Agile vs. Structured Distributed Software Development: A Case Study

H.-Christian Estler Martin Nordio Carlo A. Furia Bertrand Meyer
Chair of Software Engineering, ETH Zurich, Switzerland
E-mail: firstname.lastname@inf.ethz.ch

Johannes Schneider*
IBM Research, Zurich, Switzerland
E-mail: joh@zurich.ibm.com

***Abstract*—This paper presents a case study on the impact of development processes on the success of globally distributed software projects. The study compares agile (Scrum, XP, etc.) vs. structured (RUP, waterfall) processes to determine if the choice of process impacts: the overall success and economic savings of distributed projects; the importance customers attribute to projects; the motivation of the development teams; and the amount of real-time or asynchronous communication required during project development.**

The case study includes data from 66 projects developed in Europe, Asia, and the Americas. The results show no significant difference between the outcome of projects following agile processes and structured processes, suggesting that agile and structured processes can be equally effective for globally distributed development. The paper also discusses several qualitative aspects of distributed software development such as the advantages of nearshore vs. offshore, the preferred communication patterns, and some common critical aspects.

***Index Terms*—Distributed software development, Outsourcing, Agile, Empirical study**

I. Introduction and Overview

The importance of choosing the right development process to ensure the successful and timely completion of distributed software projects cannot be understated... Or *can* it? This paper presents an extensive case study analyzing the impact of different development processes on the success of software projects carried out by globally distributed development teams.

Globally distributed software development (GSD) has become a common practice in today's software industry; companies cross the barriers introduced by distance, cultural differences, and time zones, looking for the most skilled personnel and the most cost-effective solutions. GSD may exacerbate several of the criticalities already present in traditional *local* software development, and it often generates its own peculiar challenges originating in the difficulty of carrying out the traditional parts of a software development project—requirements elicitation, API design, project management, team communication, etc.—in environments where members of the same team live and work in different countries, or even in different continents.

Given the challenges and peculiarities introduced by GSD, it is interesting to peruse the standard methods and practices that have been successful in traditional local software development, determining if they can be applied with positive results also in globally distributed settings. From the perspective of empirical research in software engineering, this general line of inquiry materializes in questions of the form "What is the impact of X on the quality of GSD projects", where "X" is a practice, method, or technique, and "quality" may refer to different aspects such as timeliness, customer satisfaction, or cost effectiveness. Examples of GSD issues investigated empirically along these lines include the usage of contracts for API design [24], the effect of time zones on various phases of development [14], [10], [21] and on productivity and quality [28], [5], and the impact of geographic dispersion on several quality metrics [29].

The case study presented in this paper focuses on *development processes* to find out whether the choice of process has a significant impact on qualities such as programmer productivity and development cost-effectiveness in GSD. To our knowledge, this is one of very few empirical studies that explicitly investigates the impact of agile vs. structured processes on GSD project quality.

A software development process is a scheme to structure and manage the various aspects of development (requirements elicitation, design, implementation, verification, maintenance, etc.). Software engineering [12], [27], [26] has traditionally targeted so-called *structured processes*[1], such as the Rational Unified Process (RUP), the waterfall model, or the spiral model. Structured processes are characterized by a focus on rigorously defined practices, extensive documentation, and detailed planning and management. More recently, a surge of *agile* development processes have been introduced to overcome some of the limitations and unsatisfactory aspects of structured processes. Agile processes [16], [9], such as Scrum or eXtreme Programming (XP), emphasize the importance of effective informal communication among developers and of iterative improvement of implementations, and they champion small cohesive development teams over large structured units. The relative merits and applicability of structured vs. agile processes in local software development are fairly well-understood [18], [15], [3], [19], [6]: for example, for applications whose requirements are accurately known and not subject

(*) Work conducted while affiliated with ETH Zurich

[1]Other names for such processes are: heavyweight, plan-driven, disciplined. In this article, we will consistently use the term "structured" to denote "non-agile" processes; this is merely a terminological convention and does not entail that agile processes have no structure whatsoever, or that structured processes are completely inflexible.

to radical changes, a structured development may offer more controllability and better scalability; on the other hand, agile processes may be preferable when requirements are subject to frequent change and achieving a formal and structured communication with stakeholders is difficult or unrealistic.

The present paper's study re-considers the "structured vs. agile" dichotomy in the context of GSD, and tries to understand whether one of the two development approaches is more appropriate to organize software development carried out by globally distributed teams. It is clear that the results obtained in non-distributed contexts may not hold in distributed settings, where it is not obvious how to enforce some of the principles underlying structured or agile methods. Agile processes, in particular, often require that [16]:

- all project phases include communication with customers;
- face-to-face exchanges be preferred as the most efficient and effective method of communicating.

On the other hand, structured processes often emphasize the importance of maintaining accurate documentation, which can be problematic when cultural and language differences are in place. Correspondingly, effectively applying the principles of agile rather than structured development in a distributed setting has been the subject of much software engineering research [32], [25].

The question remains, however, of what are the relative merits of structured and agile processes for GSD, and whether one of them is more likely to be effective. The present paper targets this question with a study involving over 31 companies (of size from small to large) for a total of 66 software projects developed in Europe, Asia, and the Americas. The degree of distribution ranges from merely outsourced projects—where management remains in the company's headquarters while the actual development team operates in a different country—to highly distributed development projects—where members of the same team reside in different countries. According to the answers collected through questionnaires and interviews, we have classified the development process used in each project into agile or structured, and we have analyzed the correlation between process type and measures of achieved overall success, importance for the company, cost-effectiveness, developer motivation, and amount of personal communication. Our results show that the differences in any of these measures between agile and structured processes are negligible and with no statistical significance. Therefore, our study suggests that agile and structured processes can be equally effective (or ineffective) for GSD, and the sources of significant differences in project outcome should be sought in other project characteristics.

The rest of the paper is organized as follows. Section II presents the research questions investigated in the case study. Section III describes the data collection process and the research methodology. Section IV presents the quantitative results of the study, whereas Section V is devoted to a somehow informal discussion of other aspects for which only qualitative data is available. Sections VI and Section VII respectively describe threats to validity and related work. Section VIII summarizes and describes future work.

