Empirical Studies on Distributed Development: Methods and Results

Carol Alexandru

Institut für Informatik
Universität Zürich
alexandru@access.uzh.ch

Abstract. A significant corpus of empirical research has been conducted on distributed development. A number of them are taken a look at in this paper, investigating their methods, issues and results. Then, the factors which influence the communication, the development process, the software quality and other properties of distributed software development are discussed. While the research is very diverse in both method and scale and always has limited applicability, promising trends are visible indicating that distributed development can be just as efficient and effective as collocated development, given that a variety of demanding factors have been catered for during the execution of the project.

Keywords: distributed software development, distributed, collocated, communication, software quality, empirical

1 Introduction

The ever-growing interest in employing distributed software development is rooted in its various upsides, such as better access to well-educated individuals, reduced labour cost [7, 2] and other reasons such as acquisitions or government restrictions [3]. However, distributed software development also poses significant problems in areas such as communication, because the methods of communication are less rich [1, 7, 6, 3], coordination, where breakdowns can occur caused by the lack of communication and differences in how tasks are managed as well as teamwork, where members have a lessened feeling of being part of a team [3, 6]. The purpose of this paper is to give the reader an overview on some of the empirical research that has been conducted in recent years. Instead of summarizing the various papers one after another, we first take a look at the methods of data collection and analysis used by the researchers and the possible threats to their validity, before carrying on to portraying their results - stretching parallels where possible and mentioning how they confirm or contradict each other. It is important to keep in mind how diverse the pool of papers is, ranging from small-scale case studies which involve students as test subjects over studies conducted on a multinational level to large-scale studies comprising several thousands of developers scattered across multiple continents. Data has been collected from test subjects working in software development at multinational companies working in the areas of telecommunication and end-user software, but also from people working on distributed research projects or small-scale projects made for a single customer. As such, the cross-compatibility of results is not always given, and contradictions in the findings of these papers do not come as a surprise. We look at two large-scale studies aimed at understanding how *communication* is affected in a distributed environment and two small-scale case studies on the same topic. We also investigate two studies conducted on the effects of distribution on *software quality* and one study that investigates the usage of *supportive tools* across distributed development sites. It's also necessary to note that details may be omitted in the description of each study and as such, this paper should not only serve as a summary but also as a guide for where to look for further information.

2 Studies and Methods

2.1 Overview

The research conducted can be split in two categories and two types which may both be applied within a study. In one category, the researcher sets out to observe variables in an existing setting, such as a company or an institution. Observations are recorded, employers may be given surveys and data may be collected from existing sources such as version control systems. The collection of data is almost entirely passive [1, 6, 10, 3, 5]. In the second category, case studies may be conducted, where the researcher has a lot of control over how the test subjects will interact, appointing roles to subjects or deciding where and when certain tasks will be performed [7, 4].

The two different types of data collection are qualitative and quantitative. Numbers and values recorded from automated tracking systems are an example of quantitative research, while the compilation of questionnaires with open-ended questions represents qualitative research. In our examples, both categories and both types have been employed.

The following subsections shall describe the goals and methods used in the individual studies. The results will subsequently be brought together.

2.2 Research in Real Work Environments

Distributed vs. Collocated Development Teams: An empirical study has been carried out by Al-Ani et. al. [1] at a Fortune 500 organisation, investigating the similarities and differences between distributed and collocated teams. It has a focus on synchronous formal and informal[1] communication and seeks to assess the effectiveness and efficiency of a variety of communication models. Team members of distributed teams[2] were asked to choose two projects they had

[1] Communication is considered to be formal if a fixed date and length have been set, with invitations sent to all team members while informal communication occurs spontaneously and arbitrarily [1].

[2] Individuals who fulfilled one or more of the following three criteria: A) different time zones, B) different cultures, C) different languages [1].

worked on, one collocated, one distributed, and were interviewed using a mix of open-ended and closed-ended questions. The interviewees were encouraged to comment on their closed-ended choices if they wanted.

This study was conducted with a relatively small sample size at a single company, however the employees had a median of 19 years experience and it was a Fortune 500 company, lending credence to the study. Another possible problem is that it was a largely qualitative study which relies on the recollections of events, which in some cases date back a couple of years.

