Knowledge Partitioning in Outsourced Information Systems Development

Amrit Tiwana
Iowa State University
tiwana@iastate.edu

Under third review at Management Science

Acknowledgements: The support of Emory University, Russian Development Alliance, Embassy of Ireland in Washington DC, Ireland Development Agency, and India's NASSCOM is gratefully acknowledged. The inputs of the following individuals in the various stages of the study are also acknowledged: Anandhi Bharadwaj, Benn Konsynski, Arun Rai, Mark Keil, Bala Ramesh, Sridar Ramaswami, Ashley Bush, G. Premkumar, Dimitri Kroupski (Inform Link), Maria Dragan (Aplana Software), Pavel Konovalov (JS RCOM), Masha Mezhburd (Tengry software), Andrew Cooper (Ectaco), Vladimir Lando (Krista Software), Michael Vasilenko (TESIS), Alexander Sergeev (EpsylonTech), Sergey Ushakov (INT), Sergei Makedonski (Market-Visio/EDC), Valentina Bukina (SPIRIT), Irina Lantyukhova (Avalon Tree), Andrei Teterevenkov (SITech), Alexander Osipov (I-net Labs), Paul Bale (Sharptext Ltd), Seán Ryan (Aspen Grove Software), Colm Doherty (XIAM Ltd), Deepak Kumar (SoftwareDioxide), Uday Rangaswamy (Global Software), U. Subramanian (HCC Infotech), Anwer Bagdadi (Lawkim), Sanjeev Kulkarni (Advent Software), Ajai Chowdhry (HCL Infosystems), and S. Chari (Onicra).

Knowledge Partitioning in Outsourced Information Systems Development

ASBTRACT

Outsourced information systems development (ISD) traditionally relies on a requirements-driven "black-box" approach, where the need for the client and the vendor to deeply understand each others' knowledge domain is limited. This study examines the types of projects in which this approach suffices and the patterns of asymmetry in client-vendor knowledge overlaps that are necessary where it does not.

The underlying idea is that effectively organizing outsourced ISD requires solving inter-firm knowledge integration problems in the presence of specialization. The central thesis—building on the Jensen-Meckling theory—is that the appropriate pattern of client-vendor knowledge overlaps varies systematically with project novelty. Ex ante alignment in this pattern predicts interfirm knowledge integration during the ISD process, which in turn influences project outcomes. The model is tested using data on 209 projects outsourced to 209 software development organizations.

The findings offer novel theoretical insights into: (a) outsourced projects where client and vendor organizations require in-house knowledge *outside* their specialty, (b) appropriateness of various knowledge overlap asymmetry patterns, and (c) managing tensions created by deep specialization and knowledge appropriation hazards in outsourcing. Implications for both theory and practice are highlighted.

Knowledge Partitioning in Outsourced Information Systems Development

1. INTRODUCTION

Outsourced information systems development (ISD)—an inter-firm arrangement in which an external software development organization (hereinafter "vendor") develops a custom software application for the client firm—is becoming a widely used alternative to in-house development. Clients typically have considerable flexibility in choosing vendors in such arrangements, and some of these choices can make or break an outsourced project. Prior research on IS outsourcing has contributed significant insights into the structuring of contracts under uncertainty (Gopal et al. 2003; Wang et al. 1997) and agency problems *during* the development process (Ho et al. 2003). However, these works do not directly inform the choice of vendors at the outset of an ISD project.

Outsourced ISD has traditionally relied on a *black-box* approach, where the vendor relies on formal project requirements to devise a custom software solution for the client. This approach usually offers an effective means for embodying the client's knowledge about the application problem domain in the ISD process. Here, both the vendor and client organizations can enjoy the benefits of specialization: Neither organization needs in-depth knowledge of the others' domain for the outsourcing arrangement to work. The limited interfirm knowledge overlaps and specialization in *different* domains that characterize the black-box approach can however lead to poor project outcomes in some situations. The larger literature on product development has begun to recognize that outsourcing novel parts manufacturing requires higher technical knowledge in the client firm (Takeishi 2002). The underappreciated caveat is that acquiring deep knowledge of the partner firm's domain of expertise carries immediate costs but uncertain long-term benefits, requiring a finer grained understanding of when such knowledge is critically necessary. However, little is known about what such situations are or about the appropriate patterns of knowledge overlaps at the client-vendor boundary. The appropriate forms of such overlaps in the

presence of different *types* of novelty, whether they should be bilateral, and the theoretical logic for predicting when they are optimal thus are not addressed in either the new product development or in the software development literatures. This study focuses on this understudied issue. I address the following research question in this manuscript.

> **What patterns of knowledge overlaps are necessary at the client-vendor boundary in novel and routine projects?** In other words, when and why is it necessary for (1) a client to have in-depth technical knowledge and (2) the vendor to have in-depth knowledge about the client's business in outsourced ISD?

I build on the Jensen-Meckling (1992) theory to explain how and *why* the pattern of knowledge overlaps at the vendor-client boundary influences the outsourced ISD process. Central to the theory development is the idea that knowledge overlaps at the outset of a project influence knowledge integration during the development process, which in turn influences project outcomes. I test this model through a large-scale field study of 209 custom application development projects outsourced to 209 offshore vendors.[1]

This study makes three novel and original contributions. First, it shows how the value of different patterns of client-vendor knowledge overlaps at the outset of an outsourced ISD project is contingent on project novelty. I found that low knowledge overlaps suffice in routine projects but varied asymmetric patterns of overlaps are necessary in the presence of different types of novelty. The refined theoretical conceptualization of the patterns and influence of client-vendor knowledge overlaps is a unique contribution to the literature. Second, it explains when and why it is necessary for either firm to *de*specialize in its knowledge *in order to* specialize effectively. Third, it provides new insights into how clients with limited technical knowledge can effectively outsource novel projects without risking appropriation of their proprietary business knowledge.

The remainder of the paper is organized as follows. In the next section, I develop the research model, extending the well-known Jensen-Meckling approach. In §3, I describe the research methodology followed by the analyses in §4. I discuss the results in §5 and their implications in §6.

2. THEORETICAL DEVELOPMENT

2.1. Knowledge in Information Systems Development

ISD is a knowledge intensive activity that involves the coordinated application of a variety of specialized knowledge in formulating an appropriate software solution to solve a business problem (Adelson and Soloway 1985; Robillard 1999). Rus and Lindvall (2002) identify two types of pertinent knowledge: (1) technical knowledge that is used to custom-develop a system and (2) knowledge about the business application domain that guides its design. Technical knowledge is defined as knowledge about design (e.g., software architecture, design patterns, heuristics, best practices, and estimation models), programming (e.g., languages and tools), and software processes (e.g., analysis, design, testing, and debugging procedures). Business application domain knowledge is defined as knowledge about the business processes, business rules, policies and procedures, and the business objectives associated with the project's problem domain.

2.2. Knowledge Integration

These two types of complementary knowledge must be embodied in the design of the software for the resulting system to fit the client's needs (Faraj and Sproull 2000; Walz et al. 1993). The iterative nature of the ISD process requires that an understanding of the client's needs be also reflected in the intermediate artifacts of the ISD process such as project plans, requirements, specifications, and design definitions (Mookerjee and Chiang 2002; Robillard 1999). Following project-level extensions (Carlile and Rebentisch 2003; Okhuysen and Eisenhardt 2002) of Grant's (1996) conceptualization, I define this process of embodying business application domain knowledge with technical knowledge in the design of the software as *knowledge integration*. Knowledge integration thus occurs when knowledge of the application problem domain from the client organization is used together with technical knowledge from the vendor organization during the ISD process. The process of knowledge integration across the client-vendor boundary helps the client and vendor arrive at a common definition of the system being developed, ensuring that the design choices during the ISD process closely correspond to its intended purpose.[2]

Integration of complex forms of specialized knowledge is a persistent organizational challenge (Carlile and Rebentisch 2003; Grant 1996). Two attributes of the pertinent knowledge—fragmentation and stickiness—make its integration in outsourced ISD challenging. First, critical insights, ideas, and expertise are often dispersed on either side of the client-vendor organizational boundary (Ramesh 2002): A project's business needs reside in the client organization but are needed in the vendor organization while the knowledge needed by the client to monitor the project is generated in the vendor organization. Second, some of this knowledge is *sticky*, i.e., so context-dependent that it is difficult to transfer across the client-vendor boundary (von Hippel 1994).[3] Unless such sticky knowledge is adequately understood by the other firm, knowledge integration cannot be successful.

