Governing Software Process Improvements in Globally Distributed Product Development

Narayan Ramasubbu, *Member, IEEE*

Abstract—Continuous software process improvement (SPI) practices have been extensively prescribed to improve performance of software projects. However, SPI implementation mechanisms have received little scholarly attention, especially in the context of distributed software product development. We took an action research approach to study the SPI journey of a large multinational enterprise that adopted a distributed product development strategy. We describe the interventions and action research cycles enacted over a period of five years in collaboration with the firm, which resulted in a custom SPI framework that catered to both the social and technical needs of the firm's distributed teams. Institutionalizing the process maturity framework got stalled initially because the SPI initiatives were perceived by product line managers as a mechanism for exercising wider controls by the firm's top management. The implementation mechanism was subsequently altered to co-opt product line managers, which contributed to a wider adoption of the SPI framework. Insights that emerge from our analysis of the firm's SPI journey pertain to the integration of the technical and social views of software development, preserving process diversity through the use of a multi-tiered, non-blueprint approach to SPI, the linkage between key process areas and project control modes, and the role of SPI in aiding organizational learning.

Index Terms—Software process improvement (SPI), distributed teams, software engineering, process control, action research

1 Introduction

Software process improvement (SPI) continues to be the centerpiece of quality management initiatives undertaken by software firms to develop products with high quality in a cost effective way. SPI refers to the set of actions taken to improve the capabilities of a software organization, and is typically based on normative frameworks such as the ISO 9001, ISO 15504 [1], Atern [2], CMMI [3], and Bootstrap [4]. There is a long tradition of SPI research in software engineering and management information systems disciplines [for example, [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]. This work is motivated by two notable gaps in the SPI literature. First, a majority of studies investigating SPI deal with colocated software development. Distributed software teams operate in an environment where coordination and administration are more challenging [15], [16], [17], [18], [19], and the need for managing social aspects of processes along with the necessary technical aspects is more pronounced [20], [21]. Exploring how the social needs of distributed product development teams such as coordination and trust could be catered to along with the traditional emphasis on software methodologies in SPI initiatives is important and necessary. Second, past SPI studies have not considered the role of diversity in software development practices among individual units of a large software organization. Investigating how processes of a product development organization can be continuously improved by leveraging, not negating, the diversity of the development practices will contribute to an improved understanding of how a balance can be achieved between predictability and flexibility in the context of distributed product development.

In this paper we fill these gaps in the SPI literature by investigating the design, implementation, and assessment of a distributed development process framework at a leading enterprise software product development firm (labeled as *Supersoft* in this paper for confidentiality). Our research was carried out as action research in collaboration with Supersoft practitioners. Action research uses intervention into problematic social situations as a means to advance scientific knowledge and has gained wide acceptance among research scholars [14], [22], [23], [24]. Following the action research cycle of diagnosing, action planning, action taking, evaluating, and specifying learning [24], we helped the distributed product development teams at Supersoft to improve their process capabilities and software development performance. We explicate the details of two action research cycles that we employed over a period of five years, and document the progress we have made towards developing an improved understanding of the process improvement mechanisms in globally distributed software product development.

The rest of the paper is organized as follows. We introduce the problem situation faced by Supersoft in Section 2 and our research methodology in Section 3. Following this we explain Intervention 1 in Section 4, where the management of Supersoft attempted to initiate changes in its software development processes, aiming at improving the capabilities of the units involved in distributed product development. The attempted change was first met with reservations from key product line managers who had contrasting beliefs on the needs for control, flexibility, and

• *The author is with the Joseph M. Katz Graduate School of Business, University of Pittsburgh, 354 Mervis Hall, Pittsburgh, PA 15260. E-mail: narayanr@pitt.edu.*

Manuscript received 2 Dec. 2012; revised 26 June 2013; accepted 1 Dec. 2013; date of publication 11 Dec. 2013; date of current version 28 Mar. 2014. Recommended for acceptance by E. Di Nitto.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TSE.2013.58

diversity of processes. Subsequently, a collaborative effort was undertaken to develop a process maturity framework that encompassed both the standardization and customization aspects of development processes with an increased participation from Supersoft product line managers and developers. This resulted in a three-step (multi-tiered) evolutionary process maturity framework that had the buy-in of key managers and developers. The first action research cycle was completed with the description of how the newly developed process maturity framework was implemented for the projects undertaken by eight development teams and subsequently evaluated.

Following this a second initiative (Intervention 2) was undertaken by Supersoft management to institutionalize the newly developed process maturity framework. In Section 5 we describe how this institutionalization effort initially got stalled when a benchmarking event triggered product line managers to perceive the management-sponsored SPI initiative as a mechanism for exercising wider control. The implementation of the process framework was subsequently altered to convey the legitimate intentions of the management and co-opt product line managers, which led to a wider adoption of the framework. Finally, in Section 6 we discuss the broader implications for research and practice, and conclude.

This study makes three key contributions. First, we extend the line of enquiry in SPI research to address process improvement in distributed product development teams. We develop key process areas (KPAs) that specifically address the fundamental characteristics of distributed work identified in prior literature, namely, *'collaboration readiness'*, *'technology readiness'*, *'common ground'*, and *'coupling in work'* [20]. Second, by explicating the five-year SPI journey of a large software enterprise that involved the use of a custom-developed SPI framework in which the above KPAs were embedded, we add a focus on SPI implementation mechanisms to the growing body of empirical software engineering literature. Third, we establish the interlinkages between project-level process improvements and broader firm-level mechanisms, highlighting the need to design SPI operations in a way not only to assist product teams that are improving their software process capabilities, but also to contribute to broader organizational processes such as governance and learning.

2 The Problem Situation

Supersoft is a large European enterprise software product company with total revenues of over 14 billion Euros in 2012, employs more than 10,000 software developers around the globe, and has an installed base of over 100,000 customers in 120 countries. For over forty years Supersoft has been in the business of developing enterprise software application products catering to core corporate functions such as financials, human resource management, sales and order management, and customer relationship management. Supersoft's products were traditionally based on a proprietary programming language and technology framework. However, in the recent years as Supersoft faced intense competition from rivals to re-architect its products, it began accommodating the emerging Internet-enabled technological paradigms and standards. Further, Supersoft was expanding its product lines to address the needs of small and medium businesses apart from its traditional focus on large Fortune 500 corporate customers. Thus, there was a sudden surge in the need for software development personnel, which could not be satisfied by recruitment and expansion at the Europe-based home development center alone. This led to the adoption of a globally distributed product development strategy at Supersoft.

Supersoft's distributed development strategy was motivated by its prior experience in setting up large sales and distribution units around the globe. The strategy was further aided by lower labor costs and the availability of a large pool of English-educated computer science and engineering graduates in the emerging economies of Brazil, China, India, and Eastern Europe. These emerging economies were also seen by Supersoft executives as lucrative market segments for the new product lines targeted at small and medium business enterprises. Supersoft began implementing its globally distributed product development strategy by establishing 10 new development centers as wholly-owned subsidiaries around the globe. Over a period of two years following the establishment of the new development centers, software development pertaining to eight out of a total 11 Supersoft product lines were transitioned to the distributed development model.

At the end of the first product release cycle after the establishment of the distributed development model at Supersoft, an internal review was conducted to assess the distributed product development practices at Supersoft. The review revealed that there was a dramatic increase (about 300 percent) in the overall number of man-hours spent on new application development and the number of projects executed by individual product lines. Supersoft's executive management board was convinced that the distributed development strategy was heading in the right direction. However, the board members wanted to assess the effectiveness of the operational mechanisms through which distributed development was conducted. Upon the board's request an internal audit on the distributed projects was conducted by a joint committee of the firm's central Software Engineering Process Group (SEPG) and a board-appointed quality assurance team. The audit committee unearthed several shortcomings in the project performance and development processes of the distributed product development teams at Supersoft.

The audit committee findings revealed that the processes followed by the distributed teams significantly deviated from the existing organizational guidelines that had been derived from the ISO 9001 and ISO 15504 standards. The committee documented more than 400 non-conformance issues that were not properly recorded and addressed by various software teams, and warned that there was a very high risk of losing the ISO 9001 certification during the next renewal. Furthermore, the cycle time and defect density of the components developed by the distributed teams was more than two times the existing organizational benchmarks. There was a huge backlog of quality and reliability complaints from customers that were not resolved by the distributed product teams (on average about 20 conformance quality complaints per

TABLE 1
Time Line of Action Research

Stage	Time Period	Task
Project feasibility discussions	August-September 2005	Get project approval.
Setting up collaboration	January 2006-August 2006	Develop guidelines for collaboration. Setting up project goals and defining roles and responsibilities.
Intervention 1	September 2006 – September 2008	Achieving process control, consistency, and capability improvement.
Intervention 2	October 2008 – October 2010	Institutionalizing the distributed process framework.
Firm-wide global adoption and evaluation	November 2010-November 2011	Global roll-out of refined process framework and implementation mechanism.
Project closure	January 2012	Final presentation to the Supersoft executive management, signoff of SPI project, and hand over to global quality management team.

KLOC). Moreover, an internal employee survey conducted by the audit committee revealed poor job satisfaction scores reported by the distributed team members. The audit committee noted that a chief driver of employee dissatisfaction was the lack of proper guidelines for task allocation and project management for distributed teams, which resulted in huge variation in the productivity and work load allocation across distributed team members. Overall, the audit committee recommended that the board undertake a thorough examination and overhaul of the distributed development processes at Supersoft.

The audit committee findings spurred the Supersoft executive management board to convene the SEPG to devise a solution to improve the firm's distributed product development environment. This marked the beginning of an SPI journey at Supersoft that is analyzed in this paper.

3 Research Methodology

This research study emerged as part of a collaboration project between the Center for Global Resource Leverage at the University of Michigan and Supersoft. We adopted an action research approach because we wanted to solve the immediate problem situation at Supersoft and at the same time wanted to improve our understanding of the tenets of SPI in distributed product development teams, especially by linking theory and practice. A clear agreement was formed between the researchers involved in this study and the Supersoft team about the individual roles and responsibilities. As part of the agreement, researchers were allowed to take active part in the SPI effort at Supersoft and could use the experience and data collected through the effort to pursue research publications, contingent on an agreed upon non-disclosure agreement.

