MIS Quarterly

SPECIAL ISSUE

WORK DISPERSION, PROCESS-BASED LEARNING, AND OFFSHORE SOFTWARE DEVELOPMENT PERFORMANCE[1]

By: **Narayan Ramasubbu**
School of Information Systems
Singapore Management University
80 Stamford Road
Singapore 178902
SINGAPORE
nramasub@smu.edu.sg

Sunil Mithas
Decision, Operations and Information Technologies
Robert H. Smith School of Business
University of Maryland
College Park, MD 20742
U.S.A.
smithas@rhsmith.umd.edu

M. S. Krishnan
Stephen M. Ross School of Business
University of Michigan
701 Tappan Street
Ann Arbor, MI 48109
U.S.A.
mskrish@bus.umich.edu

Chris F. Kemerer
Joseph M. Katz Graduate School of Business
University of Pittsburgh
278A Mervis Hall
Pittsburgh, PA 15260
U.S.A.
ckemerer@katz.pitt.edu

Abstract

In this paper we develop a learning-mediated model of offshore software project productivity and quality to examine whether widely adopted structured software processes are effective in mitigating the negative effects of work dispersion in offshore software development. We explicate how the key process areas of the capability maturity model (CMM) can be utilized as a platform to launch learning routines in offshore software development and thereby explain why some offshore software development process improvement initiatives are more effective than others. We validate our learning-mediated model of offshore software project performance by utilizing data collected from 42 offshore software projects of a large firm that operates at the CMM level-5 process maturity. Our results indicate that investments in structured processes mitigate the negative effects of work dispersion in offshore software development. We also find that the effect of software process improvement initiatives is mediated through investments in process-based learning activities. These results imply that investments in structured processes and the corresponding process-based learning activities can be an economically viable way to counter the challenges of work dispersion and improve offshore project performance. We discuss the implication of these results for the adoption of normative process models by offshore software firms.

Keywords: Offshore software development, capability maturity model, software project performance, software engineering, software productivity, software quality, distributed teams, global service disaggregation

Introduction

Spurred by the growth of information technology outsourcing, offshore software development firms are now playing a key

[1]This paper was recommended for acceptance by Associate Guest Editor Varun Grover.

role in the strategic information technology projects of their Fortune 1000 clients, and are increasingly taking part in joint product development ventures (Apte and Mason 1995; Arora 2005; Beulen et al. 2005; Carmel 1999; Mithas and Whitaker 2007). Apart from the well-documented cost arbitrage, another important factor that has been attributed to the increase in the stature of offshore software development is the dramatic improvement in the process quality and project management capabilities of offshore software service providers (Carmel and Agarwal 2002; Ethiraj et al. 2005). A significant fraction of offshore software vendors follow the tenets of normative software process models to develop applications rapidly and in a cost effective manner without compromising on quality. Indeed, the largest pool of world-wide software firms with capability maturity model (CMM)[2] level-5 assessment in any single country are currently in India (SEIR 2007).

Work dispersion of the type that is common in offshore software development projects is vulnerable to communication, coordination, and administration problems that affect project performance (Herbsleb and Mockus 2003; Jarvenpaa and Leidner 1999; Maznevski and Chudoba 2000; Olson and Olson 2000; Sarker and Sahay 2002). Although software process improvement initiatives based on normative process maturity models, such as the CMM, have been widely deployed by offshore software firms, the efficacy of such initiatives to counter the challenges of work dispersion in offshore software development remains an open empirical question. Further, while the adoption rate of structured and high maturity processes is rising among offshore software firms, prior research reports significant variance in performance improvements resulting from software process initiatives (Kitson and Masters 1993; SEMA 2002), pointing to a need to understand why some offshore software process initiatives are more effective than others.

In this paper we develop and test models of offshore software project productivity and quality to determine the extent to which investments in structured software processes can mitigate the impact of work dispersion on offshore software project performance. Following Mukherjee and Wassenhove (1998), and Mukherjee, Lapre, and Wassenhove (1997) who use the notion of learning-mediated performance enhancement to explain the effectiveness of total quality management programs in a manufacturing context, we posit a learning-mediated effect of the CMM processes on offshore software development productivity and quality. We argue that the individual, structured work routines prescribed by the CMM can be utilized as a learning platform paving the way for *knowledge driven performance improvement*. We establish learning as one of the fundamental mechanisms through which investments in software process improvements influence final offshore software project performance. We use data collected from 42 offshore software development projects of a large, high maturity (CMM level-5) software organization to test empirical support for this learning-mediated model.

The rest of the paper is organized as follows. In the next section we draw from the organization learning and software engineering literatures, and present the theoretical background for our research. Based on this we specify our hypotheses relating offshore software project productivity and quality with work dispersion and process-based learning. We then discuss our research site and data collection, present our empirical estimation procedures, and analyze our results. Finally, we discuss our findings, implications for research and practice, and limitations of the study, and present concluding remarks.

Background and Theory

Prior Literature

The research framework for this study treats software development as an economic production process and models the software performance indicators, viz. productivity and quality, as a function of personnel and software methodology related factors. A number of research studies in the software engineering literature have used this framework to analyze drivers of project performance in the colocated development scenario. For example, Banker, Datar, and Kemerer (1991) modeled productivity of software maintenance projects as an economic production process impacted by a variety of factors including personnel capability, work hours, process methodology, and hardware response time. Banker and Slaughter (2000) proposed a model of software enhancement effort that depicted maintenance activity as a production process impacted by a variety of complexity and team experience-related factors. Other studies used a similar approach to model productivity, quality, and process improvement of colocated teams involved in software development (Harter et al. 2000; Krishnan and Kellner 1999; Krishnan et al. 2000). This study extends this modeling framework by investigating the impact of work dispersion, CMM process investments, and learning

[2]Capability maturity model (CMM) is a widely adopted software process maturity framework developed by the Software Engineering Institute at Carnegie Mellon University. The CMM specifies 18 key process areas classified into 5 evolutionary maturity levels. A basic premise of the CMM is that a higher maturity level leads to better software project performance. Level-5 is the highest level and thus represents firms operating at the level of current best practice (Paulk et al. 1993a).

routines on offshore software development project performance measured in terms of productivity and quality.

Related prior studies on software processes have explored the linkage between process investments and project performance (Harkness et al. 1996; Harter et al. 2000; Herbsleb et al. 1997; Krishnan and Kellner 1999). Collectively these studies establish a positive linkage between investments in software process investments and project performance in the *colocated* development setting where work dispersion is not a major factor in impacting eventual performance. Also, prior empirical studies have focused only on establishing whether there is a statistically significant *direct effect* of process investments on project performance, rather than on explicating and testing the mechanisms through which process investments affect project performance.

In contrast to these prior studies, our research focuses on a distributed development context and we test the linkages between process investments and project performance in an environment subjected to the effects of work dispersion. Drawing on the organizational learning literature, we explicate how the CMM facilitates process-based learning (i.e., the launch of specific learning activities along with the software development work-related activities). Building on this theoretical framework of process-based learning, we hypothesize and test a learning-mediated model of offshore software development project performance.

The Capability Maturity Model and Process-Based Learning

The CMM prescribes a set of key processes and practices that form the basic platform upon which individual tasks in a software project are operationalized. The key process areas (KPA) prescribed by the CMM are grouped under five evolutionary levels: (1) initial, (2) repeatable, (3) defined, (4) managed, and (5) optimizing. Each KPA is defined to fulfill a particular goal and contains prescriptions for a set of activities to be accomplished in the course of the software project. Also, these key process prescriptions pertain to different functional groups, such as executive organization management, project management and engineering teams (for detailed descriptions of all the KPAs in the CMM framework, see CMU-SEI 1995; Paulk et al. 1993b).

Drawing from the organizational learning literature, we elaborate how the CMM processes facilitate learning routines. Specifically, we build on the theme that routines or processes, which are the temporal structures in organizations through which work is accomplished, can be an important source of organizational learning (Levitt and March 1988). Repeatable routines or processes provide the structure for work-based learning, wherein the real-time experience acquired in the midst of performed actions can be reflected and assimilated, contributing to knowledge acquisition (Raelin 1997).

We posit that usage of the CMM to manage offshore software development processes is expected to facilitate organizational learning in two fundamental ways. First, organizational learning happens through a continuous cycle of contextualization and institutionalization of CMM processes. Second, the individual KPAs of the CMM induce learning activities and thereby contribute to the fundamental mechanisms of organizational learning. We explain each of these mechanisms in turn.

Contextualization–Institutionalization Cycle and Learning

While the CMM prescribes a predefined structure for software development, it also allows for context-specific interpretations of the prescriptions. The process of adjusting the general prescriptions of the CMM to derive a context-specific description of the processes is called *tailoring* (Ginsberg and Quinn 1994). Tailoring enables CMM software processes to acquire both *ostensive* and *performative* aspects. The ostensive aspect is the schematic or prescriptive form of a routine, whereas the performative aspect consists of the specific actions taken by practitioners of the routines, that is, the routine in practice (Feldman and Pentland 2003). Routines facilitated by the CMM include both the ostensive aspect (the KPA descriptions) and the performative aspect (variations through tailoring).

Routines facilitated by the CMM acquire the performative aspect through cycles of contextualization and institutionalization of the CMM processes. *Contextualization* of organizational processes occurs when processes are tailored and interpreted to specific environments at the project and functional group level. *Institutionalization* occurs when the effects noticed at the project level are generalized and applied at the organizational level. The CMM framework facilitates a continuous cycle of contextualization and institutionalization, as depicted in Figure 1, by formalizing the necessary steps in the cycle. For example, a hierarchical approval structure is formed to formally approve process changes originating from tailoring, and the experiences with the tailored processes are captured during the project closure meetings. Thus, acquiring performative aspects of the processes through the contextualization–institutionalization cycle leads to generalization of context specific findings and increases the chances of the new knowledge to enter into the organizational knowledge network (Raelin 1997).