Speed and Communication: Similarly, Herbsleb et. al. [6] conducted research in two departments of a geographically distributed organisation, investigating the effects of distributed development on communication in general, as well as the communication patterns used. Because this organisation works in telecommunications, where the market is fast-changing, flexible and competitive, coordination and continuous development are very important. One department consisted of about 135 people at sites in the United Kingdom, Germany and India and was involved in real-time processing and call switching. The other department consisted of 90 people in the United States and in Ireland and developed network monitoring, maintenance and administration software. Most of the employees were software engineers, but managers and administrative personnel were also included in the study. Data from the change management system, which was used for version control and change tracking, served as a data source. Herbsleb et. al. did a complete analysis of the data from one department without processing the data from the other department in order to be able to use the second batch of data to test their initial findings for independence. Also, two web-based surveys, each consisting of close-to 70 questions, were conducted in the first department. The respondents answered each question in regard to both local as well as remote co-workers.

The study by Herbsleb et. al. has also been conducted at a single company, however it has a very solid basis when it comes to data, covering both qualitative as well as quantitative figures. Their method of using a second batch of data to verify their own findings inspires confidence in their findings.

Tool Support: Prause et. al. [10] made empirical observations of different tools and their perceived usefulness. They investigated 54 international projects in the proceedings of the sixth framework programme sponsored by the European Union[3]. They contacted the coordinators of the 1201 listed projects, providing a preliminary version of the interview with the question if their project was a software development project and the request for five persons who were at the core of their project. They received 115 replies of which 73 related to a development projects. They then sent out invitations to an online questionnaire to the

[3] This programme has the purpose of strengthening the scientific and technological basis of the industry and encourages international competition and research in support of EU policies [10].

243 developers mentioned by the coordinators. The respondents were enquired on their use of support tools within their group and across the project using questions that were single- or multiple-choice but also allowed for free-form answers if the provided choices didn't suffice. They also had to rate their perceived usefulness on a scale of 1 to 5. They were inquired on tools for project management (e.g. MS Project), requirements assessment (e.g. DOORS), architectural design (e.g. Poseidon, UMLet), documentation (e.g. Wikis, BSCW), communication (e.g. Mailing lists, Skype), software configuration management (e.g. Git, Mercurial), test planning (e.g. Mantis, SiTEMPPO) and automated testing (e.g. JUnit, GoogleTests), integration (e.g. automated nightly build scripts), code reviewing (e.g. Jupiter for Eclipse), bug tracking (e.g. Redmine, eRoom, GForge), static code analysis (mostly built-in to IDEs) and software metrics (e.g. simple SLOC-counting or siMetrics). They also captured the estimated reuse rates of code and the number of contributors. The 106 complete answers they received were brought into canonical form[4] and subsequently analysed.

Prause et. al. suffered from a relatively low return rate of about 7.3% (of development projects) in their survey, which they explain by a possible disinterest in software engineering by the contacted participants or the latter being overwhelmed by the number of support tools offered as choices in their questionnaire, most of which a single respondent may not know. They also experienced inconsistencies in the replies and they did not capture the quality (i.e. extent) of the usage of a tool. Since this study has been conducted in a research context, their findings may not translate to software development at commercial businesses.

Software Quality Investigation on Windows Vista: Another empirical study was conducted by Bird et. al. [3] at Microsoft, investigating the impact of distributed development on product quality. The researchers recorded all commits, the code churn and the complexity data since the release of Windows Server 2003 until the release of Vista. At that point they collected the number of failures reported within six months after the release which they used as a gauge for the quality of the developed product. Research into historical development data of binaries both developed in a collocated as well as a distributed[5] fashion was collected to determine if there is a significant difference. They also investigated, if there was a difference in the nature, size or complexity of the distributed binaries. Research was conducted at multiple levels of separation (building, cafeteria, campus, locality, continent and world). A large corpus of over 4000 binaries and almost 3000 developers scattered across 59 buildings in Asia, Europe and North America were observed in this study. Also, the complexity and maintenance characteristics of both collocated as well as distributed binaries was examined.

[4] This means transforming different words with the same meaning into a single word, for example "svn" and "subversion" both become "subversion", or "1'000" and "a thousand" get saved as "1000" [10].

[5] A binary was considered to be made in a distributed fashion if at least 25% of the code commits came from different sites than the owner of said binary [3].

A very large corpus of data was automatically collected during this study, and the difference in quality between collocated and distributed binaries - respectively their lack thereof - has been shown with great confidence. However, the practices which Bird et. al. found to be profitable to quality in distributed development have not been explained empirically and measuring their extent provides a goal for future research. Like in the preceding studies, only one company has been investigated.