As I describe in §2.3.1, fragmentation and stickiness are less problematic in routine, well-understood design problems because formal requirements and established development processes collectively help convey to the vendor the problem that the application is intended to solve and its scope. However, these mechanisms flounder in the presence of novelty, where the differences in client-vendor knowledge increase stickiness of different specialized knowledge that must be used in the development process. Two types of novelty—conceptual newness (discussed in §2.3.2) and process newness (discussed in §2.3.2)—raise knowledge integration challenges because they make it difficult to "unstick" sticky knowledge for use across the client-vendor boundary. I next discuss how knowledge "partitioning"—the distribution of technical and business application domain knowledge across the client and vendor organizations at the outset of a project—influences the knowledge integration process in routine and novel ISD projects, and in turn project outcomes.

2.3. Knowledge Partitioning

Knowledge partitioning refers to the pattern of distribution of technical and business application domain knowledge across the client and vendor organizations at the outset of a project. The most common pattern is the black-box pattern of knowledge partitioning. Here, the client does not possess detailed technical knowledge about the project and the vendor does not have detailed knowledge of the client's business (although some minimal knowledge overlap is always necessary). There are three possible deviations

from the black-box pattern: (1) both firms possess in-depth knowledge of both types, (2) the vendor possesses in-depth understanding of the client's business application domain, or (3) the client possesses in-depth understanding of the project's technical intricacies. The last two patterns represent two different patterns of knowledge partitioning asymmetry. These patterns are illustrated in Figure 1.

The optimal pattern is one that allows successful knowledge integration during the development process while imposing minimal costs for both the client and vendor. For a variety of reasons,[4] vendors and clients are predisposed against increasing their knowledge about the others' domain beyond a minimally-sufficient black-box level, which I show in §2.3.2 suffices in routine projects but not in novel projects.

2.3.1. Predicting the effect of knowledge partitioning on knowledge integration

The Jensen-Meckling (JM) (1992) model provides a theoretical lens for analyzing the effect of knowledge partitioning at the outset of a project on knowledge integration during the development process. The crux of the JM model is that decision rights and the knowledge that is useful for exercising those decision rights must be collocated for effective decision-making. Decision rights are defined as the allocation of decision-making responsibilities across the client and vendor organizations. The key decision rights in outsourced ISD relate to: (1) design, i.e., the features, attributes, functionality, and their implementation in the software and (2) validation of the design by the client during the development process. I label these *design decision rights* and *validation decision rights* respectively. In outsourced ISD projects, the locus of the former is the vendor organization while the latter is the client organization.

The extent to which complementary business application problem domain knowledge and technical knowledge dispersed across the client and vendor organizations is utilized during the development process represents effective decision making in the JM sense. The effective exercise of these decision rights during the ISD process is reflected in successful knowledge integration. In contrast, poor knowledge integration creates a disconnect between the software design and actual needs, which is symptomatic of ineffective decision rights exercise during the ISD process. The least remediable defects

in software stem from misunderstood or inaccurately identified client requirements (Hoopes and Postrel 1999). Knowledge integration is therefore an appropriate dependent variable in the proposed model.

I build on JM's theory to first develop the logic for why black-box knowledge partitioning is sufficient in routine ISD projects (§2.3.1). I then develop the theoretical logic for two salient types of novelty where different asymmetric patterns of overlaps in specialized knowledge become necessary. The underlying logic is that knowledge overlaps at the outset of the project influence knowledge integration during the ISD process, which in turn influences project outcomes—an idea for which I subsequently provide empirical support. The central idea is that different knowledge partitioning patterns are more conducive to knowledge integration during the ISD process in the presence of different *types* of novelty (§2.3.2 and §2.3.3). These scenarios are summarized in Figure 1.

<< INSERT FIGURE 1 HERE >>

2.3.2. TYPE A: Routine Projects Involving Neither Conceptual Newness nor Process Newness

Consider first the scenario where a software project is routine i.e., it is neither conceptually new nor involves development processes that are new to the vendor. This represents a relatively well-definable problem space for which the initial state, the goal, and a set of possible operations to reach the goal from the initial state are available (Robillard 1999). A conventional black-box knowledge partitioning pattern suffices in this case for two reasons. First, formal project requirements can faithfully define the problem space and establish the interdependencies between the design of the software and the client's business needs. This allows the vendor to effectively exercise design decision rights. Second, the use of an established development process provides a mutually understood syntax for the vendor to provide information generated during the development process to the client, which allows the client to validate and refine the vendor's understanding of the application problem domain (Carlile and Rebentisch 2003; Vessey and Conger 1993). Therefore constraints and unstated requirements that might have not been fully captured in the preliminary requirements can be integrated during the development process. Such routinized, time-ordered interactions allow the client to effectively exercise validation decision rights and thus facilitates successful knowledge integration during the ISD process (Chatzoglou and Macaulay 1996;

Orlikowski 2002).[5] Knowledge integration across the client-vendor boundary can thus be achieved without requiring the client to possess in-depth technical knowledge or for the vendor to understand the intricacies of the client's business. This represents the black-box pattern of knowledge partitioning.

> ***Hypothesis 1:*** *Extensive client-vendor knowledge overlaps are not necessary for successful knowledge integration in projects that involve neither conceptual nor process newness.*

In the presence of novelty, the absence of extensive overlaps that characterize the black-box knowledge partitioning pattern raise the stickiness of some knowledge; thus its integration is more challenging. The type of knowledge that is stickier depends on the type of associated novelty. I consider the effects of two salient types of novelty—conceptual newness and process newness (Adler 1995). I focus our attention on the source of sticky knowledge associated with both instances and the difficulties that are encountered in integrating it at the locus of the associated decision rights. I show that asymmetric knowledge partitioning—or client-vendor knowledge overlap—then becomes necessary for successful knowledge integration. Asymmetry of knowledge partitioning can take two forms as illustrated in cases B and C in Figure 1. The forthcoming logic for both is based on the JM model and is summarized in Table 1.

<< INSERT TABLE 1 HERE >>

2.3.3. TYPE B: Projects involving Conceptual Newness

Conceptual newness is defined as the extent to which a project's design and application functionality involves concepts that are new to the vendor. The sticky knowledge in conceptually new projects pertains to the business needs of the client: Unique problems, novel ideas, and idiosyncratic project concepts that are conceptually new to the vendor cannot be faithfully conveyed as discrete, objective formal requirements (Rowen 1990). The source of such sticky knowledge is the client organization but its locus of use is the vendor organization. Its integration in the development process requires "unsticking" such knowledge from the client organization for use in the vendor organization.

The vendor however is unlikely to be able to understand the client's problem space or accurately interpret the client's intentions with the limited complementary knowledge about the client's business that characterizes black-box knowledge partitioning. Thus differences in client-vendor knowledge localize knowledge about the problem domain in the client organization. As a result, inconsistent, equivocal, or

incomplete interpretation of the client's needs by the vendor can lead to design decisions that are incompatible with the client's project objectives. This reflects poor knowledge integration during the ISD process.

Understanding the client's idiosyncratic needs then requires that the vendor possess a higher level of knowledge about the client's business application domain than is the norm in the black-box model. Therefore, in conceptually new projects, higher vendor knowledge about the client's business application domain facilitates successful knowledge integration. This asymmetric knowledge partitioning pattern is illustrated in case B in Figure 1.

> ***Hypothesis 2:*** *Higher levels of client business application domain knowledge in the vendor organization enhances knowledge integration in projects with high conceptual newness.*

2.3.4. TYPE C: Projects Involving Process Newness

Process newness is defined as the extent to which a project involves development processes that are new to the vendor. Process newness arises from the use of a software development methodology or software development tools with which the vendor has limited prior experience. Recall from §2.3.2 that vendors rely on established development processes to convey information generated *during* the development process—about system architecture, functionality, feature definitions, user interface, and test cases—that clients use to validate the vendor's interpretations of the problem space. For this reason, it is common for clients to evaluate vendor performance based on adherence to prescribed development procedures (Kirsch et al. 2002).

