The Impact of the Structure of Communication Patterns in Global Software Development: An Empirical Analysis of a Project Using Agile Methods

Marcelo Cataldo[1], Kate Ehrlich[2]

May 2011
CMU-ISR-11-103

Institute for Software Research
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3890

[1] Institute for Software Research, School of Computer Science, Carnegie Mellon University
[2] IBM Research, Cambridge, MA

ABSTRACT

Successful software development depends on effective communication within and across teams. Structural properties of communication have long been recognized as important drivers of project performance. However, the empirical results are mixed in terms of which particular communication structures are effective in the context of geographically distributed projects. In this paper, we examined the role of hierarchy and small-world communication structures on iteration performance and quality in a large distributed commercial software project that used agile methods. We measured iteration performance by the number of work items that needed to be rescheduled to a later iteration. There was a strong positive effect for hierarchy but a marginal negative effect for small-worlds. We measured quality by the number of defects that resulted from the work done during an iteration. There was a negative effect for hierarchy but a very strong positive effect for small-worlds. The impact of communication structure remained strong and significant even after taking into consideration the effect of dependencies and other control factors. We discuss the implications of these results for research and practice.

Keywords: Communication structures, development performance, software quality, global software development.

1. INTRODUCTION

Successful software development depends on effective communication within and across teams [9, 20] In distributed project teams, communication helps developers recognize and resolve dependencies (e.g. [26]), locate expertise (e.g. [16]) and build shared knowledge [14]. The importance of informal communication within a team raises the issue of how communication should be structured so that team members have the information they need to get their work done without incurring a heavy overhead. Hierarchical structures can reduce the cost of establishing and maintaining the communication links because they centralize such responsibilities in a few individuals. Several researchers have found support for the benefit of informal hierarchical communication structures among distributed teams [21]. In a different fashion, a tightly knit group who all communicate with each other, sometimes called a small-world, are an important source of shared knowledge [14]. Studies of open source projects found that these small clusters form spontaneously around technical topics (e.g. [3, 28]).

Structural properties of communication have long been recognized as important drivers of project performance and success in product development [1, 5, 18]. While there are a few studies that examine communication structure in software development (e.g. [8, 15]), there is little research, which has been extended to examine the specific relationship of communication's structural properties and software development outcomes such as productivity or quality.

In software projects, performance is a function of quality, time and cost; projects are evaluated by their ability to deliver high quality products on time and within budget. In order to meet their goals, teams have invested in process improvement methods such as CMM (Capability Maturity Model). While these have been shown to lead to improvements in quality [19] the benefits may be limited in distributed teams [11]. Moreover, in an effort to achieve higher quality, projects may end up increasing the length of the project. A key question for researchers and practitioners alike is whether it is necessary to trade quality for performance. Can projects achieve high levels of quality without impacting the cost of the project or the amount of time that is spent? Recent research suggests that reducing defects and rework improves quality [19] thereby arguing that both quality and time can be optimized together. However studies of distributed teams showing that team configurations for quality are contrary to those for productivity suggests that there is a tradeoff between quality and time [11, 25].

This paper examines the effect of communication structure – specifically hierarchy and small world structures – on performance and quality. By considering the effect of communication structure on performance, we seek to contribute to deepen the research into the role of communication especially in distributed software teams. By also considering two important software development outcomes, we also seek to contribute to the debate about quality and productivity by shifting the discussion away from direct tradeoffs to a third factor, communication patterns, which underlies both elements.

Our research setting was a large distributed commercial software project using agile methods. Given the characteristics of our research setting, our outcomes of interest are iteration performance and quality. Managers are increasingly turning to agile development methods to manage the unpredictable and changing nature of software development endeavors. By following disciplined incremental and iterative development practices that emphasize early and frequent testing, projects are able to achieve higher quality without sacrificing productivity [23]. Projects using agile development methods are especially well suited for research on communication because the fluid nature of the self-organizing teams lends itself to placing more emphasis on the structure of informal communication.

The rest of the document is organized as follows. We first discuss the theoretical background leading to our hypotheses. Then, we described the research setting and our empirical design followed by a discussion of the results. We conclude with a discussion of limitations of our study and the implications of our results.

2. COMMUNICATION STRUCTURES

2.1 Hierarchical Structures

Hierarchy considers the extent to which team communication flows outward from one person or a small group of people without flowing back [21]. Recent research with software teams argued that hierarchy can benefit these teams by reducing the overhead of communication. However, this benefit may be limited to distribute teams who have a higher cost of communication than collocated teams [21].

Let us first consider the role of hierarchy for iteration performance. In agile development, iterations are used to manage the volume of work by breaking it down into small units that are achievable by the team within a short time-frame, usually 1-6 weeks. Work that is not completed within an iteration may be rescheduled for a future iteration, canceled or added to a backlog for later review. Work items can be rescheduled to a later iteration for a number of reasons. However, we argue that a key driver of reduction in iteration performance, i.e. increases in the number of rescheduled tasks, is the inability of project members to adequately address their tasks for lack of relevant information and lack of clear communication of the plan and schedule. Hierarchical communication structures enable a few individuals to distribute the relevant information to project members that need it, in a very efficient way, particularly in geographically distributed teams. Thus, we expect that iteration performance, as measured by fewer reschedules, will benefit from a hierarchical communication in distributed teams. More formally, we hypothesize:

> *H1a: Hierarchical communication structure will be positively associated with iteration performance*

Hierarchical structures in the context of geographically distributed projects reduce the costs of establishing and maintaining the communication links because they centralize such responsibilities in a few individuals. As projects grow in size, so does their technical and organizational complexity. As engineers perform their development tasks, critical information and knowledge about the development tasks at hand needs to be exchanged. Despite the potential benefits that lean and efficient ways of conveying information might have on iteration performance, there are important challenges associated with integrating, and leveraging such information in the context of particular development tasks [2]. Project members might struggle to make sense of new knowledge and effectively integrate it into the relevant context because of differences in perspectives, lack of common understandings and lack of familiarity working together [4]. Hierarchical communication structures, by centralizing communication in a few individuals, and reducing the amount of communication amongst people at the same level of the hierarchy, can become ineffective as the scale of the information and communication needs of larger projects increases and the dynamism of those needs intensifies. This argument leads to the following hypothesis:

> *H1b: Hierarchical communication structure will be negatively associated with iteration quality*

2.2 Small Worlds Structures

Small-world networks have the property of having small and highly dense clusters of people connected to a few other people. In the context of communication networks, such patterns translate to structures where a person is surrounded by a dense network of people who all communicate with each other.

