The Matrix of Control: Combining Process and Structure Approaches to Managing Software Development

SARMA R. NIDUMOLU AND MANI R. SUBRAMANI

SARMA R. "RAM" NIDUMOLU is the founder and CEO of Gandiva, Inc., a software company that provides Product Chain Management solutions to design-intensive industries. He was formerly an Assistant Professor of MIS at Santa Clara University and the University of Arizona. Dr. Nidumolu's current research and practice deal with the coordination and control of software development and the adoption and impacts of new information technologies. Prior publications have appeared in *Information Systems Research, Journal of Management Information Systems, MIS Quarterly, Management Science,* and *Communications of the ACM,* among others.

MANI R. SUBRAMANI is an Assistant Professor at the Information and Decision Sciences (IDSc) Department in the Carlson School of Management, at the University of Minnesota. He received his B.E. in Electrical and Electronics Engineering from BITS, Pilani, a PGDM degree from the Indian Institute of Management, Bangalore, and a DBA from Boston University. His research focuses on the strategic role of information technology within the organization and in interorganizational relationships. His current areas of research are knowledge management and the leveraging of organizational capabilities using information technologies. Dr. Subramani has published in journals such as *Academy of Management Journal, Communications of the ACM, Information Systems Research, Journal of Management Information Systems,* and *Sloan Management Review.*

ABSTRACT: The performance of firms in the software industry depends considerably on the quality of their software development processes. Managing software development is a challenging task, as management controls need to impose discipline and coordinate action to ensure goals are met while simultaneously incorporating autonomy to motivate software developers to be innovative and produce quality work. How should such firms manage software development projects so that their development processes are flexible and predictable—resulting in products that meet quality goals and that are delivered within budget and on time? The management literature suggests two approaches to control: the process approach and the structure approach. The process approach recommends control of activities through specifying methods (behavior control) and through specifying performance criteria (outcome control). In contrast, the structure approach recommends control through centrally devised standards for activities (standardization) and by the delegation of authority for decision-making (decentralization). This study synthesizes these two approaches to suggest that formal managerial control is exerted through a matrix of control comprising four modes: *standardization of methods, standardization of performance criteria, decentralization of methods,* and *decentralization of performance criteria.*

Journal of Management Information Systems / Winter 2003–4, Vol. 20, No. 3, pp. 159–196.

0742–1222 / 2004 $9.50 + 0.00.

We test the association of the modes of control with performance in a sample of 56 firms in the software industry in the United States. The results suggest that performance is enhanced by establishing uniform performance criteria across projects (*standardization of performance criteria*) while giving each project team the authority to make decisions with respect to methods (*decentralization of methods*). However, standardization of methods across all projects and decentralization of performance criteria by delegating the authority to make decisions about performance criteria to project teams were both not significantly related to performance. The matrix of control and its relationship to performance has theoretical and practical implications for managing software development. This model of control is also likely to be useful in other knowledge-work–intensive settings.

KEY WORDS AND PHRASES: decentralization, organizational control, process performance, software development, software project management, standardization.

DEVELOPING SOFTWARE IS A COMPLEX, knowledge-intensive activity. The environment confronting software firms is marked by extreme competition and uncertainty; periods of stability are rare, and technological and market disruptions are the norm. Control strategies that align the actions of software development teams with the changing strategies of software firms are therefore important to firm performance. Devising the appropriate mechanisms of formal[1] control involves trade-offs between often conflicting requirements: (1) enforcing discipline and ensuring uniformity of approaches to enable coordinated action and to ensure goals are met, and (2) incorporating autonomy to nurture the creativity and resourcefulness of developers. Understanding how to balance these needs and how differing approaches to control software development affect software development performance and firm performance is therefore an important issue for both managers and academics.

The control of activities by software development firms is not well understood and is characterized by three problems: conflicting advice on the suitability of different approaches, divergence between prescriptions and practice, and lack of research on the issues facing software firms. First, the literature provides conflicting advice on the desirable orientation to control. On one hand, popular approaches, such as the Software Engineering Institute's Capability Maturity Model [28, 29] and the Software Factory [14], prescribe centralized control and standardization of software development across multiple projects. On the other hand, proponents of team-based approaches, such as self-managing teams [4, 40], suggest a high level of delegation of choices to groups improves outcomes. Second, a broad body of evidence suggests that there is considerable *slippage* between the canonical prescriptions of formal organizational controls and those actually enacted during software development; prior theories do not adequately capture the nuances observed in the specification and deployment of controls. For instance, Orlikowski provides this description of the actions of developers in a large consulting firm in using an elaborate development

methodology: "While younger consultants and recently promoted managers tended to interpret the methodology literally, and follow its procedures 'to the letter,' other more experienced employees seemed less constrained, relying on their initiative to direct production work, and using the methodology primarily as a coordination device to manage impressions through appropriate behavior packaging. Thus, while the methodology is prescriptive in documentation, in practice, its tenets were often modified, overridden, or ignored" [48, p. 16]. An accurate conceptualization of how control is exercised and its effects on performance is thus important for academics as well as for managers in software firms.

Third, information systems (IS) researchers have predominantly focused on the control of in-house software development to produce software for internal users. There have been few investigations of the unique challenges facing firms developing software for sale or under contract for firms. Given the importance of such activities in the economy, this context warrants greater attention. Highlighting the link between the choice of control mechanisms and the competitive performance of software firms would thus be a contribution to both research and practice.

In this paper, we propose a synthesis of two key approaches to control, *the process approach* and *the structure approach,* which differ in their emphasis on the mechanisms that guide activities. The process approach emphasizes the specification of behaviors and outcomes as the key means to guide work. In contrast, the structure approach emphasizes organizational structuring—the prespecification of standards and choices regarding the level of delegation of decision-making as the key means to guide work.

Process-oriented approaches to control highlight two broad means by which control is exercised: *behavior control*—the specification of desired behaviors, and *outcome control*—the specification of desired outcomes. However, the process approach by itself provides only a partial picture of the nature of control. For instance, it does not clarify important nuances of how behavior control and outcome control guide action. The extent to which these control strategies impose uniformity in behaviors and the extent to which they take advantage of individual initiative, imagination, resourcefulness, and ability to adjust actions to local conditions are important considerations that are either overlooked or not acknowledged. This is a major issue, as there is evidence that similar behavior controls (or outcome controls) have widely varying results in different contexts. In some contexts, the controls enable adaptation of actions to suit the specific context [57]. However, in other settings, the same set of controls form a Weberian iron cage constraining controllees to mindless rule following or counterproductive actions directed at predefined goals, even in the presence of evidence that a different set of actions is appropriate [38]. The structure approach to control highlights the level of detail to which standards are prespecified *(standardization)* and the extent to which developers have discretion in applying and interpreting standards in the context of task execution *(decentralization).* The process approach and the structure approach are clearly complementary, and combining them yields a more detailed picture of control strategies and how they guide action.

Orlikowski [48] coined the term *matrix of control* to refer to the complex control environment created through the combination of the structuring of work through information technologies and the responses of the individuals to the technologies. We draw on this notion to represent the joint effects of the structure and process approaches to control. In the context of software development, we suggest that the matrix of control comprises four modes of control reflecting the extent of predetermination of standard methods and performance criteria and the level of discretion for selecting the methods and performance criteria that are appropriate for a particular context. We propose and test hypotheses regarding the effect of different modes of control on software development performance and firm performance.

Theoretical Background

Control

A FUNDAMENTAL THEME IN THE CONTROL LITERATURE is the identification of features influencing the appropriate means to guide and direct the activities of controllees. In the context of software development, a firm's management is generally the controller that needs to ensure appropriate action by project teams, the controllees. Mechanisms used by controllers to ensure that individuals and groups act in a manner consistent with meeting organizational goals and objectives are termed the *modes of control* [37]. Control in organizations has been viewed from two perspectives: control through process and control through structure [49, 61].

Control Through Process

Control through process concerns the regulation of performance through the establishment of management processes to guide activities [21, 56]. This view of control can be traced to the seminal work of Ouchi and Maguire [49, 50, 51], who suggested that an organization's control consists of the "process of monitoring, evaluating, and rewarding" employees [49, p. 99]. Researchers have focused on two dimensions of formal control through processes: control through the specification of behaviors and through the specification of outcomes. These dimensions are often termed *behavior control* and *outcome control*[2] [37]. The appropriate level of each type of control is determined by characteristics, such as task programmability and outcome measurability [20], and contextual factors, such as the information available to effect control [53]. Each of these dimensions involves the establishment of a variety of management processes to guide activities related to achieving control [37].

Behavior Control. Behavior control processes regulate activities by *clarifying details of specific behaviors involved in task execution.* The ability to identify appropriate behaviors (and inappropriate behaviors) for different tasks is linked to the level of task programmability and the level of information available on the cause–effect relationships of different behavior choices. Detailed behavioral sequences can be identi-

fied for tasks high in programmability, but not for tasks low in programmability [20]. Behavior control in the context of software development is exercised through the specification of methods for tasks such as project management, analysis, design, programming, testing, and documentation [36]. Establishing behavior control requires creating management processes to specify and monitor behaviors and others to provide incentives and reward appropriate behaviors.

Outcome Control. Outcome control processes regulate activities by *clarifying details of the outcomes of task execution.* The ability to identify appropriate (and inappropriate) outcomes for different tasks is linked to outcome measurability [20]. Outcomes expected upon successful task execution can be clearly identified and described only for tasks high in outcome measurability. Activities for which assessing outcomes and products involves subjective judgment or tacit understanding of the context are less amenable to outcome control than those whose outcomes can be determined objectively. Outcome control in software development activities is exercised through the specification of *benchmarks, quality metrics,* and *performance criteria,* such as productivity standards, budgets, and schedules. Establishing outcome controls involves creating management processes to determine what outcome criteria are applicable, to compare performance against benchmarks, and to provide incentives linked to outcomes.