When the development process is new to the vendor, the stickiness of such knowledge generated during the development process is higher.[6] The source of such sticky knowledge is the vendor where the actual development occurs but the locus at which it is used for monitoring and control purposes is the client organization. The bottleneck to knowledge integration is the absence of a mutually understood process for representing information that emerges during the ISD process to the client. This encumbers the vendor's ability to convey such information in a pattern or to render it in a form that the client can effectively use to distinguish effective from ineffective development processes and assess the in-progress artifacts, hindering successful knowledge integration.[7]

Utilizing such information to exercise the validation decision rights then requires that the client possess a higher level of technical knowledge relative to the black-box scenario. Such knowledge can help the client lower the stickiness of knowledge generated in the vendor organization and more effectively use it to monitor, control, and steer the project. Thus, an asymmetric knowledge partitioning pattern where the client's technical knowledge is high becomes necessary for successful knowledge integration (case C in Figure 1).

> ***Hypothesis 3:*** *Higher technical knowledge in the client organization enhances knowledge integration in projects with high process newness.*

2.3.5. TYPE D: Projects Involving Both Conceptual Newness and Process Newness

When a project involves both conceptual and process newness, both types of knowledge integration challenges simultaneously arise in the development process. The client will therefore need higher technical knowledge and the vendor higher business application domain knowledge, above and beyond the black-box level of overlaps described earlier (illustrated in case D in Figure 1).

> ***Hypothesis 4:*** *Projects that simultaneously involve high conceptual and process newness require higher levels of technical knowledge in the client firm and higher levels of business application domain knowledge in the vendor firm for successful knowledge integration.*

To summarize the key thesis, the deviations from the black-box knowledge partitioning pattern that are most conductive to knowledge integration across the client-vendor boundary vary systematically with project novelty. Project-specific alignment in the knowledge partitioning pattern will lead to successful knowledge integration during the development process, and in turn improve project outcomes.

2.4. Controls: Integrating Mechanisms

Both firms can implement mechanisms that help move sticky knowledge across the client-vendor boundary to the locus of the associated decision rights, thus enhancing knowledge integration *during* the ISD process. To fully discern their effects from knowledge partitioning, the following variables identified in the preliminary interviews (details of the interviews appear in §3.1) were used a controls. (Since the dependent variable is knowledge integration, I do not control for known predictors of downstream ISD performance.) *Development coordination tools* that provide access to project artifacts during the

development process can improve knowledge integration, which is reflected in fewer design defects, identification of inconsistencies in the design, and higher overall design quality. *Client-vendor interaction* over the course of the project lowers stickiness of knowledge across the client-vendor boundary (Szulanski 1996), thus helps integrate client needs that are not captured at the outset of the development process (Gottesdiener 2003). Higher effort invested by the vendor in the front-end conceptual design phase of the ISD process (*architectural design effort)* improves knowledge integration during the formulation of the high-level design of the system from which subsequent ISD phases follow. Vendors who have more mature software development capabilities are more likely to have processes in place that might improve knowledge integration with the client during the development process. Such *capability maturity* has previously been associated with high system design quality (Harter et al. 2000). Since knowledge integration is accomplished by individuals, it is inherently more challenging in larger teams than in smaller teams (*team size*). Finally, I controlled for prior client-vendor *collaboration history*. Firm-level variables such as client organization size, vendor organization size, and client industry attributes are not included in the analyses since the unit of analysis is the project. Given the absence of empirical support for these in prior work and questionnaire length constraints, only control variables that were identified most frequently by the interviewees were included.

3. DATA AND MEASUREMENT

A multi-stakeholder field survey of 209 projects was conducted during 2002-2003 to test the hypotheses. I contacted 818 software development organizations through the three largest global software consortia in Russia, Ireland, and India. These respectively were the Russian National Software Development Alliance, Irish Investment and Development Agency (IDA), and India's National Association of Software and Service Companies (NASSCOM). To maintain comparability, only projects completed for US clients were included in the study.

Data for each project were collected from two key informants from each organization—lead project managers in the vendor organization and their client-side liaison managers. The data collection

was done in two phases, which were preceded by the questionnaire development interviews. In the first phase, I asked the CEO or president of each vendor organization to identify one major project that they had recently completed for a US client. The vendor questionnaire was forwarded to the lead manager of this project in each vendor firm. I also collected self-reported data on the percentage by which each project exceeded the original schedule and budget. I obtained 232 vendor-side responses. Following this stage, I asked the project manager to forward a shorter survey to the primary liaison in the client organization.[8] Only the 209 projects for which dyadic datasets were collected are included in the final analysis.

The breakdown of the response rates was the following: full sample 28.4% (232 responded/ 818 contacted); Russia 33.5% (59/176); Ireland 29.5% (54/183); India 25.9% (119/459). The overall response rate of 28.4% (matched pairs: 25.5%) compares favorably to typical field studies involving managers.[9] Additional checks suggest that non-response bias is not a persuasive threat to our findings,[10] although the sample is biased towards more successful projects (the threats of this bias are assessed in §5.1).

3.1. Questionnaire Development

The questionnaire used existing measurement scales where possible. New scales were developed for client's technical knowledge, vendor's business application domain knowledge, and knowledge integration based on detailed interviews with 19 software project managers (7 in Russia, 6 in the US, 2 in Ireland, and 4 in India) and seven academic experts. Responses were measured using multi-item 7-point reflective Likert scales and Guttmann scales. Principal components analysis with orthogonal rotation was used to assess the factor loading patterns for each construct. All scales exhibited convergent validity (indicated by the reliability coefficients exceeding 0.7 and eigen values exceeding unity) and discriminant validity (indicated by factor analyses). (The details of this analysis are omitted for space considerations and are available from the author). The questionnaire items, the responding organization, and reliability coefficients for each construct are summarized in the ONLINE SUPPLEMENT. The inter-construct correlations, means, and standard deviations are summarized in Table 2.

<< INSERT TABLE 2 HERE >>

Client's technical knowledge was measured as the client's initial knowledge about detailed technical design, technical constraints, development methodologies, software testing and debugging procedures, and the programming language and environment. ***Vendor's business application domain knowledge*** was measured as the vendor's knowledge the client's business processes, project objectives, business routines, business rules implemented in the system, and interoperability with client systems. ***Project's conceptual newness*** was measured using a 4-item Guttmann scale adapted from Takeishi (2002). The scale follows Adler's (1995) approach that assessed whether the project concepts and design were: (a) proven carryovers from earlier projects with minor refinements, (b) major refinements, (c) completely new but based on proven concepts, or (d) completely new and unproven. ***Project's process newness*** similarly assessed whether the ISD processes used in the project were: (a) proven carryovers from earlier projects used with minor refinements, (b) major refinements, (c) new methodology or development tools, but based on existing ones, or (d) entirely new methodology and development tools. In both cases, values of (a) and (b) were dummy coded as low and (c) and (d) as high. ***Knowledge integration*** was measured using five items that assessed the extent to which during the development process the vendor organization leveraged the client's expertise in various functional areas, the client and vendor combined their unique perspectives in developing project ideas, the client and vendor developed a shared understanding, the client and vendor collaboratively made decisions on how to maximize project outcomes, and the vendor utilized the client's expertise during the development process. A detailed discussion of each item in the scale and comparisons with similar constructs in the literature appears in the ONLINE SUPPLEMENT, which also summarizes measures for project performance and the *control variables* (development coordination tools, client-vendor interaction, architectural design effort, maturity of vendor's software development capabilities).

3.2. Descriptive Statistics

On average, each project involved 208 person-hours of development effort (s.d. 487 person-hours). The duration of the projects ranged from one month to four years, with an average duration of about 11 months (s.d. 9.6 months). The average total project team size was 20 people (s.d. 29.3).[11] This large

standard deviation suggests that the sample has considerable variation in team sizes. Of the vendors that had worked previously with the same client on earlier projects (59% of the projects in the sample), the average relationship history spanned about 3.6 years. The CMM levels of the vendors varied from none to 5. On average, the vendors in the study had been in the software development business for about 7 years (s.d. 6.2). On average, the projects in the sample exceeded budget and schedule estimates by 15.7% (s.d. 16.1%) and 18.9% (s.d. 19.7%). (The distribution of projects in the sample for both types of novelty is summarized in the ONLINE SUPPLEMENT.)

