Empir Software Eng (2014) 19:1197–1224
DOI 10.1007/s10664-013-9271-y

Agile vs. structured distributed software development: A case study

Hans-Christian Estler · Martin Nordio · Carlo A. Furia · Bertrand Meyer · Johannes Schneider

Published online: 15 August 2013

Abstract In globally distributed software development, does it matter being agile rather than structured? To answer this question, this paper presents an extensive case study that compares agile (Scrum, XP, etc.) vs. structured (RUP, waterfall) processes to determine if the choice of process impacts aspects such as the overall success and economic savings of distributed projects, the motivation of the development teams, the amount of communication required during development, and the emergence of critical issues. 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 the effects on project quality.

Communicated by: Erran Carmel, Rini van Solingen, Rafael Prickladnicki and Filippo Lanubile

A preliminary version of this paper featured at the 7th International Conference on Global Software Engineering (Estler et al. 2012).

Schneider's work conducted while affiliated with ETH Zurich.

H.-C. Estler (✉) · M. Nordio · C. A. Furia · B. Meyer
Chair of Software Engineering, ETH Zurich, Switzerland
e-mail: christian.estler@inf.ethz.ch

M. Nordio
e-mail: martin.nordio@inf.ethz.ch

C. A. Furia
e-mail: carlo.furia@inf.ethz.ch

B. Meyer
e-mail: bertrand.meyer@inf.ethz.ch

J. Schneider
IBM Research, Zurich, Switzerland
e-mail: joh@zurich.ibm.com

Keywords Distributed software development · Outsourcing · Agile · Empirical study

1 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.

1.1 Empirical Analyses of Globally Distributed Software Development

Globally distributed software development 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. Globally distributed software development 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 globally distributed software development, 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 globally distributed software development projects", where "X" is a practice, method, or technique, and "quality" may refer to different aspects such as timeliness, customer satisfaction, cost effectiveness, or the absence of problems. Examples of globally distributed software development issues investigated empirically along these lines include the usage of contracts for API design (Nordio et al. 2009), the effect of time zones on various phases of development (Herbsleb et al. 2000; Espinosa et al. 2007; Nordio et al. 2011a) and on productivity and quality (Ramasubbu and Balan 2007; Bird et al. 2009), and the impact of geographic dispersion on several quality metrics (Ramasubbu et al. 2011).

1.2 Goals of this Study: Impact of Development Processes

The case study presented in this paper focuses on *development processes* (Ghezzi et al. 2002; Pressman 2009; Pfleeger and Atlee 2005) to find out whether the choice of process has a significant impact on qualities such as programmer productivity and development cost-effectiveness in globally distributed software development. To our knowledge, this is one of very few empirical studies that explicitly investigates the impact of development processes on globally distributed software development.

1.3 Software Development Processes: Structured vs. Agile

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 (Ghezzi et al. 2002; Pressman 2009; Pfleeger and Atlee 2005) 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 (Beck et al. 2001; Cohen et al. 2004), such as Scrum or eXtreme Programming (XP), emphasize the importance of effective informal communication among developers, and of iterative improvement of implementations driven by use-case scenarios, 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 (Müller and Tichy 2001; Hulkko and Abrahamsson 2005; Begel and Nagappan 2008; Nawrocki et al. 2002; Boehm and Turner 2004): for example, for applications whose requirements are accurately known and not subject 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.

1.4 The Role of Processes in Distributed Development

The present paper's study re-considers the "structured vs. agile" dichotomy in the context of globally distributed software development, and tries to understand whether one of the two development approaches emerges as more appropriate to organize software development carried out by globally distributed teams. The facts learned about non-distributed contexts may not apply to 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 (Beck et al. 2001):

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

In contrast, 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 (Sureshchandra and Shrinivasavadhani 2008; Paasivaara et al. 2009, 2010).

[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.

The question remains, however, of what are the relative merits of structured and agile processes for globally distributed software development, 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 customer, cost-effectiveness, developer motivation, amount of personal communication, and emerge of several problematic aspects. As we discuss in detail in the rest of the paper, the data collection was designed so as to reduce potential threats to validity, and specifically the intrinsic fuzziness of ordinal-answer questionnaires (see Section 3) and the interviewer effect of structured interviews (see Section 7):

– The questionnaire (see Table 1) included, as much as possible, quantitative descriptions of the possible answers on a scale (for example, "answer 3 corresponds to 3–5 h per week"). This helps align answers coming from different participants to a common gauge, so that comparing them is meaningful.
– All participants to the study have considerable experience (managers or senior engineers) and reported data about a recently completed single project. This gives us confidence that the participants were aware of the possible pitfalls of reliably reporting complex data, and that they could report on completed projects, where the differences in process and outcome were sufficiently well defined (as opposed to ongoing projects with unclear developments).
– The quantitative analysis only uses data from the questionnaires. We still rely on the interviews to report qualitative and anecdotal data (see Section 5), which corroborate and enrich the overall picture about distributed development.
– The number of projects about which we collected data is fairly large for this type of study (see a comparison with related work in Section 8); therefore, correlations should easily emerge if present at all.

1.5 Summary of Results

The bulk of the 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 globally distributed software development, and the sources of significant differences in project outcome should be sought in other project characteristics.

These results should not be misread as suggesting that development processes are irrelevant and bear no impact on project outcome. The real take-home lesson is that the development process is not an independent variable, and hence its choice cannot single-handedly determine the successful or unsuccessful outcome of a project. Single experiences include both great success stories and utter failures with either structured

Table 1 Questions from the questionnaire used to collect data for hypotheses H_0^A–H_0^P

Variable	Question	Answers and numeric mapping
A	Rate the success of the last completed project.	1 (Complete Failure) . . . 10 (Full success)
B	How important was the last completed project for the customer?	1 (Unimportant) . . . 10 (Very critical)
C	How motivated were you working in an outsourcing project?	1 (Not at all) . . . 10 (Very much)
D	What was the financial outcome for the customer compared to onshore development?	– (Don't know) 1 (Lost more than 25%) 2 (Lost 10 % to 25 %) 3 (About even: −10 to 10%) 4 (Saved 10–25 %) 5 (Saved 25–50 %) 6 (Saved more than 50 %)
E	How often did you have real-time communication with the outsourcing partner?	1 (Never) 2 (1 to 2 times per year) 3 (3 to 5 times per year) 4 (6 to 9 times per year) 5 (10 to 14 times per year) 6 (15 to 30 times per year) 7 (more than 30 times per year)
F	How often did you have asynchronous communication with the outsourcing partner?	1 ($<$ 1 hour per week) 2 (1 to 2 hours per week) 3 (3 to 5 hours per week) 4 (6 to 9 hours per week) 5 (10 or more hours per week)
G	Was communication due to distance a problem?	– (Don't know)
H	Were cultural differences a problem?	1 (Not at all)
I	Was project management a problem?	2 (A little)
J	Was loss or fluctuations of know-how a problem?	3 (Medium)
K	Was a shortage of labor skills a problem?	4 (Severe)
L	Was reading specifications a problem?	
M	Was writing specifications a problem?	
N	Were personal conflicts a problem?	
O	Was keeping the project schedule a problem?	
P	Was protecting intellectual property a problem?	