Quality in Global Software Development Projects: Cataldo et. al. [5] looked at the same issue of quality from a different perspective, investigating 189 distributed development projects considering various dimensions, including not only the geographical distribution, but also the temporal distribution and the balance of workload between sites. They formulated a number of hypotheses, speculating that higher levels of distribution, higher levels of imbalance between sites and a higher number of sites will have a negative impact on the quality of the developed product. They went on to test these hypotheses by collecting data from a single development organisation, coming from 189 projects developed at one or more locations across Europe, India and Japan. For their investigation, they randomly picked 50% of the available 562 software components, because all of them would have been too many to process. A project usually involved adapting existing components for new customers according to their requirements, each of them being split in two phases, a *development* and an *integration and testing* phase. This split allowed the researchers to investigate the development process and then measure its outcome during the integration and testing phase. Similarly to how Bird et. al. [3] counted the post-release failures in their study on Vista, Cataldo et. al. [5] counted the defects reported during the integration and testing phase and used different statistical methods outlined in a follow-up section to make use of these numbers when comparing them to the influences of distribution. The geographical distribution was measured by an index involving the following parameters:

- The distance in kilometres
- The number of people in each location
- The total number of people
- The total number of locations

Similarly, the temporal distribution was calculated using the following parameters[6]:

- The number of time zones between locations
- The number of people in both locations

For each project, they also noted the number of sites, the number of regions involved[7] and the number of developers working on it (people-based dispersion),

[6] Both of these index calculation formulas were developed by O'Leary and Cummings in "The Spatial, Temporal and Configurational Characteristics of Geographic Dispersion in Team", MIS Quarterly, 31, 433–452 (2007).

[7] A region was considered an area of similar culture

as well as the distribution of merge requests (MRs) over the locations involved in a project (MR-based dispersion). They also collected every possible number from the version control system, such as the number of lines of code added, deleted and modified for each project and the number of modified components.

From this summary, it is obvious how in-depth the study by Cataldo et. al. [5] is concerning raw data collection. Table 1 is suited for illustrating the collected data.

Table 1. Descriptive statistics of the projects[5]

Variable	**Mean**	**Std. Dev.**	**Min.**	**Max.**	**Skewness**	**Kurtosis**
1. Project is Buggy	0.468	0.501	0	1	0.125	1.015
2. No. of Defects	4.381	16.845	0	188	8.612	90.439
3. Size	17544.3	45262.1	17	508559	8.468	88.931
4. Has Changed Design	0.812	0.391	0	1	-1.601	3.564
5. No. Arch. Comp. Changed	19.7	23.614	1	185	3.126	18.182
6. No. Modification Requests	44.137	119.89	1	1320	8.043	88.296
7. No. New Features	13.437	49.149	0	164	9.924	112.43
8. Process Maturity	3.571	1.073	2	5	-0.312	3.255
9. Spatial Distribution	869.4	1811.7	0	14041	4.865	32.396
10. Temporal Distribution	0.679	1.334	0	10	4.554	28.709
11. No. of Developers	15.187	23.558	1	211	4.136	33.961
12. No. of Locations	2.343	1.202	1	8	1.229	4.515
13. No. of Regional Units	1.693	0.644	1	4	0.666	3.711
14. People Dispersion	0.127	0.152	0	0.578	1079	3.005
15. MR Dispersion	0.163	0.201	0	0.805	1.063	3.127

A for the validity of this study, Cataldo et. al. [5] point out that they didn't have the complete development history of all projects. However they tried to gauge the impact of the missing pieces of information by redoing their analyses on a random sample of 10 projects from their pool, but removing 50% of their information. The results closely matched the initial analysis, showing that even such a large reduction in information does not need to invalidate the results. Only one company was investigated, and Cataldo et. al. mention that they did not measure developer skill, which can have a large impact on quality and productivity. This is a problem not exclusive to this study.

2.3 Case studies

Communication in Distributed Agile Development: Korkala et. al. [7] observed two teams in a controlled environment. One team consisted of one professional developer and three students of higher semester working in a collocated environment with the customer in the same city actively participating in meetings. The other team consisted of two small professional development teams from different cities with the customer spending some time in either city.

In the first team, on-site observations and final interviews were the methods of investigation, in the second team, a personal research diary as well as the email correspondence between the teams and the customer were studied.

Korkala et. al. express that it may be important to not only rely the viewpoints of the developers, but also inquire the other stakeholders, such as the customer.

Text-Based Communication: Calefato et al. [4] also used students in a case study, all attending a course on requirements engineering. They were split up into six groups composed of six to eight randomly selected students. Every student acted as a developer for one project and as a customer for another project. Each group had to develop a requirements specification document for which purpose they had to communicate with their assigned customer group to elicit and negotiate[8] the requirements. Now half of the groups proceeded to work on their tasks face-to-face during the elicitation process and computer mediated during the negotiation process or vice versa. This way, every group had used both techniques and was able to give comparative feedback. During a workshop using the eConference tool, the students were placed in different laboratories to simulate geographical distribution. After each workshop, the students were given questionnaires on two subjects: A) the satisfaction with performance and B) the comfort of the communication mode. The questions were answerable on a scale from 1 (strongly disagree) to 4 (strongly agree).