4. ANALYSES AND RESULTS

A 4-step hierarchical regression model was used to test the hypotheses. The control variables were first introduced in the model (step I), followed by the two knowledge and two novelty variables (step II). The hypotheses were then tested by adding 2-way (step III) and 3-way (step IV) residual-centered interaction terms.[12] The hypothesis testing strategy is described in §4.1 and the results of these tests in §4.3.

4.1. Hypothesis Testing Approach

Hypotheses 2 and 3 (2-way interactions) were tested by adding two 2-way interactions to the model (step III). (These are the two asymmetric knowledge partitioning scenarios B and C in Figure 1.) Hypothesis 2 proposed that higher business application domain knowledge in the vendor organization enhances knowledge integration in conceptually new projects. To test this hypothesis, I added a 2-way interaction term between vendor business knowledge and conceptual newness to the model. The hypothesis would be supported if the interaction is positive and significant. Hypothesis 3 proposed that higher levels of technical knowledge in the client organization enhance knowledge integration in projects with high process newness. To test this hypothesis, I added another 2-way interaction term between client technical knowledge and process newness to the model. The hypothesis would be supported if the interaction is positive and significant. These hypotheses can only be tested in the absence of the 3-way interactions (step III); in their presence in the model (i.e., step IV), the 2-way interaction terms reflect only the conditionalized simple effects for when the other variables in the 3-way interaction terms equal

zero (Jaccard and Turrisi 2003: 45). Since both these hypotheses are unidirectional, one-tailed T-tests are appropriate for assessing statistical significance (critical T-value ≤ 1.67).

Hypotheses 1 and 4 (3-way interactions) were tested by adding two 3-way interaction terms to the above model (step IV). Hypothesis 1 proposed that black-box partitioning suffices in routine projects; neither high levels of technical knowledge in the client organization nor high levels of application domain knowledge in the vendor organization enhance knowledge integration when conceptual and process newness are simultaneously low. To test this hypothesis, two 3-way interaction terms were used: (1) the interaction between client technical knowledge, conceptual newness, and process newness and (2) the interaction between vendor business knowledge, conceptual newness, and process newness. In a model containing these product terms, the *main* effects of client technical knowledge and vendor business knowledge reflect conditional relationships when (conceptual newness*process newness) equals zero (i.e., routine projects) (Jaccard and Turrisi 2003: 45). The hypothesis is supported if the neither the *main* effect of client technical knowledge nor vendor domain knowledge is significant when the two 3-way interaction terms are present in the same model (step IV). Hypothesis 4 proposed that in projects with simultaneously high conceptual and process newness, having both high levels of technical knowledge in the client organization and high levels of application domain knowledge in the vendor organization enhances knowledge integration. This hypothesis is supported if both the aforementioned 3-way interaction terms simultaneously exhibit a positive relationship with knowledge integration. A two-tailed T-test is necessary for assessing statistical significance of this hypothesis (critical T-value ≤ 1.98) because it is bidirectional (with Hypothesis 1 being the complementary tail of the relationship).

4.2. Assessment of Control Variables

As shown in Table 3, the controls for vendor nationality were non-significant; further analysis could proceed by pooling all projects. In the order of their importance, the following controls were statistically significant in the full models: client-vendor interaction (β_4), team size (β_7), CMM level (β_6). The most important control variable was client-vendor interaction. (To ascertain that this result was not an artifact

of measurement problems, discriminant validity between this variable, the independent variables, and knowledge integration was verified.) Since the choice of controls was based on anecdotal evidence and in absence of prior empirical work linking them to knowledge integration, the non-significance of the other three controls is not especially surprising.

4.3. Hypothesis Testing Results

The hypotheses were tested using the strategy described in §4.1. The 4-step hierarchical regression results (path coefficients, their standardized error terms, and statistical significance) are summarized in Table 3 (statistically significant results are shown in **bold**).

<< INSERT TABLE 3 HERE >>

Knowledge partitioning in projects involving conceptual newness (Hypothesis 2). Hypothesis 2 predicted that higher levels of business application domain knowledge in the vendor organization enhances knowledge integration in projects with high conceptual newness. This was tested by assessing the significance of the 2-way interaction between *vendor business domain knowledge* and *conceptual newness*. This corresponds to the β_{13} term in Table 3, step III. The interaction was positive and significant ($\beta = 0.271$; T-value 2.56; $p<0.01$, one-tailed test), supporting Hypothesis 2.

Knowledge partitioning in projects involving process newness (Hypothesis 3). Hypothesis 3 predicted that higher levels of technical knowledge in the client organization enhances knowledge integration in projects with high process newness. This was tested by assessing the significance of the 2-way interaction between *client technical knowledge* and *process newness*. This corresponds to the β_{14} term in Table 3, step III. The interaction term was positive and significant ($\beta = 0.263$; T-value 2.43; $p<0.01$, one-tailed test), supporting Hypothesis 3.

Knowledge partitioning in routine projects (Hypothesis 1). Hypothesis 1 predicted that the black-box pattern suffices in routine projects i.e., neither higher levels of client technical knowledge nor vendor business domain knowledge improve knowledge integration. This hypothesis was tested by assessing the significance of the main effects of these two types of knowledge when conceptual newness and process newness are simultaneously low. This conditionalization occurs through the introduction of two 3-way

interaction terms between (*conceptual newness*process newness*) and (a) *vendor business knowledge* and (b) *client technical knowledge* in step IV (Jaccard and Turrisi 2003: 45). These correspond to the β_9 and β_{10} terms in Table 3, step IV. Both coefficients must simultaneously be non-significant for the hypothesis to be supported. The conditionalized main effects of both *vendor business knowledge* (β_9 = 0.046; T-value 0.038; non-significant, one-tailed test) and *client technical knowledge* (β_{10} = -0.162; T-value -1.48; non-significant, one-tailed test) were non-significant, supporting the hypothesis.

Knowledge partitioning in projects simultaneously involving conceptual and process newness (Hypothesis 4). Hypothesis 4 predicted that in projects with simultaneously high conceptual and process newness, knowledge integration in enhanced by having both high levels of technical knowledge in the client organization and high levels of application domain knowledge in the vendor organization. The two 3-way interaction terms corresponding to β_{15} and β_{16} in step IV must simultaneously be significant to support this hypothesis. Neither the 3-way interaction for *vendor business domain knowledge* (β_{15} = -0.23; T-value -1.76; non-significant, two-tailed test) or *client technical knowledge* (β_{16} = 0.15; T-value 0.77; non-significant, two-tailed test). Hypothesis 4 was therefore not supported.

To summarize the key results, knowledge integration is enhanced by: (1) higher vendor knowledge of the business application domain knowledge in conceptually new projects and (2) higher client technical knowledge in projects that involve process newness. Neither of these knowledge overlaps enhance knowledge integration in routine projects.

5. DISCUSSION

The key thesis of the model was that fit between the type of novelty associated with a project and the knowledge partitioning pattern that characterizes the client-vendor dyad at the outset of the project influences knowledge integration during the ISD process, which is critical to the development of an appropriate software solution. The ideal knowledge partitioning pattern is one that facilitates successful knowledge integration at the least cost to either the client or the vendor. Since either organization incurs costs in acquiring knowledge outside of ones immediate domain, a pattern that requires the least deviation from the black-box pattern of low knowledge overlaps is preferable to either firm. The results show that

extensive overlaps in knowledge are unnecessary in routine projects (H1). They also provide new insights into how clients and vendors must deviate from this black-box pattern in novel projects, i.e., when: (1) the vendor needs higher knowledge about the client's business application domain and (2) the client needs higher technical knowledge relative to that in the black-box partitioning scenario. The former is necessary in conceptually new projects (H2) and the latter in projects that rely on new software development processes (H3), consistent with the effects hypothesized based on Jensen-Meckling's theory in §2.

Vendor's business application domain knowledge. The finding that higher levels of knowledge about the client's business are necessary in the vendor organization when a project is conceptually new is consistent with the idea that such knowledge allows the vendor to integrate relatively novel information about the project's concepts, ideas, and objectives from the client organization during the ISD process. Although prior case studies have recommended that software development organizations should unconditionally increase the business understanding of their staff (Walz et al. 1993), the absence a significant effect of vendor business application domain knowledge in routine projects cautions that the costs incurred in developing such knowledge overlaps do not carry commensurate benefits if a vendor typically undertakes routine application development projects.

