Published online ahead of print March 25, 2009

Information Systems Research
Articles in Advance, pp. 1–23
ISSN 1047-7047 | EISSN 1526-5536

informs®

DOI 10.1287/isre.1080.0205
© 2009 INFORMS

Research Note

The Role of Organizational Controls and Boundary Spanning in Software Development Outsourcing: Implications for Project Performance

Anandasivam Gopal
Robert H. Smith School of Business, University of Maryland, College Park, Maryland 20910, agopal@rhsmith.umd.edu

Sanjay Gosain
Capital Group Companies, Inc., Irvine, California 92618, gosain@gmail.com

Past research has studied how the selection and use of control portfolios in software projects is based on environmental and task characteristics. However, little research has examined the consequences of control mode choices on project performance. This paper reports on a study that addresses this issue in the context of outsourced software projects. In addition, we propose that boundary-spanning activities between the vendor and the client enable knowledge sharing across organizational and knowledge domain boundaries. This is expected to lead to facilitation of control through specific incentives and performance norms that are suited to client needs as well as the vendor context. Therefore, we argue that boundary spanning between the vendor and client moderates the relationship between formal controls instituted by the vendor on the development team and project performance. We also hypothesize the effect of collaboration as a clan control on project performance. We examine project performance in terms of software quality and project efficiency. The research model is empirically tested in the Indian software industry setting on a sample of 96 projects. The results suggest that formal and informal control modes have a significant impact on software project outcomes, but need to be finely tuned and directed toward appropriate objectives. In addition, boundary-spanning activities significantly improve the effectiveness of formal controls. Finally, we find that collaborative culture has provided mixed benefits by enhancing quality but reducing efficiency.

Key words: software outsourcing; organizational control; software quality; project overruns; boundary spanning; partial least squares; surveys
History: Vallabh Sambamurthy, Senior Editor; Shaila Miranda, Associate Editor. This paper was received on September 8, 2006, and was with the authors 10 months for 5 revisions. Published online in *Articles in Advance*.

1. Introduction

Software development outsourcing is a multifaceted and complex activity in which clients and vendors interact in many different ways to produce and deliver the software services required. Most outsourced software projects involve significant technical activities combined with a social process of acquiring and integrating knowledge from various stakeholders such as users, project managers, developers, and clients. In such a context, appropriate organizational controls are vital in reconciling the interests of stakeholders and improving project performance. Much of the recent work in the information systems discipline related to organizational controls has focused on control portfolios in projects and the role of various technical and organizational factors in determining the choice of control modes (Kirsch 1996, 1997). However, there is little work that has explicitly studied the effect of the controls on project performance. Indeed, a recent meta-analysis (Narayanaswamy et al. 2007, p. 2) points out that the "focus has been on identifying the factors that lead to the choice of controls and not the results that come from the use of controls. Even when performance outcomes are considered, the relationship to the use of various forms of control is often not clear." This study tries to address this gap in the literature by studying the effect of organizational controls on project performance.

Although organizational controls have been studied in some prior research, a significant proportion of this work has been limited to in-house projects, wherein the controller and the controllee operate within organizational boundaries. Relatively less work has examined controls in software outsourcing, particularly in relating controls to project outcomes in outsourced projects. In studying the impact of controls in the context of outsourcing, two differentiating factors from the in-house context need to be considered. First, there are two organizational interfaces that appear in outsourced projects. The first interface lies between the vendor project management and the vendor development team. The second interfaces lies at the organizational boundary between vendor and client organizations. Although prior research on controls in outsourcing hints at these multiple interfaces (Choudhury and Sabherwal 2003, Kirsch et al. 2002), they are not explicitly separated in theoretical treatment. The second difference from prior research on control is in recognizing that vendors and clients in outsourcing arrangements may have radically different knowledge domains and expertise (Levina 2005). Vendors have strong technical knowledge but typically lack business domain knowledge, whereas clients have strong business domain knowledge but not deep technical knowledge and do not directly engage with technical artifacts. Therefore, to successfully manage an outsourced project, it is important for both parties to collaborate on sharing this knowledge across knowledge boundaries, thereby necessitating boundary spanning (Carlile 2002). The vendor's ability to control the project is facilitated by the extent to which it is informed by the client on client expectations, knowledge, and input (Levina and Ross 2003).[1] Explicating the role of boundary spanning across client-vendor organizational interface in shaping the effectiveness of controls applied at the management-vendor team interface is an important contribution of our study.

Boundary spanning has emerged as an important concern in the context of knowledge sharing across fields of work or *practice* (Carlile 2002). Fields of practice refer to concentrated collections of knowledge in a given context that enables agents to specialize in them while also distinguishing them from other fields (Levina and Vaast 2005). In outsourced projects, it is necessary to not only span geographical and cultural boundaries, but also the more relevant boundary of knowledge domains. We propose that boundary spanning plays a particularly significant role in enabling the vendor organization to apply control more effectively to manage the development team. The extent of boundary spanning between the vendor and the client allows the vendor to institute controls that are more effective for the specific context and to effectively leverage relevant control parameters. The distinction between the two organizational interfaces inherent in outsourcing is thus captured in our research model; we show that boundary spanning *between* vendor and client significantly moderate the relationship between formal controls and project performance *within* the vendor organization. This approach bridges prior work studying controls in in-house development (Kirsch 1996, 1997) and boundary spanning in software outsourcing (Levina and Vaast 2005, Levina 2005) into one parsimonious model of project performance.

Our work in this paper makes *three* significant contributions. *First*, we study the role of formal controls instituted within the vendor organization on project performance in the understudied, but increasingly prevalent, context of software outsourcing, thereby extending Kirsch's (2004) and Choudhury and Sabherwal's (2003) work. Our work helps us understand the implications of organizational controls in the IT services outsourcing industry, estimated at about \$78 billion in 2007 by TPI,[2] of which cross-functional application development (specifically our focus) represented about roughly \$35 billion. *Second*,

[1] Levina and Ross (2003) call the former the vendor's *methodology competence*, which is influenced by the vendor's *relationship management competence*, which encompasses our notion of boundary spanning in this paper. We are grateful to a reviewer for this point.

[2] http://www.tpi.net/newsevents/news/releases/080116-TPI.html.

we explicitly include boundary spanning as a moderator into the controls-based model of software development (Kirsch 1996, Choudhury and Sabherwal 2003). We propose a parsimonious measure of boundary spanning in the outsourcing context, drawing from existing qualitative research on boundary spanning and knowledge transfer (Levina and Vaast 2005). Our *third* contribution is to study organizational controls as an antecedent to project performance rather than explaining the choice of controls. Existing studies have mainly used the case study method to identify the choice of control portfolios on a small set of projects (Kirsch 1996). Although this approach provides a rich dataset of microlevel interactions between the development team and the user community, there are few clear linkages made to project performance (Narayanaswamy et al. 2007). In this paper, we not only show that outcome-based controls have positive impacts on the focal outcomes, but also show that these controls may impose negative externalities on other outcomes.

In outsourced projects, the most commonly observed performance parameters are software quality and team efficiency. Quality, i.e., *product performance*, is a key outcome of software development activities (Henderson and Lee 1992). Additionally, adherence to budgeted cost and schedule, i.e., *process performance*, represents an important outcome measure (Wallace et al. 2004). Our paper provides a quantitative analysis of the link between controls and product and process performance and also enables us to tease out which type of controls have, on the margin, a greater impact on performance. We tested our hypotheses on data from a sample of 96 offshore software projects, collected using a questionnaire and company databases, from software-outsourcing vendors in India. The Indian software industry is a leading outsourcing destination for firms across the world (Meyer 2006) and forms an appropriate context to study outsourcing. Our results confirm that boundary spanning between the client and the vendor significantly moderates the efficacy of formal controls instituted within the vendor organization to direct the project team. In addition, we find support for the observations made in prior research about the dominance of outcome controls over behavior controls—outcome controls in this setting are associated with better project outcomes than behavior controls.

2. Background Theory and Hypotheses

The research model proposed integrates a basic framework of organizational control theory with a boundary-spanning perspective that draws on the knowledge-based view of firms. In the following section we review the conceptual bases of this study and position its contributions in relation to the extant literature, followed by the development of the hypotheses to be tested.

2.1. Organizational Controls

Our research model is rooted in organizational control theory, which posits that four main modes of control may be used in managing economic activity—behavior-based controls, outcome-based controls, clan-based controls, and self-control (Ouchi 1979, Eisenhardt 1985). Outcome controls and behavior controls are classified as *formal* control modes because they can be purposefully instituted based on the underlying needs of the task at hand. In outcome-based controls, the controller specifies the parameters of the desired outcome; the controllee's rewards are based on the observed outcome. In contrast, behavior-based controls are used when specific rules and procedures are established for the controllee to follow, thereby leading to the desired outcome. Controllers observe the behavior of the controllees and reward them on the basis of their adherence to such rules and procedures.

In contrast to formal control, *informal* controls, i.e., clan control and self-control, are based on social strategies that stress interpersonal and individualistic dynamics. Clan controls are implemented by promoting a set of common values and beliefs within the organization, such that the agent's desire to be identified as a valid member of the "clan" induces desired behavior. Clan controls include undocumented but socially accepted methods of activity, informal codes of conduct with respect to vendor-client relationships, and accepted behaviors that facilitate desired working conditions within the firm (Ouchi 1979). Managers can strategically induce desired behavior or outcomes by introducing socialization processes and normative methodologies within the organization. However, not all clan controls may be consistent with organizational goals, hence, considering clan controls at an abstract level may not help understand links to performance. Our work in this paper considers a specific type of clan control, *collaborative culture*, which refers to a value-based system emphasizing shared purpose in working towards common goals.

Although extant literature also points to self-control, allowing for individual discretion and intrinsic motivation as providing a basis for managing organizations (Eisenhardt 1985), they are less of a strategic tool in the manager's toolkit, being difficult to accurately assess and influence.

