9-2007

Globally Distributed Software Development Project Performance: An Empirical Analysis

Narayanasamy RAMASUBBU
Singapore Management University, nramasub@smu.edu.sg

Rajesh Krishna BALAN
Singapore Management University, rajesh@smu.edu.sg

DOI: https://doi.org/10.1145/1287624.1287643

Globally Distributed Software Development Project Performance: An Empirical Analysis

Narayan Ramasubbu, Rajesh Krishna Balan
Singapore Management University
nramasub, rajesh @ smu.edu.sg

ABSTRACT
Software firms are increasingly distributing their software development effort across multiple locations. In this paper we present the results of a two year field study that investigated the effects of dispersion on the productivity and quality of distributed software development. We first develop a model of distributed software development. We then use the model, along with our empirically observed data, to understand the consequences of dispersion on software project performance. Our analysis reveals that, even in high process maturity environments, a) dispersion significantly reduces development productivity and has effects on conformance quality, and b) these negative effects of dispersion can be significantly mitigated through deployment of structured software engineering processes.

Categories and Subject Descriptors
D.2.9 [**Management**]: productivity, programming teams, software process models, software quality assurance

General Terms
Economics, Management

Keywords
Globally distributed software development, software engineering economics, quality management, empirical analysis

1. INTRODUCTION
Globally distributed software development achieves division of labor by dispersing software development tasks among several remote development centers. This mode of software development has become a popular business model for software organizations. There are several compelling business reasons supporting the adoption of distributed software development: 1) ability to extend work beyond the regular office hours at a single site, 2) software development costs at offshore centers, like in India, are as much as four times less expensive [42], 3) the capabilities of workforce in remote centers located in emerging economies have improved significantly in the recent years [10], 4) advances in information and communication technology have facilitated easier collaboration between remote workforce [8, 9].

ESEC-FSE'07, September 3–7, 2007, Cavtat near Dubrovnik, Croatia.

At the same time several challenges in distributed software development have been reported. For example, Mockus and Weiss [40] report that distributing product maintenance work across global development centers increases the cycle time of a project. Likewise, Herbsleb *et. al.* [24-26] report that, compared to same-site work, cross-site work takes a much longer time and requires more effort for work of similar size and complexity. Also behavioral researchers investigating distributed work report that a remote workforce, even with advanced technologies in place, often encounter difficulties in coordination and administration that lead to decreased project performance [11, 30, 39, 43].

Structured and disciplined software engineering processes have often been advocated as a key remedy for addressing the aforementioned challenges [13, 20, 21, 28]. In this paper, we report our findings, from our field study, on the effectiveness of deploying structured software engineering processes and stringent quality management practices in globally distributed software development. The main contribution of this paper is in developing empirical models of distributed software project performance and verifying them using data collected from large scale, real world projects. In doing so, we answer the following research questions:

1. To what extent does "dispersion" in software tasks affect software productivity and quality?
2. To what extent can investments in structured software engineering processes mitigate the effect of dispersion?
3. What are the relative effects of individual quality management practices in improving distributed project performance?

2. MODELING GLOBALLY DISTRIBUTED SOFTWARE DEVELOPMENT
To answer the above questions, a model that captures the individual effects of factors that influence global software development project performance is necessary. The modeling framework for this study is developed based on the economic view of software development [2, 3, 7]. Researchers using this framework treat software development as a production process, and model software performance indicators, such as productivity and quality, as a function of personnel related factors and software methodology related factors. Prior studies using the economic view of software development have predominantly focused on co-located software development scenarios [6, 22, 32, 33]. In this study we extend this framework to address distributed software development. Also, prior software engineering research studies have not extensively focused on right mix of quality practices in different stages of product development to improve the net outcome in a project. To address this, we study the effects of prevention, appraisal and failure-based quality activities on distributed project performance.

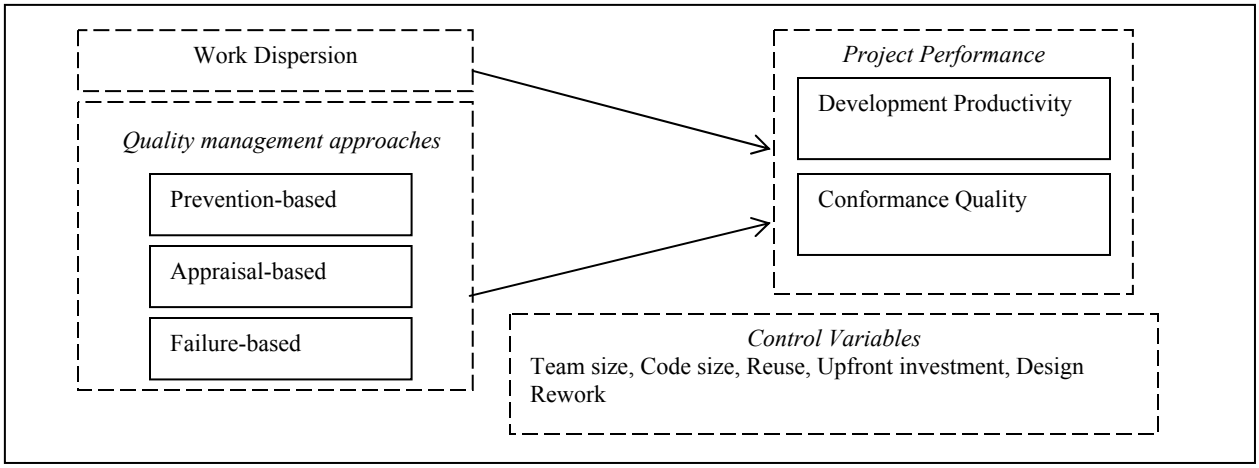


Figure 1. Research Model

Prevention	***Appraisal***	***Failure***
• Programming training • Business domain training • Process training • Configuration management • Task Planning and Scheduling	• Requirement, Specification and Design reviews • Code inspection • Status reviews	• Unit testing • Module testing • Integration Testing • System testing • Error tracking and correction

Figure 2. Categorization of individual quality management practices

2.1 Research Model

Figure 1 gives a pictorial overview of our research model. On the left side of the model are the factors affecting software development; namely work dispersion and Quality Management Approaches (QMA). On the right side of the model are the project performance indicators. To model the other factors affecting both project performance and software development, and to understand the development process in more detail, we introduce a number of control variables. We explain work dispersion, quality management approaches, project performance, and control variables in more detail in the next few subsections.