The strength of the process perspective is that it highlights the management processes that regulate action by providing clarity with respect to behaviors and with respect to outcomes.

Control Through Structure

Control through structure is traditionally linked to reliance on (1) structuring actions through predefined rules and procedures and (2) authority deriving from a legally defined office or position [58, 61]. The extent of control through structure is reflected in the extent to which uniform standards are specified in advance for activities (i.e., the level of standardization), and to the extent of employee discretion to interpret and apply standards, rules, and policies (i.e., the degree of decentralization of decision-making) [25, 33]. Brown [7] comprehensively reviews this extensive and varied research stream. In this paper, we focus on management choices related to the levels of *standardization* and *decentralization* in exerting control. Standardization and centralization both relate to governance structure: the constraining and guiding of action through rules and regulations and pattern of allocation of decision rights between managers and project teams.

Standardization. The level of standardization reflects the extent to which management mandates a uniform set of rules and procedures to guide controllees [35]. These standards can pertain to behavior or to performance criteria used to evaluate outcomes. The greater the standardization, the more detailed the uniform set of prespecified methods (or performance criteria). Standardization enables the codification of knowledge gained through experience in the form of standard methods for

software development tasks such as requirements determination and project management. In contexts characterized by high levels of behavior control, greater standardization facilitates uniform task execution. Alternatively, knowledge can be embedded in outcome metrics and in benchmarks for evaluating software development performance. Standardization in contexts of outcome control is reflected in the greater use of uniform metrics to assess outcomes, such as the number of defects per line of code delivered by the project and the number of man-hours used to perform requirements specifications. Standardization of performance criteria enables the outcomes of multiple projects and project outcomes from different periods to be directly compared. Examples of software development approaches emphasizing control through standardization include the capability maturity model (CMM) [28] and practices in the software factory described by Cusumano [14].

Discussions of standardization often gloss over the trade-off involved between the benefits of creating uniform, broadly applicable standards and the detrimental effects of the mismatch of global standards to the local context of action. The work of Orlikowski [48] is an exception that takes a critical view of the deployment of universal standards. She quotes a manager in a software development firm that had instituted a very standardized development approach as joking that "they send you halfway around the world and because of the common way of doing things and the common training and knowledge, your only requirement is to be shown the coffee machine and the toilet, and you're productive" [48, p. 16]. Clearly, this approach overlooks the presence of contextual differences and affects outcomes by assuming a *one size fits all* approach.

Decentralization. Applying standards for software development methods or for performance criteria presumes a level of stability that is rarely found in software development [8]. Effective task performance involves significant improvisation and the use of discretion to devise variations to prescribed methods (or prescribed outcome standards) [48, 64]. Decentralization reflects the extent to which project teams have the discretion to modify their actions to suit the specific context of task execution. In contrast, where control is centralized, controllees have little discretion allowed to them. In such a context, a software development team is expected to adhere rigidly to prespecified standards, even when superior alternatives are available. Alternatively, the team may be able to refer discrepancies between standards and realistic practice to the hierarchy for resolution [44]. In contexts where control is decentralized, project teams can modify standards on the basis of their judgment and their information on the local context [19, 55]. Decentralization therefore reflects a choice to delegate authority and rely on autonomous decision-making by individuals and teams.

Mills and Baker's notion of the "chief programmer team" [39], in which one person (the chief programmer) makes all decisions, and Weinberg's concept of the "egoless programming team" [63], where decision-making is distributed among team members, are instances of control through structure—through centralization and through decentralization, respectively. The issue of centralization versus decentralization is likely to be more important in a software development context to the extent the developers view themselves as professionals; it is increasingly recognized that professional conduct in a variety of disciplines involves more than merely "following the rules" [16].

The Matrix of Control: Synthesizing Process and Structure Control Approaches

The central thesis of this paper is that the two approaches we have discussed—control through process and control through structure—are complementary and that the overall control exercised in a firm is the joint effect of these approaches. Orlikowski [48] coined the term *matrix of control* to refer to the complex control environment created through the interplay of the structuring of work through information technologies and the responses of the individuals to the technologies. We draw on this notion to represent the joint effects of the two approaches to control. A matrix of control in which structural and process approaches to control both operate is a more accurate depiction of the nature of formal organizational control than either the process approach or the structure approach alone. We view such a matrix of control as a "portfolio" [37] of four modes of control, each derived from the combined effect of one of the two dimensions of control through process (methods and performance criteria) and one of the dimensions of control through structure (standardization and decentralization). In the context of software development, the four modes are *standardization of methods, standardization of performance criteria, decentralization of methods,* and *decentralization of performance criteria* (see Table 1).

Control strategies can thus be represented as combinations of each of the four modes of control comprising the matrix of control, and the combination of modes chosen influences performance. This view of control is consistent with broader findings that firm outcomes are influenced by structure-setting actions combined with process-focused efforts. For instance, in a study of software quality practices, Ravichandran and Rai [52] found successful initiatives to be characterized by top management leadership providing an overarching framework, in conjunction with policies encouraging process-focused improvement efforts by empowered individuals.

Joint Effect of Methods Control and Dimensions of the Structural Approach

Control through the specification of behavior in software projects occurs in the form of "how to's," or methods, for undertaking software development activities. These are available in the form of codified standards, procedures, and practices for tasks such as project management, requirements elicitation, system design, programming, and documentation [27, 36]. Two distinct modes of control arise from the combined effect of methods control and firm choices with respect to each of the dimensions of control through structure (standardization and decentralization):

1. *Standardization of methods.* The a priori specification of process standards for software development tasks, reflecting the degree of complexity of efforts to develop uniform procedures for tasks.
2. *Decentralization of methods.* The discretion provided to project teams to determine the methods they use for software development tasks, reflecting the extent of delegation of decision-making to teams with respect to methods to be followed and the latitude permitted to enable adaptation to the local context.

Table 1. Modes Comprising the Matrix of Control

Approaches to Control		
Through process	Through structure	Modes of control
Controlling activities through the specification of methods for activities involved in software development.	Standardization	*Standardization of methods:* Standard methods defined for activities (high–low depending on degree of comprehensiveness and detail of the specification of methods).
	Decentralization	*Decentralization of methods:* Discretion to determine methods used rests with project teams (high–low depending on the extent of discretion to choose methods available to project teams).
Controlling activities through the specification of performance criteria for outputs and outcomes of software development.	Standardization	*Standardization of performance criteria:* Uniform definition of standard performance criteria applied to project teams (high–low depending on degree of comprehensiveness and detail of the specification of performance criteria).
	Decentralization	*Decentralization of performance criteria:* Discretion to determine performance criteria rests with project teams (high–low depending on the extent of discretion in determining performance criteria available to project teams).

Joint Effect of Outcome Control and Dimensions of the Structural Approach

Outcome control in software development takes the form of specifying performance criteria pertaining to products or processes [36, 37, 45]. Product performance specifications define standards for the functionality or quality of software, and process performance specifications define standards for operational aspects of a project (such as meeting schedules and budgets).

Two distinct modes of control arise from the joint effect of outcome controls (performance criteria specifications, benchmarks, etc.) and firm choices with respect to each of the two dimensions of control through structure:

1. *Standardization of performance criteria:* The extent to which performance standards are defined a priori for tasks involved in software development.
2. *Decentralization of performance criteria:* The extent of discretion provided to individual project teams to determine the performance criteria used to judge outcomes in projects.

In software development contexts, it is intuitive to conceive of high standardization accompanying centralized decision-making with respect to performance criteria and methods. Cusumano's software factories exemplify such an approach; they build new systems using proven modules drawn from central repositories and make all key decisions centrally. Similarly, it is intuitive to conceive of low standardization as coexisting with high levels of local decision-making and discretion regarding both processes and outcomes in situations characterized by considerable fluidity of goals, emergent action, and local improvisation [64]. Viewing control as a matrix comprising a combination of these four modes highlights that the process and structure approaches can coexist; for instance, that control mechanisms can be highly standardized while simultaneously allowing decentralized decision-making with respect to methods and performance criteria. This idea is consistent with recent evidence that firms adopt different stances with respect to these two aspects of control [1]. This view contrasts with the traditional view that organizations exhibit an inverse correlation between these dimensions—that is, high standardization coupled with decentralization and low standardization linked to high decentralization [61].

This coexistence is supported by examples from contexts in which standards are centrally determined for a wide range of activities (standardization is high), but individual work groups have considerable latitude regarding the procedures for task execution (decentralization is high) [4, 17]. Upton [60] indicates that performance is enhanced when manufacturing firms establish standards for tasks (high standardization) and also allow computer numerically controlled (CNC) machine operators discretion in determining tool changeover routines (high decentralization). A manual change can be faster than a computer-controlled change when an operator takes advantage of local conditions (for instance, advantageous tool location, information about the next job). In the same situation, the computer-based routine would apply general procedures and fail to exploit potential efficiencies from specific circumstances. Similarly, contractors working on U.S. Department of Defense (DOD) contracts receive

standards manuals for software development (such as MIL-STD 498) that define outcome metrics for software in great detail (high standardization of performance criteria). However, the contracts allow project teams to have discretion in adopting the standards to suit the context of specific projects (high decentralization of performance criteria).

Instances of low standardization coexisting with low decentralization are also common in software development. For example, in many small, but rapidly growing, software firms, few standards for evaluating individual software project performance are specified in advance (low standardization of performance criteria). Senior managers typically determine the criteria used to evaluate these projects on a case by case basis (low decentralization of performance criteria).

