

Control in Internal and Outsourced Systems Development Projects

Amrit Tiwana (contact author)
Iowa State University
College of Business
Ames, Iowa 50011-1350
tiwana@iastate.edu

Mark Keil
Georgia State University
Robinson College of Business
35 Broad Street, Atlanta, Georgia 30302
mkeil@gsu.edu

April 2008, revised February 2009 and July 2009

JMIS-5739

Forthcoming in *Journal of Management Information Systems*

Acknowledgements: Developmental feedback from Ashley Bush is gratefully acknowledged.

Amrit Tiwana is an associate professor and holds the Union Pacific Professorship in Iowa State University's College of Business. His research on systems development and IT governance has appeared in journals including *Information Systems Research*, *Strategic Management Journal*, *Journal of Management Information Systems*, *Decision Sciences*, *California Management Review*, *ACM Transactions on Software Engineering & Methodologies*, *IEEE Transactions on Engineering Management*, and other journals. He currently serves as an associate editor at *Information Systems Research* and has served as a special associate editor at *MIS Quarterly*.

Mark Keil is the Board of Advisors Professor of Computer Information Systems in the J. Mack Robinson College of Business at Georgia State University. His research focuses on IT project management and includes work on preventing IT project escalation, identifying and managing IT project risks, and improving IT project status reporting. His interests also include IT implementation and use. Keil has published more than 100 refereed publications including papers that have appeared in *MIS Quarterly*, *Journal of Management Information Systems*, *Decision Sciences*, *Strategic Management Journal*, and many other journals. He currently serves on the editorial boards of *Information Systems Research*, *Journal of Management Information Systems*, *Decision Sciences*, and *Information Systems Journal*. He has also served on the editorial boards of *MIS Quarterly*, *IEEE Transactions on Engineering Management*, and *The DATA BASE for Advances in Information Systems*.

Control in Internal and Outsourced Systems Development Projects

Abstract

Although the choice of control mechanisms in systems development projects has been extensively studied in prior research, differences in such choices across internal and outsourced projects and their effects of systems development performance have not received much attention. This study attempts to address this gap using data on 57 outsourced and 79 internal projects in 136 organizations. Our results reveal a paradoxical overarching pattern: Controllers attempt greater use of control mechanisms in outsourced projects relative to internal projects, yet controls enhance systems development performance in internal projects but not in outsourced projects. We introduce a distinction between attempted control and realized control to explain this disconnect, and show how anticipated transaction hazards motivate the former but meeting specific informational and social prerequisites facilitate the latter.

Our results contribute three new insights to the systems development control literature. First, controllers attempt to use controller-driven control mechanisms to a greater degree in outsourced projects but controllee-driven control mechanisms to a greater degree in internal projects. Second, we establish a hitherto-missing control-performance link. The nuanced differences in internal and outsourced projects simultaneously confirm and refute a pervasive assertion in the IS controls literature that control enhances performance. Finally, we show how requirements volatility—which can be at odds with control—alters the control-performance relationships. Implications for theory and practice are also discussed.

Keywords: Project controls, control theory, systems development, attempted control, realized control, survey.

INTRODUCTION

A central problem in managing systems development projects is obtaining cooperation among organizations or departments with partially congruent objectives. Organizations attempt to foster this cooperation by deploying control mechanisms on the group that develops the system (“controllee”) by the group that will eventually use it (“controller”). Effective control of systems development projects is crucial considering that nearly 40% of IT investments in recent years have failed to deliver their intended benefits [5].

Three gaps in prior research on systems development control are particularly noteworthy. First, it has focused almost exclusively on predicting the controller’s *choice* of control mechanisms [e.g. 10, 23, 32]. Further, most of this research has focused on internal projects, even though organizations frequently outsource. A direct contrast of how controllers attempt to control internal projects relative to outsourced projects is therefore missing. However, control can be costly [21], and the opportunism hazards that control attempts to mitigate might be less pronounced in internal projects relative to outsourced projects. Whether organizations invest in control in a manner consistent with its role in internal and outsourced projects remains unclear. Since both internal and outsourcing arrangements are prevalent for systems development, it is important to understand these differences.

Second, much of the prior research embodies an intuitively appealing but untested assertion that greater control enhances systems development performance. However, there is little empirical evidence for the relationship between control and performance [24]. Nidumolu and Subramani [32] for example proposed but did not test the relationship between formal control and performance, and Henderson and Lee [17] studied only formal controls in internal projects.

It is plausible that control mechanisms will differentially impact performance in internal projects (where the control relationship spans an internal departmental boundary) than in outsourced projects (where it spans an interorganizational boundary). However, a theoretical explanation for this remains absent.

Third, the flexibility to cope with requirements volatility can be at odds with the rigidity required by control mechanisms. The intent to control is to foster adherence to a project plan. Defining such a plan depends on the controller's ability to clearly predefine the project outcomes that the controller expects the controllee to produce. Yet, projects increasingly contend with volatile requirements [2], which hinder prespecifying expected outcomes and a stable development plan. For example, changes in the external business environment, technological advances, or a shift in strategic focus might require changes to the initial project requirements while development is underway. Greater requirements volatility requires greater flexibility to deviate from a fixed plan—the opposite of what greater control attempts to encourage. It is plausible that while some control mechanisms might impede such flexibility others might facilitate it. However, no prior research has directly incorporated requirements volatility in its theory development, leaving unexplored the question of how requirements volatility alters (moderates) the relationship between various forms of control and performance in internal or outsourced projects.

These issues represent significant problems for practice because managers might choose control mechanisms that are ineffective in a particular project context but might have worked well in others. This can create an unrealistic optimism about project success or impose unnecessary costs. We address these gaps guided by the following three research questions:

1. How do control mechanisms that controllers attempt to use to influence systems development in internal projects differ relative to outsourced projects?

2. How and why does control impact systems development performance in internal and outsourced projects?
3. How does requirements volatility alter the control-performance relationships?

Building on control theory, we introduce a distinction between attempted control and realized control to address these questions. We then explain how attempts to influence the controllee using a control mechanism (“attempted control”) is driven by anticipated transaction hazards, but the controller’s ability to successfully exercise a control mechanism (“realized control”) depends on meeting the theoretically-neglected informational and social requirements specific to each control mechanism. Such differences across internal and outsourced projects influence both the controllers’ choice of control mechanisms and their effects on systems development performance. We also show how requirements volatility moderates the control-performance relationships by affecting the extent to which these requirements for each control mechanism are met in internal and outsourced projects. Tests using data from 57 outsourced and 79 internal projects in 136 organizations provide considerable support for the proposed ideas.

Our overarching pattern of results shows that controllers attempt greater use of control mechanisms in outsourced projects relative to internal projects, yet they enhance systems development performance only in internal projects but not in outsourced projects. We offer theoretical insights on this pattern by showing how the motivations of attempting control are different from the requirements for realizing it. Specifically, our study makes three novel contributions to research. First, it shows that controllers attempt to use controller-driven control mechanisms (outcome, behavior, and clan control) to a greater degree in outsourced projects relative to internal projects. They attempt to use controllee-driven control mechanisms (i.e., self control) to a greater extent in internal projects relative to outsourced projects. Second, we offer insights into how the same control mechanisms vary in their influence on systems development

performance in internal and outsourced projects. We show that controllee-driven and some controller-driven control mechanisms enhance performance in internal projects but not in outsourced projects; and controllee-driven control mechanisms decrease performance in outsourced projects. Third, we explain why and show how requirements volatility moderates these relationships. Collectively, our findings provide a missing link between project control and performance in internal and outsourced projects.

The rest of the paper is structured as follows. The next section develops the hypotheses, followed by the data collection and analyses. We conclude by discussing the implications of our findings for research and practice.

THEORETICAL DEVELOPMENT

In this section, we first review control theory and then introduce the distinction between attempted and realized control in systems development projects. We build on two underexplored ideas in control theory to theorize differences in how and why controllers attempt to control, and differ in their ability to realize control in internal and outsourced projects. Each control mechanism will therefore have differential performance effects, independent of the extent to which controllers attempt to use it. Finally, we theorize how requirements volatility moderates these control-performance relationships.

Control Theory and Control Mechanisms

Ouchi's [33] control theory, which is derived from agency theory, provides a useful theoretical foundation for conceptualizing both the differences in control portfolios and their effects on systems development performance in internal and outsourced projects. The notion of control is based on the premise that the controller (the principal) and controllee (the agent) have divergent interests, which control mechanisms attempt to align [14]. Two ideas form the crux of control

theory: (1) control is used by a controller as a device for discouraging anticipated opportunistic behavior by a controllee [35] and (2) each control mechanism has informational requirements and social requirements that must be met for it to be effective [33]. We subsequently build on the former idea in control theory to explain differences in control mechanisms attempted in internal and outsourced projects, and on the latter idea to theorize how control distinctively influences systems development performance in these two contexts.

Of these two elements of control theory, the emphasis in prior research has been the former while the latter has received less attention. For example, a series of studies by Kirsch and her colleagues predict the choice of control mechanisms that client departments attempt to use to influence internal IT departments [23, 25]. With a few recent exceptions focused exclusively on outsourced projects [10, 39], the majority of the controls literature has focused on internal projects and we are aware of no prior research that has directly contrasted control in internal and outsourced projects.

Broadly speaking, control refers to the process and rules governing controllee actions implemented by the controller to promote desirable controllee behaviors [10, 12, 21]. Following Choudhury and Sabherwal [10], we use the term controller to refer to teams of individuals in the client department or organization responsible for designing and implementing controls and controllee to refer to individuals in the IT department or vendor organization responsible for systems development [10, 23]. In internal projects, the controller is usually the user department for which the application is being developed and the controllee is the internal IT unit that is developing it [23]. In outsourced projects, the controller is the client organization's IT department and the controllee is an outside vendor contracted to develop the application [10].

Control is implemented through a variety of *control mechanisms*. These can broadly be classified as: (a) formal controls, which rely on adherence to performance standards or

prescribed processes and (b) informal controls, which rely on social or norm-emphasizing strategies [10, 25, 34].

Control theory suggests two forms of formal control: (1) outcome controls, which refer to the prespecification of the desired interim and final outputs without regard to the process by which the outputs are achieved and (2) behavior controls, which prescribe documented and repeatable methods and procedures to the controllee for accomplishing project activities [14, 33]. Outcome controls therefore prespecify what the controllee should accomplish while process controls prescribe how the controllee should achieve that outcome.

Informal control can similarly take two forms: (1) clan control, which refers to mechanisms that minimize differences between the controller and controllee's goals by promulgating common values, beliefs, and acceptable behaviors and (2) self control, which refers to a relational form of control that relies on the controllee engaging in behaviors that are consistent with the best interests of the controller without formal controls [10, 23]. Clan control therefore operates through socialization. It attempts to foster common goals between the controller and controllee by propagating norms, values, and a shared ideology. The premise is that this results in a shared understanding of project goals and objectives, which should enhance project outcomes [34]. Unlike all other forms of control that are implemented by a controller, self control is implemented by the controllee through self-specification of self-monitored standards for its own behavior [10]. Self control can therefore be viewed as a controllee-driven control mechanism and the other three as controller-driven control mechanisms. In practice, project control is achieved through a combination or "portfolio" of the four control mechanisms [10, 23].