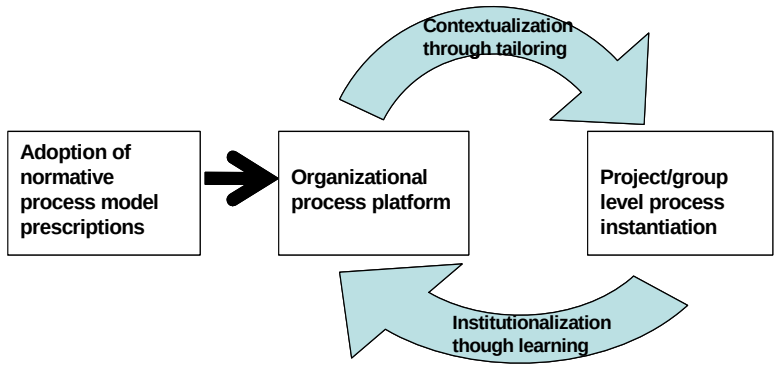

Figure 1. The Cycle of Contextualization–Institutionalization

Mapping CMM KPAs and Learning

Huber (1991), drawing on organizational learning literature, lists four processes as fundamental contributors to organizational learning: (1) knowledge acquisition, (2) information distribution, (3) information interpretation, and (4) organizational memory. *Knowledge acquisition* is the method by which knowledge is obtained. *Information distribution* is the process by which information from different sources is shared across different participants. *Information interpretation* is the process by which a common meaning is attached to the distributed information. *Organizational memory* refers to the mechanisms through which the interpreted information is stored for reference. We next elaborate how the CMM KPAs facilitate each of these fundamental organizational learning processes.

CMM and Knowledge Acquisition and Information Distribution. The CMM facilitates knowledge acquisition by providing a platform of processes that induce both single and double loop learning (Argyris and Schon 1978). Single loop learning focuses on modifying *action strategies*, which are the plans used by participants to meet their goals, to achieve desired consequences within the constraints of *governing variables*, which are the dimensions that managers want to keep under control. On the other hand, double loop learning happens when the governing variables themselves are subject to scrutiny and changes. CMM facilitates single loop learning through the process definition and quantitative process management KPAs. These KPAs help offshore software managers to design their individual action strategies (i.e., the work routines) based on existing governing variables (i.e., the project goals and organizational benchmarks). Double loop learning happens in CMM-enabled offshore software development when the governing variables themselves are subjected to review and reflection through the process change management and organizational process focus KPAs. Continuous improvement of the governing variables, and thereby double loop learning, are further facilitated by the IDEAL model of CMM process improvement, which consists of a cycle of initiating, diagnosing, establishing, acting, and leveraging stages (Ginsberg and Quinn 1994).

The CMM framework also facilitates activities for social interaction that, in turn, are a fundamental dimension of knowledge creation and information dissemination in organizations (Nonaka 1994). Activities for social interactions help organizational units to systematically recognize knowledge created at the individual level and institutionalize them into organizational knowledge. Specific KPAs in the CMM model that govern social interaction activities are peer reviews and intergroup coordination. Further, the organization process focus KPA lays out activities to disseminate the best practices found locally in a functional unit to the entire organization by institutionalizing the discovered set of best practices. Thus the CMM framework contributes to knowledge acquisition and information dissemination through a variety of KPAs and process implementation guidelines.

CMM and Information Interpretation. CMM facilitates interpretation of new information both at the project group level and the organizational level. The tailoring process ensures that there is a common understanding of the organizational level processes and key practices before context-specific interpretations are defined at the group level. Again, when the context-specific group level information is institu-

tionalized, key practices governed by the organizational process focus KPA ensure that there is a uniform understanding of the meaning of this information at the organizational knowledge network level. Moreover, one of the key CMM prescriptions is to conduct peer discussions for important decisions and to move away from individualistic, *ad hoc* decision making (Paulk 1995). Peer discussions minimize heterogeneity of the interpretation of information and aid in consensus building.

CMM and Organizational Memory. The CMM facilitates mechanisms for organizational memory through organizational process assets such as the software process database, document libraries and the repository of policy documents (Jalote 1999, p. 103). The software process database contains historical data from all of the organizational units and accumulates new event data from the various activities conducted according to the process prescriptions. These stored data can be used for management reporting, statistical analysis, and to derive organization-wide benchmark governing variables. Organizational memory mechanisms, such as the central software engineering process database facilitated by CMM, are central to knowledge management processes and contribute to organizational learning (Alavi and Leidner 2001).[3]

The foregoing discussion clarifies how the platform of processes and practices prescribed by the CMM framework facilitates the fundamental processes that contribute to organizational learning. Table 1 presents a summary of the mapping between the fundamental processes that contribute to organizational learning and the facilitating mechanisms found in the CMM process model.

Hypotheses

Effect of Work Dispersion on Offshore Software Project Productivity and Quality

Execution of offshore software projects requires a significant amount of work dispersion across different development locations. Although labor costs are lower at offshore centers (a major motivation for the offshoring decision in the first place), there are certain activities that are best performed at the customer's premises where the end users and hardware are located. For example, researchers have argued the need for customer contact and physical presence at customer sites in distributed environments for improving customer service (Apte and Mason 1995; Mithas and Whitaker 2007). The presence of onsite resources also aids in the gathering of end-user requirements and provides for quicker feedback on prototypes. Further, implementation of large-scale software is a complex task and often involves several configuration settings and changes that require physical presence at the customer site where hardware artifacts are located. Moreover, resources are often required to be present at the customer's site to handle urgent production failures. Thus, even in a primarily offshore software model of development, resources need to be present both offshore and onsite.

Although prior research studies have not specifically investigated the effect of work dispersion on software productivity and quality in the offshore development context, there is some general evidence for reduced performance among distributed software teams. For example, Mockus and Weiss (2001) report that distributing product maintenance work across development centers significantly affects the cycle time of a project. Likewise, Herbsleb and Mockus (2003) report that cross-site software development takes a much longer amount of time and requires more people for work of similar size and complexity as compared to same-site work. Sarker and Sahay (2002) identified various collaboration inhibitors in distributed system development teams including problems arising from geographical separation, different cultural contexts, and different systems development philosophies, approaches, and infrastructure. In a recent study at Intel Corporation examining distributed work, Lu et al. (2006) found that employees faced difficulties in managing coordination overheads that eventually lead to reductions in performance. Previous research has attributed these negative effects of work dispersion to two mechanisms: (1) difficulty in establishing mutual knowledge to accomplish interdependent tasks (Blackburn et al. 1996; Cramton 2001; Herbsleb and Grinter 1999; Victor 1990), and (2) coordination and administration challenges in managing distributed resources (Andres 2002; Barkhi et al. 2006; Olson and Olson 2000).

We expect that the aforementioned effects will be prevalent in offshore software teams as well. Interdependence in performing tasks across onsite and offshore can be expected to lead to task uncertainty, fluctuations in resource availability, and goal conflicts which are all negatively associated with group performance. In addition, successful completion of interdependent tasks across developers located at onsite and offshore locations requires common understanding or mutual knowledge about customer requirements, task-specific details, and work schedules. Similar to other virtual teams, physical dispersion among offshore software development teams is likely to negatively affect the means by which mutual knowl-

[3]Part of the data for this research study was drawn from such a central process database at our research site, which further indicates that the CMM facilitates organizational memory mechanisms.

Table 1. Summary of Organizational Learning Processes and the CMM

Organization Learning Process	Facilitating KPAs in the CMM Process Model
Knowledge Acquisition	Training programs IDEAL stage activities Change management Peer reviews
Information Distribution	Organizational process focus Peer reviews Intergroup coordination
Information Interpretation	Tailoring Organizational process focus Peer Reviews
Organizational Memory	Process database Configuration management Lifecycle document library Policy repository
Social Interaction	Peer reviews Intergroup coordination

edge is established (Cramton 2001). This may be because of unevenly distributed information, failure to communicate and retain contextual information, differences in the salience of information, and relative differences in the speed of access to information. Thus, the difficulty in achieving mutual knowledge to execute interdependent tasks may lead to task uncertainty and rework, and hence negatively affect the productivity and quality of offshore software projects.

Further, in offshore software development teams, a large number of the coordination activities between teams located at the customer's premises and those located at the offshore development center occur primarily through electronic media. Researchers have argued that even advanced communication technologies may not be able to replicate the characteristics of colocated interactions (Andres 2002; Herbsleb and Grinter 1999; Mithas and Whitaker 2007; Olson and Olson 2000). Thus, difficulties in coordinating interdependent tasks through electronic communication lead to increased occurrences of lapses in specification, design, coding, and testing, with the consequent rework leading to increased development effort (poor productivity) and a higher number of errors (poor quality).

In summary, the presence of geographically distributed resources in a project can give rise to difficulties in establishing mutual knowledge, coordinating interdependent tasks, and enforcing consistent working patterns which, in turn, negatively affect project productivity and quality. Therefore, we hypothesize that

H1. Higher dispersion in offshore software development work is associated with lowered project productivity and quality.

Learning-Mediated Impact of CMM Process on Project Performance

As theorized in the previous section, the process prescriptions of the CMM can be a robust platform on which different learning activities can be launched. The investments in learning activities are made possible because of the work structure facilitated by the CMM KPAs. For example, single and double loop learning activities will not take place without the implementation of the associated CMM KPAs (such as peer reviews and organizational process focus) nor in the absence of the contextualization–institutionalization cycle facilitated by the CMM process framework. It is also well documented that investments in learning activities help the evolution of dynamic capabilities (Zollo and Winter 2002). Dynamic capabilities, as opposed to ordinary capabilities that merely assist in the conduct of day-to-day tasks, help managers to design project routines which aid context-specific resource reconfigurations. In the offshore software development context, dynamic capabilities could assist vendors in designing appropriate resource reconfigurations to address coordination and administration challenges due to the emergence of new technological paradigms, development methodologies, and customer requirements. Previous research suggests that the capability to design and execute timely

resource reconfigurations eventually leads to improved performance (Eisenhardt and Martin 2000; Winter 2003).

Based on this discussion, we expect that investments in learning activities have a positive influence on productivity and quality of offshore software projects. At the same time, investments in these learning activities are triggered through the corresponding CMM process activities. Thus, we posit process-based learning investments as a key mediating mechanism through which investments in CMM processes affect offshore project productivity and quality.

> *H2. The effect of software process investments on offshore software project productivity and quality is mediated through learning investments.*

Conceptual Learning and Operational Learning

Researchers examining the deployment of total quality management (TQM) in manufacturing plants have reported that the relative emphasis of the deployment scheme on two different learning activities, conceptual learning and operational learning, play an influential role in the effectiveness of the TQM program (Mukherjee et al. 1998; Mukherjee and Wassenhove 1997).