The participation of the researchers in the project was funded by the University of Michigan Center for Global Resource Leverage. The author was physically present at the Supersoft development centers for 18 months and maintained continuous contact with the project members at Supersoft during the entire time period of the research project. The research infrastructure for the project included a research group that involved three academic researchers, the Supersoft SPI team consisting of four key SEPG managers, eight representatives from the quality management group of the individual product lines, and an executive management steering group that consisted of two directors and an extended board member. We followed iterative action research cycles with each cycle consisting of diagnosing, action planning, action taking, evaluating, and specifying learning phases [24]. The timeline of this action research project is shown in Table 1.

Mapping action research items with the related research literature. The action research methodology facilitated constant interaction between Supersoft practitioners and researchers throughout the active period of the study. As researchers we were interested in regularly organizing and mapping the emerging themes in the project to related research literature. This helped us to prepare a structured approach for our interviews and discussions. However, given the multi-year nature of this study, organizing the discussion insights and mapping the uncovered issues to the research literature was an iterative and evolving process. We initially started with a checklist of items derived from the distributed teams literature from the software engineering and management literature. These checklist items are shown as the row headers in Table 2; similar to snowball sampling, with every discussion the range of issues and the corresponding literature references kept expanding. Table 2 shows the range of issues that we touched upon in our discussions, organized according to research themes in the literature.[1]

4 Intervention 1: Aiming at Process Control and Consistency

4.1 Intervention 1: Diagnosing

As noted before, an internal audit committee report had highlighted the key shortcomings of the distributed product development processes at Supersoft. The first step in our diagnosing effort was to understand the findings of the internal audit committee. After independently analyzing the audit committee report, the entire SPI project group met to deliberate and perform root-cause analysis. The SPI project group interacted with the internal audit committee members to seek additional explanations and clarifications on their audit exercise, following which a detailed agenda for a diagnosis effort was drawn. Based on the initial findings of the audit committee (refer to Section 2), the SPI project team agreed that there were three distinct areas that need to be looked at as problem domains:

1. Performance of distributed product development teams (doubling of defect density and cycle time and the high backlog of customer complaints).
2. Process control of distributed development projects (about 400 undocumented issues related to non-conformance to ISO standards).
3. Human resource issues of distributed development teams (poor employee satisfaction, high task variability, and poor productivity).

1. This is not meant to be a systematic literature review, but only a motivating starting point for the related research literature. Specific studies that we used to develop key process components are presented later in the paper (Tables 6, 7, 8, and 9).

TABLE 2
Organizing and Mapping Problem Issues with Research Literature

S.No.	Problem event related to	Literature Reference
1	Project Administration a) Project planning, estimation, scheduling b) Roles and responsibilities, who-is-who of project c) Task allocation, disparity in work load, etc. d) Specification of project goals and deliverables	[8, 9, 16, 18, 25]
2	Coordination a) Policies and procedures for coordination b) Dispute over project milestones, schedules and deliverables c) Ambiguity over who is in-charge of critical tasks d) Difficulty in coordinating expertise and garnering support e) Difficulty in scheduling meetings and informal communication among distributed team members	[15, 17, 20, 21, 26, 27, 28]
3	Software Methodology a) Availability of defined and agreed upon development methodology b) Consistent processes followed across development centers c) Project tracking mechanisms in place d) Quality management guidelines defined and adopted	[6, 7, 8, 9, 13, 17]
4	Human Resources Management a) Availability of resources b) Capability of resources to execute project tasks c) Attrition d) Motivation of resources to perform well and positively contribute to project success e) Ambiguity of job description and role played in project f) Cultural differences and disparity in working styles	[18, 28, 29, 30, 31]
5	Knowledge Integration a) Access to knowledge regarding the business function enabled by the developed product b) Access to technical knowledge related to the product development platform c) Mechanisms to train and exchange knowledge among distributed team members	[17, 32]
6	Technology Factors a) Unstable, unreliable development platform b) New paradigms with insufficient documentation, etc. c) Infrastructure to support synchronization of work in distributed product development d) Infrastructure to support communication mechanisms in distributed teams	[18, 20, 33]

The next step in the diagnosing stage was to conduct in-depth discussions with the key stake holders involved in distributed product development at Supersoft. We collected detailed inputs from product line managers, developers, and quality management personnel of eight product lines who had adopted the distributed development model. We conducted a total of 32 interviews that were structured to get the stakeholder observations on the three problem domains noted above. A detailed description of our analysis of the problem domains follows.

Problem Domain 1—Project performance. The audit committee had documented that the cycle time and defect density of the Supersoft distributed development projects were well above prior organizational benchmarks. We sought to understand the underlying reasons for this in our discussion with the stake holders of the distributed development projects. Three propositions on underperformance emerged from our discussions with the project stake holders. First, the firm's traditional performance management benchmarks, their measurement, and metrics such as cycle time and defect density were perceived as inapplicable to the distributed development projects. A product development manager offered this insight:

> *"…these new greenfield projects cannot be pitted against the existing benchmarks. The benchmarks are a reflection of the past. We are in a new phase of product development. We are trying many new things—newer technologies, newer teams, newer ways of developing software. The risk-performance scopes of these new projects are completely different from the projects that originally contributed to the benchmarks. So comparison with the older benchmarks is neither useful nor fair."*

Second, there was a lack of support structure for project administration in globally distributed teams. The current ISO 9001 and 15504 standards-based process infrastructure at Supersoft was not adequate to handle distributed development. The following excerpts from interviews with development managers highlight the crucial missing parts in the process framework:

> *"….by the time we started planning for the project, the estimation template could not accommodate two locations delivering code at different time schedules. There was no function point or functionality point counting scheme for the new programming framework we were about to use. We used the older counting method and later found out that there is a 56 percent slippage in the size estimation…"*

> *"In my previous projects we used to spend a maximum of 5 percent effort in project administration. But now we are spending more than 20 percent of effort on administrative issues, especially because we are helping the quality group to set up the administration infrastructure. From a budgeting point of view, this should not be accounted in our project. There is no external customer at this point and every account is a cost center, so we did not pay attention to that."*

A third proposition that emerged from our discussions to diagnose the performance issues was related to task

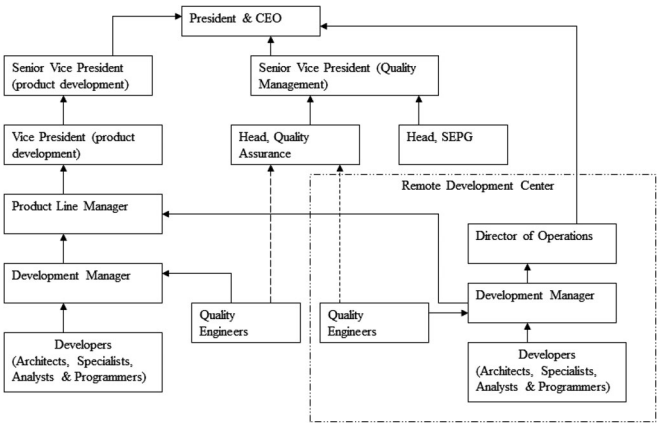

Fig. 1. Project organization structure.

allocation mechanisms and managing the overall division of labor. Task allocation in Supersoft development projects traditionally followed a proactive culture wherein developers signed up for fulfilling tasks enlisted by the development manager. However, in the distributed development setting proactive sign-up for project tasks had led to a high proportion of tightly integrated tasks to be split across the development centers. This, in turn, seemed to increase the system integration effort and also the number of integration errors. A development manager noted:

> *"I did not want to disrupt the proactive culture in my project team by pre-assigning tasks. I strongly believe that even if tightly linked tasks get split across the development centers, it contributes to the learning across the project team. This is an organic way to disseminate product knowledge to everyone. There might be a cost to this decision, but at this stage of team development I am willing to pay that cost."*

Most developers held the notion that while the proactive task pickup culture was an asset to the organization, there was a need to allocate tasks in a better way, and that the development managers need to interfere to optimize the tradeoffs between the costs of proactive task selection and its benefits. A senior developer shared the following opinion in an interview:

> *"Proactive signup for tasks gives us [a] lot of flexibility and autonomy in choosing our daily work. When we sign up for the tasks there is only a 50 percent chance that we are aware of the cross-site dependencies. There is only so much that you can figure out from the design documents. Many integration issues are discovered only as we progress with the task. People who are in charge of the tasks that have dependencies often do not have deep knowledge about the integration of the components that they are working on. As a result, several people who have not signed up for a task eventually end up work[ing] on it anyways; one-to-one mapping of tasks to developers exists only on the roster. This situation can be improved if there is some interference in the task planning by development managers who have the knowledge and experience to suggest a desired [task] allocation pattern as a recommendation."*

Summarizing the propositions that emerged from our discussions, the first order reasons that directly limited the performance of distributed product development teams were: (1) usage of older and irrelevant measurement and performance metrics and benchmarks, (2) lack of project management administration support structure, and (3) non-optimal task allocation patterns for distributed teams.