Attempted Versus Realized Control

We begin our theory development by introducing the distinction between attempted control and realized control. We define *attempted control* as the extent to which a controller attempts to utilize a given control mechanism to influence controllee behavior. Attempted control therefore refers to the control mechanisms that the controller implements in a given project, independent of whether or how they are exercised. We define *realized control* as the extent to which the controller is able to successfully exercise a given control mechanism during the systems development process. An attempted control mechanism must be effectively exercised, or realized, for it to enhance systems development performance. Whether the influence tactics associated with each control mechanism translate into improved performance therefore depends on its realization.

We subsequently delineate how the motivations for attempting each control mechanism can differ from the requirements for realizing it. Although anticipated opportunism hazards influence the control mechanisms that a controller attempts to use to influence controllee behavior, whether such attempts translate into improved performance (i.e., control is realized) depends on whether the informational and social requirements for each control mechanism are met in a given project context. We emphasize that almost all prior IS controls research has implicitly focused on what we conceptualize as attempted control rather than realized control. In other words, that body of work explains what control mechanisms controllers will attempt to use (and largely in internal projects), not whether those control mechanisms were realized.

Differences in Attempted Control in Internal and Outsourced Projects

Control structures are designed in response to anticipated hazards [1]. Thus the controller's expectations of likely hazards rather than the viability of control ought to influence attempted control. Prior research on *internal* projects has shown that measurability of project outcomes,

observability of controllee behavior, and the controller's technical knowledge influence the choice of control mechanisms that controllers attempt to use. To this list derived from studies of internal projects, we add a fourth factor: perceived risk of controllee opportunism. In projects with high outcome measurability, with high behavior observability, and where the controller possesses high levels of technical knowledge, controllers are more likely to attempt to use formal control mechanisms [23, 24]. Otherwise, controllers are more likely to rely on informal control mechanisms. Perceived fears of opportunism enter the picture when we contrast internal with outsourced projects.

<< INSERT TABLE 1 HERE >>

Table 1 summarizes the salient differences between internal and outsourced projects. Three thematic differences that can differentiate attempted control in internal and outsourced projects are noteworthy. (1) Control spans an intraorganizational boundary in internal projects but an interorganizational boundary in outsourced projects; (2) the controller's technical knowledge is likely to be higher in outsourced projects than in internal projects (we subsequently confirm this assertion); and (3) controllee behavior observability is likely to be higher in internal projects relative to outsourced projects.

To conceptualize how these differences are likely to systematically differentiate attempted control in internal projects and outsourced projects, we return to the first element of control theory i.e., the intent of control as a device to discourage opportunism. We expect controllers to attempt greater formal control in outsourced projects relative to internal projects for reasons that follow from the foregoing differences. First, the controller and controllee in internal projects are two departments within the same organization whereas in outsourced projects they are two independent organizations bound by a market contract. The perceived threat of opportunism is therefore likely to be greater in outsourced projects relative to internal projects [11, 12, 44].

Therefore, the controllee has an incentive to measure and monitor controllee performance using formal controls to a greater degree in outsourced projects relative to internal projects. This can be accomplished by clearly prespecifying the expected project outcomes (outcome control) and by dictating methods and procedures to the controllee (behavior control).

Second, specifying formal controls requires greater technical knowledge [23, 24]; having more technical knowledge can influence the extent to which it is attempted.

The controller in internal projects is typically another internal department whereas in outsourced projects it is the IT department of the client organization [10, 24]. Thus controllers are in a better position to technically prespecify desired controllee outputs as well as to prescribe methods and procedures in outsourced projects relative to internal projects. Therefore, controllers are more likely to attempt greater use of both outcome control and behavior control in outsourced projects relative to internal projects. This leads to our first set of hypotheses.

***Hypothesis 1a.** Organizations will attempt to use outcome control to a greater degree in outsourced projects than in internal projects.*

***Hypothesis 1b.** Organizations will attempt to use behavior control to a greater degree in outsourced projects than in internal projects.*

However, the behavior of an internal IT department is likely to be more observable relative to a project team in a different organization. Therefore, behavior observability is likely to be lower in outsourced projects relative to internal projects since the controller and controllee are separated by an organizational boundary rather than a departmental boundary. According to control theory, such inability to observe controllee behavior is likely to encourage the controller to supplement the formal controls with informal clan control [24]. Furthermore, controllers likely have lower direct authority over controllees in outsourced projects because, unlike internal projects, they belong to different organizations. Therefore formal controls are likely to be supplemented to a greater extent by informal social pressure exerted by the controller on the

controllee [10]. These informal approaches represent clan control. Since departments within the same organization are more likely to share the same norms and values than two independent organizations, the controller and controllee are likely to share norms and values to a lesser extent in outsourced projects relative to internal projects. Therefore, controllers in outsourced projects attempt clan control through active participation in project meetings and regular interactions with controllees in order to cultivate shared goals, values, and norms [10]. A second motivation for attempting to use clan control is that the transaction costs associated with clan control can be lower than formal control [28]. Such costs are more likely to be a more pronounced concern in outsourced projects, wherein the control relationship spans an interorganizational boundary. These reasons therefore are likely to encourage controllers to attempt clan control to a greater extent in outsourced projects. Therefore, we expect that controllers will attempt to use clan control to a greater degree in outsourced relative to internal projects. This leads to our next hypothesis.

Hypothesis 1c. *Organizations will attempt to use clan controls to a greater degree in outsourced projects than in internal projects.*

Finally, self control can be viewed as abdication of formal control by the controller. Self control therefore represents a controllee-driven *non-controlling*, informal relational governance approach that is unlike the other three control mechanisms. (We subsequently empirically confirm this.) Controllers incur costs in terms of managerial effort in attempting control [43], and the perceived threat of opportunistic controllee behavior that formal control attempts to discourage is likely less pronounced in internal projects relative to outsourced projects. So controllers are likely to be predisposed against incurring such costs in internal projects compared to outsourced projects. Therefore, the perceived lower benefits vis-à-vis costs of formal control are likely to encourage controllers to grant greater autonomy to controllees in internal projects. This represents greater

self control. Furthermore, as previously discussed, the controller in internal projects is likely to have lower technical knowledge than is required to implement formal controls. When formal controls are usable to a lesser degree, self control is often used as a substitute for them [24]. This further increases the proclivity of controllers to attempt to rely on self control to a greater degree in internal projects than they would in outsourced projects. This leads to our next hypothesis.

***Hypothesis 1d.** Organizations will attempt to use self controls to a greater degree in internal projects than in outsourced projects.*

In summary, we expect that controllers will attempt to rely more heavily on controller-driven controls in outsourced projects and controllee-driven controls in internal projects.

Influence of Control Mechanisms on Systems Development Performance

The relationship between control and systems development performance is widely asserted [e.g., 24, 32], but is neither fully theoretically developed nor simultaneously tested in internal and outsourced projects. While we agree with the premise that control mechanisms attempt to align the interests of the controller and controllee, it is important to differentiate the controllers' attempt to use these mechanisms from the extent to which controllers are actually able to exercise the control that they attempt. Merely specifying a control does not enhance systems development performance if it is not effectively exercised or enforced [21]. It is only the extent to which a given control mechanism is realized during the systems development process that influences performance. We believe that the pervasive assertion in prior research that control enhances performance is not theoretically nuanced to recognize the subtlety between attempted and realized control. Some observed differences between internal and outsourced projects in the controls literature might simply be an artifact of confounding attempted control with realized control. For example, Kirsch's [24] study of internal projects observed greater reliance on formal controls whereas Choudhury and Sabherwal's [10] study of outsourced projects observed

informal controls to a greater degree. In our perspective, the former study appears to have examined attempted control whereas the latter examined realized control..

Unlike attempted control that is motivated by anticipated agency hazards, realized control depends on specific informational and social requirements. As Ouchi describes it, formal control is informationally-demanding whereas informal control is socially-demanding. This point is explicitly recognized in Ouchi's [33] control theory but is largely overlooked in subsequent theory development and empirical work (e.g., [10, 14, 23]).¹ These informational and social preconditions influence whether an attempted control mechanism can be realized, and are key to understanding whether controls influence performance. Informational requirements refer to the information that the controller needs for a formal control mechanism to function. Social requirements refer to a minimally-necessary set of agreements between controller and controllee that enable an informal control mechanism to influence controllee performance [33]. Table 2 summarizes the informational and social requirements for attempting and realizing each of the four control mechanisms. The conceptual logic for these relationships is developed next.

<< INSERT TABLE 2 HERE >>

Outcome Control and Systems Development Performance

Realizing outcome control requires that the controller be able to measure actual project outcomes against prespecified benchmarks such as the schedule, budget, project deliverables, acceptable defect levels, and overall software quality [10, 23]. Such outcome expectations can be specified for a given project irrespective of whether it is internal or outsourced, allowing it to be attempted in both contexts. The controllee is then rewarded or penalized based on how well it met the prespecified benchmarks (e.g., budget, schedule, and quality). Therefore, outcome controls provide guidance to the controllee for how the controller will evaluate the work that is produced.

¹ The only exception is Wang et al.'s [42] study that recognized the role of cost information asymmetry (which is comparable to Ouchi's notion of informational prerequisites for formal control) in controlling systems development projects; they did not however do so building on control theory.

Realizing output control, however, requires that the controllee outputs be evaluated by the controller in relation to the prespecified benchmarks, which are likely to be assessable in both internal and outsourced projects. For example, timeliness of delivered milestones, compliance with budget, and reliability can be evaluated in relation to target benchmarks that were specified at the outset of the project independent of whether the project is internal or outsourced. We therefore expect outcome controls to enhance systems development performance in both internal and outsourced projects.

Hypothesis 2a. Greater use of outcome control positively affects systems development performance in internal projects.

Hypothesis 2b. Greater use of outcome control positively affects systems development performance in outsourced projects.

Behavior Control and Systems Development Performance

Behavior control depends on the prespecification of systems development methods and procedures that the controller expects the controllee to follow and that it believes will ensure better systems development outcomes. The key informational requirement for realizing behavior control is controllee behavior observability by the controller [23]. The controller then imposes penalties for noncompliance with the prescribed methods and procedures. Ex post sanctioning is effective only if the controller can monitor and accurately assess controllee behavior [7].

Realizing behavior control therefore requires surveillance and monitoring of controllee behaviors through direct observation or using software-based tools to evaluate whether the controllee followed the prescribed methods and procedures during the development process [14, 33]. Information that is needed from the controllee to realize behavior control is costlier to obtain when systems development work is being conducted in a different—sometimes geographically separated—organization. The absence of preexisting controller-controllee informational conduits in outsourced projects makes it difficult for the controller to observe controllee behaviors [10,

18]. Therefore, the informational requirements for realizing behavior control are less likely to be met in outsourced projects.