Client's technical knowledge. Higher technical knowledge in the client organization is necessary when a project involves process newness. This is consistent with the idea that higher technical knowledge in the client organization facilitates utilization of information generated through new processes during development in the vendor organization to exercise validation decision rights. Such knowledge overlap provides no benefits in routine projects. Interactions between the knowledge partitioning pattern and project novelty add significant explanatory power to the model, as indicated by a significant ΔR^2 12.4%.

The lack of support for Hypothesis 4 suggests that in the simultaneous presence of conceptual and process newness, neither type of knowledge overlap improves knowledge integration. An explanation is that in such projects, even extensive knowledge overlaps cannot facilitate knowledge integration because different types of knowledge are *simultaneously* highly-sticky in both organizations. Surprisingly, the effect of vendor business domain knowledge borders near significance in the negative direction here,

suggesting that it might *impede* knowledge integration. One explanation for this is that vendors might make assumptions about the correctness of the system's design before it is fully validated by the client. This aura of overconfidence that results from prior experience with seemingly-similar ISD projects has been documented in a prior study of electronic tax filing systems (Martins and Kambil 1999).

Relationship between knowledge integration and project outcomes. While knowledge integration was the focal dependent variable in the study, it might be irrelevant if it bears no relationship to other measures of project success (which is causally downstream from knowledge integration). The relationship between knowledge integration and team outcomes has previously been established (Faraj and Sproull 2000; Mookerjee and Chiang 2002; Walz et al. 1993) and is not the focus of this paper; I conducted additional tests to confirm whether higher knowledge integration is associated with better project outcomes (effectiveness and efficiency). I found a positive and statistically significant relationship between knowledge integration and client assessments of development effectiveness ($\beta = 0.24$, T-value = 2.84, $p<0.01$) and a negative association with budget overruns ($\beta = -0.238$, T-value = -2.28, $p<0.05$) that was a proxy for *in*efficiency. Sobel mediation tests further confirmed that the influence of the independent variables on project outcomes is fully mediated by knowledge integration.[13]

5.1. Limitations

Six limitations of this study should be noted in generalizing its findings. First, knowledge integration was used to assess how effectively decision rights were exercised during the ISD process; future work should validate the findings using a more direct measure for decision right exercise effectiveness, possibly building on Fama and Jensen's (1983) distinction between decision control and decision management. Second, newness was assessed only from the vendor's perspective; a fuller understanding of knowledge partitioning also requires examination of newness to the client organization. Third, all projects in the study were offshore outsourcing arrangements where both firms were geographically dispersed; it remains unclear how well the results will generalize to internal projects or those outsourced to an on-shore vendor (see discussion under future research directions). Fourth, the model did not control for the use of various ISD control mechanisms by the client (Kirsch et al. 2002), which plausibly might help overcome some

knowledge integration problems in the presence of novelty. Fifth, ex ante knowledge partitioning patterns were measured cross-sectionally; thus might be distorted by recall bias or influenced by learning that might have occurred over the course of the project. This inherent constraint of a cross-sectional design poses a serious limitation to the generalizability of the results and requires longitudinal or multi-period measurement in future work. Finally, the sample was biased towards successful projects (higher effectiveness). To assess how this imposes generalizability boundary conditions, I checked for differences across all key variables between more successful (top 50 percentile) and less successful projects (bottom 50 percentile). More successful projects had: (a) significantly higher levels of client-vendor interaction (F-value 5.462, $p<0.001$), (b) greater development coordination tools usage (F-value 5.99; $p<0.001$), and (c) higher client technical knowledge (F-value 4.03; $p<0.001$), and consistent with the literature also had higher levels of knowledge integration (F-value 7.92; $p<0.001$).[14] This implies that the results are more readily generalizable to client-vendor dyads that have highly collaborative relationships but less readily to those with arm-length relationships; to technically more knowledgeable clients; and to projects in which vendors extensively use development coordination tools.

6. IMPLICATIONS AND DIRECTIONS FOR FUTURE RESEARCH

Organizations face an inherent tension between specializing in their own domain and acquiring knowledge about the domains of other organizations with which they collaborate. This poses a dilemma for outsourcing ISD projects that tread into novelty: While client-vendor dependencies in developing a custom software application create the need for knowledge integration, their differences in knowledge can constrain it.

One solution to this dilemma is for collaborating firms to possess deeper knowledge of their partner's domain. While the need for such overlaps is recognized in prior research (Takeishi 2002), their appropriate form and contingent value are theoretically undeveloped. Furthermore, prior work has not recognized the constraint that organizations incur immediate costs but derive uncertain benefits in developing such overlaps. The perspective advanced here is that when the pattern of client-vendor

knowledge partitioning fits the type of novelty in a project—as predicted by the Jensen-Meckling theory—successful knowledge integration across the client-vendor boundary can be achieved during the development process. This in turn leads to better project outcomes.

The study offers new insights into the various patterns of knowledge partitioning and the theoretical rationale for their value. The results show that low knowledge overlaps—to which both firms are predisposed—suffice in routine projects but not in novel projects, where different forms of knowledge asymmetry become necessary. The pattern of knowledge overlaps that is most conducive to knowledge integration systematically varies with *type* of project novelty. I discuss next the theoretical contributions, implications for knowledge management and ISD research, and five avenues for future research.

6.1. Theoretical Contributions

The paper makes three novel and original theoretical contributions. The primary contribution is a theoretical explanation for how the value of different patterns of inter-firm knowledge partitioning in outsourced ISD varies systematically with project novelty. The idea that certain patterns of knowledge overlaps lead to superior project outcomes through the process of knowledge integration is noteworthy because it theoretically separates performance from its source in distinguishing capable behaviors from project outcomes.

While recent new product development research has acknowledged that the absence of knowledge overlaps in manufacturing outsourcing can be problematic in technologically innovative products, neither a fine grained conceptualization of the forms of knowledge partitioning nor a theoretical explanation for their effect in outsourcing arrangements have been developed. The development of the theoretical link from knowledge partitioning to knowledge integration (and in turn on project outcomes) is an original contribution of this paper. The conceptual development of the four patterns of knowledge partitioning and the associated extension of Jensen-Meckling's theory to predict their influence on project outcomes via knowledge integration are a noteworthy advances from antecedent works on knowledge partitioning such as Takeishi (2002). The idea of project-specific fit in client-vendor knowledge partitioning developed here also contributes new insights to the ISD outsourcing literature, which has

focused primarily on the agency-theoretic issues *during* the development process and on contract structuring under uncertainty (e.g., Gopal et al. 2003; Ho et al. 2003; Lee and Kim 1999) but offers little guidance for the choice of vendors at the outset of a project.

A second contribution of this work is in showing when and why firms must maintain in-house knowledge *outside* their specialty *in order to* specialize effectively. This is a notable insight considering that both clients and vendors have strong disincentives to do so—a subtlety is recognized neither in the software development literature or in the new product development literature. The insight that the value of knowledge despecialization is contingent on novelty is a notable refinement over the "more is better" assumption in the ISD literature.

The third contribution is in showing the types of outsourced projects where the client firm might need to share closely-guarded knowledge about its business with a vendor. The results show that this is only necessary in conceptually new projects. The results also show that firms with limited technical knowledge can successfully outsource novel projects by minimizing deviation from established processes. The development of new scales to measure knowledge partitioning variables is also noteworthy. Sampling vendors from Russia, Ireland, and India where the majority of global outsourced software development occurs increases the generalizability of our findings.

6.2. Implications for Research

Implications for knowledge management research. The overarching theoretical implication of the study is that effectively organizing outsourced ISD requires solving inter-firm knowledge integration problems in the presence of specialization. The first implication of these results for knowledge management theory is that the same client-vendor knowledge partitioning patterns can have vastly different effects, which vary systematically with project novelty. Unlike routine projects where low knowledge overlaps suffice, different asymmetric knowledge overlaps become necessary for integration of either sticky business application domain knowledge from the client firm or technical knowledge from the vendor firm in novel projects.