The study of controls in software development, emerging from Henderson and Lee (1992), has been developed in the Informaton Systems literature to the point where a significant body of work exists. Kirsch (1996, 1997) pioneered the study of control in IS development by analyzing the choice of control portfolios in in-house projects based on controllee behavior observability and outcome measurability. In subsequent work, she extends her analysis to understand how dimensions of control portfolios change over the course of the project (Kirsch 2004). In much of this literature, the controller is the IS manager/user contact while the controllee is the development team. Extending this literature to the outsourcing domain requires cognizance of the existence of two interfaces—one between the vendor manager and the vendor team and the other between the client and the vendor. In their analysis of controls in outsourcing, Choudhury and Sabherwal (2003) refer to individuals in the client organization as controllers and individuals on the vendor site as controllees (p. 292), but do not explicitly distinguish between interactions occurring at the vendor-client interface and those at the project manager-project team interface. This represents a gap in the literature.

Although the focus on much of the controls literature in IS research has been on establishing the choice of control portfolios, there is little by way of establishing the impact of chosen controls on project performance (Narayanaswamy et al. 2007). Behavioral controls can be likened to software processes in that both require adherence to a set of prescribed activities and methods to develop and maintain software (Paulk et al. 1993, Kirsch 1996). Thus, some of the existing research of the positive effect of processes on outcomes (Krishnan et al. 2000, Gopal et al. 2002a) could be seen as indicative of the positive effects of behavioral control. Similarly, some existing work that attests to the beneficial impacts of strong incentives on vendors to achieve quality and efficiency targets is suggestive of the positive effects of outcome-based controls (Gopal et al. 2003, Banerjee and Duflo 2000). However, the aggregate effects of these different forms of controls have not been studied together in a holistic manner in the outsourcing domain. It is possible that they have differing effects on dimensions of performance when evaluated together—this represents another gap in the literature.

The above discussion suggests three unexplored avenues in the IS literature on controls. First, much of the current work pertains to in-house software development, with the exception of Choudhury and Sabherwal (2003). The increasing prevalence of outsourcing would indicate a need to extend the study of controls to outsourced projects. Second, most existing work has studied the performance implications of software processes and incentives in isolation; a more holistic approach to evaluating their performance implications within an organizational control framework could provide additional insights. Third, extending controls literature to outsourcing requires recognition of the two interfaces that exist in most software-outsourcing engagements. We augment the conceptual model of control in view of these gaps in the literature by incorporating the impact of the interactions between the client and the vendor, which we treat as manifestations of *boundary spanning*, on the performance implications of different types of controls examined together.

2.2. Boundary Spanning

Software development is a knowledge-intensive activity wherein firms' competitive advantage emerges from their unique ability to recombine individual and organizational knowledge (Kogut and Zander 1992). The effectiveness with which firms can collate, share, and build on individual knowledge bases depends on the scope and the flexibility of the knowledge integration activities (Grant 1996). The firm's integrative abilities need to combine the appropriate forms of knowledge with the suitable vehicles for knowledge sharing. Thus, successful firms can create and manage inimitable and flexible models for knowledge sharing.

In software outsourcing, the vendor is a repository of both declarative and procedural technical knowledge. This typically includes understanding of the base platforms, enterprise-level software applications, operating systems, and the user-facing applications that will be developed according to specifications (Pressman 2001). However, the vendor requires extensive knowledge about the business domain that the system will support (and be embedded in) from the client. Traditional requirements analyses are limited in the extent to which they can capture the true complexity of the business domain that the clients operate in Wallace et al. (2004). Furthermore, requirements may also change or evolve over time. The client's business domain contains large amounts of tacit knowledge that may not be adequately captured in the declarative knowledge elements traditionally used, such as a priori functional specifications. Thus, a large part of the typical outsourcing relationship revolves around knowledge integration activities between the vendor and client, aimed at bridging the gap between their respective knowledge domains. These activities are thus reflective of *boundary spanning* as an organizational capability (Carlile 2002).

Empirical work studying boundary-spanning activities has focused on three important facets of boundary spanning—boundary *spanners*, boundary *objects*, and boundary-spanning *processes* (Star 1989, Carlile 2002). Boundary spanners refer to individuals who are responsible for ensuring that the required knowledge is able to flow across the boundaries. Spanners "facilitate the sharing of expertise by linking two or more groups of people separated by hierarchy, location, or function" (Levina and Vaast 2005). Boundary spanners in practice may be nominated or may emerge without nomination (Levina and Vaast 2005).

In addition to spanners, boundary objects that can be used to share knowledge are needed. Star (1989) defines boundary objects as "objects that are plastic enough to adapt to local needs and constraints of several parties using them yet robust enough to maintain a common identity across sites." Boundary objects include physical prototypes, accounting ledgers, design documents, software, and engineering sketches (Levina and Vaast 2005). In the IT context, boundary objects can refer to document archives, software code, and design artifacts (Pawlowski and Robey 2004, Levina 2005). Effective boundary objects should establish a shared syntax between parties, provide a concrete means for individuals to specify and learn about their differences and represent these differences in the object itself for better understanding. Like boundary spanners, it is possible to differentiate between designated boundary objects from actual boundary objects in use (Levina and Vaast 2005).

Finally, it is important to establish a process by which individuals can jointly transform their knowledge or share knowledge. Carlile (2002, 2004) describes the need for a process using which members across knowledge domains can share information bidirectionally to integrate, transform, or apply existing knowledge to the economic activity at hand. The presence of a boundary-spanning process also reduces the cognitive costs on individuals by establishing a common syntax for the boundary-spanning activities. Our definition of boundary spanning in the outsourcing context thus draws from these three building blocks of boundary spanning—spanners, objects, and processes.

2.3. Software-Outsourcing Project Performance

Software project performance has been broadly characterized in terms of effectiveness and efficiency (Hoegl and Gemuenden 2001, Wallace et al. 2004). The effectiveness dimension assesses the degree to which the developed software meets the requirements of the customer, and is also referred to as product performance (Henderson and Lee 1992). This construct captures the quality-specific attributes of the developed software. One approach views software development as indicative of a service with certain properties provided to clients. Wallace et al. (2004) use five Likert-scaled items to capture this attribute with the focus being on reliability, maintainability, meeting requirements, and response time of the application developed. An alternative line of research attempts to capture this information through objective data such as defects per line of code (LOC) (Ethiraj et al. 2005) or modification requests (Herbsleb and Mockus 2003).

In contrast, the efficiency dimension captures the extent to which the development process is well managed, i.e., process performance (Hoegl and Gemuenden 2001). The construct addresses whether the software was developed on time and within budget. Past research has studied efficiency by focusing on project effort and cycle times (Harter et al. 2000), project profitability (Gopal et al. 2003), and project costs (Krishnan et al. 2000). Cost and schedule overruns have been used to characterize efficiency (Ethiraj et al. 2005, Tiwana 2004) and allow for better comparisons across projects—we use reverse-coded overrun-based measures in assessing project efficiency.

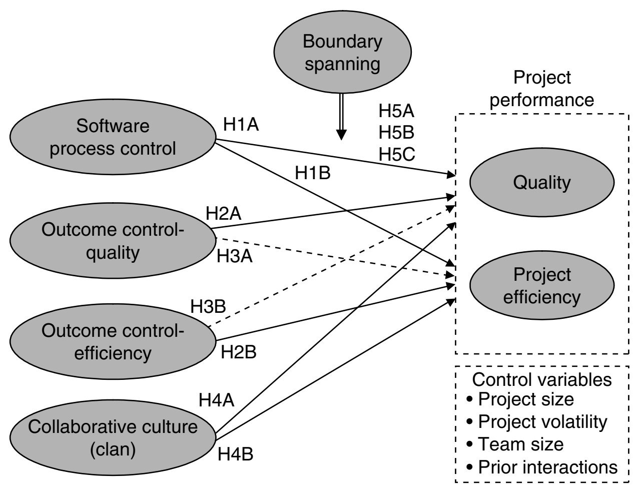
Although it is possible to study the antecedents of project performance along the two dimensions of performance individually, as seen in the literature, we argue that to fully understand the efficacy of control on performance, it is important to study both dimensions of project performance together. It is often easy to cut costs (and enhance efficiency) by underperforming on quality (Pressman 2001). Similarly, the practitioner press has discussed the prevalent practice of cutting cycle times and releasing software products with known quality issues to enhance efficiency (Thibodeau and Rosencrance 2002). In this paper, we thus study the effect of control on both performance dimensions together. The research hypotheses are discussed next.

2.4. Hypotheses Development

We first present hypotheses related to the effects of control and then moderation by boundary spanning, based on the research model (Figure 1).

Effect of Formal Controls. We first address our hypotheses pertaining to behavioral control and then describe those with respect to outcome controls.

Figure 1 Empirical Research Model



Behavioral control involves managers prescribing appropriate procedures that have to be followed by agents in carrying out their activities. The reasoning for instituting such control is that adherence to prescribed behavior norms will lead, under the right contingencies, to higher performance. Absent such behavior control, instrumental behaviors leading to desired outcomes may not be apparent to individual participants; the presence of such control leads to more focused efforts and performance.

In the outsourcing context, appropriate behavior controls can include development methodology specification (Necco et al. 1987, Choudhury and Sabherwal 2003), clearly defined procedures, and documentation-related behaviors (Kirsch 1997). This reasoning is also reflected in software process management in software organizations, wherein process initiatives lay out disciplined and metrics-based methodologies that are required to be followed in all projects. Research has shown that controlled and disciplined processes, analogous to strict behavior controls, have resulted in higher quality for in-house software products (Krishnan et al. 2000). We extend this reasoning to the outsourcing domain and test for similar effects of behavioral control on project outcomes. To the extent that adherence to process and methodologies within the project team on the vendor side are associated with better quality, we propose

Hypothesis 1A (H1A). *Higher levels of behavioral (process specifications-based) control will be associated with higher levels of software quality.*

Software development is an uncertain activity with a high degree of risk surrounding the causal relationship between input and output (Pressman 2001). Therefore, it is possible to have all requisite inputs into the development process and still experience technical or managerial issues during the project that adversely affect the ability to complete the project on time or within budgeted costs. Behavioral controls aim to mitigate this inefficiency by encapsulating appropriate planning templates and development practices most suited to the industry context. This has also been empirically shown to enhance the efficacy of the development process (Krishnan et al. 2000, Gopal et al. 2002a). Indeed, the prevalent process models such as the ISO 9000 and the CMM are collections of best practices culled from experience (Paulk et al. 1993). Hence, we propose that,

Hypothesis 1B (H1B). *Higher levels of behavioral (process specifications-based) control will be associated with higher levels of project efficiency.*

Organizations implementing outcome control specify desired goals, and employees are rewarded based on the extent to which they meet these goals. A cybernetic view of control suggests that individuals and teams would be motivated to take corrective action if they failed to meet project goals on an ongoing basis, thereby leading to higher output quality and efficiency (Jaworski 1988). Alternatively, a behavioral view of outcome control indicates that once the controllee has been provided with information about desired outcomes, his/her incentives should ensure appropriate behavior to meet these goals (Kirsch 1997). This view is supported by agency theory, wherein the presence of incentive alignment between the principal and agent leads to better outcomes by enabling the appropriate agent behavior (Jensen and Meckling 1976).

