in one of the meetings, and none of the employees from XCo-IN participated since their engagement in the collaboration was over at the time of investigation. The implication is that some key information can be missed that can help to better understand the reasons behind the changes in quality during evolution for certain releases. However, we believe that this issue was mitigated by two key characteristics of those involved in the meetings. First, the participants of the meetings had extensive knowledge about the evolution of PLAY through their involvement in development and testing activities, designing architecture, and managerial responsibilities. Second, the employees that helped with this study covered diverse and vital roles that were critical to the development and evolution of PLAY. The overall evaluation of the link between the organizational changes and the quality is measured quantitatively and is thus not affected by the selection of focus group participants.

External Validity: This is concerned with the extent of generalizability of a study (Robson, 2011). In Section 3.1, there is a detailed description of the studied context to help improve the understanding of the extent of generalizability and validity of the conclusions that are drawn from the empirical investigation (Petersen

and Wohlin, 2009). There were also many releases in a relatively short period. This may not be the case for, for example, telecommunication software systems. Therefore it is important to evaluate the applicability and usefulness of the reported findings in other contexts.

GSD contexts differ from case to case, but analytical induction facilitates with determining the generalizability between cases (Wieringa, 2013). Hence, in this study we provide an in-depth analysis of a specific case, and carefully describe the context, and provide clear insights of a particular context. The rationale for doing this is to make it easier for practitioners and researchers to compare the studied context and their own context. For example, there may be similarities in the offshoring activities and team sizes in the presented case with other cases. In such situations, practitioners may be able to use some or all of our findings as input when making decisions for their own projects. For researchers, the detailed description of the context can make it easier for them to compare and synthesize our study findings with other studies of similar characteristics, and thereby produce stronger evidence that can be useful to practitioners.

Conclusion Validity: This is concerned with conclusions (Wohlin et al., 2012). Two actions were taken in order to mitigate this threat. First, two researchers (i.e., the first two authors of this paper) independently analyzed the visualizations of the data, and reached consensus on the observations. Second, all observations and findings were triangulated with employees at the case company through focus group meetings, and multiple telephone and email exchanges. These employees had extensive knowledge about the product and its evolution due to their involvement in development and maintenance activities over long periods, across many releases.

7 Conclusions and Future Work

The context of the studied case is an offshore outsourcing setting, which consists of globally distributed teams. The data used is from a software product that incurred organizational changes, in terms of number of vendors involved, whilst it evolved. The observation made was that the company ensured post-release quality, and practitioners of the studied case company offered similar and supporting observations during focus group meetings. Among other practices, the following are some that contributed to the success: reduced dependencies between geographically distributed teams by allocating independent tasks to each team, knowledge transfer to ensure sufficient competencies across locations, refactoring activities to manage source code complexity during evolution.

The study also highlights the importance of taking into consideration the development setting when measuring proxies for quality and making inferences about the development setting. Though the case company ensured post-release quality, similar observations could not be made about the effect on quality of the development setting. This is because, given that the distributed teams worked in isolation, it is possible that integration issues may have occurred, but this cannot be observed from post-release defects. Therefore, the effect of either the development setting or the organizational changes on quality could not be sufficiently investigated with just post-release defects, which was unfortunately the only reliable source of defects available. Similar concerns can be raised for studies on distributed development reported in literature that also uses the same proxy (post-release defects) for quality.

Generally, project arrangements in global software development projects have implications on how development teams collaborate. There can be distributed teams with each team working independently but working towards the development of the same software, or distributed teams can work as virtual teams in which team members are dispersed across different locations. A possible future work is the formulation of a method for investigating quality for products developed in distributed development settings that takes into account contextual information linked to the distinguishing characteristics of such settings. Such a method can help improve consistency in the manner in which studies on distributed development are performed. Consistency would make it easier to synthesize and compare findings across studies, and would facilitate with building useful information to support practitioners.

Acknowledgments

This research work is partially supported by the Swedish Knowledge Foundation in Sweden under the grant 20120200 (2013-2016) and Ericsson Software Research.

A Appendix

A.1 Shapiro-Wilk Test (Normality Test)

Table 5 shows normality test results. Null hypothesis of the Shapiro-Wilk test is that the data is normal. Since the p-value for all variables (under column “sign” for Shapiro Wilk) is less than 0.01 (which is the significance level), we can reject the null hypothesis. Thus, they are not normally distributed.