Operational learning deals with the acquisition of "know-how" (i.e., developing skills to deal with tasks at hand). On the other hand, conceptual learning deals with the acquisition of "know-why" (i.e., developing an understanding of cause-and-effect relationships). Researchers have argued that know-how is primarily associated with learning-by-doing, and know-why typically results from learning-before-doing, either through experimentation before doing a task or from reflection of the experiences gained through a completed task (Dutton and Thomas 1985; Garud 1997). Research on process development in the pharmaceutical industry has shown that activities associated with conceptual learning lead to better performance in environments characterized by high process maturity and deep theoretical and practical knowledge of process technology. In contrast, learning-by-doing is more effective in environments in which processes were less structured (Pisano 1994). In the research setting of this study, all of the projects were executed in a high process maturity environment (CMM level 5). High process maturity environments are characterized by a highly structured engineering approach in their work practices. The following quote from a project group working in a high maturity environment illustrates the extent of structured activities encountered in their day-to-day work:

> The most important things the shuttle group does is carefully planning the software in advance, writing no code until the design is complete, making no changes without supporting blueprints, and keeping a completely accurate record of the code (Fishman 1997).

With the structured processes in place high process maturity work environments provide a conducive environment for enacting and benefiting from learning-before-doing activities that contributes to the acquisition of know-why (i.e., conceptual learning). Thus, we expect that investments aimed at acquiring conceptual learning will have a relatively higher impact on project productivity and quality than will operational learning investments. Therefore, we hypothesize

> *H3. In high process maturity environments, investments in conceptual learning activities have higher impact on offshore software project productivity and quality, than investments in operational learning.*

Figure 2 shows the conceptual framework for this research.

Research Site and Data Collection

We collected detailed distributed software development project data from a leading offshore software service company that employs over 19,000 people in 17 countries worldwide and had annual revenue of more than one billion dollars at the time of our data collection. This firm provides an ideal research setting to study the effects of process and learning investments because the firm employs a high maturity development process. Our research site was assessed to operate at the highest maturity level (5) of the CMM.[4] These high maturity operations at our research site allow us to test industry best practice, as well as helping to ensure that the data collected to test our research hypotheses are more likely to be reliable.

All of the projects we studied involved outsourced software development of commercial business applications at two software development centers of the offshore vendor, one each in the United States and in India. All of the clients involved in

[4]This study is based on the version 1.1 of the CMM framework, which our research site had adopted. A newer version of the framework, CMMi®, has since been released. However, there is no reason to believe that the results with the newer framework would be qualitatively different than those reported here.

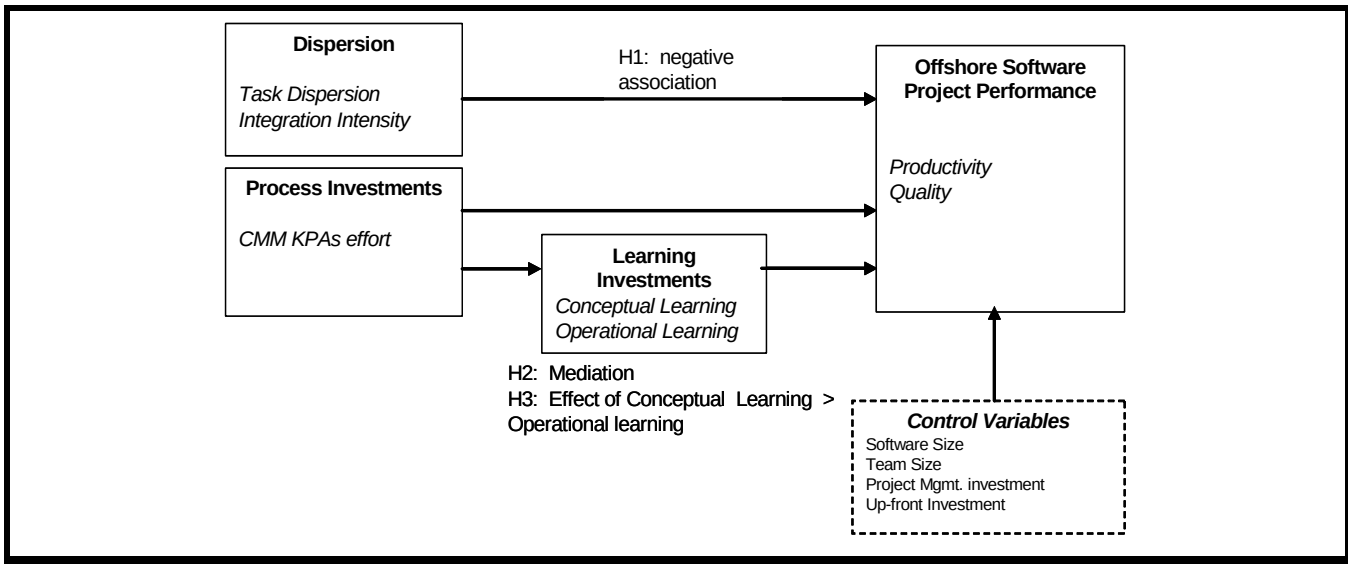

Figure 2. Learning-Mediated Model of Project Performance

the projects were located in the United States. The projects were governed by fixed-price contracts between the clients and the offshore vendor. Software development involved high-level programming languages and relational databases.

Our data collection involved gathering information on 42 projects completed in a recent 2-year time period. These projects were randomly selected for an audit of CMM level-5 compliance by an independent agency. We utilized the opportunity provided by the assessment audit to gather data for our research. We retrieved project data from a central process database that is regularly audited and maintained by the quality management division of the organization. Various steps involved in recording of data in the central repository at our research site are presented in Figure 3.

To get further clarification on the data retrieved from the central process database, we also engaged with the project coordinators of the teams. Discussions with the project coordinators were particularly useful in understanding the project-level process tailoring data. We also conducted discussion sessions with two senior business development managers responsible for the projects to understand the firm's policies on allocating work between the offshore and onsite development centers. We further interviewed 10 randomly selected project team members from a list provided by the firm's human resource department to get developers' opinions on the structured processes adopted by the firm. These interviews were helpful in understanding how the developers populated the central process database with data related to their tasks.

Individual project managers at our research site were bound by the company-wide goal of operating according to the prescriptions of the CMM process maturity framework. However, we found extensive process tailoring done at the project level. Formal tailoring of the general prescriptions of the organizational-level CMM KPAs allowed individual software project managers to adjust their practices for differences in the specific development environments faced in their projects. At a project level it was the responsibility of project managers to decide not only the set of KPAs that will be used in the project, but also the level of effort spent in the individual activities prescribed by the KPAs. Thus we observed significant variation in the pattern of usage of processes and the effort expenditure on the CMM processes across the 42 projects.[5] We use these variations to study the implementation of specific learning routines in the projects and their impact on project productivity and quality.

To assess if there were significant differences in human resource practices among the different units, we analyzed employee performance appraisal templates, code of conduct guidelines, and incentive structures from different units. We found that the firm had a uniform human resource policy throughout the world, and that there were no significant differences across project units. Also, our research site was

[5]These variations of the processes adopted at the project level were formally approved by the software engineering process group of the firm. These variances were not due to the deviation from the prescribed processes by individual work practices of project personnel. We thank an anonymous reviewer for the opportunity to clarify this point.

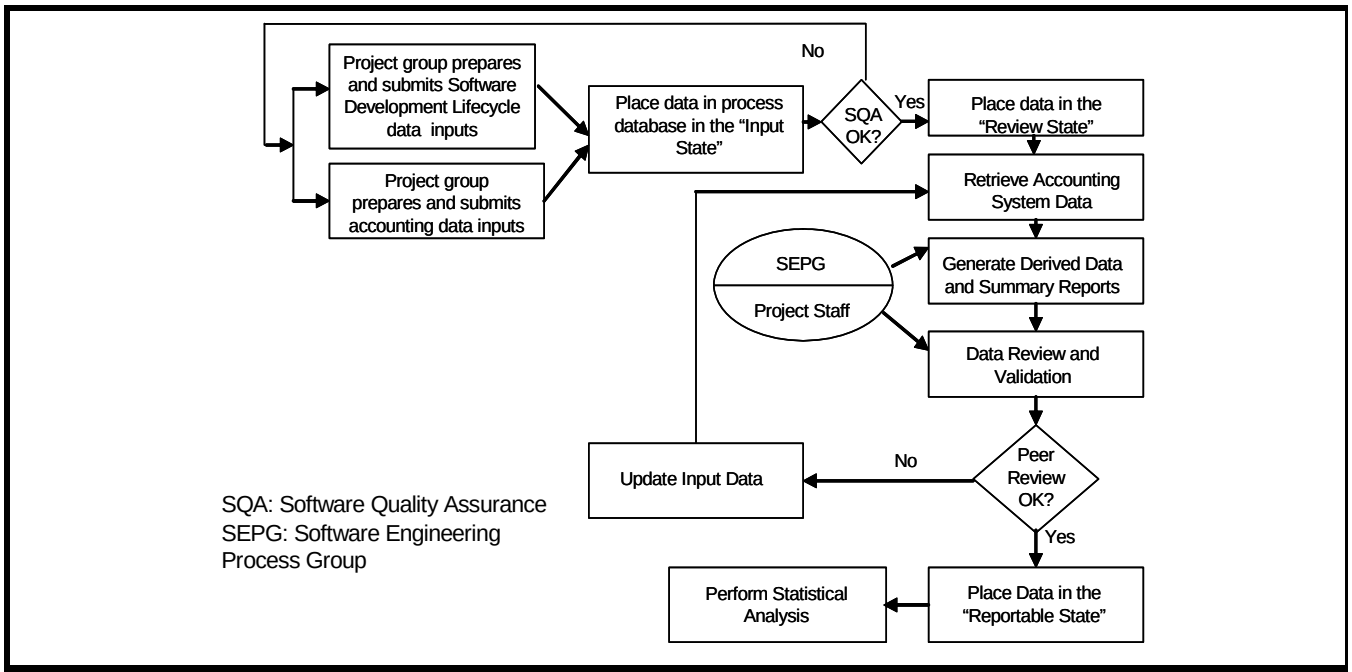

Figure 3. Process Flow to Store Information in Process Database at Our Research Site

recently assessed for maturity in human resources management practices and was assessed as complying with the prescriptions of the level-5 of the people capability maturity model (Curtis et al. 2001). This further shows the consistency and homogeneity in the human resource practices followed at different development centers of the firm.