The second implication of the results is that it is sometimes necessary for firms to despecialize in their knowledge to task-specialize effectively. The decision to acquire specialized knowledge about the other organization's domain can be systematically guided by predicting the type of knowledge that is stickier and cultivating the pattern of knowledge asymmetry that will facilitate successful knowledge integration. The results show that clients who outsource ISD need to maintain in-house technical knowledge that is deeper than the basic black-box level only when novel development processes are used. Thus, outsourcing novel projects does not entail outsourcing systems development skills; clients must maintain in-house technical knowledge. Similarly, vendors that routinely take on conceptually novel projects will enjoy significant advantages in the outsourcing marketplace by maintaining deep in-house business application domain knowledge.

The third implication is the conditions under which transferring valuable internal knowledge to a partner firm is valuable. Although clients that outsource ISD often fear sharing firm-specific, proprietary business knowledge with vendors, the results show that this might be necessary when conceptually novel projects are outsourced.[15]

Implications for ISD outsourcing research. The most intriguing implication of the results is that *ex ante* fit between client-vendor knowledge partitioning and project novelty is critical in outsourcing ISD. Knowledge integration—a central explanatory mechanism in this perspective—is recognized in prior ISD research, but how different forms of knowledge partitioning influence project outcomes via knowledge integration is not. In a notable departure from prior ISD outsourcing research, these results inform the choice of vendors at the outset of a project based on their knowledge-novelty fit. While the IS requirements elicitation literature implies that more business domain knowledge in the development organization is unconditionally beneficial (Shaft and Vessey 1995; Vessey and Conger 1993), the results show it to be so only in conceptually new projects. Its benefits are incommensurate with the potential costs in all other types of projects—a downside that prior ISD research has not explored.

The second insight for ISD is that vendors are more likely to use somewhat new processes in projects that are also conceptually new to them (suggested by the high and positive correlation of 0.353

between conceptual and process newness). Strict adherence to a known set of development processes is only possible for projects that are similar to ones that the vendor has done before. In contrast, fully understanding a conceptually novel target system usually requires cyclic interaction between detailed design and the usage environment itself. In other words, the process itself embodies some knowledge and is more readily reusable when the vendor sufficiently understands the application problem domain relatively well, but might require some degree of modification for conceptually novel projects. This implies that outsourcing of conceptually novel projects is feasible by clients with limited technical knowledge provided a development process familiar to the vendor is closely followed.

The third insight is that development coordination tools that are conceived to aid the ISD process (e.g., architectural modeling, test automation, configuration management, and change request tracking tools) do not compensate for knowledge partitioning misalignments. This opens up the question of whether the effectiveness of various tools is contingent on their ability to integrate knowledge that is sticky for a given type of project (discussed in §6.4).

6.3. Implications for Practice

a. Apriori choice of vendors. The results inform the ISD partnering decision at the project level and also provide insights into when organizations should consider *not* outsourcing ISD. Although it is difficult to mold a vendor to fit the knowledge partitioning pattern appropriate for a given project, it is relatively easier to recognize *a priori* the scope of knowledge that a vendor and client need to cover before choosing a vendor (see Figure 2). If a prospective vendor does not fit the corresponding pattern, the challenge of knowledge integration can overwhelm the potential benefits of outsourcing. An organization might be better off developing the software internally when the cost of creating non-existing knowledge overlaps with a vendor exceeds the marginal benefits of outsourcing.

<< INSERT FIGURE 2 HERE >>

b. Judging the value of investing in developing knowledge overlaps. Both clients and vendors incur costs in deepening their knowledge outside their immediate specialty beyond the black-box level. Software development organizations should invest in enhancing the business expertise of their developers only if

they routinely undertake conceptually novel projects. Firms that outsource ISD need to acquire technical knowledge only when the outsourced development work involves novel development processes; they do not necessarily need deep technical knowledge to outsource innovative, conceptually new projects.

6.4. Directions for Future Research

These ideas raise five research questions that may be fruitful avenues for future research. Does alienability of decision rights (Jensen and Meckling 1992)—whether one firm can viably delegate its own decision rights to the other—compensate for knowledge partitioning misfits? Are knowledge partitioning patterns in in-house ISD consistent with outsourced ISD, given that less formal intrafirm boundaries and a shared organizational context might ease knowledge integration? Can the theory developed here inform a project-specific choice of software development coordination tools?[16] Do the appropriate patterns of knowledge overlaps systematically vary based on whether the client is dependent on the vendor for capacity or expertise?[17] How does the knowledge partitioning perspective inform the organizing logic of the internal IT function? Organizing logic—centralized, federated, or decentralized (Sambamurthy 2000)—can be viewed as the allocation of different types of decision rights; its fit with IS-line function knowledge partitioning should predict how well the corporate IT function delivers business value.

7. CONCLUSIONS

Specialization is a competitive necessity, yet it is sometimes necessary for organizations to have knowledge beyond their domain of specialty *in order to* specialize effectively. The objective of this paper was to understand when client and vendor organizations must maintain knowledge of the other's domain in ISD outsourcing arrangements. The central idea developed was that project-specific client-vendor knowledge partitioning—novelty fit facilitates successful knowledge integration, which is critical to desirable project outcomes. Matched-pair data on 209 projects outsourced by US firms to vendors in Russia, Ireland, and India were used to test the proposed relationships. Building on Jensen-Meckling's work, I theoretically developed and empirically demonstrated how the pattern of knowledge overlaps that are most conducive to knowledge integration across the client-vendor boundary vary systematically with

the *type* of project novelty. While low overlaps suffice in routine projects, asymmetric overlaps are necessary in novel projects. The fine grained conceptualization of knowledge partitioning patterns and a theoretical explanation for its novelty-contingent value are noteworthy original contributions.

Overall, the results suggest that if organizations fail to systematically consider knowledge overlaps at the client-vendor boundary in ISD outsourcing decisions, they forego the opportunity to fully exploit available expertise. An appreciation of the contingencies described here can help both software development organizations and organizations that outsource ISD tread the fine line between freewheeling despecialization and myopic specialization, without compromising their ability to collaborate effectively.

Outsourced ISD Project Characteristic	Vendor's know ledge about customer's business application domain	Customer's technical know ledge about the project	Know ledge partitioning
a) Routine (no newness)	Low	Low	} Black-box
b) High conceptual newness	High	Low	} Asymmetric
c) High process newness	Low	High	
d) Simultaneous conceptual & process newness	High	High	} Bilateral overlap

Figure 1: Knowledge partitioning in outsourced information systems development.

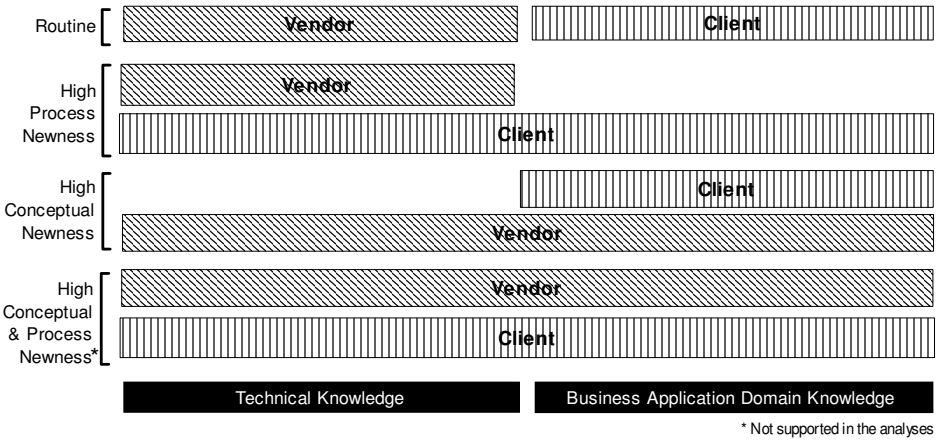

Figure 2: Knowledge Partitioning Patterns in Routine, Conceptually New, and Procedurally New Projects.

TABLE 1: A Jensen-Meckling analysis of knowledge partitioning in novel projects.