For each measured variable in the first column, the questionnaire formulates a question (second column) and an ordinal range of answers with quantitative references (third column, also given to participants). Each question also included the option not to answer or agile practices. The choice of development process is thus something to be considered with great care, but based on the characteristics of a project and of the development team working on it, as well as on its overall goals—not in a vacuum based on a priori expectations for one-size-fits-all blue-sky solutions.

1.6 Outline

The rest of the paper is organized as follows. Section 2 presents the research questions investigated in the case study. Section 3 describes the data collection process and the research methodology. Section 4 presents the quantitative results of the study, whereas Section 5 is devoted to a somewhat informal discussion of other aspects for which only qualitative data is available. Section 6 draws the big picture of the study from a practical standpoint. Sections 7 and 8 respectively describe threats to validity and related work. Section 9 summarizes and describes future work.

2 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 two overall research questions, the first focusing on project outcome and the second focusing on the emergence of problematic aspects.

RQ1: In software development carried out in globally distributed settings, what is the impact of adopting structured vs. agile processes on the overall *success* (A), *importance* for customers (B), team *motivation* (C), *cost-effectiveness* (D), and amount of *real-time* (E) and *asynchronous communication* (F)?

RQ2: In software development carried out in globally distributed settings, what is the impact of adopting structured vs. agile processes on the emergence of *communication* difficulties (G), *cultural* differences (H), ineffective project *management* (I), loss or fluctuation of *know-how* (J), shortage of labor *skills* (K), ineffective reading or writing of *documentation*[2] (L, M), interpersonal *conflicts* (N), difficulties in keeping to the project *schedule* (O), and in protecting *intellectual property* (P)?

The choice of project outcome aspects A–P reflects the major dimensions studied in the research literature on distributed development (reviewed in Section 8). The specific questions asked in the questionnaires and interviews (see Section 3.1 and Table 1) outline the definitions we assumed for aspects A–P targeted by the research questions; in the simplest cases, we can just assume dictionary definitions.

[2]For structured processes, the term "documentation" mainly denotes requirement specifications; for agile processes, it mainly denotes use-case scenarios and test cases.

2.1 Hypotheses for RQ1 and RQ2

For each aspect A–F of RQ1 (overall success, cost-effectiveness, etc.), a null-hypothesis states the *absence* of correlation between development process type and outcome relative to the aspect; all hypotheses refer to projects developed 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* (compared to onshore development) for customers in projects using agile methods vs. projects 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.

Similarly, for each potentially problematic aspect G–P of RQ2 (communication difficulties, cultural differences, etc.), a null-hypothesis states the *absence* of correlation between development process type and the emergence of the problematic aspect; again, all hypotheses refer to projects developed in distributed settings.

H_0^G There is no difference in the emergence of *communication difficulties* across geographically distributed units in projects using agile methods vs. projects developed using structured methods.

H_0^H There is no difference in the emergence of *cultural differences* in projects using agile methods vs. projects developed using structured methods.

H_0^I There is no difference in the emergence of *ineffective project management* in projects using agile methods vs. projects developed using structured methods.

H_0^J There is no difference in the emergence of *loss or fluctuation of know-how* in projects using agile methods vs. projects developed using structured methods.

H_0^K There is no difference in the emergence of *shortage of labor skills* in projects using agile methods vs. projects developed using structured methods.

H_0^L There is no difference in the emergence of problems with *ineffective reading of documentation* in projects using agile methods vs. projects developed using structured methods.

H_0^M There is no difference in the emergence of problems with *ineffective writing of documentation* in projects using agile methods vs. projects developed using structured methods.

H_0^N There is no difference in the emergence of *interpersonal conflicts* in projects using agile methods vs. projects developed using structured methods.

H_0^O There is no difference in the emergence of difficulties in *keeping to the project schedule* in projects using agile methods vs. projects developed using structured methods.

H_0^P There is no difference in the emergence of difficulties in *protecting intellectual property* in projects using agile methods vs. projects developed using structured methods.

If the collected data manages to falsify, with a degree of statistical significance, any null-hypothesis, there is evidence supporting the corresponding *alternative* hypothesis: the choice of agile rather than structured development processes has an impact on the outcome of a certain aspect or the emergence of a certain problem in globally distributed projects.

2.2 Discussion of the Hypotheses

Familiarity with software development practices might suggest a priori that not all investigated aspects have the same importance or the same dependence on the development process. For example, among the issues pertaining RQ1, the amount of asynchronous communication may be less relevant than the overall success of a project, which is arguably the most important overall goal. Among the aspects mentioned in RQ2, the difficulty of protecting intellectual property may seem to be largely independent on the choice of development process compared to, say, issues with project management. Nonetheless, we investigate all hypotheses independently, to determine to what extent intuition is supported by hard evidence. This is also helpful to reduce the chance that we miss any significant correlation between process type and other aspects due to incorrect prior expectations.

The following sections describe how data was collected to support or falsify the hypotheses above: Section 3 discusses the data collection process and Section 4 quantitatively analyzes the data. Section 5 complements the quantitative analysis with a qualitative presentation of issues related to RQ1 and RQ2.

3 Research Methodology

The data was collected in two phases.[3] 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 about their distributed development efforts. 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, problematic aspects that emerged, and whether they were organized according to a structured or agile process. As we discuss in the rest of the section, the questionnaires included, whenever possible, descriptive answers with quantitative references, so as to make comparable the answers coming from different participants.

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

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; after all, agile processes emerged in explicit contrast with traditional structured processes (Beck et al. 2001). The complete data set contains information about 66 distributed projects (details about the distribution are in Section 3.1), of which 36 deployed agile methods and 30 structured methods.

3.1 Questionnaires

In the first phase, we contacted through questionnaires companies in various countries and continents worldwide; 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 people with significant experience: 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.