The fact that this study was conducted in a simulated environment using students may impact its applicability to real work environments. However Calefato et. al. point out that the differences between students and real developers may be smaller than expected. While possibly bored with the task, the students were nevertheless motivated to produce quality deliverables because they were graded on them.

2.4 Techniques of Empirical Data Analysis

When quantitative data collection methods were employed, tests for significance such as the *t-test* [6], the *Kruskall Wallis one-way analysis of variance test* [1] or the *Friedman test* [4] were conducted. For comparison of values, tests like the Mann-Whitney test are used [1, 3]. However, the most in-depth description of and elaborate analysis in the pool of research discussed in this paper was performed by Cataldo et. al. [5] which is why it shall serve as the prime example.

In a previous section, their methods for data acquisition were discussed and we can now describe how they analysed their data: First they perform a pair-wise correlation analysis, trying to identify correlating values. As they found several pairs with high correlation, they became aware of the multi-collinearity among

[8] Eliciting is a rather creative process, trying to approximate what the customer has in mind, proposing details which he may not have anticipated, while negotiating denotes the process of adjusting the resulting requirements into a final state.

their variables. They used a *variance inflation factor* (VIF[9]) to calculate the multicollinearity within a model. Using the VIF analysis as a tool, they went on to use five different statistical computations in sequence. Model I contains all the independent variables and because Cataldo et. al. observed VIF values above 10 for the number of modification requests and the Number of Developers, they removed these variables in model II. In this model, a VIF value higher than 5 was found for the number of regional units so they removed that variable arriving at model III, with all the values not affected by multicollinearity successfully identified. The VIF values can be found in Table 2.

Table 2. Collinearity Diagnostics [5]

Independent Variable	**Model I**	**Model II**	**Model III**
	VIF (Tolerance)	**VIF (Tolerance)**	**VIF (Tolerance)**
Size°	4.73 (0.2115)	4.64 (0.2153)	4.64 (0.2155)
Changed Design	2.99 (0.3340)	2.99 (0.3346)	2.94 (0.3401)
No. Arch. Comp. Change°	6.32 (0.1583)	2.52 (0.3897)	2.37 (0.4228)
No. Modification Req.°	25.77 (0.0388)	–	–
No. New Features°	4.96 (0.2015)	3.61 (0.2771)	3.56 (0.2806)
Process Maturity	1.31 (0.7615)	1.34 (0.7443)	1.20 (0.8333)
Spatial Distribution°	4.58 (0.2185)	4.71 (0.2123)	2.12 (0.4725)
Temporal Distribution°	2.06 (0.4860)	1.97 (0.5086)	1.95 (0.5134)
No. of Developers°	24.43 (0.0409)	–	–
No. of Locations	6.05 (0.1653)	4.38 (0.2282)	2.80 (0.3573)
No. of Regional Units	6.40 (0.1564)	5.49 (0.1823)	–
People Dispersion	1.94 (0.5148)	2.31 (0.5445)	1.84 (0.5445)
MR Dispersion	1.95 (0.5134)	1.34 (0.7443)	1.34 (0.7441)

(° log)

Cataldo et. al. then performed several logistic regression calculations on these models, although only using the control factors for Model I, adding the spatial and temporal distribution measures in Model II and splitting up model III into models IIIa and IIIb, separating the impact by the number of locations versus the number of regions, since these two values are highly correlated. Finally, they included the variables for people dispersion and MR-dispersion in model IV. Table 3 contains the factors by which the distributed setting increases the likelihood of reduced quality - for example, a factor of 2.0 doubles the probability for reported defects. We'll come back and review the interpretation of this data when discussion quality in distributed software engineering in section 4.2. Meanwhile, this example illustrates how a hierarchical data analysis can be

[9] The VIF is the reciprocal of the tolerance, with a tolerance near 1 indicating a low likelihood for multicollinearity, and a tolerance near 0 indicating a high probability for multicollinearity. As such, VIF values above 10 are an indicator for high collinearity, with values above 5 still requiring attention for possible multicollinearity.

performed in an empirical study - trying to separate multicollinearity and using only relevant data depending on the type of hypothesis tested.