Problem Domain 2—Process controls. The next common theme that emerged during our diagnosis effort was related to process controls employed on the distributed development projects. Recall that the audit committee had noted about 400 issues related to non-conformance to ISO standards. The diagnosis effort to uncover the root cause of the non-conformance problem highlighted the lack of autonomy for quality engineers and the inadequate participation of SEPG in the quality assurance processes of distributed teams. Fig. 1 presents the structure of a typical distributed development project organization at Supersoft. As the figure shows the SEPG was not directly in control of development activity in remote development centers of Supersoft. Product line managers held direct control of development activity

TABLE 3
Portfolios of Control

(Column 1) Control Mode	(Column 2) Mechanisms of control observed in literature [34-36]	(Column 3) Encounters of control mechanisms in Supersoft distributed development projects	(Column 4) Control Elements encountered in Supersoft distributed development projects
Outcome Control Emphasize on results and focus on measurement and benchmarking.	Project milestone; Budget, Implementation dates; Design document; Functional Specifications; Software testing.	• Project Plan • Specifications document • Design document • Functionality added in product release	*Controller:* Product line manager, Development Manager *Measurement:* Observing product functionality releases *Evaluation:* Map release pattern with original vision *Rewards and Sanctions:* Nil
Behavior Control Restrict undesirable behaviors through surveillance, and develop a link between behaviors and formal rewards.	Development methodology; Job description, roles and responsibilities; Task allocation; Walk-throughs, weekly reports.	• Monthly Meetings • Team performance report • Individual performance report	*Controller:* Product line manager, Development Manager *Measurement:* Progress in terms of % task completed, Contributions to project *Evaluation:* Performance assessment exercise *Rewards and Sanctions:* Individual and Team bonus
Clan Control Induce desirable shared values, beliefs, and culture in a team through peer pressure	Socialization; Hiring and training practices.	• Product release vision • Team outings • Mentor-Mentee programs	*Controller:* Product line manager *Measurement:* Nil *Evaluation:* Nil *Rewards and Sanctions:* Nil
Self Control Focus on autonomy and build control through self-motivation, and self-regulation of individual team members	Self set goals; Self-monitoring through personal software process; autonomy in task picking and work.	• Proactive signing-up for tasks • Self learning	*Controller:* Team members *Measurement:* Nil *Evaluation:* Nil *Rewards and Sanctions:* Nil

TABLE 4
Disparate Value Systems

Contributions highly valued by management in development center 1	Contributions highly valued by management in development center 2
• Innovative ideas that significantly enhance the product features or performance (Product focus). • Technical excellence – be on the forefront of new technology paradigms and apply them in product development. • Developing ties with other product development groups at Supersoft and ability to push forward multi-product enhancements. • Ability to be a specialist with deep knowledge in a chosen role.	• Contributions to organizational development – administration, setting up processes, training, team building effort, etc. (Organization focus). • Ability to bring more work to the development center. • Ability to develop generic skills that can be leveraged in multiple roles.

and quality engineers in the remote centers. Individual product line managers sponsored distributed development projects within their units, and did not engage with centralized resources such as the SEPG while enacting policies governing remote development centers.

To further understand the role of SEPG and quality assurance, we analyzed the structure of controls in Supersoft's distributed development projects using the portfolios of controls framework developed in the information systems development literature. Prior research studies [34], [35], [36] note that organizations tend to employ a portfolio of controls for software project management, which have been organized into four control modes, namely, *outcome, behavior, clan*, and *self-control* modes (see first two columns in Table 3). We mapped the project mechanisms at Supersoft to these control modes (column 3 in Table 3), and documented who initiated and owned the control mechanisms (controller), the metrics involved in the control mechanism (measurement), how the project activity was evaluated (evaluation), and how the teams were rewarded or penalized for their performance (rewards and sanctions).

As shown in Table 3 (column 4), our analysis indicated that outcome and behavior controls modes were used more often than clan and self-control modes in Supersoft distributed development projects. Although a conspicuous proactive and autonomous working culture existed among Supersoft distributed development members, it was not leveraged fully in terms of exercising project controls (i.e., no usage of self or clan control modes). Further, the implementation of control elements in the outcome and behavior control modes was not consistent and rigorous across the global development sites. For example, there were no specific measurement guidelines to capture project performance metrics. The centralized SEPG and quality management teams were not involved in ensuring the effectiveness of measurement and evaluation schemes across different projects and development sites. The absence of these central resources in performance assessment meant that the best practices and learning from individual distributed product development projects could not be explicitly integrated into the firm's organizational memory mechanisms.

The evaluations involved in performance assessment exercises were largely development manager specific, and there were no firm-wide agreed upon guidelines or standards. Furthermore, the project controls were based on outcomes of prior tasks executed to satisfy other functional processes, rather than on the distributed development context of the teams. For example, the controllers extensively used aggregated information from product release notes such as functionality releases and product release patterns and compared it with market competition. However, they did not use the teams' project plan and design documents, which were rich artifacts created from the software methodology steps, to analyze deviations of actual outcomes from the planned ones. Thus, we listed ineffective project controls as the next problem area that needed to be addressed by the SPI team.

TABLE 5
Problem Areas and Proposed Solutions

S. No.	Column 1: Problem Domain	Column 2: Proposed Solutions	Column 3: SPI Implementation Action Items
1	***Project Performance:*** Benchmarks for performance of distributed development teams are not available	Ensure autonomy of quality engineers to collect independent data and contribute to a global benchmarking exercise using relevant metrics.	Reorganize reporting structure of quality engineers.
2	***Project Performance:*** Lack of support structure for project management, administration	Develop distributed development process and project framework on top of the existing ISO 9001 process framework. Enhance participation of central SEPG and quality teams in distributed development.	Extend current project processes with additional key process areas in new process framework.
3	***Project controls*:** lack of autonomy for quality engineers and no participation of SEPG	Change organization structure of distributed development projects to enhance transparency and visibility. Devise a distributed development project assessment framework.	Implement new organization structure to have direct supervision and participation of SEPG in quality assurance activities of distributed teams.
4	***Human Resources:*** Task allocation and schedule variability for distributed teams	Enforce balance between proactive signup and optimal integration effort; technical specialists to recommend optimal spread of tasks for each product lines. Task allocation mechanisms will be brought under the purview of new distributed development process framework.	Extend current project processes with additional key process areas in new process framework.
5	***Human Resources:*** Problems with incentives and performance reward for distributed teams	Enforce dual roles for development managers and uniform performance evaluation methods.	Extend current project processes with additional key process areas in new process framework.

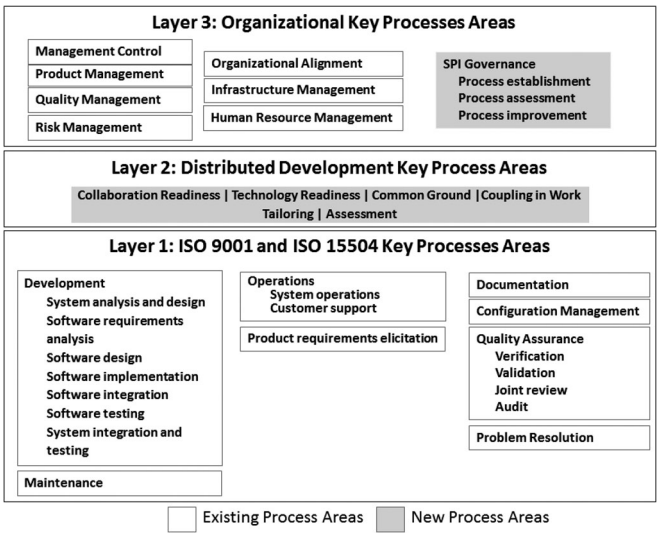

Fig. 2. Multi-tiered process framework overview.

Problem Domain 3—Human resources issues: There were several human resources-related issues that we observed through our discussions with the stake holders and the examination of the audit committee report. As mentioned before, although performance assessment was used as a key outcome control mechanism, there were no consistent guidelines or established performance assessment criteria to evaluate individual performance. Instead the development managers used their own judgment to assess performance, often in a variety of subjective ways. Also, from our discussions with the managers and developers, we noticed that the 'value system' through which contributions were assessed varied across the development centers. Table 4 shows the contributions that were highly valued at two development centers, highlighting the differences in what were perceived as superior performance across the centers. The differences in the value systems were especially problematic to personnel who frequently transitioned between the development centers. These boundary spanning resources that played a crucial role in product development missed out on rewards such as bonuses because of the differences in the managerial value systems. The following excerpts from our interviews with the developers further illustrate these issues:

> *"Since last year I have not received any quarterly bonus and salary rises. My manager thinks travel is a privilege and is a learning opportunity that my other colleagues do not enjoy. But when I go back to my home center I am at a lower growth path. Next time I will think twice before I agree for business travel."*

> *"When I was asked to move to <development center> for a year, there was no clear transition plan for my return. When I was away, my team back home got restructured and I was reporting to three development managers. None of them included me for team bonus considerations and I was left in the lurch."*

Another key people-related issue that we uncovered was the impact of task variability on individuals due to work dispersion across different development centers. Distributed development teams faced a work environment where work load patterns varied significantly. An interview quote from a developer illustrates this:

> *"I know that this is not a 9-5 job. But there were long periods in this project where I had to do night shifts and that was not by my choice. I want to avoid night shifts in the future and I know that it can be done if we pay closer attention to the release schedules upfront."*

While the proactive work culture at Supersoft encouraged developers to design their own work schedules, there was a need to choreograph task allocation across different sites in order to prevent wide fluctuations in work schedules for individual developers. Thus, improving the performance-based incentive structure and task scheduling patterns for distributed teams were identified as the final important problem area to be addressed by the SPI project.

4.2 Intervention 1: Action Planning

Following the problem diagnosis phase we collaborated with the various Supersoft groups involved in the SPI effort to consider remedial measures for the key issues that were identified. The SEPG collated proposals and organized workshops to discuss and debate the proposed solutions. Table 5 summarizes the problem areas (column 1) and the corresponding solutions (column 2) that the SPI team proposed to remedy the problems. Two major implementation action items emerged from the proposed solutions (column 3):

1. Devise a new distributed development process framework that will expand the current Supersoft ISO 9001 and 15504 standards-based processes to address the specific needs of distributed development teams. This would address the two problem domain areas of poor project performance due to outdated processes and metrics, and human resource-related issues. An overview of the proposed extensions to the process framework as envisioned by the SPI team is presented in Fig. 2 (more details in Section 4.3).
2. Implement structural changes to the distributed development organization to enhance the autonomy of the quality function, including greater role for central resources such as the firm's SEPG. This relates to the alteration of the reporting structure of the quality engineers involved in distributed development. In the proposed solution, quality engineers would be associated with distributed development teams as full time members, but would have a direct reporting line to the quality assurance and SEPG heads.