Considering the work involved in iteration planning and execution, having communication concentrated in several small, closely knit groups is likely to be detrimental to getting the work communicated, understood by the relevant people and, consequently, completed on time. Firstly, dense communication patterns increase the amount of time spent on interactions, which can impact the time available for the work. Secondly, it moves communication away from project and technical leads who have responsibility for directing the work. Thirdly, it reduces the consistency of communication and messages to the whole team by reducing the role and voice of the project and technical leads., Thus, we hypothesize:

> *H2a: Small world communication structure will be negatively associated with iteration performance*

On the other hand, small world patterns may be especially well suited to development tasks that are challenging and highly interdependent, for instance, the need to coordinate a cross-cutting feature. The high level of shared

communication makes it more likely for those people to share similar attitudes and norms and to trust each other and have strong relationship [6, 14]. These commonalities may help reduce team problems due to miscommunication and conflict especially in distributed teams. They also make members more likely to give each other preferential access to important resources and knowledge, consider each other's information, integrated knowledge, and combine information effectively when making decisions [6, 14]. This argument leads to the following hypothesis:

H2b: Small world communication structure will be positively associated with iteration quality

3. RESEARCH SETTING

We studied development of IBM® Rational® Team Concert™. RTC is a development environment developed by IBM that integrates development, collaboration and project management features[17]. IBM made available the data from the software repositories that the project generated during the development of the first version of the product. Although development activities started as early as May 2006, we focused our analysis on the 13 months prior to the release of the product's first version, which was the period of time between July 2007 and August of 2008. The reason to focus on this particular period of time is two-fold. First, in July of 2007 the project underwent a significant change in process opening up the development process to the Jazz community. Second, prior to July 2007, the amount of development activities was a small fraction (498 out of 12886 work items) of the overall development effort. We focused on development activity corresponding to release 0.6. A total of 161 developers, worked on the project during the period analyzed. There were 26 functional teams distributed in 21 different sites across U.S., Canada and Europe. Project members belonged to one or more of these teams, each team having its own manager.

The project used elements of agile methods, in particular, time-boxing was implemented in the form of iterations. The work of the various teams was planned for each iteration. However, the amount of development work varied across teams and within teams across iterations. Then, our unit of analysis is the pair <iteration, team>. In release 0.6 of the product there were 14 iterations. The nature of the development is such that there are also teams who are part of the development effort for only a short period of time, who are very small or who are called upon to contribute a small amount of code. In order to develop a robust dataset, we only included those teams who had at least 20 work items in an iteration and had work items planned in at least 4 iterations. This resulted in data for 16 out of the 26 teams. The 14 iterations and 16 teams resulted in 197 pairs <iteration, team>. Note that we did not have 224 pairs because each team did not necessarily have work items planned in all of the iterations.

3.1 Description of the Measures

In order to examine our hypotheses, we collected a number of measures from various data sources including the project's software repositories. We also collected information about the project member's location from personnel records. The rest of the section is organized as follows. First, we describe our outcome measures, Iteration Performance and Iteration Quality. Second, we describe how we captured the communication in the computer-mediated medium as well as measures to assess the hierarchical and "small-worldness" of those communication patterns.

3.1.1 Measuring Iteration Delivery Performance

Performance on a single iteration, as is the case of software development iteration performance in general, can be assessed in multiple ways. One important measure of iteration performance is related to the number of tasks planned at the beginning of the iteration that are not completed by the end of the iteration. We refer to this particular measure of iteration performance as delivery iteration performance because it focuses on how well the project members that participated in the iteration were able to execute the planned set of activities as well as capturing aspects of the development process such as adequate planning and adequate handling of unexpected situations. We measured *Iteration Delivery Performance* as the number of tasks that were planned for the iteration but were rescheduled to the following iteration because they were not completed. Although other measures such as effort or completion time have been used to measure performance, rescheduling of tasks is an integral part of agile methods and therefore we think it may be less subject to the variability and reliability typically associated with the time-based measures or effort data.

3.1.2 Measuring Iteration Quality

In order to assess the quality outcomes of an iteration, we examined the defects reported in each iteration. An ideal measure of quality is one that is a function of the defects found in an iteration that can be traced back to changes made in a previous iteration (e.g. through a root-cause analysis). Unfortunately, we were constrained by the *ex post* nature of the study and the lack of access to all the relevant projects members who could assist us in linking all defects to particular changes. Instead, we used the following procedure to identify the relevant defects. For a given iteration i_t, we first selected all the defects reported in the following iteration i_{t+1}. Then, we selected a random sample of 10% of the defects. This step resulted in the selection of 527 defects across all X iterations. We inspected the reports of those defects in order to understand if there were indicators that could assist us in the process of linking a defect to the iteration that introduced it. We identified three groups of defects. A first group of defects consisted of work items that were created and resolved in a matter of hours. The comments in the reports suggested that there were issues found while working on other development tasks and the defects were blocking the other activities so they were resolved quickly but limited information existed to link the defect to prior changes in the code except for the information about what source code files were changed. The second group of defects involved work that took 1 or more days to be resolved and involved changing source code files that were modified in the prior iteration. The third group consisted of defects that were also resolved in a matter of days but involved source code files that were not modified in the previous iteration. Our measure of *Iteration Quality* captured the number of defects in the last two groups. In order to remove the first group, we did not consider the defects that had a resolution time below 8 hours, an adequate break point suggested by the distribution of resolution times across all defects. In the results section, we discussed several robustness analyses we performed in order to assess the adequacy of this iteration quality measure.