Table 5: Shapiro-Wilk Test: Normality Test Results

	Statistic	df	Sig.
Total Defects	.392	132	.000
RFC	.935	132	.000
LOC	.842	132	.000
LOC XRS	.922	132	.000
Churned LOC XRS	.343	132	.000
Num of Companies	.672	132	.000
Features Added	.419	132	.000

References

- Ali, S. and Khan, S. (2014). Critical success factors for software outsourcing partnership (sop): A systematic literature review. In *Proceedings of the 9th International Conference on Global Software Engineering*, pages 153–162.
- Alsudairi, M. A. and Dwivedi, Y. K. (2010). A multi-disciplinary profile of is/it outsourcing research. *Journal of Enterprise Information Management*, 23(2):215–258.
- Babar, M. A., Verner, J. M., and Nguyen, P. T. (2007). Establishing and maintaining trust in software outsourcing relationships: An empirical investigation. *Journal of Systems and Software*, 80(9):1438 – 1449.
- Bilal Raza, S. G. M. and Clear, T. (2013). Topics and treatments in global software engineering research: A systematic snapshot. In *Proceedings of the 8th International Conference on Evaluation of Novel Approaches to Software Engineering*, ENASE’13, pages 85–96.
- Bird, C. and Nagappan, N. (2012). Who? where? what? examining distributed development in two large open source projects. In *Proceedings of the 9th IEEE Working Conference on Mining Software Repositories*, pages 237–246.
- Bird, C., Nagappan, N., Devanbu, P., Gall, H., and Murphy, B. (2009). Does distributed development affect software quality? An empirical case study of Windows Vista. In *Proceedings of the 31st International Conference on Software Engineering*, pages 85–93.
- Bird, C., Nagappan, N., Murphy, B., Gall, H., and Devanbu, P. (2011). Don’t touch my code!: Examining the effects of ownership on software quality. In *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering*, ESEC/FSE ’11, pages 4–14. ACM.
- Carmel, E. and Tjia, P. (2005). *Offshoring information technology: Sourcing and outsourcing to a global workforce*. Cambridge University Press, Cambridge, UK.
- Cataldo, M. and Herbsleb, J. D. (2011). Factors leading to integration failures in global feature-oriented development: An empirical analysis. In *Proceedings of the 33rd International Conference on Software Engineering*, pages 161–170, New York, NY, USA. ACM.