A software firm with two development centers, one in San Jose, California, and the other in Bangalore, India, offers an instance of high standardization of methods coexisting with decentralization of methods. The firm had established a very detailed set of methods for the management of code under development. Individuals were expected to check code into a central repository after modules were tested; individuals needing to work on modules were expected to follow checkout procedures to obtain the latest versions of code and to check them in with a new version number after modification (high standardization of methods). However, because all the programmers in Bangalore shared a common e-mail address (whereas each developer in the San Jose facility had an individual e-mail address), the actual code management procedures followed by the members of the two groups were quite different (high decentralization of methods). Project members sharing the same address in Bangalore evolved procedures, local to the Bangalore center, to manage the interdependence between modules checked out to the same e-mail address. This allowed the Indian group to conform to the method standards of the firm while taking advantage of the billing structure of the local telecom authority that made it cost-effective to minimize the number of e-mail addresses assigned. This example fits Adler's [1] description of an enabling bureaucracy characterized by a standard set of work procedures to cover expected contingencies (standardization of methods is high), and where flexibility is allowed to individuals and groups in interpreting and implementing them (decentralization of methods is high).

Overall, our conceptualization represents organizational control as a matrix comprising four distinct modes of control: *standardization of methods, decentralization of methods, standardization of performance criteria,* and *decentralization of performance criteria.*

Research Model and Hypotheses

Performance

WE CONCEIVE OF THE BENEFITS TO FIRMS of control as accruing in two stages—the choice of control strategies influencing software development process performance (first-order benefits), and in turn creating competitive benefits (second-order benefits).

Software Development Process Performance. We conceptualize process performance in terms of two dimensions: *process flexibility* and *process predictability.* Prior research on outcomes of control suggests that process flexibility is a primary benefit of the use of appropriate control strategies [32, 45]. A predictable stable process that can be controlled is also an important attribute of software development performance [28]. Firms whose costs, schedules, and performance are unpredictable are at the lowest levels of the SEI's capability maturity model [30].

Competitive Performance. The ability of a firm to control its software development process is expected to lead to favorable competitive outcomes in the marketplace. We view competitive performance in terms of two dimensions, *product cost efficiency* and *market responsiveness,* which correspond to the efficiency and effectiveness dimensions of performance [27, 46].

Hypotheses

We use the "control through process" viewpoint to identify propositions linking methods control and performance criteria control to process performance. We then employ the "control through structure" view to generate hypotheses to relate standardization and decentralization of methods and standardization and decentralization of performance criteria to performance. We also hypothesize a positive relationship between software development process performance and competitive performance. An important assumption we make, drawing on the literature and accounts in the popular press [9, 15], is that the software industry is characterized by high levels of environmental uncertainty. This assumption underlies many of the arguments employed in generating the hypotheses, and we subsequently verify it empirically.

Method Controls and Process Performance

To assess the performance impacts of control, we distinguish between two levels at which software development activities are undertaken: (1) a higher, or macro, level depicting the general approaches to develop software; and (2) a lower, or micro, level detailing the specific methods used for developing software in a particular context or situation. In a particular development context, adequate information may be available on *macro* aspects such as life cycles and development phases. Consequently, these aspects can be generally predetermined in a variety of environments in which a firm operates. In this paper, we focus exclusively on *micro* level behaviors, which pose the greatest challenge in software development.

The choice of appropriate methods for projects is linked to the availability of information about the environment, such as consumer preferences, behaviors of hardware and software technologies, and actions of competitors. When environmental uncertainty is high, uncertainty with regard to requirements, technologies, and other critical aspects of software development is also high. In turn, this leads to incomplete information about methods that should be used to develop software, reflecting a low programmability of desirable behaviors. This follows the logic of Govindarajan and

Fisher [23], who indicate that if the task environment of an organizational unit is uncertain, the task programmability for that unit is also low. Consequently, organizational attempts to preprogram behaviors are likely to be counterproductive [20, 51]. Hence, the control-through-process approach suggests that behavior controls are unlikely to be effective under uncertain conditions. We therefore advance the following general proposition based on the control-through-process approach:

> *Proposition 1. In a software firm, increased method controls lead to* decreased *software development process performance.*

We now explore a more nuanced view of control by recognizing the joint influence of the methods dimension of the *control-through-process* approach and the standardization and decentralization dimensions of the *control-through-structure* approach.

Standardization of Methods. When task uncertainty increases, it becomes more difficult to prespecify the standard set of rules and procedures to be undertaken to perform them [55]. Standardizing rules and procedures when information about requirements, technologies, competition, and other aspects of the environment is incomplete can lead to situations in which the defined standards are inappropriate or ineffective for a large variety of tasks. A long tradition of control-through-structure research thus suggests that imposing standardized rules and procedures in contexts of high uncertainty reduces work unit performance [3, 22]. In software development, the rules and procedures for undertaking work are embedded and enforced by software development methods. Consequently, the control-through-structure approach suggests that efforts to apply a standard set of methods across software projects are likely to be counterproductive in the uncertain environments that software firms confront. Hence,

> *Hypothesis 1. In a software firm, increased standardization of methods controls leads to* decreased *software development process performance.*

Decentralization of Methods. Decentralization of decision-making leads to enhanced outcomes when uncertainty is high [19, 22, 44]. As uncertainty increases, the volume of information that project members need to process in order to handle exceptions also increases [22]. If all exceptional cases are referred up the hierarchy, lengthy delays are likely to occur because of the large volumes of information that need to be processed by the centralized decision-maker. There can be excessive summarization or distortion of information, which can result in inappropriate decisions being made by decision-makers not in direct contact with task execution [55]. Consequently, decentralization of decision-making enables more effective information processing [19, 44, 55]. IS research also suggests that software development teams are more effective if team members have greater control over how they undertake their work, especially if their task requires high levels of technical expertise [27, 62]. Anecdotal evidence suggests that one reason for Microsoft's effectiveness in developing software is the significant discretion that functional experts such as program managers, developers, and testers have over development methods in projects [15]. These arguments suggest:

Hypothesis 2. In a software firm, increased decentralization of methods to individual projects leads to increased *software development process performance.*

Outcome Controls and Process Performance

In the context of software development, outcomes can be more easily specified than methods [37]. Outcome controls are preferred when knowledge of the transformation process is low [20, 23, 56]. Outcome controls are also simpler to implement than behavior controls [36]. Consequently, firms increasingly rely on outcome controls when environmental uncertainty is high and information on requirements, technologies, competitors, and other aspects of the environment is incomplete. For outcome controls to be effective, criteria for judging performance need to be specifiable [56], and outcomes need to be measurable against these standards [23, 37]. Considerable progress has been made in this area; prior research has identified a variety of outcome metrics for software and provided measures to assess outcomes [18, 30]. We therefore suggest the following proposition for software development drawing on the control-through-process perspective:

Proposition 2. In a software firm, increased outcome controls lead to increased *software development process performance.*

We now explore the finer-grained view of outcome control made possible by recognizing the joint influence of the outcome control dimension of the control-through-process approach with each of the standardization and decentralization dimensions of *control through structure.*

Standardization of Outcome Controls. The control-through-structure approach [3, 22] suggests that for high levels of environmental uncertainty, standardization of outcomes is negatively associated with process performance. However, IS research suggests that this may not be the case for software development. For example, Japan's high-performing software factories achieved significant predictability in software development by developing a standard set of performance criteria to monitor and control their software projects [14]. In general, a standard set of criteria for measuring performance enables comparability of progress across work groups performing different tasks [49]. Defining standardized performance criteria for software development thus leads to consistency in measuring performance across projects. In turn, these measurements help establish baselines against which subsequent improvements can be assessed, enabling superior process performance. We therefore suggest the following hypothesis:

Hypotheses 3. In a software firm, increased standardization of outcome controls lead to increased *software development process performance.*

Decentralization of Outcome Controls. Since the control-through-structure approach does not discriminate between outcomes and behaviors, reliance on this approach alone would lead to the conclusion that decentralized control of outcomes under high uncertainty would be positively associated with performance. However, IS research

on software development again suggests the need to reconsider this view. At the project level, centralized control over software development outcomes helps focus a group's efforts on meeting the project's schedules and costs [45]. Software design teams in which decentralization of control is high often fail to complete their tasks in a timely manner [27, 39]. Henderson and Lee [27] found that strong managerial control of outcomes in software projects was important and, in all cases, resulted in more efficient project execution. These arguments suggest that:

> *Hypothesis 4. In a software firm, increased decentralization of outcome controls leads to* decreased *software development process performance.*

Process Performance and Competitive Performance

We suggest a positive relationship between a software firm's development process performance and its competitive performance as software development is central to value creation by such firms. This association is also implicit in process improvement approaches that are shown to enhance performance; examples include the CMM's emphasis on software process maturity [28] and the focus on predictable, flexible processes in the software factories Cusumano studied [14]. We consequently hypothesize that:

> *Hypothesis 5. Software development process performance is positively related to competitive performance.*

Research Design

Survey Design

THE RESEARCH HYPOTHESES WERE TESTED using data collected via a two-stage survey of software firms in a random sample of the membership of American Software Division of the Information Technology Association of America (ITAA). This sampling frame was selected on account of the diversity of its members in size and in types of software products and services. In the first stage of the survey administration, we obtained data on control mechanisms and firm performance from a senior manager in charge of software development (typically titled the VP for development or the chief technology officer). We expected managers in these roles to be best informed to respond to the questions on the survey. From these respondents, we also obtained contact details of the head of the marketing function in their firms (typically the VP of marketing). We surveyed these individuals in the second stage of the survey and obtained reports on firm performance. We expected that the independent responses of managers engaged in different roles would reveal systematic role-related biases potentially confounding our results.

As resources for research were constrained, we believed it judicious to concentrate our efforts on a small sample of firms and attempt to achieve a high rate of response to both stages of our survey. Accordingly, we selected 100 software firms from the

350 members of the ITAA. We expected the random selection of firms would control for firm-level variables that might confound the results.