2.1.1 Work Dispersion

Work dispersion is a key variable in our model, and describes how distributed a project's development process is. We measure work dispersion between development centers using a variable similar to the Herfindahl-Hirschman Index [47]. This index is a well tested and widely used measure that quantifies how diversified a large corporation or a particular industry is. Since there are only two development centers in our data set (explained in more detail in Section 3), the work dispersion measure is defined as

$$\textit{Work dispersion} = 100^2 - (\%\ \textit{effort at first development center})^2 - (\%\ \textit{effort at second development center})^2$$

A value of zero for our work dispersion measure indicates that the software project is completely co-located, and an increasing value represents increasing levels of work dispersion. For example, when 90% of the project is performed at one development center with the remaining 10% performed at another center, the value of our work dispersion variable is 1800 ($100^2 - (90)^2 - (10)^2 = 1800$). Similarly, for an 80/20 scenario, the dispersion variable value is 3200 ($100^2 - (80)^2 - (20)^2 = 3200$). The maximum value of dispersion in the two development center scenario is 5000 when the work allocation percentage is 50/50.

2.1.2 Quality Management Approaches

A key component of the model is determining how to categorize software management quality. Instead of creating our own categories, we use the well studied and accepted categorization used in manufacturing quality research [16, 37, 41]. This categorization has three components; prevention-based, appraisal-based, and failure-based QMAs. Figure 2 provides a detailed breakdown of the elements of each approach.

Prevention-based approach: Prevention-based quality management practices in software development involve activities such as training that are primarily done to avoid the occurrence of errors. For our model, we compute a score for this approach based on the percentage of total development effort spent on training, project planning and configuration management activities.

Appraisal-based approach: Appraisal-based activities involve proactively assessing progress, performance and quality of intermediate artifacts at various stages of development. We assign a score for this approach based on the percentage of total development effort spent on peer reviews of requirement, design, status reviews and code inspection.

Failure-based approach: Failure-based quality approaches include testing the adherence of applications to customer specifications and subsequent defect correction activities. We assign a score for this approach based on the percentage of total development effort spent on unit tests, module integration tests and system tests.

2.1.3 Project Performance

Similar to past software engineering economics studies [21, 34], we use two different performance indicators to determine the quality of the software product. They are:

1) **Development productivity**: Development productivity is defined as the ratio of software code size in function points to the total development effort in person-hours. The advantage of function point measurement for code size is that it incorporates measures of customer perceived software functionality as well as complexity [29]. Function points have also been shown to be a reliable output measure in the context of commercial systems [31]. The development effort includes effort incurred in all stages of development until the customer signs off the project.

2) **Conformance Quality:** Our quality measure captures the number of unique problems reported by customers during the acceptance tests and production trials before the project signoff. It is calculated as follows:

$$Conformance\ Quality = \frac{1}{(defects + 1)}$$

This reciprocal formulation represents quality as a decreasing function as the number of defects increases.

2.1.4 Control Variables

We introduced a number of control variables into the model. These variables serve two purposes; a) provide a deeper understanding of the distributed software development process, and b) allow us to create empirical models that can be computed (shown in Section 2.2). We use five variables; two that affect primarily productivity, two that affect primarily quality, and one that affects both.

2.1.4.1 Productivity Variables

We used two control variables that affect primarily productivity; Team Size and Reuse.

Team Size: Team size is the headcount of the number of persons involved in the project. Team size is expected to be a good surrogate for the coordination difficulties that could occur within the software project team. An increased team size poses difficulties in both administrative and expert coordination [17].

Reuse: Reuse in this study is measured as the percentage of lines of code that have been utilized from the generic code libraries maintained centrally in the knowledge database at our research site. All reused modules and objects in the projects we studied were maintained with unique tags for readability and hence could be easily identified in applications. Reuse in software enables developers to use standardized routines that have been stored in organization-wide repositories and libraries to accomplish certain functionality. Usage of such standardized and pre-tested functions helps developers to avoid reinventing the wheel, and to focus on customer specific needs. Thus we believe reuse plays an important role in impacting development productivity. Reuse has other indirect influences such as a potential effect on software quality as well. However, similar to other software modeling studies [4], we use it primarily to observe productivity.

2.1.4.2 Quality Variables

We used two control variables that affect primarily quality; Code Size, and Upfront Investment.

Code Size: Code size is measured as function points over the entire project code base. Code size is a widely recognized control variable for software quality models as software size captures both the magnitude of the project and much of the complexity involved in developing the application.

Upfront Investment: Upfront investment is measured as the percentage of total effort spent during the requirements and design stages of the life cycle. Higher levels of investment in activities done before commencing actual coding of system, such as requirement analysis and high level design, have been shown to positively influence system quality [34].

2.1.4.3 Common Variable

We use a control variable, design rework, which affects both productivity and quality as it gives deeper insights into the effect of dispersion on the overall project performance.