- Chidamber, S. R. and Kemerer, C. F. (1994). A metrics suite for object oriented design. *IEEE Transactions on Software Engineering*, 20(6):476–493.
- Colomo-Palacios, R., Casado-Lumbreras, C., Soto-Acosta, P., García-Peñalvo, F. J., and Tovar, E. (2014). Project managers in global software development teams: a study of the effects on productivity and performance. *Software Quality Journal*, 22(1):3–19.
- da Silva, F., Costa, C., FranÁga, A., and Prikladinicki, R. (2010). Challenges and solutions in distributed software development project management: A systematic literature review. In *Proceedings of the 5th International Conference on Global Software Engineering*, pages 87–96.
- Damian, D., Izquierdo, L., Singer, J., and Kwan, I. (2007). Awareness in the wild: Why communication breakdowns occur. In *Proceedings of the 2nd International Conference on Global Software Engineering*, pages 81–90.
- Dibbern, J., Winkler, J., and Heinzl, A. (2008). Explaining variations in client extra costs between software projects offshored to india. *MIS quarterly*, pages 333–366.
- El-Emam, K. (2002). Object-oriented metrics: A review of theory and practice. In Erdogmus, H. and Tanir, O., editors, *Advances in Software Engineering*, pages 23–50. Springer New York.
- Fabrieck, M., Brand, M. v. d., Brinkkemper, S., Harmsen, F., and Helms, R. (2008). Reasons for success and failure in offshore software development projects. In *Proceedings of the European Conference on Information Systems*, pages 446–457.
- Gomes, V. and Marczak, S. (2012). Problems? we all know we have them. do we have solutions too? a literature review on problems and their solutions in global software development. In *Proceedings of the 7th International Conference on Global Software Engineering*, pages 154–158.
- Gyimóthy, T., Ferenc, R., and Siket, I. (2005). Empirical validation of object-oriented metrics on open source software for fault prediction. *IEEE Transactions on Software Engineering*, 31(10):897–910.
- Herbsleb, J. D. and Mockus, A. (2003). An empirical study of speed and communication in globally distributed software development. *IEEE Transactions on Software Engineering*, 29(6):481–494.
- Herraiz, I. and Hassan, A. E. (2012). *Making software – What really works, and why we believe it*, chapter Beyond lines of code: Do we need more complexity metrics?, pages 125–141. O’Reilly Media.
- Iacovou, C. L. and Nakatsu, R. (2008). A risk profile of offshore-outsourced development projects. *Communications of the ACM*, 51(6):89–94.
- ISO/IEC-25010 (2010). Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models. International organization for standardization.
- ISO/IEC/IEEE-24765 (2010). Systems and software engineering – Vocabulary. International organization for standardization.
- Jabangwe, R., Börstler, J., and Petersen, K. (2014). Handover of managerial responsibilities in global software development: a case study of source code evolution and quality. *Software Quality Journal*, pages 1–28.
- Jabangwe, R., Börstler, J., Šmite, D., and Wohlin, C. (2015). Empirical evidence on the link between object-oriented measures and external quality attributes: A systematic literature review. *Empirical Software Engineering*, 20(3):1–54.
- Jabangwe, R., Petersen, K., and Šmite, D. (2013). Visualization of defect inflow and resolution cycles: Before, during and after transfer. In *Proceedings of the 20th Asia-Pacific Software Engineering Conference*, volume 1, pages 289–298. IEEE.
- Jabangwe, R. and Šmite, D. (2012). An exploratory study of software evolution and quality: Before, during and after a transfer. In *Proceedings of the 7th IEEE International Conference on Global Software Engineering*, pages 41–50.
- Jalali, S., Gencel, C., and Šmite, D. (2010). Trust dynamics in global software engineering. In *Proceedings of the 4th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, page 23. ACM.
- Kannabiran, G. and Sankaran, K. (2011). Determinants of software quality in offshore development—an empirical study of an indian vendor. *Information and Software Technology*, 53(11):1199–1208.
- Khan, S. U., Niazi, M., and Ahmad, R. (2009). Critical success factors for offshore software development outsourcing vendors: A systematic literature review. In *Proceedings of the*