II. Research Questions

While the benefits of deploying structured vs. agile processes have been extensively studied in the context of traditional local development, their applicability to and impact on globally distributed development are still largely unknown. This paper contributes to filling this knowledge gap by investigating the impact of using different processes—structured rather than agile—on the outcome of software projects carried out in distributed settings. This leads to the following fundamental research question:

RQ: *In software development carried out in globally distributed settings, what is the impact of adopting structured vs. agile processes on the overall success, cost-effectiveness, team motivation, importance for customers, and amount of communication?*

For each aspect (overall success, cost-effectiveness, etc.), we formulate a null-hypothesis that states the *absence* of correlation between development process type and outcome; all hypotheses refer to development projects in *distributed* settings.

H_0^A There is no difference in the *overall success* of projects developed using agile methods vs. projects developed using structured methods.

H_0^B There is no difference in the *importance* (i.e., criticality for customers) of projects assigned to development using agile methods vs. projects assigned to development using structured methods.

H_0^C There is no difference in the *motivation* of teams following agile processes vs. teams following structured processes.

H_0^D There is no difference in the estimated *economic savings* achieved in projects developed using agile methods vs. projects developed using structured methods.

H_0^E There is no difference in the amount of *real-time communication* (e.g., in person or by phone) required by projects developed using agile methods vs. projects developed using structured methods.

H_0^F There is no difference in the amount of *asynchronous communication* (e.g., emails or wikis) required in projects developed using agile methods vs. projects developed using structured methods.

If we manage to falsify, with a degree of statistical significance, any null-hypothesis, we show evidence in support of the corresponding *alternative* hypothesis: the choice of agile rather than structured development processes has an impact on the outcome of a certain aspect of GSD projects.

The following section describes how data was collected to support or falsify the hypotheses above. In addition to quantitative hypotheses, we have also investigated the research question RQ qualitatively in Section V.

III. Research Methodology

The data was collected in two phases[2]. In the first phase, we sent out questionnaires to companies in Europe, Asia, and the Americas, about their offshore and distributed development projects. In the second phase, we interviewed representatives of several companies located in Switzerland, also about distributed development projects. Both the questionnaires and the interviews targeted distributed projects, collecting data about: their success for the companies, their cost-effectiveness, team motivation, importance for customers, amount of communication, and whether they were organized according to a structured or agile process. We did not distinguish among different types of agile (e.g., Scrum vs. extreme programming) or different types of structured (e.g., RUP vs. waterfall) processes; this is consistent with the observation that, while different processes may involve different practices, the principles underlying agile rather than structured methods are normally visibly different and straightforward to identify in practice. The complete data set contains information about 66 distributed projects (with various degrees of distribution), of which 36 deployed agile methods and 30 structured methods.

A. Questionnaires

In the first phase, we contacted companies in various countries and continents worldwide through questionnaires; each questionnaire targeted one software project, containing several questions about the project.

We sent out the questionnaires to over 60 contacts worldwide, and we received replies about 48 projects developed by companies in the USA (14 projects), Nordic countries (12 projects), Germany (6 projects), the UK (4 projects), Russia (1 project), the Netherlands (1 project), Latin America (1 project), and Switzerland (3 projects, from companies other than those involved in the interviews of the second phase)—the countries of origin of the remaining 6 projects were unspecified. 22 of the 48 projects were in collaboration with remote units in Russia, 20 in India, 2 in Argentina, and 1 in each of China, Bulgaria, Hungary, Romania.

The information in the questionnaires was provided by 19 high managers, 19 project leaders, and 10 software engineers, architects and researchers. 19 projects out of 48 followed structured processes, and 29 applied agile processes.

B. Interviews

In the second phase, we contacted 13 Swiss software companies including: 3 large companies with more than 10'000 employees worldwide; 8 mid-size companies with 200 to 900 employees each; and 2 small companies with less than 100 employees. 6 of the companies also develop hardware products.

We individually interviewed 18 employees from these 13 companies, that have experience with globally distributed development. Of the 18 employees, 9 were high managers (CEOs, CTOs, or business unit leaders), 9 were project managers and senior software engineers.

Each interview discussed a recently completed software project the interviewee had been involved with. All the projects were in collaboration with distributed units from companies in Western Europe, Eastern Europe, Russia, or Asia. In 12 of the 18 projects, the units outside Switzerland were subsidiaries of the main company; the collaboration in the other 6 projects can be characterized as off-shore development provided by external companies. Finally, 11 projects out of 18 followed structured processes, and 7 applied agile processes.

The questions asked during the interviews were similar to those used in the questionnaires; it was much harder, however, to get quantitative data from the interviews, because the interviewees were often evasive when asked to characterize precisely measures such as the success or economic savings of a project, and only gave generic answers such as "the project was successful" or "the savings were small". For this reason, we used the data from the interviews only in the qualitative analysis of Section V.

[2]The data-set is available at http://se.inf.ethz.ch/data/icgse12.zip. Statistical analysis was performed using IBM SPSS v. 20.

IV. Quantitative Results

This section presents the quantitative data analysis of the hypotheses presented in Section II on the data of the questionnaires (see Section III); subsections IV-A to IV-F discuss the six hypotheses H_0^A to H_0^F.

The initial data-set included information about 48 projects; we removed one of them, as it consisted of a questionnaire with clearly bogus answers, leaving us with data about 47 projects. Some questionnaires were incomplete, in that answers to some questions were missing. The analysis that follows excludes the missing answers for each question; therefore, the number of projects evaluated may vary from question to question.

We performed analyses for each of the hypotheses H_0^A to H_0^F, in order to determine whether our data showed a statistically significant difference between answers about projects using agile processes and answers about projects using structured processes.

The questionnaire consisted of multiple-choice questions; given the nature of the available choices, we should consider the emerging data as *ordinal* but not interval-scale. For each answer, we visually inspected the distribution of data and we performed a Shapiro-Wilk normality test; none of them gave evidence to consider the underlying distributions as normal. In all, the presence of ordinal data and non-normal distributions suggests to deploy the Mann-Whitney-Wilcoxon U test, a non-parametric statistical hypothesis test [2], [31].

Each hypothesis H_0^X, for $X = A, \ldots, F$, refers to a certain quantity (overall success, importance, motivation, etc.), measured by the corresponding random variables AG^X and ST^X—respectively in agile and structured projects. For example, H_0^C tests the *motivation* of teams, hence AG^C models team motivation in agile projects and ST^C models team motivation in structured projects. With this notation, the null

hypothesis H_0^X is expressible as:

$$H_0^X \quad : \quad \mathbb{P}\left(AG^X > ST^X\right) = \mathbb{P}\left(ST^X > AG^X\right) ;$$

that is, the probability that random samples of quantity X are larger in agile projects than in structured projects equals the probability that the samples are larger in structured projects than in agile projects. Correspondingly, the alternative hypothesis H_1^X is that there is a difference in probability, that is:

$$H_1^X \quad : \quad \mathbb{P}\left(AG^X > ST^X\right) \neq \mathbb{P}\left(ST^X > AG^X\right) .$$

Notice that we do not directly test for differences in medians and means of the random variables AG^X and ST^X using the U test, because that would require that the underlying distributions of AG^X and ST^X have the same shape; however, our data does not meet this requirement.