Table 2 Quantitative analysis of hypotheses H_0^A to H_0^F

Aspect	T	#	Median	Min/Max	Mean	Mean rank	Std. dev.	U	p
H_0^A: success	A	29	8	7/10	23.14	8.34	1.078	236	0.571
	S	18	9	5/10	25.39	8.44	1.381		
H_0^B: project importance	A	29	9	4/10	23.95	8.69	1.417	259	0.973
	S	18	9	7/10	24.08	8.83	1.043		
H_0^C: team motivation	A	28	8.5	5/10	22.3	8.61	1.449	218	0.887
	S	16	9.5	4/10	22.84	8.5	1.932		
H_0^D: savings	A	15	5	4/6	18	5	0.655	90	0.247
	S	16	4.5	4/6	14.13	4.69	0.793		
H_0^E: real-time communications	A	29	3	1/7	23.62	4.1	2.273	250	0.805
	S	18	4	1/7	24.61	4.33	2.376		
H_0^E: asynchronous communication	A	29	3	1/5	24.48	3.38	1.293	247	0.75
	S	18	3	2/5	23.22	3.22	0.943		

Column T identifies the process type: agile (A) or structured (S); column # reports the number of projects that provided data about the aspect. Columns U and p report the results of applying the U test. For the reported severity of each aspect, the remaining columns report Median, Minimum and Maximum, Mean Rank, Mean, and Standard deviation. See Table 1 for the ordinal scales of each variable

Table 1 lists the questionnaire questions measuring aspects A–P to assess hypotheses H_0^A–H_0^P. As shown in the rightmost column, the ordinal range of available answers for each question came, whenever possible, with a description that refers to quantitative data (to make comparable answers to the same questions coming from different participants). As it is common practice in empirical research based on questionnaires, we used closed questions (that is, questions with predefined answers), which can be answered in a moderate amount of time. We selected the possible answers and their ranges based on our intuition and previous experience—but not on a pilot study. Participants could choose to not answer individual questions if they did not have access to or were not allowed to disclose the relevant data. Tables 2 and 3 report the number of valid answers received for each question.

3.2 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.

Table 3 Quantitative analysis of hypotheses H_0^G to H_0^P

Aspect	T	#	Median	Min/Max	Mean rank	Mean	Std. dev.	U	p
H_0^G: communication	A	29	2	1/3	23.81	1.97	0.778	255.5	0.898
	S	18	2	1/3	24.31	2.00	0.907		
H_0^H: cultural differences	A	29	2	1/4	26.24	1.83	0.848	196	0.119
	S	18	1	1/4	20.39	1.5	0.857		
H_0^I: project management	A	27	2	1/3	22.24	2.11	0.974	243	0.931
	S	17	2	1/4	23.71	1.88	0.928		
H_0^J: loss of know-how	A	29	2	1/3	23.81	1.97	0.778	222.5	0.86
	S	17	2	1/4	22.91	2.18	1.015		
H_0^K: labor skills	A	28	1	1/3	22.71	1.54	1.015	230	0.578
	S	18	1.5	1/4	24.72	1.67	0.840		
H_0^L: reading doc.	A	28	1	1/3	22.88	1.54	0.693	234	0.926
	S	18	1	1/3	23.21	1.59	0.795		
H_0^M: writing doc.	A	28	2	1/4	24.18	1.68	0.772	205	0.388
	S	17	1	1/3	21.06	1.47	0.624		
H_0^N: personal conflicts	A	29	1	1/2	22.36	1.45	0.506	213.5	0.242
	S	18	2	1/3	24.31	2.00	0.907		
H_0^O: project schedule	A	29	2	1/4	24.69	2.07	0.884	241	0.636
	S	18	2	1/4	22.89	1.94	0.802		
H_0^P: intellectual property	A	26	1	1/3	21.56	1.15	0.464	209.5	0.358
	S	18	1	1/3	23.86	1.28	0.575		

Column T identifies the process type: agile (A) or structured (S); column # reports the number of projects that provided data about the aspect. Columns U and p report the results of applying the U test. For the reported severity of each aspect, the remaining columns report Median, Minimum and Maximum, Mean Rank, Mean, and Standard deviation. All variables are on a 1–4 ordinal scale (see Table 1)

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 5.

4 Quantitative Results

We now present the quantitative data analysis of the hypotheses presented in Section 2 on the data of the questionnaires (see Section 3). Section 4.1 discusses the six hypotheses H_0^A to H_0^F related to RQ1 about project outcome; subsection 4.2 collectively presents the findings for the ten hypotheses H_0^G to H_0^P related to RQ2 about problematic aspects.

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.

The analysis for each of the hypotheses H_0^A to H_0^P aims to determine whether the data shows 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 shown in Table 1, 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. The presence of ordinal data and non-normal distributions suggests to deploy the Mann-Whitney-Wilcoxon U test, a non-parametric statistical hypothesis test (Arcuri and Briand 2011; Sprent and Smeeton 2007).

4.1 Analytical Formulation of Hypotheses H_0^A–H_0^P

Each hypothesis H_0^X, for $X = A, \ldots, P$, refers to a certain quantity (overall success, importance, motivation, etc.) or to the severity of a certain potentially problematic aspect (communication difficulties, cultural differences, etc.) measured by the corresponding random variables X, AG^X, and ST^X—respectively in all projects, in agile projects, and in structured projects. For example, H_0^C tests the *motivation* of teams, hence AG^C models team motivation in agile projects, ST^C models team motivation in structured projects, and C models team motivation across all 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:

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

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.

4.2 Summary of the Quantitative Findings

The overall outcome of our quantitative analyses is that we *never* reject the null-hypothesis; specifically, the p values are normally much greater than any reasonable significance level, and hence no correlation emerges as significant in spite of the fairly large sample size. Therefore, the data provides no evidence to distinguish between using agile vs. structured processes as independent variables.

The following subsections describe the results in more detail. As an afterthought following the quantitative analysis, Section 4.3 shows that the data possesses some statistically significant correlations among certain random variables, such as the overall success and the importance of a project. Such correlations are, however, completely independent of the process type—agile or structured.

4.3 Quantitative Analysis of RQ1: Project Outcome

The distribution of the data for hypotheses H_0^A to H_0^F is shown in Fig. 1. While a cursory visual inspection seems to indicate that some aspects—such as the success, economic savings, and amount of asynchronous communication—are impacted by the choice of process, the quantitative analyses show that the differences are not statistically significant.