A second factor might exacerbate realizing behavior control in outsourced projects but not in internal projects. In outsourced projects, the controllers likely have lesser direct formal authority to prescribe systems development methods to controllee personnel [10]. Thus the controllee might not perceive the controller's authority to prescribe methods and procedures as legitimate, and might choose not to follow them knowing that it would be difficult to detect noncompliance. In contrast, controllers have more direct formal authority in internal projects, which allows them to more readily realize behavior control. This ensures that the controllee's work fits the controller's needs, enhancing systems development performance. In summary, the greater difficulty in monitoring the controllee's compliance with prescribed procedures in outsourced projects and its relative ease in internal projects will hinder realization of behavior control in outsourced projects but not in internal projects. We therefore expect behavior control to enhance systems development performance internal projects but not in outsourced projects, independent of the extent to which it is attempted. This leads to our next set of hypotheses.

***Hypothesis 2c.** Greater use of behavior control positively affects systems development performance in internal projects.*

***Hypothesis 2d.** Greater use of behavior control does not influence systems development performance in outsourced projects.*

Clan Control and Systems Development Performance

Clan control is an attempt to supplement formal control mechanisms, particularly when the controller doubts their effectiveness as safeguards against controllee opportunism [24]. However, realizing clan control requires that the controller and controllee embrace similar values, beliefs, goals, and problem-solving approaches [24], and that sanctions be imposed by the controller on the controllee for noncompliance with these purportedly-shared norms [34]. These social

requirements are less likely to be met in outsourced projects because the controller and controllee are less likely to embrace shared values, beliefs, organizational cultures and norms, preferences, and practices that are necessary to realize clan control [12]. Furthermore, it might be difficult for the controller to impose its own values on the controllee or to detect noncompliance with the prescribed norms for the threat of sanctioning to be credible. Consistent with our position, exploratory studies have also shown that clan control is difficult to exercise in outsourced projects [10].

But these problems also exist in internal projects, though less pronounced than in outsourced projects. For example, disagreements in priorities, objectives, and goals among IT and line functions even within the same organization have been extensively documented [42]. The controller and controllee in internal projects are heterogeneous departments that usually do not have an integrated set of goals and objectives [42]. This notion is the foundational premise of the research stream on IT-line goal alignment [9]. Thus, while controllers might attempt to use clan control as a supplementary mechanism for applying social pressure on the controllee, the social requirements for clan control are seldom met even in internal projects. We therefore do not expect clan control to enhance systems development performance in internal or outsourced projects.

Self Control and Systems Development Performance

Attempting self control involves the controller abdicating all controller-driven control mechanisms based on the belief that the controllee will act in the best interests of the controller without overt controls. Self control therefore requires that the controller grant autonomy to the controllee without imposing any other forms of control. and can be interpreted as choosing non-control while rejecting other forms of controller-exercised control. (We subsequently confirm this assertion in our analyses.) Granting such autonomy requires a certain degree of trust in the

intentions of the controllee, which can be viewed as the primary social prerequisite for realizing self control. In internal projects, even when the controller and controllee lack an integrated set of objectives, the controllee is less likely to act in opportunistic ways that would hurt the controller's organization. Granting autonomy to the controllee in internal projects is therefore more likely to enhance systems development performance because it allows the IT department greater autonomy in using its technical expertise without interference to complete the project in a manner that it believes will best satisfy the controller's needs. In contrast, the objectives of the controller and controllee are likely to be more divergent in outsourced projects. Abdicating controller-driven control mechanisms is more likely to increase the controller's vulnerability to controllee opportunism, and the likelihood that the controller will trust the controllee to act in its best interests is much lower. Therefore, the social prerequisites for realizing self control are less likely to be met in outsourced projects. Some support for this perspective is offered by Choudhury and Sabherwal's [10] case study of outsourced projects, where controllers who had relied heavily on self control felt that it was a mistake to do so because it led to poor development performance. Therefore, we expect self control to be negatively associated with systems development performance in outsourced projects. These arguments lead to our next two hypotheses.

Hypothesis 2e. Greater use of self control positively affects systems development performance in internal projects.

Hypothesis 2f. Greater use of self control negatively affects systems development performance in outsourced projects.

The Moderating Role of Requirements Volatility

A key source of uncertainty in systems development projects is volatile requirements [40].

Requirements volatility is defined as the extent to which project requirements unpredictably change over the course of the systems development lifecycle [2]. This is a common problem in

practice because changing business needs or competitive pressures can introduce new requirements during the development process [31]. Controls are intended to increase the predictability of the systems development process and project outcomes. As requirements volatility increases, greater controllee flexibility is necessary to ensure that the project is adapted to evolving requirements. However, some control mechanisms might conflict with the controllee's need to be flexible whereas others might facilitate it. The rate and unpredictability of change in project requirements can therefore differentially suppress or strengthen (i.e., moderate) control-performance relationships in internal and outsourced projects, as discussed next.

Outcome control relies on the controller's assessment of controllee outputs based on an unambiguous, prespecified set of criteria for what the controllee ought to accomplish [11, 24]. Such outcome targets must be stable for the controller to judge whether the controllee has achieved them. In other words, stable requirements are necessary for systems development to proceed according to plan [36]. However, requirements volatility exacerbates prespecifying a set of stable requirements because it is difficult to know *ex ante* what the controllee ought to produce. This also makes it difficult to accurately estimate the time and development effort that a project will require [38]. Outcome controls dull the controllee's incentives to be flexible when requirements are volatile, much like formal contracts that are ill-suited to environmental change due to their inflexibility [7]. Greater requirements volatility leaves the controller with a dynamic set of tasks to perform, raising the difficulty of meeting goals established at the outset of the project. The controllee might even be reluctant to adapt the project to changing requirements because it might lower its performance on outcome metrics anchored to requirements defined at the outset. Therefore, greater requirements volatility increases the difficulties associated with measurement and evaluation of controller outputs on predefined criteria which is necessary to realize outcome control.

Another pitfall of outcome control is that the controllee may focus so much attention on meeting the predefined metrics that adapting to evolving project requirements becomes secondary. Instead, the controllee might focus solely on ensuring that predefined outcome targets are met even when they no longer capture the controller's evolved needs [12]. This amplifies the risk of delivering a technically sound system that does not meet the controller's needs, which might subsequently necessitate additional correctional rework in later stages of the systems development process. Thus an emphasis on outcome controls under high requirements volatility is likely to decrease systems development performance in internal and outsourced projects alike. This leads to our next set of hypotheses.

***Hypothesis 3a.** Outcome control negatively affects systems development performance as requirements volatility increases in internal projects.*

***Hypothesis 3b.** Outcome control negatively affects systems development performance as requirements volatility increases in outsourced projects.*

Behavior control relies on the controller's assessment of controllee performance based on adherence to the prescribed systems development methods and processes. While volatile requirements in internal projects make it difficult to manage the project, greater use of agreed upon methods and procedures allow the controller and controllee to coordinate changing requirements. The monitoring associated with behavior control further allows the controller to detect inconsistencies between the evolved requirements and intermediate system functionality and features. This allows problems to be detected and corrected before they propagate to the later systems development stages (where they are costlier to fix). Therefore, we expect requirements volatility to strengthen (positively moderate) the positive effect of behavior control on systems development performance in internal projects. However, in outsourced projects unlike internal projects, behavior control is difficult to realize due to the difficulty in monitoring controllee behavior across an interorganizational boundary and weak direct authority of the controller over

the controllee as previously discussed. Therefore, we expect requirements volatility to strengthen the positive effect of behavior control on performance in internal projects but not in outsourced projects. This leads to our next set of hypotheses.

***Hypothesis 3c.** An increase in requirements volatility will strengthen the positive relationship between behavior control and systems development performance in internal projects.*

***Hypothesis 3d.** An increase in requirements volatility will not affect the relationship between behavior control and systems development performance in outsourced projects.*

Clan control was theorized to be ineffective in improving systems development performance in internal and outsourced projects because its social requirements are unlikely to be met in either case. An increase in requirements volatility increases ambiguity about project outcomes. As such ambiguity increases, social sanctioning mechanisms remain ineffective because social sanctions cannot reliably be matched to opportunistic behavior [7]. We therefore have little theoretical basis for expecting clan controls to influence performance as requirements volatility increases in either internal or outsourced projects.

Self control grants autonomy to the controllee to self manage the systems development process, provided the controller can predefine a clear set of expectations for the controllee [24]. However, increasing requirements volatility, by making it more difficult for the controller to stably prespecify desirable project outcomes, impedes realization of self control in internal projects, suppressing its positive effect on systems development performance. We therefore expect requirements volatility to suppress the positive effect of self control on systems development performance in internal projects.

In outsourced projects, we hypothesized self control—which implies abdication of formal control by the controller—to negatively influence systems development performance. Our premise was that the social preconditions for self control are unlikely to be met in outsourced

projects, where it would simply create conditions conducive for controllee opportunism. This problem is likely to be worsened by an increase in requirements volatility, which should then strengthen the negative relationship between self control and systems development performance. This is theoretically consistent with the broader perspective that the effectiveness of relational contracts (which self control represents) progressively declines as ambiguity about expected outcomes increases [7]. We therefore expect requirements volatility to strengthen the negative effect of self control on outsourced systems development performance but suppress its positive effect in internal projects. This leads to our final set of hypotheses.

***Hypothesis 3e.** An increase in requirements volatility will strengthen the extent to which self control decreases systems development performance in outsourced projects.*

***Hypothesis 3f.** An increase in requirements volatility will suppress the positive effect of self control on systems development performance in internal projects.*

METHODOLOGY

Data on 136 systems development projects were collected from 136 organizations using a large-scale survey of IT project managers and managers of the project's user group. Two surveys were conducted, one focused on internal projects and another on outsourced projects. Two random samples of 500 organizations each were drawn from the Dun and Bradstreet directory of corporate executives for the initial mailing. In internal projects, the controller was the functional end-user department for which the system was primarily developed and the controllee was the internal IT unit. In outsourced projects, the controller was the internal IT unit and the controllee was the vendor to which the project was outsourced. To mitigate the threats of common methods bias, we attempted to collect performance data at the project level both from controllees and controllers using a two-phase, multi-informant approach. In internal projects, the controllee respondent was the MIS manager from the internal IT department and the controller respondent was the manager of the line function for which the application was primarily developed. In

outsourced projects, the controllee was the lead project manager in the vendor organization to which the project was outsourced and the controller was the IT department in the client organization. We obtained responses on 59 (57 usable) outsourced projects (response rate: 19.6%²) and on 80 (79 usable) internal projects (response rate: 19.1%³). All projects were completed within the last twelve months to ensure that the respondents could reliably respond to the survey. We then requested follow-up systems development performance assessments from the user departments for which the application was intended, obtaining 42 such assessments for the outsourced subsample (response rate: 71.1%) and 44 for the internal subsample (response rate: 55.7%). Performance assessments of both matched-pair informants for the projects in the sample were cross-validated to verify inter-rater agreement using the interclass coefficient [29]. Post-hoc tests comparing the early (first 20) and late (last 20) respondents on all key independent variables for both subsamples provided assurance against nonresponse bias.