The diagnosed problems and the envisioned plans were presented to a Supersoft management committee consisting of vice-presidents of three product development divisions, two development center directors, and the senior vice-president of quality management. The committee welcomed the move to extend the current processes to be adapted to the needs of the distributed development context, but raised concerns on how the flexibility of their processes could be maintained in a regime of centralized process control. The SEPG and the quality assurance team agreed that the processes followed at team levels should be the responsibility of the development teams, but highlighted the need for the

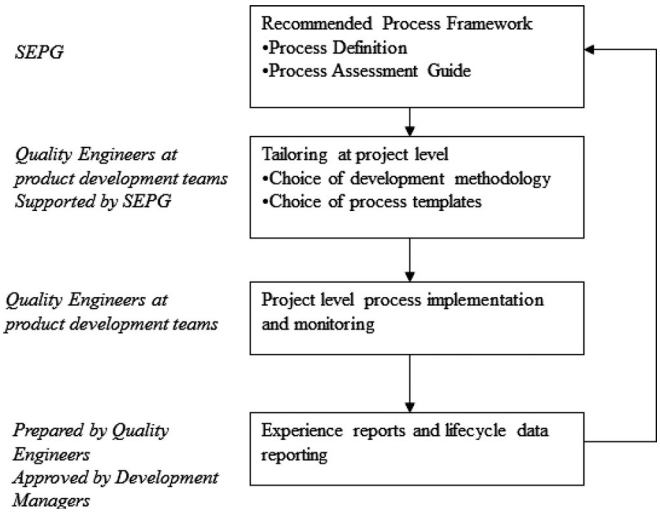

Fig. 3. New key process area implementation steps.

autonomy of their quality engineers. Autonomy of quality engineers would ensure reliability of the benchmark data constructed from individual projects and would help the flow of best practices and learning to other parts of the organization. This relates to the concepts advanced by scholars that that effort spent on operational and conceptual learning can improve the overall effectiveness of process improvements [17], [37], [38].

To balance the control requirements sought by the SEPG and the flexibility needs sought by product development teams, extensive tailoring of KPAs in the new framework was allowed. Tailoring provides flexibility to product development teams to choose the specific process implementation steps (choices in development methodology, templates, process steps, etc.) under the supervision and guidance of centralized SEPG governance. Following initial deliberations, an agreement was reached on the roles, responsibilities, and the process implementation stages for the distributed product development teams. Fig. 3 represents the agreed upon SPI installation steps that would ensure the autonomy of the quality function without jeopardizing the flexibility of process tailoring by individual product line teams. The management committee also approved the next steps to develop the extensions to current process framework, and to run a pilot program based on distributed development projects of eight teams.

4.3 Intervention 1: Action Taking—Developing the Key Process Areas

The first step in the action taking stage was to implement the structural changes to the quality function in the eight distributed development teams that participated in the pilot study. While the quality engineers took part in the development projects as before, they now directly reported to the central quality assurance and SEPG heads. This assured the autonomy of their operations and at the same time did not disrupt their commitment to the specific product development team that they belonged to.

The next step was to develop the extensions to the process framework of Supersoft to address the specific needs of distributed software development teams. The starting point for this effort was the consideration of four key concepts synthesized by Olson and Olson [20] and shown as essential for geographically distributed work: (1) collaboration readiness, (2) common ground, (3) coupling in work, and (4) technology readiness. We discuss each of these foundation concepts and describe how the necessary distributed development key process areas of Supersoft were built on them.

'Collaboration readiness' refers to the willingness and ability of team members to participate effectively in distributed development. It has been shown in prior research that the main driver of collaboration readiness comes from the individual beliefs and trusts that developers place on their remote colleagues [39]. In the absence of mutual trust, interdependent tasks cannot be successfully negotiated and accomplished. Collaboration readiness of distributed teams could be positively influenced by appropriate organizational governance structures that explicitly specify the collaboration requirements, administration structure, and individual participants' goals, roles, and incentives. In the absence of a clear governance structure, cross-functional and cross-locational conflicts, and ambiguous authority could arise in project management that would eventually hamper project performance [17]. Also, the role of management leadership is important in propagating the needs for adopting a collaborative strategy and in steering distributed product development teams towards the acceptance of the governing principles for collaboration [40]. Thus, the concept of collaboration readiness informs us of the need for process areas that facilitate trust formation, goal setting, design administrative, incentive and performance structures, and top management participation. Table 6 details the

TABLE 6
Key Process Areas for Improving Collaboration Readiness

Key Process Area	Supporting Literature
Trust and belief: Assess and induce belief and willingness of stake holder to participate in distributed work	[39, 41, 42]
Team structure: Develop guidelines for distributed team structure and participatory roles	[43, 44]
Shared goal setting: Facilitate shared business goals between participating teams	[45, 46]
Tailoring of goals: Tailor higher organizational and business goals to the operational and project level	[45, 46]
Budgeting and cost ownership: Assess, estimate and allocate budgets for distributed work	[47]
Contractual stability: Monitor contractual relationships to nurture long term stability	[48, 49]
Management communication channel: Create efficient flow of information between top management executives and participating teams and organizations	[40]
Managerial Span: Monitor the social and professional networks, and span of influence of managers	[43]
Performance management: Assess and track contributions of individual participants of distributed work	[50, 51, 52]
Competency management: Identify core competency of each participating organization and appropriate consideration in organizational strategy	[53, 54]
Strategic significance: Facilitate mutual participation of remote teams in each other's mission, vision and strategy	[46, 51, 54]

TABLE 7
Key Process for Establishing Common Ground

Key Process Area	Supporting Literature
Communication skills: Assess and train personnel for improved communication skills	[18, 51, 62]
Cultural awareness: Nurture and enrich mutual awareness to counter differences	[18, 56, 57, 58]
Remote people management: Develop processes to help managers communicate with and manage remote personnel	[59, 60]
Project management consistency: Ensure consistent usage of tools, processes, project reporting and tracking among distant teams	[6, 61]
Knowledge management: Facilitate knowledge transfer and establish policies for knowledge management	[55, 62, 63, 64, 65]
Best practices: Identify best practices and proliferate among remote teams	[66]

TABLE 8
Key Process Areas for Managing Coupling in Work

Key Process Area	Supporting Literature
Division of Labor: Institute policies to guide managers in task allocation, distribution and coordination of resources	[18, 68, 69, 70, 71]
Functional Ownership: Identify and monitor individual functional ownership and responsibilities among dispersed team members	[72, 73, 74]
Complexity management: Facilitate mapping of product and team modularity, and optimizing it for managing interdependencies	[68, 75, 76, 77]
Innovation management: Institute policies to manage cross-locational and cross-functional innovations	[49, 78, 79]

TABLE 9
Key Process Areas for Aiding Technology Readiness

Key Process Area	Supporting Literature
Distributed technology infrastructure: Deploy, monitor and upgrade technical infrastructure for cross-site communication	[18, 49, 81, 82]
Integrated tool environment: Deploy, monitor and upgrade infrastructure for participative coding, reviewing, debugging, server synchronization and other programming related work	[83, 84]
Infrastructure for social development: Deploy infrastructure to facilitate non-work related communication among remote colleagues	[18, 26, 85]

KPAs we developed to improve collaboration readiness along with the references in literature that guided us.

The next foundational concept, *'common ground'*, emphasizes the mutual knowledge and awareness that remote participants share with each other such as knowledge about the history and legacy of the software product they are working on, their company culture, common experience with customers, and basic understanding of their working styles. Prior research informs us that development of such mutual knowledge is crucial for effective communication between teams separated by distance [55]. Thus, the concept of common ground informs us of the need for process areas necessary to nurture common understanding and mutual knowledge. Table 7 details the KPAs related to mutual knowledge and the associated literature that inspired us.

The third foundational concept, *'coupling in work'*, reflects the mechanisms for division of labor that control interdependencies in distributed product development. In order to minimize interdependencies and integration costs, we have to pay close attention to the relationship between the architecture of the system and the organization of system development resources [15], [27]. One of the key system architecture characteristics that could be exploited to organize software development tasks is modularity [67], [68], [69]. It is important that the distributed product development process framework elicits guidelines on how product development work could be modularized appropriately, and how work units can be spanned across distributed teams. Thus, the concept of coupling in work highlights the need for including process areas for managing interdependencies of tasks and the appropriate division of labor. Table 8 details the key process areas for managing coupling in work along with the supporting representative studies in literature.

Finally, *'technology readiness'* refers to the availability and usage of collaborative technology infrastructure such as groupware, video conferencing, multi-authoring tools, server synchronization, etc., to conduct distributed product development. Distinguishing mere availability from efficient usage, Orlikowski [80] points out that efficient utilization of collaborative technologies depend on two important factors, the structural properties of the organization (such as policies and reward structures) and personnel's cognition about technology and their work. Thus, to ensure technology readiness in distributed teams, there is a need to focus on process areas that emphasize the availability and accessibility of enabling technologies along with a provision for monitoring the utilization patterns of the technologies. Table 9 lists the KPAs we developed to satisfy the 'technology readiness' need.

*Evolutionary distributed development process framework***.** The next step was to arrange the KPAs in an evolutionary framework. An evolutionary framework would help firms identify and assess their current process situation and plan for future improvements in an incremental fashion. Also, since implementation of software process models demand significant and sometimes risky organizational changes, an incremental and evolutionary approach might result in a wider adoption of the proposed practices while simultaneously improving the capabilities of the firm for further changes. To arrange the key process areas in an evolutionary framework, we sought expert opinion within Supersoft. We formed a review committee that consisted of 34 randomly selected experts from a variety of roles such as certified quality assurance managers, development managers, program directors, and software product development vice-presidents; these experts had more than ten years of industry experience. We conducted a workshop with the review committee where we first presented the back ground of our project and its objectives, and then asked the experts to rank order the KPAs we developed along an evolutionary maturity path. We provided the experts with a list of the distributed development KPAs (Tables 6, 7, 8, and 9) in randomly sorted order, and asked them to group and rank order the

TABLE 10
Evolutionary Distributed Process Maturity Framework

Maturity Level	Key Process Areas
Level 0: No systematic processes level	NA
Level 1: Initial level	Trust and belief
	Communication skills
	Distributed technology infrastructure
	Team structure
	Cultural awareness
	Remote people management
	Shared goal setting
	Tailoring of goals
	Budgeting and cost ownership
Level 2 : Consolidation level	Division of labor
	Integrated tool environment
	Project management consistency
	Functional ownership
	Knowledge management
	Management communication channel
Level 3 : High productivity level	Performance management
	Complexity management
	Managerial span
	Infrastructure for social development
	Competency management
	Innovation management
	Best practices
	Contractual stability
	Strategic significance

KPAs on a framework consisting of three levels in an increasing order of capability. Responses from this exercise were then analyzed, and the positioning of all the KPAs was resolved without ambiguity. The interrater reliability value of the ranking derived by the 34 experts from the above exercise was 0.8 as calculated by the Kendall's coefficient of concordance, which is above the recommended threshold in the literature [86]. Table 10 shows the final evolutionary process maturity framework that was agreed upon by the expert review committee.