Table 2 lists the analysis results for hypotheses H_0^A–H_0^F. For each hypothesis, the table reports standard descriptive statistics (such as median and mean) split according to the process type; and the outcome of U test between data in the two partitions (as described earlier in this section). The descriptive statistics refer to the ordinal scales assigned to answers of each question, shown in Table 1; notice that different questions may have different scales (which is not an issue since each

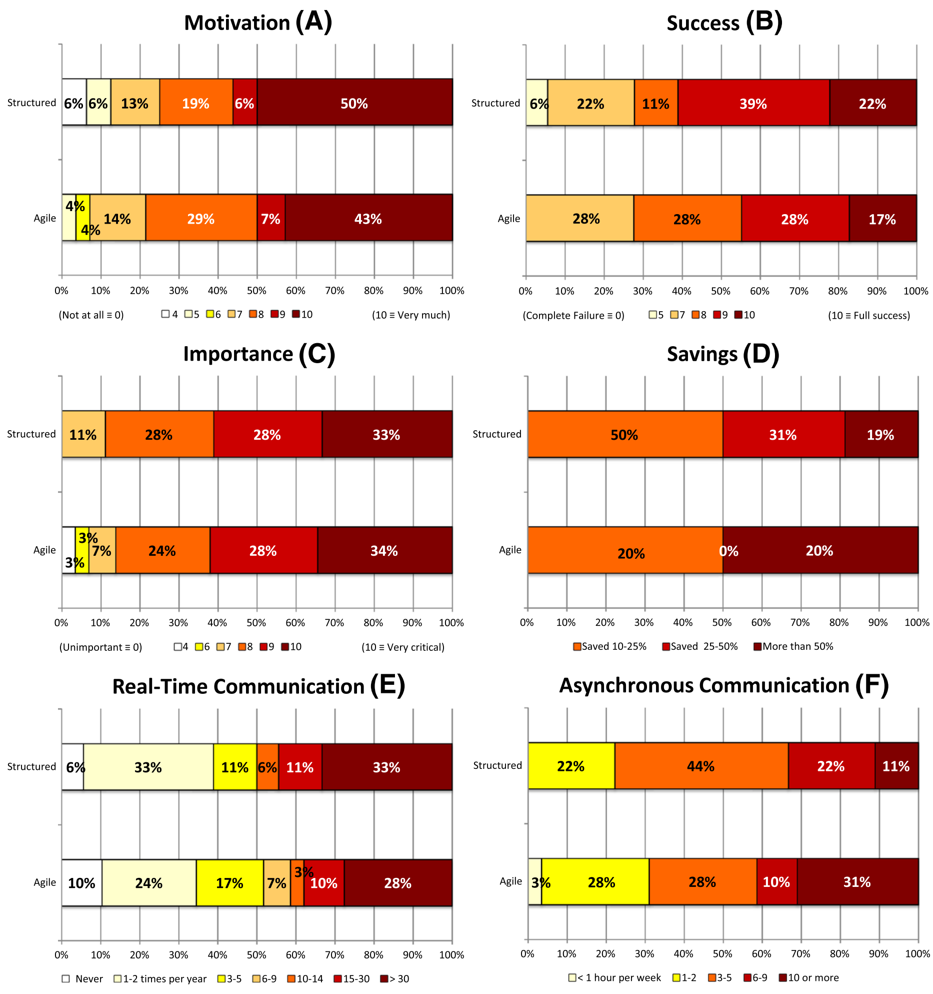

Fig. 1 Bar charts for the data corresponding to hypotheses H_0^A to H_0^F; see Section 4.3 for a detailed analysis

hypothesis refers to a single question). In all, none of the U tests achieves statistical significance; in fact, p is greater than 0.5 in all cases but one (where it still is about 0.25), which means that we are very far from any statistical significance.

4.4 Quantitative Analysis of RQ2: Problematic Aspects

The distribution of the data for hypotheses H_0^G to H_0^P is shown in Fig. 2. Visual inspection may seem to indicate that some problematic aspects emerge more frequently with one process type than with the other type—in particular, communication difficulties are reported as more common with structured processes, and ineffective management and cultural differences tend to be more severe with agile