Design Rework: As stated earlier, we measure code size in terms of function points. Hence, we define design rework as the effort, in terms of person hours, spent per function point to implement the new design. Agreeing on a common non-volatile design early in the project life cycle is likely to be very important in a distributed environment. We hypothesize that changing the basic design framework often results in cascading changes and rework in individual components that affects project performance. Hence, we also account for design rework when determining the effect of dispersion on project performance.

2.2 Empirical Equations

Given the research model presented in Section 2.1, we formulate the following empirical formulations:

Development productivity = *f (conformance quality, work dispersion, prevention-based approach, appraisal-based approach, failure-based approach, reuse, design rework, team size)* •••••• **(Eq. 1)**

Conformance quality = *f (development productivity, work dispersion, prevention-based approach, appraisal-based approach, failure-based approach, design rework, code size, upfront investment)* •••••• **(Eq. 2)**

Equation (1) states that development productivity is functionally dependent on conformance quality, work dispersion, the various quality management approaches as well as the values of certain variables. Equation (2) states that conformance quality is functionality dependent on development productivity, work dispersion, the various quality management approaches, and some control variables. Note that development productivity depends on conformance quality and vice versa and also that most of the variables in the two equations are common. The unique control *variables* (productivity has team size and reuse etc.) were introduced specifically to resolve this circular dependency.

Before converting the generic functional equations (Equations 1 and 2) into the final equations (with coefficients and errors etc.), we note that the effects of size and effort on quality and productivity are not linear, and that scale economies exist in software development [5]. Hence, we posit that the effects of

dispersion and quality management practices on both conformance quality and development productivity are not linear either. We thus use a general multiplicative specification and derive the final empirical equations (equations (3) and (4)) using a log-log transformation of equations (1) and (2). We present the values obtained using these equations in Section 4.

$$\ln(\text{development productivity}) = \alpha_0 + \alpha_1 * \ln(\text{conformance quality}) + \alpha_2 * \ln(\text{work dispersion}) + \alpha_3 * \ln(\text{prevention}) + \alpha_4 * \ln(\text{appraisal}) + \alpha_5 * \ln(\text{failure}) + \alpha_6 * \ln(\text{reuse}) + \alpha_7 * \ln(\text{design rework}) + \alpha_8 * \ln(\text{team size}) + \varepsilon_1 \quad \text{(Eq. 3)}$$

$$\ln(\text{conformance quality}) = \beta_0 + \beta_1 * \ln(\text{development productivity}) + \beta_2 * \ln(\text{work dispersion}) + \beta_3 * \ln(\text{prevention}) + \beta_4 * \ln(\text{appraisal}) + \beta_5 * \ln(\text{failure}) + \beta_6 * \ln(\text{design rework}) + \beta_7 * \ln(\text{Code size}) + \beta_8 * \ln(\text{Upfront investment}) + \varepsilon_2 \quad \text{(Eq. 4)}$$

3. DATA COLLECTION AND METHODOLOGY

In this section we provide details of our research site, and data collection procedure, and present our empirical models. We collected detailed distributed development data from a leading software service company that employs over 19,000 people in 17 countries and has annual revenue of more than one billion dollars, at the time of our data collection. This firm provides an ideal research setting to study software productivity and quality in a distributed development scenario because the firm has adopted a global delivery model for its services, and employs a very high maturity development processes. Our research site was assessed to operate at the highest maturity level (level 5) of the software Capability Maturity Model instituted by the Software Engineering Institute at Carnegie Mellon University. High maturity operations at our research site helped us to gather reliable data to empirically investigate our research questions.

Our data collection involved gathering information on forty-two completed projects in a recent two year time period. During this time, one of the authors was present in the field and observed the software development processes in the projects using an ethnographic observation approach. To complement our understanding of the software processes and culture at our site, we conducted structured interviews with two senior business development managers, four project managers, and with ten randomly selected project team members.

We obtained the data required for the various components in our model (project performance, control variables, etc.) from an internal company software engineering process database maintained by the quality division of the organization. Each of the forty-two projects studied in this paper were required to regularly report these values in order to achieve CMM-level 5 compliance. The company routinely internally audited these project teams to ensure that they were compliant with the reporting requirements. In addition, the quantitative data we gathered had been audited by an independent external assessment auditor for CMM and ISO 9001 conformance certifications. We are thus confident that the data used in this paper is reliable and of high quality and that we have a rich understanding of the context in which these software projects were executed.

All the projects studied were development projects of commercial business applications using high level programming languages and relational databases. We found very little variance across different projects with respect to adherence to key process areas specified by CMM-level 5.

Each of the forty-two projects studied was executed using two software development centers, one in India and one in the United States. The primary reason for this split was because the clients were located in North America with additional development resources located in India. It is interesting to note that human resource allocation across the two centers was primarily sourced from the Indian center.

The project development process occurred as follows; When the firm won a customer bid to build an application, the business development manager would form the project team. As the project work started, a part of the team traveled from India to the development center in the United States and typically stayed there until project sign off. There were occasional rotations among team members across the development centers.

Variable	Unit	Mean	Std. Dev.	Min	Max
Productivity	Func. Pts / Person Hrs	0.26	0.15	0.02	0.64
Quality	1/ (No. defects +1)	0.28	0.35	0.01	1.00
Dispersion	Unit less measure	4220.58	739.66	1781.80	4994.30
Prevention QMA	% of total project hrs	4.57	4.56	1.00	25.67
Appraisal QMA	% of total project hrs	22.45	7.59	8.89	42.69
Failure QMA	% of total project hrs	19.85	9.18	4.11	44.53
Code Size	No. of Function Points	2191.83	2927.72	33.00	18247.00
Team Size	No. project personnel	11.19	6.43	2.00	30.00
Design Rework	% of total project hrs	0.44	0.72	0.00	4.39
Upfront Investment	% of total project hrs	13.64	7.71	0.43	32.32
Reuse	% of lines of code	5.26	3.25	1.00	20.00