Given a significance level $\alpha = 0.05$, the U test gives a probability p that the data supports the null hypothesis: if $p < \alpha$, the data gives evidence to reject the null hypothesis, if $p > \alpha$, one can *not* reject the null hypothesis. Figure 2 shows the overall picture: in summary, we never reject the null-hypothesis, that is there is no evidence to distinguish between using agile vs. structured processes; the following subsections describe the results for each hypothesis in detail.

A. Overall Success

For each project out of a total of 47, question A asks to rank the *overall success* of the project on a scale from 1 to 10, where 1 is "complete failure" and 10 is "full success". The answers were as follows:[3]

	#	Median	Min/Max	Mean Rank	Mean	Std.dev
Agile	29	8	7/10	23.14	8.34	1.078
Struc.	18	9	5/10	25.39	8.44	1.381

The U test gives $U = 236$ and $p = 0.571$; as $p \gg \alpha$, we do not reject the null hypothesis H_0^A.

B. Project Importance

For each project out of a total of 47, question B asks to rank the *importance* of the project for the customer on a scale from 1 to 10, where 1 is "unimportant" and 10 is "very critical". The answers were as follows:

	#	Median	Min/Max	Mean Rank	Mean	Std.dev.
Agile	29	9	4/10	23.95	8.69	1.417
Struc.	18	9	7/10	24.08	8.83	1.043

The U test gives $U = 259$ and $p = 0.973$; as $p \gg \alpha$, we do not reject the null hypothesis.

C. Team Motivation

For each project out of a total of 44, question C asks to rank the *motivation* of the team working on the project on a scale from 1 to 10, where 1 is "not at all" and 10 is "very much". The answers were as follows:

	#	Median	Min/Max	Mean Rank	Mean	Std.dev.
Agile	28	8.5	5/10	22.3	8.61	1.449
Struc.	16	9.5	4/10	22.84	8.5	1.932

The U test gives $U = 218$ and $p = 0.887$; as $p \gg \alpha$, we do not reject the null hypothesis.

[3]In all following tables, we report sample mean and standard deviation even if the data is on an ordinal scale.

D. Economic Savings

For each project out of a total of 31, question D asks to rank the estimated *economic savings*—achieved with distributed development compared to entirely local development—on the following scale:[4]

SAVINGS	RANK
Saved more than 50%	6
Saved 25% to 50%	5
Saved 10% to 25%	4
About even -10% to 10%	3
Lost 10% to 25%	2
Lost more than 25%	1
Don't know	–

The answers were as follows:

	#	Median	Min/Max	Mean Rank	Mean	Std.dev.
Agile	15	5	4/6	18	5	0.655
Struc.	16	4.5	4/6	14.13	4.69	0.793

The U test gives $U = 90$ and $p = 0.247$; as $p \gg \alpha$, we do not reject the null hypothesis.

E. Amount of Real-Time Communication

For each project out of a total of 47, question E asks to rank the estimated amount of *real-time communication*—occurring during project development—on the following scale:

R-T COMMUNICATION	RANK
more than 30 times per year	7
15 to 30 times per year	6
10 to 14 times per year	5
6 to 9 times per year	4
3 to 5 times per year	3
1 to 2 times per year	2
Never	1

The answers were as follows:

	#	Median	Min/Max	Mean Rank	Mean	Std.dev.
Agile	29	3	1/7	23.62	4.1	2.273
Struc.	18	4	1/7	24.61	4.33	2.376

The U test gives $U = 250$ and $p = 0.805$; as $p \gg \alpha$, we do not reject the null hypothesis.

F. Amount of Asynchronous Communication

For each project out of a total of 47, question E asks to rank the estimated amount of *asynchronous communication*—occurring during project development—on the following scale:

ASYNCHRONOUS COMMUNICATION	RANK
10 or more hours per week	5
6 to 9 hours per week	4
3 to 5 hours per week	3
1 to 2 hours per week	2
$<$ 1 hour per week	1

The answers were as follows:

	#	Median	Min/Max	Mean Rank	Mean	Std. dev.
Agile	29	3	1/5	24.48	3.38	1.293
Struc.	18	3	2/5	23.22	3.22	0.943

The U test gives $U = 247$ and $p = 0.76$; as $p \gg \alpha$, we do not reject the null hypothesis.

[4]"Don't know" answers were excluded from the analysis.