Variable Definition

Software project performance: We consider two measures of project performance: productivity and quality. We define productivity as the ratio of the software size (output) delivered in the project to the total effort (input) invested in the project. Total effort includes effort incurred in all stages of project life cycle until the project is completed. Our quality measure is defined, in a manner similar to other studies, as the inverse error rate or the ratio of software code size to the number of unique problems reported by customers during the acceptance tests and during the warranty period before the project is finally accepted (Harter et al. 2000; Harter and Slaughter 2003).

Process Investments: A software process is defined as the set of activities, methods, practices, and transformations that people use to develop and maintain software and its associated products (e.g., project plans, design documents, code, test cases, and user manuals). We measure process investments as the ratio of effort spent in a software project for defining, tailoring, assessing, and implementing the project specific process model that is derived from the CMM framework to the total effort spent on the project. Process investment effort in the projects was reported to the central software engineering process group of the firm by different functional groups (e.g., management, the quality assurance team, and the engineering team) involved in the project. This effort was reported separately from the other software lifecycle activities because it was not billable to the customer account.

Work Dispersion: Based on a review of the literature and an understanding of the outsourced software development centers used in our empirical study, we developed two work dispersion measures: *task dispersion* and *integration intensity*. The projects in our sample were executed at both of the development centers involved in the software development cycle. Division of labor between the development centers was not based on the individual development life cycle stages, but was governed by the individual functionality built in the business application. Even when categorized by the individual functionality of the application, we noticed significant joint ownership of functionality across all 42 projects in the sample. Thus, we needed a task dispersion measure to capture the overall work division between the development centers *and* an integration intensity measure to assess the inten-

sity of effort necessary to combine the individual work units into a whole delivery package as seen by the customer.

Task dispersion measures the extent to which the activities of the project were dispersed between the two development centers. As there is no prior widely accepted software engineering measure for this in the literature, we measure task dispersion using a variable similar to the Herfindahl-Hirschman index (Scherer and Ross 1990). Since there are only two development centers in our data set, the work dispersion measure is defined as

> Task dispersion = [100^2 – (% effort at first development center)^2 – (% effort at second development center)^2]

A value of zero for our task dispersion measure indicates that the software project is completely colocated, and an increasing value represents increasing levels of task dispersion. For example, when 90 percent of project activities are located at a development center and the remaining 10 percent are located at a remote center, the value of our task dispersion variable is given by 1800 (10000 – 100 – 8100 = 1800). Similarly, for an 80-20 scenario, the task dispersion variable value is 3200 (10000 – 400 – 6400 = 3200). The maximum value of task dispersion in the two development center scenario is 5000 when the work allocation percentage between the development centers is 50-50.

Integration intensity is another dimension of work dispersion and measures the extent to which the distributed teams spent effort on integrating the individual work done by them to form a whole artifact that is deliverable to the customer. It is measured as the ratio of effort spent on integration tasks to the total effort spent on the project. Higher scores on the integration intensity variable indicate that effort spent on integration activities was larger which, in turn, indicates that project was dispersed to a greater extent between the participating development centers.

Appendix A illustrates how we computed the work dispersion variables for project #32 (of 42).

Learning Investments: This variable is measured as the ratio of cumulative effort spent on learning activities in a software project to the total effort spent on the project. The central process database at our research site consisted of records of the time spent on different activities of the project at the key practices level defined in the project process model. Each activity recorded in the database had a corresponding link to the document repository that consisted of an archive of the document artifacts generated from the activities. We retrieved the descriptions of the activities performed from these documents. Three independent analysts (coders) from the software engineering process group were asked to identify if the activities performed and recorded in the database pertained to learning. The following definition of a learning activity was used:

> Activities conducted with the intention of analyzing experienced events, understanding cause-and-effect relationships that explain the experienced events, and acquiring specific skills and know-how to deal with the experienced events (Mukherjee et al. 1998, p. 38).

The coders were further asked to map the identified learning activity to either conceptual learning or operational learning. The following coding rule was used to map the activities to a specific type of learning:

> *Conceptual learning* consists of *assessing* cause-and-effect relationships that govern experienced events, and *designing* an abstract concept—a theory—to explain this experience. Conceptual learning is trying to understand why events occur, i.e., the acquisition of know-why. In contrast, *operational learning* consists of *implementing* changes and *observing* the results of these changes. Operational learning is basically developing a skill of how to deal with experienced events, i.e., the acquisition of know-how (Mukherjee et al. 1998, p. 38).

After the coders' independent analysis of the activities, we analyzed their results. Two coders returned the exact coding for both the identification of learning activities and the mapping to conceptual and operational learning. However, the third coder identified 13 additional key practices performed in the projects as learning related activities. Cohen's Kappa, a statistical measure of inter-rater reliability (Landis and Koch 1977) for the mapping of the learning activities to operational and conceptual learning categories, was 0.9, indicating that there was a high level of agreement among the coders.[6] Once all the learning activities had been identified, we retrieved the effort spent on these activities from the process database and calculated the cumulative effort spent on the learning activities for each of the 42 projects in our data set.

Software Size: We measure software size using both the function points metric, as well as the traditional kilo lines of

[6]The coders met again to resolve any differences. A detailed mapping of the processes invoking conceptual and operational learning is presented in Appendix B.

code (KLOC).[7] While KLOC is a traditional measure of size, the advantage of function point measurement is that it incorporates measures of customer perceived software functionality as well as complexity (IFPUG 1999).

Team Size: Team size is the full-time equivalent headcount of the number of people involved in the project.

Project Management Investment: This is the ratio of the effort spent by the project manager for project management activities to the overall total project effort.

Up-Front Investment: The ratio of effort spent on the initial stages of a project, including customer requirements gathering and sequencing the requirements into releases, negotiating the priority of the requirement releases to the overall project effort. It has been shown in prior research that up-front investments significantly impact software project performance, and therefore controlling for up-front investments is important while analyzing project performance (Krishnan et al. 2000). Further, in the context of offshore software development, the up-front investment variable may also control for other factors, such as relationship establishment between the customer and the offshore service provider.

Table 2 presents summary statistics of the variables. Table 3 reports correlations among the log transformed variables.

Empirical Models and Econometric Issues

To test the individual paths in our learning-mediated model, we use the seemingly unrelated regression (SUR) technique (Zellner 1962). SUR is the appropriate technique for estimating regression equations when the errors from the regression equations are correlated with each other. Since the data for the regression equations for our dependent variables productivity, quality, and learning (when testing mediation) are collected from the same set of projects from a single firm, it is likely that the error terms in these regression equations are correlated with each other. Hence, we use the SUR technique to estimate our regression coefficients.

Before proceeding with the regression, our analysis indicated that the variables in our data set were distributed in a non-normal fashion. We therefore transformed the variables using a logarithmic transformation to reduce the skewness of the variables. Other research studies in the software engineering economics literature also recommend logarithmic transformation of variables while studying project performance to accommodate the effect of economies of scale found in software development (Banker and Kemerer 1989; Harter et al. 2000; Krishnan and Kellner 1999). We tested our model specification using the J-test (Davidson and Mackinnon 1981) and the Cox-Pesaran-Deaton test (Cox 1961; Pesaran and Deaton 1978). These model specification tests rejected both the linear additive models and the additive models with interaction terms in favor of our multiplicative log-log empirical specification.[8]

Tables 4 and 5 present the results from our regression models. We checked for multicollinearity effects in our models using variance inflation analysis. As the largest variance inflation factor (VIF) in our analysis was 1.93 and the mean VIF was 1.52, we concluded that no significant undesirable multicollinearity was present. We checked for outliers in our dataset by visually examining the residual plots from our regression and also by using Cook distance statistic (Cook and Weisberg 1999). They did not indicate that the results were unduly impacted by outliers. Finally, the Breush-Pagan test (Breusch and Pagan 1980) to verify the independence of error terms in the estimation equations indicated significant correlation between the error terms of the empirical models at the 5 percent level, supporting our use of the SUR regression technique. All of the empirical models are statistically significant at the 1 percent level and we observe reasonable adjusted R-squared values in our regression results, indicating that the models have good explanatory power.

Results

Hypothesis 1 predicted a negative association between work dispersion and offshore project productivity and quality and we find support for this hypothesis in our results. We find that both measures of work dispersion used in this study, task dispersion and integration intensity, have a negative impact on both offshore project productivity and quality (refer to coefficients β_1 and β_2 in columns 1 and 2, Table 4). Given that the regression coefficients in the log-log empirical model denote elasticity, a 1 percent increase in task dispersion is associated with about a 0.7 percent decrease in productivity and about a 1.5 percent decrease in quality. This result empi-

[7]We thank an anonymous reviewer for this suggestion.

[8]Because of its multiplicative nature, log-log specification accounts for moderation effects among independent variables. Hence, we do not include explicit interaction terms in our regression models and interpret the moderation effects through appropriate plots as we describe later.

Table 2. Summary Statistics

Variable	Mean	Std. Dev.	Min	Max
Productivity	0.212	2.076	0.017	0.636
Quality	1.117	1.426	0.036	5.486
Task Dispersion	4142.099	1.232	1781.799	4994.302
Integration Intensity	0.231	2.771	0.025	2.303
Process Investments	5.537	2.325	0.399	29.023
Learning Investments	4.477	1.795	0.782	20
Conceptual Learning	1	2.376	0.07	4.035
Operational Learning	3.289	1.823	0.712	15.965
Software Size (FP)	1327.608	2.891	33	18247
Software Size (KLOC)	139.466	177.256	2.409	1076.573
Team Size	9.627	1.757	2	30
Project Management Investment	7.665	1.608	3.041	17.409
Up-front Investment	0.526	16.656	0.003	13.365

Table 3. Correlation among Variables (Log Transformed Variables)

		1	2	3	4	5	6	7	8	9	10	11	12	13
Productivity	1	1.000												
Quality	2	0.018	1.000											
Task Dispersion	3	-0.305	-0.207	1.000										
Integration Intensity	4	-0.492	-0.198	0.059	1.000									
Process Investments	5	0.079	0.218	0.147	-0.331	1.000								
Learning Investments	6	0.809	0.109	-0.094	-0.439	0.156	1.000							
Conceptual Learning	7	0.521	0.161	-0.024	-0.279	0.099	0.756	1.000						
Operational Learning	8	0.784	0.051	-0.120	-0.428	0.150	0.958	0.557	1.000					
Software Size (FP)	9	0.664	-0.366	-0.039	-0.147	-0.040	0.500	0.357	0.485	1.000				
Team Size	10	0.057	-0.232	0.136	0.261	0.084	0.023	0.006	0.043	0.624	1.000			
Project Management Investment	11	0.148	-0.019	-0.204	-0.043	-0.042	-0.043	-0.161	0.021	0.122	-0.051	1.000		
Up-front Investment	12	-0.027	0.210	0.046	-0.101	0.056	0.223	0.427	0.106	-0.189	-0.132	-0.195	1.000	
Software Size (KLOC)	13	0.651	-0.375	-0.039	-0.119	-0.053	0.491	0.338	0.481	0.996	0.631	0.124	-0.203	1.00