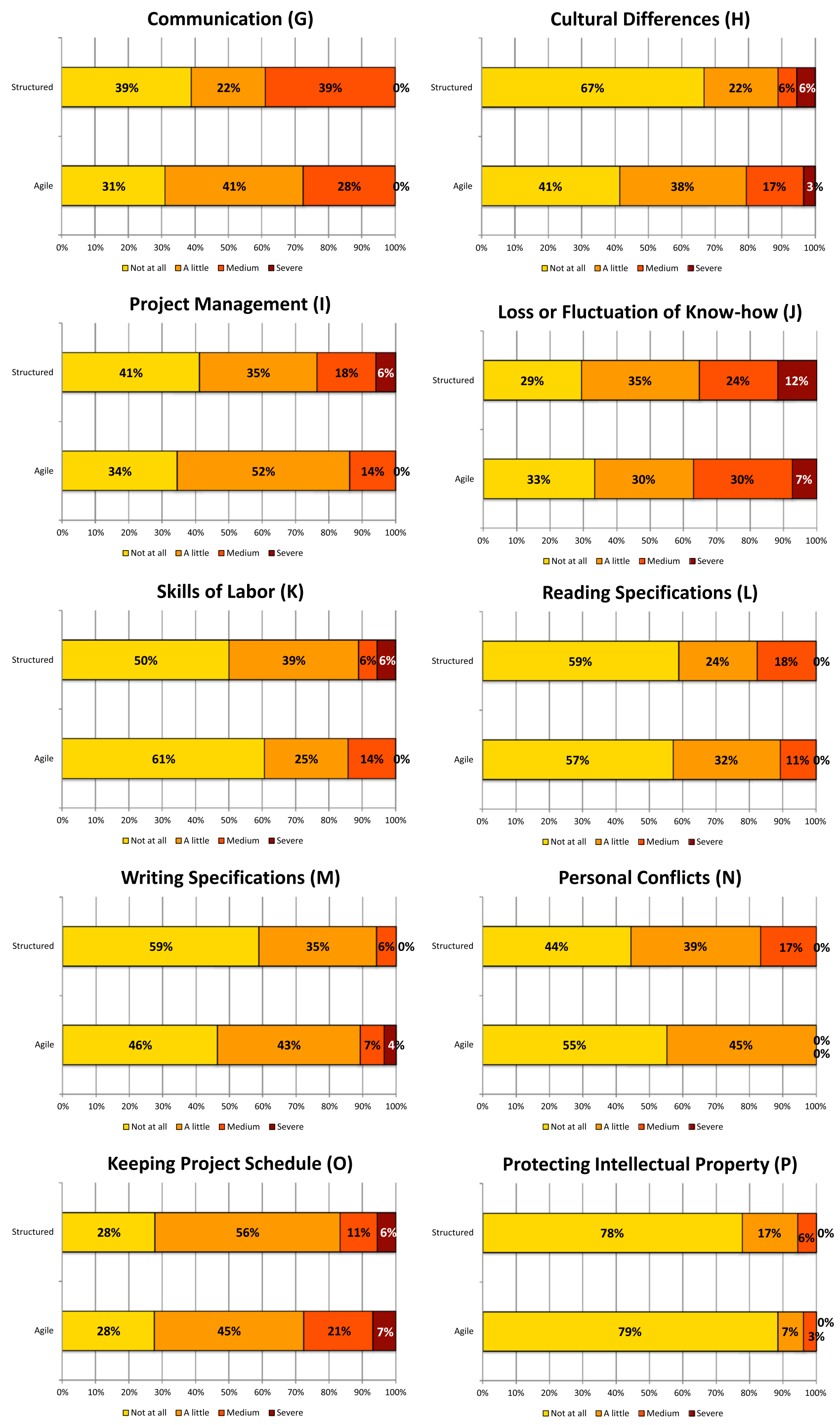

Fig. 2 Bar charts for the data corresponding to hypotheses H_0^G to H_0^P; see Section 4.4 for a detailed analysis

Table 4 Correlation analysis based on Kendall's τ correlation coefficient

Pair of quantities	τ	p (two-sided)
A (overall success) / B (importance)	0.505	< 0.001
L (reading doc.) / K (labor skills)	0.493	< 0.001
L (reading doc.) / M (writing doc.)	0.645	< 0.001
N (personal conflicts) / K (labor skills)	0.407	$= 0.003$

The pairs of random variables are those with a correlation coefficient $\tau \geq 0.4$ or $\tau \leq -0.4$ and with significance $p < 0.01$

processes. The quantitative analyses, however, show that none of the differences is statistically significant.

Table 3 lists the analysis results for hypotheses H_0^G–H_0^P H_0^A. For each hypothesis, the table reports the same statistics as Table 2 but about variables G–P. The descriptive statistics refer to the ordinal scale 1–4 of severity shown in Table 1, which is the same for all variables related to RQ2. In all, none of the U tests achieves statistical significance; in fact, p is greater than 0.24 in all cases but one (where it still greater than 0.11), which means that we clearly have no statistical significance.

4.5 Other Correlations Between Variables

The analysis using the Mann-Whitney-Wilcoxon U test did not provide any evidence of statistically significant correlations between the project type (agile or structured) and the reported measures of different quantities (success, team motivation, communication problems, etc.). Are there statistically significant correlations between other pairs of variables, not involving the project type?

To answer this question, we calculated Kendall's τ rank correlation coefficient for all pairs of random variables A to P measuring the quantities involved in hypotheses H_0^A to H_0^P across *all* 47 projects targeted by the questionnaires. Table 4 lists the cases of significant correlations: the pairs of variables with a correlation coefficient $\tau \geq 0.4$ or $\tau \leq -0.4$ and a significance $p < 0.01$.

These correlations may suggest lines for future studies about significant aspects of globally distributed software development. At the same time, none of them is really surprising in hindsight. If a project is important for the customer, it is reasonable that more effort is put into it so as to achieve overall success; this justifies the correlation between A and B. Properly handling documentation (whether formal documents or use-case scenarios) requires skilled developers (correlation between L and M), who can also abate the likelihood of interpersonal conflicts due to lack of confidence in other team members (correlation between N and K). Finally, poorly written documentation is also hard to read, reflected in the correlation between variables L and M.

5 Qualitative Results

This section discusses various aspects of distributed software development that emerged from all the collected data: 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 issues of personnel fluctuation and intellectual property management, the criteria for choosing development process, and the typical team size.

Unlike the results of Section 4, the data is mostly qualitative and deals with aspects complementary to the choice of development process that also affect the outcome and success of projects. Sections 5.1 through 5.7 report data from the interviews only (for a total of 18 projects, see Section 3), whereas the other Sections 5.8 and 5.9 also incorporates data from the questionnaires (totaling 65 projects).

Unlike the previous Section 4, the results in the current section are mainly *qualitative*. The measures we report in the following subsections should therefore mainly be considered elements to articulate the discussion and corroborate the quantitative picture of the rest of the paper.

5.1 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 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 %;[4] outsourced projects reported very similar savings for an average of 33 % and a standard deviation of 11 %.[5] 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.

5.2 Project Success

Out of the 18 projects surveyed in the interviews: 11 are considered "complete success"; and 5 are "overall success" which, however, suffered non-trivial problems during development. The major sources of problems and difficulties were:

[4]Data about 8 projects.

[5]Data about 10 projects.

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).

The data does not show any visible correlation between the motivation of a development team and the overall success (while Section 4.3 showed correlation between project importance and success). Other factors, however, seem to positively affect the final outcome: companies that are comfortable with frequent changes of customer, working on series of short-term independent projects rather than on the long-term incremental development of a single product, tend to encounter fewer problems. The interviews suggest that companies adapt to frequently changing customers by improving their process and management skills; companies bound to fewer customers, in contrast, accumulate valuable knowledge but tend to have less robust organizational structures. This may suggest that flexibility is a central skill for distributed software development, required to react to and minimize the effects of inevitable problems.

5.3 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 (where problems were 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.

5.4 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.

5.5 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.