In software outsourcing, appropriate outcomes used in predicating agent control or rewards include software quality. It is possible for vendor managers to stipulate quality benchmarks for the project, and hence incentivize appropriate actions from the development team. Anecdotal evidence from the Indian outsourcing context indicates that this is often the case (Thibodeau 2004). Software project participants can then be rewarded or sanctioned based on their individual or team's performance in meeting these goals. In the presence of established outcome control and the resulting incentives, the output from the project should, correspondingly, be of higher quality: Hence,

Hypothesis 2A (H2A). *Higher levels of quality-based outcome control will be associated with higher levels of software quality.*

Agents may be incentivized along different dimensions in their activities (Jensen and Meckling 1976, Allen and Lueck 1999). Certain projects can be heavily incentivized along quality for mission-critical applications, whereas for other projects, time to market or time to delivery are more critical. Therefore, the dimension along which the outcome control is deployed will also play a role in determining outcomes, whereas the other dimension of performance should not be impacted. Efficiency-based controls are often applied in offshore settings (Thibodeau 2004). Research has shown that appropriate incentives for developers can lead to motivation of developer performance in software developer settings (Henderson and Lee 1992). Therefore, we expect that variance in the level of efficiency-based outcome controls will be associated with varying levels of efficiency.

Hypothesis 2B (H2B). *Higher levels of efficiency-based outcome control will be associated with higher levels of project efficiency.*

An interesting but unexplored implication of outcome-based controls that we consider in this study is the effect of one type of outcome-based control on the other dimension of performance.[3] Specifically, we consider the spillover effects of *quality*-based outcome controls on *efficiency* and vice versa. There is little in the controls literature that has explicitly addressed these dynamics, although Choudhury and Sabherwal (2003) allude to these effects. There are two lines of research in the literature to suggest that the externalities of instituted outcome-based controls may negatively affect other outcomes. The first line of reasoning is based on the attention-based view of managerial action (Ocasio 1997). The fundamental premise of the attention-based theory is that managers' actions and behaviors depend on the issues that trigger their attention. What issues draw such attention as well as the consequent actions by the manager is a function of the context in which the manager is embedded (Ocasio 1997) and the limited information and capacities available to him/her. This perspective highlights the managerial tendency to focus on some limited aspects of a problem, possibly to the detriment of other considerations (March and Shapira 1987). Thus, a focus of managerial attention on quality as an outcome and the resulting quality-based controls could lead to inadequate attention (whether intentionally or otherwise) paid to efficiency parameters on the project. This would imply a negative effect of quality-based outcome control on efficiency and vice versa.

[3] We thank the AE for bringing this unexplored avenue of analysis to our attention.

The second line of reasoning emerges from software engineering, wherein researchers have always addressed the dichotomy between quality and efficiency as outcomes. Krishnan et al. (2000) suggest that software processes that strongly focus on quality outcomes often lead to higher overruns and detrimental impact on productivity metrics because of uncertainty and causal ambiguity in the software improvement efforts. Similar arguments are made in other work (cf. Harter et al. 2000 and Westland 2004). On the basis of this reasoning, we postulate the following hypothesis:

Hypothesis 3A (H3A). *Higher levels of quality-based outcome control will be associated with lower levels of project efficiency.*

A similar negative externality could be found in the effects of efficiency-based outcome controls on quality. Indeed, it has been argued that strong incentives to reduce costs in a project could allow the formation of perverse incentives to underprovide on quality (Westland 2004). From a practitioner viewpoint, it is often observed that interventions that seek to minimize cycle time or duration on a project (indicative of project efficiency) often result in code that underperforms on quality (Thibodeau and Rosencrance 2002, Pressman 2001). Therefore, it is likely that outcome-based controls that focus on and incentivize team activities around quality might result in compromising efficiency. Thus, we propose:

Hypothesis 3B (H3B). *Higher levels of efficiency-based outcome control will be associated with lower levels of software quality.*

Effect of Informal Control. Informal or clan-based controls operate through social norms such that the controlled entities internalize organizational goals and act in a manner consistent with the controlling entities expectations (Covaleski et al. 1998, Ouchi 1979). Clan control operates when all members of the work group embrace the same values, adopt similar problem-solving approaches, and commit to achieving group goals (Ouchi 1979). Although it is difficult to study and observe clan controls directly because they are deeply embedded in the minds and practices of participants, a *collaborative* team culture can be seen as an important specific manifestation of clan control. Collaborative culture leads members to freely share resources and ideas with others. Because the success of work conducted in software teams depends on how well team members collaborate (Hoegl and Gemuenden 2001), a collaborative team culture will facilitate the achievement of project objectives. Because of likely heterogeneity in skills and expertise within a project team, with collaborative exchanges, the team can leverage the competence of each member in their area of expertise and bring it to bear on relevant task activities. Also, in a collaborative culture, team members will be less defensive about exposing their individual work outcomes to scrutiny by others in the team. This is expected to reduce defects and improve software quality. Thus, we propose:

Hypothesis 4A (H4A). *Higher levels of collaborative culture-based clan controls will be associated with higher levels of software quality.*

Because of mutual expectations of helpful behaviors, enabling individuals to be more open about their deficiencies and get help from others as needed, collaborative culture can reduce the need for costly rework. Also, helpful behaviors are expected to eliminate situations where the team may be suboptimal in leveraging its resources because individuals may act in their own interest and hoard resources or not respond in a timely manner to requests from others (Guzzo and Dickson 1996). A collaborative culture helps by mitigating these tendencies and fostering mutually beneficial joint work, thereby enhancing efficiency. Therefore,

Hypothesis 4B (H4B). *Higher levels of collaborative culture-based clan controls will be associated with higher levels of project efficiency.*

Moderating Influence of Boundary Spanning. Client organizations have knowledge of the business domain relevant to the project. This knowledge is marshalled through a requirements definition process that converts the requirements to functional specifications. However, the process of acquiring the specific knowledge, needed for creating functional specifications, is often inefficient and open to contingencies that unfold over time (Pressman 2001, Thibodeau and Rosencrance 2002). Therefore, even after requirements elicitation, continued dialogue with the client is necessary to address open issues about the business domain for vendor managers (Gopal et al. 2002a). From the vendor side, vendors have deep technical knowledge required for making decisions about technology choices for the project. In most cases, the relevant knowledge about the technological details have to be shared with the client to make informed decisions throughout the project. Thus, there is clearly a need for sustained bidirectional knowledge sharing across respective knowledge domains of the client and the vendor. This requires the presence of specific persons on the vendor team that are responsible for knowledge sharing between vendor and client (Gopal et al. 2002a) and who can act as a conduit for this knowledge to percolate down to the development team. Thus, the importance of the boundary spanner in this role is crucial to the process of knowledge sharing.

There are significant benefits to knowledge sharing carried out through boundary spanning. Important project-level information necessary for successful execution of the project is shared between the vendor and the client. In addition to these direct gains, which are not the primary focus of this study, knowledge integration enables vendor managers to create facilitating conditions that increase the efficacy of applied controls. This includes *operationalizing* the control parameters for the specific project context and *instantiating* controls to the specific context of the project. The access to knowledge about the project's business domain (and the resulting knowledge integration) will better inform vendor management about critical aspects of the project from the client's viewpoint. In addition, this interaction will also provide vendor management with an enhanced understanding of both the technical issues surrounding the project as well as the touch points between the technical and the business aspects. For instance, the use of a prototype could deepen the client's appreciation for exactly how the proposed system can enable existing business processes or, in some cases, lead to enhanced business processes that takes advantage of the new technology (Pressman 2001). This enhanced understanding of client requirements will enable vendor managers to translate this knowledge into a more fine-tuned set of control parameters and facilitating conditions that can be applied to the development team. These artifacts (prototype code, design documents), viewed through the lens of boundary spanning objects, are highly instrumental in enabling the fine-tuning of behavioral control because they enable higher-order knowledge to be shared between clients and vendors, particularly in cases where there might be significant gaps in understanding between the vendor's view of the desired system and that of the client (Pressman 2001).

With behavioral control, enhanced knowledge sharing from boundary spanning could result in more appropriate methodologies and procedures applied to the team. Boundary spanning activities enhance communication and coordination between the vendor and the client, which leads to quicker resolution of open issues (Gopal et al. 2002a) as well as allows for adjustments to the behavioral specifications that are instituted as the context evolves (Kirsch 2004, Krishnan et al. 2000). Boundary spanning will also allow control to be calibrated to conditions of the project's context and given the greater knowledge sharing, such control will be less likely to be seen as imposed without basis or understanding (Levina 2005). Therefore, we propose

Hypothesis 5A (H5A). *Boundary-spanning activities between the vendor and the client will positively moderate the relationship between behavioral control and project performance.*

In the case of outcome controls, the importance of client input and knowledge in determining the right set of requirements has already been established (Pressman 2001, Kirsch et al. 2002). In addition, boundary spanning provides vendor managers with two additional benefits. First, it helps vendor management choose the right outcomes in the project and link these outcomes to the appropriate business-level metrics required for the project. In many cases, the outcome benchmark changes over the course of the project as more knowledge between the vendor and the client is shared and a clearer shared mental model of the required software emerges (Levina 2005, Choudhury and Sabherwal 2003). This process requires not only the use of appropriate boundary objects, but also the presence of a boundary-spanning process, particularly in the case of outcome controls. Without systematic and consistent feedback based on an established process, it would be hard to reach a shared mental model and thereby apply the appropriate metrics. Second, it enables the vendor to create the appropriate outcome control that reflects the trade-off between quality and efficiency (or cost) in the project.