The quality assurance and SEPG departments at Supersoft took the responsibility to train development and managerial personnel of the eight pilot teams in the new distributed process framework. Individual project teams and line managers were trained in all the KPAs of the process framework. After the training was completed, the process framework was released to the pilot teams for implementation. The quality engineers of each product line teams took responsibility for the implementation and monitoring of the framework in the projects. Monthly internal audits and training sessions were conducted by the quality engineers as well.

4.4 Intervention 1: Evaluating

Our evaluation of the actions taken during intervention 1 consisted of two stages. In the first stage, we analyzed if the problems identified during the diagnosis step were resolved after the actions of intervention 1. We noticed that after the restructuring of the quality function in the distributed development teams, quality engineers and SEPG members were able to play a greater role in the process control of the distributed project teams. The roles and responsibilities of the quality engineers had expanded significantly from a predominantly reactive testing and verification function before the restructuring to a proactive project management support role after empowering them with autonomy. They conducted monthly review sessions with the distributed product development teams, and also supported project administration in terms of providing assistance in project effort and size estimation, keeping track of project progress, and enacting process control. The data collected and verified by the quality engineers were passed on to the central SEPG repository and contributed to the development of benchmarks specific to the distributed development projects. Our assessment of the project control modes used in the projects after the adoption of the expanded process maturity framework indicated that the main improvements were reflected in the increased use of quantitative and

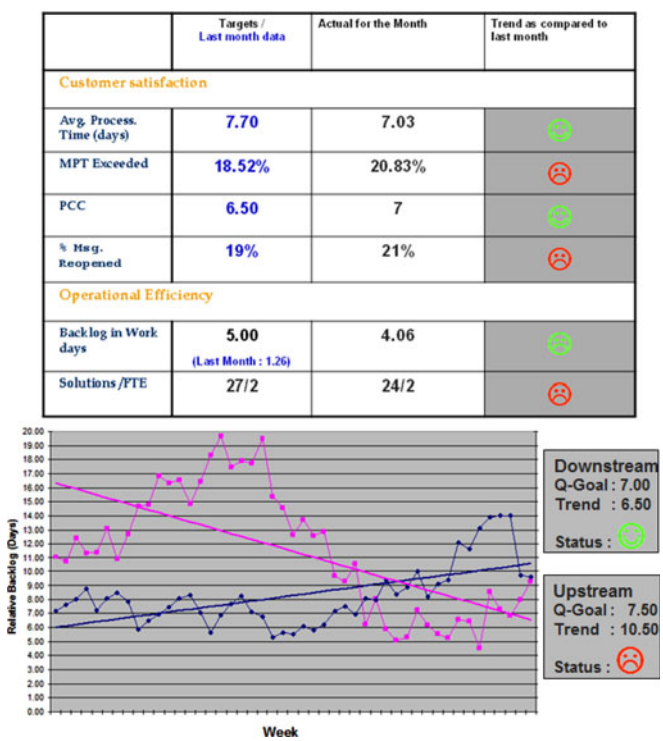

Fig. 4. Sample reports from autonomous quality engineers.

TABLE 11
Quantitative Assessment of SPI Effort

11a) Summary Statistics of Evaluation Data

Construct	Measurement	Mean	Improvement	Std dev.	Min	Max
Customer satisfaction	Rating on 1-10 Likert scale Online survey of customers	7.36	20%	0.93	4.33	9.5
Process maturity[2]	% KPAs from Levels 1-3 (Table 10) fulfilled satisfactorily as assessed by the audit committee	54.16	N/A	27.08	37.5	100
Dispersion	% of effort done at a single development center to the total effort spent on the project	0.36	36%	0.18	0.05	0.5
Personnel capability	Rated by development manager on 1-10 Likert scale	7.49	N/A	1.37	3	10
Post-sale support	Extent of support given to the product support team Rating of 1-10 Likert scale	7.05	3%	1.02	4.5	10
SLA failure	Number of times the project team failed to meet the service level agreements (SLA)	33.53	-10%	46.53	0	226
Solution fill-rate	% of solutions delivered by the team to the total solutions requested by customers	45.94	15%	11.59	21.95	78.57
Team size	Head count of the product development team	4.90	N/A	3.47	1	18
Processing time	Number of days to resolve a customer message	11.55	-10%	5.26	2.2	33.77
Complexity	Feature points delivered by the team	198.92	N/A	204.08	10	816

11b) Regression Models

SLA failure =	$\beta 0 + \beta 1 * \text{process maturity} + \beta 2 * \text{dispersion} + \beta 3 * \text{personnel capability} + \beta 4 * \text{team size} + \beta 5 * \text{complexity} + \beta 6 * \text{processing time} + \epsilon$ ..... *Equation 1*
Customer satisfaction =	$\beta 0 + \beta 1 * \text{process maturity} + \beta 2 * \text{dispersion} + \beta 3 * \text{SLA-failure} + \beta 4 * \text{personnel capability} + \beta 5 * \text{post-sale support} + \beta 6 * \text{solution fill-rate} + \epsilon$ ..... *Equation 2*

11c) Regression Results (N=62)[≠]

Variable		SLA failure	Customer satisfaction
Process maturity	β1	-10.479** (0.018)	0.289** (0.019)
Dispersion	β2	-3.864 (0.768)	1.26*** (0.000)
SLA failure	β3	NA	0.002 (0.842)
Personnel capability	β4	-1.471 (0.391)	0.254 *** (0.000)
Post sale support	β5	NA	0.462*** (0.000)
Solution fill-rate	β6	NA	- 0.015 *** (0.006)
Team size	β7	-4.725*** (0.000)	NA
Complexity	β8	0.267*** (0.000)	NA
Processing time	β9	-1.06 (0.018)	NA
Constant	β0	46.12*** (0.007)	1.951*** (0.000)
R-squared		85.29%	73.15 %
CHI-squared		359 .41*** (0.000)	169.33 *** (0.000)

[2] *Process maturity was initially assessed using an ordinal scale (Levels 1-3). As per the recommendation of anonymous reviewers, to avoid using ordinal variables in the quantitative analysis, we report process maturity as a continuous variable using the % of KPAs fulfilled by a project team. Results were consistent across the usage of process maturity as an ordinal variable or as a % of KPAs measure in the regressions.*

[≠] *Two tailed P-values in parenthesis;* $^{***}P < 0.01, ^{**}P < 0.05, ^{*}P < 0.1$

objective data to measure and evaluate outcomes. Project managers could readily access quantitative data at the task level activities in the project and could track the trends over time. Fig. 4 shows a sample quality report, illustrating the increased use of relevant objective quantitative metrics. Thus, the need for better control modes and project management and administration support, which was identified as a problem area before, was successfully fulfilled through the restructuring of the quality function that was enacted in intervention 1.

The second stage of our evaluation involved a quantitative assessment of the impacts of the process improvement efforts undertaken by the eight product development teams. We assessed 62 projects completed by the eight product line teams using the new process maturity framework, two years after the teams had adopted the framework for their operations. Results from this assessment were used to predict the performance measures of the distributed projects executed by the eight teams using statistical regression models. This helped us to understand if process improvement in the distributed teams contributed to better project performance.

The details of the quantitative study are presented in Table 11; the summary statistics of the data we collected is shown in Table 11a and the regression models are described in Table 11b. The regression models capture the effects of improvements in process maturity (β1) and work dispersion (β2) on two performance measures, namely, SLA failures and customer satisfaction, controlling for the other variables shown in Equations (1) and (2).

The quantitative evaluation results indicated that all the project teams showed improvement in their project performance after the adoption of the new distributed development process framework for their operations. As shown in Table 11a, in the 62 projects that the teams completed, there was a significant increase in work dispersion (36 percent on average). On average, teams had adopted 14 out of the total 24 KPAs and moved from level-0 to level-2 in process maturity; there were 14 projects that had achieved level-3 in process maturity.[3] Through higher process maturity, the teams were able to improve customer satisfaction and reduce quality failures and error processing times. Specifically, the regressions results (Table 11c) show that as the project teams rise in the process maturity levels β1, for example, from level-0 (adhoc) to level-2 two (initiation phase), failures were reduced by as much as ten times and customer satisfaction scores increased by more than 20 percent (see β1 in Table 11c). Moreover, in the presence of process maturity improvements, work dispersion resulting from the distributed development strategy had a significant positive effect on the performance metrics (see β2 in Table 11c).

Thus, in evaluating the actions of intervention 1,[4] we saw that the SPI initiative in the 62 projects of the eight distributed development teams had positive impacts on the performance of the distributed teams. The distributed development strategy of Supersoft was working effectively in the presence of ongoing improvements in process maturity.

4.5 Intervention 1: Specifying Learning

Three key lessons emerged from our experience with intervention 1. First, in a distributed development environment, significant benefits can be reaped if the governing process framework caters to both the technical and social needs of teams. On one hand, views of SPI as epitomized by propositions such as "software processes are software too" [87] emphasize the technical, rational, and mechanistic design aspects of software development processes. On the other hand, a social view of software development emphasizes that software development involves the production of a technical artifact that is socially produced and consumed [88], [89], [90]. In this study we embraced a sociotechnical approach by developing and implementing an SPI framework that included KPAs pertaining to both the technical and social needs of distributed teams. Referring to Fig. 2, we note that most of the technical needs of Supersoft product development were satisfied through layer 1 of the SPI framework. The social needs of distributed development were met through the layer 2 KPAs, and the firm-level process needs were fulfilled through the layer 3 KPAs.

Second, to ensure effective SPI implementation it is important to strike a balance between the needs for flexibility and control. Our experience shows that even when diverse practices exist in a development environment, one can achieve meaningful process control through a multi-tiered SPI approach. Extensive tailoring choices in both the layer-1 ISO 9001, 15504 standards-based KPAs and the layer-2 KPAs of the new distributed development process framework allowed teams to choose diverse development methodologies, quality verification procedures, and project organization structures. By not following a standardized, one-size-fits-all blueprint approach to SPI [91], [92], we were able to leverage the existing process diversity to help teams adhere to a continuous improvement culture.

