Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2007 Proceedings

Americas Conference on Information Systems (AMCIS)

December 2007

IT Value of Software Development: A Multi-theoretic Perspective

Venugopal Balijepally
Prairie View A&M University

Sridhar Nerur
University of Texas at Arlington

Radha Mahapatra

IT VALUE OF SOFTWARE DEVELOPMENT: A MULTI-THEORETIC PERSPECTIVE

VenuGopal Balijepally
Prairie View A&M University
vebalijepally@pvamu.edu

Sridhar Nerur
University of Texas at Arlington
snerur@uta.edu

RadhaKanta Mahapatra
University of Texas at Arlington
mahapatra@uta.edu

Abstract

Software development in organizations is evolving and increasingly taking a socio-technical hue. While empirical research guided by common sense reasoning has informed researchers and the software community in the past, the increasing social character of software development provides us with the context and the motivation to provide theoretical underpinnings to our empirical work. In this paper we sample three theoretical domains that could serve our empirical research efforts: social capital, organizational learning and knowledge based view of the firm. We illustrate the utility of these theoretical perspectives by articulating a research model that captures the IT value created by software development teams practicing different methodologies.

Keywords: software development, social capital, agile methodologies, organizational learning, knowledge management

Introduction

The field of software development has undergone significant changes in recent years. A hypercompetitive business environment characterized by change and uncertainty has prompted the software community to evolve new ways of building software. The emerging methodologies follow an evolutionary delivery model (Gilb 1987) that allows developers to adapt to changing requirements. This is counter to the traditional plan-driven approach that uses a linear process in accomplishing the tasks involved in software development. In such an approach, an enormous amount of time and effort are spent in gathering requirements and evolving specifications with a view to preparing for all foreseeable changes during the lifecycle of the product. In contrast, emerging approaches such as agile methodologies rely on short iterative cycles with continuous stakeholder feedback and frequent planning to cope with and leverage change (Nerur et al. 2005).

Agile methodologies have attracted a lot of attention in the recent past. This new approach differs considerably from traditional software development in many ways. Foremost, there is an increased emphasis on self-organizing teams that enjoy greater latitude in goal-setting and decision-making with regard to setting priorities, deadlines, etc. Team members are encouraged to play multiple roles, such as developer, architect and analyst. A shared understanding and vision of the evolving product is facilitated by practices such as joint code ownership, role rotation and reflection workshops. This is in contrast to specialized roles assigned to developers in traditional methodologies. In the agile approach, specifications evolve

through constant dialogue and feedback between developers and customers, while in the traditional approach, extensive specifications gathered upfront from customers guide the development process. Thus, there is a perceptible shift from a hierarchical, process driven, and command-and-control based approach to one that emphasizes people-orientation, collaboration, and leadership (Highsmith 2003).

The changes in methodologies highlight the underlying transformation of software development from a technical enterprise to a more socio-technical endeavor. The lack of theory-driven empirical work in software development may be attributed to the emphasis that was placed on technical aspects of software development. Theoretical grounding of empirical research is still not considered an essential requirement, especially in the software engineering domain. Though there is some evidence of increasing awareness of theoretical issues (Hannay et al. 2007), the dominant thinking is predicated on the primacy of common sense reasoning over generalizable theory (Lindblom 1987). While use of theory is taken for granted in various business disciplines, including several areas of IS research, software development research is still grappling with the issue of whether theory should be used (Hannay et al. 2007). The current social "makeover" of software development provides us with the context and the opportunity to refocus empirical research in software development towards theory building and testing. The centrality and importance accorded to teams and collaboration among team members by emerging methodologies affords an opportunity to draw on the extensive body of knowledge in organizational and management theory.

The primary objective of this paper is to demonstrate the applicability of theoretical perspectives in software development research. One of the critical problems that confront software managers today is the choice of methodologies. This poses an interesting research problem that can be theoretically investigated. Keeping this in mind, we articulate three theoretical streams, namely social capital, knowledge-based view of the firm, and organizational learning. Traditionally, human capital is considered as the main resource of a software team that produces valued outcomes. While the resource value of relationships of members from within and across teams is understood by the software community at an intuitive level, the current trend towards increasing socialization of software development calls for capturing such resource value of social relationships in more explicit ways. We bring in social capital as a new explanatory factor to capture such social dynamics of software development in general and agile development in particular. Further, we propose a research model, informed by these theories, that explores the IT value created by software development teams. While many researchers have focused on the differences between agile and plan-driven approaches, or on the differences between pairs and individuals in software development, there has hardly been any effort to understand how these approaches create value by emphasizing different knowledge outcomes. Such an understanding should help software practice in making more informed methodological choices. The proposed model is a small step in this direction.