Measures

We drew items from the literature wherever possible and adapted them to the context of software development. For novel constructs not studied in prior IS research, we created items to capture the specific details of the context. We created the final version mailed to firms through iteratively refining the instrument in test administrations to IS academics and managers. Appendix A provides details of the items and their sources.

Response Rate and Nonresponse Bias

We received 58 responses in the first stage of survey administration, a response rate of 58 percent. We received 36 responses to the second stage, a response rate of 62 percent (36 of 58 firms). Two of the 58 firms were eliminated from our analysis because they were start-ups and reported zero sales.

Validation of Measures

Construct Validity and Reliability. We assessed the reliability of measurement using Cronbach's alpha. We employed factor analyses to assess construct and discriminant validity. Validation procedures for the scales used for key constructs in the model suggested satisfactory psychometric properties. Appendix B provides details of the psychometric analyses. We created composite scores for each multidimensional construct through principal component analysis.

External Validity and the Effect of Firm Size. External validity refers to the extent to which findings can be generalized over time, persons, and settings [13, 59]. External validity is enhanced if a sample itself is not systematically biased with regard to key characteristics such as organization size. With firm size operationalized as number of employees, the median size of firms in the sample is 43 (Table 2). The large number of relatively small firms in the sample reflects the preponderance of small firms in the software industry as a whole.

To examine the influence of size, we compared the groups created by a median split of the data on firm size. There were no significant differences between the groups on 10 of the 11 dimensions in the research model (Table 3). The groups differed significantly on only one dimension—the level of standardization of front-end methods. Smaller firms in the sample were less likely to define organization-wide standards with respect to front-end methods for software development (e.g., project management, analysis, and design) than larger firms.

Adequacy of Sample Size. EQS was used in analyses to test the hypotheses. For reliable estimates in EQS, the sample size needs to be five times greater than the

Table 2. Sample Characteristics ($N = 56$)

	Mean	Standard deviation	Median
Number of years in business	21	18.7	15
Number of employees (firm size)	645	1,932	43
Annual sales volume ($ million)	2.7	1.7	4
Number of software projects undertaken per year	39	77	10
Percentage of software development outsourced (%)	7.7	14.8	0
Number of new products introduced per year	9	24.7	2
Number of years software development process in use	7	6.1	6

number of parameters estimated [6]. Power analyses suggest a sample size greater than 51 to detect significant effects [11].[3] The sample size of 56 satisfies both these requirements.

Analysis

Test of Research Models. We used structural equations modeling (SEM) techniques as implemented in EQS [6] to test the model and research hypotheses. EQS is used in a variety of contexts by researchers for both path-analytic and latent variable modeling [26, 45].

In EQS, the p-value for a model's chi-square statistic reflects the level of fit between the data and the research model. Values greater than 0.05 indicate an acceptable fit between the research model and data [6]. Model fit is also assessed from the values of the comparative fit index (CFI) and the normed fit index (NFI). The value of the average absolute standardized residuals (AASR) indicates the proportion of the variance not explained by the model. In EQS, model re-specification is undertaken based on information provided by the Wald and Lagrange multiplier (LM) tests. The former identifies effects in the model that can be dropped to improve fit, whereas the latter suggests effects that may be added to significantly improve fit. The significance of path coefficients in the model provides support for hypothesized relationships [6].

Results

Sample Characteristics

THE SAMPLE CHARACTERISTICS ARE PRESENTED in Table 2. The median firm in the sample employed 43 employees, had an annual sales volume of $4 million, undertook about 10 software projects in a year, largely developed software internally, introduced

Table 3. Values of Constructs for Small and Large Firms

Construct	Dimension	Mean of large firms	Mean of small firms	*t*-value	*p*-value
Standardization of controls	Performance criteria	3.21	2.96	1.03	0.30
	Front-end methods	3.28	2.68	*2.33*	0.02*
	Back-end methods	3.69	3.38	1.37	0.18
Decentralization of controls	Performance criteria	3.10	3.16	–0.30	0.77
	Methods	3.11	3.41	–1.35	0.18
Process predictability		2.85	3.18	–1.36	0.18
Process flexibility	Labor	3.40	3.03	1.87	0.07
	Market	3.35	3.34	0.12	0.91
	Regulatory	3.43	3.18	1.27	0.21
Product cost efficiency		3.51	3.75	–1.32	0.19
Market responsiveness		3.04	2.95	0.43	0.67

Note: * $p < 0.05$.

Table 4. Pearson Correlation Coefficients ($N = 56$)

Number	Construct	1	2	3	4	5
1	Standardization of performance criteria	1.00				
2	Decentralization of performance criteria	–0.26	1.00			
3	Standardization of methods	0.61**	–0.13	1.00		
4	Decentralization of methods	–0.19	0.43**	–0.54**	1.00	
5	Development process performance	0.48**	–0.11	0.21	0.21	1.00
6	Competitive performance	0.30*	–0.27	0.02	0.06	0.56**

Notes: * $p < 0.05$, ** $p < 0.01$.

about two new software products each year, and had been in business for about 15 years. On the average, the firms in the sample had used their current approaches to software development for six years, an observation suggesting that the information provided is likely to pertain to long-standing control strategies. The inter-construct correlations are provided in Table 4.

Assumption Testing and Correlations

We empirically confirmed the expectation of high levels of environmental uncertainty faced by firms in the software industry. The mean uncertainty was 4.20 (s.d. = 0.95), higher than the level reported in prior uses of the scale by Miller and Droge [41] (mean = 3.44, s.d. = 1.16), and by Miller et al. [42] (mean = 3.53, s.d = 1.14). The uncertainty scores in our sample are also higher than those reported by Sabherwal and King [54] (mean = 3.51, s.d. = 0.93) for large manufacturing and service companies in the United States.

Testing of the Research Model

Model Fit. The EQS results are presented in Figure 1. The chi-square statistic is significant ($p = 0.001$); the magnitude of fit indices (NFI = 0.34, CFI = 0.34) were lower than 0.90; and the standardized residuals (AASR = 0.19) were higher than the recommended value of 0.05. The Wald test suggested that model fit could be significantly improved by dropping the direct effects of decentralization of performance criteria and standardization of methods on process performance. The revised model incorporating these changes (see Figure 2) indicated an adequate fit of the model to the data: it had a nonsignificant chi-square statistic ($p = 0.55$), and fit indices above 0.9 (NFI = 0.95, CFI = 1.00). The value of the standardized residual, reflecting the proportion of unexplained variance, was acceptable (AASR = 0.05). This pattern of findings suggests the revised model fits the data well, and we therefore examine the research hypotheses using this revised model. The results are presented in Table 5.

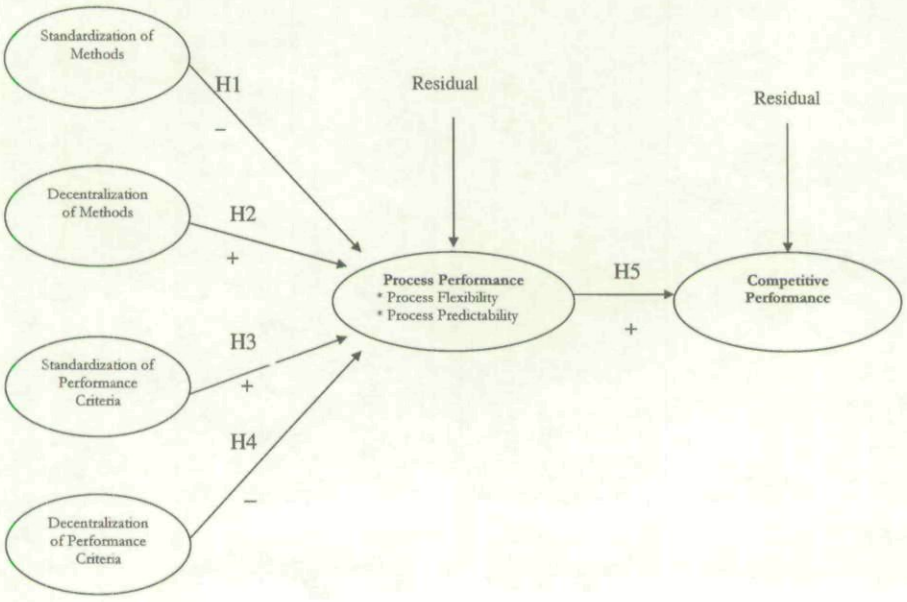

Figure 1. Modes of Control and Performance

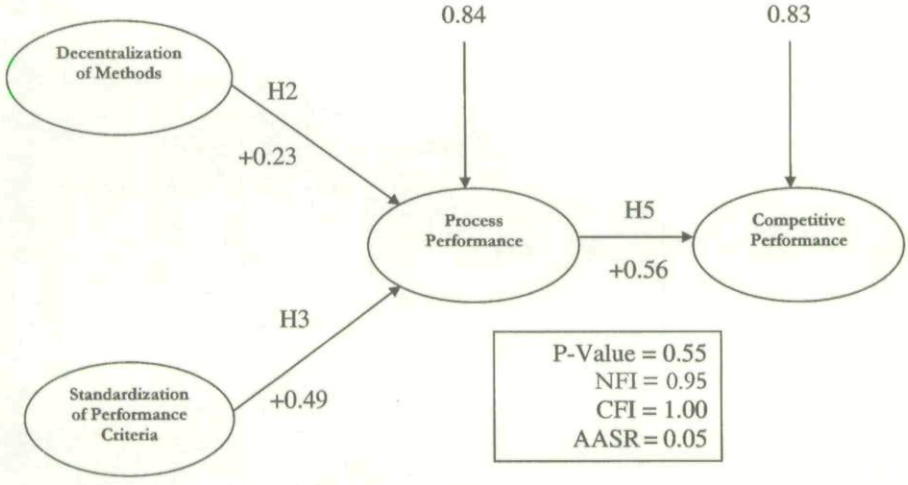

Figure 2. Modes of Control and Performance (Revised Model)