Figure 3. Summary Statistics of the Variables Used in the Analysis

Variables	Model 1 Development Productivity		Model 2 Conformance Quality	
Development Productivity	NA		**0.893**** (0.046)	β_1
Conformance Quality	**-0.308***** (0.002)	α_1	NA	
Dispersion	**-1.018***** (0.004)	α_2	-0.621 (0.444)	β_2
Prevention-based quality management	0.059 (0.584)	α_3	**0.470**** (0.026)	β_3
Appraisal-based quality management	**0.573**** (0.014)	α_4	0.363 (0.549)	β_4
Failure-based quality management	**0.324**** (0.035)	α_5	**0.656**** (0.037)	β_5
Reuse	**0.845***** (0.000)	α_6	NA	
Design Rework	-0.106 (0.177)	α_7	0.233 (0.229)	β_6
Team Size	-0.182 (0.247)	α_8	NA	
Code Size	NA		**-1.067***** (0.000)	β_7
Upfront Investment	NA		**0.375**** (0.049)	β_8
Constant	2.530 (0.418)	α_0	8.050 (0.236)	β_0
F (8, 33)	**10.89***** **(0.000)**		**4.89***** **(0.001)**	
Observations	**42**		**42**	

This figure shows the regression results of our data using equations 3 (model 1) and 4 (model 2). Probability-values are shown in parentheses; results significant at 5% are indicated by **; results significant at 1% are indicated by ***. Other values, which are not in bold, are not statistically significant. We use a two-tailed hypothesis test. The values indicate the effect of each variable on productivity (model 1) and quality (model 2). E.g, Reuse has a strong positive effect on productivity while Code Size has a strong negative effect on quality.

Figure 4. Regression Results

4. DATA ANALYSIS, MODEL VERIFICATION AND RESULTS

The empirical models described in section 2, indicate the presence of endogeneity between conformance quality and development productivity. That is, development productivity and conformance quality affect each other. We confirmed (with 1% significance level) the presence of such an effect in our dataset by using the Durbin-Wu-Hausman endogeneity test [15]. Because of the presence of endogenous variables in our models, classical Ordinary Least Square regression is not suitable. Therefore, we used an instrumental variable regression method, Two Stage Least Squares, to estimate the coefficients of equations (3) and (4).

The summary statistics of our data set is presented in Figure 3 while the results of our regression tests on our data are presented in Figure 4. Overall, the results of our regression analysis indicate that our empirical models are valid. The Fisher statistic (F) values for both our empirical models are significant at 1% and indicates that our regression results are statistically valid.

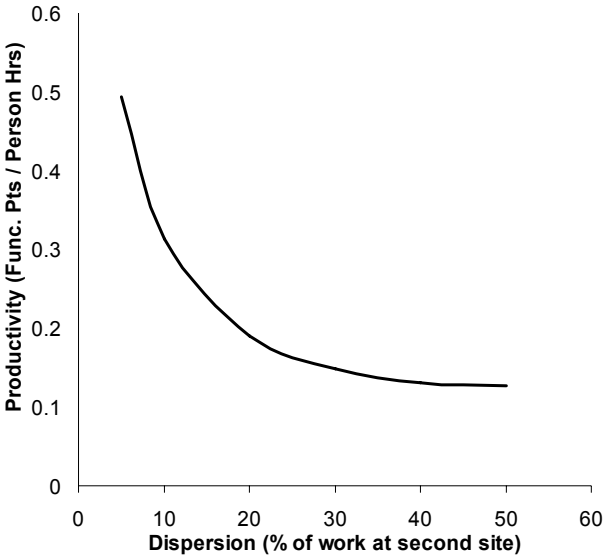
4.1. Results: Effect of Dispersion on Project Performance

Our results indicate that work dispersion, even in high process maturity environments, negatively affects development productivity. However, our analysis, as shown in Figure 5, indicates an exponential decrease in productivity as dispersion increases. Thus the marginal decrease in productivity is much higher as dispersion starts to increase and this has to be taken into consideration when teams initiate distributed development.

Our analysis indicates that in high process maturity environments, dispersion does not have a statistically significant direct effect on conformance quality. However, dispersion does have an indirect effect on conformance quality because of the endogeneity present between productivity and conformance quality.

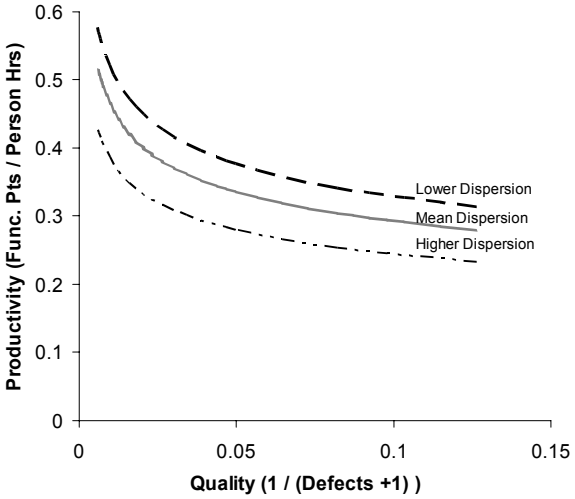
Figure 6 shows an example of this indirect effect. In the figure, we plot productivity against quality for various levels of dispersion. Note that while the shape of the curves look about the same, different dispersion levels affect two things; 1) the rate at which productivity decreases as you increase your quality expectations. In particular, highly dispersed teams will have to spend more time to create high quality code relative to less dispersed teams. 2) When dispersion is high, the productivity

achievable, for the same output quality, is significantly affected (compare the three dispersion lines in the graph).