The next section describes the three theoretical perspectives that can potentially inform research on software development challenges. This is followed by an articulation of the research model and the propositions. Finally, conclusions are drawn.

Theoretical Perspectives

Three theoretical perspectives of potential interest to software development research are briefly described below.

Social Capital

Social capital is an interesting theoretical lens attracting increasing research attention in sociology and management literatures. Social capital is defined as "the sum of the actual and potential resources embedded within, available through, and derived from the network of relationships possessed by an individual or social unit" (Nahapiet and Ghoshal 1998). Traditionally human capital embedded in the team in terms of skills and abilities of members is considered a critical success factor for software development teams. Social capital theory seeks to position network of relationships as another form of capital that provides significant benefits to the individual or the collectivity in the conduct of social affairs. Unlike human capital, which rests in the individuals, social capital is embedded in the networks of mutual acquaintances and relationships of the individual, group or the organization (Bourdieu 1986; Coleman 1988; Putnam 1993). Social capital lens could be very useful to explain several IS phenomena such as software development, IS outsourcing, organizational knowledge management, and IT-based inter-organizational networks (Balijepally et al. 2004).

Organizational Learning

Organizational learning is a mature theoretical stream of enduring value used in several research domains. According to this view, learning is the process of acquiring, interpreting or distributing information, that changes the range of potential

behaviors for the entity (Huber 1991). For Argyris, learning is about detecting and correcting mismatches or errors. Learning occurs when a match is produced for the first time between intentions and actuality, that is genuinely new to the actors producing it (Argyris 1996). In organizational learning, the entity involved is the organization as against the individual. Organization learning is "the process of improving actions through better knowledge and understanding." It leads to cognitive systems, associations and memories, developed and shared by organizational members (Fiol and Lyles 1985). Learning in organizations may be conceptualized as adaptive learning and generative learning. Adaptive learning occurs when an organization seeks to accomplish performance requirements through active shaping of actions and responses based on feedback from previous actions. When the organization starts questioning its values and assumptions and seeks "new ways of looking at the world" in its actions and behaviors, the resulting learning is considered as generative learning. While adaptive learning underscores coping behaviors, generative learning is about creating (Senge 1990). This has parallels to Argyris's single loop and double loop learning (Argyris 1977; Argyris 1996) and Fiol and Lyles's low level and high level learning (Fiol and Lyles 1985).

Knowledge Based View of the Firm

Knowledge based view is based on the notion that knowledge is a valuable organizational resource, and the firm is a dynamic, evolving, quasi-autonomous system of knowledge and application (Spender 1996). Based on Polanyi, a distinction is made between two forms of organizational knowledge – explicit and tacit (Polanyi 1966). Explicit knowledge is the codified knowledge that could be transmitted through formal systematic language. Tacit knowledge on the other hand is "deeply rooted in action, commitment, and involvement in a specific context" (Nonaka 1994). It is therefore intangible and cannot easily be articulated. The tacit and explicit knowledge types could be at both individual and group levels, the aggregate of which constitutes the intellectual capital for the collectivity or the organization (Nahapiet and Ghoshal 1998). Knowledge based view conceives organizations as innovating and knowledge creating entities where exchange, combination and dynamic interaction of explicit and tacit knowledge at both individual and aggregate level creates new knowledge. Innovation is typically viewed as "a process in which the organization creates and defines problems and then actively develops new knowledge to solve them" (Nonaka 1994).

Theoretical Model and Research Propositions

The focus of the theoretical model developed here is on the software development project team undertaking a software project using traditional plan-driven methodology or one of the agile methodologies. In case of large projects, this could be a project team working on an identifiable project subcomponent. We adopt social capital theory along with knowledge management and organizational learning perspectives to examine the IT value created by the software team. Figure 1 showcases the research model. We view the network of relationships among members of a software development team as a form of social capital for the team. This network of relationships could be from within the team or across project teams within the organization or even beyond the organization. This is in addition to the human capital available to the focal team in terms of skills and abilities of its members and captures the resources potentially available to the team and its members through the network of relationships.