3.1.3 Construction of the Communication Networks

The RTC development environment provides a number of features that provides projects members with contextualized interaction for work items. These interactions took the form of comments in a work item report and were stored in the tool's database, allowing us to identify the work items, the individuals that made the comment(s) as well as the content of the comment. Since multiple work items could refer to the same issue and later be resolved as duplicates, we considered a work item and all the duplicates as a single work item. In this way, the comments made in the different work item reports were all combined under a single report. Once all the work items associated with a particular iteration were identified, we constructed iteration-level communication networks using the following procedure. First, we considered only those comments made in the work item reports that were within the time frame of the iteration. Second, the first comment in the work item linked the person that made the comment with the owner of the task at the time the comment as made. If the owner and the commenter were the same person, the link was ignored. Third and for the subsequent comments in the work item report, a link in the communication network between two project members was made if the comments were contiguous in time and they were associated with the task at hand. These comments also resulted in a link between the commenter and the owner of the work item at the time it was made. We identified comments in the work item report that were requesting status rather than providing information or contributing to the resolution of the work item. We removed these comments from our analysis.

Finally, it is important to point out that we considered the directionality of the communication. An examination of a random sample of work items revealed that interaction tended to follow sequences of questions or requests for information followed by answers. Identifying the directionality of the flow of information is important to understand how communication patterns evolved during the development effort. This approach is consistent wth, other methods used to extract communication patterns from software repositories (e.g. [3, 28, 31]). As the earlier description suggests, our approach is conservative in terms of assuming the existence of communication relatively to other approaches because it only establishes a communication link between pairs of individuals that made the communication explicit. Other approaches (e.g. [31]) tend to assume communication among individuals when they are subscribed to the notifications generated by the tools which could result in biased structural patterns of communication.

Figure 1: Example of Interaction in a Work Item Report

Dev B:	We could disable it. But again, I still have a problem tying the enablement of the action to the global setting. What happens if the setting is set but the check-in failed?
Dev B:	Would it make more sense to show a dialog when the preference is on explaining auto check-in and possibly allowing the user to toggle it directly from there? I'm not sure that removing or changing the visibility of an action based on a global preference is prudent.
Dev C:	… would leave the action visible and enabled, and let the user know that there are no changes to check in for the selected resource(s). I think that hiding the action when auto-check-in is on is too much of an implicit mode.
Dev D:	This is only an implicit mode if the user sometimes works with auto-checkin on and sometimes works with it off. In most cases I've seen, users always work one way or the other but rarely switch between the two. This is similar to the "pin editor" action in Eclipse.
Dev A:	Auto check-in can fail (e.g. server behaves crazily for 15 minutes). Then user must rearm it by manually checking-in (e.g. after user reads an email saying server has been patched live). In that scenario it is nice to still have this action.
Dev D:	> Auto check-in can fail (e.g. server behaves crazily > for 15 minutes). Then user must rearm it by >manually checking-in(e.g. after user reads an email > saying server has been patched live). In that >scenario it is nice to still have this action. Good point. However, the action is still available from the Pending Changes view. That should be enough to cover this use-case.

We illustrate how the construction of the communication networks was done using an example from one of the work items. Figure 1 shows the comments associated with work item id 4276, which was about hiding "check-in" actions when auto-save is enabled. For simplicity sake, we consider that all the comments were made during a single iteration. Developer A (Dev A) is the owner of the task. Developer B (Dev B) made the first two comments, which provide suggestions on how to address the issue, generating a link from developer B to developer A in the communication network. Developer C (Dev C) expresses his/her opinion regarding Developer's B suggestion generating a link between developer C and B as well as a link between Developer C and Developer A (the owner of the task). Then, Developer D (Dev D) made a comment related to Developer C's suggestion which generated a link between Developer D and Developer C. Developer A (the owner) then follows up with a response to D's comment so a link from Developer A to Developer D is created in the communication network. The process continues in this manner until all the comments associated with the work items that fall between the start and end time of an iteration are processed.

3.1.4 Measuring Hierarchical Communication Patterns

In this paper, we computed the *Hierarchical Communication Pattern* measure based on the hierarchy metric proposed by Krackhardt [22]. Network hierarchy measures the direction of flow in the network. Krackhardt's approach counts the number of times there is a path from a node in the network to another but not a path back. In order for this measure to be computed, we consider the communication network as a dichotomous directed graph. We used the ORA tool [7] to compute the measure. There are other approaches to measure a central communication structure. A more traditional approach would have been to consider network centralization, which captures

the extent to which one or a few individuals are highly connected to the rest of the individuals in the network. Although network centralization and network hierarchy share some similarities, as discussed by Hinds and McGrath [21], the two measures refer to two distinct constructs. A network in which communication is centralized is not necessarily hierarchical because communication could be reciprocal between the central individuals and those in the periphery. Centralization is often used to measure the power or influence of the central individuals. On the other hand, hierarchical networks might not be centralized since information could be delivered through multiple nodes without reciprocal links. Hierarchy is better suited to measure the flow of communication.

3.1.5 Measuring Small World Communication Patterns

Small world networks are those characterized by short average path lengths among any pair of nodes and high levels of a clustering coefficient [29]. Watts and Strogatz [29] proposed such a definition by comparing a class of networks to Erdos-Renyi random graphs. Then, establishing that a network is a "small world" requires a comparison between the focal network and random graphs. In order to measure the "small worldness" of our communication networks, we build on the approach proposed by Uzzi and Spiro [27] which utilizes two separate metrics: the *Clustering Coefficient ratio* and *Path Length ratio.* Both ratios are determined by comparing the clustering coefficient and the average path length measures of our communication networks against the corresponding measures computed from random graphs with the same number of nodes and density. Uzzi and Spiro carried out this step by computing the random graph using a model proposed by Newman [24]. In this particular step, we diverge from Uzzi and Spiro's approach because we had access to ORA [7] that used a different approach to generate random graphs. We computed 100 random graphs and then computed the average clustering coefficient and average path length measures across the 100 graphs. Those averages were used in the comparison to the communication network metrics. The computation of the network metrics was done using the ORA tool. Once the clustering coefficient and average path length ratios were computed, the *Small World Communication Pattern* variable was calculated as *CC ratio* ÷ A*PL ratio* as proposed in Uzzi and Spiro [27].