Boundary spanning between the vendor and the client team would help ensure that when outcome controls based on quality are applied, they are appropriate to the requirements of the client. Quality is a multidimensional attribute, and the client may value different attributes to a varying extent. Therefore, knowledge gained through boundary spanning would ensure that the appropriate mix of attributes is identified and that outcome controls direct incentives towards the appropriate outcomes valued by the client. The use of outcomes controls is thus enhanced by the use of appropriate boundary objects that emphasize the right attributes and thereby apply the right incentives. For instance, prototypes can be used within such situations to great effect because they can be used to convey not only what the client requirements are, but also to clarify where design-level trade-offs can be made to enhance client value. Because much of outsourced development tends to be based on specific client requirements, unlike products where the requirements are more general, the ability to fine-tune quality aspects to the specific expectations of the client and the business domain are critical. This capability is enhanced through boundary spanning. Thus, we propose the following.

Hypothesis 5B (H5B). *Boundary-spanning activities between the vendor and the client will positively moderate the relationship between quality-based outcome control and software quality.*

In addition to quality, boundary-spanning activities between the vendor and the client would similarly ensure that the specific efficiency parameters valued by the client are identified and incentives are directed towards achieving appropriate cost or duration targets. In some cases, the client may prioritize a set of intermediate milestones that need to be met. Appropriate boundary-spanning activities would ensure that outcome controls are specifically directed and operationalized to match these needs. Within the gamut of boundary spanning, the importance of the appropriate boundary-spanning processes are highlighted here; the presence of a process that conveys the right level of knowledge about project progress using efficiency metrics (such as cycle times, costs, and project management information) will significantly enhance the ability of the vendor to leverage efficiency-based controls. Hence, we propose that:

Hypothesis 5C (H5C). *Boundary-spanning activities between the vendor and the client will positively moderate the relationship between efficiency-based outcome control and project efficiency.*

The concept of boundary spanning can be seen as being determined by (or formed from) the individual elements (process, boundary-spanning roles, and boundary objects) without any assumptions as to the patterns of intercorrelation between these elements. Prior research has shown that boundary-spanning processes may exist by themselves to add value to a project (Levina 2005). In addition, the software engineering literature has shown that the use of prototypes and code-related artifacts enhance the output on software development projects (Gopal et al. 2002b, Choudhury and Sabherwal 2003), without there necessarily being a boundary spanner or process in place. As Levina and Vaast (2005) propose, an undesignated boundary spanner may arise in projects when there are no processes in place. Although each of these individual boundary-spanning components add value, it is not necessary that they should coexist in all cases of boundary spanning. In effect, we argue that boundary spanning can be viewed as a *formative* construct, with each component adding to the boundary-spanning capability. The presence of all three components will significantly enhance the extent to which an outsourcing engagement can gain from knowledge sharing between vendor and client.

Turning to informal controls, there is little leeway to reorient organizational values over the duration of a project. Research has shown that informal controls are relatively "sticky" and more difficult to strategically influence (Eisenhardt 1985). Thus, these controls may not be specifically conditioned to the project context and may not operate differently under different levels of boundary spanning.[4] Our research model

[4] We do, however, estimate a PLS model incorporating interactions between collaborative clan values and boundary spanning. We see no significant results, in individual coefficients or increases in R^2.

controls for variables such as project size, team size, prior interactions, and project volatility that important antecedents of project performance in past research (Harter et al. 2000, Ethiraj et al. 2005). Including these variables allows us to estimate the specific and relative contribution of organizational controls to performance beyond those effects seen in the literature.

3. Research Design and Data Analysis

To capture information on modes of control and boundary spanning observed in software outsourcing, we felt that a field study in a single organization (cf. Ethiraj et al. 2005) may not capture adequate variance reflective of the industry. On the other hand, a broad-based survey targeted at the industry as a whole generally results in low response rates and inadequate control over the final sample. Therefore, we chose a focused survey methodology; data were collected from a small set of organizations using structured surveys administered personally onsite.

We chose to draw on Indian firms in collecting data for this study. From a database of about 600 software firms maintained by NASSCOM (an industry-level software services trade association), companies assessed at or above CMM Level 3 were selected. Limiting our respondent pool to CMM Level 3 companies had three significant advantages in this study. First, it allows us to reduce some heterogeneity in the sampling distribution. Although there were nearly 900 outsourced vendors in the Indian market identified by NASSCOM at the time of data collection, many of these companies were marginal players. Including them would necessitate the use of stringent controls to account for heterogeneity and would also enhance the risk of bias in the sample. Second, restricting the sample to Level 3 companies provided us with greater assurance that these firms would have incorporated the focal constructs in our study in some measure. In other words, the firms would have some level of formal control, boundary spanning, and informal controls in place. It is often the case that smaller and newer vendor firms do not incorporate the range of organizational controls, but function in a more ad hoc manner. Furthermore, firms at lower CMM levels do not have generally have stringent methods and systems to accurately report on variables of interest such as quality. Thirdly, the more mature organizations are typically also the ones with a number of different contemporaneous projects, and this allowed us to obtain data from different projects while controlling for organizational factors. The downside of this approach is in a possible range restriction issue because of our sampling framework, and this may reduce the generalizability of the findings.

In addition, we also test the moderation of boundary spanning and quality-based (efficiency-based) outcome controls on efficiency (quality). These results are insignificant as well. The detailed analysis results are available from the authors.

A total of 267 software companies met the criterion of Level 3 certification. From these 267 companies, 45 firms were randomly selected and senior management at these firms was contacted with a request to participate in the research study. Once they agreed to participate, the executives were requested to provide us with a list of projects completed in the previous six months. Recently concluded projects reduce the effect of recall bias. All projects chosen were outsourced in entirety to the vendor firm and were carried out by the vendor firm. The executives granted access to individual project managers on projects who provided the responses for each project. A total of 23 firms finally agreed to participate in our research, for a response rate of 51%. These firms had, on average, 3,200 employees, with the smallest firm having 450 employees at the time of data collection. To ensure coverage of a broad range of projects from each organization, we only retained projects from firms where we had at least four or more projects. Our final sample for this study consisted of a total of 96 projects from 10 firms. Table 1 provides a descriptive summary of our sample, showing the breakdown of the sample by client industry, software domain, type of project, and the technical platform used.

The questionnaire used for data collection was created by adapting scales from prior research. The objective data such as project-level metrics on project efficiency and project size were obtained from the company databases. The questionnaire was pretested in two ways. First, the questionnaire was presented to a set of quality auditors certified by the Software Engineering Institute (SEI). Their comments on the questionnaire were elicited and incorporated into the questionnaire. Second, a Web-based version of

Table 1 Sample Description

Project characteristic	Sample description
Software domain[1]	Application (65.1%)
	System (16.9%)
Client industry[2]	Biotech/healthcare (10.8%)
	Finance (8.4%)
	Information technology (34.9%)
	Automobile, transport & Logistics (10.8%)
	Manufacturing (8.4%)
	Retail (4.8%)
Project type[1]	New development (51.8%)
	Maintenance (19.3%)
	Reengineering (18.1%)
Platform[2]	Mainframe (16.9%)
	Unix variants (31.3%)
	Windows (30.1%)

Notes. $n = 96$.
[1]Remaining projects could not be clearly classified into one of these categories.
[2]Remaining projects belong to additional categories.

the questionnaire was created and a pilot survey was conducted using 15 software organizations (not in final sample). Respondents to the pilot were asked to fill out the survey and their feedback was incorporated into the final survey. The specific measures are described next.

Operationalization of Constructs.

1. *Software Quality*: The quality construct measures the extent to which the final software product conforms to client requirements. Because the primary focus of the outsourcing industry is to provide IT services and not a software product per se, we choose to measure quality subjectively. Although prior work has measured quality through defect rates per line of code (Krishnan et al. 2000) or modification requests (Herbsleb and Mockus 2003), in the outsourcing domain these measures may not capture the service-oriented, iterative nature of the interaction between the client and the vendor. Moreover, given the variety of applications and projects executed by vendors, there is no consistent definition of defects, lines of code, or indeed, technological platform across firms and projects.[5] Many firms in our sample did not track defect-rate metrics on their projects, possibly because of the limited ability of defect-rate data to capture product performance. Therefore, we created a measure of quality based on a taxonomy of software quality attributes proposed by Szejko (1999) to identify four important quality attributes required in the outsourcing context. We then adapted prior items measuring product performance (Tiwana 2004, Ravichandran and Rai 2000) to capture these four quality attributes. To verify the validity of our measure, we reran a short survey for software quality from the same respondents about 18 months after the original data collection. We received 20 complete responses. The new quality measures using our quality items were highly correlated with the old quality data for these projects ($r = 0.63$, $p < 0.01$). In addition, we collected defect density data (postrelease bugs per line of code) where available for a subset of projects from the largest category of projects in our sample (Windows platform-based development projects). Our subjective quality measure was significantly correlated with the objective defects measure of this subset of projects ($r = -0.498$, $p < 0.001$), providing strength to our quality measure.

2. *Project Efficiency*: Measured as the project's overruns on actual schedule, person-month effort, and cost overruns as a proportion of duration of the project in calendar weeks, the budgeted effort, and budgeted cost provided by the project managers. The overrun measures are multiplied by -1 (reverse coded) to create the measure such that numerically higher values represent greater project efficiency.

3. *Behavioral Control*: As discussed before, the conceptual model of behavioral control in software development matches well with the software process literature. We therefore use prior scales measuring behavioral process control (Kirsch 1996, Krishnan et al. 2000) and augment these with items pertaining to measurement in software projects (Gopal et al. 2002b). Eisenhardt (1985) observes that control consists of three dimensions: measurement, evaluation, and rewards. Consistent with this definition, we use six items that pertain to the presence of a process

[5] Much of the prior work that uses defect rates to measure quality is based on in-house IT development, where technologies, programming languages, and platforms are controlled for. Alternatively, these quality measures are useful in cases of software products (Krishnan et al. 2000), where bug rates are noted before the product ships.

methodology in the project and the extent to which the methodology was used in evaluating the progress in the project.