Table 4. Regression Results

Column →		1	2	3	4	5	6	7	8	9
Variables		Productivity	Quality	Process-Based Learning	Productivity	Quality	Productivity	Quality	Productivity	Quality
Task Dispersion	β_1	-0.732** (0.005)	-1.525** (0.035)	- 0.158 (0.298)	-0.777*** (0.003)	-1.760** (0.025)	-0.663*** (0.001)	-1.423** (0.036)	-0.678*** (0.001)	-1.641** (0.015)
Integration Intensity	β_2	-0.173*** (0.004)	-0.384** (0.021)	-0.079 (0.143)	-0.129** (0.036)	-0.154 (0.215)	-0.098** (0.024)	- 0.260* (0.079)	-0.087* (0.055)	- 0.094 (0.311)
Process Investments	β_3	NA	NA	0.123* (0.065)	0.105* (0.084)	0.551*** (0.005)	NA	NA	0.03 (0.304)	0.461** (0.014)
Learning Investments	β_4	NA	NA	NA	NA	NA	0.630*** (0.000)	0.939** (0.001)	0.617*** (0.000)	0.744** (0.028)
Software Size (KLOC)†	β_5	0.606*** (0.000)	-0.682*** (0.002)	0.462*** (0.000)	0.639*** (0.000)	-0.507*** (0.013)	0.342*** (0.000)	-1.075*** (0.01)	0.357*** (0.000)	-0.847*** (0.002)
Team Size	β_6	- 0.508*** (0.001)	0.570 (0.1)	-0.461*** (0.003)	- 0.579*** (0.000)	0.198 (0.325)	- 0.272** (0.011)	0.920** (0.019)	-0.298** (0.010)	-0.536 (0.120)
Proj. Mgmt. Investments	β_7	-0.029 (0.411)	0.061 (0.436)	-0.151 (0.126)	-0.034 (0.393)	0.033 (0.462)	0.062 (0.259)	0.197 (0.291)	0.058 (0.270)	0.150 (0.333)
Up-front Investment	β_8	0.018 (0.202)	0.053 (0.199)	0.060*** (0.003)	0.018 (0.192)	0.056 (0.170)	-0.018 (0.139)	-0.002 (0.485)	-0.018 (0.150)	0.011 (0.427)
Intercept	β_0	1.159 (0.320)	13.166** (0.033)	0.623 (0.236)	1.351 (0.289)	14.643*** (0.014)	0.906 (0.310)	11.740** (0.042)	0.967 (0.298)	13.269*** (0.02)
Chi-squared		119.55*** (0.000)	16.82*** (0.010)	54.30*** (0.000)	126.79*** (0.000)	26.17*** (0.005)	255.15*** (0.000)	24.40*** (0.001)	257.01*** (0.000)	32.02 (0.000)
R-squared		0.74	0.285	0.465	0.70	0.38	0.80	0.37	0.82	0.41

Note: p-values in parentheses; *significant at 10%; **significant at 5%; ***significant at 1%

†Results are similar with usage of either function points or KLOC as software size measure.

Table 5. Effects of Conceptual Learning and Operational Learning

Column →		1	2	3	4
Variables		Productivity	Quality	Productivity	Quality
Task Dispersion	β_1	-0.644*** (0.002)	-1.463** (0.032)	-0.666*** (0.002)	-1.684** (0.013)
Integration Intensity	β_2	-0.102** (0.026)	- 0.307** (0.047)	-0.084* (0.073)	- 0.133 (0.241)
Process Investments	β_3	NA	NA	0.046 (0.227)	0.468** (0.012)
Conceptual Learning Investments	β_4	0.056 (0.230)	0.532** (0.023)	0.049 (0.261)	0.469** (0.032)
Operational Learning Investments	β_5	0.488*** (0.000)	0.284 (0.223)	0.477*** (0.000)	0.172 (0.315)
Software Size (KLOC)†	β_6	0.393*** (0.000)	-1.071*** (0.000)	0.416*** (0.000)	-0.847*** (0.001)
Team Size	β_7	- 0.333*** (0.004)	0.926** (0.018)	- 0.373*** (0.003)	0.544 (0.113)
Project Management Investment	β_8	0.029 (0.387)	0.250 (0.226)	0.024 (0.408)	0.198 (0.280)
Up-front Investment	β_9	-0.01 (0.363)	-0.035 (0.308)	-0.010 (0.394)	-0.019 (0.385)
Intercept	β_0	0.949 (0.312)	12.916*** (0.027)	1.021 (0.298)	14.328*** (0.013)
Chi-squared		224.88*** (0.000)	25.46*** (0.001)	228.36*** (0.000)	33.67*** (0.000)
R-squared		0.84	0.37	0.84	0.45

†Results are similar with usage of either function points or KLOC as software size measure.

rically establishes the relationship between work dispersion and tangible project performance indicators, and thereby provides quantifiable evidence for the suggestions from prior research that work dispersion in distributed software teams affects project performance (Herbsleb and Mockus 2003; Olson and Olson 2000; Sarker and Sahay 2002; Wong and Burton 2000).

To check the validity of the learning-mediated model proposed by our second hypothesis we followed the procedure detailed by Baron and Kenny (1986). In our research model *productivity* and *quality* are the dependent variables, *process investments* is the primary independent variable and *learning investments* is the mediating variable. The first step in the mediation analysis is to check if the direct effects of the independent variable and the mediating variable are significant on the dependent variable. Referring to columns 4 and 5 in Table 4, we note that the coefficient for process investments (β_3) in the productivity model and quality model is positive and significant. Similarly, referring to columns 6 and 7 in Table 4, we note that the coefficient for learning investments (β_4) is positive and significant in both the productivity and quality models. Specifically, a 1 percent increase in learning investments is associated with about a 0.6 percent increase in productivity and a 0.9 percent increase in quality. These results suggest that the direct effects of both process investments and learning investments on offshore software project productivity and quality are positive and significant. These strong direct effects of process-based learning investments suggest that investments in the structured processes that facilitate learning activities are an economically viable method of countering the work dispersion challenges of offshore software development.

The second step in the mediation analysis is to verify if the primary independent variable can explain the variations in the mediating variable. To assess this we ran a regression with learning investments (mediator) as the dependent variable and the process investments as the independent variables, controlling for the effects of software size, team size, dispersion, project management, and up-front investments (see column 3 of Table 4). The positive and statistically significant regression coefficient of the process investments variable in this model (β_3) shows that investments in process investments are positively associated with investments in learning investments.

The final step to validate the learning-mediated-impact model is to predict the outcome variable with the primary independent variable and the mediator variable in the same equation. If the mediator variable is statistically significant, and if the effect of the primary independent variable on the outcome variable is significantly reduced, then mediation is confirmed. To accomplish this analysis we included both process investments and learning investments, along with the control variables to predict productivity and quality (see columns 8 and 9 of Table 4). We find that the coefficient of process investments (β_3) is insignificant in the productivity model at the 10 percent level, and the significance of the same variable in the quality model has been reduced to the 5 percent level, as compared to the 1 percent significance level in the direct model. At the same time, the coefficient for learning investments (β_4) is positive and significant at the 5 percent level for both the productivity and quality outcomes. This confirms the presence of mediation and empirically validates Hypothesis 2, which posited that the effect of process investments on offshore software project performance is mediated through the investments in learning.

To analyze the impact of different modes of learning as specified in Hypothesis 3, we categorized the overall process-based learning investments into operational learning and conceptual learning activities. We included these variables in our regression models to assess the impact of these individual learning activities on offshore project performance (see Table 5).[9] Our results indicate that, while operational learning significantly improves productivity, conceptual learning activities are significant in improving quality. Our statistical tests indicated that the effects of operational and conceptual learning significantly differ at the 1 percent level. This analysis reveals that even in highly structured environments, investments in different learning activities focusing on know-how (operational learning) or know-why (conceptual learning) have differing impacts on different dimensions of project performance.

Discussion

Offshore software development is becoming pervasive in the software industry and firms are increasingly adopting global delivery business models for sourcing their information technology needs. At the same time, the number of offshore software organizations deploying high process maturity environments to counter the challenges of distributed software teams is also expected to increase rapidly. Consistent with the calls in previous research (Ngwenyama and Nielsen 2003; Ravichandran and Rai 2003), our goal in this study was to relate the prescriptions of normative software process improvement models to fundamental organizational mechanisms grounded in theory. We developed and tested a learning-mediated

[9]The mediation analysis results hold even when we split the aggregated learning investments variable into conceptual learning and operational learning.

model of the impact of software process investments on offshore software project performance. We next discuss our key findings, followed by a discussion of implications, limitations, and suggestions for further research.

Key Findings

Our results indicate that adoption of structured process models have both a direct and a learning-mediated effect in mitigating the negative effect of work dispersion in offshore software development. Figure 4 depicts the mitigating effect of process investments. We plotted the graph by holding all other variables at their mean levels and estimating the effect of dispersion on project productivity at varying levels of dispersion using our regression results. The first (lower) series in the graph was plotted by estimating the effect of dispersion for zero levels of process investments and the second (upper) series was plotted by estimating the effect of dispersion at the mean level of process investments. The graphs show that investments in activities to establish a structured process platform based on the CMM boost productivity and thereby mitigate the effect of dispersion. Further, Figure 4 shows that the negative slope of the curve with no process investments is steeper than that of the curve with process investments at the mean level. This illustrates that the negative effect of dispersion is more pronounced in the absence of process investments. Similar effects are present for the quality dimension of project performance as well. Thus, investments in the structured key process areas specified by the CMM framework tend to mitigate the negative effect of work dispersion on productivity and quality.[10]

We also find that the learning investments play a crucial role in explaining why there is a significant variation in the effectiveness of process investments on offshore software performance. Benefits of investments in establishing a structured process platform can be fully reaped only when corresponding investments are made to utilize the process platform for learning activities that enable creation and assimilation of new knowledge. In addition, our results indicate differential effects of conceptual learning and operational learning on different dimensions of performance. While investments in conceptual learning contributed to improved quality, operational learning investments were associated with improved productivity. This interplay between investments in structured software processes and the different process-based learning activities impacts the overall returns from the software process improvement initiatives. This indicates the importance of considering the individual tradeoffs in allocating resources toward different process-based learning activities. One reason why we observe the differing impacts of conceptual learning and operational learning may be because of the difference in the rates at which the benefits from these learning activities are realized in a distributed setting. Thus, in addition to the nature of process environments (e.g., relatively less or relatively more structured) which have been the primary focus of prior related studies (Hatch and Mowery 1998; von Hippel and Tyre 1995; Pisano 1994) there is a need to further understand how the distributed nature of the work environment impacts the realization of investments in individual learning activities.