Hypotheses Testing. Since dropping the effect of standardization of methods on process performance enhanced model fit, Hypothesis 1 is not supported. The effect of decentralization of methods on process performance was positive and significant, as predicted, suggesting support for Hypothesis 2. The effect size (given by the absolute value of the standardized path coefficient) for this effect was 0.23, suggesting that decentralization of methods explains over 5 percent of the variation in process

Table 5. Results of EQS Analysis

Dependent variable	Corresponding independent variable	Hypothesis	Raw path coefficient	Standard error	Z-score	Standardized path coefficient
Process performance	Standardization of methods	H1		Dropped from the model		
	Decentralization of methods	H2	0.33	0.16	2.06*	0.23
	Standardization of performance criteria	H3	0.66	0.15	4.32**	0.49
	Decentralization of performance criteria	H4		Dropped from the model		
Competitive performance	Process Performance	H5	0.60	0.12	5.01**	0.55

Notes: * $p < 0.05$, ** $p < 0.01$.

performance.[4] The effect of standardization of performance criteria on process performance was positive and significant, suggesting support for Hypothesis 3. The effect size was 0.49, suggesting that standardization of performance criteria explains over 24 percent of the variation in process performance. Since the model fit improved when we dropped the effect of decentralization of performance criteria on process performance, Hypothesis 4 is not supported. As hypothesized, the effect of process performance on competitive performance was positive and significant, suggesting support for Hypothesis 5. The effect size was 0.56, indicating that process performance explained over 31 percent of the variation in competitive performance. Overall, the average of the three effect sizes (0.42) is higher than the average effect size of 0.35 suggested for research in management information systems (MIS) [5].

Validation of Findings

Cross-Validation with Data from Second Respondent. To cross-validate the findings obtained from the single-respondent sample collected in the first phase of survey administration (where the data on control strategies as well as performance were provided by the firms' development managers), we tested the model using data from the set of 36 firms for whom we had development managers' responses on control and marketing managers' responses on performance. The revised model (Figure 2) had a nonsignificant chi-square statistic ($p = 0.15$), suggesting an acceptable fit between the data and the model. However, the fit indices had values lower than 0.90 (NFI = 0.76, CFI = 0.85), and the standardized residuals (AASR = 0.10). The LM test suggested that model fit would be significantly increased by introducing a covariance between standardization of performance criteria and decentralization of methods. This modified model (Figure 3) fit the data well ($p = 0.84$, NFI = 0.98, CFI = 1.00) with low residuals (AASR = 0.01). The Wald and LM tests suggested no further changes to improve the model's fit with the double-respondent sample. The results are similar to those obtained with both control and performance data provided by development managers (see Table 6).

Internal Validity. Internal validity reflects the extent to which an alternative explanation, such as an omitted variable, may account for the effects [43]. Organization size is one such variable. To examine this possibility, we included size as a control variable in the revised model to examine if it modified the effects of the modes of control on process and competitive performance (see Figure 4). Size has a significant and *negative* effect on process performance, indicating that smaller firms in the sample were likely to have higher levels of software development performance than the larger firms. The effect size was 0.26, which indicates that size explains almost 7 percent of the variation in competitive performance. However, the existing relationships between the modes of control and performance were unaffected, and Hypotheses 2, 3, and 5 continued to be supported. These results provide evidence that the influence of the modes of control comprising the control matrix on process and competitive performance are not contingent on firm size.

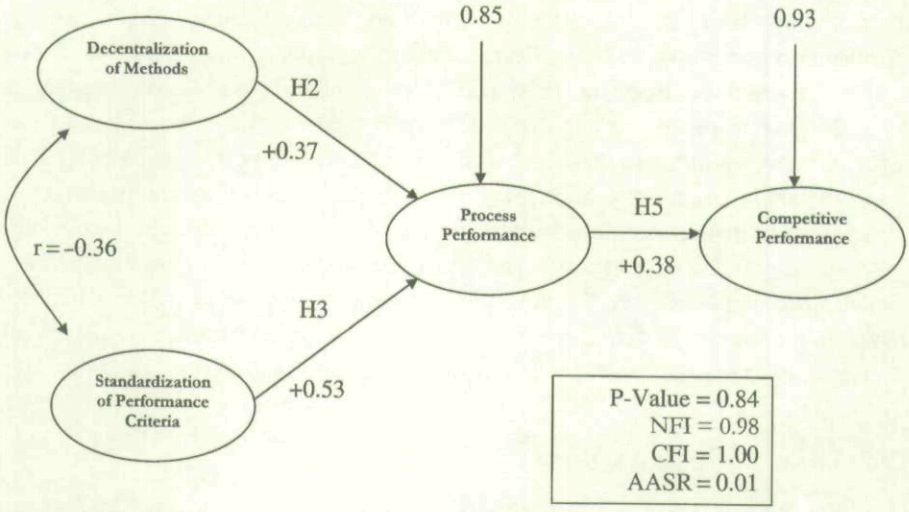

Figure 3. Modes of Control and Performance (Cross-Validated Model)

Discussion and Conclusions

Discussion

THE RESULTS INDICATE THAT: (1) decentralization of methods is positively related to process performance (Hypothesis 2), (2) standardization of performance criteria is positively related to process performance (Hypothesis 3), and (3) software development process performance is positively related to competitive performance (Hypothesis 5). There was no support for the hypotheses that standardization of methods (Hypothesis 1) and decentralization of performance criteria (Hypothesis 4) are inversely related to performance. These results thus present a more complex picture of the performance implications of the four modes comprising the matrix of control than either the control-through-process or control-through-structure approaches used alone would suggest. Three general conclusions illustrate this complexity.

First, an application of the control-through-process approach alone would suggest that, in the highly uncertain environment of the software industry, managers should rely on outcome controls, such as performance criteria, rather than on methods. Our results provide a more nuanced view of the choice between outcome and behavior control. Performance is enhanced by standardization of outcome controls, such as the establishment of detailed performance criteria for projects. On the other hand, performance is enhanced through decentralization of behavior controls—for instance, by allowing discretion to individual project teams regarding methods to be used for tasks such as requirements determination.

Second, focusing exclusively on structural aspects as in the control-through-structure approach is also prone to oversimplification. For example, this approach would suggest that controls should be destandardized and decentralized, given the high environmental uncertainty in the software industry. Our results, however, indicate that

Table 6. EQS Results of Model Using Performance Data from Second Respondent

Dependent variable	Corresponding independent variable	Hypothesis	Raw path coefficient	Standard error	Z-score	Standardized path coefficient
Process performance	Standardization of methods	H1		Dropped from the model		
	Decentralization of methods	H2	0.49	0.21	2.37*	0.37
	Standardization of performance criteria	H3	0.83	0.24	3.39**	0.53
	Decentralization of performance criteria	H4		Dropped from the model		
Competitive performance	Process performance	H5	0.38	0.16	2.43**	0.38

Notes: * $p < 0.05$, ** $p < 0.01$.

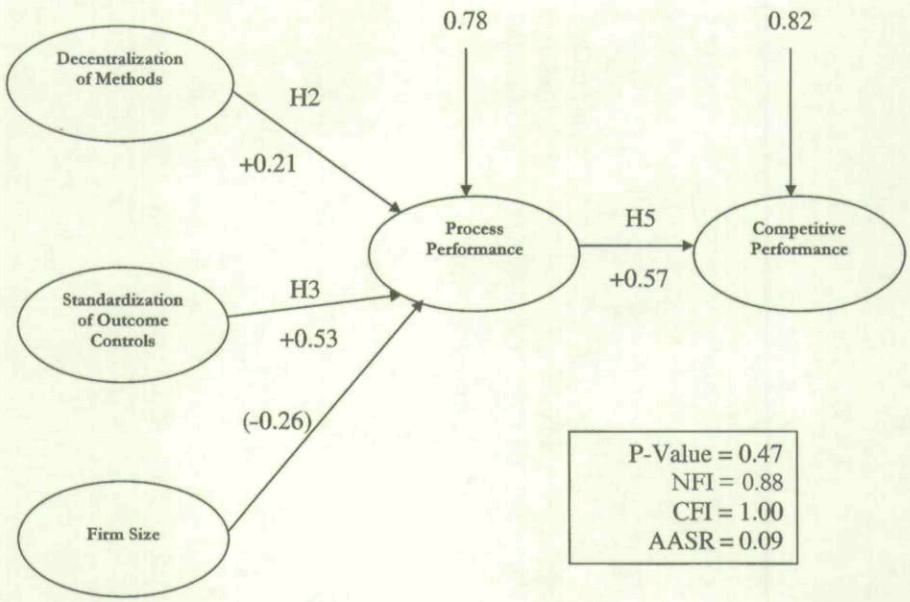

Figure 4. Modes of Control and Performance (Including Firm Size)

the aspect of process being controlled influences the performance implications of standardization and centralization. For instance, contrary to what structural contingency theory and control-through-structure approaches would suggest, standardization of outcome controls, such as performance criteria, are associated with *increased* rather than decreased process performance, but there is no evidence for the positive effects of decentralized outcome controls. Whereas the decentralization of process controls is positively associated with performance, the destandardization of process controls is not.