We plotted the graph by holding all variables at their mean levels and using the regression results for model 1 (Equation 3)

Figure 5. Effect of Dispersion on Productivity

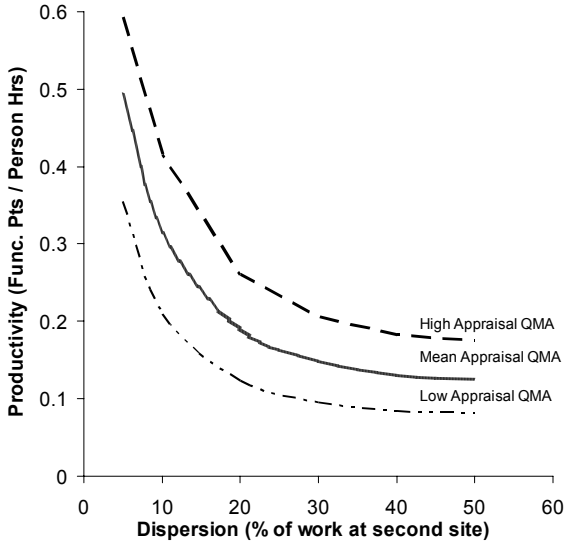


We see that for a given quality output requirement, projects with higher dispersion show lower development productivity.

Figure 6. Productivity – Quality Tradeoff

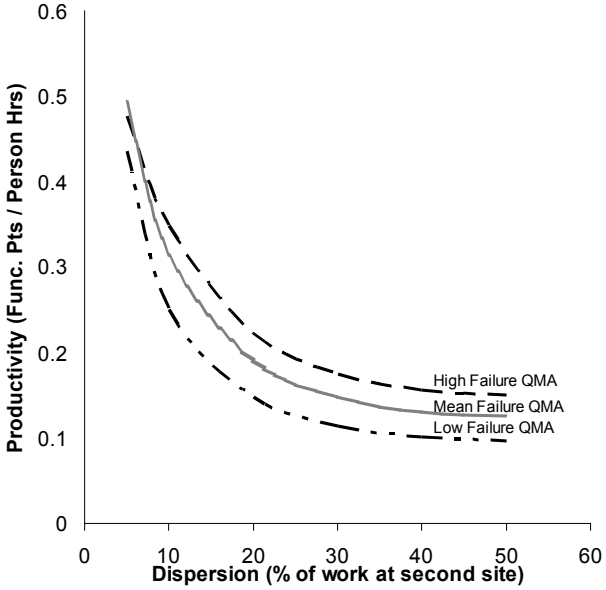
4.2. Results: Reducing Dispersion Effects through QMA

Our results show that investments in structured quality management practices, by using processes based on the three approaches described in Section 2.1.2, can mitigate the negative effect of work dispersion on project performance. For example, ceteris paribus, we notice that a 1% loss of productivity caused by increased work dispersion, can be reduced to just a 0.1% loss by investing in increased appraisal and failure-based quality management processes. Figure 7 shows the effect of appraisal-based practices in reducing the productivity loss of dispersion while Figure 8 shows the same result for failure-based practices.



We see that more appraisal QMA effort helps to reduce the productivity loss caused by dispersion.

Figure 7. Mitigating Dispersion Effect - Appraisal QMA



We see that more Failure QMA effort helps to reduce the productivity loss caused by dispersion.

Figure 8. Mitigating Dispersion Effect - Failure QMA

4.3. Results: Relative Effects of Different QMAs

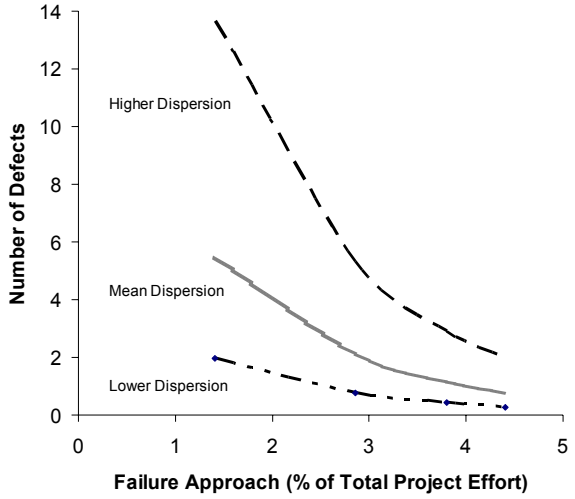
In this section, we discuss the relative effects of each QMA on distributed project performance.

Effect on Productivity: From our results we find that appraisal-based approaches have the highest impact on development productivity followed by failure-based approaches, Prevention-

based approaches seem to have no statistically significant effect on productivity.

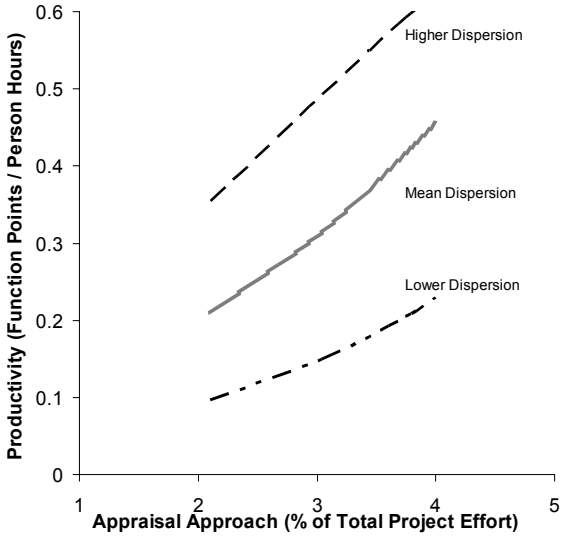
Effect on Conformance Quality: We found that failure-based approaches have the highest impact on conformance quality followed by prevention-based approaches. Appraisal-based approaches seem to have no statistically significant effect on conformance quality.