4. *Outcome Controls*: We focus on two kinds of outcome control—quality-based outcome and efficiency-based outcome controls, captured by two sets of items. Quality-based outcome control and efficiency-based outcome control were measured by five and six questionnaire items, respectively, adapted from prior work (Ravichandran and Rai 2000, Kirsch et al. 2002). Existing scales did not adequately capture all three dimensions of outcome control. Therefore, we adapted existing scales by adding dimensions of measurement and evaluation from Krishnan et al. (2000) and Gopal et al. (2002b) to the existing scales, as shown in Table A.1 in the appendix.

5. *Collaborative Culture*. Three questionnaire items were adapted from Hurley and Hult (1998) to measure the extent to which the project team had a collaborative culture.

6. *Boundary Spanning*. Our operationalization of boundary spanning captures the role of spanners, objects, and processes. Given the complexity of our research model and the sample size, we do not model boundary spanning as a second-order construct. Instead, we average measures reflecting each aspect and then treat the averages as formative indicators of boundary-spanning construct. We measure boundary spanners through two questionnaire items pertaining to whether formal and informal boundary spanners were appointed on the vendor side for the project. Objects were measured by the extent to which code inspections and design reviews were jointly conducted between the vendor and the client. Prior research in boundary spanning has established that code and design documents may become key boundary objects in use (Levina 2005, Levina and Vaast 2005). Finally, processes were measured by the extent to which project management meetings and status review meetings were held over the life of the project between the vendor and the client. Project management meetings and status review meetings often are forums in which client and vendor liaison share knowledge through the use of "boundary objects in use" such as design documents and code walkthroughs (Gopal et al. 2002a, Levina and Vaast 2005). These meetings are reflective of the extent to which a process was in place to share knowledge about the project across organizational and domain boundaries. We define boundary spanning as a formative construct comprising these three dimensions.

7. *Project Volatility*. This control variable is measured as a formative construct with three dimensions capturing requirements uncertainty, personnel turnover, and employee retention on the project.

8. *Prior Interactions*. Prior research has shown that prior interactions between client and vendor influences vendor capabilities, and hence project performance (Ethiraj et al. 2005). We model this control variable as a formative construct with two dimensions—number of projects completed and number of concurrent projects being executed for the same client.

9. *Project Size*. The project-size variable measures the total project effort in person-days. This variable exhibited some nonnormality. Therefore, consistent with previous work (Ethiraj et al. 2005), we use log of total effort in our analysis. Project size is an important control variable in most performance models in software development outsourcing; in general, larger projects are more difficult to manage, with implications for project performance (Pressman 2001).

10. *Team Size*. The size of the core team has been studied as a relevant control variable in outsourcing and software projects (Gopal et al. 2003, Wallace et al. 2004). Specifically, prior research has postulated that larger teams increase the communication overheads in projects and therefore could have a detrimental impact on outcomes (Guzzo and Dickson 1996). In other research, scholars have postulated an optimal team size for a given project (Pressman 2001), with the understanding that suboptimal team size could have implications for team productivity and efficiency.

The questionnaire items used, along with their sources, are shown in Table A.1 in the appendix to this paper. Table 2 provides summary statistics and interconstruct correlations for our variables. For multi-item reflective constructs, the composite reliabilities are indicated and all are above the 0.70 threshold. All constructs show composite reliabilities higher than 0.50, thereby establishing their reliability. Divergent validity is confirmed as the average variance extracted (i.e., the average variance shared between a construct and its measures, AVE) is greater than the

Table 2 Descriptive Statistics, Composite Reliability, and Correlations Among Constructs

	Mean	S.D	C.R.	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)
1. Quality	5.862	0.645	0.84	0.756										
2. Project efficiency	−17.118	20.578	0.92	0.070	0.888									
3. Behavioral control	5.300	1.039	0.87	0.495	0.130	0.758								
4. Outcome control—Quality	4.773	1.227	0.84	0.555	0.019	0.656	0.730							
5. Outcome control—Efficiency	5.862	0.645	0.85	0.434	0.026	0.481	0.622	0.629						
6. Collaborative culture	5.590	0.819	0.88	0.540	−0.026	0.494	0.455	0.350	0.773					
7. Boundary spanning	0.00	1.000	—	0.174	0.327	0.161	0.052	0.031	0.240	0.799				
8. Project size	9.430	6.973	—	−0.096	−0.137	0.014	−0.146	0.014	0.043	0.071	1.000			
9. Project volatility	3.267	1.299	—	−0.295	−0.267	−0.222	−0.213	−0.140	−0.240	0.163	0.166	0.677		
10. Team size	15.783	21.845	—	−0.094	0.052	0.097	0.106	−0.044	−0.114	−0.026	−0.058	0.240	1.000	
11. Prior interactions	0.000	1.000	—	0.142	0.085	0.102	0.153	0.165	0.091	−0.031	−0.133	−0.143	−0.012	0.948

Notes. 1. Diagonal elements show the square-root of average variance extracted for each construct. 2. Boundary spanning is a formative construct with three dimensions: process (mean = 27.23), objects (mean = 13.56) and boundary-spanning roles (mean = 5.11). 3. Project volatility is a formative construct with three dimensions: requirements volatility (mean = 4.08), personnel turnover (mean = 2.65), and retention (mean = 3.06). 4. Prior interactions is a formative construct with two dimensions: prior projects (mean = 22.487) and concurrent projects (mean = 3.650).

variance shared with other constructs in the model (Fornell and Bookstein 1981).

As a first step in establishing the psychometric properties of our measures, we subject the reflective constructs to an exploratory factor analysis using varimax rotation and a threshold eigenvalue of 1.0. Each set of items pertaining to an underlying construct loaded well together on the construct, as shown in column two of Table A.2 in the appendix. A more stringent test of scale validity is to use oblique rotation techniques that allow individual factors to be correlated (Ford et al. 1986). Although the oblique rotated factor structure is typically harder to interpret, it is a more accurate representation of real-world complexity of the examined variables. We conducted factor analysis on our scales using maximum-likelihood extraction and applying promax rotation. The analysis provides us with five distinct factors capturing collaborative culture, overruns, quality, behavioral control, and a composite factor that accounted for both quality-based outcome control and efficiency-based outcome control. In other words, the higher interitem correlation between the items capturing the outcome-based controls did not separate out as expected. This is not entirely surprising because it is possible that a firm that implements outcome-based controls along the quality dimension will have a higher probability of implementing outcome-based controls along efficiency as well. This factor analysis procedure, however, attests to the scale validity of our other constructs and suggests that our measures meet the psychometric properties required for further analysis.[6]

To further verify the factor structure of our reflective constructs, we subjected the data to confirmatory factor analysis (CFA) using LISREL, shown in last column of Table A.2. All the hypothesized paths from the indicator variables to the hypothesized latent variables are significant at $p < 0.05$. In addition, we assessed convergent and discriminant validity of our constructs with Partial Least Squares (PLS) using correlations between items and latent factor scores. These results (not reported here) were consistent with the LISREL-based CFA and establish the psychometric properties of our items.[7]

We used PLS to estimate the research model used to test our hypotheses. PLS was used in estimating the research model because it is appropriate for situations where sample sizes are small and models are complex, and the goal of the research is explaining variance (Fornell and Bookstein 1982, Gefen et al.

[6] We are grateful to the AE for pointing out the benefits of non-orthogonal rotation to us.

[7] Detailed results of all the psychometric properties for our measures are available on request from the authors.

Table 3 Model Estimation: PLS

	DV = Software quality			DV = Project efficiency		
	Controls	Main effects	Main effects and interactions	Controls	Main effects	Main effects and interactions
Behavioral control		0.098	0.106		0.270^{ψ}	0.260^{ψ}
Outcome control (quality)		0.269^{*}	0.272^{**}		-0.212^{**}	-0.239^{***}
Outcome control (efficiency)		0.097	0.092		0.022^{*}	0.032^{**}
Boundary spanning		0.099^{***}	0.045		0.303^{ψ}	0.043
Behavioral control × Boundary spanning			0.102^{ψ}			0.188^{**}
Outcome control (quality) × Boundary spanning			0.049^{*}			
Outcome control (efficiency) × Boundary spanning						0.098^{ψ}
Collaborative culture		0.275^{*}	0.281^{**}		-0.035^{*}	-0.050^{**}
Project volatility	-0.412^{***}	−0.122	−0.123	−0.088	−0.206	−0.214
Project size	−0.065	-0.058^{ψ}	−0.056	-0.121^{***}	−0.104	-0.128^{ψ}
Prior interactions	0.090	0.025	0.031	0.073	0.037	0.039^{*}
Team size	0.024	−0.068	−0.066	0.059	0.081^{***}	0.075^{***}
Firm controls	Yes	Yes	Yes	Yes	Yes	Yes
R^2	0.191	0.457	0.494	0.033	0.208	0.248
N	96	96	96	96	96	96

$^{\psi}$ $p < 0.10$, $^{*}p < 0.05$, $^{**}p < 0.01$, $^{***}p < 0.001$.

2000). It also supports the modeling of formative constructs.[8]

Because moderation models require the use of interaction terms, we use the approach recommended by Chin (2000) in this paper. For interactions involving reflective indicators, we centered indicators for the main and moderating constructs and created all pairwise product indicators where each indicator from the main construct is multiplied with each indicator from the moderating construct. For interactions involving formative constructs, we used the following suggested two-step process. The first step entails using the formative indicators in conjunction with PLS to create underlying construct scores for the predictor and moderator variables. Step two consists of taking those single composite construct scores to create a single interaction term. Recent work in the IS area has used this approach in modeling interactions in PLS analysis (see http://e-companions.pubs.informs.org/ISR/1526-5536-2003-02-SupplA.pdf). The results of the PLS analysis are shown in Table 3. The path coefficients in the model were assessed using the jackknife routine.

[8] Survey-based data collection approaches raise the potential issue of common method bias. We test for this using both the Harmon's one-factor test as well as the more stringent test described in Podsakoff et al. (2003). In both cases, we see no support for the influence of method bias on our results, indicating that this issue may not affect our results.