Third, the findings suggest a lack of symmetry in the relationships between the structural dimensions of control (standardization and decentralization) and performance. For example, although a priori specification of detailed performance criteria (high performance criteria standardization) enhances process performance, there is no evidence that formulating detailed methods standards and guidelines for software development tasks (high method standardization) improves process performance. Similarly, although delegation of decision-making regarding software development methods to individual project teams (methods decentralization) is associated with enhanced performance, there is no evidence that the discretion allowed to teams to devise performance criteria (high performance criteria decentralization) significantly affects process performance adversely.

These results suggest that the firms observed here combine the two structure dimensions (decentralization and standardization) with the process dimensions (behavior and outcome controls) in complex ways in achieving control over software development activities. Software development performance is enhanced when firms

adopt hybrid control strategies, combining high levels of standardization of performance criteria with decentralized decision-making with respect to software development methods.

Limitations

The results are based on a modest sample size of 56 firms. Performance was assessed via subjective reports because the sample included a significant proportion of privately held software firms whose financial performance data were not available. Future studies of larger samples of publicly traded software firms using objective performance data would help validate the results and enhance their generalizability.

Conclusions and Suggestions for Future Research

This study examines control strategies employed by software development firms, an issue that has received little attention. The notion of a control matrix comprising four modes of control helps synthesize two distinct, but related, approaches to organizational control in software development—*control through process,* achieved through regulation of behavior and through specifying outcomes, and *control through structure,* achieved through standardization and decentralization. Such a nuanced understanding of control strategies, as representing choices about deploying four control modes that together constitute a matrix of control, can inform firms' attempts to enhance software development performance and competitive performance.

Examining the influence of variables such as the type of software developed and the role of information incompleteness is a promising alternative approach to integrating the effects of process-based and structure-based control. Further, agency theory [24], in conjunction with the notion of information asymmetry between principal (the firm) and agents (software developers) can explicate how the combinations of process and structure features of control influence outcomes.

In keeping with the accepted view in the control literature [14, 37], we conceptualized the control matrix as comprising four modes of control created by process control choices (behavior control, outcome control) and structural control choices (standardization, decentralization). A refinement of this view could account for the role of individual perception in determining the level of centralization or decentralization, a view suggested by recent work in managerial cognition [10]. This idea is consistent with Orlikowski's observation that experienced managers may interpret control as more decentralized and more amenable to the application of discretion than less experienced employees [48]. This is an avenue for future research that can provide interesting insights on the dynamics of how individual action shapes control strategies. Future research could also examine the performance implications of the interaction of informal controls such as clan control [51] with formal controls. Further, examining the effect of contingencies such as the nature of task interdependence and the level of goal conflict [2] on the choice of modes of control would lead to additional insights.

Our results suggest that managing software development involves a complex choice of formal modes of control comprising the matrix of control. As hybrid organizational forms supplant simpler mechanistic and organic organizational forms in the software industry and in other competitive industries, researchers need to recognize the complex combination of control modes that enable performance. To this end, the modes of control we highlight may be applicable in other uncertain knowledge-work–intensive contexts and future research on the matrix of control in other domains is likely to yield useful insights.

Acknowledgments: The authors thank Anne Hickey and Vijay Khatri for assistance in conducting the research. They are also thankful to many colleagues, particularly to Gordon Davis and Andy Van de Ven, for their help in shaping the paper. The authors also thank three anonymous reviewers for their comments, which significantly improved the quality of the paper.

Notes

1. Organizational control comprises both formal and informal controls; this paper deals exclusively with formal controls.

2. Behavior control refers to *control exercised through the specification of expected behaviors,* and outcome control refers to *control exercised through the specification of expected outcomes.* Both of these dimensions of the process approach enable the regulation and guiding of the *activities* of controllees.

3. For an effect size of 0.35, Type I error of 0.05 and power of 0.80.

4. The variance explained is calculated as the square of the magnitude of the effect size.

References

1. Adler, P.S., and Borys, B. Two types of bureaucracy: Enabling and coercive. *Administrative Science Quarterly, 41,* 1 (1996), 61–89.

2. Andres, H.P., and Zmud, R.W. A contingency approach to software project coordination. *Journal of Management Information Systems, 18,* 3 (Winter 2001–2), 41–70.

3. Argote, L. Input uncertainty and organizational coordination in hospital emergency units. *Administrative Science Quarterly, 27,* 3 (1982), 420–434.

4. Barker, J.R. Tightening the iron cage: Concertive control in self-managing teams. *Administrative Science Quarterly, 38,* 3 (1993), 408–437.

5. Baroudi, J.J., and Orlikowski, W.J. The problem of statistical power in MIS research. *MIS Quarterly, 13,* 1 (March 1989), 87–106.

6. Bentler, P.M. *Structural Equations Program Manual.* Los Angeles: BMDP Statistical Software, 1989.

7. Brown, C.V. Examining the emergence of hybrid governance solutions: Evidence from a single case site. *Information Systems Research, 8,* 1 (March 1997), 69–94.

8. Brown, J.S., and Duguid, P. Organizational learning and communities-of-practice: Toward a unified view of working, learning and innovation. *Organization Science, 2,* 1 (1991), 40–57.

9. Brown, S.L., and Eisenhardt, K.M. Product development: Past research, present findings, and future directions. *Academy of Management Review, 20,* 2 (April 1995), 343–378.

10. Carpenter, M.A., and Golden, B.R. Perceived managerial discretion: A study of cause and effect. *Strategic Management Journal, 18,* 3 (1997), 187–206.

11. Cohen, J. *Statistical Power Analysis for the Behavioral Sciences,* revised ed. New York: Academic Press, 1977.

12. Conger, S. *The New Software Engineering.* Belmont, CA: Wadsworth, 1994.

13. Cook, T.D., and Campbell, D.T. *Quasi-Experimentation.* Boston: Houghton Mifflin, 1979.

14. Cusumano, M.A. *Japan's Software Factories: A Challenge to U.S. Management.* New York: Oxford University Press, 1991.

15. Cusumano, M.A., and Selby, R.W. *Microsoft Secrets: How the World's Most Powerful Company Creates Technology, Shapes Markets and Manages People.* New York: Free Press, 1995.

16. Davis, M. Professional responsibility: Just following the rules? *Business and Professional Ethics Journal, 18,* 1 (1999), 65–88.

17. Dean, J.W., Jr.; Joon, Y.S.; and Susman, G.I. Advanced manufacturing technology and organization structure: Empowerment or subordination. *Organization Science, 3,* 2 (May 1992), 203–229.

18. DeLone, W.H., and McLean, E.R. Information systems success: The quest for the dependent variable. *Information Systems Research, 3,* 1 (1992), 60–95.

19. Drazin, R., and Van de Ven, A.H. Alternative forms of fit in contingency theory. *Administrative Science Quarterly, 30,* 4 (1985), 514–539.

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

21. Flamholtz, E.G.; Das, T.K.; and Tsui, A. Toward an integrative framework of organizational control. *Accounting Organizations and Society, 10,* 1 (1985), 35–50.

22. Fry, L.W., and Slocum, J.W., Jr. Technology, structure and work group effectiveness: A test of a contingency model. *Academy of Management Journal, 27,* 2 (1984), 221–246.

23. Govindarajan, V., and Fisher, J. Strategy, control systems, and resource sharing: Effects on business-unit performance. *Academy of Management Journal, 33,* 2 (1990), 259–285.

24. Gurbuxani, V., and Kemerer, C.F. An agent theoretic perspective on the management of information systems. In R.H. Sprague, Jr. (ed.), *Proceedings of the Twenty-Second Annual Hawaii International Conference on System Science.* Los Alamitos, CA: IEEE Computer Society Press, 1989, pp. 141–150.

25. Hambrick, D.C., and Finkelstein, S. Managerial discretion: A bridge between polar views of organizational outcomes. In B.M. Staw and L.L. Cummings (eds.), *Research in Organizational Behavior Interactional Psychology.* Greenwich, CT: JAI Press, 1987, pp. 369–406.

26. Hartwick, J., and Barki, H. Explaining the role of user participation in information systems use. *Management Science, 40,* 4 (1994), 440–465.

27. Henderson, J.C., and Lee, S. Managing I/S design teams: A control theories perspective. *Management Science, 38,* 6 (June 1992), 757–777.

28. Herbsleb, J.; Zubrow, D.; Goldenson, D.; Hayes, W.; and Paulk, M. Software quality and the capability maturity model. *Communications of the ACM, 40,* 6 (1997), 30–40.

29. Humphrey, W.S. *Managing the Software Process.* Reading, MA: Addison-Wesley, 1989.

30. Humphrey, W.S. *A Discipline for Software Engineering.* Reading, MA: Addison-Wesley, 1995.

31. Iacono, C.S.; Subramani, M.; and Henderson, J.C. Entrepreneur or intermediary: The nature of the relationship manager's job. In G. Ariav, C. Beath, J.I. DeGross, R. Hoyer, and C. Kemerer (eds.), *Proceedings of the Sixteenth International Conference on Information Systems.* Atlanta: Association for Information Systems, 1995, pp. 289–299.

32. Jordan, W.C., and Graves, S.C. Principles on the benefits of manufacturing. *Management Science, 41,* 4 (April 1995), 577–594.

33. Keidel, R.W.; Bell, S.M.; and Lewis, K.J. Rethinking organizational design: Executive commentary. *Academy of Management Executive, 8,* 4 (1994), 12–30.

34. Keller, R.T. Technology-information process fit and the performance of R&D project groups: A test of contingency theory. *Academy of Management Journal, 37,* 1 (1994), 167–179.

35. Kim, K.K., and Umanath, N.S. Structure and perceived effectiveness of software development subunits: A task contingency analysis. *Journal of Management Information Systems, 9,* 3 (Winter 1992–93), 157–181.

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

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

38. Klagge, J. Approaches to the iron cage. *Administration & Society, 29,* 1 (1997), 63–77.

39. Mantei, M. The effects of programming team structures on programming tasks. *Communications of the ACM, 24,* 3 (1981), 106–113.