Figure 9 shows the effect that failure-based approaches have on quality while Figure 10 shows the effect that appraisal-based approaches have on productivity.



We see that more Failure QMA effort helps to reduce the number of defects across different dispersion levels

Figure 9. Effect of Failure Approach on Quality



More Appraisal QMA effort improves development productivity across different dispersion levels

Figure 10. Effect of Appraisal Approach on Productivity

Our analysis indicates that no one approach is best and that the approach to choose greatly depends on the circumstances. For example, failure-based approaches appear to be the best option to improve quality and appraisal-based approaches seem to be best at improving productivity. However, we have to caution that these trends 1) were observed in high process maturity environments, 2) were for data that was tracked for only a one to two year period, and 3) do not capture all the subtleties involved in software development. Hence, they are only indicative of possible trends and may not reflect the full potential of each approach. For example, even though the data suggests that prevention-based approaches have minimal effect, in reality, the effects of activities such as training 1) have longer term impacts, and 2) impact the effectiveness of other approaches as well. Hence, these trends should not be construed as prescriptive.

4.4. Summary of Results

We presented our analysis to answer the three question posed in the Introduction. Namely:

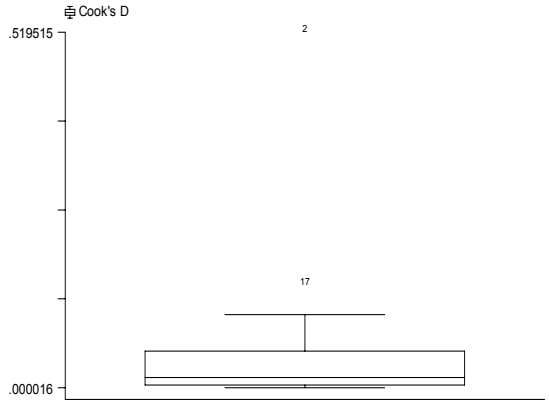
1. *To what extent does "dispersion" in software tasks affect software productivity and quality?* We found, as shown in Section 4.1, that dispersion has a significant effect on productivity and a harder-to-capture secondary effect on quality.
2. *To what extent can investments in structured software engineering processes mitigate the effect of dispersion?* We showed, in Section 4.2, that the effect of dispersion can be significantly mitigated through the use of structured software engineering processes.
3. *What are the relative effects of individual quality management practices in improving distributed project performance?* Finally, we showed, in Section 4.3, that different QMAs have significantly different impacts on different dimensions of project performance.

5. DISCUSSION

In this section, we discuss some of the questions raised by this study. In particular, we discuss the robustness and the limitations of this study as well as provide some intuitive explanations for some of the observed effects.

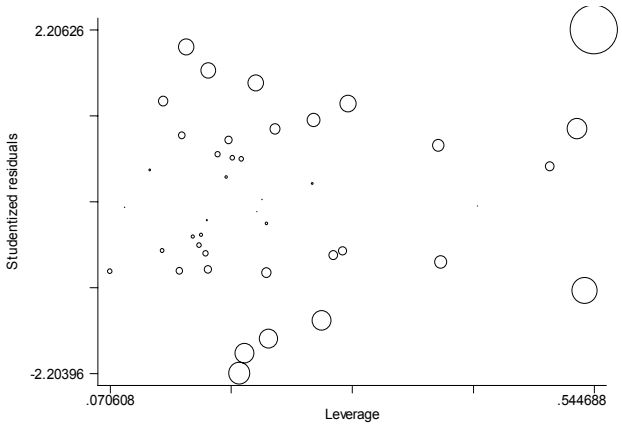
5.1 Robustness of Analysis

We verified the robustness of our data analysis in multiple ways. First, we checked for outliers by deriving Cook's distance statistic. Figures 11 and 12 show the cook's distance plot and residual plot of our data. This revealed two observations that appeared as outliers. We removed these observations from our data set and re-computed the regression results. This revealed no significant changes in any of our results.



The box-plot shows the distribution of the Cook's D statistic for our regression results. The two outliers lie above the main box-plot.

Figure 11 Cooks' Distance Plot



This graph plots standardized residuals (y-axis) against leverage (x-axis) and shows a uniform funnel pattern of distribution indicating that no on data point is significantly impacting the results

Figure 12 Residual Leverage Plot

Further, we checked for multicollinearity problems by analyzing the Variance Inflation Factor (VIF) statistic of independent variables in our models. The maximum value of the VIF statistic in the productivity model was 1.96, and 3.65 in the quality model, which are both well below permissible limits [19].

5.2 Why do we see these Results?

To determine why dispersion negatively affect productivity, we noticed (from our field observations) that distributed team members often had difficulties in managing uncertainties caused by interdependent tasks. Uncertainty in the observed projects' interdependent tasks arose primarily because of 1) information asymmetry between the remote teams, and 2) ambiguous authority. Information asymmetry (either related to customer initiated changes, updated schedules, etc.) between remote teams hinders coordination and task orchestration, which in turn affects project performance. Ambiguous authority refers to the break down in a planner's decision making authority because of a lack of complete control over the processes at both the remote sites. Ambiguous authority leads to poor project management and hence eventually impacts project performance.

One reason for the observed differences in the relative effectiveness of different QMAs could be because of the different learning related benefits that these approaches facilitate. For example, preventive and appraisal-based approaches facilitate learning-before-doing whereas failure-based approach facilitates learning-by-doing. Past organizational learning research has reported different effects of learning-before-doing and learning-by-doing [44]. We believe that in a distributed development environment, learning-by-doing as facilitated by a failure based approach might be the most effective learning method.