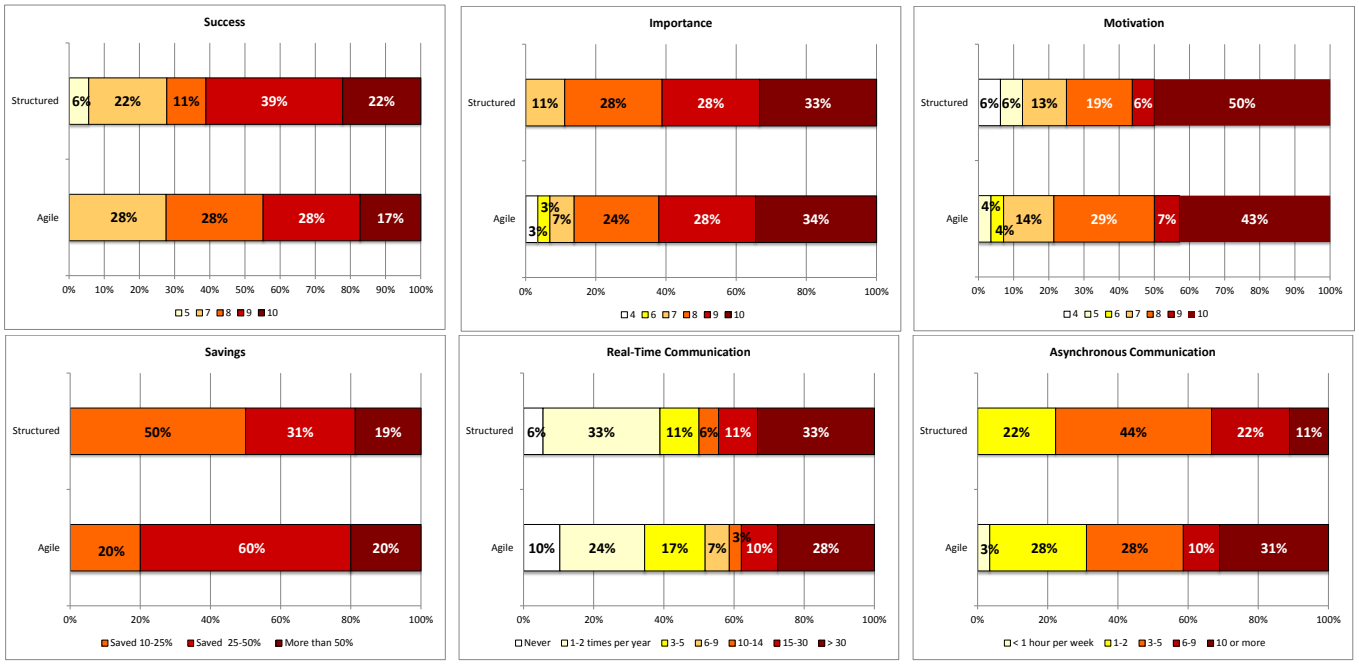

Fig. 1. Bar charts for the data corresponding to hypotheses H_0^A to H_0^F; see Section IV-A to IV-F for a detailed analysis.

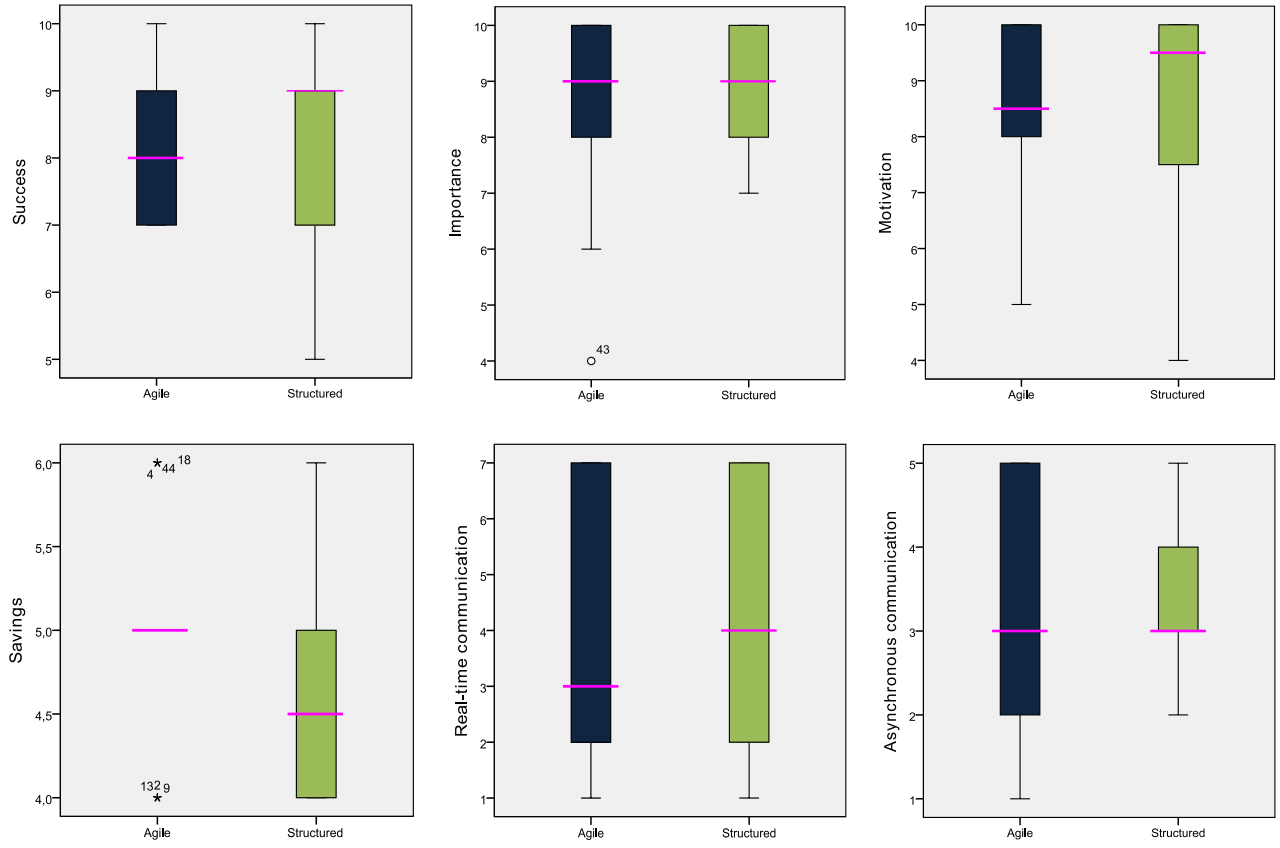

Fig. 2. Boxplots for the data corresponding to hypotheses H_0^A to H_0^F; see Section IV-A to IV-F for a detailed analysis.

V. Qualitative Results

This section discusses various aspects of distributed software development that emerged from all the data we collected: the costs of nearshore vs. offshore development, the average success and quality achieved in distributed projects, the role of personnel skills and of communication patterns, the criteria for choosing development process, the typical team size, and various critical issues.

Unlike the results of Section IV, the data is mostly qualitative and it deals with aspects complementary to the choice of development process that also affect the outcome and success of projects. Sections V-1 through V-5 report data from the interviews only (for a total of 18 projects, see Section III), whereas the other Sections V-6 through V-8 also include data from the questionnaires (totaling 66 projects).

1) Costs of Nearshore vs. Offshore: The conventional wisdom is that nearshore development (where developers work close to customers, in terms of time zones and distance) makes team coordination and collaboration with customers easier, but it is significantly more expensive than offshore development (which can use less expensive developers in countries such as India and China). The interviews with Swiss companies representatives revealed, however, that most companies that prefer nearshore development do it for legal reasons and because it reduces the amount of traveling, rather than directly to reduce costs.

In fact, our data does not show correlation between costs and location: the overall costs in nearshore development are not necessarily higher than in offshore development. While the salaries of programmers in Asia is typically between $1/5$ and $1/3$ of the corresponding positions in Switzerland, the overall project costs are also affected by factors such as productivity, communication and management overhead, and various costs for setting up and maintaining offices, which weakens the dependence between total costs and location.