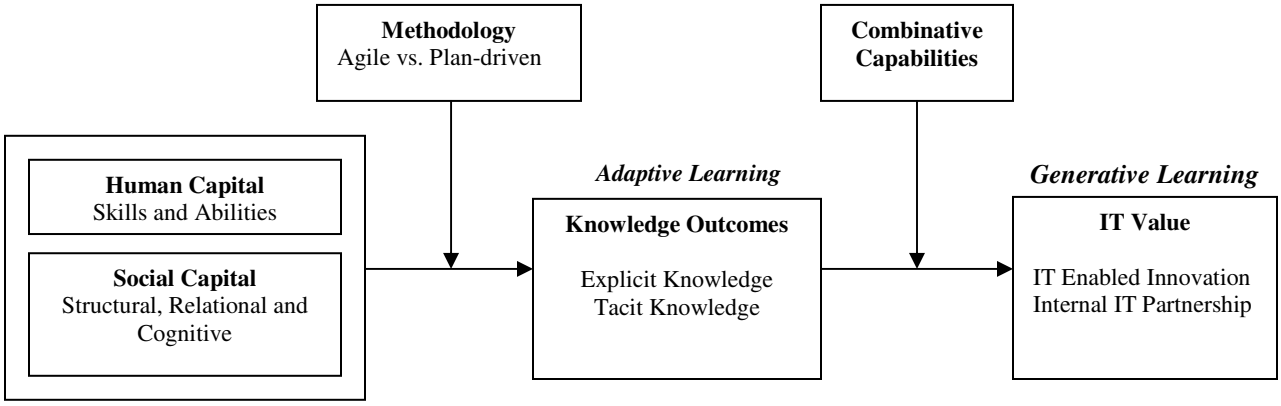

Figure 1– Value Creation in Agile and Plan-driven Software Development Teams

We argue that software development is a knowledge creating activity, the knowledge outcomes of which are both tacit and explicit. IT artifacts such as plans and designs, test cases, data models, data flow diagrams, source code and documentation exemplify explicit knowledge created by the team (Boehm and Turner 2004). The tacit knowledge generated could be in the form of mental schemas of the system and know-how regarding the technologies used and the business context. This tacit knowledge created could be both at the individual (objectified) and group (collective) level. It is the combination of these knowledge types that exemplifies the knowledge outcomes of software development effort.

We consider the two-step process of organizational learning as underpinning the value derived from software development activity in project teams. In a recent article Ye and Agarwal adopted a similar two stage learning process to articulate the value derived from strategic outsourcing partnerships (Ye and Agarwal 2003). The adaptive learning for the software development team involves intuition, interpretation, and integration (Crossan et al. 1999) of team knowledge resources derived from human and social capital in creating the design artifacts, the software and the new knowledge that contributes to the knowledge stock of the team. The generative learning occurs when new ways of looking at things and recombining existing knowledge stock and capabilities take effect to create higher valued outcomes for the team and the organization. This is consistent with capabilities integration articulated by Grant (Grant 1996b) and involves concepts such as information interpretation and distribution and organizational memory (Huber 1991; Ye and Agarwal 2003). The next few sections showcase propositions related to the model.

Knowledge Creation in Software Teams

Software development is a dynamic knowledge creation endeavor wherein the business and technical knowledge of team members in both tacit and explicit forms is exchanged and combined in complex ways to generate new knowledge some of which remains tacit within the team while the explicit form of the knowledge manifests in the software product and the documentation produced by the team. All knowledge creation occurs through the two generic processes of combination and exchange (Moran and Ghoshal 1996). In a software development team, the knowledge resources of the individuals are combined to create the various IT artifacts and the software. There is exchange of information and knowledge between various members of the team such as between users and systems analysts, systems analysts and developers, developers and testers. The increased emphasis placed by agile methodologies on collaboration and self-organization among team members as well as on role rotation is likely to enhance the quality and intensity of knowledge exchanged.

Combination is the process of reconfiguring, sorting, adding, and conceptualizing existing explicit knowledge to create new explicit knowledge. It involves reconfiguring of existing information (explicit knowledge) through, adding, sorting, re-contextualizing and reappraising to provide new insights (Nonaka 1994). In a software development team, explicit knowledge of analysts, developers and users is combined to produce new explicit knowledge through articulation to one self and to the team and applying to the problem context to create new explicit IT artifacts. Internalization is the process of converting explicit knowledge into tacit knowledge, which is akin to the conventional notion of 'learning.' Action or 'learning by doing' helps in the internalization process. Through trial and error, ideas and concepts are articulated and reconfigured till a more concrete form emerges (Nonaka 1994). Internalization occurs at the individual level when explicit knowledge available such as user requirements and technical documentation and the resultant interactions during actual software development are absorbed as know-how.