In contrast to behavioral oriented studies, we have not extensively focused on cultural issues to explain the trends. Cultural effects are minimal in our data set because all the human resources, for both the development centers, were effectively sourced from a single country's employee base. Further, we assessed if there were significant differences in human resource practices among the different centers by analyzing employee performance appraisal templates, code of conduct guidelines, and incentive structures from different centers. We found that the firm had a uniform human resource policy throughout the world and there were no significant differences across development centers. Also, the research site was recently assessed, for maturity in human resources management practices, and was certified at level 5 of the People Capability Maturity Model standards [14].

5.3 Limitations of Study

Some of the limitations of this study are: 1) Our model did not consider the task level interdependencies among individual team members. We, instead, focused on aggregate project level data and did analyze the interdependencies at the team member level. 2) Our data set was collected from a firm that had already attained high process maturity operational capability. Hence we can not generalize our results to scenarios with different process maturity levels. 3) All the projects we observed were custom, business application software development projects. Hence, even though the broad results we observe in this study are still relevant, other types of software development projects such as re-engineering, product development or maintenance projects might require additional analysis.

6. RELATED WORK

There are, two streams of research work that are directly relevant to this research study. First, there is a growing body of work that examine globally distributed software development from a software engineering point of view. A variety of practical issues faced by practitioners of distributed development was presented in the special issues of IEEE software [23] and Communications of the ACM [1]. Also, there are several experience reports that elicit lessons learnt from real world projects [27, 36]. There is also an emerging body of work that specifically investigates the appropriate software process frameworks that are suitable for distributed development [45, 46] and specific architectural methods that could be employed to facilitate distributed division of labor[35, 40].

A second stream of research work that is relevant is the quality management literature that provides insights for analyzing the effectiveness of individual quality management practices in improving project performance. The classical model of economics

of quality [37] posits that there exists a cost minimizing conformance quality level resulting from the tradeoffs between the costs of attaining higher quality (prevention and appraisal) and the costs of having produced poor quality (failure costs). However, an alternate school of thought [12, 18] believes that producing high quality products is always less costly and posits a zero defects process as the optimal in the longer run. Recent studies extend the above mentioned operations oriented research views to the software context [34, 38, 48].

7. CONCLUSION

In this paper, we presented data collected at a leading software company that has multiple projects that distribute work between India and the United States of America. This dispersion of work had significant effects on productivity and, indirectly, on the quality of the software. Fortunately, this effect can be mitigated by using structured quality management approaches.

This paper suggests that companies need to account for the inevitable loss in productivity and quality when deciding to move software production to a second or third location (to reduce labour costs, etc.). It also suggests that companies that institute high quality software processes are far more likely to overcome the effects of dispersion than companies that don't. However, it is currently difficult to specifically state which processes are best suited for which types of companies – this remains an item for future work.

In addition to determining the processes that best suit different companies, we plan to work on the following pieces of future work; a) determining how to reduce the interdependence between tasks in distributed environments (this will reduce the loss of productivity), b) understanding how these results apply to other types of companies, and c) developing a general model that applies across different types software methodologies.

8. REFERENCES

[1] P. J. Ågerfalk and B. Fitzgerald, "Flexible and distributed software processes: old petunias in new bowls?: Introduction," *Communications of the ACM,* vol. 49, pp. 26-34, 2006.

[2] R. D. Banker, S. M. Datar, and C. F. Kemerer, "A model to evaluate variables impacting the productivity of software maintenance projects," *Management Science,* vol. 37, pp. 1-18, 1991.

[3] R. D. Banker, G. B. Davis, and S. A. Slaughter, "Software development practices, software complexity, and software maintenance: A field study," *Management Science,* vol. 44, pp. 433-450, 1998.

[4] R. D. Banker and R. J. Kauffman, "Reuse and productivity in integrated computer-aided software engineering: An Empirical Study," *MIS Quarterly,* vol. 15, pp. 375-401, 1991.

[5] R. D. Banker and C. F. Kemerer, "Scale economies in new software development," *IEEE Transactions on software engineering,* vol. 15, pp. 1199-1205, 1989.

[6] R. D. Banker and S. A. Slaughter, "The moderating effects of structure and volatility and complexity in software environment," *Information Systems Research,* vol. 11, pp. 219-240, 2000.

[7] B. W. Boehm, *Software engineering economics*. Upper Saddle River, NJ: Prentice Hall, 1981.

[8] F. Cairncross, *The Death of Distance: How the Communications Revolution Will Change Our Lives*. Boston, MA: Harvard Business School Press, 1997.

[9] E. Carmel, *Global software teams: Collaborating across borders and time zones*. Upper Saddle River, NJ: Prentice Hall, 1999.

[10] E. Carmel and R. Agarwal, "The maturation of offshore sourcing of information technology work," *MIS Quarterly Executive,* vol. 1, pp. 65-76, 2002.

[11] C. D. Cramton, "The mutual knowledge problem and its consequences for dispersed collaboration," *Organization Science,* vol. 12, pp. 346-371, 2001.

[12] P. B. Crosby, *Quality is Free: The art of making quality certain*. New York: McGraw-Hill, 1979.

[13] B. Curtis, "The global pursuit of process maturity," *IEEE Software,* vol. 17, pp. 76-78, 2000.

[14] B. Curtis, W. Hefley, and S. Miller, "People Capability Maturity Model," Carnegie Mellon University, Pittsburgh CMU/SEI-2001-MM-01, 2001.

[15] R. Davidson and J. G. Mackinnon, *Estimation and Inference in Econometrics*. New York: Oxford University Press, 1993.

[16] W. E. Deming, *Quality, Productivity and Competitive Position*. Cambridge, MA: MIT Center for Advanced Engineering Study, 1982.

[17] S. Faraj and L. Sproull, "Coordinating expertise in software development teams," *Management Science,* vol. 46, pp. 1554-1568, 2000.