3.1.6 Additional Control Factors

Past research work suggests a wide range of factors that impact performance and quality in software development (e.g. [4, 10, 20, 32]). We organized our control variables into five groups:

Controls for team-related factors: We computed several measures that capture attributes of the teams we studied. The *Team Size* counted the number of people that formally belong to the team. Since comments on work items associated with a pair (iteration *i*, team *t*) could also come from people outside the team we computed two additional measures: *Number of Project Members Involved* captures the total of team and non-team members that participated in the work associated with each pair (iteration, team) and *Number of Multi-Team Project Members* captures the subset of those individuals that were part of more than one formal team.

The project involved engineers in 21 different development location, so geographical dispersion can play an important role in terms of communication and, consequently, in terms of delivery iteration performance and software quality. *GSD* is a dichotomous variable where 1 indicates that the project members involved in the work associated with a pair (iteration, team) were geographically distributed; otherwise it was set to 0.

We collected measures of experience based on the approaches used by Boh and colleagues [4], which utilize the data in software repositories as the basis for assessing experience. We measured *Average Level of Technical Experience* as the average number of work items that the team member worked on prior to the focal iteration. *Average Level of Shared-work Experience* measures the average number of times that team members worked together in work items prior to the beginning of the focal iteration. Finally, we measured *Average Level of Team Workload* as the average number of work items owned per team member during the iteration.

Controls for communication-related factors: Since our independent measures focus on patterns of communications, it is important to control for factors related to the propensity of the project member to communicate more or less with other project members or with particular project members. We measured the *Total Number of Comment*s generated across all work items in each <iteration, team> pair and the *Total Comments Size* as the size in bytes of all the comments generated across all work items in each <iteration, team> pair. We also computed two measures that captured the project member's amount of interaction at the dyad level. *Amount of Intra-Team*

Communication captured the number of dyads from the communication networks where both members belonged to the same team. On the other hand, *Amount of Inter-Team Communication* captured the number of dyads where at least one of the persons in the pair also belonged to a different team. Finally, we also measured the *Density* of the communication network associated with each <iteration, team> pair.

Controls for task-related factors: Since iterations vary in the amount of work they generate and in the importance of the work, we measured the *Number of Work Items* that were associated with each (iteration, team) pair as well as the *Ratio of High Priority Work Items* that each team faced in each iteration. We also computed the *Total Number of Commits* associated with the work items in each (iteration, team) pair as well as the *Average Size of Work Items* as the average number of files modified in the commits associated with the work items.

Controls for technical-related factors: The work items associated with an iteration (including all teams) were associated with changes in the version control system. Building on past work by Cataldo and Nambiar [12] on the effect of technical dependencies on quality, we constructed an iteration-wide matrix containing the logical dependencies among system's components based on the commit information associated with the work items. In this case, the resulting matrix is symmetric against the diagonal. Since the ownership of each architectural component was allocated to a single team, teams define a relevant technical boundary. We combined such information with the dependency matrices to construct control measures that capture the nature of the technical dependencies in the system. The *Internal Logical Coupling* measure refers to the density of the sub-matrix based on the set of relationships defined in the logical dependencies matrix that are contained within a team's area of responsibility. External Logical Coupling was operationalized as the number of components not under the responsibility of the focal team that had technical dependencies with those components changed by the focal team.

Controls for unobserved effects: Finally, we included a few additional controls for unobserved, team- and iteration-specific factors. These dummy variables effectively control for any factors, which we could not explicitly measure such as the cultural or organization aspects of the teams as well as temporal effects of the iterations. These were recorded as Dummy A through D for the teams and Dummy Iteration A through E for the iterations.

3.2 Description of the Model

Our two dependent variables, *Iteration Delivery Performance* and *Iteration Quality*, are count variables. Traditional linear regression models are not adequate in this context. Count variables could be treated as continuous variables and, then, used in linear regression models. However, the obtained estimates could be inefficient, inconsistent and biased. We used a Negative Binomial regression model instead of a Poisson model because our data did not satisfied the assumption of the Poisson models that the conditional mean equals the conditional variance. We examined our hypotheses by creating several negative binomial models, which included the different control variables as well as our independent measures. In order to assess the fit of each model, we report the log-likelihood of each model as well as the percentage of deviance explained by the model. The deviance of a model is defined as "-2 * log-likelihood of the model"; lower values are associated with a better fit of the model to the data. The percentage of the deviance explained is a ratio of the deviance of the null model (contains only the intercept) and the deviance of the final model. In order to simplify the interpretation of the results, we report the odds ratios associated with each measure instead of reporting the regression coefficients.

4. RESULTS

4.1 Preliminary Analyses

We first computed descriptive statistics of the measures described in the previous section. All variables with the exception of our two independent measures and the GSD measure had skewed distributions so they were log-transformed. We also performed various collinearity diagnostics. We first ran a variance inflation factors (VIF) analysis, which revealed that several of the measures were highly collinear. In accordance with well-established recommendations, we removed from our analyses all the variables with VIF values above 5 [23]. Table 1 shows the VIF and the tolerances of the final set of variables included in the analyses. A pair-wise correlation analysis among those remaining measures showed levels of correlation that were not problematic.

Table 1: Results of the Collinearity Diagnostics

	VIF (Tolerance)
Controls for unobserved effects	
Team Dummy A	1.49 (0.6762)
Team Dummy B	1.36 (0.7355)
Team Dummy C	1.42 (0.7050)
Team Dummy D	1.65 (0.6047)
Iteration Dummy A	1.17 (0.8568)
Iteration Dummy B	1.12 (0.8893)
Iteration Dummy C	1.40 (0.7146)
Iteration Dummy D	2.22 (0.4497)
Iteration Dummy E	1.96 (0.5095)
Controls for Team-related factors	
Number of Project Members Involved	4.26 (0.2309)
Number of Multi-team Project Members	4.07 (0.2456)
GSD	2.05 (0.4883)
Average Level of Technical Experience	1.97 (0.5076)
Average Level of Shared-work Experience	4.27 (0.2324)
Average Level of Team Workload	1.20 (0.8329)
Controls for Communication-related factors	
Amount of Inter-Team Communication	2.07 (0.4838)
Controls for Task-related factors	
Ratio of High Priority Work Items	1.20 (0.8329)
Average Size of Work Items	1.28 (0.7822)
Controls for Technical-related factors	
Internal Logical Coupling	2.09 (0.4784)
External Logical Coupling	4.79 (0.2088)
Independent Variables	
Hierarchical Communication Pattern	2.04 (0.4908)
Small World Communication Pattern	1.07 (0.9348)

Although, regression coefficients are typically reported as part of the results, we opted for reporting the incidence rate ratios (IRRs) associated with the negative binomial regressions because they simplify the interpretation of the results. IRRs indicate the rate at which the dependent variable would change for one unit change in one independent variable keeping the rest of the measures constant. For example in Table 2, the IRR associated with variable GSD is 2.631 suggesting that teams that are geographically distributed (GSD=1) tend to have 2.6 times higher levels of rescheduled work items than those teams that are collocated (GSD=0).