Implications for Research and Practice

The research contributions of this study are three-fold. First, we extend the prevalent economic view of the software development perspective to model project performance in the distributed offshore software development context. In doing so, we take the first step in empirically capturing the effects of dispersion in offshore software development and analyzing the linkages between work dispersion and software project performance. Second, our study sheds new light on the existing body of work on the cost of quality in software development. Given our finding that the linkage between process investments and project performance is mediated through a variety of learning investments, assessing and quantifying returns on quality improvement initiatives in software development projects may be seen as more complex than previously imagined. In this scenario, a straightforward linkage between the traditional cost of quality measures and project performance might reveal a biased valuation result. To counter this, we accounted for the learning-mediated effects of process investments and provided an alternate way to quantify the returns on process investments.

Third, this study lays a new theoretical foundation for empirical software engineering studies attempting to explicate the linkages between organizational processes and performance in the software domain. By linking the CMM key process areas to the fundamental organizational learning processes we lay the ground for structured data collection in high process maturity environments, as well as for richer construct development for future software engineering economics studies. We also believe that relating the prescriptions of normative software process improvement models to fundamental organizational mechanisms, such as organizational learning as done in this study, will spur new research focused

[10]It is also important to note that the negative effect of dispersion on final project productivity is not incrementally negative beyond a point. Further, the negative effect of work dispersion does not completely vanish even in a very high process maturity environment.

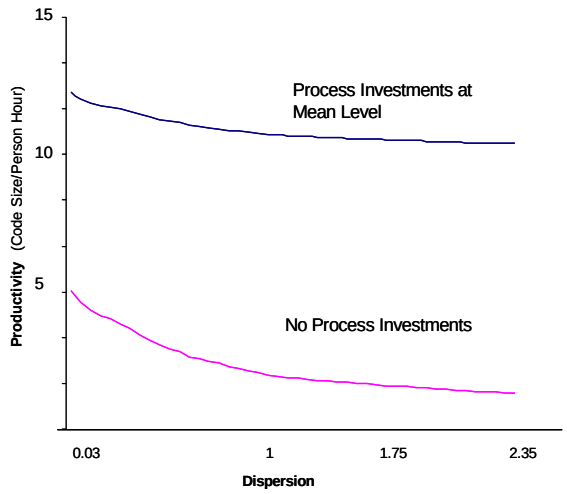

Figure 4. Effect of Dispersion and Investment in Structured Processes

on validating the tenets of normative software process models both theoretically and empirically, especially in the context of offshore software development.

Our findings have three main managerial implications. First, our results indicate that dispersion of tasks in offshore software development negatively impact software development performance, even in a high process maturity environment. Managers should not discount the effect of dispersion while estimating the potential cost-benefits of their offshore software development strategy. Potential savings from distributing work to offshore centers have to be weighed against the possible loss of overall development productivity and software quality that arise because of task dispersion and the challenges in managing interdependent resources.[11]

A second practice-related implication of this study relates to the adoption of normative software process improvement models by offshore software development firms. Implementation of the prescriptions from the normative process model should not be narrowly looked at from the software development lifecycle perspective alone. Designing and implementing complementary routines on top of the process platform infrastructure prescribed by the software process models is necessary to fully realize the benefits of process improvement. A third and final managerial implication of the results from this study is the necessity to design software processes platforms that explicitly address the needs of distributed software development for improved project performance (Ramasubbu et al. 2005).

Limitations and Directions for Further Research

As with all such empirical research, our study contains limitations that create opportunities for future research. First, our projects are set in a unique environment characterized by high maturity processes, a global delivery model of engagement, and with personnel primarily sourced from a single location. These factors may limit the immediate generalizability of our results across potentially broader models of outsourcing engagements and other types of offshore software vendors. Future research studies could explore the efficacy of the learning-mediated model of offshore software performance across different process maturity levels.

Second, all of our data were collected from development projects. Other researchers may wish to test the results observed in our study with other types of software projects, such as maintenance and reengineering. Also, we did not test the efficacy of structured processes under a variety of alternative development methodologies. Researchers attempting to extend this study could verify if the results hold, for example, when software development is performed using agile development methodologies. Third, our study was based on version 1.1 of the CMM framework. A revised version of the framework, CMMi, has since been released and includes a

[11]It is important to acknowledge that the dataset used in this study does not include a project where work dispersion is completely absent (a completely colocated scenario) and hence our results should not be seen as an explicit comparison of completely different sourcing strategies.

broader set of key process areas. While the results reported in this study are likely to be applicable in newer process models such as the CMMi, there is a need to map the KPAs of the new process models to the organizational learning framework used in this study. Fourth, this study employed a cross-sectional analysis, and hence potential longer term spill-over effects of learning activities have not been accounted for. An extension of this study could be to employ longitudinal analysis to capture the potential long-term benefits of learning routines and their impact at the organizational level.

Finally, we investigated work dispersion in offshore software development at an aggregate level. Drawing on similar measures in the economics literature, we have adapted a Herfindahl-like measure and proposed a complementary integration intensity measure for capturing dispersion of software development work for the first time. These measures should be further refined and validated. Also, there is a need for research to probe the implication of different modes of division of labor for the achievement of optimal work dispersion between onsite and offshore teams. For each of the labor division modes, the learning-mediated performance model developed in this study could be used to study the potentially differential impacts of various types of process investments on software project performance.

Our study also paves the way for studying how the structured software development processes and knowledge management processes can be complementary. We theorized how the different aspects of the CMM can be mapped to learning mechanisms. This can be utilized to integrate the knowledge management processes with day-to-day work routines of offshore software teams, and thereby help the implementation of process-oriented knowledge management strategies (Maier and Remus 2003; Rus and Lindvall 2002).

Conclusion

To conclude, this study takes an organizational learning perspective and develops a learning-mediated model of offshore software project productivity and quality. We elaborate how the key process areas of the CMM could be utilized as a platform to launch performance enhancing learning activities. We validate our learning-mediated model of offshore software project performance by utilizing data collected from 42 offshore software projects of a large firm that operates at the CMM level-5 process maturity. Our results indicate that investments in structured processes and process-based learning routines mitigate the negative effects of work dispersion in offshore software development. We also find that the effect of software process improvement initiatives on offshore software project performance is mediated through the investments in learning activities, signifying the important role of process-based learning routines in improving offshore project performance. These results are important for the adoption of normative process models by offshore software firms and to understand how structured processes can be leveraged to execute software projects for competitive success.

Acknowledgments

We thank the senior management and developers at our research site for their support and help in obtaining the necessary data for this research. We acknowledge helpful comments received from Judy Olson, Sandra Slaughter, and three anonymous referees and the associate editor on an earlier version of this paper. Financial support for this study was provided in part by the Office of Research at the Singapore Management University, the Robert H. Smith School of Business at University of Maryland, the Michael R. and Mary Kay Hallman Fellowship at Ross School of Business, and the Sloan Software Industry Center at Carnegie Mellon University (Kemerer).

References

Alavi, M., and Leidner, D. E. 2001. "*Review:* Knowledge Management and Knowledge Management Systems: Conceptual Foundations and Research Issues," *MIS Quarterly* (25:1), pp. 107-136.

Andres, H. P. 2002. "A Comparison of Face-to-Face and Virtual Software Development Teams," *Team Performance Management* (8:1-2), pp. 39-48.

Apte, U. M., and Mason, R. O. 1995. "Global Disaggregation of Information-Intensive Services," *Management Science* (41:7), pp. 1250-1262.

Argyris, C., and Schon, D. 1978. *Organizational Learning: A Theory of Action Perspective*, New York: McGraw-Hill.

Arora, A. 2005. "From Underdogs to Tigers: the Emerging Offshore Software Industries and the U.S. Economy," in *Offshoring White-Collar Work: The Issues and Implications*, L. Brainard and S. M. Collins (eds.), Washington, DC: Brookings.

Banker, R. D., Datar, S. M., and Kemerer, C. F. 1991. "A Model to Evaluate Variables Impacting the Productivity of Software Maintenance Projects," *Management Science* (37:1), pp. 1-18.

Banker, R. D., and Kemerer, C. F. 1989. "Scale Economies in New Software Development," *IEEE Transactions on Software Engineering* (15:10), pp. 1199-1205.

Banker, R. D., and Slaughter, S. A. 2000. "The Moderating Effects of Structure and Volatility and Complexity in Software Environment," *Information Systems Research* (11:3), September, pp. 219-240.

Barkhi, R., Amiri, A., and James, T. L. 2006. "A Study of Communication and Coordination in Collaborative Software Development," *Journal of Global Information Technology Management* (9:1), pp. 44-61.

Baron, R. M., and Kenny, D. A. 1986. "The Moderator-Mediator Variable Distinction in Social Psychological Research: Conceptual, Strategic and Statistical Considerations," *Journal of Personality and Social Psychology* (51), pp. 1173-1182.

Beulen, E., van Fenema, P., and Currie, W. 2005. "From Application Outsourcing to Infrastructure Management: Extending the Offshore Outsourcing Service Portfolio," *European Management Journal* (23:2), pp. 133-144.

Blackburn, J. D., Hoedemaker, G., and Wassenhove, L. N. 1996. "Concurrent Software Engineering: Prospects and Pitfalls," *IEEE Transactions on Engineering Management* (43:2), pp. 179-188.

Breusch, T. S., and Pagan, A. R. 1980. "The LaGrange Multiplier Test and its Application to Model Specifications in Econometrics," *Review of Economic Studies* (47), pp. 239-253.

Carmel, E. 1999. *Global Software Teams: Collaborating Across Borders and Time Zones*, Upper Saddle River, NJ: Prentice Hall.