	Conceptual newness (B)	**Process newness (C)**
Sticky knowledge	Client needs	Vendor's interpretation of client needs
Source of sticky knowledge	Client	Vendor
Decision rights that draw on this sticky knowledge	Design	Validation
Locus of these decision rights	Vendor organization	Client organization
Solution to enhance knowledge integration at that locus	Higher business application domain knowledge in vendor organization	Higher technical knowledge in client organization

Table 2: Construct Correlations and Distributions

Variable	**Mean**	**St. Dev.**	**Construct Correlations**								
			1	2	3	4	5	6	7	8	9
1.Client's technical knowledge	1.40	1.15	—								
2.Vendor's app. domain knowledge	1.02	0.95	-.005	—							
3.Conceptual newness	2.87	0.94	-.023	-.128	—						
4.Process newness	1.89	1.05	.019	.005	.353**	—					
5. Client-vendor interaction	5.41	1.17	.094	.027	.111	.009	—				
6. Maturity of SD capabilities	2.60	1.07	.125	.005	-.113	-.016	-.025	—			
7. Architectural design effort	20.7	8.47	-.066	.051	.009	.020	-.104	.026	—		
8. Development coordination tools	4.02	1.35	.246**	.056	-.004	-.056	.246**	.306**	.237**	—	
9.Knowledge integration	4.03	0.76	-.093	-.019	.086	.075	.274**	-.013	.126	.159*	—

** p< 0.01 one-tailed test
* p< 0.05 one-tailed test

Table 3: Results

	Model I Control Variables Only			Model II Main Effects			Model III 2-way Interaction Effects			Model IV 3-way Interaction Effects		
	β	SE	T-value	β	SE	T-value	β	SE	T-value	β	SE	T-value
Constant (β_0)	.019	.376	8.838		.417	7.548		.392	8.236		.405	8.469
Russia (β_1)	-.032	.131	.169	-.006	.133	-.052	.063	.127	.565	.075	.126	.678
Ireland (β_2)	.049	.157	-.268	-.047	.160	-.391	-.051	.150	-.454	-.043	.149	-.379
Development coordination tools (β_3)	.301	.008	.400	.113	.009	.876	.055	.008	.435	.012	.009	.090
Client-vendor interaction (β_4)	**.067****	**.053**	**2.668**	**.302****	**.053**	**2.651**	**.284****	**.050**	**2.664**	**.222***	**.052**	**1.983**
Architectural design effort (β_5)	.098	.007	.596	.047	.007	.412	.025	.007	.233	.005	.007	.049
CMM level (β_6)	-.205	.064	.813	.091	.064	.753	**.209**	**.063**	**1.770**	**.218**	**.062**	**1.854**
Team size (β_7)	**-.106***	**.073**	**-1.736**	**-.214***	**.076**	**-1.752**	**-.274***	**.072**	**-2.378**	**-.244***	**.072**	**-2.102**
Prior collaboration history (β_8)	.019	.121	-.951	-.105	.125	-.912	-.115	.117	-1.069	-.104	.116	-.969
Vendor's business domain knowledge (β_9)				.035	.084	.310	.034	.079	.322	.046	.078	.438
Client's technical knowledge (β_{10})				-.188	.077	-1.642	-.178	.073	-1.637	-.162	.073	-1.484
Conceptual newness (β_{11})				-.015	.138	-.123	-.028	.129	-.242	.000	.130	.002
Process newness (β_{12})				.072	.133	.613	.090	.124	.819	.052	.128	.458
Vendor's business domain knowledge*Conceptual newness(β_{13})							**.271****	**.054**	**2.564**	**.343*****	**.057**	**3.040**
Client's technical knowledge*Process newness (β_{14})							**.263****	**.059**	**2.435**	.166	.101	.899
Conceptual newness*Process newness* Vendor's bus knowledge (β_{15})										-.230	.070	-1.76
Conceptual newness*Process newness*Client's tech knowledge(β_{16})										.150	.101	.770
Model R^2 (Model-F)	15.3% (1.71*)			18.8% (1.39)			31.2% (2.27*)			34.3% (2.21*)		
ΔR^2 (F-change)	—			3.6% (0.79)			12.4% (6.31***)			3.0% (1.57)		

N = 209 projects
** p< 0.01 one-tailed test
* p< 0.05 one-tailed test
+ p< 0.05 two-tailed test (for model IV only)

Endnotes

[1] Although many of the challenges described here might also arise in internal ISD projects, their effect in outsourced ISD can be especially pronounced because: (a) the vendor is less likely to understand the client's business as well an internal IT department and (b) the separation imposed by organizational boundaries can potentially hinder sharing of project-relevant knowledge. Outsourced ISD thus provides an appropriate initial context for exploring this question.

[2] Related empirical work has shown that knowledge integration enhances both ISD efficiency and effectiveness (Curtis et al. 1988; Faraj and Sproull 2000; Hoopes and Postrel 1999; Walz et al. 1993). This relationship is empirically verified in this study.

[3] Such stickiness is not a characteristic of the knowledge itself but a function of the preexisting knowledge of the holder and recipient of that knowledge. The same piece of knowledge can be stickier depending on the knowledge of the two organizations between which the exchange occurs. For example, software architects need to precisely understand requirements of potential users in the client firm to formulate an appropriate high-level systems design. This requires that the client organization be able to accurately represent its knowledge of the problem domain in a form that the vendor's software designers and programmers can utilize in the ISD process (e.g., formal requirements or project specifications). While knowledge that can be accurately represented in documents, stored in design artifacts, or embodied in development methodologies can be moved across the client-vendor boundary with relative ease, much of the necessary specialized expertise, skills, and perspectives of various project stakeholders can be sticky. Stickiness thus depends on the pre-existing knowledge of both the client and the vendor.

[4] There are two reasons for this. First, vendors face tradeoffs between specialization in the software development domain and acquiring some knowledge about the business domains of prospective clients (Becker and Murphy 1992). Vendors will always acquire knowledge about a compact set of common business problems—and only those problems—since they reduce the costs of communicating with a broad array of prospective clients without resulting in unfettered despecialization. This means that vendors will invest in acquiring only limited and generic business application domain knowledge. Second, when a vendor is faced with relatively routine ISD problems, investing in increasing business application domain knowledge that engenders an ability to solve exceptional and idiosyncratic ISD problems is suboptimal because acquiring such knowledge involves a fixed cost independent of its utilization (Garicano 2000). Clients incur similar costs in deepening their technical knowledge unless these costs can be spread across several projects. Rapid technological advances in software development techniques, languages, and processes reduce the likelihood that developing deep, in-house technical expertise will carry sustainable benefits for the client. Therefore, clients will invest in acquiring just enough specialized technical knowledge to be able to communicate with a variety of prospective vendors.

[5] For example, in using incremental software development methodologies, vendors use prototypes, mock-up screens, inspections, user-walkthroughs, and beta testing to gain corrective feedback from clients during development.

[6] For example, a vendor that is unfamiliar with integrated object-oriented CASE design tools or agile programming methodology might have more difficulty in conveying information that is generated through the use of such processes to the client organization in a form that the client can use to validate the vendor's interpretation of the client's application problem domain and to approve project deliverables and intermediate artifacts of the development process.

[7] As a corollary, this pattern of knowledge overlaps will also impede knowledge integration when the process is new to the *client* organization.

[8] Details of the items included in this survey are shown in the questionnaire. It is typically the responsibility of the liaison manager to coordinate the outsourced development with the vendor organization. To ensure that these responses came only from the client organization, I asked for an address to which I could mail a token of appreciation to the client liason.

[9] Three steps were taken to maximize response rates: (a) I offered a summary of the key findings of the study customized to each participating vendor firm, (b) I send a pre-notification letter to all firms in the Russian and Irish cluster, and to the Indian firms for which mailing addresses could be obtained. Then I simultaneously sent the survey via email and in hard copy format with international return envelopes. Three reminders were sent via fax, telephone, and email, and (c) I sent preliminary comparative results and a token gift to each responding vendor along with the request for a client-side evaluation.

[10] I made follow-up calls to nine randomly-selected non-responding vendors (3 each in Ireland, Russia, and India). Most non-participating organizations declined to participate for lack of time, concerns about sensitivity of the data, or because they no longer were in the custom development business. T-tests comparing the early (first 50) and late respondents (last 50) revealed no significant differences in cost project characteristics {such as cost overrun (T=1.54), knowledge integration (T=-1.37), client-vendor interaction (T=0.64), project complexity in person-months (T=-1.57), recorded defect density (T=0.75)} or vendor characteristics {such as CMM level (T=-1.46), vendor's business application domain knowledge (T=0.185), age of responding vendor (T=-0.95)}. The only significant difference was client assessments of ISD effectiveness (T=-2.36), suggesting a bias towards projects deemed more successful by clients.