40. Manz, C.C., and Sims, H.P. *Business Without Bosses: How Self-Managing Teams are Building High Performance Companies.* New York: John Wiley & Sons, 1993.

41. Miller, D., and Droge, C. Psychological and traditional determents of structure. *Administrative Science Quarterly, 31,* 4 (1986), 539–560.

42. Miller, D.; Droge, C.; and Toulouse, J. Strategic process and content as mediators between organizational context and structure. *Academy of Management Journal, 31,* 3 (1988), 544–569.

43. Mitchell, T.R. An evaluation of validity of correlational research conducted in organizations. *Academy of Management Review, 10,* 2 (1985), 192–205.

44. Nemetz, P.L., and Fry, L.W. Flexible manufacturing organizations: Implications for strategy formulation and organizational design. *Academy of Management Review, 13,* 4 (1988), 627–638.

45. Nidumolu, S.R. The effect of coordination and uncertainty on software project performance: Residual performance risk as an intervening variable. *Information Systems Research, 6,* 3 (September 1995), 191–219.

46. Nidumolu, S.R., and Knotts, G.P. The effects of reusability and customizability on the perceived process and competitive performance of software firms. *MIS Quarterly, 22,* 2 (June 1998), 105–137.

47. Olsen, N.C. Survival of the fastest: Improving service velocity. *IEEE Software, 12,* 5 (September 1995), 28–38.

48. Orlikowski, W.J. Integrated information environment or matrix of control: The contradictory implications of information technologies. *Accounting, Management and Information Technologies, 1,* 1 (1991), 19–42.

49. Ouchi, W.G. The relationship between organizational structure and organizational control. *Administrative Science Quarterly, 22,* 1 (March 1977), 95–113.

50. Ouchi, W.G. The transmission of control through organizational hierarchy. *Academy of Management Journal, 2,* 2 (1978), 173–192.

51. Ouchi, W.G., and Maguire, M.A. Organizational control: Two functions. *Administrative Science Quarterly, 20,* 4 (December 1975), 559–569.

52. Ravichandran, T., and Rai, A. Quality management in systems development: An organizational system perspective. *MIS Quarterly, 24,* 3 (2000), 381–416.

53. Rotemberg, J. Process- versus function-based hierarchies. *Journal of Economics and Management Strategy, 8,* 4 (1999), 453–487.

54. Sabherwal, R., and King, W.R. Decision processes for developing strategic applications of information systems: A contingency approach. *Decision Sciences, 23,* 4 (1992), 917–943.

55. Schoonhoven, C.B. Problems with contingency theory: Testing assumptions hidden within the language of contingency "theory." *Administrative Science Quarterly, 26,* 3 (1981), 349–377.

56. Snell, S.A. Control theory in strategic human resource management: The mediating effect of administrative information. *Academy of Management Journal, 35,* 2 (1992), 292–327.

57. Storey, J. The management of new office technology: Choice, control and social structure in the insurance industry. *Journal of Management Studies, 24,* 1 (1987), 43–62.

58. Thompson, J.D. *Organizations in Action.* New York: McGraw-Hill, 1967.

59. Tochlin, W. *The Research Methods Knowledge Base,* 2d ed. Cincinnati: Atomic Dog Publishing, 2000.

60. Upton, D.M. What really makes factories flexible? *Harvard Business Review, 73,* 4 (1995), 74–81.

61. Van DeVen, A.H.; Delbecq, A.L.; and Koenig, R.J. Determinants of coordination modes within organizations. *American Sociological Review, 41,* 2 (1976), 322–338.

62. Weber, M. *The Theory of Social and Economic Organizations.* New York: Free Press, 1947.

63. Weinberg, G. *The Psychology of Computer Programming.* New York: Van Nostrand Reinhold, 1971.

64. Zack, M.H. Jazz improvisation and organizing: Once more from the top. *Organization Science, 11,* 2 (2000), 227–235.

Appendix A. Details of Items

Competitive Performance

COMPARED TO YOUR COMPETITORS, how does your organization rate on each of the following: five-point scale: 1 = at the very bottom, 3 = in the middle, 5 = at the very top.

Product Cost Efficiency

1. Ability to produce software at low cost for current product lines.
2. Ability to charge competitive prices for software for current product lines.
3. The efficiency of software production for current product lines.
4. The productivity of your software developers for current product lines.

Market Responsiveness

1. The speed of response to new customer needs.
2. Ability to tailor software products to individual customer needs.
3. The speed at which new software markets can be entered.
4. The rate of introduction of new software products/services.

(Adapted from Nemetz and Fry [44].)

Process Performance

Please rate the speed with which your current software development approach(es) can be used to respond effectively to the following compared to your competitors in the same situation: 1 = slower than all, 2 = slower than most, 3 = in the middle, 4 = faster than most, 5 = faster than all.

Process Flexibility

1. Quantity of developers in the labor market decreases.
2. Quality of developers in the labor market decreases.
3. Customer's software product requirements change.
4. A new market for your software products opens.
5. New competitors enter your software market.
6. Changes in the basis of competition in your industry occurs.
7. New laws regulating your business are enacted.
8. Laws regulating your business are repealed.
9. New software technologies become available.
10. New hardware technologies become available.

Process Predictability

On average, how predictable is each of the following at the start of a software development project in your organization? Five-point scale: 1 = very unpredictable, 5 = very predictable.

1. Actual date when project will be completed.
2. Actual developer-months that will be required.
3. Actual budget that will be consumed by the project.
4. Actual resources that will be required to complete the project.
5. Actual quality of finished software.
6. Actual functionality of finished software.

Standardization

To what extent has your organization defined standards for the following for your software development projects? Five-point scale: 1 = no defined standards, 3 = general guidelines, 5 = detailed standards.

Standardization of Performance Criteria

1. Criteria to be met for software functionality.
2. Criteria to be met for software quality.
3. Criteria to be met for productivity.
4. Criteria to be met for budgets.
5. Criteria to be met for schedules.

(Adapted from Keller [34] and Nidumolu [45].)

Standardization of Methods

a. Front-end methods

1. Methods to be used for project management.
2. Methods to be used for analysis.
3. Methods to be used for design.

b. Back-end methods

1. Methods to be used for coding.
2. Methods to be used for testing.
3. Methods to be used for maintenance/update.
4. Methods to be used for documentation.

(Items derived from Conger [12] and Cusumano [14].)

Decentralization

How much discretion do individual project teams actually have in determining the following for their own projects? Five-point scale: 1 = no discretion, 3 = some discretion, 5 = complete discretion.

Decentralization of Performance Criteria

1. Criteria to be met for software functionality.
2. Criteria to be met for software quality.
3. Criteria to be met for productivity.
4. Criteria to be met for budgets.
5. Criteria to be met for schedules.

(Adapted from Keller [34] and Nidumolu [45].)

Decentralization of Methods

1. Methods to be used for project management.
2. Methods to be used for analysis.
3. Methods to be used for design.
4. Methods to be used for coding.
5. Methods to be used for testing.
6. Methods to be used for maintenance/update.
7. Methods to be used for documentation.

(Items derived from Conger [12] and Cusumano [14].)

Environmental Uncertainty

Please indicate where your organization fits between the two extremes shown:

1. 1 = our organization must rarely change its marketing practices to keep up with the market and competitors, 7 =our organization must change its marketing practices extremely frequently, for example, semiannually.
2. 1 = the rate at which our products/services become obsolete is very slow, 7 = the rate of obsolescence is very high.
3. 1 = actions of our competitors are easy to predict, 7 = actions of our competitors are extremely unpredictable.
4. 1 = demand and consumer tastes for our products/services are fairly easy to forecast, 7 = demand and tastes are extremely unpredictable.
5. 1 = our hardware technology is not subject to very much change and is well established, 7 = hardware technology changes very often and in major ways.
6. 1 = our software technology is not subject to very much change and is well established, 7 = software technology changes very often and in major ways.

Appendix B. Psychometric Analyses

Competitive Performance

COMPETITIVE PERFORMANCE DESCRIBES the software firm's performance relative to its competition and was determined from the viewpoint of both the development and the marketing manager. Specifically, it was measured by the respondent's assessment of the software firm's position *relative to its competitors* (1 = at the very bottom, 5 = at the very top) on the following.

Market Responsiveness. This concept describes the organization's ability to respond to changing market conditions and is important for coping with uncertainties in the demand for a firm's products. For software firms, it describes the firm's speed, relative to its competitors, in responding to changes in marketplace demands for its software products and is measured by the firm's (1) speed of response to new customer needs, (2) ability to tailor software products to individual customer needs, (3) speed at which new software markets can be entered, and (4) rate of introduction of new software products/services.

Product Cost Efficiency. This concept describes the efficiency with which the firm produces its products and is an important characteristic of both mass production firms and new forms of production such as mass customization. For software firms, product cost efficiency was measured by the firm's (1) ability to produce software at low cost for current product lines, (2) ability to charge competitive prices for software for current product lines, (3) efficiency of software production for current product lines, and (4) productivity of software developers for current product lines.

Test for Construct Validity. The factor structure of the confirmatory factor analysis using varimax rotation for the competitive performance dimensions is provided in Table B1. Whereas product cost efficiency consisted of four items, market responsiveness was composed of two items that evaluated the firm's rate of new product development and the speed of entry into new software markets. Two items had to be dropped from the scale for market responsiveness because they loaded above 0.5 on the other factor from what they were intended to measure.

Software Development Process Performance

Software development process performance operationalized in terms of software process flexibility and software process predictability.