4. Results and Discussion

The results from PLS shown in Table 3 provide broad support for the impact of organizational control on project performance. We follow Carte and Russell's suggested approach (2003) by first providing the model with control variables, followed by the main effects and the interaction models. We discuss results for the main effects in the quality model first. These are displayed for quality in the first two columns of Table 3. Our results show no significant effect of behavioral control on quality, indicating no support for H1A. Hypothesis 2A pertains to the direct effect of quality-based outcome control on quality in the project and receives some support in our analysis. Although we do not formally hypothesize a direct effect of boundary spanning, the results indicate a significant and positive effect of these activities on quality. Knowledge sharing between the client and vendor teams eliminates gaps in understanding requirements and ensures clarity of expectations, thereby improving the client team's ability to deliver to functional requirements. We see some support for this effect of boundary spanning in our analysis.

With respect to the project efficiency model, shown in the appropriate columns of Table 3, the results mirror those observed with respect to quality. The direct effect of behavioral control on project efficiency is marginally significant and in the hypothesized direction, showing some support for H1B. Hypothesis 2B postulated a direct effect of efficiency-based outcome control on project efficiency, and we see support for this hypothesis. Although the effect is significant, the magnitude is small, suggesting that further research is needed to confirm whether this finding can be effectively applied in practice.[9] Similar to the quality model, we see a direct effect of boundary spanning on project efficiency significant at the $p < 0.10$ level, indicating the value accruing from boundary spanning.

Hypotheses 3A and 3B postulated that outcome-based formal controls focused on a certain outcome dimension will have negative externalities with respect to other dimensions of outcomes; we find partial support for these effects. Specifically, we see that, in accordance with H3A, quality-based outcome controls are associated with a decrease in project efficiency, thus suggesting that formal controls may have unintended consequences that may dampen their overall utility. There is no such corresponding effect in the effect of efficiency-based outcome controls on quality. In our analysis, it is not possible to tease out whether these negative externalities arising from quality-based outcome controls are strategically intended—i.e., it is possible that the vendor manager intentionally focuses his/her attention on controls associated with quality with full knowledge of the externalities on efficiency. However, our analysis shows that instituting outcome controls focused on one dimension may have negative effects on other outcome dimensions in the outsourced project; this effect has not been highlighted or established empirically in extant literature and has implications for the use of formal outcome-based control in outsourcing engagements.

Hypotheses 4A and 4B pertained to the influence of the collaborative culture observed in the teams on project outcomes. Our results with respect to collaborative culture show mixed results; although collaboration enhances the quality in the project, it also leads to reduced project efficiency. Collaborative teams display a collectivist attitude where the focus is on enhancing the quality of the team output (Guzzo and Dickson 1996). Managers therefore find value in fostering a culture of collaboration, seen often in software organizations (Sheremata 2002) as well as in knowledge-intensive domains such as new product development (Swink and Calantone 2004). However, it is possible that collaboration creates overheads on the team, which leads to higher levels of overruns on the project. Although a collectivist orientation might lead to better-quality software, it could also be instrumental in reducing individual productivity, which aggregates up to reduced project efficiency, all else being equal. Our analysis provides some notion of the trade-off facing managers in charge of knowledge work where collaboration is beneficial, but may have costs associated with it.

Although the control variables are not the focus of our study, we briefly describe the results obtained. The greater the volatility in the project, the lower the quality achieved by the project, consistent with prior work establishing the negative effects of requirements uncertainty (Harter et al. 2000) and personnel issues (Gopal et al. 2003). Larger projects are also observed to lead to lower quality and reduced efficiency, as discussed in most software engineering contexts (Pressman 2001). Surprisingly, we see no significant effect of prior interactions or team size on

[9] It is conceivable that efficiency-based outcome control may not enhance efficiencies. Frequent measurement of outcomes during the development phase adds overheads to the project in terms of effort and cost, which may impede progress on the project. Thus, the positive effects of such controls may be nullified by these added costs. Although we observe a significant effect in the analysis, this result needs to be interpreted keeping in mind the magnitude of the effect.

Table 4 Tests of Moderation Effects

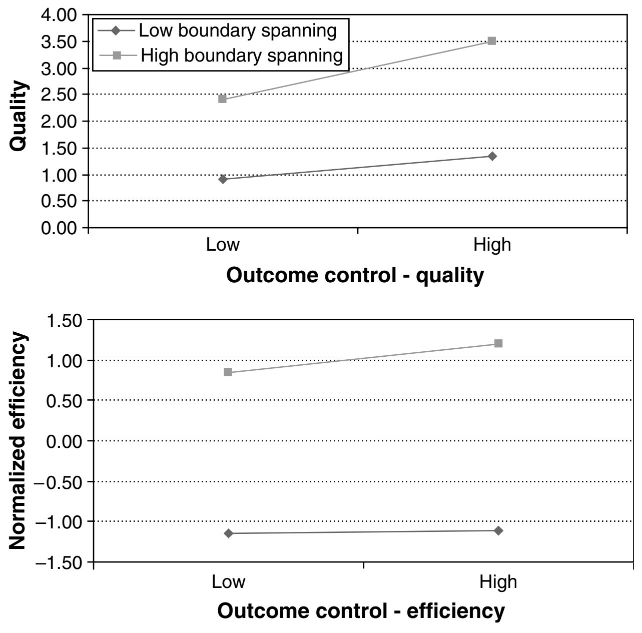
Boundary spanning as moderator of...	Included model R^2	Excluded model R^2	F^2	Pseudo-F (1, 88)	Conclusion
Behavioral control → Software quality	0.458	0.457	0.002	0.162	Not significant
Behavioral control → Project efficiency	0.243	0.208	0.046	4.069	Significant ($p < 0.05$)
Outcome control-quality → Software quality	0.482	0.457	0.048	4.247	Significant ($p < 0.05$)
Outcome control-efficiency → Project efficiency	0.233	0.208	0.033	2.868	Significant ($p < 0.10$)

quality, but a significant effect on efficiency, which is again consistent with prior work (Ethiraj et al. 2005). Finally, larger teams lead to higher efficiency, which is counterintuitive. Although there is research to indicate an optimal or suitable team size for a project (Pressman 2001), it is possible that the teams in our sample were on the increasing side of the curve, i.e., they were smaller than the theoretical optimal size. This may also stem from competitive pressures in this context to aggressively manage costs. Although this result is not central to our analysis in this paper, it indicates a need for more research in establishing the link between team size and project outcomes.

To test H5A–H5C, pertaining to the moderating influence of boundary spanning, we followed a hierarchical process where we compared the results of models with and without interaction constructs (Carte and Russell 2003). The significance of the interaction terms is assessed using a pseudo F-test (Chin et al. 1996). The f^2 statistic is computed based on the R^2 difference calculated as $(R^2_{\text{full}} - R^2_{\text{excluded}})/(1 - R^2_{\text{full}})$. The pseudo F statistic is calculated as $f^2 * (n-k-1)$, with 1, $(n-k)$ degrees of freedom when n is the sample size and k is the number of constructs in the model. These results, shown in Table 4, confirm that boundary-spanning activities interact with both behavioral and outcome control in impacting quality and project efficiency. They also show marginally more consistent and robust results with respect to outcome control, as discussed next.

Hypothesis 5A pertains to the positive moderating effect of boundary spanning on the relationship between behavioral control and project performance; we see partial support for this hypothesis in the columns with interaction results in Table 3 for both quality and project efficiency. The interaction of boundary spanning and behavioral control is statistically significant in enhancing project efficiency. However, based on tests of moderation, we do not see evidence of a significant interaction effect of behavioral control and boundary spanning in impacting quality. Hypotheses 5B and 5C pertain to the moderation effect of boundary spanning on the relationship between outcome control and performance and are both strongly supported. Boundary spanning positively moderates the effect of quality-based outcome control on quality and the effect of efficiency-based outcome control on project efficiency. We graph the interaction results holding all other variables at their means, as shown in Figure 2. Interestingly, in the low boundary spanning case (where the aggregate boundary-spanning construct is held at one standard

Figure 2 Interactions of Outcome-Based Controls and Boundary Spanning on Software Quality and Project Efficiency



deviation below the mean), the total direct and indirect returns to formal controls are small in the case of quality and almost nonexistent in the case of efficiency. However, in the high boundary-spanning case, the effect of formal control on quality and efficiency is significantly higher. These results indicate that boundary-spanning activities are important facilitators for the efficacy of formal outcome controls; our analysis provides clear pointers to inform the current debate about how the efficacy of formal controls in outsourced IT projects (Narayanaswamy et al. 2007) may be enhanced.

It is noteworthy that behavioral control does not appear to have a significant impact on quality as an outcome, but affects project efficiency. One reason for this dichotomy could be the relative ease with which it is possible to establish appropriate behavioral control with respect to schedules, costs, and effort in a project. The process management literature in software engineering stresses that it is easier to institute control over costs and schedules than over output quality (Pressman 2001). In addition, software process initiatives in most organizations have more clear-cut methodologies and behavioral norms addressing productivity and project costs (Paulk et al. 1993). The benefits of boundary spanning allow the vendor to better align project plans and costs with understanding of the business domain, thereby leading to a stronger moderating effect.

On the other hand, establishing such behavioral norms and methodologies for quality is much harder. Software development is comparable to R&D for the uncertainty that exists in the development process (Pressman 2001). In such uncertain contexts, prior research shows that it is hard to monitor or indeed establish behavioral control with respect to quality or even establish behavioral norms that can be rigorously enforced (Ravichandran and Rai 2000). In such contexts, any additional knowledge gained from boundary spanning does not appear to make a significant difference in quality.

A consistent result in our analysis is the overarching efficacy of outcome-based control in enabling better outcomes. Our results speak to the power of incentives that drive agent behavior; these incentives appear to guide the right agent behavior, providing support for the observations in past work (Choudhury and Sabherwal 2003, Kirsch 1996). The impact of these incentives also enables the vendor and the vendor team to use all the knowledge that they have access to, explaining the moderating effect of boundary spanning. Any business domain knowledge that can be gleaned and used in improving the project is elicited because these lead to better outcomes for the development team (Levina and Vaast 2005). In a comparative sense, our analysis indicates the relative strength of outcome control over behavioral control in ensuring performance.