Measures

Existing scales were adapted to the study context, all of which were multi-item, 7-point reflective Likert scales operationalized at the project level (see Appendix A). Measures for the four control mechanisms and controller knowledge of the systems development process were adapted from Kirsch et al. [24]. Systems development performance was measured using a 9-item scale adapted from Faraj and Sproull [15]. The scale items assessed the controllee's work using items such as adherence to budgets and schedules, work quality, meeting project goals and objectives, and overall efficiency and effectiveness of the delivered software. We adapted Nidumolu's [31]

² A random sample of IT function managers in 500 US organizations from Dun and Bradstreet's Million Dollar Directory was contacted to identify organizations that had existing or previous outsourcing relationships. Of these 131 were not eligible because of lack of prior experience with software services outsourcing; 68 were unreachable. Of the remaining 301 organizations, 59 responded for a conservative 19.6% response rate that compares favorably to other studies of mid-level managers. Following this phase, alliance performance assessments from a second respondent in the 59 organizations were requested. 42 responses were obtained, representing a 71.1% response rate.

³ 73 were undeliverable and 13 organizations were unable to participate due to company policy, leaving 413 possible respondents.

requirements instability scale to measure requirements volatility. The measures and sources of the control variables are shown in Appendix A. The scales were pretested and refined using a convenience sample of nine software project managers and seven academic experts. Standard psychometric techniques were used to validate all measures. Factor analyses confirmed discriminant validity among the various constructs, and scale alphas of 0.72 and greater and Eigenvalues >1 confirmed convergent validity [6]. (Principle component extraction with varimax rotation was used; details of this analysis are summarized in Appendix B.) The model was robust in an alternative formative specification of systems development performance, which was significantly and strongly correlated with the construct score and had small variance in item weights when tested in SmartPLS 2.0 with a bootstrap size of 1,000.⁴ The inter-construct correlations, means, standard deviations, and alphas for the constructs are summarized in Table 3.

<< INSERT TABLE 3 HERE >>

On average, the projects in the study involved teams of 5.75 (sd 7.5) individuals and their duration was 8.4 months (sd 7.8 months). The respondents were highly experienced, with an average of 17.6 years (sd 8.6 years) of IT experience. The controller-side respondents were also highly experienced with outsourcing, with an average prior direct experience with 21 (sd 27) outsourced projects. Consistent with our argument that self control can be viewed as non-use of

⁴ We set up a PLS model in which we used the averaged-items construct score used in our regression and a 9 item formative construct with the nine performance items. We estimated the formative model using a bootstrap of 1,000 to assess how strongly the formative measure and the performance construct score computed by averaging items correlated. A weak correlation would indicate that the results would change and a strong correlation would indicate potential robustness. The nine measurement items had strikingly similar weights—all in the .11 to .14 range—on the performance construct, and all item loadings were significant at the 0.001 level (with T-statistics ranging from 33.4 to 107.1). We then estimated the correlation between the latent formative variable and the averaged item factor score used for regression. Correlation between construct averaged-item score and formative estimate in PLS with a bootstrap of 1,000 was 1.0 (rounded by PLS) significant at $p < 0.001$. This strong correlation suggests that a formative measure and the conventional construct score produce identical factor scores. We repeated the same procedure with a reflective specification and got comparable results ($p < 0.001$). We then replaced the performance variable with a construct score computed using formative weights. The results remain unchanged except for very slight changes in betas. Therefore, the results in the model tests are unchanged by the formative or reflective specification using PLS. The results are therefore robust to reflective or formative conceptualizations of performance; we therefore followed the conventional practice of using construct scores in the regression analyses.

formal control, self control is negatively correlated with both outcome control and behavior control in the correlation matrix in Table 3.

ANALYSIS AND RESULTS

Hypothesis 1_{a-d} predicted differences between internal and outsourced projects in the extent to which controllers attempt to use the four control mechanisms. We used a one-way ANOVA test to assess the differences in the means for each control mechanism in outsourced versus internal projects. As the results in Table 4 show, significantly higher levels of outcome ($F = 19.96$; $p < 0.001$), behavior ($F = 16.46$; $p < 0.001$), and clan controls ($F = 38.79$; $p < 0.001$) were attempted in outsourced projects, while internal projects relied more on self control than outsourced projects ($F = 19.89$; $p < 0.001$). The differences were statistically significant, supporting hypotheses 1a through 1d.

<< INSERT TABLE 4 HERE >>

Hypotheses 2_{a-f} predicted the effects of the four control mechanisms on systems development performance and Hypotheses 3_{a-f} the moderation of these relationships by requirements volatility. Two identical three-step hierarchical regression models were used to test Hypotheses 2 and 3, one for internal and the other for outsourced projects. Control variables were added in step 1, the main effects to test Hypothesis 2 in step 2, and finally four interaction terms⁵ to test Hypothesis 3 in step 3. The results are presented in Table 5.

<< INSERT TABLE 5 HERE >>

The main effect of outcome control was nonsignificant for internal projects ($\beta = .17$, T -value=1.48, $p < 0.05$) and outsourced projects ($\beta = .14$, T -value=1.06, ns). Therefore, Hypothesis 2a and 2b were not supported. The main effect of behavior control was positive and significant

⁵ The simultaneous analysis of main effects and interaction terms in the regression models in step 3 distorts the coefficients for the main effects variables because they tend to be highly correlated with the interaction terms [20]. We therefore used Lance's [26] residual centering technique. In this two-stage procedure, each interaction term is regressed on its component parts and the resulting residual is used instead of the interaction term in estimating the full model. This procedure reduces multicollinearity between the interaction terms and main effects, yielding a regression coefficient for the cross-product term that can be directly interpreted as the effect of the interaction term.

for internal projects ($\beta = .28$, T-value=2.45, $p < 0.01$), supporting Hypothesis 2c. It was nonsignificant for outsourced projects, supporting Hypothesis 2d. The main effect of clan control was nonsignificant as expected. The main effect of self control was positive and significant for internal projects ($\beta = .32$, T-value=3.07, $p < 0.01$) and negative and significant for outsourced projects ($\beta = -.31$, T-value=-2.32, $p < 0.05$). Therefore Hypotheses 2e and 2f were supported.

The interaction terms were used to assess the remaining moderation hypotheses. Figure 1 illustrates the three significant interaction effects in Table 5. The values are mean centered. High (the solid lines) and low (the dotted lines) lines in the interaction plot represent ± 2 standard deviations from the mean (the patterns were consistent but less visually pronounced for ± 1 standard deviations). The interpretation of interaction effects plots relies on comparing the slope rather than absolute values of the relationship for varying levels of the moderator [13].

<< INSERT FIGURE 1 HERE >>

The interaction between outcome control and requirements volatility had a negative and significant effect on performance ($\beta = -.27$, T-value=-1.79, $p < 0.05$) in internal projects, supporting Hypothesis 3a. Figure 1 panel (a) illustrates this moderation effect. Under high requirements volatility, the relationship between outcome control and systems development performance is negatively sloped, supporting the observed relationship. The interaction between outcome control and requirements volatility was nonsignificant in outsourced projects, failing to support Hypothesis 3b. The interaction between behavior control and requirements volatility had a significant, positive effect on performance ($\beta = .25$, T-value=1.83, $p < 0.05$) in internal projects, supporting Hypothesis 3c. Figure 1 panel (b) illustrates this moderation effect. Under high requirements volatility, the slope of the line representing the relationship is steeper compared to the less steep slope under low requirements volatility. Requirements volatility did not change the already nonsignificant relationship between behavior control and performance in outsourced

projects ($\beta = .05$, T-value= .23, ns), supporting Hypothesis 3d. The interaction between self control and requirements volatility had a negative and significant effect on systems development performance in outsourced projects ($\beta = -.33$, T-value=-2.21, $p < 0.05$), supporting Hypothesis 3e. Figure 1 panel (c) illustrates this relationship. Under high requirements volatility, the relationship is negatively sloped whereas it is slightly positively sloped under low requirements volatility. Hypothesis 3f proposed that requirements volatility would suppress the positive effect of self control on in internal projects. Such suppression would occur if the positive relationship observed in Hypothesis 2e becomes weaker or disappears altogether. The interaction term was nonsignificant ($\beta = .08$, T-value= .8, ns), supporting Hypothesis 3f. Both internal and outsourced models were statistically significant at the $p < 0.01$ level in each regression step, with model F-values ranging from 3.32 to 5.41.⁶ The model explained 47.6% of the variance in internal projects and 55% in outsourced projects.

Rival Explanations

Five control variables were used to account for rival explanations. First, the controller's knowledge of the systems development process or what control theory calls knowledge of the transformation process can influence systems development performance [10, 24]. Second, the

⁶ As a further robustness check, we added each interaction term separately in the outsourced and internal models to assess the significance of the change in explained variance from adding individual interaction terms. Following the procedure outlined by Carte and Russell [8], we evaluated eight sets of regression runs, with only one interaction term (control mechanism*requirements volatility) added in Step 3 in Table 5. The model remained significant at $p < 0.05$ or better for each of the eight regressions. The change in R^2 and the corresponding F-statistic for the interaction term of each control mechanism and requirements volatility was as follows. For outsourced projects: outcome control ($\Delta R^2 = .2\%$, F-change .120, ns); behavioral control ($\Delta R^2 = .3\%$, F-change .241, ns); clan control ($\Delta R^2 = 2.4\%$, F-change 1.97, $p < 0.05$); and self control ($\Delta R^2 = 6.4\%$, F-change 5.64, $p < 0.001$). For internal projects: outcome control ($\Delta R^2 = 4.4\%$, F-change .044, ns); behavioral control ($\Delta R^2 = 2.2\%$, F-change 2.52, $p < 0.01$); clan control ($\Delta R^2 = .7\%$, F-change .763, ns); and self control ($\Delta R^2 = .4\%$, F-change .423, ns). The results are mostly consistent with Table 5 although caution must be exercised in interpreting ΔR^2 since each interaction term is evaluated in the absence of other interaction terms in the model. Although Carte and Russell [8] recommend this procedure in addition to examining beta values, it conflicts with Jaccard and Turrissi's [20: 65] guidelines to avoid under specification of interaction models. Therefore, we evaluated ΔR^2 for Step 3 in Table 5. The change from adding the four interaction terms was statistically significant for outsourced projects (F-change = 1.74; $p < 0.05$) and marginally significant for internal projects (F-change = 1.5; $p < 0.1$). Both models were statistically significant in Step 3 at the $p < 0.001$ level.

relative complexity and relative size of the project can influence performance [4]. Requirements analyzability, or the extent to which the requirements elicitation process can be reduced to a series of objective, mechanical steps can affect performance [31]. Finally, requirements diversity, defined as the extent to which users differ in their requirements for the project, can also affect performance [31]. Additional control variables such as project technological uncertainty, environmental uncertainty, controllee team member count, residual risk [31], and use of a formal systems development methodology [15] were also considered in the model but were subsequently dropped for lack of significance. Four of the five control variables were significant for either or both subsamples. The control variables respectively explained 15.9% and 29.8% of the variance in internal and outsourced projects.