4.2 The Impact on Iteration Delivery Performance

In this section we discuss the results pertaining to hypotheses 1a and 2a, which focused on the relationship between the hierarchical and "small-worldness" nature of communication patterns and the *Iteration Delivery Performance*. Table 2 reports the results of the regression analyses. Models I and II show the incident rate ratios (IRRs) associated with the control factors. We separated the control factors in two different models because we wanted to assess the relative impact of the logical technical dependencies, a factor that has been shown to have major impact on coordination needs [9] and therefore, communication as well as a major impact on software failures [10]. The results reported by models I and II are consistent with past research. We also observe that several of the dummy variables for the teams and the iterations are statistically significant, suggesting there are idiosyn-

cratic attributes of the teams as well as of the iterations that impact the Delivery Iteration Performance of an iteration. Models III and IV test hypothesis 1a by introducing the *Hierarchical Communication Pattern* variable into the analysis. The IRR associated with our independent variable is statistically significant and between 0 and 1 suggesting that higher levels of hierarchy reduce the amount of work items that are rescheduled in an iteration. In other words, increased hierarchical is associated with improved delivery iteration performance providing support for H1a. Model IV shows that the effect of hierarchy remains relevant even when considering the role of the technical dependencies. It is also worth pointing out that the effect size of hierarchy (2.7% of deviance explained) is similar to the effect size of technical dependencies (3.1% of deviance explained).

Finally, models V and VI in Table 2 assessed hypothesis H2a that posited that small-world communication patterns would be associated with lower iteration performance or in our specific case, associated with higher number of rescheduled work items. Our results show partial support for the hypothesis. The IRR associated with the variable *Small-World Communication Patterns* are above 1 but they are only marginally significant.

4.3 The Impact on Iteration Quality

We now turn our attention to hypotheses H1b and H2b, which posited that hierarchical communication patterns would be detrimental to iteration quality while small-world communication patterns would be beneficial. Table 3 reports the results of the regression analyses. As before, Models I and II show the incident rate ratios (IRRs) associated with the control factors and, again, we separate the control factors in the two models to assess the specific impact of logical technical dependencies. The results are consistent with past research. Models III and IV examined hypothesis 1b by introducing into the analysis our independent variable *Hierarchical Communication Patterns*. Our results provide strong support for the H1b suggesting that hierarchical communication patterns in computer-mediated communication are detrimental to the quality outcomes of iterations. In fact, the negative effect of hierarchy is significantly larger than technical dependencies, which prior research suggested as a primary driver for software failures (e.g. [10, 12]). Finally, models V and VI introduced the Small-World Communication Pattern variable into the analysis, which found support for H2b. Higher levels of "small-worldness" in the communication patterns reduced the estimated number of defects associated with the iteration. However, it is worth pointing out that the effect size of the variable is not as high as our other independent variable (hierarchy) since it only explained 4.2% of the deviance.

4.4 The Trade-Off in Structure of the Communication Patterns

Our hypotheses suggest that there is a trade-off between the structural properties of the communication patterns that emerge during an iteration and how those patterns impact two outcomes of interest: delivery iteration performance and quality. The next step in our investigation explored in more detail whether the two different patterns of communication can co-exist or are mutually exclusive. In other words, we would like to understand if teams get "locked into" a particular communication patterns or whether they are able to evolve and adapt their communication patterns. For example, if a team gets locked into a hierarchical structure, we could argue based on our results that the team would trade off better outcomes on one dimension, delivery iteration performance, at the expense of poorer results on the other dimension, iteration quality.

In order to gain some insights into those issues, we examined the evolution of the correlations between the *Hierarchical Communication Patterns* and *Small-World Communication Patterns* measures. Given that our independent measures have opposite effects on our outcome measures, we expected to see a negative correlation. In fact, the variables had a -0.1857 correlation when considering all the (iteration, team) pairs used in the regression analyses. However, the level of correlation was not sufficiently high to suggest that the teams developed one pattern over the other across all iterations. In fact, the relatively low correlation suggests the two types of communications structure could co-exist in a team.

A second related question is whether teams can avoid a trade-off between quality and iteration performance. In Table 4, we coded our independent variables as HIGH if the value was in the top 33 percentile and LOW if it was in the bottom 33 percentile. In each of the 4 cells, we determine whether iteration performance and quality were in the top or bottom 33 percentile. The results suggest that certain teams were able to overcome the trade-off and achieve high performance and quality (top-left cell) by combining hierarchical and small-world communication

structures. On the other hand, the teams that focused solely on one communication pattern faced a trade-off between performance and quality or both outcomes were not particularly high.