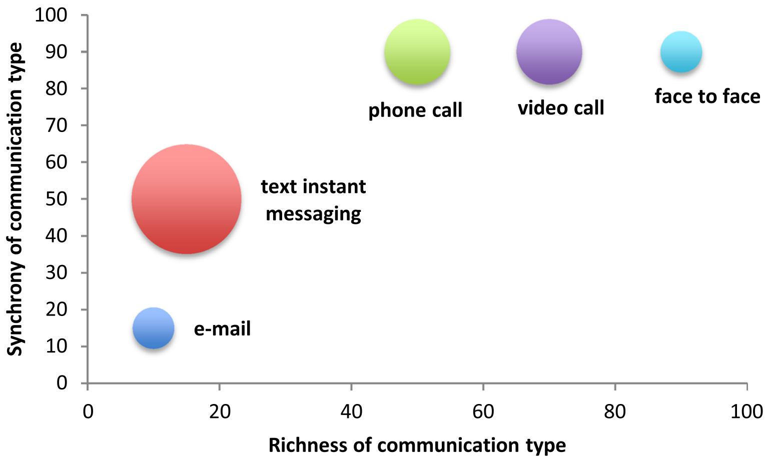

Fig. 3 Communication means categorized by richness and synchrony of different communication types. The diameter of the circles is proportional to the amount of communication (*calculated as means*) used in the distributed projects examined through interviews

5.6 Personnel Fluctuation

Personnel fluctuation refers to the phenomenon of frequently changing teams, whose members are likely to quit on a short notice (or no notice at all) and have to be rapidly replaced. Fluctuation often causes loss of know-how, delays, and cost increases for training and recruitment. While 6 out of 18 projects report fluctuation levels higher than those they experience in projects developed entirely in Switzerland, only one of them found fluctuation to be a critical problem in distributed development.

The interviewees mentioned different mentalities of work and the availability of many job opportunities as the main reasons for why offshore personnel may fluctuate. In some cases, fluctuation is just a consequence of the less stimulating tasks being often outsourced: the offshore personnel would be motivated were it assigned to interesting jobs.

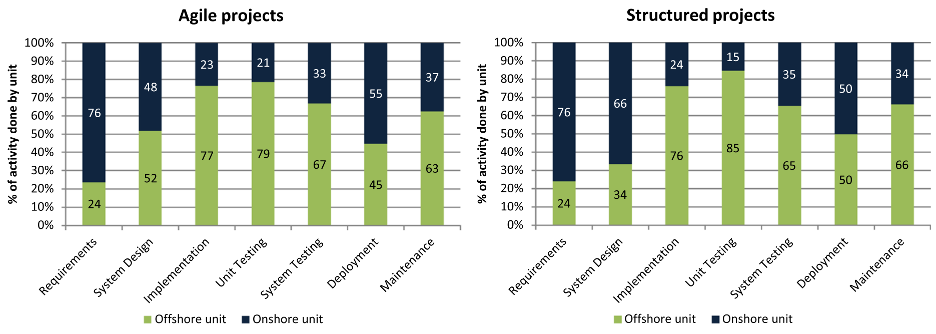

Fig. 4 Allocation (percentage of time assigned to offshore or onshore units) of development phases in agile and structured processes

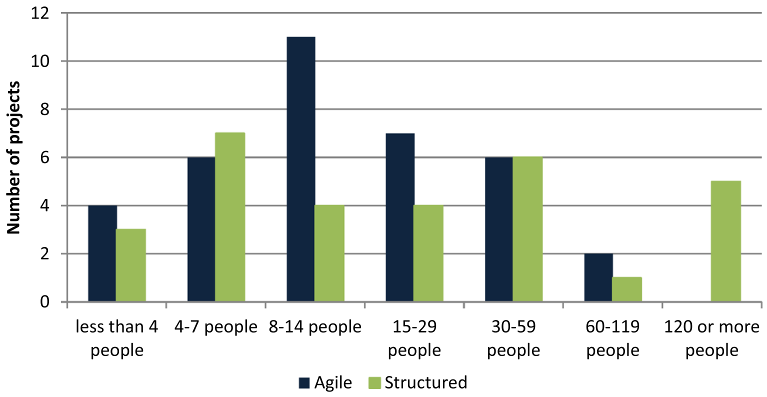

Fig. 5 Team size in agile and structured projects

5.7 Intellectual Property

Companies consider intellectual property a key asset, as it provides fundamental protections against competitors. None of the projects encountered problems regarding intellectual property. This was probably the results of two factors: first, intellectual property is project-specific, and hence it is hardly marketable to others in the case of custom software projects (all the projects targeted by the interviews are custom). Second, the critical components of each project are developed onshore, where the most valuable intellectual property can be adequately protected.

5.8 Onshoring vs. Offshoring in Different Development Phases

For each development phase (requirements, design, implementation, unit and system testing, deployment, and maintenance), we asked which percentage of the time spent on that activity was assigned to the onshore rather than to the offshore units of a distributed project.[6] Figure 4 shows the data about 66 projects, discussed during the interviews and reported in the questionnaires (in addition to the data of the questionnaire data analyzed in Section 4). The graphs only suggest a slight difference in the system design phase—where offshoring is more common with agile processes—and in the unit testing phase—where offshoring is more common with structured processes. The overall qualitative trends are, however, quite similar in the two plots of Fig. 4, consistently with the general observations of the study.

5.9 Team Size

Agile development practices focus on small teams, whereas structured processes often are designed with larger teams in mind; the data we collected about team size

[6]We did not provide a definition of the various activities to interviewees, relying on their conventional understanding of the terms in each context. While the same phase may have fairly different connotations in agile rather than structured processes (e.g. requirements in agile projects might refer to user stories), no interviewee voiced doubts about how to assign activities within their projects or asked for clarifications about this aspect.

confirms these expectations. Figure 5 shows the team size of 66 projects, discussed during the interviews and reported in the questionnaires (in addition to the data of the questionnaire data analyzed in Section 4); agile projects tend to have smaller teams than structured projects: most agile projects deploy teams of 30 people or fewer, whereas structured projects may deploy large teams, up to 120 people or more. It is noteworthy, however, that agile practices have been scaled up to teams of more than 60 developers in a couple of cases; and that structured processes have been followed even with quite small (< 4) teams.

6 Discussion: Practical Implications

The bulk of our study showed no statistically significant correlations between variables measuring outcomes of distributed processes and the type of development project followed—agile or structured. What are the take-home lessons of this overall result for practitioners?

During the informal discussions following the presentation of an initial version of this article (Estler et al. 2012), our results were often perceived as provocative, as they seemed to clash with the direct experience of many experienced practitioners of distributed development. Everyone had their success stories, reporting noticeable improvements as soon as they switched from a process type to another; how can our results so blatantly contradict their experience?

A common thread we noticed in all of such success stories is that they were mostly anecdotal and, even in the few cases where they were based on rigorously collected quantitative data, they invariably involved only a handful of projects (in many cases, only one) managed and performed by the same advocate of the particular choices made. Our study, in contrast, targets a substantial number of different projects, involves quantitative data (even taking into account the limitations of questionnaires discussed in Section 7), and does not target any projects in which we were even remotely involved. The implications of sharpening the experimental conditions are well-known: *if* there are significant correlations, they should be magnified by better designed experiments and larger sample sizes.