[18] C. H. Fine, "A quality control model with learning effects," *Operations Research,* vol. 36, pp. 437-444, 1988.

[19] J. Fox, *Applied Regression, Linear Models, and Related Methods*. Thousand Oaks, California: Sage, 1997.

[20] A. Gopal, T. Mukhopadhyay, and M. S. Krishnan, "The role of software processes and communication in offshore software development," *Communications of the ACM,* vol. 45, pp. 193-200, 2002.

[21] D. E. Harter, M. S. Krishnan, and S. A. Slaughter, "Effects of process maturity on quality, cycle time, and effort in software product development," *Management Science,* vol. 46, pp. 451-466, 2000.

[22] D. E. Harter and S. A. Slaughter, "Quality Improvement and Infrastructure Activity Costs in Software Development: A Longitudinal Analysis," *Management Science,* vol. 49, pp. 784-800, 2003.

[23] J. Herbsleb and D. Moitra, "Global Software Development," *IEEE Software,* vol. 18, pp. 16-20, 2001.

[24] J. D. Herbsleb and A. Mockus, "An Empirical Study of Speed and Communication in Globally Distributed Software Development," *IEEE Transactions on Software Engineering,* vol. 29, pp. 481-494, 2003.

[25] J. D. Herbsleb, A. Mockus, T. A. Finholt, and R. E. Grinter, "Distance, dependencies, and delay in a global collaboration," in *2000 ACM conference on computer supported cooperative work*, Philadelphia, PA, 2000, pp. 319-328.

[26] J. D. Herbsleb, A. Mockus, T. A. Finholt, and R. E. Grinter, "An empirical study of global software development: distance and speed," in *23rd International Conference on Software Engineering*, Toronto, Canada, 2001, pp. 81-90.

[27] J. D. Herbsleb, D. J. Paulish, and M. Bass, "Global software development at Siemens: experience from nine projects," in *International Conference on Software Engineering*, St. Louis, MO, USA, 2005, pp. 524-533.

[28] W. S. Humphrey, "Characterizing the software process: a maturity framework," *IEEE Software,* vol. 5, pp. 73-79, 1988.

[29] IFPUG, "Function point counting practices manual," International Function Point Users group, Mequon, Wisconsin 1999.

[30] S. L. Jarvenpaa and D. E. Leidner, "Communication and Trust in Global Virtual Teams," *Organization Science,* vol. 10, pp. 791-815, 1999.

[31] C. F. Kemerer, "Reliability of function points measurement: a field experiment," *Communications of the ACM,* vol. 36, pp. 85-97, 1993.

[32] M. S. Krishnan, "The role of team factors in software cost and quality: an empirical analysis," *Information technology and people,* vol. 11, pp. 20-35, 1998.

[33] M. S. Krishnan and M. I. Kellner, "Measuring Process Consistency: Implications for Reducing Software Defects," *IEEE Transactions on Software Engineering,* vol. 25, pp. 800-815, 1999.

[34] M. S. Krishnan, C. H. Kriebel, S. Kekre, and T. Mukhopadhyay, "An empirical analysis of productivity and quality in software products," *Management Science,* vol. 46, pp. 745-759, 2000.

[35] O.-K. Lee, P. Banerjee, K. H. Lim, K. Kumar, J. v. Hillegersberg, and K. K. Wei, "Aligning IT components to achieve agility in globally distributed system development," *Communications of the ACM,* vol. 49, pp. 48-54, 2006.

[36] E. Lindgvist, B. Lundell, and B. Lings, "Distributed development in an intra-national, intra-organisational context: an experience report," in *International Conference on Software Engineering*, Shanghai, China, 2006, pp. 80-86.

[37] D. M. Lundvall and J. M. Juran, "Quality Costs," in *Quality control handbook*, 3 ed, J. M. Juran, Ed. San Francisco, CA: McGraw-hill, 1974.

[38] A. MacCormack, C. F. Kemerer, M. Cusumano, and B. Crandall, "Trade-offs between productivity and quality in selecting software development practices," *IEEE Software,* vol. 20, pp. 78-85, 2003.

[39] M. L. Maznevski and K. M. Chudoba, "Bridging Space over time: global virtual team dynamic and effectiveness," *Organization Science,* vol. 11, pp. 473-492, 2000.

[40] A. Mockus and D. M. Weiss, "Globalization by chunking: A quantitative approach," *IEEE Software,* vol. 18, pp. 30-37, 2001.

[41] P. Nandakumar, S. M. Datar, and R. Akella, "Models for measuring and accounting for cost of conformance quality," *Management Science,* vol. 39, pp. 1-16, 1993.

[42] Nasscom-McKinsey, "NASSCOM-McKinsey Report," National Association of Software and Service Companies, New Delhi 2002.

[43] G. M. Olson and J. S. Olson, "Distance Matters," *Human-computer interaction,* vol. 15, pp. 139-178, 2000.

[44] G. P. Pisano, "Knowledge, integration and the locus of learning: An empirical analysis of process development," *Strategic Management Journal,* vol. 15, pp. 85-100, 1994.

[45] N. Ramasubbu, M. S. Krishnan, and P. Kompalli, "Leveraging global resources: A process maturity framework for managing distributed development," *IEEE Software,* vol. 22, pp. 80-86, 2005.

[46] B. Ramesh, L. Cao, K. Mohan, and P. Xu, "Can distributed software development be agile?," *Communications of the ACM,* vol. 49, pp. 41-46, 2006.

[47] F. M. Scherer and D. Ross, *Industrial market structure and economic performance*. Boston: Houghton Mifflin, 1990.

[48] S. A. Slaughter, D. E. Harter, and M. S. Krishnan, "Evaluating the Cost of Software Quality," *Communications of the ACM,* vol. 41, pp. 67-73, 1998.