Table 2: The Impact of Communication Patterns on Iteration's Delivery Iteration performance

	Model I	Model II	Model III	Model IV	Model V	Model VI
Controls for unobserved effects						
Team Dummy A	1.291**	1.357**	1.085	1.364**	1.413**	1.327**
Team Dummy B	1.069	1.108	1.039	1.119	1.128	1.168
Team Dummy C	0.841+	0.921**	0.707**	0.920**	0.840*	0.928**
Team Dummy D	0.624**	0.934*	0.823*	0.911*	0.661**	0.752*
Iteration Dummy A	0.966	0.862	1.133	0.887	1.137	1.052
Iteration Dummy B	1.046	0.592	0.995	0.997	0.954	0.817
Iteration Dummy C	0.591**	0.514**	0.575**	0.515**	0.517**	0.462**
Iteration Dummy D	0.717+	0.388	0.872	0.400	0.702	0.465
Iteration Dummy E	0.139**	0.097**	0.101**	0.062**	0.116**	0.085**
Controls for Team-related factors						
Number of Project Members Involved	2.823**	5.801**	2.398**	5.710**	3.249**	4.263**
Number of Multi-team Project Members	0.568**	0.816*	0.740*	0.809*	0.590**	0.726*
GSD	2.631**	2.486**	2.238**	2.509**	2.280**	2.061**
Average Level of Technical Experience	0.937	1.013	0.992	1.041	0.983	1.011
Average Level of Shared-work Experience	0.857	0.675*	0.879*	0.601*	0.603*	0.618*
Average Level of Team Workload	1.643**	1.754**	1.792**	1.734**	1.382**	1.694**
Controls for Communication-related factors						
Amount of Inter-Team Communication	1.004**	1.022*	1.003**	1.012*	1.002**	1.003*
Controls for Task-related factors						
Ratio of High Priority Work Items	0.226**	0.701**	0.362**	0.639**	0.363**	0.331**
Average Size of Work Items	1.152	1.066	1.101	1.049	1.203	1.262
Controls for Technical-related factors						
Internal Logical Coupling		1.302		1.094		1.036
External Logical Coupling		2.331**		2.277**		1.869**
Independent Variables						
Hierarchical Communication Pattern			0.669*	0.843*		
Small World Communication Pattern					1.011+	1.016+
Log-Likelihood	-523.41	-502.08	-498.41	-480.71	-518.59	-500.63
Deviance Explained	24.8%	27.9%	28.4%	31.1%	25.5%	28.1%
(+ p < 0.10, * p < 0.05, ** p < 0.01)						

Although Table 4 suggest that teams might be able to avoid a trade-off between quality and iteration performance, there might be factors which lead a team to commit to a particular communication pattern which locks them into certain consequences. Figure 2 shows the evolution of the correlation between the *Hierarchical Communication Patterns* and *Small-World Communication Patterns* measures across the various iterations the project followed while working on release 0.6. We observed some degree of volatility in the correlation levels. A high level of negative correlation (e.g. iteration 06_M1 or 06_M3) would suggest that teams were selecting one particular communication structure over the other, while low levels of correlation would suggest that there was no particular preference made by the teams (e.g. iterations 06_RC3 and 06_RC4). It is worth pointing out that we also have two data points where the correlations are positive (06_M2 and 06_M4D1), however, the value of the correlation is

small. We could argue that these data points are still examples of cases where the teams did not commit to a particular communication pattern.

Table 3: The Impact of Communication Patterns on Iteration Quality

	Model I	Model II	Model III	Model IV	Model V	Model VI
Controls for unobserved effects						
Team Dummy A	1.049	1.035	1.051	1.106	1.049	1.005
Team Dummy B	0.751**	0.802**	0.874**	0.923**	0.773**	0.831**
Team Dummy C	0.846**	0.739**	0.900*	0.875**	0.846	0.870**
Team Dummy D	0.983	0.798	0.975	0.828	0.936	0.835
Iteration Dummy A	0.423**	0.367**	0.417**	0.534**	0.431**	0.449**
Iteration Dummy B	0.921*	0.792*	0.972	0.895*	0.977	0.807*
Iteration Dummy C	1.239**	1.171**	1.401**	1.293**	1.281**	1.265**
Iteration Dummy D	0.445**	0.379**	0.364**	0.609**	0.487**	0.512**
Iteration Dummy E	0.136**	0.092**	0.067**	0.147**	0.102**	0.120**
Controls for Team-related factors						
Number of Project Members	1.374**	1.529**	2.701**	1.995**	1.351**	1.929**
Number of Multi-team Project Members	1.403**	1.091**	1.140	1.254**	1.463**	1.018*
GSD	1.336**	1.351**	1.039*	1.539**	1.282**	1.530**
Average Level of Technical Experience	0.998	0.868**	0.871**	0.881**	0.983+	0.942*
Average Level of Shared-work Experience	0.959	0.801**	0.799*	0.728**	0.894*	0.908*
Average Level of Team Workload	2.361**	1.958**	1.771**	1.554**	2.231**	1.901**
Controls for Communication-related factors						
Amount of Inter-Team Communication	1.001*	1.002	1.236*	1.001	1.001+	1.005
Controls for Task-related factors						
Ratio of High Priority Work Items	0.458**	0.377**	0.904+	0.243**	0.487**	0.230**
Average Size of Work Items	1.565**	1.953**	2.419**	1.983**	1.326**	2.248**
Controls for Technical-related factors						
Internal Logical Coupling		3.971**		2.441**		1.602**
External Logical Coupling		1.518**		1.502**		1.367**
Independent Variables						
Hierarchical Communication Pattern			2.230**	1.271**		
Small World Communication Pattern					0.884*	0.835*
Log-Likelihood	-735.95	-649.95	-521.13	-481.57	-691.09	-607.22
Deviance Explained	27.7%	36.1%	48.9%	52.7%	32.1%	40.3%

(+ $p < 0.10$, * $p < 0.05$, ** $p < 0.01$)

In Figure 3 we depict the correlation between the *Hierarchical Communication Patterns* and *Small-World Communication Patterns* measures grouped by team. In this case, we do observe very distinct patterns across the teams. The correlations suggest that some of the teams (e.g. Team 7 and Team 16) locked into a particular pattern of communication given the relatively high correlation values. On the other hand, other teams such as Team 1 and Team 4 did not exhibit those patterns. These results suggest that teams differ in their ability to alter their communication patterns although further work is needed to examine the extent to which teams evolve their communication in response to task demands or differences in team composition.