Similarly, our data shows no significant cost differences between globally distributed projects (where members of the same development team operate in different locations) and outsourced projects (where management is in a location, and all development takes place in a different location): compared with purely local development, globally distributed projects reported savings for an average 28% and a standard deviation of 11%[5]; outsourced projects reported very similar savings for an average of 33% and a standard deviation of 11%[6]. The overhead (for communication, management, and office costs) is also almost identical in globally distributed and in outsourced projects, ranging between 35% and 45%. For example, creating a new unit in an outsourced location requires 2 to 5 months to setup the office, plus another 3 months to become productive; the investment pays back after 3-4 years on average.

2) Project Success: Out of the 18 projects surveyed in the interviews: 11 are considered "complete success"; 5 are "overall success" which, however, suffered non-trivial problems during development. The major sources of problems and difficulties were: unqualified personnel, cultural and communication difficulties, deficiencies of the infrastructure, insufficient interaction among units. The major sources of success were: skilled personnel and effective team building (after a solid team is established, the members will work proficiently even if distributed in different locations).

3) Project Quality: The overall quality of the majority of projects was reported as "good" or better, but our interviews revealed that development problems related to quality are not uncommon, especially at the beginning of projects (reported in over 50% of the projects). It seems that quality correlates positively with timeliness: late projects are unlikely to achieve a good quality; nonetheless, compromises on quality are often accepted to meet deadlines. We observed no significant differences between nearshore and offshore development.

4) Personnel Skills: Personnel skills are a major factor of project success. Most interviewees think that personnel skills decrease with distance: the most skilled personnel is in Switzerland, followed by the personnel in nearshore locations (typically, Eastern Europe), and then by personnel in offshore locations (India and China). The deterioration of skills is attributed to difficulties in communication and collaboration, and more generally to the challenges introduced by distributed software engineering.

5) Communication Patterns: Effective communication is another major factor of project success, and in fact 13 out of 18 projects required a weekly (virtual) meeting among all project members. Figure 3 shows the means of communication used in the 18 projects, classified by their richness (i.e., perceived effectiveness) and synchrony. Most of the communication among developers takes place using instant messaging, which is preferred over voice calls and face-to-face communication because it helps bypass communication obstacles—for example, strong accents—and because it is a good compromise between real-time and asynchronous communication. Time zones were not reported as an issue for communication.

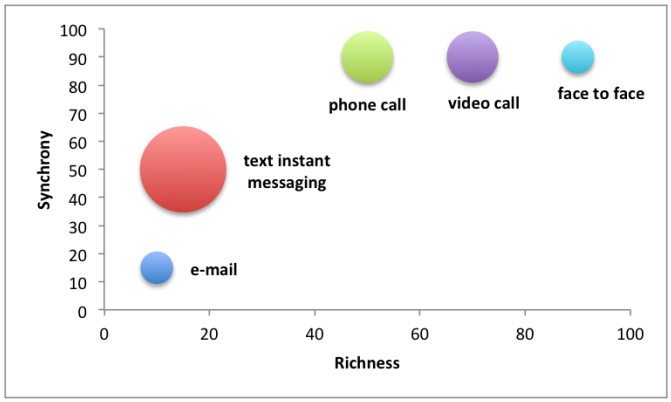

Fig. 3. Communication means categorized by richness and synchrony.

6) Development Process: Does the choice of development process used (agile vs. structured) affect the way the various

[5]Data about 8 projects.
[6]Data about 10 projects.

activities (from requirements to maintenance) are assigned to onshore rather than offshore units? Figure 4 shows[7] only a slight difference in the system design (where offshoring is higher with agile processes) and unit testing (where offshoring is higher with structured processes) phases; the data is about 66 projects.

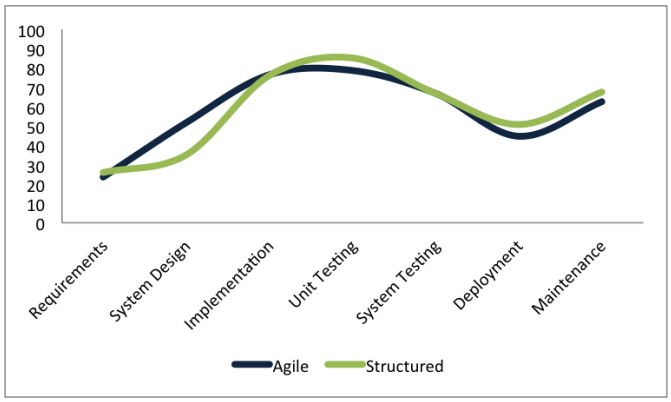

Fig. 4. Allocation (percentage of activity assigned to offshore units) of development activities in agile and structured processes.

7) Team Size: Figure 5 shows the team size of the 66 projects of our study. On average, agile projects have smaller teams than structured projects: most agile projects deploy teams of 30 people or fewer, whereas structured projects tend to have larger teams, and 5 of them even used teams including more than 120 people.

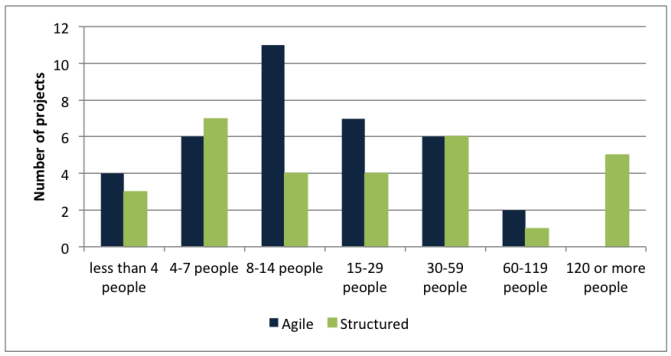

Fig. 5. Team size in agile and structured projects.

8) Critical Issues: Figure 6 displays the main problematic aspects emerged in distributed projects, classified in agile and structured projects: communication problems; cultural differences; management problems; and problems with keeping projects on schedule.

Agile projects seem to have experienced fewer communication problems than structured projects (28% of the agile project had "medium" communication problems, against 47% of the structured projects). Cultural differences often are only a moderate problem in both agile and structured projects, with over 60% of structured and 40% of agile projects reporting to problem at all in this area. Difficulties in project management seem to be more frequent in structured projects: 21% of the structured (resp. 14% of the agile) projects reported "medium" management problems and 7% of the structured (resp. 0% of the agile) reported "severe" management problems. The difficulties encountered in keeping projects on schedule are very similar in structured and agile projects.