Table 3. Odd ratios from logistic regressions [5]

	Model I	**Model II**	**Model IIIa**	**Model IIIb**	**Model IV**
Size$^\circ$	1.0137+	1.0001	1.0001	1.0006	1.0002
Changed Design	2.0484**	2.8328**	3.0051**	3.3252**	2.6872**
No. Arch. Comp. Chan.$^\circ$	3.4225**	3.0181**	2.7465**	2.8063**	2.0891**
No. New Features$^\circ$	2.8237**	2.5859**	2.4723**	2.6443**	2.1834**
Process Maturity	0.9465*	0.8803*	0.8746*	0.8869*	0.8607*
Spatial Distribution$^\circ$		1.0702	1.0041	1.0015	1.0006
Temporal Distribution$^\circ$		1.3404**	1.2557**	1.3184**	1.3639**
No. of Locations			1.3776*		
No. of Regional Units				2.3221**	1.6415**
People Dispersion					1.6723*
MR Dispersion					1.4822*

($^\circ$ log; + $p < 0.10$; * $p < 0.05$; ** $p < 0.01$)

3 Communication

A key point of interest when looking at distributed software engineering in contrast to collocated team work is the way team members need to communicate. Communication is considered a mediating factor which is needed for both coordination as well as control, since software engineering tasks often consist of multiple activities which are performed by various individuals. And even if a person works on a project on its own, reporting to superiors or customers remains a necessity [1].

3.1 Duration, Frequency, Amount and Delay of Communication

Al-Ani et. al. [1] found that distributed team members communicate as often and for the same duration as collocated team members on a formal level, but that informal communication occurred more seldom in distributed teams. Herbsleb et. al. [6] found that informal communication did almost not occur at all. However the duration of informal communication was found to be comparable by Al-Ani et. al. [1] and at the same time it has been found that the formal communication methods used by distributed teams worked just as well as those used in collocated teams. Considering that distributed teams were generally larger, this actually means that they are more effective. Al-Ani et. al. also found that their subjects were under the impression of having more than enough information, which they however did not perceive as a problem.

Yet in the study by Korkala et. al [7], where agile development (which has volatile requirements and changing details) was a focus, a lack of information was indeed found to be the main problem. It resulted in poor coordination and reduced awareness of what the other team members were actually doing.

Respondents of the survey by Herbsleb et. al [6] reported almost exactly the same average number of delays per month for the retrieval of information from the local or remote sites, however the mean duration of the delay was 0.9 days at the local site and 2.4 days for remote sites, which is a significant difference.

3.2 Communication models and patterns

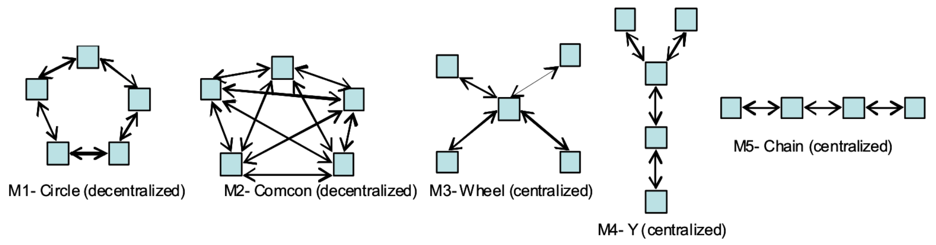

Fig. 1. Initial communication models [1]

It has been found by Al-Ani et. al. [1] that both in collocated as well as distributed teams, the communication patters adapt to the needs of the developers. Yet the distributed teams rated the efficiency and effectiveness of all communication patterns generally lower than collocated teams, which had less to do with the patterns themselves but rather the other typical problems of distributed work, such as cultural differences or different time zones. As part of the closed-ended questions, the interviewees were asked to rate different communication models as shown in Fig. 1.

However, two additional models shown in fig. 2 were derived from the respondents answers, since the provided models did not accurately describe the communication in distributed teams. Over half of the respondents reported to be using model M6, followed by M2 and M3 in approximately equal number. M7 received a single vote, and the other models were apparently not used at all. It was found that models M2 and M3 are appropriate for small teams with five or less members, while model M6 is used for larger teams. Almost all interviewees reported that the required model evolved by itself and was not set to be used at the beginning of a project.