DISCUSSION AND IMPLICATIONS

This study's objective was threefold: (1) to examine how controllers vary in their attempt to use different control mechanisms in internal and outsourced projects, (2) controls' effects on systems development performance, and (3) the moderating role of requirements volatility in these relationships. We theorized that the motivations for attempting control are different from the requirements for realizing it. Our theory development emphasized that attempted control is motivated by anticipated transaction hazards but realizing control requires meeting specific informational and social prerequisites. We theoretically developed three underlying ideas that contribute to this overarching perspective. First, controllers place greater emphasis on controller-driven control mechanisms in outsourced projects and on controllee-driven control mechanisms in internal projects. Second, we introduced the notions of attempted and realized control to explain how each control mechanism influences systems development performance. Third, we developed an explanation for how requirements volatility alters these control-performance relationships. Our results support these ideas. The overall pattern of results paradoxically reveal

that controllers attempt to exert more control in outsourced projects relative to internal projects, yet control enhances systems development performance in internal projects but not in outsourced projects. These findings represent three contributions with considerable implications for research.

Contributions and Implications for Research

Our first contribution is explaining the contrasts in how controllers attempt to exert control in internal projects relative to outsourced projects, an issue that no prior studies have directly examined. We demonstrated that controllers attempt to use controller-driven control mechanisms (outcome, behavior, and clan control) to a greater degree in outsourced projects relative to internal projects. Controllers also attempt to use controllee-driven control mechanisms (i.e., self control) to a greater degree in internal projects relative to outsourced projects. These findings represent a novel contribution to the IS controls literature, complementing prior studies that predict control mechanism choices exclusively in internal projects [e.g., 23, 24] or outsourced projects [e.g., 10, 39]. They also provide direct evidence for Anderson's [1] assertion that organizations will attempt to align control structures with transaction hazards.

A greater perceived risk of opportunism in outsourced projects relative to internal projects, we showed, leads to a stronger inclination of controllers to attempt to control the systems development process in outsourced projects. In contrast, these perceived agency hazards are relatively less pronounced in internal projects, thus the greater use of self control, which grants the controllee greater autonomy. However, the lower emphasis on self control in outsourced projects relative to internal projects runs contrary to Choudhury and Sabherwal's [10] expectation that controllers, having vetted the controllee's technical skills through careful upfront screening in outsourced projects, will grant them greater autonomy. Our results show that although controllers allow some self control in outsourced projects, they are likely to

provide an internal IT department greater autonomy than outsiders. Controllers in outsourced projects in our study had significantly greater technical knowledge (mean 4.42) relative to controllers in internal projects (mean 2.23; F-test for differences = 95.23; $p < 0.001$), consistent with our theoretical assertion. Greater controller technical knowledge therefore encourages greater attempted control in outsourced projects.

These results reconcile what appear on the surface to be contradictions across prior studies. For example, Kirsch et al.'s [24] study of internal projects observed greater reliance on formal controls whereas Choudhury and Sabherwal's [10] study of outsourced projects observed greater reliance on informal controls. These studies focused exclusively on internal and external projects respectively, and are not in disagreement considering that we found both formal and informal controls to be used extensively in both internal and outsourced projects. Furthermore, it appears that Kirsch et al observed attempted control whereas Choudhury and Sabherwal observed realized control, and attempting to compare them confounds the attempted-realized distinction introduced here. Overall, these findings complement prior studies in showing how perceived agency concerns and controller knowledge differences in internal and outsourced projects influence the relative emphasis that controllers put into attempting control over the systems development process.

Our second contribution is in empirically showing a paradoxical disconnect between attempted control and how it affects systems development performance: Control does not enhance performance in outsourced projects as it does in internal projects, yet controllers attempt it to a greater degree in outsourced projects than in internal projects. Our underlying explanation was that even when control is attempted, it is more difficult to realize in outsourced projects because the necessary social and informational requirements are less readily met relative to internal projects. The overarching implication of these findings is that the pervasive assertion in

the IS literature that control enhances performance is largely valid in internal projects but fails to hold up in outsourced projects.

By establishing the link between control and performance, we directly address Kirsch et al.'s [24] concern about scant empirical evidence for it. The overall pattern of results supports our idea that the realization of rather than the attempt to use a control mechanism influences systems development performance. The extent to which the informational (social) requirements specific to each formal (informal) control mechanism are met depends on whether a control relationship spans an intraorganizational or interorganizational boundary. By recognizing the nuanced differences in control-performance relationships across internal and outsourced projects, we draw a generalizability boundary condition around Henderson and Lee [17] who studied only formal controls in internal projects. Our findings therefore complement prior studies of internal [e.g., 23, 24, 32] and outsourced projects [e.g., 10, 39] that predict controls' choice rather than their impact on performance. Our findings about each of the four control mechanisms' relationship with systems development performance has important implications for research, as discussed next.

Outcome control—contrary to our expectations—did not enhance performance in outsourced projects. One interpretation is that the market-based nature of outsourced projects and a realistic threat of vendor interchangeability [27] might obviate the need for outcome control because a controllee's prospects for obtaining future contracts provides a powerful disincentive from behaving opportunistically [12, 30]. Further, the controllee being a specialized software firm has stronger, market-driven incentives to adjust to changing controller needs, therefore making formal control less critical [12]. This implies that the positive association between outcome control and performance observed in internal projects by Henderson and Lee [17] does not generalize to outsourced projects, and could not be replicated in the context of internal

projects in our study. A plausible explanation for the latter result is that the threat of a penalty imposed by outcome controls cannot sufficiently and credibly motivate an internal IT department because it is a part of the same organization as the client department. Additionally, the client, being a non-IT department, might not be able to readily evaluate (i.e., realize outcome control) the vendor on outcome control metrics of a technical nature.

Behavior control enhanced performance only in internal projects, as predicted. This implies that the difficulty of monitoring controllee behavior across an interorganizational boundary and weaker direct authority of the controller to dictate methods and procedures to the controllee makes it more difficult to realize behavior control in outsourced projects relative to internal projects. This finding adds a boundary condition to the widely accepted idea in the user involvement literature [e.g., 3, 16] that active involvement of a system's future users enhances project outcomes. As expected, clan control did not influence performance in internal or outsourced projects, supporting the view that the social requirements for realizing it are unlikely to be met in either context irrespective of how much the controller attempts to use it.

Finally, self control enhanced performance in internal projects but lowered it in outsourced projects, as predicted. This supports our idea that the social requirements for realizing self control are more readily met in internal projects. In contrast, attempting self control in outsourced projects relinquishes formal control, potentially encouraging undetectable opportunism by the controllee that can impair systems development performance. An important theoretical implication of this result is that high role expectations can lead controllers to attempt greater self control in outsourced projects, yet such projects are less conducive for realizing self control. For example, in their multi-case study of six outsourced projects, Choudhury and Sabherwal [10] observed clients' high role expectations for vendors to exercise self control, which led clients to encourage considerable self control. It also appears intuitively reasonable

that controllers would not try to dictate to a knowledgeable, carefully-screened vendor how to develop the system. Yet controllers in their study subsequently acknowledged that relying on self control was a mistake. The prevalence of attempted self control without it being realizable in outsourced projects might further imply that controllers intend to use it more as a signal to discourage opportunism without expectations of it enhancing performance. Nevertheless, it is plausible that the potential benefits of self control might not be captured in terms of our narrow effectiveness and efficiency view of performance. Self control, by signaling controller's good faith and confidence in the controllee, might motivate the controller to be more flexible or agile in accommodating controller requests outside the scope of the contract. (We discuss this in our ideas for future work.) In the broader context of organization theory, this finding provides direct evidence for Anderson and Dekker's [1] idea that misfits in control in market transactions exacts performance penalties.

Given that control mechanisms are costly for the controller to design and implement [21], why do controllers still attempt to use them in outsourced projects if they do not enhance performance? Our results offer a simple explanation for this paradoxical disconnect by showing how anticipated agency hazards motivate attempted control, which is different from meeting specific informational and social requirements for realizing it. Anticipated agency hazards are likely to be higher in outsourced projects relative to internal projects, and the informational requirements for formal controls and social requirements for informal controls are less likely to be met in the former context. This might explain the observed disconnect in why controllers attempt to use control mechanisms that subsequently do not influence outsourced systems development performance.

Our third contribution is in showing how requirements volatility moderates the control-performance relationships because it interferes with the informational (social) requirements for

realizing formal (informal) control. This is a departure from prior work where requirements volatility, though widely observed in practice, has not been incorporated in theory development. Our results show that greater requirements volatility in internal projects strengthens the positive effect of behavior control on performance (see Figure 1 panel b) but weakens the effect of outcome control. In fact, outcome control becomes detrimental rather than beneficial as requirements volatility increases (see Figure 1 panel a). This finding complements and draws a boundary condition around Henderson and Lee's [17] unreplicated observation from internal projects that outcome control improves performance. This finding also squares nicely with Wang et al.'s [42] emphasis on the role of information asymmetry in controlling systems development projects. In the broader organization theory context, while these findings provide direct evidence for Carson et al.'s [7] assertion that the effectiveness of formal contracts will decrease with volatility, Carson et al, like Poppo and Zenger [37], did not distinguish the outcome and process facets of formal contracts as we do.

In outsourced projects, requirements volatility amplifies the extent to which self control is detrimental to systems development performance (see Figure 1 panel c). This implies that the potential problems created by granting the controllee greater autonomy in outsourced projects are worsened as requirements volatility increases. Conversely, self control enhances systems development performance in outsourced projects when requirements are stable. Paradoxically, controllers are even more likely to attempt to use self control in outsourced projects with greater requirements volatility for two reasons. First, careful upfront screening increases their confidence in the controllee's abilities [10], encouraging them to increase controllee autonomy. Second, controllers will increase their reliance on self control as a substitute for formal control, which becomes more difficult to attempt as volatile requirements makes prespecification of desirable outcomes and behaviors more challenging [17]. Thus conditions that lead the controller

to increase dependence on self control also magnify its negative effects. In a broader organization theory context, these results warrant caution against the recent idea that the optimal response to uncertainty is less formal contracting and more reliance on informal/ relational contracts [7]. Instead, our results imply that relational forms of control such as clan and self control are ineffective in breaking the link between volatility and performance vulnerability in outsourced projects. These results also imply that Carson et al.'s [7] suggestion to use outcome controls when behaviors cannot unambiguously be measured can be an ineffective strategy in outsourced projects, and can be detrimental in internal projects with volatile requirements. The overarching implication of the study is that controllers might attempt greater control using mechanisms that are less viable in outsourced projects.

Limitations

Four limitations of this study are noteworthy. First, the cross-sectional data precludes an assessment of the causal ordering between control and systems development performance, which requires longitudinal data. Second, we introduced the distinction between attempted control and realized control in our theory development. However, the direct measurement of this distinction is required in future work to complement the theoretical distinction introduced here. Third, although several traditional rival explanations were included as control variables, outcome measurability was not included as a control variable. Fourth, controllers and controllees might have different perceptions about the extent to which various control mechanisms were attempted in a project [see 22]; our use of IT informants for this data limits our results' generalizability.