[7]We did not provide a definition of the various activities to interviewees.

VI. Threats to Validity

We discuss the threats to validity in two categories: internal and external. Internal validity refers to whether the study supports the findings. External validity refers to whether the findings can be generalized.

A. Internal Validity

A number of threats to internal validity may surface in studies based on surveys and interviews. Interviews feature a trade-off between minimizing "interviewer effects" [11]—the interviewer giving subtle clues about preferred answers—and ensuring quality of answers. The last author of this paper carried out the interviews using brief and schematic questions (like those used in the questionnaires) in order to minimize interviewer effects. With some interviewees, however, this resulted in insufficiently clear or vague answers. For example, several interviewees responded to multiple-choice questions with open answers that did not stick to the available choices. In these cases, the interviewer sometimes tried to improve the quality of answers by using a more dialectical style of inquiry, possibly at the risk of introducing interviewer effects. We cannot guarantee that the optimal trade-off was achieved in all cases.

Another potential threat to internal validity is the risk that the granularity of multiple-choice answers (in questionnaires and interviews) is too coarse. In particular, the dichotomy between agile and structured processes does not allow for "hybrid processes", which may be used in practice. Also, the different backgrounds of study participants could have resulted in different interpretations of the same ordinal scales (for example, the "overall success" of the same project would be ranked at a different level on a scale of 1 to 10 by different individuals). Finally, the absence of a control group and the fact that we did not have direct access to data about projects make it impossible to evaluate the genuineness of the data collected with interviews and questionnaires: we do not know how precise (and objective) the assessment of quantities such as "overall success" or "economic savings" was, nor whether processes classified as "agile" (resp. "structured") properly followed the agile (resp. structured) principles and practices.

While these threats are inherent in studies based on questionnaires and interviews (like this paper's), we have reasons to assume they had only limited impact. First, participants were asked to report on a single (agile or structured) GSD project, hence they had a chance to select one "champion" that unambiguously fits the agile or structured paradigm, rather than a hybrid. Furthermore, the differences among agile (or

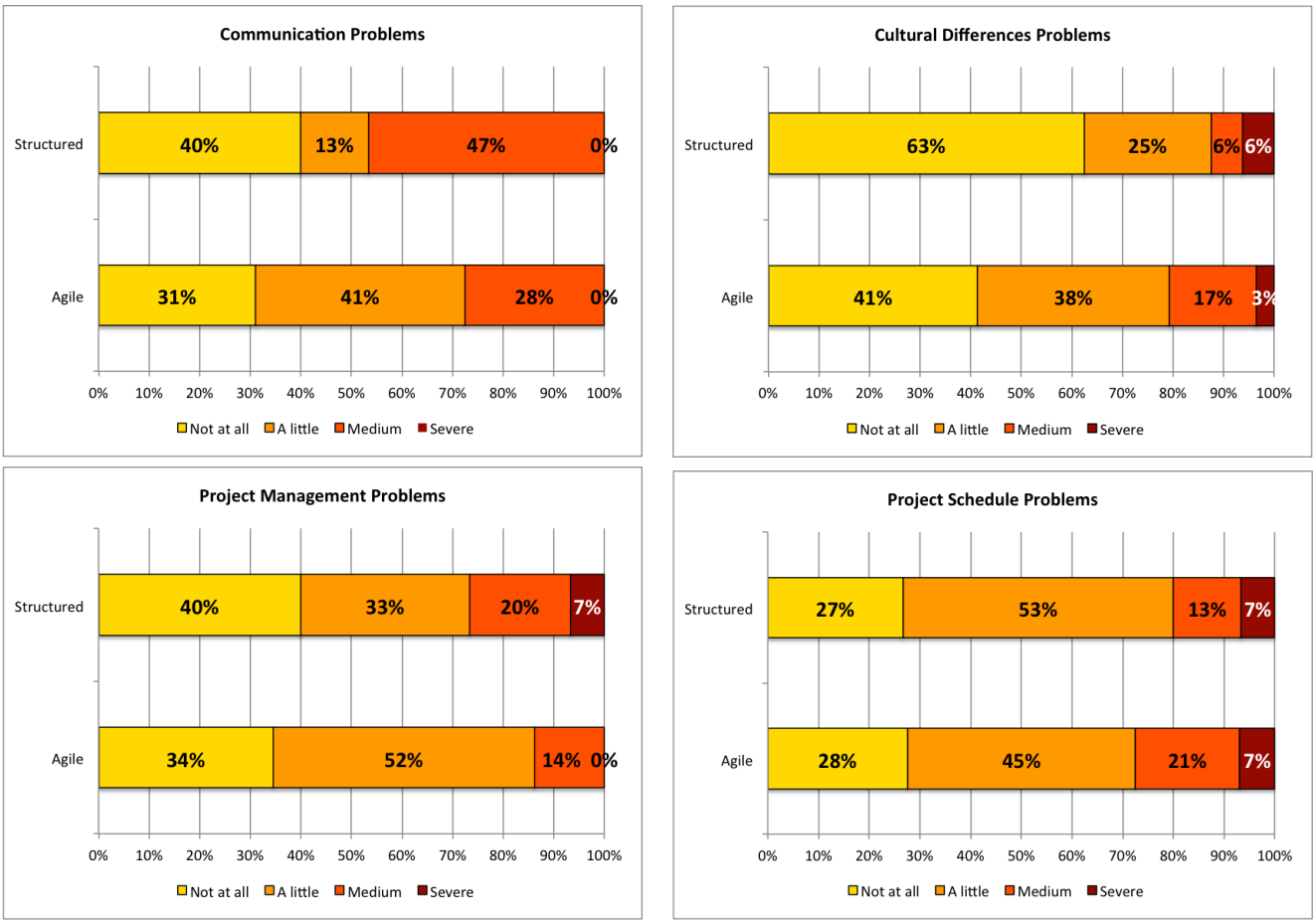

Fig. 6. Critical issues in globally distributed projects, classified in agile and structured: communication problems (top left); cultural differences (top right); management problems (bottom left); problems with keeping to the schedule (bottom right).

among structured) processes are likely to be small compared to the differences between any one agile and any one structured process. In fact, agile processes emerged as a reaction [16] against mainstream development practices, hence the "agile vs. structured" classification is reasonably robust. About the remaining threats, the rank and experience of most participants to the study positively reflect on the chances of having obtained quantitative estimates of fair and uniform quality.