In related work, Prikladnicki et. al. [11] who not only investigated the relationships between distributed development teams but also between the project team, customers and users, devised a more elaborate communication model as shown in fig. 3. They differentiate between inter-group physical distance and intra-group physical distance. For example, if the developers are distributed, that would indicate a high intra-group physical distance. Korkala et. al. [7] picked up

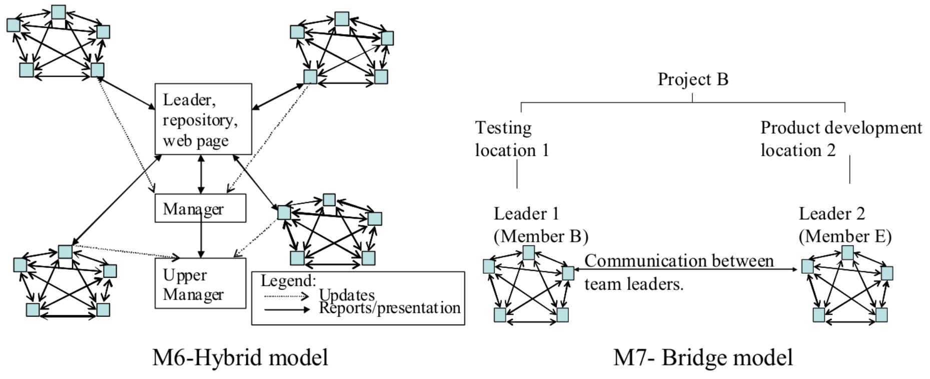

Fig. 2. Derived communication models [1]

this model in their research and used it to empirically test three of the four recommendations postulated by Layman et. al. [8]. They were able to confirm two out of the three recommendations they covered:

- Defining a person who represents the customer, can make decisions upfront, is available and shows real interest in the project.
- Assigning a team member to be working closely with all development teams and the project management on a daily basis

However, their study contradict the findings of Layman regarding the recommendation of using mailing lists for communication if synchronous communication is not possible: Korkala et. al. found a defect rate of 41.8% in their case study, with 62.6% of the defects being caused by inefficient communication: The customer replied only to a third of the questions asked by developers using email.

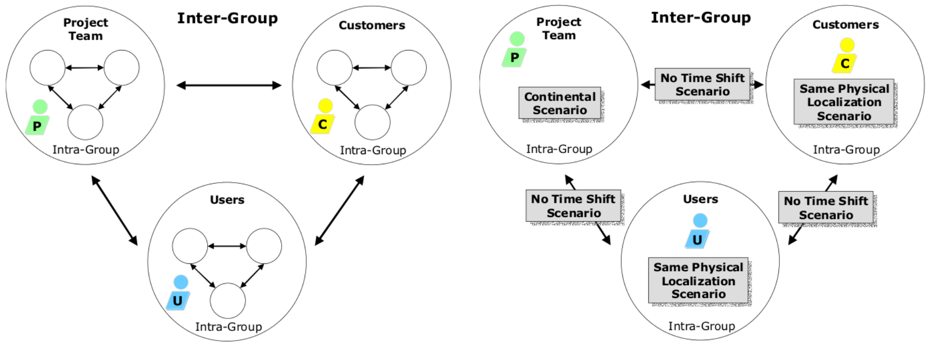

Fig. 3. DSD Distribution Level and Oikodomo Global Technologies as an example [11]

3.3 The value of face-to-face communication

It can be noted, that the complexity of communication is significantly lower in collocated teams, for example because it can be initiated without needing to care for time zones or holidays or as it is easier to resolve conflicts and because emotional cues such as facial expressions, gestures or utterances (such as "mm", "uhu") can aid the communicational process [1, 2]. Students in the controlled environment case study on requirements engineering conducted by Calefato et. al. [4] did not perceive a significant difference in performance and comfort during the elicitation stage, however they did feel that face-to-face communication was better during the negotiation stage. This confirms earlier findings by Murthy et. al [9], who found that distributed groups often outperform collocated groups in tasks revolving about idea-generation (in this case elicitation) while collocated groups outperformed distributed groups in problem-solving tasks (in this case negotiation).

Al-Ani et. al. [1] came to the interesting conclusion that, contrary to what they expected, face-to-face communication seems only to be necessary during initial kick-off meetings and that email or instant messaging was just as common and useful. This suggests that companies should employ more alternatives to face-to-face meetings, even in collocated teams. However, for difficult problems or conflicts, face-to-face meetings are still preferred. Secondly they came to the conclusion that while video conferencing can improve the communication in distributed development, the overhead caused by technical difficulties is unacceptably high. In contrary, Bird et. al. [3] interpreted their quantitative results, coupled with observation of the practices at Microsoft, in such a way that synchronous communication between distributed sites is actually very useful, showing that it simply needs to be employed correctly, which is the case at Microsoft.

4 Impact of Distribution on the Development Process

4.1 Development Speed

In the study conducted by Herbsleb et. al. [6], the data recorded from the change management system yielded a similar result to the findings on the delay in communication described earlier. It showed an average of 5 days for a local modification request to be completed, and over 12 days for a modification request involving more than one site. The data from the other department looks similar, with an average of 7 days for a local, and 18 days for a distributed modification request. Hence, Herbsleb et. al. concluded that a distributed development task takes about 2.5 times as long as a local one. However, they found that the interval between modification requests was actually not significantly different for both scenarios and instead depended largely on the number of people involved in a project, its size and its diffusion.