Implications for Practice

These results have five important practical implications for project managers. First, managers must recognize that the costs of attempting to control a project must be weighed against their realizable benefits. Specifically, since control mechanisms effective in internal projects are not

effective in outsourced projects, managers should not attempt to manage outsourced projects as they would manage internal projects. Second, although most control mechanisms enhance performance in internal projects, these benefits are less likely to yield enhanced performance in outsourced projects. Therefore managers must temper a seemingly-rational inclination to attempt greater control in outsourced projects because of the difficulty in exercising it in practice. Third, granting greater autonomy to the controllee (self control) enhances systems development performance in internal projects but impedes it in outsourced projects. Therefore managers should rely on the project team's ability to self manage the project in internal projects but not in outsourced projects. Fourth, proponents of lean systems development methods have recently emphasized that clients should not be members of development teams due to the conflicts of interest that it creates [19: 98, 101]. This prescription runs contrary to popular beliefs among IS practitioners but is entirely consistent with our results regarding clan control (which did not enhance performance in internal or outsourced projects). Fifth, managers must factor requirements volatility in their control selection decisions. This is particularly important in internal projects because greater requirements volatility negates the potential benefits of outcome control mechanisms. Therefore, it is more appropriate for managers to rely on behavior controls when requirements are relatively unstable.

Directions for Future Research

These findings raise four promising questions for future research. First, how does attempted control relate to sharing of project decision rights between the controller and controllee? The unexplored linkages between what a project should do ("decision control rights") and how it should do it ("decision management rights") conceptualized in recent research [41] present new opportunities for theory development. Second, what interventions can overcome the impediments to meeting informational and social prerequisites in outsourced projects? Future work should

also further explore the paradoxical pattern of results spanning internal and outsourced projects wherein controllers attempt to make greater use of control mechanisms in settings where they are also less likely to be effective. A third avenue for future research is further exploring the role of self control. The range of potential benefits of self control might not be captured by our conventional performance measure. Self control signals good faith on the controller's part, which might motivate the controllee to act in the controller's interest even without other controls. In particular, self control might enhance project agility or flexibility, which should be included as a criterion variable in future studies. Thus, even though clan control does not enhance performance using it might not be as irrational as it appears to be on the surface. Furthermore, some control mechanisms might engender flexibility in ways that this study did not examine. Finally, the use of Internet-based software "build management" systems is growing. Does the use of such tools increase behavior observability that might help more effectively realize behavior control?

Overall, this study is an initial step towards refocusing theory development to when and how controls enhance systems development and away from predicting their choice.

REFERENCES

1. Anderson, S. and Dekker, H. Management Control for Market Transactions. *Management Science*, 51, 12 (2005), 1734-1752.
2. Banker, R.D. and Slaughter, S.A. The Moderating Effects of Structure on Volatility and Complexity in Software Enhancement. *Information Systems Research*, 11, 3 (2000), 219-240.
3. Barki, H. and Hartwick, J. User Participation, Conflict, and Conflict-Resolution - the Mediating Roles of Influence. *Information Systems Research*, 5, 4 (1994), 422-438.
4. Barki, H., Rivard, S., and Talbot, J. An Integrative Contingency Model of Software Project Risk Management. *Journal of Management Information Systems*, 17, 4 (2001), 37-69.
5. Benko, C. and McFarlan, W. *Connecting the Dots: Aligning Projects with Objectives in Unpredictable Times*. Boston: Harvard Business School Press, 2003.
6. Campbell, D.T. and Fiske, D.W. Convergent and Discriminant Validation by the Multitrait-Multimethod Matrix. *Psychological Bulletin*, 56, 2 (1959), 81-105.
7. Carson, S., Madhok, A., and Wu, T. Uncertainty, Opportunism and Governance. *Academy of Management Journal*, 49, 5 (2006), 1058-1077.
8. Carte, T. and Russell, C. In Pursuit of Moderation: Nine Common Errors and Thier Solutions. *MIS Quarterly*, 27, 3 (2003), 479-502.

9. Chan, Y. and Reich, B. IT Alignment - What Have We Learned? *Journal of Information Technology*, 22,(2007), 297-315.
10. Choudhury, V. and Sabherwal, R. Portfolios of Control in Outsourced Software Development Projects. *Information Systems Research*, 14, 3 (2003), 291-314.
11. Das, T. and Teng, B.S. Between Trust and Control: Developing Confidence in Partner Cooperation in Alliances. *Academy of Management Review*, 23, 3 (1998), 491-512.
12. Domberger, S. *The Contracting Organization*. Cambridge University Press, 1998.
13. Edwards, J. and Lambert, L. Methods for Integrating Moderation and Mediation. *Psychological Methods*, 12, 1 (2007), 1-22.
14. Eisenhardt, K. Control: Organizational and Economic Approaches. *Management Science*, 31, 2 (1985), 134-149.
15. Faraj, S. and Sproull, L. Coordinating Expertise in Software Development Teams. *Management Science*, 46, 12 (2000), 1554-1568.
16. Hartwick, J. and Barki, H. Explaining the Role of User Participation in Information Systems Development. *Management Science*, 40, 4 (1994), 440-465.
17. Henderson, J. and Lee, S. Managing I/S Design Teams: A Control Theories Perspective. *Management Science*, 38, 6 (1992), 757-777.
18. Herbsleb, J. and Mockus, A. An Empirical Study of Speed and Communication in Globally Distributed Software Development. *IEEE Transactions on Software Engineering*, 29, 6 (2003), 481-494.
19. Hibbs, C., Jewett, S., and Sullivan, M. *The Art of Lean Software Development*. Sebastopol, CA: O'Reilly, 2009.
20. Jaccard, J. and Turrisi, R. *Interaction Effects in Multiple Regression*. 2 ed., Thousand Oaks, CA: Sage, 2003.
21. Jensen, M. and Meckling, W. Specific and General Knowledge and Organizational Structure, In Werin and Wijkander, *Contract Economics*, Oxford, 1992, pp. 251-274.
22. Keil, M., Tiwana, A., and Bush, A. Reconciling User and Project Manager Perceptions of IT Project Risk: A Delphi Study. *Information Systems Journal*, 12,(2002), 103-119.
23. Kirsch, L. The Management of Complex Tasks in Organizations: Controlling the Systems Development Process. *Organization Science*, 7, 1 (1996), 1-21.
24. Kirsch, L., Sambamurthy, V., Ko, D., and Purvis, R. Controlling Information Systems Development Projects: View from the Client. *Management Science*, 48, 4 (2002), 484-498.
25. Kirsch, L.J. Portfolios of Control Modes and IS Project Management. *Information Systems Research*, 8, 3 (1997), 215-239.
26. Lance, C. Residual Centering, Exploratory and Confirmatory Moderator Analysis, and Decomposition of Effects in Path Models Containing Interactions. *Applied Psychological Measurement*, 12, 2 (1988), 163-175.
27. Lee, J., Miranda, S., and Kim, Y. IT Outsourcing Strategies: Universalistic, Contingency, and Configurational Explanations. *Information Systems Research*, 15, 2 (2004), 110-131.
28. McEvily, B., Perrone, V., and Zaheer, A. Trust as an Organizing Principle *Organization Science*, 14, 1 (2003), 91-103.
29. McGraw, K. and Wong, S. Forming Inferences About Some Interclass Correlation Coefficients. *Psychological Methods*, 1, 1 (1996), 30-46.
30. Nickerson, J. and Zenger, T. A Knowledge-Based Theory of the Firm-the Problem-Solving Perspective. *Organization Science*, 15, 6 (2004), 617-632.

31. Nidumolu, S. The Effect of Coordination and Uncertainty on Software Project Performance: Residual Performance Risk as an Intervening Variable. *Information Systems Research*, 6, 3 (1995), 191-219.
32. Nidumolu, S. and Subramani, M. The Matrix of Control: Combining Process and Structure Approaches. *Journal of Management Information Systems*, 20, 3 (2004), 159 - 196.
33. Ouchi, W. A Conceptual Framework for the Design of Organizational Control Mechanisms. *Management Science*, 25, 9 (1979), 833-848.
34. Ouchi, W. Markets Bureaucracies and Clans. *Administrative Science Quarterly*, 25, 1 (1980), 129-141.
35. Ouchi, W. and Maguire, M. Organizational Control: Two Functions. *Administrative Science Quarterly*, 20,(1975), 559-569.
36. Pitts, M.G. and Browne, G.J. Stopping Behavior of Systems Analysts During Information Requirements Elicitation. *Journal of Management Information Systems*, 21, 1 (2004), 203-226.
37. Poppo, L. and Zenger, T. Do Formal Contracts and Relational Governance Function as Substitutes or Complements? *Strategic Management Journal*, 23, 8 (2002), 707-725.
38. Rowen, R. Software Project Management under Incomplete and Ambiguous Specifications. *IEEE Transactions on Engineering Management*, 37, 1 (1990), 10-21.
39. Rustagi, S., King, W., and Kirsch, L. Predictors of Formal Control Usage in IT Outsourcing Partnerships. *Information Systems Research*, 19, 2 (2008), 126-143.
40. Sengupta, K. and Abdel-Hamid, T. The Impact of Unreliable Information on the Management of Software Projects: A Dynamic Decision Perspective. *IEEE Transactions on Systems, Man, and Cybernetics*, 26, 2 (1996), 177-189.
41. Tiwana, A. Does Technological Modularity Substitute for Control?. *Strategic Management Journal*, 29, 7 (2008), 769-780.
42. Wang, E.T.G. and Barron, T. Controlling Information-System Departments in the Presence of Cost Information Asymmetry. *Information Systems Research*, 6, 1 (1995), 24-50.
43. White, S. and Lui, S. Distinguishing Costs of Cooperation and Control in Alliances. *Strategic Management Journal*, 26, 10 (2005), 913-932.
44. Willis, D. Penalties and Incentives in Outsourcing Relationships. *CIO Magazine*, <http://www2.cio.com/analyst/report2481.html>,(2005).