B. External Validity

The major threats to external validity for studies based on surveys come from insufficient *coverage* or *responsiveness*. Coverage measures to what extent the data-set supports generalization of the findings. Responsiveness quantifies the amount of "non-respondents", that is contacts who received a questionnaire but did not reply with meaningful data. A low responsiveness is a threat to external validity, as non-respondents may exhibit some characteristics relevant for the study and underrepresented among respondents.

In our study, we sent out questionnaires to over 60 contacts and we received 48 replies (one was discarded). Even though we do not know for sure whether some of the contacts forwarded the questionnaires to others (the replies were anonymous for confidentiality reasons), the figures seem to indicate a low risk of bias due to lack of responsiveness.

Assessing the coverage is harder for our study. The data collected through interviews was limited to projects developed by Swiss companies; the online questionnaires reached 18 different companies worldwide, but the vast majority of these companies have their headquarters in Europe or North America. We cannot prove that the experience of our respondents is representative of the entire population of distributed software projects, which may affect the generalizability of our findings.

VII. Related Work

This section presents related work in three areas: empirical studies on agile processes in local development settings (Section VII-A), on the issues and challenges raised by distributed development (Section VII-B), and on applying agile processes in distributed settings (Section VII-C). Section I lists general references on development processes and their role in the software development life-cycle.

A. Agile Processes for Local Development

The effectiveness of agile processes in collocated projects has been widely investigated empirically. Müller and

Tichy [18] studied the outcome of extreme programming practices in a programming course for graduate students. Their data characterizes the performance of 12 students grouped in pairs, each pair carrying out a 40-hour programming project and 3 smaller assignments (totaling about 600 minutes of work). The study showed that groups using extreme programming produced high-quality code, but it also exposed some difficulties in applying this agile methodology at best in practice.

Hulkko and Abrahamsson [15] also analyzed extreme programming practices, and in particular pair programming. Their study involved both students and professional programmers in a controlled setting, where they developed implementations of size up to 8000 lines of code. The results provided no evidence that pair programming practices improve productivity or the quality of the produced code, compared with programmers working solo.

Bergel and Nagappan [3] surveyed the results of pair programming practices at Microsoft. Professional programmers, testers, and manager with about 10 years of experience took part in the survey; the majority reported that pair programming works well for them and produces higher-quality code.

Bhat and Nagappan [4] conducted an empirical study about test-driven development—another practice of extreme programming—at Microsoft. The study compared test-driven development against more traditional practices, showing an increase in code quality but also in development time (by about 15%) with the adoption of test-driven development.

Nawrocki et al. [19] compared development with extreme programming against development following CMM level 2. Their study, targeting university students, revealed that CMM implementations are more stable and contain fewer bugs, but programmers following CMM practices perceive their job as more tedious.

B. Empirical Studies on Distributed Development

Empirical studies on globally distributed development have analyzed different aspects, including communication patterns, the effect of time zones, and achievable quality and productivity.

Several studies focused on the amount and type of communication required by distributed projects. For example, Allen [1] reported that the frequency of communication among engineers whose offices are more than 30 meters apart drops to almost the same level as that of engineers separated by several miles. In the same vein, Carmel [7] identified *loss of communication* as one of four major risk factors that can lead to the failure of distributed software projects.

Herbsleb and Mockus [13] analyzed the impact of globally distributed development on the amount of communication needed to agree on and implement requests for modifications of existing implementations. They found that, when developers are geographically distributed, the overall time increases by a factor of 2.5 on average. If, however, the effect of other variables such as the number of people involved in a task and the size of the required modifications is properly taken into account, the differences in communication time between distributed and collocated teams are no longer significant. Other findings were that communications is much more frequent among collocated than among remote developers; and that the size of the social network (i.e., the number of colleagues a developer ever interacts with) is significantly smaller for programmers working in distributed teams.

In previous work of ours [21], we studied the effect of time zones and locations on communication within distributed teams; we performed the study as part of our DOSE [23], [22] university course. We found that the amount of communication is larger in two-location projects than in projects distributed across three locations; and that it decreases the more time zones separate the developers of a distributed team.

Other studies of distributed development focus on achievable quality. For example, Bird et al. [5] present a case study on the development of Windows Vista, comparing the failures of components developed by distributed teams with those of components developed by collocated teams. Their results show no significant differences in the two cases. Spinellis [30] examined how distributed development affects defect density, coding styles, and productivity in an open source project. He found that there is little or no correlation between geographic distribution and these quality metrics.

None of these studies targeted the type of processes adopted in distributed development, which is instead the focus of the present paper. Taking a different angle, Cataldo et al. [8] analyzed the mutual impact of process maturity and distribution on project quality. Their study classified companies according to their CMMI level, showing that the advantages of processes at higher maturity levels decrease with the distribution of teams. They do not compare structured and agile processes, as the present paper does.

C. Agile Processes for Distributed Development

There are some clear challenges involved in applying agile processes—such as Scrum and extreme programming—to distributed projects. Researchers have proposed changes to agile practices that render them applicable in globally distributed settings. Correspondingly, the remainder of this section summarizes a few empirical studies that have analyzed distributed projects using agile methods. The present paper complements such work, as it compares the impact of agile methods against that of structured methods for distributed development.

Layman et al. [17] studied the communication practices of extreme programming teams distributed between the USA and the Czech Republic. The developers created an environment for informal communication in a distributed setting, which helped develop user-story specifications and solve technical problems quickly and efficiently. Face-to-face communication was effectively replaced by other means of communication.

Paasivaara et al. [25] studied how Scrum is performed in distributed projects. Their interview of 19 team members in 3 companies in Finland identified best practices, benefits, and

challenges involved in the various Scrum activities such as "daily scrum", "sprints", and "sprint planning meeting".

Sureshchandra et al. [32] developed another evaluation of agile processes, targeting only one company. Besides surveying best practices and lessons learned, they make a brief comparison of the productivity (measured as lines of code over time) of 15 projects using agile and non agile processes, and they report a 10% increase in the productivity with agile methods.

VIII. Conclusions and Future Work

We presented a case study analyzing the impact of software processes on distributed development. We have examined a total of 66 industry projects, classified them into agile and structured, and evaluated the correlation between process type and success, importance, and economic savings of projects, team motivation, and real-time and asynchronous communication. The collected data suggests that the correlations between process type and these other measures are negligible and without statistical significance: choosing an agile rather than a structured process does not appear to be a crucial decision for globally distributed projects.