Our third learning point relates to the need for autonomy of the quality function at the project level. By ensuring autonomy for the quality function, managers can ensure that the experiences of the teams can contribute to organizational learning. Experience at the project level (operational learning) can enter the organizational knowledge network, where other agents such as the central SEPG can help reason out the experiences to gain conceptual learning, which eventually assists in the sustenance of the continuous improvement journey [17].

5 Intervention 2: Institutionalizing Process Improvement Framework

After intervention 1, Supersoft management had expanded the distributed process maturity framework to all the distributed product development projects of the firm. The framework and process assessments were made an essential part of the annual audit of the teams. Following an annual audit in 2010, results of the firm-wide benchmarking exercise were published in an internal newsletter to the distributed development teams. This was the first time that the relative positions of individual project teams in terms of performance indicators was published within Supersoft. The publication was widely perceived as a signal that top management was intent on comparing the performance of development teams at a global level through the SPI initiative. Typically, Supersoft project teams were organized according to the product lines, and individual product line teams were rarely compared with each other. The move to benchmark individual product line teams globally created a significant debate in staff meetings and other employee forums within Supersoft. Individual product line managers resisted the move to compare their teams with others. Development managers of distributed development teams withdrew support to the institutionalization process of the SPI initiative and withheld project data from being updated in the central SEPG data repository. Intervention 2 was introduced at this time to help the management understand and resolve the challenges in institutionalizing the SPI framework.

5.1 Intervention 2: Diagnosing

Similar to Intervention 1, the diagnosing step involved a series of discussions with the development managers, product line managers and senior developers. The central argument of all the development managers and product line managers who resisted the adoption of the new process framework and restructuring of quality function was that

3. We thank the anonymous reviewers for pointing out the heterogeneity in improving process maturity and performance among the project teams. This variance in the adoption of KPAs and the corresponding performance outcomes in the 62 projects formed the basis of the regressions results shown in Table 11 c.

4. Regression robustness tests and diagnostics are omitted due to page limit considerations. We tested for heteroskedasticity, normality of residuals, influence of outliers, and multi-collinearity, and did not find any problems.

the SPI initiative could be used as an intrusive control mechanism by the firm's executive management. While the SPI program was perceived as improving the capabilities of the product development teams, it was also seen as an enabler of behavior observability and outcome measurability at a fine granular level, which the individual team managers were anxious about. When the behavior observability and outcome measurability of tasks improve, the portfolio of control modes that can be used by the controller expands significantly [34], [35], [36]. The publication of the global benchmarking data at Supersoft was perceived by individual teams as a breach of trust in using self-reported data for the purposes of implementing top management control.

5.2 Intervention 2: Action Planning

Our discussions with the executive management committee involving the vice-presidents of product development and quality assurance revealed that Supersoft top management had not planned to use data from the SPI project in the ways perceived by the middle managers. Even without the data generated by the SPI initiative, data on the product line revenues, customer complaints, and market penetration were widely available to the top management to assess the performance of individual product lines. Supersoft top management indicated that such high-level data were more appropriate for exercising controls at the product line level than the low-level data collected through the SPI project such as the number of errors and number of backlog support messages. This clarification confirmed that the management was genuinely interested in improving the capabilities of the development organization rather than exercising wider controls using the data from the SPI initiative. Based on this the SPI committee recommended the following:

1. The reporting procedure of quality engineers need not be altered from the original situation. The metrics compiled by the autonomous quality engineers need to be jointly vetted and approved by the product line managers so that they do not perceive it as unwarranted spying. While this reporting procedure compromises the autonomy of the quality function in the distributed development teams, it aids in removing the misconceptions that the SPI initiative is aimed at exercising excessive management control.
2. Product line managers have full ownership of the processes and software engineering data derived from their projects. They may disclose it to the central SEPG at an aggregate level and on a voluntary basis for the broader purpose of organization learning.
3. Process control mechanisms of distributed development must be audited through the regular annual audits conducted by the central SEPG and quality assurance teams. Any separate benchmarking exercises need approval of the individual product line managers who have the authority to decline participation in such exercises.

These proposals were approved by the management committee for wider implementation.

5.3 Intervention 2: Action Taking

Detailed workshops were conducted for all the product development teams involved in distributed development to advertise the changes made to the administration of the software process improvement framework. The distributed development teams that had benefited from the SPI initiative shared their individual experiences with the other participants in the workshops. The management committee members disseminated their position and the proposed changes to the SPI administration in staff meetings and through their individual communication with the stake holders. Following this the central SEPG started scheduling the formal induction of distributed product development teams into the newly developed multi-tier process maturity framework.

5.4 Intervention 2: Evaluating

The evaluation of intervention 2 was fairly straight forward. After six months from the start of the intervention 2, 90 percent of the distributed product development teams had signed up for the SPI initiative. Also, a majority of the teams that had adopted the new process framework voluntarily completed the SEPG's formal audit of their distributed development processes. Subsequently, the SPI framework was inducted into the training program of new hires of the development centers. These developments indicated that the SPI initiative was beginning to be widely adopted at Supersoft and a culture of proactive and continuous process improvement was setting in among the distributed teams.

5.5 Intervention 2: Specifying Learning

A key inference from intervention 2 is that implementation of process frameworks would not succeed without the support of the entire chain of agents involved in the organizational change process. A top-down approach to implement SPI would be resisted unless the goals of the desired organizational change are perceived as legitimate at the grassroots level. While SPI initiatives have the potential to expand the portfolios of control of top management, a control-oriented SPI implementation is likely to fail in the longer run due to principal-agency related problems. Unlike in the production of physical goods, agents involved in knowledge work have more control over the source of process improvement data. Thus, when SPI is approached as a control mechanism rather than as a capability improvement and empowerment mechanism, the costs of implementing the controls become far higher than the realized benefits.

A second learning from intervention 2 relates to a tradeoff between autonomy of project-level agents and broader organizational learning requirements. At Supersoft, to solve the principal-agency problem in institutionalizing the SPI initiative, the autonomy of the quality function had to be compromised. This limited the flow of knowledge from individual project learning experience to the organizational knowledge repository. Thus, other complementary investments such as rewards for sharing best practices are necessary to enhance the integration of local project-level learning experience into the organizational-level knowledge repositories.

6 Discussion

6.1 Implications for Research

Several studies have associated SPI to improved performance outcomes such as cycle time, project effort, productivity, and quality [6], [7], [13], [93]. While the success of SPI initiatives depend on the cooperation and agreement of the agents involved in software production, much of the theorized benefits do not directly relate to the agents who perform the day-to-day steps of SPI programs. While there could be benefits of the SPI initiatives to the individual agents such as developers, these benefits have not been predominantly theorized or studied in SPI research. Our action research indicates that agents do not readily see the benefits of SPI initiatives and tend to perceive the move to implement SPI as a control mechanism in the beginning. They agree to adopt SPI practices only when there are sufficient guarantees to prevent direct and fine-grained behavior observability and outcome measurability. Thus, the observations of this study points to the need for further research to explicate how the benefits realized through process improvements can be modeled from the perspectives of both the principal (management sponsoring SPI initiatives) and the agent (developers who operationalize the SPI steps).

A second implication that arises for research is regarding the relationship between process improvements and the portfolios of controls employed in software development. In this study we observed that there are strong links between KPAs of SPI frameworks such as the one used at Supersoft and the control mechanisms (such as methodology, project plans and goals, etc.) employed through the outcome, behavior, clan and self-control modes. An implication that arises because of this linkage is that the portfolios of control modes and their effectiveness can vary as a particular development team journeys through the SPI evolution. Thus, there is a need to study the interaction dynamics of process capabilities of development teams and the effectiveness of control portfolios employed for them.

Furthermore, this research highlighted the interaction between the technical and social aspects of task allocation in distributed development teams. There is an implicit relationship between the social coordination and the technical integration costs resulting from the task allocation patterns in distributed development. Thus, researchers studying coordination patterns in distributed software teams need to account for the endogenous relationships between architectural dependencies in the underlying systems and the dependencies resulting from team organization and task allocation.

Finally, the findings of this research related to the relationship between the autonomy of the quality function and the flow of knowledge to organizational repositories highlights a new opportunity for exploring the linkages between SPI and organizational learning in a software development context. Understanding how best practices and learning in a local environment can be shared to a wider audience at the organizational level, especially when the actors involved in socializing the best practices are globally distributed, could be an important first step in that direction.

6.2 Implications for Practice

We offer three prescriptions for practitioners based on our experience with the process improvement effort at Supersoft. First, when software organizations are looking to improve the process capabilities of distributed development teams, they should pay attention to both the social and technical needs of these teams. The choice of the normative process framework to be used in the SPI initiative should be guided by the need to fulfill both the social and technical aspects of software development. This research demonstrates how KPAs specific to distributed software teams can be accommodated on top of the process frameworks based on popular and widely adopted standards such as the ISO 9001 and 15504.

Second, to successfully propagate and sustain continuous process improvement efforts in a large organization, SPI proponents should leverage the diversity of practices present in the organization by facilitating a multi-tiered process framework that mandates adherence to fundamental principles, but offers customization in the ways to operationalize team-level implementations.

Finally, proponents of SPI in an organization have to pay close attention to the potential principal-agent problem that could arise during institutionalization of process frameworks. This can be avoided by 1) demonstrating the realization of agents-specific benefits of the SPI initiative, and 2) by establishing transparent procedures and policies governing the usage of such agent-generated SPI data for wider organizational usage.

Acknowledgments

The author thanks Alain Lesaffre, Prasad Kompalli, Martin Prinz, Andre Wachholz-Prill, Ganapathy Subramanian, and numerous other employees at the research sites for their support. Judy Olson, Gary Olson, M.S. Krishnan, Giri Kumar Tayi, Rajiv Kohli, S. Sadagopan, Pankaj Jalote, seminar participants at the University of Michigan, Michigan State University, Singapore Management University, and Indian Institute of Information Technology, Bangalore provided insightful comments on an earlier version of the paper.