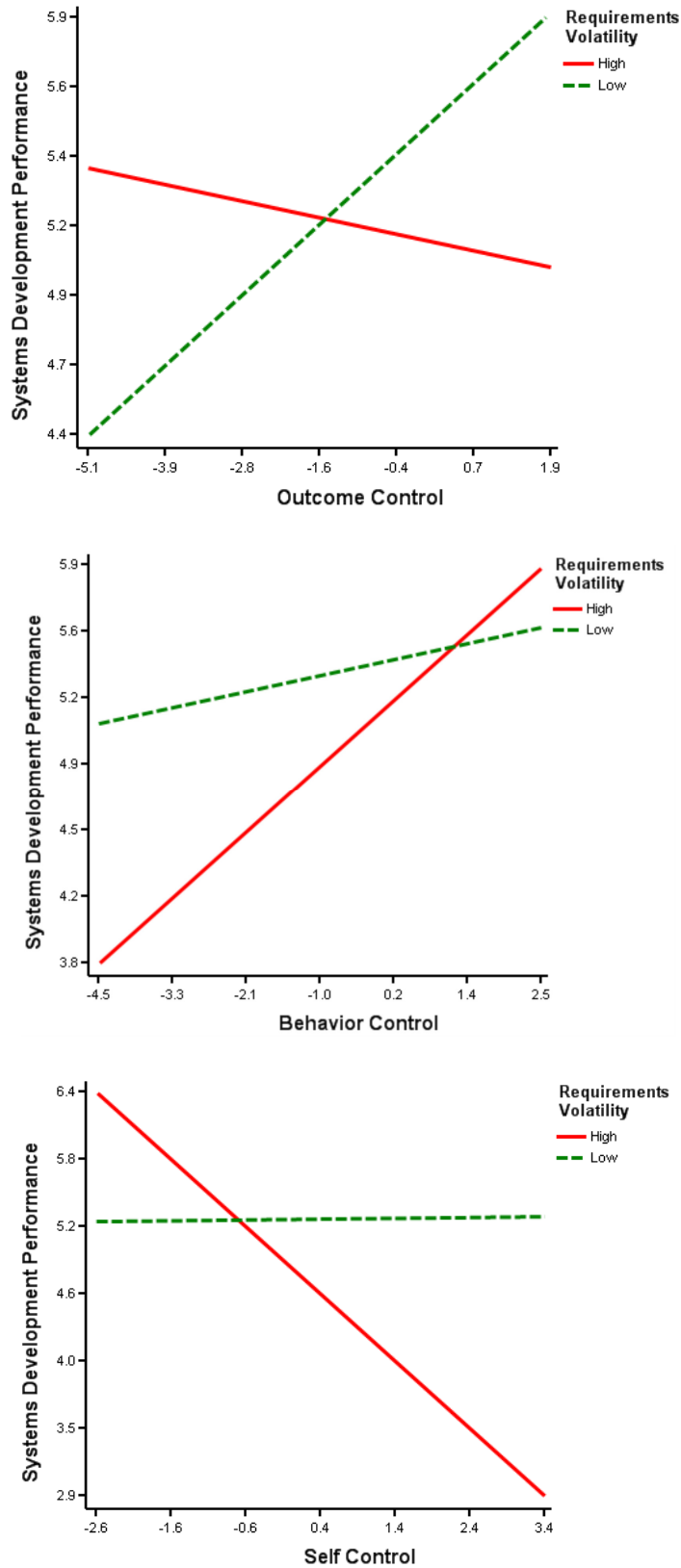


Figure 1: Interaction effects between requirements volatility and: (a) outcome control in internal projects, (b) behavior control in internal projects, and (c) self control in outsourced projects. High (low) are + (-) 2 standard deviations.

Table 1: Control in Internal and Outsourced Projects

| Controller-Controllee Characteristics | Internal Projects | Outsourced Projects |
|--|---------------------------------------|---------------------------------------|
| Boundary spanned | Intraorganizational boundary | Interorganizational boundary |
| Controllee(controller) | Internal IT department(line function) | External vendor(client IT department) |
| Controller-controllee relationship | Organizational departments | Market-based contract |
| Direct authority over controllee | Higher | Lower |
| Controllee selection criteria | None | Upfront screening [10] |
| Agency hazards | Lower | Higher |
| Shared values and beliefs | Higher | Lower |
| Controller technical knowledge | Lower (internal department) | Higher (IT department) [10] |
| Controllee behavior observability | Higher | Lower |

Table 2: Attempted versus Realized Control

| | | Requirements for... | |
|----------------------------------|----------------------------|--|--|
| Control Mechanism (exercised by) | Dominant prerequisite [33] | Attempted Control | Realized Control |
| Outcome control (controller) | Informationally-demanding | Prespecification of precise project controllee output requires the controller to possess greater technical knowledge about systems development. | Measuring whether the controllee met prespecified project goals requires measurement of controllee outputs. |
| Behavior control (controller) | Informationally-demanding | Prespecification of appropriate systems development procedures requires the controller to possess greater technical knowledge about systems development as well as formal authority to prescribe them. | Monitoring compliance with prespecified procedures requires observation/ monitoring of controllee behavior, and acceptance of the legitimacy of the controller's authority to impose methods and procedures. |
| Clan control (controller) | Socially-demanding | Shared values, beliefs, and traditions between the controllee and the controller. | Sanctioning of the controllee by the controller for noncompliance with the purportedly-shared values, beliefs, and traditions. |
| Self control (controllee) | Socially-demanding | Abdication of control over the controllee by the controller. | Granting autonomy to controllee requires trust that the controllee will act in the best interests of the controller without overt controls. |

Table 3: Construct correlations

| | mean | sd | # | alpha | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|-------------------------------------|------|------|---|-------|--------|-------|--------|-------|-------|-------|-------|--------|------|------|
| 1. Controller know of SD process | 3.13 | 1.67 | 7 | .95 | | | | | | | | | | |
| 2. Relative project complexity | 4.02 | 1.38 | 5 | .86 | -.04 | | | | | | | | | |
| 3. Relative project size | 4.54 | 1.36 | 4 | .91 | .03 | .50** | | | | | | | | |
| 4. Requirements analyzability | 4.75 | 1.15 | 4 | .75 | .16 | .02 | .01 | | | | | | | |
| 5. Requirements diversity | 3.68 | 1.34 | 3 | .72 | .13 | .32** | .42** | .06 | | | | | | |
| 6. Outcome control | 5.47 | 1.18 | 4 | .84 | .27** | .07 | .28** | .40** | .20* | | | | | |
| 7. Behavioral control | 4.90 | 1.48 | 4 | .89 | .30** | .08 | .22* | .41** | .26** | .39** | | | | |
| 8. Clan control | 4.91 | 1.52 | 4 | .94 | .43** | .05 | .22* | .31** | .20* | .51** | .48** | | | |
| 9. Self control | 4.24 | 1.58 | 3 | .77 | -.39** | .10 | -.04 | -.15 | .07 | -.17* | -.15 | -.38** | | |
| 10. Requirements volatility | 3.89 | 1.43 | 4 | .78 | -.03 | .31** | .45** | -.02 | .57** | .12 | .09 | .16 | .19* | |
| 11. Systems development performance | 5.15 | 1.18 | 9 | .96 | .09 | -.03 | -.29** | .38** | -.11 | .09 | .16 | .02 | .01 | -.09 |

***p> 0.001; **p> 0.01; *p> 0.05

Table 4: Differences in Control Strategies in Internal and Outsourced Projects

| | Internal | | Outsourced | F-value for differences |
|------------------|--------------------|---|--------------------|-------------------------|
| | Mean (SD) | | Mean (SD) | |
| Outcome control | 5.11(1.26) | < | 5.97 (.83) | 19.96*** |
| Behavior control | 4.48(1.64) | < | 5.48 (.97) | 16.46*** |
| Clan control | 4.29(1.53) | < | 5.76 (1.01) | 38.79*** |
| Self control | 4.72 (1.47) | > | 3.56(1.49) | 19.89*** |

The larger value is indicated in **bold**; *** p<0.001

Table 5: Results

| | Internal Projects (N = 79) | | | Outsourced Projects (N = 57) | | |
|--|--------------------------------|----------------------------|----------------------------------|---------------------------------|----------------------------|----------------------------------|
| | Step 1 Control variables | Step II Main effects | Step III Interaction terms | Step 1 Control variables | Step II Main effects | Step III Interaction terms |
| | $\beta(T)$ | $\beta(T)$ | $\beta(T)$ | $\beta(T)$ | $\beta(T)$ | $\beta(T)$ |
| Constant | (-9.06) | (4.81) | (3.66) | (1.46) | (1.39) | (1.27) |
| Controller knowledge of SD process | .13(1.2) | .2(1.93) | .14(1.33) | .18(1.36) | .15(1.16) | .15(1.24) |
| Relative project complexity | .21(1.5) | .15(1.18) | .17(1.29) | .09(.75) | .07(.57) | .14(1.16) |
| Relative project size | -.34(-2.22)* | -.4(-2.67)** | -.39(-2.64)** | -.25(-1.92)* | -.25(-1.92)* | -.28(-2.17)* |
| Requirements analyzability | .37*** (3.38) | .26(2.26)* | .22(1.84)* | .41(3.00)*** | .33(2.19)* | .26(1.6) |
| Requirements diversity | -.18(-1.40) | -.33(-2.54)** | -.25(-1.83)* | .09(.71) | .27(1.85) | .27(1.87)* |
| Outcome control | | .17(1.48) | .27(2.19)* | | .14(1.06) | .13(.9) |
| Behavioral control | | .28(2.45)** | .23(1.97)* | | 0(.02) | .1(.62) |
| Clan control | | -.09(-.78) | -.05(-.42) | | -.03(-.17) | .01(-.02) |
| Self control | | .32(3.07)** | .37(3.33)*** | | -.31(-2.32)* | -.29(-2.13)* |
| Requirements volatility | | .19(1.39) | .16(1.16) | | -.16(-1.06) | -.41(-1.9)* |
| Requirements volatility*Outcome control | | | -.27(-1.79)* | | | .17(.98) |
| Requirements volatility*Behavioral control | | | .25(1.83)* | | | .05(.23) |
| Requirements volatility*Clan control | | | .16(.98) | | | .04(.18) |
| Requirements volatility*Self control | | | .08(.8) | | | -.33(-2.21)* |
| R ² | 21.5% | 42.4% | 47.6% | 36.5% | 46.8% | 55.0% |
| R ² _{Adj} | 15.9% | 33.5% | 35.5% | 29.8% | 34.1% | 38.5% |
| ΔR^2 (F-change) | — | 2.9%*** (4.71) | 5.2% ⁺ (1.5) | — | 1.3% ⁺ (1.62) | 8.2%*(1.74) |
| Model F | 3.84*** | 4.79*** | 3.95*** | 5.41*** | 3.69*** | 3.32*** |

***p < 0.001; **p < 0.01; *p < 0.05; +p < 0.1 one-tailed test; significant results are in **bold**.

Appendix A: Construct Measures

Separate versions of the survey were created for internal and outsourced projects. All constructs were operationalized at the project level and the project name provided by the primary respondent was used to create a subset of matched pairs of responses from respondents in the subsequent phase of data collection. The controllee was explicitly defined to include all employees of the IT department (in internal projects) or vendor organization (in outsourced projects) who worked on this project and the controller was defined to include all employees of the user department/ client organization with whom the IT department/ outside vendor worked with. All scales were 7-point Likert scales.

Measures for all four types of control mechanisms were adapted from Kirsch et al [24].

Outcome control^b (4 items; $\alpha = 0.84$) was measured using 4 items that assessed the extent to which the controller placed significant weight on: (1) timely project completion, (2) completion within budget, (3) meeting requirements, and (4) accomplishing project goal.

Behavior control^b (4 items; $\alpha = 0.89$) was measured using 4 items that assessed the extent to which members of the controllee organization working on the project were: (1) expected to follow an understandable written sequence of steps toward accomplishing project goals, (2) expected to follow an understandable written sequence of steps to ensure this system met user department requirements, (3) expected to follow an understandable written sequence of steps to ensure the success of this project, and (4) were assessed on the extent to which they followed existing written procedures & practices during the development process by the controller.