As future work, we plan to investigate various agile and structured projects in more detail, to determine which agile practices are followed in practice, and to identify common practices across different projects. We also plan to study how developers write programs in distributed settings and, in particular, how they communicate and coordinate API changes. This study will be performed with our CloudStudio IDE [20], which supports software development in the cloud with real-time concurrent editing.

Acknowledgments: The authors thank all the participants to the study, and the anonymous reviewers for useful comments. This research has been partially funded by the Gebert-Rüf Stiftung.

References

[1] T. J. Allen. *Managing the Flow of Technology*. MIT Press, 1977.

[2] A. Arcuri and L. Briand. A Pactical Guide for Using Statistical Tests to Assess Randomized Algorithms in Software Engineering. In *ICSE*, pages 1–10. ACM, 2011.

[3] A. Begel and N. Nagappan. Pair Programming: What's in it for Me? In *ESEM*, pages 120–128. ACM, 2008.

[4] T. Bhat and N. Nagappan. Evaluating the Efficacy of Test-Driven Development: Industrial Case Studies. In *ISESE*, pages 356–363. ACM, 2006.

[5] C. Bird, N. Nagappan, P. Devanbu, H. Gall, and B. Murphy. Does Distributed Development Affect Software Quality? An Empirical Case Study of Windows Vista. In *ICSE*, pages 518–528. IEEE, 2009.

[6] B. Boehm and R. Turner. *Balancing Agility and Discipline: A Guide for the Perplexed*. Addison-Wesley, 2004.

[7] E. Carmel. *Global Software Teams: Collaborating Across Borders and Time Zones*. Prentice Hall PTR, 1999.

[8] M. Cataldo and S. Nambiar. On the Relationship between Process Maturity and Geographic Distribution: an Empirical Aanalysis of their Impact on Software Quality. In *ESEC/FSE*, pages 101–110. ACM, 2009.

[9] D. Cohen, M. Lindvall, and P. Costa. An Introduction to Agile Methods. *Advances in Computers*, pages 1–66, 2004.

[10] J. A. Espinosa, N. Nan, and E. Carmel. Do Gradations of Time Zone Separation Make a Difference in Performance? A First Laboratory Study. In *ICGSE*, pages 12–22. IEEE, 2007.

[11] F. J. Fowler and T. W. Mangione. *Standardized Survey Interviewing: Minimizing Interviewer-Related Error*. Sage Publications Inc., 1989.

[12] C. Ghezzi, M. Jazayeri, and D. Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, 2nd edition, 2002.

[13] J. Herbsleb and A. Mockus. An Empirical Study of Speed and Communication in Globally Distributed Software Development. *IEEE Transactions on Software Engineering*, 29(6):481–494, 2003.

[14] J. D. Herbsleb, A. Mockus, T. A. Finholt, and R. E. Grinter. Distance, dependencies, and delay in a global collaboration. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, CSCW '00, pages 319–328, New York, NY, USA, 2000. ACM.

[15] H. Hulkko and P. Abrahamsson. A Multiple Case Study on the Impact of Pair Programming on Product Quality. In *ICSE*, pages 495–504. ACM, 2005.

[16] K. Beck et al. Manifesto for Agile Software Development. http://www.agilemanifesto.org, Apr. 2001.

[17] L. Layman, L. Williams, D. Damian, and H. Bures. Essential Communication Practices for Extreme Programming in a Global Software Development Team. *Information and Software Technology*, 48(9):781–794, 2006.

[18] M. M. Müller and W. F. Tichy. Case Study: Extreme Programming in a University Environment. In *ICSE*, pages 537–544. IEEE, 2001.

[19] J. R. Nawrocki, B. Walter, and A. Wojciechowski. Comparison of CMM Level 2 and eXtreme Programming. In *ECSQ*, pages 288–297. Springer-Verlag, 2002.

[20] M. Nordio, H.-C. Estler, C. A. Furia, and B. Meyer. Collaborative Software Development on the Web, 2011. arXiv:1105.0768v3.

[21] M. Nordio, H.-C. Estler, B. Meyer, J. Tschannen, C. Ghezzi, and E. Di Nitto. How do Distribution and Time Zones affect Software Development? A Case Study on Communication. In *ICGSE*. IEEE, 2011.

[22] M. Nordio, C. Ghezzi, B. Meyer, E. D. Nitto, G. Tamburrelli, J. Tschannen, N. Aguirre, and V. Kulkarni. Teaching Software Engineering using Globally Distributed Projects: the DOSE course. In *CTGDSD*. ACM, 2011.

[23] M. Nordio, R. Mitin, and B. Meyer. Advanced Hands-on Training for Distributed and Outsourced Software Engineering. In *ICSE*, pages 555–558. ACM, 2010.

[24] M. Nordio, R. Mitin, B. Meyer, C. Ghezzi, E. D. Nitto, and G. Tamburrelli. The Role of Contracts in Distributed Development. In *SEAFOOD*, volume 35 of *LNBIP*, pages 117–129. Springer, 2009.

[25] M. Paasivaara, S. Durasiewicz, and C. Lassenius. Using Scrum in Distributed Agile Development: A Multiple Case Study. In *ICGSE*, pages 195 –204. IEEE, 2009.

[26] S. L. Pfleeger and J. Atlee. *Software Engineering: Theory and Practice*. Prentice Hall, 3rd edition, 2005.

[27] R. Pressman. *Software Engineering: A Practitioner's Approach*. McGraw-Hill, 7th edition, 2009.

[28] N. Ramasubbu and R. Balan. Globally Distributed Software Development Project Performance: An Empirical Analysis. In *ESEC/FSE*, pages 125–134. ACM, 2007.

[29] N. Ramasubbu, M. Cataldo, R. K. Balan, and J. D. Herbsleb. Configuring Global Software Teams: A Multi-Company Analysis of Project Productivity, Quality, and Profits. In *ICSE*, pages 261–270. ACM, 2011.

[30] D. Spinellis. Global Software Development in the FreeBSD Project. In *GSD*, pages 73–79. ACM, 2006.

[31] P. Sprent and N. Smeeton. *Applied Nonparametric Statistical Methods*. Texts in Statistical Science. Chapman & Hall/CRC, 2007.

[32] K. Sureshchandra and J. Shrinivasavadhani. Adopting Agile in Distributed Development. In *ICGSE*, pages 217 –221. IEEE, 2008.