4.2 Software Quality

Bird et. al. [3] found in their study on Windows Vista, that the increase in the number of failures is about 6% for binaries developed in a distributed fashion when put in perspective to the number of people working on it. They were able to confirm the findings by Herbsleb et. al. [6] in the regard that the increase in complexity and failures stems mostly from an increased size or complexity of the team, and that the distribution only has a rather small effect. However, Bird et. al. concluded that the distributed binaries had virtually the same post-release failure rate as the collocated ones, contradicting the results by Herbsleb et. al. They also investigated the code size and complexity, code churn, test coverage and the dependencies of 40 binaries, half of them being the ones with the highest failure rates, the other half being the ones with the lowest failure rates, in order to reveal possible differences between collocated and distributed binaries. However, no significant trends have been found.

Cataldo et. al. [5] based their study on previous research which stated that the temporal distribution between sites makes concurrent work and synchronous communication more difficult and that the balance of workload and difference in size between sites may induce inequalities that impact the quality in a distributed development process. Using their extensive data collection, they computed several statistical analyses as described previously, as described in section 2.4, accounting for statistical significance and correlation. They were able to confirm past research which indicates that projects which alter the platform design or add new features will probably have more than twice as many defects during the integration phase and that modifications that affect multiple architectural components are increasing the probability of problems in a project by more than a factor of three. They also found that there is no significant evidence that the size of the code base has an impact on the quality. However, when it comes to the data more relevant to the context of this paper - distributed development - they show some interesting results: In contrary to the expectations of Cataldo et. al. there is no support for their hypothesis that *geographical* distribution affects the software quality negatively, thus confirming the finding of Bird et. al. [3] that distributed software development does not need to result in lower quality. Contrasting this result, they were indeed able to support their hypothesis that *temporal* distribution has a degrading effect on quality. Cataldo et. al. explain this results through the nature of the majority of the tasks performed by their subjects which apparently don't profit from spontaneous communication (which it is more scarce in *geographically* distributed development) but would profit from synchronous communication (which is harder to come by in a *temporally* distributed setting). Then again, they found that the number of regions a project is distributed across has a negative effect on quality - an effect twice as strong as just a distribution across several locations within the same region, indicating cultural barriers. Additionally, a significant effect was found rooting from the imbalance in the distribution of the people allocated to a project.

4.3 Allegiance Within and Between Teams

The survey by Herbsleb et. al. [6] revealed that distributed social networks are much smaller and that people perceive contacts at remote sites to be less accessible and harder to identify. There was also an indication that the people felt to be less a part of a team when working with remote co-workers and that remote co-workers were less helpful, however this effect could be lessened by having lead team members join another site for a period of time to strengthen the feeling of common ground.

Bird et. al [3], who observed that distributed development at Microsoft works really well, found that all of the sites in their study have existed for a long time and that they have been working together for many years. Also, the pay scale of all sites are the same, and all of these reasons lead to a general feeling of concordance, with little to no competitive feelings among different sites. The findings of Herbsleb et. al. [6] which recommend having some team members visit other sites are reflected in the initial phase of the development of Vista, where a group of engineers and executives from Redmond visited the Indian site at Hyderabad. All of them were employed at Microsoft for a long time, and most of them were originally from India. This helped bridge many gaps, making people gain trust in working towards a common goal. Practising strong code ownership, having common milestone plans and spanning organisational structures over several sites - for example having project managers be responsible for teams at different sites rather than having a hierarchical structure that reflects the geographical distribution - are other useful tools employed by Microsoft to minimize coordination problems.

4.4 Supportive Tools

The study by Prause et. al. [10], who had investigated 54 distributed development projects in a research context, revealed that tools for project management, architectural design and code management are most commonly used. While support tools for communication were regularly employed, the majority of communication still went over email. Even bits of code, documentation or specification were transferred over email and requirements management tools were rarely used. This caused the needed information to be scattered across the sites. In many projects tools for test planning, automated testing and continuous integration were either missing or not used. While the participants considered code reviewing and issue tracking to be important, it was often not supported by tools. Prause et. al. came to the conclusion that most of the tools were rejected by the participants because they felt them to be heavy-weight and cumbersome. They asserted that maybe there even exists a discipline problem in these projects, because in contrast to production organisations like Microsoft, there is little pressure from above and the use of the support tools is mostly optional. As such, Prause et. al. concluded that there should be an active process in place to motivate participants to use these tools for the better of both the team and the goal. To put this in perspective: Microsoft made daily use of synchronous communication methods and they use the exact same array of development and managing tools on all sites [3].