Of particular note in our analysis is the significance of collaborative culture as a clan value. Unlike behavior and outcome controls, there is no need for explicit incentives to align the goals of members of the project team with the organization because of the existence of shared goals that rely on cultural transmission. One of the unexpected findings related to clan controls is that an increase in the extent to which the project team subscribes to a collaborative culture leads to likelihood of reduced project efficiency. While more work is needed to understand this finding, our work points to the need to strike a balance between the collectivist and individualistic values in managing outsourcing projects.

5. Limitations

We point out some limitations of this study that could be addressed by future research. First, our quality measure is subjective, hence prone to bias. Although we test our quality measures for reliability and validity, it is not possible to rule out bias in the measures. Second, our research methodology necessitated relatively less granular measures for our constructs. This was because of the cross-organizational data collection effort as well as the need to minimize the size of the instrument with manager time constraints in mind. However, we observe significant relationships even at this level of measurement. Third, our sample size of 96 is relatively low. Although our sample size is not unlike past work in outsourcing and software development, it restricts the number of covariates we can introduce in the model. Past research has identified several factors that affect project performance such as complexity (Pressman 2001), human resources (Gopal et al. 2003), and the use of tools (Krishnan et al. 2000). Although these were not the focus of our study, note that our model of performance is not meant to be exhaustive and also does not imply that other factors mentioned above do not matter. Fourth, our sampling framework involves Indian firms at and above CMM level 3. Therefore, although our results generalize to similar firms, they may not be directly applicable to firms that are not CMM-certified or those that operate in smaller niche or boutique firms. Fifth, we exclude self-control from our analysis and focus only on one aspect of clan control. Clearly, there are contexts where these controls have a strong role to play in determining project performance; our model does not consider these effects. Sixth, we adapt items from prior research in measuring the dimensions of control in our paper; however, there are some gaps in their measurement that future research will hopefully fill. For instance, our definition of behavioral control does not explicitly address the "rewards" dimension of control. Finally, prior research has pointed out that effect sizes in PLS research of 0.02 and 0.15 constitute a small impact on dependent variables (Chin et al. 1996). Three of the four interaction coefficients are below 0.15 and therefore constitute a small effect size. Although these effects are significant, more research is needed to establish the relative magnitude of the impact of boundary spanning on control and performance.

6. Conclusion

The main contribution of this study is in showing that organizational control portfolios need to be carefully tuned and directed towards key objectives using the appropriate controls portfolio mix, especially as the organization balances disparate goals. A key contribution of this paper is in integrating the control perspective with prior research in boundary spanning, particularly focusing on integration of knowledge from disparate domains. We find that outcome-based controls are effective in addressing the outcomes of choice positively but have possible negative implications for other outcomes. We also find that boundary spanning moderates the impact of outcome control on respective outcomes as well as the impact of behavioral control on project efficiency. In the process, we also develop a measure for boundary spanning based on research on boundary spanning in both software development and product development literatures (Levina and Vaast 2005, Carlile 2002).

Apart from contributing to the literature by showing the link between control and project outcomes, our work has practical implications for managers seeking to leverage outsourcing. It suggests that while up-front attention to the design of appropriate control mechanisms is important, it is also critical to have effective liaisons, boundary objects, and interaction processes at the interface between client and vendor organizations on an ongoing basis to make sure that control is finely tuned to the unfolding contextual conditions. While anecdotal evidence of boundary spanning exists in the practitioner press, we specifically recommend that managers strategize about boundary spanners and objects "in practice" (Levina and Vaast 2005) and institutionalize them in ongoing projects. While the appropriate choice of objects and spanners is a question that we have not considered in this paper, this clearly reflects an avenue for future research. Additionally, although the software process movement has gained sufficient traction and behavioral control is seen as part of the manager's arsenal, our research points to the value of outcome control and the strength of incentives. Although the contract aligns incentives between the vendor and the client (Gopal et al. 2003), similar incentive alignment mechanisms within the vendor firm are required to motivate appropriate behavior within the development team and enhance knowledge-sharing. Our work also indicates that although the choice of control portfolios is well-established in extant literature, the link between control choices and performance needs further research.

Acknowledgments
The authors gratefully acknowledge the help received from Dr. David P. Darcy in the earlier stages of this study. They are also grateful to Professors Samer Faraj and Ritu Agarwal for their help during the preparation of this manuscript. Thanks are due to the valuable advice provided by the AE and three anonymous reviewers on various versions of this manuscript during the review process at *ISR*. This research has been supported by a summer research grant from the Lattanze Center at Loyola College of Maryland. Feedback from seminar participants at the Lattanze Center is gratefully acknowledged.

Appendix

Table A.1 Questionnaire Items and Sources

	Questionnaire items (variable name)	Sources—*dimension*
Behavioral control—*reflective*	• The project followed documented processes for software development (Behcon1)	Kirsch (1996)—*Behavioral process control*
	• Project estimates (such as project size, cost, and schedule) were documented regularly for use in planning and tracking the project (Behcon2)	Adapted from Krishnan et al. (2000) and Gopal et al. (2002b)—*Measurement*
	• Changes to software system requirements resulted in appropriate changes to project plans and schedules (Behcon3)	Krishnan et al. (2000)—*Evaluation*
	• The project's actual results on the project plan were compared regularly with estimates in the project plan (Behcon4)	Krishnan et al. (2000)—*Evaluation*
	• Corrective action was taken proactively when actual results deviated from the project plan (Behcon5)	Krishnan et al. (2000)—*Evaluation*
	• All project members agreed to the commitment of their time as per the project plan (Behcon6)	Gopal et al. (2002b)—*Behavioral Process Control*
Outcome control—quality *Reflective*	• Software quality was assessed and compared to the quality goals in the project plan (Outq1)	Krishnan et al. (2000)—*Measurement /Evaluation*
	• The project had measurable and quantified goals for the quality of released software (Outq2)	Krishnan et al. (2000)—*Evaluation*
	• Software quality was used as a basis for rewards for project members (Outq3)	Ravichandran and Rai (2000, p. 411, Quality orientation of reward schemes)—*Rewards*
	• Software quality measures like defect rates were taken into account in rewarding project members (Outq4)	Ravichandran and Rai (2000, p. 411, quality orientation of reward schemes)—*Rewards*
	• User satisfaction with software quality was an important factor in determining rewards for project members (Outq5)	Ravichandran and Rai (2000, p. 411, Quality orientation of reward schemes)—*Rewards*
Outcome control—efficiency *Reflective*	• To the best of your knowledge, how often was development effort tracked on this project? (1—very infrequently to 7—very frequently) (Outeff1)	Gopal et al. (2002b)—*Measurement*
	• To the best of your knowledge, how often was project schedules tracked on this project? (1—very infrequently to 7—very frequently) (Outeff2)	Gopal et al. (2002b)—*Measurement*
	• Timely achievement of project goals was used as a basis for rewarding project members (Outeff3)	Ravichandran and Ravi (2000, p. 411, Adapted replacing quality with costs and schedule)—*Evaluation/Rewards*
	• Completing the project on time was an important factor in determining rewards for project members (Outeff4)	Ravichandran and Ravi (2000, p. 411, Adapted replacing quality with costs and schedule)—*Evaluation/Rewards*
	• The project's performance on cost was used as a basis for rewarding project members (Outeff5)	Ravichandran and Ravi (2000, p. 411, Adapted replacing quality with costs and schedule)—*Evaluation/Rewards*
	• Adherence to planned schedule was used as a basis for rewards for project members (Outeff6)	Ravichandran and Ravi (2000, p. 411, Adapted replacing quality with costs and schedule)—*Evaluation/Rewards*
Collaborative culture—*Reflective*	• People in the project team were supportive and helpful (Collab1)	Harley and Hult (1998)
	• There was willingness to share responsibility for failure (Collab2)	
	• There was willingness to collaborate across different groups (Collab3)	
Boundary Spanning—Roles	• One or more members of the project team had been formally designated to facilitate coordination with client	Adapted from discussion in Levina and Vaast (2005)
	• One or more members of the project team informally facilitated coordination with the client	
Boundary spanning—process	• Project planning meetings	Adapted from Gopal et al. (2002a) and Carlile (2002)
	• Status review meetings	
Boundary spanning—objects	• Code Inspections	Adapted from Gopal et al. (2002a) and Carlile (2002); discussion also present in Levina and Vaast (2005)
	• Design Reviews	

Table A.1 (Cont'd.)

	Questionnaire items (variable name)	Sources—*dimension*
Boundary spanning—objects	• Code Inspections • Design Reviews	Adapted from Gopal et al. (2002a) and Carlile (2002); discussion also present in Levina and Vaast (2005)
Software quality—*reflective*	• Response time • Flexibility • Usability • Reliability	Adapted from Szejko (1999). Matches with Tiwana (2004) and Ravichandran and Rai (2000)
Project efficiency—*reflective*	Measured using actual project performance compared to budget: • Schedule over-runs • Cost over-runs • Effort over-runs	Adapted from Gopal et al. (2002a)
Project volatility—*formative*	• Employee turnover from the project team was a major problem (Turnover) • Requirements significantly fluctuated over the course of the project (Req Instability) • It was difficult to retain employees with the skills required in this project within the organization (Retention)	Adapted from Gopal et al. (2003), based on Nidomolu (1995)

Table A.2 Exploratory and Confirmatory Factor Analysis of Reflective Factors

Item (% variance explained)	EFA with varimax rotation	Confirmatory factor analysis measurement model standardized loadings all loadings significant at $p < 0.01$ unless indicated			
Software quality (58.5%)					
Flexibility	0.673	1.00#			
Usability	0.813	0.84			
Reliability	0.811	0.91			
Response time	0.719	0.77			
Eigenvalue	2.29				
Project efficiency (79.7%)					
Schedule overruns	0.910		1.00#		
Cost overruns	0.917		0.75		
Effort overruns	0.849		0.83		
Eigenvalue	2.389				
Behavioral control (58.8%)					
BehCon1	0.772			1.00#	
BehCon2	0.691			0.62	
BehCon3	0.836			0.55	
BehCon4	0.753			0.79	
BehCon5	0.733			0.86	
BehCon6	0.710				
Eigenvalue	2.876				
Outcome control-quality (55.6%)					
Outq1	0.606				1.00#
Outq2	0.732				0.84
Outq3	0.799				0.86
Outq4	0.755				0.79
Outq5	0.760				0.83
Eigenvalue	2.68				

#Fixed parameter for scale, $^*p < 0.05$.