References

[1] K.E. Emam, J.-N. Drouin, and W. Melo, *SPICE: The Theory and Practice of Software Process Improvement and Capability Determination*. Wiley-IEEE Computer Soc. Press, 1997.
[2] DSDM, http://www.dsdm.org/dsdm-atern/, June 2013.
[3] M. West, *Real Process Improvement Using the CMMI*. Auerbach Publications, 2008.
[4] P. Koch, S. Kuvaja, L. Mila, A. Krzanik, S. Bicego, and G. Saukkonen, *Software Process Assessment and Improvement: the Bootstrap Approach*. Blackwell, 1994.
[5] A. Fuggetta and G.P. Picco, "An Annotated Bibliography on Software Process Improvement," *ACM SIGSOFT Software Eng. Notes*, vol. 19, no. 3, pp. 66-68, 1994.
[6] M.S. Krishnan and M.I. Kellner, "Measuring Process Consistency: Implications for Reducing Software Defects," *IEEE Trans. Software Eng.*, vol. 25, no. 6, pp. 800-815, Nov./Dec. 1999.
[7] D.E. Harter and S.A. Slaughter, "Quality Improvement and Infrastructure Activity Costs in Software Development: A Longitudinal Analysis," *Management Science*, vol. 49, no. 6, pp. 784-800, 2003.
[8] W.S. Humphrey, *A Discipline for Software Engineering*. Addison-Wesley Professional, 1994.
[9] B.W. Boehm, *Software Engineering Economics*. Prentice Hall, 1981.

[10] I. Aaen, J. Arent, L. Mathiassen, and O. Ngwenyama, "A Conceptual Map of Software Process Improvement," *Scandinavian J. Information Systems*, vol. 12, pp. 123-146, 2001.

[11] B. Fitzgerald and T. O'Kane, "A Longitudinal Study of Software Process Improvement," *IEEE Software*, vol. 16, no. 3, pp. 37-45, May/June 1999.

[12] T. Ravichandran and A. Rai, "Structural Analysis of the Impact of Knowledge Creation and Knowledge Embedding on Software Process Capability," *IEEE Trans. Eng. Management*, vol. 50, no. 3, pp. 270-284, Aug. 2003.

[13] J. Herbsleb, D. Zubrow, D. Goldenson, W. Hayes, and M. Paulk, "Software Quality and the Capability Maturity Model," *Comm. ACM*, vol. 40, no. 6, pp. 30-40, 1997.

[14] J.H. Iversen, L. Mathiassen, and P.A. Nielsen, "Managing Risk in Software Process Improvement: An Action Research Approach," *MIS Quarterly*, vol. 28, no. 3, pp. 395-433, 2004.

[15] J.D. Herbsleb and A. Mockus, "An Empirical Study of Speed and Communication in Globally Distributed Software Development," *IEEE Trans. Software Eng.*, vol. 29, no. 6, pp. 481-494, June 2003.

[16] M.L. Maznevski and K.M. Chudoba, "Bridging Space over Time: Global Virtual Team Dynamic and Effectiveness," *Organization Science*, vol. 11, no. 5, pp. 473-492, 2000.

[17] N. Ramasubbu, S. Mithas, M.S. Krishnan, and C.F. Kemerer, "Work Dispersion, Process-Based Learning, and Offshore Software Development Performance," *MIS Quarterly*, vol. 32, no. 2, pp. 437-458, 2008.

[18] E. Carmel, *Global Software Teams: Collaborating Across Borders and Time Zones*. Prentice Hall, 1999.

[19] A. Espinosa and E. Carmel, "The Impact of Time Separation on Coordination in Global Software Teams: A Conceptual Foundation," *J. Software Process: Practice and Improvement*, vol. 8, no. 4, pp. 249-266, 2004.

[20] G.M. Olson and J.S. Olson, "Distance Matters," *Human-Computer Interaction*, vol. 15, no. 2, pp. 139-178, 2000.

[21] M.B. O'Leary and J.N. Cummings, "The Spatial, Temporal and Configurational Characteristics of Geographic Dispersion in Teams," *MIS Quarterly*, vol. 31, no. 3, pp. 433-452, 2007.

[22] R.L. Baskerville and J. Stage, "Controlling Prototype Development through Risk Analysis," *MIS Quarterly*, vol. 20, no. 4, pp. 481-504, 1996.

[23] R. Kohli and W.J. Kettinger, "Informating the Clan: Controlling Physicians Costs and Outcomes," *MIS Quarterly*, vol. 28, no. 3, pp. 363-394, 2004.

[24] G.I. Susman and R.D. Evered, "An Assessment of the Scientific Merits of Action Research," *Administrative Science Quarterly*, vol. 23, no. 4, pp. 582-603, 1978.

[25] R.K. Smith, J.E. Hale, and A.S. Parrish, "An Empirical Study Using Task Assignment Patterns to Improve the Accuracy of Software Effort Estimation," *IEEE Trans. Software Eng.*, vol. 27, no. 3, pp. 264-271, Mar. 2001.

[26] R.E. Kraut and L.A. Streeter, "Coordination in Software Development," *Comm. ACM*, vol. 38, no. 3, pp. 69-81, 1995.

[27] J.D. Herbsleb and A. Mockus, "Formulation and Preliminary Test of an Empirical Theory of Coordination in Software Engineering," *Proc. Ninth European Software Eng. Conf. Held Jointly with the 11th ACM SIGSOFT Int'l Symp. Foundations of Software Eng.*, pp. 138-147, 2003.

[28] M.M. Montoya-Weiss, A.P. Massey, and M. Song, "Getting It Together: Temporal Coordination and Conflict Management in Global Virtual Teams," *Academy of Management J.*, vol. 44, no. 6, pp. 1251-1262, 2001.

[29] N. Zakaria, A. Amelinckx, and D. Wilemon, "Working Together Apart? Building a Knowledge-Sharing Culture for Global Virtual Teams," *Creativity and Innovation Management*, vol. 13, no. 1, pp. 15-29, 2004.

[30] D. Vogel, M. Genuchten, D. Lou, S. Verveen, M. Eekout, and A. Adams, "Exploratory Research on the Role of National and Professional Cultures in a Distributed Learning Project," *IEEE Trans. Professional Comm.*, vol. 44, no. 2, pp. 114-125, June 2001.

[31] J.A. Espinosa, J.N. Cummings, J.M. Wilson, and B.M. Pearce, "Team Boundary Issues Across Multiple Global Firms," *J. Management Information Systems*, vol. 19, no. 4, pp. 157-190, 2003.

[32] A. Tiwana, "Knowledge Partitioning in Outsourced Software Development: A Field Study," *Proc. Int'l Conf. Information Systems*, pp. 259-270, 2003.

[33] J. Lipnack and J. Stamps, *Virtual Teams: Reaching Across Space, Time and Organizations with Technology*. John Wiley & Sons, 1997.

[34] K.M. Eisenhardt, "Control: Organizational and Economic Approaches," *Management Science*, vol. 31, no. 2, pp. 134-149, 1985.

[35] L. Kirsch, "The Management of Complex Tasks in Organizations: Controlling the Systems Development Process," *Organization Science*, vol. 7, no. 1, pp. 1-21, 1996.

[36] V. Choudhury and R. Sabherwal, "Portfolios of Control in Outsourced Software Development Projects," *Information Systems Research*, vol. 14, no. 3, pp. 291-314, 2003.

[37] M.S. Feldman and B.T. Pentland, "Reconceptualizing Organizational Routines as a Source of Flexibility and Change," *Administrative Science Quarterly*, vol. 48, no. 1, pp. 94-118, 2003.

[38] A.S. Mukherjee, M.A. Lapre, and L.N. v. Wassenhove, "Knowledge Driven Quality Improvement," *Management Science*, vol. 44, no. 11, pp. S35-S49, 1998.

[39] S.L. Jarvenpaa and D.E. Leidner, "Communication and Trust in Global Virtual Teams," *Organization Science*, vol. 10, no. 6, pp. 791-815, 1999.

[40] T.R. Kayworth and D.E. Leidner, "Leadership Effectiveness in Global Virtual Teams," *J. Management Information Systems*, vol. 18, no. 3, pp. 7-41, 2002.

[41] S.L. Jarvenpaa, K. Knoll, and D. Leidner, "Is Anybody Out There? Antecedents of Trust in Global Virtual Teams," *J. Management Information Systems*, vol. 14, no. 4, pp. 29-64, 1998.

[42] D. Meyerson, K.E. Weick, and R.M. Kramer, "Swift Trust and Temporary Groups," *Trust in Organizations: Frontiers of Theory and Research*, R.M. Kramer and T.R. Tyler, eds., pp. 166-195, Sage Publications, 1996.

[43] M. Ahuja and K. Carley, "Network Structure in Virtual Organizations," *Organization Science*, vol. 10, no. 6, pp. 741-757, 1999.

[44] J. Eveland and T. Bikson, "Work Group Structures and Computer Support," *ACM Trans. Office Information Systems*, vol. 6, no. 4, pp. 354-379, 1988.

[45] G. Hertel, U. Konradt, and B. Orlikowski, "Managing Distance by Interdependence: Goal Setting, Task Interdependence, and Team-Based Rewards in Virtual Teams," *European J. Work and Organizational Psychology*, vol. 13, no. 1, pp. 1-28, 2004.

[46] S.A. Furst, M. Reeves, B. Rosen, and R.S. Blackburn, "Managing the Lifecycle of Virtual Teams," *The Academy of Management Executive*, vol. 18, no. 2, pp. 6-20, 2004.

[47] E. McDonough, K. Khan, and G. Barczak, "An Investigation of the Use of Global Virtual and Collocated New Product Development Teams," *J. Product Innovation Management*, vol. 18, no. 2, pp. 110-120, 2001.

[48] A. Gopal, K. Sivaramakrishnan, M.S. Krishnan, and T. Mukhopadhyay, "Contracts in Offshore Software Development: An Empirical Analysis," *Management Science*, vol. 49, no. 12, pp. 1671-1683, 2003.

[49] D.W. Karolak, *Global Software Development*. IEEE Computer Soc. Press, 1998.