- 4th International Conference on Global Software Engineering Conference*, pages 207–216. IEEE.
- Kocaguneli, E., Zimmermann, T., Bird, C., Nagappan, N., and Menzies, T. (2013). Distributed development considered harmful? In *Proceedings of the 35th International Conference on Software Engineering*, pages 882–890. IEEE.
- Krishna, S., Sahay, S., and Walsham, G. (2004). Managing cross-cultural issues in global software outsourcing. *Communications of the ACM*, 47(4):62–66.
- Lacity, M. C., Willcocks, L. P., and Rottman, J. W. (2008). Global outsourcing of back office services: lessons, trends, and enduring challenges. *Strategic Outsourcing: An International Journal*, 1(1):13–34.
- Marques, A., Rodrigues, R., and Conte, T. (2012). Systematic literature reviews in distributed software development: A tertiary study. In *Proceedings of the 7th International Conference on Global Software Engineering*, pages 134–143.
- McCabe, T. J. (1976). A complexity measure. *IEEE Transactions on Software Engineering*, 2(4):308–320.
- Mockus, A. (2010). Organizational volatility and its effects on software defects. In *Proceedings of the 18th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 117–126.
- Moe, N. B., Šmite, D., Hanssen, G. K., and Barney, H. (2014). From offshore outsourcing to insourcing and partnerships: four failed outsourcing attempts. *Empirical Software Engineering*, pages 1–34.
- Moe, N. B. and Šmite, D. (2008). Understanding a lack of trust in global software teams: A multiple-case study. *Software Process Improvement and Practice*, 13(3):217–231.
- Nagappan, N., Murphy, B., and Basili, V. (2008). The influence of organizational structure on software quality: An empirical case study. In *Proceedings of the 30th International Conference on Software Engineering*, pages 521–530.
- Nguyen-Duc, A., Cruzes, D. S., and Conradi, R. (2015). The impact of global dispersion on coordination, team performance and software quality - a systematic literature review. *Information and Software Technology*, 57:277–294.
- Nurdiani, I., Jabangwe, R., Šmite, D., and Damian, D. (2011). Risk identification and risk mitigation instruments for global software development: Systematic review and survey results. In *Proceedings of the 6th International Conference on Global Software Engineering Workshop*, pages 36–41.
- Oshri, I., Kotlarsky, J., and Willcocks, L. P. (2011). *The handbook of global outsourcing and offshoring*. Palgrave Macmillan, 2nd edition.
- Petersen, K. and Wohlin, C. (2009). Context in industrial software engineering research. In *Proceedings of the 3rd ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 401–404.
- Pannenstein, L. L. and Tsai, R. J. (2004). Offshore outsourcing: Current and future effects on american it industry. *Information Systems Management*, 21(4):72–80.
- Posnett, D., Filkov, V., and Devanbu, P. (2011). Ecological inference in empirical software engineering. In *Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering*, pages 362–371, Washington, DC, USA. IEEE Computer Society.
- Ramasubbu, N. and Balan, R. K. (2007). Globally distributed software development project performance: An empirical analysis. In *Proceedings of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, pages 125–134.
- Razali, N. M. and Wah, Y. B. (2011). Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. *Journal of Statistical Modeling and Analytics*, 2(1):21–33.
- Robson, C. (2011). *Real world research*. John Wiley & Sons, West Sussex, UK, 2nd edition.
- Rottman, J. W. and Lacity, M. C. (2006). Proven practices for effectively offshoring it work. *Sloan Management Review*, 47:56–63.
- Runeson, P., Höst, M., Rainer, A., and Regnell, B. (2012). *Case study research in software engineering*. John Wiley & Sons, New Jersey, USA.
- Schaaf, J. (2004). Offshoring: Globalisation wave reaches services sector. *Deutsche Bank Research. E-conomics*, (45).
- Singh, Y., Kaur, A., and Malhotra, R. (2010). Empirical validation of object-oriented metrics for predicting fault proneness models. *Software Quality Journal*, 18(1):3–35.

- Spinellis, D. (2006). Global software development in the freeBSD project. In Kruchten, P., Hsieh, Y., MacGregor, E., Moitra, D., and Strigel, W., editors, *Proceedings of the International Workshop on Global Software Development for the Practitioner*, pages 73–79.
- Sutherland, J., Schoonheim, G., and Rijk, M. (2009). Fully distributed scrum: Replicating local productivity and quality with offshore teams. In *System Sciences, 2009. HICSS’09. 42nd Hawaii International Conference on*, pages 1–8. IEEE.
- Verner, J., Brereton, O., Kitchenham, B., Turner, M., and Niazi, M. (2012). Systematic literature reviews in global software development: A tertiary study. In *Proceedings of the 16th International Conference on Evaluation Assessment in Software Engineering*, pages 2–11.
- Šmite, D. (2014). Distributed project management. In Ruhe, G. and Wohlin, C., editors, *Software project management in a changing world*, pages 301–320. Springer Berlin Heidelberg.
- Šmite, D. and Wohlin, C. (2011). Strategies facilitating software product transfers. *IEEE Software*, 28(5):60–66.
- Šmite, D., Wohlin, C., Aurum, A., Jabangwe, R., and Numminen, E. (2013). Offshore insourcing in software development: Structuring the decision-making process. *Journal of Systems and Software*, 86(4):1054–1067.
- Šmite, D., Wohlin, C., Galviņa, Z., and Prikladnicki, R. (2014). An empirically based terminology and taxonomy for global software engineering. *Empirical Software Engineering*, 19(1):105–153.
- Wieringa, R. (2013). Case study research in information systems engineering: how to generalize, how not to generalize, and how not to generalize too much. In *Proceedings of the 25th International Conference on Advanced Information Systems Engineering*, pages xii–xii.
- Winkler, J. K., Dibbern, J., and Heinzl, A. (2008). The impact of cultural differences in offshore outsourcing—case study results from german–indian application development projects. *Information Systems Frontiers*, 10(2):243–258.
- Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2012). *Experimentation in software engineering: An introduction*. Springer-Verlag Berlin Heidelberg, New York, USA.