In our study, we observed the opposite: the claimed superiority of one or the other development process for distributed development vanished as we looked at correlations of many variables and over a larger number of projects. As discussed in Section 4, we do not get even close to statistical significance: most of our p values are larger than 50 %. Rather than resuscitating the hopes that “one process type is intrinsically better”, the qualitative analysis of Section 5 corroborates the quantitative results whenever it targets similar aspects, since there is no distinctive trend that emerges based on the structured vs. agile dichotomy. While experiments can never establish a negative, the only reasonable conclusion is that there are no significant intrinsic correlations between development process type and the numerous variables we considered to measure project outcome and problematic aspects.

While we maintain that this result is sound, it should not be misinterpreted as to suggest that “processes don’t matter”. The only sensible way to reconcile the individual personal successes of many practitioners with the big picture drawn by our study is concluding that the process is not an independent variable. We could not find any obvious aspect whose outcome is *single handedly* influenced by the choice of process. Therefore, we cannot expect to positively affect the chances of success (or other dimensions) of a project just by choosing to be agile rather than structured. Practitioners should choose the process carefully, based on the project characteristics, organizational structure of the company, and their previous successful or unsuccessful experiences, rather than a priori expecting that a single choice can be a silver bullet. To summarize with an alternative slogan: "processes do matter, but they're not all that matters".

7 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.

7.1 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" (Fowler and Mangione 1989)—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" (or "structured") properly followed the agile (or 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 the latest completed single (agile or structured) globally distributed software development project; hence they had a chance to select one well-rounded example that unambiguously fits the agile or structured paradigm, rather than a hybrid. Furthermore, the differences among agile (or 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 (Beck et al. 2001) against mainstream development practices, hence the "agile vs. structured" classification is reasonably robust. Second, Table 1 shows how we provided descriptions for the ordinal values of answers to questions in the questionnaires; the descriptions typically include quantitative references (e.g., "less than 2 hours per week"). This helps ground the various answers on quantitative data, and reduces the risk of wildly different interpretations by different respondents. Third, we limited the quantitative analysis (Section 4) to the more reliable data coming from the questionnaires, whereas we used the possibly somewhat iffy data coming from the interviews only qualitatively, to corroborate the overall picture drawn by the study (Section 5). Fourth, the wide array of variables analyzed independently (also see the discussion in Section 4.3) increases the chance that, if some significant correlation were present, we would have detected it. 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.

7.2 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.

8 Related Work

This section presents related work in three areas: empirical studies on agile processes in local development settings (Section 8.1), on the issues and challenges raised by distributed development (Section 8.2), and on applying agile processes in distributed settings (Section 8.3). Section 1 listed general references on development processes and their role in the software development life-cycle.

8.1 Agile Processes for Local Development

The effectiveness of agile processes in collocated projects has been widely investigated empirically. Müller and Tichy (2001) 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-h programming project and 3 smaller assignments (totaling about 600 min 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 (2005) 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.

Begel and Nagappan (2008) 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 (2006) 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. (2002) 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.

8.2 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 (1977) 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 (1999) identified *loss of communication* as one of four major risk factors that can lead to the failure of distributed software projects.

Herbsleb and Mockus (2003) 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 (Nordio et al. 2011a), we studied the effect of time zones and locations on communication within distributed teams; we performed the study as part of our DOSE (Nordio et al. 2010, 2011b) 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. (2009) 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 (2006) 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 and Nambiar (2009) 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.

8.3 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. (2006) 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. (2009) 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 and Shrinivasavadhani (2008) 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.

9 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, economic savings of projects, team motivation, and real-time and asynchronous communication, as well as the emergence of problematic aspects such as management difficulties and personal conflicts. The collected data shows that the correlations between process type and the 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 (Estler et al. 2013), which supports software development in the cloud with real-time concurrent editing.

Acknowledgements The authors thank all the participants to the study; the ICGSE 2012 attendees—where a preliminary version of this work was presented—for their lively and interesting comments and remarks about our work; the anonymous reviewers of the conference and journal papers; and the guest editors Erran Carmel, Rini van Solingen, Rafael Prickladnicki, and Filippo Lanubile for organizing and managing this special issue of EMSE. This research has been partially funded by the Gebert-Rüf Stiftung.

References

Allen TJ (1977) Managing the flow of technology. MIT Press, Cambridge

Arcuri A, Briand LC (2011) A practical guide for using statistical tests to assess randomized algorithms in software engineering. In: Proceedings of the 33rd international conference on software engineering, (ICSE). ACM, Waikiki, Honolulu , HI, USA, 21–28 May 2011, pp 1–10

Beck K, Beedle M, van Bennekum A, Cockburn A, Cunningham W, Fowler M, Grenning J, Highsmith J, Hunt A, Jeffries R, Kern J, Marick B, Martin RC, Mellor S, Schwaber K, Sutherland J, Thomas D (2001) Manifesto for agile software development. http://www.agilemanifesto.org

Begel A, Nagappan N (2008) Pair programming: what's in it for me? In: Proceedings of the second international symposium on empirical software engineering and measurement, (ESEM). ACM, Kaiserslautern, Germany, 9–10 October 2008, pp 120–128

Bhat T, Nagappan N (2006) Evaluating the efficacy of test-driven development: industrial case studies. In: 2006 International symposium on empirical software engineering, (ISESE). ACM, Rio de Janeiro, Brazil, 21–22 September 2006, pp 356–363

Bird C, Nagappan N, Devanbu PT, Gall H, Murphy B (2009) Does distributed development affect software quality? an empirical case study of Windows Vista. Commun ACM 52(8):85–93

Boehm B, Turner R (2004) Balancing agility and discipline: a guide for the perplexed. Addison-Wesley, Reading, MA

Carmel E (1999) Global Software teams: collaborating across borders and time zones. Prentice Hall PTR, Englewood Cliffs

Cataldo M, Nambiar S (2009) On the relationship between process maturity and geographic distribution: an empirical analysis of their impact on software quality. In: Proceedings of the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT international symposium on foundations of software engineering, 2009. ACM, Amsterdam, The Netherlands, 24–28 August 2009, pp 101–110

Cohen D, Lindvall M, Costa P (2004) An introduction to agile methods. Adv Comput 62:1–66

Espinosa JA, Nan N, Carmel E (2007) Do gradations of time zone separation make a difference in performance? a first laboratory study. In: 2nd IEEE international conference on global software engineering, (ICGSE). IEEE, Munich, Germany, 27–30 August 2007, pp 12–22

Estler HC, Nordio M, Furia CA, Meyer B, Schneider J (2012) Agile vs. structured distributed software development: A case study. In: Carmel E, van Solingen R (eds) Proceedings of the

7th international conference on global software engineering, (ICGSE). IEEE Computer Society, pp 11–20

Estler HC, Nordio M, Furia CA, Meyer B (2013) Unifying configuration management with merge conflict detection and awareness systems. In: Dietrich J, Noble J (eds) Proceedings of the 22nd Australasian software engineering conference, (ASWEC). IEEE Computer Society, pp 201–210

Fowler FJ, Mangione TW (1989) Standardized survey interviewing: minimizing interviewer-related error. Sage Publications Inc.