[50] E.E. Lawler, "Pay Systems for Virtual Teams," *Virtual Teams that Work: Creating Conditions for Virtual Team Effectiveness*, C.B. Gibson and S.G. Cohen, eds., Jossey-Bass, 2003.

[51] R. Blackburn, S. Furst, and B. Rosen, "Building a Winning Virtual Team: KSAs, Selection, Training and Evaluation," *Virtual Teams that Work: Creating Conditions for Virtual Team Effectiveness*, C.B. Gibson and S.G. Cohen, eds., Jossey-Bass, 2003.

[52] R.W. Collins and S.J. Birkin, "Challenges of Managing the Global IS/IT Workforce," *Strategies for Managing IS/IT Personnel*, M. Igbaria and C. Shayo, eds., pp. 195-212, Idea Group Publishing, 2003.

[53] G. Hamel and C.K. Prahalad, "The Core Competence of the Corporation," *Harvard Business Rev.*, vol. 68, no. 3, pp. 79-93, 1990.

[54] D.C. Galunic and S. Rodan, "Resource Recombinations in the Firm: Knowledge Structures and the Potential for Schumpeterian Innovation," *Strategic Management J.*, vol. 19, pp. 1193-1201, 1998.

[55] C.D. Cramton, "The Mutual Knowledge Problem and Its Consequences for Dispersed Collaboration," *Organization Science*, vol. 12, no. 3, pp. 346-371, 2001.

[56] N. Zakaria, A. Amelinckx, and D. Wilemon, "Working Together Apart? Building a Knowledge-Sharing Culture for Global Virtual Teams," *Creativity and Innovation Management*, vol. 13, no. 1, pp. 15-29, 2004.

[57] D. Vogel, M. Genuchten, D. Lou, S. Verveen, M. Eekout, and A. Adams, "Exploratory Research on the Role of National and Professional Cultures in a Distributed Learning Project," *IEEE Trans. Professional Comm.*, vol. 44, no. 2, pp. 114-125, 2001.
[58] E. MacGregor, Y. Hsieh, and P. Kruchten, "Cultural Patterns in Software Process Mishaps: Incidents in Global Projects," *Proc. Workshop Human and Social Factors of Software Eng.*, pp. 1-5, 2005.
[59] R.W. Collins and S.J. Birkin, "Challenges of Managing the Global IS/IT Workforce," *Strategies for Managing IS/IT Personnel*, M. Igbaria and C. Shayo, eds., pp. 195-212, Idea group publishing, 2003.
[60] K. Fisher and M. Fisher, *The Distance Manager: A Hands on Guide to Managing Off-Site Employees and Virtual Teams*. McGraw-Hill, 2000.
[61] P.F. Rad and G. Levin, *Achieving Project Management Success Using Virtual Teams*. J. Ross Publishing, 2003.
[62] J. Suchan and G. Hayzak, "The Communication Characteristics of Virtual Teams: A Case Study," *IEEE Trans. Professional Comm.*, vol. 44, no. 3, pp. 174-186, Sept. 2001.
[63] D. Sole and L. Applegate, "Knowledge Sharing Practices and Technology Use Norms in Dispersed Development Teams," *Proc. 21st Int'l Conf. Information Systems*, pp. 581-587, 2000.
[64] A. Majchrzak, R. Rice, N. King, A. Malhotra, and S. Ba, "Computer-Mediated Inter-Organizational Knowledge-Sharing: Insights from a Virtual Team Innovating Using a Collaborative Tool," *Information Resources Management J.*, vol. 13, no. 1, pp. 44-53, 2000.
[65] J. Cummings, "Workgroups, Structural Diversity and Knowledge Sharing in a Global Organization," *Management Science*, vol. 50, no. 10, pp. 1348-13651, 2004.
[66] J. Lurey and M. Raisinhgani, "An Empirical Study of Best Practices in Virtual Teams," *Information and Management*, vol. 38, no. 8, pp. 523-544, 2001.
[67] D.L. Parnas, "On the Criteria to Be Used in Decomposing Systems into Modules," *Comm. ACM*, vol. 15, no. 12, pp. 1053-1058, 1972.
[68] N. Ramasubbu, C.F. Kemerer, and J. Hong, "Structural Complexity and Programmer Strategies: An Experimental Test," *IEEE Trans. Software Eng.*, vol. 38, no. 5, pp. 1054-1068, Sept./Oct. 2012.
[69] J.D. Herbsleb and R.E. Grinter, "Splitting the Organization and Integrating the Code: Conway's Law Revisited," *Proc. 21st Int'l Conf. Software Eng.*, pp. 85-95, 1999.
[70] T.W. Malone and K. Crowston, "The Interdisciplinary Study of Coordination," *ACM Computing Surveys*, vol. 26, no. 1, pp. 87-119, 1994.
[71] J. Duggan, J. Byne, and G.J. Lyons, "A Task Allocation Optimizer for Software Construction," *IEEE Software*, vol. 21, no. 3, pp. 76-82, May/June 2004.
[72] R.C. Ford and A. W. Randolph, "Cross-Functional Structures: A Review and Integration of Matrix Organization and Project Management," *J. Management*, vol. 18, no. 2, pp. 267-294, 1992.
[73] P. Kouvelis and M.A. Lariviere, "Decentralizing Cross-Functional Decisions: Coordination through Internal Markets," *Management Science*, vol. 46, no. 8, pp. 1049-1058, 2000.
[74] M.B. Pinto, J.K. Pinto, and J.E. Prescott, "Antecedents and Consequences of Project Team Cross-Functional Cooperation," *Management Science*, vol. 39, no. 10, pp. 1281-1297, 1993.
[75] M.E. Sosa, S.D. Eppinger, and C.M. Rowles, "The Misalignment of Product Architecture and Organizational Structure in Complex Product Development," *Management Science*, vol. 50, no. 12, pp. 1674-1689, 2004.
[76] J.W. Rivkin and N. Siggelkow, "Balancing Search and Stability: Interdependencies among Elements of Organizational Design," *Management Science*, vol. 49, no. 3, pp. 290-311, 2003.
[77] A. Donnellon, "Crossfunctional Teams in Product Development: Accommodating the Structure to the Process," *J. Product Innovation Management*, vol. 10, no. 5, pp. 377-392, 1993.
[78] C. Christensen, "The Rules of Innovation," *The Human Side of Managing Technological Innovation: A Collection of Readings*, R. Katz, ed., pp. 622-627, Oxford Univ. Press, 2003.
[79] R.K. Moenaert, F. Caelddries, A. Lievens, and E. Wauters, "Communication Flows in International Product Innovation Teams," *J. Product Innovation Management*, vol. 17, no. 5, pp. 360-377, 2000.
[80] W. Orlikowski, "Learning from Notes: Organizational Issues in Groupware Implementation," *Proc. ACM Conf. Computer-Supported Cooperative Work (CSCW '92)*, pp. 362-369, 1992.
[81] L.R. Daft and R.H. Lengel, "Organizational Information Requirements, Media Richness and Structural Design," *Management Science*, vol. 32, no. 5, pp. 554-571, 1986.
[82] J.R. Carlson and R.W. Zmud, "Channel Expansion Theory and the Experiential Nature of Media Richness Perceptions," *Academy of Management J.*, vol. 42, no. 2, pp. 153-170, 1999.
[83] L.-T. Cheng, C.R.B. Desouza, S. Hupfer, J. Patterson, and S. Ross, "Edit, Compile, Run, Debug, Collaborate? Building Collaboration in to IDEs," *ACM Queue*, vol. 1, no. 9, pp. 42-50, 2004.
[84] J.G. Davis, E. Subrahmanian, S. Konda, H. Granger, M. Collins, and A.W. Westerberg, "Creating Shared Information Spaces to Support Collaborative Design Work," *Information Systems Frontiers*, vol. 3, no. 3, pp. 377-392, 2001.
[85] R.E. Kraut, C. Egido, and J.R. Galegher, "Patterns of Contact and Communication in Scientific Research Collaborations," *Intellectual Teamwork: Social and Technological Foundations of Cooperative Work*, J.R. Galegher, R.E. Kraut, and C. Egido, eds., pp. 149-172, Lawrence Erlbaum Assoc., 1990.
[86] M. Kendall and J.D. Gibbons, *Rank Correlation Methods*. Edward Arnold, 1990.
[87] L. Osterweil, "Software Processes Are Software Too," *Proc. Ninth Int'l Conf. Software Eng.*, pp. 540-548, 1997.
[88] J. Iivari and P. Kerola, "A Sociocybernetic Framework for the Feature Analysis of Information Systems Design Methodologies," *Information Systems Design Methodologies: A Feature Analysis*, T.W. Olle, H.G. Sol, and C.J. Tully, eds., North-Holland, 1983.
[89] R. Kling and W. Scacchi, "Computing as Social Action: The Social Dynamics of Computing in Complex Organizations," *Advances in Computers*, vol. 19, pp. 249-327, 1980.
[90] D. Robey and M. Newman, "Sequential Patterns in Information Systems Development: An Application of a Social Process Model," *ACM Trans. Information Systems*, vol. 14, no. 1, pp. 30-63, 1996.
[91] M. Deck, "Managing Process Diversity While Improving Your Practices," *IEEE Software*, vol. 18, no. 3, pp. 21-27, May/June 2001.
[92] I. Aaen, "Software Process Improvement: Blueprints versus Recipes," *IEEE Software*, vol. 20, no. 5, pp. 86-93, Sept./Oct. 2003.
[93] D.E. Harter, M.S. Krishnan, and S.A. Slaughter, "Effects of Process Maturity on Quality, Cycle Time, and Effort in Software Product Development," *Management Science*, vol. 46, no. 4, pp. 451-466, 2000.

Narayan Ramasubbu received the bachelor of engineering degree from Bharathiar University, India, and the PhD degree from the University of Michigan, Ann Arbor. He is an assistant professor at the Katz Graduate School of Business at the University of Pittsburgh. Previously, he was an assistant professor at the School of Information Systems at the Singapore Management University. Prior to his academic career, he was a senior developer at SAP AG and CGI Inc. His current research interests include software engineering economics with a focus on globally distributed product development and service delivery; the design, implementation, and governance of enterprise information systems; end-user interaction and user-led innovation; and IT strategy and generating business value from IT. He is a member of the IEEE.