Clan control^b (4 items; $\alpha = 0.94$) was measured using 4 items that assessed the extent to which members of the controller organization: (1) attempted to be “regular” members of the *project team*, (2) attempted to understand the *project team’s* goals, values, & norms, (3) placed a significant weight on understanding the *project team’s* goals, values, & norms, and (4) actively participated in project meetings to understand the *project team’s* goals, values, & norms.

Self control^b (3 items; $\alpha = 0.77$) was measured using 4 items that assessed the extent to which members of the controllee organization working on the project: (1) self-managed the development process, (2) set specific goals for this project without the controller’s involvement, (3) defined specific procedures for this project’s activities without the controller’s involvement.

Requirements volatility (4 items, $\alpha = 0.78$) was measured using four items that assessed the degree to which: (1) requirements fluctuated quite a bit in earlier phases, (2) requirements fluctuated quite a bit in later phases, (3) requirements will fluctuate quite a bit in the future, (4) requirements identified at beginning of project were quite different from those existing at the end. Adapted from Nidumolu’s [31] requirements instability scale.

Requirements diversity (3 items, $\alpha = 0.72$) assessed whether: (1) users of this software differed a great deal among themselves in the requirements to be met by it, (2) a lot of effort had to be spent in reconciling the requirements of various users of this software, and (3) it was difficult to customize the software to one set of users without reducing support to other users. Source [31].

Requirements analyzability (4 items; $\alpha = 0.75$) assessed whether: (1) there was a clearly known way to convert user needs to requirements specifications, (2) available knowledge was of great help in converting user needs to requirements specifications, (3) established procedures and practices could be relied upon to generate requirements specifications, (4) an understandable sequence of steps could be followed for converting user department needs to requirements specifications. Source [31].

Relative Project Size^c (4 items; $\alpha = 0.91$) How would you compare this project to other IT projects in your company? (1) Person-months of development work, (2) project duration, (3) dollar budget, (4) project size.

Relative Project Complexity^c (5 items; $\alpha = 0.86$) assessed whether the project relative to other IT projects in the respondent’s company: (1) was technically complex to develop, (2) used non-routine technology, (3) used a complex development process, (4) required pioneering innovations, and (5) was relatively complex.

Systems development performance^a (respondents: both controller and controllee; 9 items; $\alpha = 0.96$) was measured using 9 items from Faraj and Sproull [15] that assessed the controllee’s team on the following relative to other IT project teams that they were familiar with: (1) work quality, (2) team operations, (3) meeting project goals, (4) meeting design objectives, (5) work excellence, (6) overall effectiveness, (7) adherence to schedules, (8) adherence to budgets, and (9) overall efficiency.

Controller’s knowledge of the systems development process^d (7 items; $\alpha = 0.95$) was measured as the extent to which the controllee understood the following specific to the project: (a) the programming language, (b) detailed technical design, (c) technical design constraints, (d) code testing & debugging procedures, (e) development tools & coding environment, (f) effective versus ineffective development practices, and (g) the systems development process

Scale anchors

^a1=much worse; 7=much better

^b1=strongly disagree; 7=strongly agree

^c1=much lower; 7=much higher

^d1=not at all; 7=to a great extent

Appendix B: Exploratory Factor Analyses

| Factor | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|--------------------------------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| Systems development performance 9 | .947 | .018 | .007 | -.026 | -.074 | .042 | -.043 | -.039 | .109 | -.026 | -.004 |
| Systems development performance 6 | .902 | -.048 | .031 | .073 | -.087 | -.01 | .066 | -.05 | .082 | .002 | .002 |
| Systems development performance 5 | .899 | .033 | -.019 | .071 | -.084 | .043 | .065 | .061 | .009 | .045 | .012 |
| Systems development performance 3 | .886 | .001 | -.034 | -.041 | -.122 | .026 | .028 | .023 | .062 | -.036 | -.041 |
| Systems development performance 4 | .866 | .076 | -.062 | .029 | -.186 | .047 | -.029 | .015 | .113 | .034 | -.031 |
| Systems development performance 1 | .861 | .097 | .005 | .067 | .037 | .158 | .114 | .013 | .056 | -.042 | -.013 |
| Systems development performance 7 | .841 | .06 | -.025 | -.15 | -.099 | .003 | .113 | -.116 | .044 | .025 | .045 |
| Systems development performance 2 | .835 | .121 | .019 | .058 | .03 | .12 | -.035 | .117 | .101 | -.066 | -.166 |
| Systems development performance 8 | .761 | -.047 | .114 | -.073 | -.085 | -.044 | -.094 | -.161 | .115 | .092 | -.017 |
| Controller knowledge of SD process 5 | .006 | .898 | .034 | -.007 | -.031 | .133 | .049 | -.021 | -.038 | -.026 | .119 |
| Controller knowledge of SD process 7 | .127 | .889 | .167 | .038 | .014 | .028 | .079 | -.037 | .084 | -.036 | .014 |
| Controller knowledge of SD process 3 | .084 | .874 | .139 | -.057 | -.051 | .075 | .105 | .042 | -.013 | -.208 | -.004 |
| Controller knowledge of SD process 6 | .039 | .873 | .132 | .023 | .029 | .073 | .062 | .024 | .094 | -.041 | .048 |
| Controller knowledge of SD process 2 | -.012 | .859 | .196 | -.028 | -.002 | .037 | .108 | .001 | .045 | -.142 | .03 |
| Controller knowledge of SD process 1 | -.068 | .844 | -.002 | -.046 | .009 | .102 | .047 | -.02 | .05 | -.065 | -.055 |
| Controller knowledge of SD process 4 | .138 | .824 | .214 | -.023 | .066 | .071 | -.013 | -.061 | -.063 | -.059 | .055 |
| Clan Control 4 | -.037 | .24 | .828 | .024 | .04 | .203 | .135 | .096 | .108 | -.114 | .049 |
| Clan Control 1 | .041 | .258 | .8 | -.035 | .104 | .16 | .114 | -.011 | -.061 | -.151 | .097 |
| Clan Control 3 | .002 | .273 | .796 | .02 | .081 | .225 | .246 | .061 | .12 | -.06 | .016 |
| Clan Control 2 | -.04 | .239 | .765 | .056 | .027 | .227 | .295 | .052 | .162 | -.172 | -.019 |
| Relative project complexity 3 | -.006 | -.065 | .04 | .859 | .129 | .181 | .042 | .091 | -.076 | .043 | .147 |
| Relative project complexity4 | .028 | -.009 | -.037 | .795 | .189 | .038 | -.084 | .008 | -.081 | .106 | .231 |
| Relative project complexity 1 | .027 | .088 | .09 | .746 | .367 | -.038 | .031 | .215 | -.086 | -.079 | -.078 |
| Relative project complexity 2 | .06 | -.105 | -.12 | .742 | .047 | -.072 | -.025 | -.14 | .104 | .2 | .25 |
| Relative project complexity 5 | -.087 | -.021 | .146 | .68 | .326 | -.023 | .134 | .294 | .075 | -.144 | -.262 |
| Relative project size 2 | -.208 | -.053 | .101 | .261 | .819 | .03 | -.008 | .103 | .006 | .04 | .182 |
| Relative project size 4 | -.16 | .018 | .04 | .303 | .775 | .12 | .091 | .226 | .042 | -.049 | .059 |
| Relative project size 1 | -.206 | .029 | .138 | .213 | .76 | .03 | .131 | .195 | -.096 | -.023 | .183 |
| Relative project size 3 | -.184 | .059 | -.027 | .152 | .741 | .209 | .275 | .143 | -.037 | -.025 | .053 |
| Behavioral control 1 | .102 | .108 | .15 | .044 | .093 | .852 | .168 | -.051 | .146 | -.011 | .048 |
| Behavioral control 3 | .103 | .108 | .223 | .005 | .174 | .844 | .093 | -.041 | .246 | -.027 | .041 |
| Behavioral control 2 | .099 | .143 | .238 | -.035 | .186 | .792 | .135 | -.079 | .264 | -.032 | .029 |
| Behavioral control 4 | .077 | .219 | .171 | .12 | -.094 | .658 | .06 | .169 | -.028 | -.008 | .217 |
| Outcome control 1 | .095 | .097 | .147 | -.05 | .068 | .111 | .812 | .168 | .033 | .068 | -.043 |
| Outcome control 2 | -.057 | .242 | .028 | .049 | .055 | .245 | .783 | -.113 | .072 | -.032 | .082 |
| Outcome control 4 | .085 | .031 | .42 | -.007 | .173 | .153 | .705 | -.008 | .126 | -.069 | .016 |
| Outcome control 3 | .084 | .067 | .325 | .048 | .193 | -.061 | .674 | .021 | .197 | -.133 | .159 |
| Requirements volatility 2 | -.02 | -.056 | .08 | .077 | .18 | -.045 | .041 | .794 | -.067 | .028 | .19 |
| Requirements volatility 1 | .03 | -.061 | .223 | .075 | .344 | .051 | -.006 | .699 | -.008 | -.065 | .225 |
| Requirements volatility 4 | -.086 | -.012 | .104 | .159 | .07 | .058 | -.007 | .612 | -.228 | .28 | .434 |
| Requirements volatility 3 | -.066 | .007 | -.273 | .075 | .16 | -.08 | .045 | .574 | .138 | .239 | .145 |
| Requirements analyzability 4 | .263 | .074 | .079 | .033 | .099 | .249 | .057 | -.129 | .708 | -.036 | -.084 |
| Requirements analyzability 1 | .111 | .028 | .03 | -.165 | -.14 | .284 | -.022 | -.15 | .668 | .19 | .009 |
| Requirements analyzability 2 | .248 | -.066 | .169 | -.016 | .093 | -.019 | .274 | .022 | .661 | -.304 | .143 |
| Requirements analyzability 3 | .232 | .126 | .079 | .035 | -.173 | .188 | .232 | .255 | .631 | .019 | -.02 |
| Self control 3 | -.042 | -.18 | -.27 | -.001 | -.064 | .051 | .03 | .138 | -.016 | .804 | .104 |
| Self control 2 | .067 | -.167 | -.164 | .124 | .057 | .025 | -.108 | -.018 | -.048 | .801 | .003 |
| Self control 1 | .033 | -.392 | .027 | .037 | -.101 | -.268 | -.006 | .221 | .062 | .621 | 7.67E-06 |
| Requirements diversity 1 | -.097 | .113 | .146 | .026 | .156 | .144 | .108 | .281 | -.035 | .12 | .752 |
| Requirements diversity 3 | -.011 | .106 | -.108 | .315 | .049 | .013 | -.05 | .181 | .036 | -.038 | .644 |
| Requirements diversity 2 | -.087 | .008 | .109 | .099 | .255 | .156 | .138 | .238 | .047 | .003 | .643 |
| % variance explained | 19.3% | 16.2% | 11.1% | 7% | 5% | 4.6% | 3.4% | 3.1% | 2.7% | 2.4% | 1.9% |
| Eigen value | 9.8 | 8.3 | 5.68 | 3.57 | 2.54 | 2.34 | 1.75 | 1.56 | 1.37 | 1.23 | .99 |