Table 4: The Trade-Off between Performance and Quality

	Hierarchy HIGH	Hierarchy LOW
Small-World HIGH	Perf. = HIGH Quality = HIGH	Perf. = LOW Quality = HIGH
Small-World LOW	Perf. = HIGH Quality = LOW	Perf. = LOW Quality = LOW

Figure 2: Correlation between Hierarchical and Small-World Communication Patterns Across Iterations

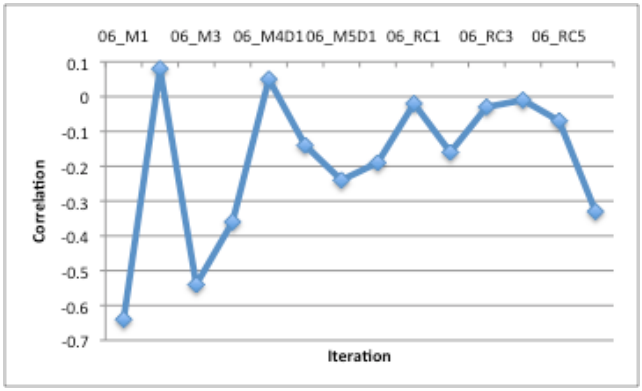
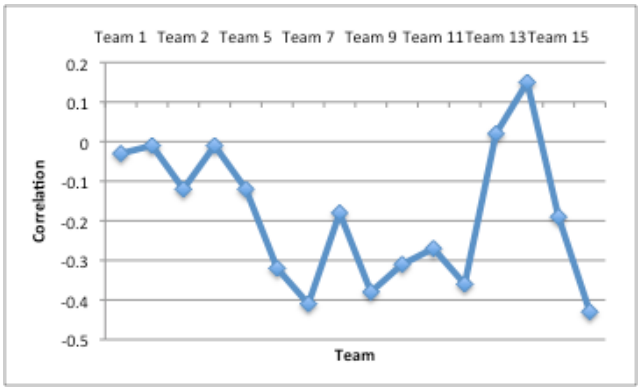


Figure 3: Correlation between Hierarchical and Small-World Communication Patterns Across Teams



5. DISCUSSION

This paper examined the effect of communication structure, specifically hierarchy and small-world structures on iteration performance and quality. We measured iteration performance by the number of work items that needed to be rescheduled to a later iteration. There was a strong positive effect on performance for hierarchy but a marginal negative effect for small-worlds. These results confirm H1a and provide marginal support for H2a. We measured quality by the number of defects that resulted from the work done during an iteration. There was a negative effect on quality for hierarchy but a very strong positive effect for small-worlds confirming H1b and H2b. We especially draw attention to the significance of the findings after taking technical dependency structures

into account. A number of research studies have established that dependencies can drive communication structure [13, 28]. The current study confirmed a positive effect of both intra- and inter-team dependencies on iteration performance and quality. More importantly, the impact of the structure of the communication patterns remained strong, even after taking into consideration the effect of dependencies and a number of other control factors.

First and foremost, these results confirm the importance of informal communication for software teams. They further promote the small but growing research on communication structures [21] by providing evidence of the role for hierarchy and small-world structures in the context of geographically dispersed teams interacting through computer-mediated means. The study also begins to address issues related to how teams can manage their communication by having enough communication without getting overloaded. As past research has noted, it is not the amount of communication that matters but how it relates to the task at hand [28]. In a hierarchical structure, there is less overall communication because the communication flows in a particular direction and there is not a lot of communication between people in the same level of the hierarchy. This structure is suited to tasks associated with planning and executing against a plan. In contrast, there is a lot of localized communication within each small-world cluster although not necessarily much communication between clusters. This structure is better suited to tasks associated with individual development tasks that can benefit from the contextualization and trust that comes from high communication with a small group of people.

Finally, our study extends existing research on communication structure to demonstrate significant associations between communication structure and important software development outcomes. While there is a rich line of work linking communication to general measures of team performance, there is less research examining the relationship of communication structures and outcome variables such as iteration performance and quality. In particular, the positive effect of hierarchy confirms previous research (Hinds and McGrath) and extends their findings to demonstrate measurable benefits on iteration performance. The earlier research was based on correlational rather than log data, which limited the ability to measure performance effects directly. We were also able to verify their surprising observation of negative effects of high density, which in our study occurs in the small-worlds structure, and show that negative effects may be associated with planning rather than development tasks, The results for the small-worlds structure is consistent with research in management science and demonstrates that small-worlds also have important consequences for software engineering.

The results of this study are especially relevant to distributed software development where there can be a high cost associated with communication as well as a high penalty if critical information is missing or misinterpreted [20]. The results suggest that distributed development can reap benefits in both performance and quality from teams that are able to modulate their communication in accordance with the type of work being done, as well as in alignment with the requirements imposed by technical dependencies [9]. Our results further suggest that teams, as well as successive iterations, do vary in their ability to change their communication structure, which bears further investigation.

5.1 Limitations

Our study has several limitations worth highlighting. First, we studied a single project, which might raise external validity concerns. Although we recognize such a risk, we think our research setting represents an unique opportunity to study a geographically distributed project that embraced agile methods through their practices and through the tool the project built.

A second limitation of the study is the focus on the first release of the product. The project has continued to work together and produce other releases of the product. Then, we could expect maturation effects that might impact the patterns of communication and their evolution. Unfortunately, the data associated with subsequent releases are not currently available. Finally, several teams had collocated members and in our study we were not able to collect data to control for face-to-face interactions that might impact how the communication patterns in the work items reports evolve.

5.2 Implications for Future Research

The results of our study have important implications for research. First, there is a growing body of work suggesting the existence of trade-offs associated with important outcomes in software development. Ramassubu and colleagues

[25] showed that the team configurations that are beneficial for productivity in GSD projects tend to be highly detrimental in terms of quality. Our results also suggest a trade-off between performance and quality stemming from the relationship of those outcomes with the structure of communication patterns. Future research should examine how communication structures change over time in response to changes in the tasks demands as well as other exogenous factors. In this way, we can gain insight on the specifics nature of the trade-off.

A second area of interest for future research should be devising ways of supporting software development teams to adapt their communication and coordination needs as their development activities evolve. Although a large body of work in the CHI and CSCW communities has proposed communication, coordination and awareness tools to support software development activities, our results suggest the need to go a step further and understand how the patterns of interaction relate to the tasks at hand.