Carmel, E., and Agarwal, R. 2002. "The Maturation of Offshore Sourcing of Information Technology Work," *MIS Quarterly Executive* (1:2), pp. 65-76.

CMU-SEI. 1995. *The Capability Maturity Model: Guidelines for Improving the Software Process*, Boston: Addison-Wesley Professional.

Cook, D. R., and Weisberg, S. 1999. *Applied Regression Including Computing and Graphics*, Hoboken, NJ: Wiley-Interscience.

Cox, D. R. 1961. "Tests of Separate Families of Hypotheses," in *Fourth Berkeley Symposium on Mathematical Statistics and Probability*, pp. 105-123.

Cramton, C. D. 2001. "The Mutual Knowledge Problem and its Consequences for Dispersed Collaboration," *Organization Science* (12:3), May-June, pp. 346-371.

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

Davidson, R., and Mackinnon, J. G. 1981. "Several Tests for Model Specification in the Presence of Alternative Hypotheses," *Econometrica* (49), pp. 781-793.

Dutton, J. M., and Thomas, A. 1985. "Relating Technological Change and Learning by Doing," *Research on Technological Innovation, Management and Policy* (2), pp. 187-224.

Eisenhardt, K. M., and Martin, J. A. 2000. "Dynamic Capabilities: What Are They?," *Strategic Management Journal* (21:10-11), pp. 1105-1121.

Ethiraj, S., Kale, P., Krishnan, M. S., and Singh, J. 2005. "Where Do Capabilities Come from and How Do They Matter? A Study in the Software Services Industry," *Strategic Management Journal* (26:1), pp. 425-445.

Feldman, M. S., and Pentland, B. T. 2003. "Reconceptualizing Organizational Routines as a Source of Flexibility and Change," *Administrative Science Quarterly* (48:1), pp. 94-118.

Fishman, C. 1997. "They Write the Right Stuff," *Fast Company* (available online at http://www.fastcompany.com/magazine/06/writestuff.html).

Garud, R. 1997. "On the Distinction Between Know-How, Know-What and Know-Why," in *Advances in Strategic Management*, A. Huff and J. Walsh (eds.), Greenwich, CT: JAI Press, pp. 81-101.

Ginsberg, M. P., and Quinn, L. H. 1994. "Process Tailoring and the Software Capability Maturity Model," CMU/SEI-94-TR-024, Software Engineering Institute, Carnegie Mellon University, Pittsburgh.

Harkness, W. L., Kettinger, W. J., and Segars, A. H. 1996. "Sustaining Process Improvement and Innovation in the Information Services Function: Lessons Learned at the Bose Corporation," *MIS Quarterly* (20:3), March, pp. 349-368.

Harter, D. E., Krishnan, M. S., and Slaughter, S. A. 2000. "Effects of Process Maturity on Quality, Cycle Time, and Effort in Software Product Development," *Management Science* (46:4), April, pp. 451-466.

Harter, D. E., and Slaughter, S. A. 2003. "Quality Improvement and Infrastructure Activity Costs in Software Development: A Longitudinal Analysis," *Management Science* (49:6), June, pp. 784-800.

Hatch, N. W., and Mowery, D. C. 1998. "Process Innovation and Learning by Doing in Semiconductor Manufacturing," *Management Science* (44:11), pp. 1461-1477.

Herbsleb, J. D., and Grinter, R. E. 1999. "Splitting the Organization and Integrating the Code: Conway's Law Revisited," in *Proceedings of the 21st International Conference on Software Engineering*, New York: ACM Press, pp. 85-95.

Herbsleb, J. D., and Mockus, A. 2003. "An Empirical Study of Speed and Communication in Globally Distributed Software Development," *IEEE Transactions on Software Engineering* (29:6), June, pp. 481-494.

Herbsleb, J. D., Zubrow, D., Goldenson, D., Hayes, W., and Paulk, M. 1997. "Software Quality and the Capability Maturity Model," *Communications of the ACM* (40:6), June, pp. 30-40.

Huber, G. P. 1991. "Organizational Learning: The Contributing Processes and Literatures," *Organization Science* (2:1), pp. 88-115.

IFPUG. 1999. "Function Point Counting Practices Manual," International Function Point Users Group, Mequon, Wisconsin.

Jalote, P. 1999. *CMM in Practice: Processes for Executing Software Projects at Infosys*, Boston: Addison-Wesley Professional.

Jarvenpaa, S. L., and Leidner, D. E. 1999. "Communication and Trust in Global Virtual Teams," *Organization Science* (10:6), pp. 791-815.

Kitson, D. H., and Masters, S. M. 1993. "An Analysis of SEI Software Process Assessment Results: 1987-1991," in *Proceedings of the 15th International Conference on Software Engineering*, Los Alamitos, CA: IEEE Computer Society Press, pp. 68-77.

Krishnan, M. S., and Kellner, M. I. 1999. "Measuring Process Consistency: Implications for Reducing Software Defects," *IEEE Transactions on Software Engineering* (25:6), pp. 800-815.

Krishnan, M. S., Kriebel, C. H., Kekre, S., and Mukhopadhyay, T. 2000. "An Empirical Analysis of Productivity and Quality in Software Products," *Management Science* (46:6), June, pp. 745-759.

Landis, J. R., and Koch, G. G. 1997. "The Measurement of Observer Agreement for Categorical Data," *Biometrics* (33), pp. 259-274.

Levitt, B., and March, J. G. 1988. "Organizational Learning," *Annual Review of Sociology* (14), pp. 319-340.

Lu, M., Watson-Manheim, M. B., Chudoba, K. M., and Wynn, E. 2006. "Virtuality and Team Performance: Understanding the Impact of Variety of Practices," *Journal of Global Information Technology Management* (9:1), pp. 4-23.

Maier, R., and Remus, U. 2003. "Implementing Process-Oriented Knowledge Management Strategies," *Journal of Knowledge Management* (7:4), pp. 62-74.

Maznevski, M. L., and Chudoba, K. M. 2000. "Bridging Space over Time: Global Virtual Team Dynamic and Effectiveness," *Organization Science* (11:5), September-October, pp. 473-492.

Mithas, S., and Whitaker, J. 2007. "Is the World Flat or Spiky? Information Intensity, Skills and Global Service Disaggregation," *Information Systems Research* (18:3), pp. 237-259.

Mockus, A., and Weiss, D. M. 2001. "Globalization by Chunking: A Quantitative Approach," *IEEE Software* (18:2), March-April, pp. 30-37.

Mukherjee, A. S., Lapre, M. A., and Wassenhove, L. N. 1998. "Knowledge Driven Quality Improvement," *Management Science* (44:11), pp. S35-S49.

Mukherjee, A. S., and Wassenhove, L. N. 1997. "The Impact of Knowledge on Quality," in *The Practice of Quality Management*, P. J. Lederer and U. S. Karmarkar (eds.), Boston: Kluwer Academic Publishers, pp. 147-166.

Ngwenyama, O., and Nielsen, P. A. 2003. "Competing Values in Software Process Improvement: An Assumption Analysis of CMM from an Organizational Culture Perspective," *IEEE Transactions on Engineering Management* (50:1), pp. 100-112.

Nonaka, I. 1994. "A Dynamic Theory of Organizational Knowledge Creation," *Organization Science* (5:1), pp. 14-37.

Olson, G. M., and Olson, J. S. 2000. "Distance Matters," *Human-Computer Interaction* (15:2), pp. 139-178.

Paulk, M. C. 1995. "How ISO 9001 Compares with the CMM," *IEEE Software* (12:1), January, pp. 74-83.

Paulk, M. C., Curtis, B., Chrissis, M. B., and Weber, C. V. 1993a. "Capability Maturity Model," *IEEE Software* (10:4), pp. 18-22.

Paulk, M., Weber, C. V., Garcia, S. M., Chrissis, M. B., and Bush, M. 1993b. "Key Practices of the Capability Maturity Model, Version 1.1," CMU/SEI-93-TR-025, Carnegie Mellon University, Pittsburgh.

Pesaran, M. H., and Deaton, A. S. 1978. "Testing Non-Tested Nonlinear Regression Models," *Econometrica* (46), pp. 677-694.

Pisano, G. P. 1994. "Knowledge, Integration and the Locus of Learning: An Empirical Analysis of Process Development," *Strategic Management Journal* (15:Special Issue on Competitive Organizational Behavior), Winter, pp. 85-100.

Raelin, J. A. 1997. "A Model of Work-Based Learning," *Organization Science* (8:6), pp. 563-578.

Ramasubbu, N., Krishnan, M. S., and Kompalli, P. 2005. "Leveraging Global Resources: A Process Maturity Framework for Managing Distributed Development," *IEEE Software* (22:3), pp. 80-86.

Ravichandran, T., and Rai, A. 2003. "Structural Analysis of the Impact of Knowledge Creation and Knowledge Embedding on Software Process Capability," *IEEE Transactions on Engineering Management* (50:3), pp. 270-284.

Rus, I., and Lindvall, M. 2002. "Knowledge Management in Software Engineering," *IEEE Software* (19:3), pp. 26-38.

Sarker, S., and Sahay, S. 1001. "Information Systems Development by US-Norwegian Virtual Teams: Implications of Time and Space," in *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, Los Alamitos, CA: IEEE Computer Society Press.

Scherer, F. M., and Ross, D. 1990. *Industrial Market Structure and Economic Performance*, Boston: Houghton Mifflin.

SEIR. 2007. "Software Engineering Information Repository," Software Engineering Institute, Carnegie Mellon University, Pittsburgh (online at http://seir.sei.cmu.edu).

SEMA. 2002. "Process Maturity Profile of the Software Community 2002 Mid-Year Update," SEMA 8.02, Software Engineering Institute, Pittsburgh.

Victor, B. 1990. "Coordinating Work in Complex Organizations," *Journal of Organizational Behavior* (11:3), pp. 187-199.

von Hippel, E., and Tyre, M. J. 1995. "How Learning by Doing Is Done: Problem Identification in Novel Process Equipment," *Research Policy* (24:1), January, pp. 1-12.

Winter, S. G. 2003. "Understanding Dynamic Capabilities," *Strategic Management Journal* (24:10), pp. 991-995.

Wong, S., and Burton, R. M. 2000. "Virtual Teams: What Are Their Characteristics, and Impact on Team Performance?," *Computational & Mathematical Organization Theory* (6:4), pp. 339-360.