[11] The average size of the team dedicated to the project by the vendor and client respectively was 16 people (s.d. 22.9) and 6.3 people (s.d. 11.4). Since the distinction in the literature is between smaller versus larger teams and given the large standard deviation, I dummy coded the top two-third percentile of the total team size as large teams.

[12] The simultaneous analysis of main effects and interaction terms tends to introduce multicollinearity, which residual centering addresses. Following Lance's (1988) 2-stage procedure, I first regressed each interaction term (e.g., client technical knowledge*process newness) on its component parts (i.e., client technical knowledge and process newness). I then used the resulting residual instead of the interaction term in testing the interaction effects model. This reduces multicollinearity, yielding a

regression coefficient for the cross-product term that can be directly interpreted as the effect of the interaction term on the dependent variable.

[13] I conducted mediation tests for knowledge integration following the procedures outlined by Baron and Kenny (1986). No direct effects were found from vendor business knowledge and client technical knowledge on project effectiveness and efficiency, suggesting full mediation by knowledge integration. This is consistent with Faraj and Sproull's (2000) prior finding that the utilization/integration rather than presence of expertise influences ISD outcomes.

[14] Differences in vendor business knowledge (F-value 0.475; n.s.), development efficiency (F-value 0.586; n.s.), conceptual newness (F-value 1.688; n.s.), process newness (F-value 1.612; n.s.), CMM level (F-value 0.388; n.s.), architectural design effort (F-value 0.358; n.s.) were non-significant.

[15] If the application domain is novel to the vendor and the client incurs the upfront costs of transferring such knowledge to a vendor, it is desirable to keep an ongoing relationship with the same vendor to avoid repeatedly educating different vendors about the same application domain. Alternatively, if the application domain is generic to their industry, they might pre-screen vendors for such knowledge before outsourcing.

[16] The effectiveness of each tool might be contingent on: (1) whether the type of knowledge it helps integrate is sticky in that type of project and (2) whether it can instead help move specific decision rights to the source of sticky knowledge.

[17] Even firms with sufficient technical expertise outsource ISD to extend internal capacity (reduce costs, save time, or preserve scarce internal resources). However, as vendors develop distinctive technical capabilities and industry expertise sometimes superior to the client's own expertise, such arrangements might be motivated primarily by the client's desire to gain access to such expertise.

REFERENCES

Adelson, B., and Soloway, E. 1985. The Role of Domain Experience in Software Design. *IEEE Transactions on Software Engineering*. 11(11) 1351-1360.

Adler, P.S. 1995. Interdepartmental Interdependence and Coordination - the Case of the Design/Manufacturing Interface. *Organization Science*. 6(2) 147-167.

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

Becker, G., and Murphy, K. 1992. The Division of Labor, Coordination, and Knowledge. *Quarterly Journal of Economics*. CVII(4)

Carlile, P., and Rebentisch, E. 2003. Into the Black Box: The Knowledge Transformation Cycle. *Management Science*. 49(9) 1180-1195.

Chatzoglou, P.D., and Macaulay, L.A. 1996. Requirements Capture and IS Methodologies. *Information Systems Journal*. 6(3) 209-225.

Curtis, B., Krasner, H., and Iscoe, N. 1988. A Field Study of the Software Design Process for Large Systems. *Communications of the ACM*. 31(11) 1268-1287.

Fama, E., and Jensen, M. 1983. Separation of Agency and Control. *The Journal of law & economics*. 26(June) 301-326.

Faraj, S., and Sproull, L. 2000. Coordinating Expertise in Software Development Teams. *Management Science*. 46(12) 1554-1568.

Garicano, L. 2000. Hierarchies and the Organization of Knowledge in Production. *Journal of Political Economy*. 108(5) 874-904.

Gopal, A., Sivaramakrishnan, K., Krishnan, M., and Mukhopadhyay, T. 2003. Contract in Offshore Software Development: An Empirical Analysis. *Management Science*. 49(12) 1671-1683.

Gottesdiener, E. 2003. Requirements by Collaboration. *IEEE Software*. March-April) 52-55.

Grant, R. 1996. Prospering in Dynamically-Competitive Environments: Organizational Capability as Knowledge Integration. *Organization Science*. 7(4) 375-387.

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

Ho, V., Ang, S., and Straub, D. 2003. When Subordinates Become IT Contractors: Persistent Managerial Expectations in IT Outsourcing. *Information Systems Research*. 14(1) 66-86.

Hoopes, D., and Postrel, S. 1999. Shared Knowledge, "Glitches," and Product Development Performance. *Strategic Management Journal*. 20 837-865.
Jaccard, J., and Turrisi, R. 2003. *Interaction Effects in Multiple Regression*. Sage, Thousand Oaks, CA.
Jensen, M., and Meckling, W. 1992. *Specific and General Knowledge and Organizational Structure*. in: Contract Economics, (ed.), Blackwell, Oxford, England, 251-274.
Kirsch, L., Sambamurthy, V., Ko, D., and Purvis, R. 2002. Controlling Information Systems Development Projects: The View from the Client. *Management Science*. 48(4) 484-498.
Lance, C. 1988. Residual Centering, Exploratory and Confirmatory Moderator Analysis, and Decomposition of Effects in Path Models Containing Interactions. *Applied Psychological Measurement*. 12(2) 163-175.
Lee, J., and Kim, Y. 1999. Effect of Partnership Quality on IS Outsourcing Success. *Journal of Management Information Systems*. 15(4) 29-61.
Martins, L., and Kambil, A. 1999. Looking Back and Thinking Ahead: Effects of Prior Success on Managers' Interpretations of New Information Technologies. *Academy of Management Journal*. 42(6) 652-661.
Mookerjee, V.S., and Chiang, R. 2002. A Dynamic Coordination Policy for Software System Construction. *IEEE Transactions on Software Engineering*. 28(6) 684-694.
Okhuysen, G., and Eisenhardt, K. 2002. Integrating Knowledge in Groups: How Formal Interventions Enable Flexibility. *Organization Science*. 13(4) 370-386.
Orlikowski, W. 2002. Knowing in Practice: Enacting a Collective Capability in Distributed Organizing. *Organization Science*. 13(2) 249-273.
Ramesh, B. 2002. Process Knowledge Management with Traceability. *IEEE Software*. May/June) 50-55.
Robillard, P. 1999. The Role of Knowledge in Software Development. *Communications of the ACM*. 42(1) 87-92.
Rowen, R. 1990. Software Project Management under Incomplete and Ambiguous Specifications. *IEEE Transactions on Engineering Management*. 37(1) 10-21.
Rus, I., and Lindvall, M. 2002. Knowledge Management in Software Engineering. *IEEE Software*. 19(3) 26-38.
Sambamurthy, V. 2000. Research Commentary: The Organizing Logic for an Enterprise's IT Activities in the Digital Era--a Prognosis of Practice and a Call for Research. *Information Systems Research*. 11(2) 105-114.
Shaft, T.M., and Vessey, I. 1995. The Relevance of Application Domain Knowledge - the Case of Computer-Program Comprehension. *Information Systems Research*. 6(3) 286-299.
Szulanski, G. 1996. Exploring Internal Stickiness: Impediments to the Transfer of Best Practice within the Firm. *Strategic Management Journal*. 17(2) 27-43.
Takeishi, A. 2002. Knowledge Partitioning in the Interfirm Division of Labor: The Case of Automotive Product Development. *Organization Science*. 13(3) 321-338.
Vessey, I., and Conger, S. 1993. Learning to Specify Information Requirements: The Relationship between Application and Methodology. *Journal of Management Information Systems*. 10(2) 177-201.
von Hippel, E. 1994. Sticky Information and the Locus of Problem Solving: Implications for Innovation. *Management Science*. 40(4) 429-439.
Walz, D., Elam, J., and Curtis, B. 1993. Inside a Software Design Team: Knowledge, Sharing, and Integration. *Communications of the ACM*. 36(10) 63-77.
Wang, E., Barron, T., and Seidmann, A. 1997. Contracting Structures for Custom Software Development: The Impacts of Informational Rents and Uncertainty on Internal Development and Outsourcing. *Management Science*. 43(12) 1726-1744.