Ghezzi C, Jazayeri M, Mandrioli D (2002) Fundamentals of software engineering, 2nd edn. Prentice Hall

Herbsleb JD, Mockus A (2003) An empirical study of speed and communication in globally distributed software development. IEEE Trans Softw Eng 29(6):481–494

Herbsleb JD, Mockus A, Finholt TA, Grinter RE (2000) Distance, dependencies, and delay in a global collaboration. In: Proceedings of the 2000 ACM conference on computer supported cooperative work. ACM, pp 319–328

Hulkko H, Abrahamsson P (2005) A multiple case study on the impact of pair programming on product quality. In: 27th International conference on software engineering, (ICSE). ACM St. Louis, Missouri, USA, 15–21 May 2005, pp 495–504

Layman L, Williams L, Damian D, Bures H (2006) Essential communication practices for extreme programming in a global software development team. Inf Softw Technol 48(9):781–794

Müller MM, Tichy WF (2001) Case study: extreme programming in a university environment. In: Proceedings of the 23rd international conference on software engineering, (ICSE). 12–19 May 2001, Toronto, Ontario, Canada, IEEE, pp 537–544

Nawrocki JR, Walter B, Wojciechowski A (2002) Comparison of CMM Level 2 and eXtreme programming. In: Proceedings of the 7th international conference on software quality, (ECSQ). Springer, pp 288–297

Nordio M, Mitin R, Meyer B, Ghezzi C, Di Nitto E, Tamburrelli G (2009) The role of contracts in distributed development. In: Proceedings of software engineering advances for offshore and outsourced development, (SEAFOOD). Springer, pp 117–129

Nordio M, Mitin R, Meyer B (2010) Advanced hands-on training for distributed and outsourced software engineering. In: Proceedings of the 32nd ACM/IEEE international conference on software engineering, (ICSE), vol 1. ACM, Cape Town, South Africa 1–8 May 2010 pp 555–558

Nordio M, Estler HC, Meyer B, Tschannen J, Ghezzi C, Di Nitto E (2011) How do distribution and time zones affect software development? a case study on communication. In: 6th IEEE international conference on global software engineering, (ICGSE). IEEE, Helsinki, Finland, 15–18 August 2011

Nordio M, Ghezzi C, Meyer B, Di Nitto E, Tamburrelli G, Tschannen J, Aguirre N, Kulkarni V (2011b) Teaching software engineering using globally distributed projects: the DOSE course. In: Collaborative teaching of globally distributed software development – community building workshop, (CTGDSD). ACM, pp 36–40

Paasivaara M, Durasiewicz S, Lassenius C (2009) Using Scrum in distributed agile development: Aa multiple case study. In: 4th IEEE international conference on global software engineering, (ICGSE). IEEE, Limerick, Ireland, 13–16 July 2009, pp 195–204

Paasivaara M, af Ornäs NH, Hynninen P, Lassenius C, Niinimaki T, Piri A (2010) Practical guide to managing distributed software development projects. Tech. rep., Aalto University, School of Science, Dept. of Computer Science and Engineering

Pfleeger SL, Atlee J (2005) Software engineering: theory and practice, 3rd edn. Prentice Hall, Englewood Cliffs

Pressman R (2009) Software engineering: a practitioner's approach, 7th edn. McGraw-Hill, New York

Ramasubbu N, Balan R (2007) Globally distributed software development project performance: an empirical analysis. In: Proceedings of the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT international symposium on foundations of software engineering, 2007. ACM, Dubrovnik, Croatia, 3–7 September 2007, pp 125–134

Ramasubbu N, Cataldo M, Balan RK, Herbsleb JD (2011) Configuring global software teams: a multi-company analysis of project productivity, quality, and profits. In: Proceedings of the 33rd international conference on software engineering, (ICSE). ACM, pp 261–270

Spinellis D (2006) Global software development in the FreeBSD project. In: Proceedings of the 2006 international workshop on global software development for the practitioner, (GSD). ACM, pp 73–79

Sprent P, Smeeton N (2007) Applied nonparametric statistical methods. Texts in statistical science. Chapman & Hall/CRC, London

Sureshchandra K, Shrinivasavadhani JJ (2008) Adopting agile in distributed development. In: 3rd IEEE international conference on global software engineering, (ICGSE). IEEE, Bangalore, India, 17–20 August 2008, pp 217–221

Hans-Christian Estler is a PhD student at the Chair of Software Engineering at ETH Zurich. Before moving to Switzerland in 2010, he completed his Bachelor and Master degrees at the University of Paderborn, Germany. His research interests include collaborative and distributed software development.

Martin Nordio is a lecturer at the Chair of Software Engineering at ETH Zurich. He completed his Dr. Sc. ETH in October 2009 working on proofs and proof transformations for object-oriented programs. He joined ETH in July 2005 after a Bachelor's degree at University of Rio Cuarto, Argentina and a Master's degree at University of Republica, Uruguay. Together with Christian Estler, he is the author of CloudStudio (http://cloudstudio.ethz.ch): a platform for distributed software development.

Carlo A. Furia is a lecturer and researcher at the Chair of Software Engineering of ETH Zurich. His main research interests are in formal methods for software engineering, including formal specification, real-time verification, program proving, and dynamic analysis. He has a PhD in Computer Science from the Politecnico di Milano.

Bertrand Meyer is professor of Software Engineering at ETH Zurich, Head of the Software Engineering Laboratory at ITMO (St. Petersburg, Russia), and Chief Architect at Eiffel Software, California.

Johannes Schneider completed his PhD at ETH Zurich in 2011. Currently, he works as a researcher at IBM's Zurich Research Laboratory in the Information Analytics Group.