Zellner, A. 1962. "An Efficient Method of Estimating Seemingly Unrelated Regressions and Tests for Aggregation Bias," *Journal of American Statistics Association* (57), pp. 348-368.

Zollo, M., and Winter, S. 2002. "Deliberate Learning and the Evolution of Dynamic Capabilities," *Organization Science* (13:3), pp. 339-351.

About the Authors

Narayan Ramasubbu is an assistant professor at the School of Information Systems at the Singapore Management University. He has a Ph.D. in Business Administration from the University of Michigan, Ann Arbor, and an Engineering degree from Bharathiyar University, India. Prior to pursuing the Ph.D., he was a senior developer and product management specialist, first at CGI Inc. and then at SAP AG. His research focuses on software engineering economics, distributed software product development, and software product standardization and customization.

Sunil Mithas is an assistant professor at the Robert H. Smith School of Business at the University of Maryland. He has a Ph.D. in Business from the University of Michigan and an Engineering degree from IIT, Roorkee. Prior to pursuing the Ph.D., he worked for about 10 years in engineering, marketing, and general management positions with Tata Steel and Tata Ryerson. His research focuses on strategic management and impact of information technology resources and has appeared or is forthcoming in journals that include *Management Science, Information Systems Research, MIS Quarterly, Journal of Marketing, Production and Operations Management, Journal of Management Information Systems*, and *Statistical Science.*

M. S. Krishnan is Mary and Mike Hallman e-Business Fellow and Professor of Business Information Technology at the University of Michigan Ross School of Business. Dr. Krishnan is also a co-director of the Center for Global Resource Leverage: India at the Ross School of Business. His research interests includes corporate IT strategy, the business value of IT investments, management of distributed business processes, software engineering economics, and metrics and measures for quality, productivity and customer satisfaction for software and other IT products. In January 2000, the American Society for Quality (ASQ) selected him as one of the 21 voices of quality for the 21st century. His research has appeared in several journals including *Management Science, Information Systems Research, Strategic Management Journal, Harvard Business Review*, and *Sloan Management Review*. He currently serves on the editorial board of *Information Systems Research*.

Chris F. Kemerer is the David M. Roderick Professor of Information Systems at the University of Pittsburgh and an Adjunct Professor of Computer Science at Carnegie Mellon University. Previously, he was an associate professor at MIT''s Sloan School of Management. His research interests include management and measurement issues in information systems and software engineering, and he has published more than 60 articles on these topics in a variety of professional and academic journals, as well as editing two books. He is the recipient of numerous teaching awards, and was the winner of the Distinguished Professor Award by the most recent University of Pittsburgh Executive MBA class. Dr. Kemerer is a past Departmental Editor for Information Systems at *Management Science*, immediate past Editor-in-Chief of *Information Systems Research*, and is a current Senior Editor for *MIS Quarterly*.

Appendix A

Dispersion Calculation for Project #32 (of 42)

Software Development Lifecycle Phase	Activity	Person-Hours at Center-1		Person-Hours at Center-2		Column Total
		Completing assigned deliverables	Integrating deliverables with customer delivery package	Completing assigned deliverables	Integrating deliverables with customer delivery package	
Requirements	End-user interaction to gather business requirements	240	40	107	13	400
	Feasibility analysis	33	4	150	28	215
	Proposing alternatives	96	21	229	39	385
	Review of requirements	121	17	54	8	200
	Rework in requirements phase	32	5	60	23	120
Design	High-level and low-level design	387	49	1165	183	1784
	Review of design	78	8	162	28	276
	Rework in design phase	5	1	11	3	20
Coding	Development of new modules	204	27	740	91	1062
	Integration of new modules with existing old applications	273	63	671	93	1100
	Code-inspections	175	24	420	7	626
	Rework during coding phase	77	18	188	17	300
Unit Testing	Development of use cases	126	10	200	35	371
	Executing unit tests and documenting errors	64	9	66	11	150
	Root-cause analysis	20	3	42	8	73
	Fixing unit testing errors	33	5	71	13	122
	Review of unit testing	28	4	54	9	95
	Rework during unit testing phase	31	4	65	10	110
Integration Testing	Development of use cases	46	6	97	15	164
	Executing integration tests and documenting errors	85	12	183	30	310
	Root-cause analysis	7	1	1	1	10
	Fixing integration testing errors	30	3	54	6	93
	Review of integration testing	35	5	75	13	128
	Rework during integration testing phase	3	0	0	0	3
Acceptance Testing	Development of use cases	113	15	106	16	250
	Executing acceptance tests and documenting errors	47	6	85	12	150
	Root-cause analysis	42	5	78	10	135
	Fixing acceptance testing errors	54	7	91	13	165
	Review of acceptance testing	16	3	25	6	50
	Rework during acceptance testing phase	13	3	20	15	51

Software Development Lifecycle Phase	Activity	Person-Hours at Center-1		Person-Hours at Center-2		Column Total
		Completing assigned deliverables	Integrating deliverables with customer delivery package	Completing assigned deliverables	Integrating deliverables with customer delivery package	
Project Management	Effort spent by project managers	132	15	243	34	424
Configuration Management	Configuration of deliverables	118	18	275	55	466
	Synchronization of builds	60	2	96	3	161
Others	Training and other learning activities, process improvement activities, and other miscellaneous effort	1003	10	2623	23	3659
Row Total		3827	423	8507	871	13628
		Total person-hours at center-1 = 4250		Total person-hours at center-2 = 9378		

Calculation of Work Dispersion Variables				
Task Dispersion	=	$[100^2 – (\%\ \text{effort at center-1})^2 – (\%\ \text{effort at center-2})^2]$	$[100^2 – ((4250/13628)*100)^2 – ((9378/13628)*100)^2]$	4292.051
Integration Intensity	=	(total person-hours spent on integrating deliverables) / (total person-hours of project)	(423+871) / 13628	0.095

Appendix B

Expert Mapping of Processes Invoking Conceptual and Operational Learning

CMM Maturity Level	Key Process Area	Processes Invoking Conceptual Learning	Processes Invoking Operational Learning
Level 2: Repeatable	Configuration Management	• Analyzing evolution pattern (functionality gain, defect rates, productivity) from several versions • Exploring usage of design patterns from configurations analysis • Analyzing junk code reduction in versions to improve system maintainability • Analysis of versions to form system replacement strategy	• Implementing version control mechanisms—manual or automated • Choosing individual configurations for system integration • Analyzing configurations update mechanism for accurate customization
	Quality Assurance	• Analysis of comprehensive data from past history to identify broad patterns • Use of statistical procedures to set benchmarks	• Designing templates, checklists , metrics and processes to implement quality check • Conducting quality audits
	Project Tracking and Oversight	• Analyzing efficiency of project management policies and managerial control mechanisms.	• Conducting status reviews • Conducting project postmortem • Monitoring plans and actual outcomes
	Planning	• Analyzing estimation effectiveness—finding reasons for overruns • Setting project policy for estimating effort for different technologies • Analyzing resource utilization pattern for hiring and human resource management	• Using estimation tools, templates • Compiling resource-task allocation maps
	Requirements Management	• Assessing effectiveness of requirements gathering policies including end user involvement • Comparing performances of requirements gathering strategies—which work best? • Gathering and analyzing metadata of requirements changes—customer characteristics, technology characteristics and internal project characteristics	• Creating baselines from requirements data • Creating prototypes, natural language documents from requirements data • Auditing the requirements gathering process

CMM Maturity Level	Key Process Area	Processes Invoking Conceptual Learning	Processes Invoking Operational Learning
Level 3: Defined	Organization Process Focus and Definition	• Control group analysis for new process standards • Monitoring and analyzing the results of process innovations for effectiveness and generalization • Conducting periodic time series analysis for analyzing project productivity gains and other outcomes	• Coordinating process improvement via software engineering group • Enhancing process data collection (for example via web enabled systems) • Creating project reports for senior management
	Training Program	• Choosing and coordinating university, customer or other specialty alliances for curriculum design • Specific training for skills enhancement—both technical and managerial	• Conducting role based training programs • Hands-on training based on simulated production environment
	Inter-group Coordination	• Analysis of team structures and communication patterns • Objective comparison of division of labor policies • Social networks analysis	• Infrastructure for formal and informal communication set and used • Collaboration schemes for aligning commitments • Schedules and deadlines integrated across related groups
	Peer Reviews	• Monitoring effectiveness of inspections– what makes inspections work? • Comparing peer review metrics with other verification schemes • Analyzing cost effectiveness of peer reviews	• Planning and conducting peer reviews • Analyzing and correcting defects from peer reviews • Communicating peer review improvements across teams
	Integrated Software Management	• Monitoring of the deviations from organizational standards—should the organizational process standards be changed?	• Tailoring organizational processes for project context
	Software Product Engineering	• Evaluating the effectiveness of measurement programs • Designing and testing software metrics suitable for projects in lab	• Designing and integrating release cycles with all software schedules • Collecting software metrics (e.g., CK metrics, Function Point, component metrics, etc.) and using them in planning
Level 4: Managed	Quantitative Process Management	• Designing project specific data analysis methods and tools for software engineering and process group • Assessing cost of quantitative process, management activities and accomplishment of milestones for quantitative process management activities	• Collecting and auditing project data that can be used by software engineering process group to identify anomalies and opportunities • Designing and implementing project data collection enabling tools
	Software Quality Management	• Comparing various quality management approaches for given project • Conducting statistical and simulation based analysis to derive quality goals based on past history • Analyzing of cost of quality	• Translating software product quality goals to individual developers task list • Implementing verification schemes such as inspections and testing
Level 5: Optimizing	Defect Prevention	• Analyzing cost effectiveness of defect prevention efforts • Investigating factors affecting performance of defect prevention activities • Exploring methods and choices for resource allocation to defect prevention vs. appraisal vs. failure approaches	• Conducting root-cause analysis meetings for defects prevention • SQA audit of defect prevention activities • Maintaining causes of defects in knowledge management database
	Technological Change Management	• Acquiring knowledge about new technological standards • Participation in standards setting • Testing, evaluating and selecting new technologies	• Migration techniques to new technological standards—both automated and manual • Audit of technology replacement
	Process Change Management	• Participation in industry bodies of software process improvement (example ISO, CMM) • Evaluation of process improvement (ROI) • Comparing generic process frameworks for organization suitability • Capturing best practices for key processes areas	• Training project personnel to adapt process changes • Adapting data collection templates, audit process to new processes • Synchronization between project quality database and organizational quality database