Table A.2 (Cont'd.)

Item (% variance explained)	EFA with varimax rotation	Confirmatory factor analysis measurement model standardized loadings all loadings significant at $p < 0.01$ unless indicated	
Outcome control-efficiency (53%)			
Outeff1	0.879	$1.00^{\#}$	
Outeff2	0.851	0.84	
Outeff3	0.797	0.75	
Outeff4	0.832	0.77	
Outeff5	0.649	0.60^{*}	
Outeff6	0.535	0.58^{*}	
Eigenvalue	2.53		
Collaboration (59.9%)			
Collab1	0.692		$1.00^{\#}$
Collab2	0.802		0.86
Collab3	0.822		0.75
Eigenvalue	1.79		
Chi-square (279 d.o.f) = 492.47, $p < 0.01$			
Goodness of fit = 0.81, Adjusted goodness of fit = 0.76			
Comparative fit index = 0.91, Incremental fit index = 0.91			
Normed fit index = 0.89, Standardized root mean residual = 0.08			

$^{\#}$Fixed parameter for scale, $^{*}p < 0.05$.

References

Allen, D., D. Lueck. 1999. The role of risk in contract choice. *J. Law, Econom. Organ.* **15**(3) 704–736.

Banerjee, A. V., E. Duflo. 2000. Reputation effects and the limits of contracting: A study of the Indian software industry. *Quart. J. Econom.* **115**(3) 989–1017.

Carlile, P. R. 2002. A pragmatic view of knowledge and boundaries: Boundary objects in new product development. *Organ. Sci.* **13**(4) 442–455.

Carlile, P. R. 2004. Transferring, translating, and transforming: An integrative framework for managing knowledge across boundaries. *Organ. Sci.* **15**(5) 555–568.

Carte, T. A., C. J. Russell. 2003. In pursuit of moderation: Nine common errors and their solutions. *MIS Quart.* **27**(3) 479–501.

Chin, W. W., B. L. Marcolin, P. R. Newsted. 1996. A partial least squares latent variable modeling approach for measuring interaction effects: Results from a Monte Carlo simulation study and voice mail emotion/adoption study. *Proc. Seventeenth ICIS*, Cleveland, OH, 21–41.

Choudhury, V., R. Sabherwal. 2003. Portfolios of control in outsourced software development projects. *Inform. Systems Res.* **14**(3) 291–314.

Covaleski, M. A., M. W. Dirsmith, J. B. Heian, S. Samuel. 1998. The calculated and the avowed: Techniques of discipline and struggles over identity in Big Six public accounting firms. *Admin. Sci. Quart.* **43** 293–327.

Eisenhardt, K. M. 1985. Control: Organization and economic approaches. *Management Sci.* **31**(2) 134–149.

Ethiraj, S. K., P. Kale, M. S. Krishnan, J. V. Singh. 2005. Where do capabilities come from and how do they matter? A study in the software services industry. *Strategic Management J.* **26**(1) 25–45.

Ford, J. K., R. C. MacCallum, M. Tait. 1986. The application of exploratory factor analysis in applied psychology: A critical review and analysis. *Personnel Psych.* **39** 291–314.

Fornell, C., F. L. Bookstein. 1982. Two structural equation models: LISREL and PLS applied to consumer exit-voice theory. *J. Marketing Res.* **19** 440–452.

Gefen, D., D. W. Straub, M.-C. Boudreau. 2000. Structural equation modeling and regression: Guidelines for research practice. *Comm. Assoc. Inform. Systems* **4**(7) 2–77.

Gopal, A., T. Mukhopadhyay, M. S. Krishnan. 2002a. The role of software processes and communication in offshore software development. *Comm. ACM* **45**(4ve) 193–200.

Gopal, A., T. Mukhopadhyay, M. S. Krishnan, D. R. Goldenson. 2002b. Measurement programs in software development: Determinants of success. *IEEE Trans. Software Engrg.* **28**(9) 863–875.

Gopal, A., K. Sivaramakrishnan, M. S. Krishnan, T. Mukhopadhyay. 2003. Contracts in offshore software development: An empirical analysis. *Management Sci.* **49**(12) 1671–1683.

Grant, R. M. 1996. Prospering in dynamically-competitive environments: Organizational capability as knowledge integration. *Organ. Sci.* **7**(4) 375–387.

Guzzo, R. A., M. W. Dickson. 1996. Teams in organizations: Recent research on performance and effectiveness. *Annual Rev. Psych.* **47** 307–338.

Harter, D. E., M. S. Krishnan, S. A. Slaughter. 2000. Effects of process maturity on quality, cycle time, and effort in software product development. *Management Sci.* **46**(4) 451–467.

Henderson, J. C., S. Lee. 1992. Managing IS design teams: A control theories perspective. *Management Sci.* **38**(6) 757–777.

Herbsleb, J. D., A. Mockus. 2003. An empirical study of speed and communication in globally distributed software development. *IEEE Trans. Software Engrg.* **29**(6) 481–494.

Hoegl, M., H. G. Gemuenden. 2001. Teamwork quality and the success of innovative projects: A theoretical concept and empirical evidence. *Organ. Sci.* **12**(4) 435–449.

Hurley, R. F., G. T. M. Hult. 1998. Innovation, market orientation and organizational learning: An integration and empirical examination. *J. Marketing* **62**(3) 42–54.

Jaworski, B. J. 1988. Toward a theory of marketing control: Environmental context, control types, and consequences. *J. Marketing* **52** 23–39.

Jensen, M. C., W. H. Meckling. 1976. Theory of the firm: Managerial behavior, agency costs, and ownership structure. *J. Financial Econom.* **3** 305–360.

Kirsch, L. J. 1996. The management of complex tasks in organizations: Controlling the systems development process. *Organ. Sci.* **7**(1) 1–21.

Kirsch, L. J. 1997. Portfolios of control modes and IS project management. *Inform. Systems Res.* **8**(3) 215–239.

Kirsch, L. J. 2004. Deploying common systems globally: The dynamics of control. *Inform. Systems Res.* **15**(4) 374–395.

Kirsch, L. J., V. Sambamurthy, D.-G. Ko, R. L. Purvis. 2002 Controlling information systems development projects: The view from the client. *Management Sci.* **48**(4) 484–498.

Kogut, B., U. Zander. 1992. Knowledge of the firm, combinative capabilities, and the replication of technology. *Organ. Sci.* **3**(3) 383–397.

Krishnan, M. S., C. H. Kriebel, S. Kekre, T. Mukhopadhyay. 2000. An empirical analysis of productivity and quality in software products. *Management Sci.* **46**(6) 745–759.

Levina, N. 2005. Collaborating on multiparty information systems development projects: A collective reflection-in-action view. *Inform. Systems Res.* **16**(2) 109–130.

Levina, N., J. Ross. 2003. From the vendor's perspective: Exploring the value proposition in IT outsourcing. *MIS Quart.* **27**(3) 331–364.

Levina, N., E. Vaast. 2005. The emergence of boundary spanning competence in practice: Implications for implementation and use in information systems. *MIS Quart.* **29**(2) 335–363.

March, J. G., Z. Shapira. 1987. Managerial perspectives on risk and risk taking. *Management Sci.* **33**(11) 1404–1418.

Meyer, B. 2006. The unspoken revolution in software engineering. *IEEE Comput.* **39**(1) 121–124.

Narayanaswamy, R. S., R. M. Henry, R. L. Purvis. 2007. Understanding the effect of control on information systems project performance via meta-analysis. *Academy of Management Annual Meeting*. Philadelphia.

Necco, C. R., C. L. Gordon, N. W. Tsai. 1987. Systems analysis and design: Current practices. *MIS Quart.* **11**(4) 461–475.

Ocasio, W. 1997. Towards an attention-based view of the firm. *Strategic Management J.* **18** 187–206.

Ouchi, W. G. 1979. A conceptual framework for the design of organizational control mechanisms. *Management Sci.* **25**(9) 833–848.

Paulk, M. C., B. Curtis, M. B. Chrissis, C. V. Weber. 1993. Capability maturity model for software version 1.1. Technical Report SEI-93-TR-24, Software Engineering Institute, Pittsburgh.

Pawlowski, S. D., D. Robey. 2004. Bridging user organizations: Knowledge brokering and the work of information technology professionals. *MIS Quart.* **28**(4) 645–672.

Podsakoff, P. M., S. B. MacKenzie, J. Lee, N. P. Podsakoff. 2003. Common method biases in behavioral research: A critical review of the literature and recommended remedies. *J. Appl. Psych.* **88**(5) 879–903.

Pressman, R. S. 2001. *Software Engineering: A Practitioner's Approach*, Fifth ed. McGraw Hill, New York.

Ravichandran, T., A. Rai. 2000. Quality management in systems development: An organizational system perspective. *MIS Quart.* **24**(3) 381–415.

Sheremata, W. A. 2002. Finding and solving problems in software new product development. *J. Product Innovation Management* **19** 144–158.

Star, S. L. 1989. The structure of ill-structured solutions: Boundary objects and heterogeneous distributed problem solving. M. Huhn, L. Gasser, eds. *Readings in Distributed Artificial Intelligence*. Morgan Kaufman, Menlo Park, CA, 37-54.

Swink, M. L., R. Calantone. 2004. Design-manufacturing integration as a mediator of antecedents to new product design quality. *IEEE Trans. Engrg. Management* **51**(4) 472–482.

Szejko, S. 1999. An exercise in evaluating the significance of software quality criteria. *ACM SIGCSE Bull.* **31**(3) 199.

Thibodeau, P. 2004. More IT jobs go offshore. *Computerworld* **38**(14) 13.

Thibodeau, P., L. Rosencrance. 2002. Users losing billions due to bugs. *Computerworld* **36**(27) 1–2.

Tiwana, A. 2004. An empirical study of effect of knowledge integration on software development performance. *Inform. Software Tech.* **46**(13) 899–906.

Wallace, L., M. Keil, A. Rai. 2004. How software project risk affects project performance: An investigation of the dimensions of risk and an exploratory model. *Decision Sci.* **35**(2) 289–321.

Westland, J. C. 2004. The cost behavior of software defects. *Decision Support Systems* **37** 229–238.