5 Conclusions

Reflecting on the research reviewed in this paper, distributed software development is a multifaceted topic where findings can rarely be generalised. For example Al-Ani et. al. [1] surprisingly found that a lack of information is not a problem and that the methods of communication in a distributed setting evolve and adapt to the needs of the developer. This result is contradicted by the findings of Herbsleb et. al. [6] who saw a very strong decline in communication and development speed across a distributed system, to a large degree because remote team members had difficulties finding relevant information or contacts. Prause et. al. [10] observed that the way the developers adapted their methods to their needs was actually degrading their performance, because they rejected most of the support tools. Yet another surprising result was revealed by the studies of Bird et. al. [3] who found that it is possible to develop in a distributed fashion within a very large company without any significant impact on software quality and while Cataldo et. al. [5] were able to observe some problems with distributed software development - particularly with temporal distribution - their results regarding geographical distribution turned out to be quite promising.

To be successful in a distributed development project, the management needs to take care of a variety of factors: When it comes to communication, it seems that an over-abundance of information is clearly preferred over filtered, yet possibly lacking information. Judging from the investigated research, it seems easier to skim over or ignore messages of little interest, than it is to search for missing information. And while face-to-face communication is valued for difficult problems or to solve conflicts, textual communication can be just as valuable throughout the development phase if proper facilities are provided. If development is done for a customer, it is important to have a representative available to avoid stalling and to steer the development towards the desired outcome. Another important aspect mentioned in several studies is that it is of crucial importance for the members of a distributed team to acquire a feeling of "togetherness". The more interconnected the teams are, the smaller cultural barriers or feelings of competition become, making it possible to work with far-away team mates towards a common goal. Methods of achieving this include having lead team members and project managers visit the other sites, making sure that the sites have an evenly distributed number of people and workload and that they make good use of well-functioning synchronous communication on a regular basis. Another important factor is making sure that coordination problems are kept to a minimum, because those are much harder to solve in a distributed environment, inducing delays and unhappiness. They can be minimised by avoiding a hierarchical structure that reflects the spatial distribution and instead stretching organisational structures across several sites.

It can be concluded that - given a good set-up of the system - distributed development can be just as efficient and effective as collocated development.

References

1. Al-Ani, B., Edwards, H. K.: A Comparative Empirical Study of Communication in Distributed and Collocated Development Teams. In the proceedings of the 2008 IEEE International Conference on Global Software Engineering (2008)
2. Andres, H. P.: A comparison of face-to-face and virtual software development teams. Team Performance Management, Volume 8, 39–48 (2002)
3. Bird C., Nagappan N., Devanbu P., Gall H., Murphy B.: Does Distributed Development Affect Software Quality? An Empirical Case Study of Windows Vista. In the proceedings of the 2009 IEEE International Conference on Software Engineering (2009)
4. Calefato, F., Damina, D., Lanubile, F.: An Empirical Investigation on Text-Based Communication in Distributed Requirements Workshops. In the proceedings of the 2007 IEEE International Conference on Global Software Engineering (2007)
5. Cataldo, M., Nambiar, S.: Quality in Global Software Development Projects: A Closer Look at the Role of Distribution. In the proceedings of the 2009 IEEE International Conference on Global Software Engineering (2009)
6. Herbsleb, J.D., Mockus, A.: An empirical study of speed and communication in globally distributed software development. IEEE Transactions on Software Engineering (2003)
7. Korkala, M., Abrahamsson P.: Communication in Distributed Agile Development: A Case Study. In the proceedings of the 33rd EUROMICRO Conference on Software Engineering and Advanced Applications (2007)
8. Layman L., Williams, L., Damian, D., Bures, H.: Essential communication practices for Extreme Programming in a global software development team. Information and Software Technology, Volume 48, 781–794 (2006)
9. Murthy, U.S., Kerr, D.S.: Task/Technology Fit and The Effectiveness of Group Support Systems: Evidence in The Context of Tasks Requiring Domain Specific Knowledge. In the Proceedings of the 33rd Hawaii International Conference on System Sciences (2000)
10. Prause, C. R., Reiners, R., Dencheva, S.: Empirical Study of Tool Support in Highly Distributed Research Projects. In the proceedings of the 2010 IEEE International Conference on Global Software Engineering (2010)
11. Prikladnicki, R., Audy J., Evaristo R.: Distributed software development: Toward an understanding of the relationship between project team, users and customers. In the Proceedings of the 2003 International Conference on Enterprise Information Systems (2003)