Finally, agile methods emphasize communication, particularly, face-to-face interaction when the configuration of the teams so permits. Our results suggest that extending agile methods into a geographically distributed context may require some refocus on specific communication patterns. Certainly further research is required to understand the implications of such a change on the success of these methods as well as understanding how such a refocus can be achieved.

5.3 Practical Implications

Our results also have important practical implications. In particular the differential effect of our two measures of communication structure on rescheduled work items and defects suggests that project leaders need to, first, be able to recognize how communication patterns are evolving in a project and, second, encourage the right kind of communication in the teams. Recent work (e.g. [30]) has shown that managers that gather and distribute information among the geographically distributed members of a team help improve the performance of that team. Our results go a step further and argue that managers need to be able to assess the communication patterns and deficiencies that exist in a development team and support the establishment of communication paths that are structured in a particular way to help the team's outcomes.

6. REFERENCES

[1] T. J. Allen, *Managing the flow of technology : technology transfer and the dissemination of technological information within the R&D organization*. [Cambridge, Mass.: MIT Press], 1977.

[2] L. Argote, *Organizational learning : creating, retaining, and transferring knowledge*. Boston: Kluwer Academic, 1999.

[3] C. Bird*, et al.*, "Latent social structure in open source projects," 2008, pp. 24-35.

[4] W. F. Boh*, et al.*, "Learning from Experience in Software Development: A Multilevel Analysis," *Management Science,* vol. 53, pp. 1315-1331, 2007.

[5] S. Brown and K. Eisenhardt, "Product development: past research, present findings, and future directions," *Academy of Management Review,* vol. 20, pp. 343-378, 1995.

[6] R. Burt, *Brokerage and closure: An introduction to social capital*: Oxford University Press, USA, 2005.

[7] K. M. Carley*, et al.*, "Ora: Organization risk analyzer," Citeseer, 2004.

[8] M. Cataldo and J. D. Herbsleb, "Communication patterns in geographically distributed software development and engineers' contributions to the development effort," presented at the Proceedings of the 2008 international workshop on Cooperative and human aspects of software engineering, Leipzig, Germany, 2008.

[9] M. Cataldo*, et al.*, "Socio-technical congruence: a framework for assessing the impact of technical and work dependencies on software development productivity," presented at the Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement, Kaiserslautern, Germany, 2008.

[10] M. Cataldo*, et al.*, "Software Dependencies, Work Dependencies, and Their Impact on Failures," *IEEE Trans. Softw. Eng.,* vol. 35, pp. 864-878, 2009.

[11] M. Cataldo and S. Nambiar, "On the relationship between process maturity and geographic distribution: an empirical analysis of their impact on software quality," presented at the Proceedings of the the 7th joint

meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering, Amsterdam, The Netherlands, 2009.

[12] M. Cataldo and S. Nambiar, "The impact of geographic distribution and the nature of technical coupling on the quality of global software development projects," *Journal of Software Maintenance and Evolution: Research and Practice,* 2010.

[13] M. Cataldo*, et al.*, "Identification of coordination requirements: implications for the Design of collaboration and awareness tools," presented at the Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work, Banff, Alberta, Canada, 2006.

[14] J. S. Coleman, *Foundations of social theory*. Cambridge, MA: Harvard University Press, 1990.

[15] K. Crowston and J. Howison, "The social structure of free and open source software development," *First Monday,* vol. 10, p. 1ñ100, 2005.

[16] K. Ehrlich and K. Chang, "Leveraging expertise in global software teams: Going outside boundaries," presented at the Proceedings of the IEEE international conference on Global Software Engineering, 2006.

[17] R. Frost, "Jazz and the eclipse way of collaboration," *IEEE Software,* pp. 114-117, 2007.

[18] B. Gokpinar*, et al.*, "The Impact of Misalignment of Organizational Structure and Product Architecture on Quality in Complex Product Development," *Management Science,* vol. 56, pp. 468-484, 2010.

[19] D. E. Harter*, et al.*, "Effects of Process Maturity on Quality, Cycle Time, and Effort in Software Product Development," *Manage. Sci.,* vol. 46, pp. 451-466, 2000.

[20] J. D. Herbsleb and A. Mockus, "An Empirical Study of Speed and Communication in Globally Distributed Software Development," *IEEE Trans. Softw. Eng.,* vol. 29, pp. 481-494, 2003.

[21] P. Hinds and C. McGrath, "Structures that work: social structure, work structure and coordination ease in geographically distributed teams," 2006, pp. 343-352.

[22] D. Krackhardt, "Graph theoretical dimensions of informal organizations," *Computational organization theory,* pp. 89-111, 1994.

[23] M. Lindvall*, et al.*, "Empirical findings in agile methods," *Extreme Programming and Agile MethodsóXP/Agile Universe 2002,* pp. 81-92, 2002.

[24] M. E. J. Newman, "Models of the small world," *Journal of Statistical Physics,* vol. 101, pp. 819-841, 2000.

[25] N. Ramassubu*, et al.*, "Configuring Global Software Teams: A Multi-Company Analysis of Project Productivity, Quality, and Profits," presented at the International Conference on Software Engineering, Honolulu, Hawaii., 2011.

[26] C. R. B. d. Souza*, et al.*, "How a good software practice thwarts collaboration: the multiple roles of APIs in software development," presented at the Proceedings of the 12th ACM SIGSOFT twelfth international symposium on Foundations of software engineering, Newport Beach, CA, USA, 2004.

[27] B. Uzzi and J. Spiro, "Collaboration and creativity: The small world problem," *AJS,* vol. 111, pp. 447-504, 2005.

[28] P. Wagstrom*, et al.*, "Communication, Team Perforance and the Individual: Bridging Technical Dependencies," presented at the Academy of Management Annual Meeting, Montreal, Canada, 2010.

[29] D. J. Watts and S. H. Strogatz, "Collective dynamics of ësmall-worldínetworks," *Nature,* vol. 393, pp. 440-442, 1998.

[30] S. P. Weisband, "Leadership at a distance: research in technologically-supported work," 2007.

[31] T. Wolf*, et al.*, "Predicting build failures using social network analysis on developer communication," 2009, pp. 1-11.

[32] T. Zimmermann and N. Nagappan, "Predicting defects with program dependencies," presented at the Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement, 2009