Software Process Flexibility. We focus particularly on the speed of response to environmental changes because of its importance to software development [31, 47]. Respondents were asked to indicate the speed, when compared to competitors in the same situation (1 = slower than all, 5 = faster than all), with which their current software development approach could be used to respond effectively to changes in the following five business environment dimensions:

1. *Labor supply:* (a) quantity of developers in the labor market decreased, and (b) quality of developers in the labor market decreased.

Table B1. Confirmatory Factor Analysis for Competitive Performance

Measurement item	Product cost efficiency	Market responsiveness
Ability to produce software at low cost for current product lines	**0.52**	0.12
Ability to charge competitive prices for software for current product lines	**0.51**	0.31
The efficiency of software production for current product lines	**0.83**	0.35
The productivity of software developers for current product lines	**0.71**	0.38
The speed of response to new customer needs*	0.86	0.12
Ability to tailor software products to individual customer needs*	0.55	0.14
The speed at which new software markets can be entered	0.16	**0.96**
The rate of introduction of new software products	0.30	**0.69**

Notes: * Deleted from subsequent analysis as item loaded on a different factor from what was expected. Highest loading of item on factors are shown in boldface to aid interpretation.

2. *Customer needs:* (a) customer's software product requirements changed, and (b) a new market for the software products opened.
3. *Competition:* (a) new competitors entered the software market, and (b) changes in the basis of competition in the industry occurred.
4. *Regulation:* (a) new laws regulating the business were enacted, and (b) laws regulating the business were repealed.
5. *Technology:* (a) new software technologies became available, and (b) new hardware technologies became available.

Process Predictability. This construct can be defined as the ability of the organization to accurately estimate the needed resources, time and performance of its software projects. In this study, it was measured by the extent to which each of the following were predictable (1 = very unpredictable, 5 = very predictable) at the start of a software development project in their organization [27, 34]: (1) actual date of completion of project, (2) actual developer-months required for the project, (3) actual budget that would be consumed by the project, (4) actual resources that would be required to complete the project, (5) the actual quality of the finished software, and (6) the actual functionality of the finished software.

Test for Construct Validity. This was performed through confirmatory factor analysis using varimax rotation. The factor structure identified four, rather than six, factors (see Table B2). Five of the six software process predictability items loaded only on one factor, whereas the sixth item loaded just below 0.50. The flexibility items loaded on three—instead of five—factors. The items relating to labor and regulatory flexibility loaded on one factor each as expected, while those for flexibility regarding changes in customer needs, competition, and technology, all loaded on the same fac-

Table B2. Factor Analysis Results—Competitive Performance

Measurement item	Process predictability	Product market flexibility	Regulatory flexibility	Labor flexibility
Predictability of actual date of completion of a project	**0.83**	0.08	0.22	0.22
Predictability of actual developer-months for a project	**0.93**	0.10	0.12	0.08
Predictability of actual budget required for a project	**0.91**	0.14	–0.04	–0.03
Predictability of actual resources required to complete a project	**0.81**	0.19	–0.21	0.04
Predictability of actual quality of finished software	**0.51**	0.23	0.14	0.19
Predictability of actual functionality of software*	0.48	0.20	0.24	0.01
Flexibility when developers' quantity in labor market decreases	0.15	–0.07	0.04	**0.87**
Flexibility when developers' quality in labor market decreases	0.08	0.03	–0.04	**0.72**
Flexibility when customers' software requirements change	0.18	**0.63**	0.32	0.00
Flexibility when a new market for software products opens	0.05	**0.57**	0.18	–0.02
Flexibility when new competitors enter market	0.13	**0.69**	0.06	–0.13
Flexibility when changes occur in basis of industry competition	0.11	**0.54**	0.21	–0.05
Flexibility when new laws regulating business are enacted	0.06	0.18	**0.92**	0.00
Flexibility when laws regulating business are repealed	0.13	0.13	**0.89**	0.00
Flexibility when new software technologies become available	0.27	**0.79**	–0.06	0.18
Flexibility when new hardware technologies become available	0.13	**0.82**	–0.11	0.08

Notes: * Deleted from subsequent analysis due to loading below 0.50 on all factors. Highest loading of item on factors are shown in boldface to aid interpretation.

tor. This pattern of loadings could be reflective of the high interdependence between needs, competition and technologies in the software industry, where changes in one trigger changes in the other. This factor was labeled "product market flexibility" for subsequent analysis.

It is possible that the respondents may not have perceived any practical difference between process flexibility and market responsiveness. To assess whether there was common variance between these variables, a factor analysis of their measurement scales pooled together was also conducted. The analysis revealed four factors, that is, three for process flexibility and one for market responsiveness, demonstrating that these conceptually distinct constructs were also empirically different.

Standardization of Software Development Controls

Standardization of control was described by the extent to which the organization defined standards for its software development projects (1 = no defined standards, 5 = detailed standards) for the following items.

Standardization of Performance Criteria. (1) Criteria to be met for software functionality, (2) criteria to be met for software quality, (3) criteria to be met for productivity, (4) criteria to be met for budgets, and (5) criteria to be met for schedules. These items were derived from Keller [34] and Nidumolu [45].

Standardization of Methods. (1) Methods to be used for project management, (2) methods to be used for analysis, (3) methods to be used for design, (4) methods to be used for coding, (5) methods to be used for testing, (6) methods to be used for maintenance/update, and (7) methods to be used for documentation. These items were derived from Conger [12] and Cusumano [14].

Test for Construct Validity. An exploratory factor analysis using varimax rotation of all the standardization items pooled together revealed three factors as shown below (see Table B3). All the performance criteria items loaded as predicted on one factor, except for performance criteria regarding budgets, which also loaded at 0.50 on a second factor. This latter item was dropped from the scale for performance criteria standardization due to lack of discriminant validity. The methods items separated into two factors, one related to software development aspects such as project management, analysis and design, and the other related to aspects such as coding, testing, maintenance/update, and documentation. These two factors were labeled front-end and back-end methods standardization, respectively, a distinction that is commonly made in practice. The score for methods standardization was then computed by taking the first principal component from a principal components analysis of these two factors.

Decentralization of Software Development Controls

Decentralization of control was operationalized as the extent of *discretion* [55] that individual project teams actually had in determining the control process aspects identified above, that is, performance criteria and methods. Within each control process

Table B3. Factor Analysis Results—Standardization

Measurement items	Performance criteria	Back-end methods	Front-end methods
Criteria to be met for software functionality	**0.59**	0.47	0.08
Criteria to be met for software quality	**0.78**	0.42	0.18
Criteria to be met for productivity	**0.72**	0.13	0.27
Criteria to be met for budgets*	0.69	0.06	0.50
Criteria to be met for schedules	**0.69**	0.13	0.33
Methods to be used for project management	0.05	0.34	**0.77**
Methods to be used for analysis	0.20	0.23	**0.85**
Methods to be used for design	0.36	0.22	**0.78**
Methods to be used for coding	0.39	**0.58**	0.32
Methods to be used for testing	0.39	**0.61**	0.22
Methods to be used for maintenance/update	0.12	**0.85**	0.22
Methods to be used for documentation	0.06	**0.80**	0.19

Notes: * Dropped from analysis due to ambiguity due to loading above 0.50 on more than one factor (lack of discriminant validity). Highest loading of item on factors are shown in boldface to aid interpretation.

aspect, the same items were used as those above for standardization. The five-point Likert scale for each item below was anchored by 1 = no discretion, 5 = complete discretion.

Decentralization of Performance Criteria. (1) Criteria to be met for software functionality, (2) criteria to be met for software quality, (3) criteria to be met for productivity, (4) criteria to be met for budgets, and (5) criteria to be met for schedules.

Decentralization of Methods Controls. (1) Methods to be used for project management, (2) methods to be used for analysis, (3) methods to be used for design, (4) methods to be used for coding, (5) methods to be used for testing, (6) methods to be used for maintenance/update, and (7) methods to be used for documentation.

Test for Construct Validity. An exploratory factor analysis using varimax rotation (see Table B4) suggested that all the items loaded on two factors. The items related to performance criteria loaded above 0.50 on one factor, whereas the items related to methods loaded above 0.50 on a second factor. These factors correspond to the two decentralization variables in this study, that is, decentralization of performance criteria and decentralization of methods controls, respectively. The mean, standard deviation, and reliabilities of the measurement scales are provided in Table B5.

Table B4. Factor Analysis Results—Decentralization

Measurement items	Performance criteria	Development methods
Criteria to be met for software functionality	**0.70**	0.34
Criteria to be met for software quality	**0.75**	0.28
Criteria to be met for productivity	**0.65**	–0.20
Criteria to be met for budgets	**0.70**	0.06
Criteria to be met for schedules	**0.62**	0.35
Methods to be used for project management	0.09	**0.87**
Methods to be used for analysis	0.10	**0.87**
Methods to be used for design	0.13	**0.88**
Methods to be used for coding	0.44	**0.70**
Methods to be used for testing	0.10	**0.76**
Methods to be used for maintenance/update	0.19	**0.84**
Methods to be used for documentation	0.14	**0.73**

Note: Highest loading of item on factors are shown in boldface to aid interpretation.

Table B5. Reliabilities of Measurement Scales

Construct	Dimension	Number of items	Mean	Standard deviation	Cronbach's alpha
Standardization	Performance criteria	4	3.08	0.86	0.78
	Front-end methods	3	2.96	0.98	0.86
	Back-end methods	4	3.52	0.85	0.82
Decentralization	Performance criteria	5	3.13	0.67	0.72
	Methods	7	3.27	0.81	0.91
Process predictability		5	3.03	0.92	0.89
	Labor	2	3.20	0.71	0.79
Process flexibility	Market	6	3.34	0.57	0.84
	Regulatory	2	3.28	0.72	0.91
Product cost efficiency		4	3.64	0.69	0.81
Market responsiveness		2	2.99	0.76	0.83