Existing tacit knowledge of individuals and the team is converted into new explicit and tacit knowledge through the processes of externalization and socialization respectively Experience is considered critical to the transfer of tacit knowledge. The externalization mode triggers through a series of interactions and dialogue between software team members. Metaphors typically play an important role in articulating otherwise difficult to articulate individuals' perspectives and technical know-how. Being situated in practice, tacit knowledge of the software development team members is externalized while working on the problem at hand. Experienced programmers and systems analysts could externalize, through demonstration to other members of the team, tacit knowledge inherent in activities such as abstracting object designs, creating and using design patterns, identifying classes from the generic and organizational class libraries, using appropriate control structures and debugging techniques. Socialization is the process through which existing tacit knowledge of individuals is converted to new tacit knowledge. Socialization occurs through interaction between developers as in apprenticeship. Knowledge transfer by socialization could occur even without the help of language, through observation, imitation and practice. The shared experience during software development provides the context for people to transfer their thinking processes (Nonaka 1994). The amount of knowledge creation through socialization in software development teams is however dependent upon the methodology used.

Human Capital and Team Knowledge Outcomes

The human capital of the software development team, reflected in the skills and abilities of software developers, is an important factor determining the effectiveness of all aspects of software development irrespective of the methodology used. Skills and abilities reflect the existing knowledge resources inherent in the members of the team. These skills include programming ability and technical domain knowledge. For systems analysts, generalist skills (Benbasat et al. 1980; Green 1989) and technical skills that complement business skills (Todd et al. 1995) become important. For customers or users who may be part of systems development team either full time or part time, the ability to understand and articulate the business requirements becomes important. Unsuitable customer representation is a known project risk factor (Boehm and Turner 2004). High ability developers, analysts and testers with higher levels of tacit skills are better equipped to work on the various explicit knowledge artifacts such as requirement specs, architecture plans and existing code artifacts in producing higher explicit and tacit outcomes by pooling their knowledge and skills through the four generic processes of knowledge conversion, that is combination, internalization, externalization and socialization.

We therefore argue that higher skills and abilities of software developers (human capital) in software development team contribute to higher knowledge outcomes, both tacit and explicit.

> *Proposition 1: The human capital embedded in a software development team is positively related to both explicit and tacit knowledge outcomes of the team*

Social Capital and Team Knowledge Creation

Social capital is a multidimensional construct with three analytically distinct but related facets: structural, relational and cognitive. Structural dimension of social capital refers to the overall pattern of connections between actors. In a software development team the networks of relationships could be both internal and external to the team. While ties between the members of the team capture the internal structure, the nature of relationships with other project teams within the organization or even external to the organization constitute external network structures. Strength/weakness of ties (both internal and external) and the resource content of these networks (e.g., network heterogeneity) are some structural aspects of social capital. (Nahapiet and Ghoshal 1998). The relational dimension of social capital refers to personal relationships such as respect or friendship individuals develop among themselves through a history of interactions that help fulfill different social motives such as approval, sociability, and prestige. Trust and identification capture the relational aspects of social capital. The cognitive dimension refers to the resources that provide shared representations, interpretations, and systems of meanings among parties (Nahapiet and Ghoshal 1998). Shared vision and transactive memory systems capture this dimension of social capital. Although individual dimensions of social capital are of immense research interest, in the interest of brevity, social capital is treated as an aggregated construct in deriving propositions in the rest of the paper.

As conceptualized in Figure 1, we argue that the social and human capital embedded in a software development team yield knowledge outcomes and IT value through a combination of adaptive and generative learning. Social capital, conceptualized as the actual and potential resources available through the network of relationships of the focal actor, signifies the resource value of relationships. In software development project teams, we view the network of relationships of members of a software development team as a form of social capital for the team. This network of relationships enable access to potential resources that supplement the human capital available in the team and contribute to valued team outcomes by fostering adaptive and allocative efficiencies (Nahapiet and Ghoshal 1998). Allocative efficiency results from improvement in information flow in the social network of the team members through reduction in structural redundancies in the network (Burt 1997).

During software development, analysts and developers frequently look for advice and tips while working with new technologies. The non-redundant information available through their network contacts within and outside the team is a critical input to the software task at hand. Such information may provide opportunities to improvise during the software development process. Social capital facilitates such information flow by reducing the transaction costs for information retrieval (Putnam 1993). Adaptive efficiencies result from the cooperative behavior encouraged by social capital that results in creativity and learning (Nahapiet and Ghoshal 1998). The new information and knowledge available and accessed from the network of relationships of team members result in superior software products. Thus, social capital potentially facilitates creation of higher levels of both tacit and explicit knowledge.

> *Proposition 2 - The social capital embedded in a software development team is positively related to both explicit and tacit knowledge outcomes of the team*

Interaction of Methodology on Human Capital and Social Capital

Software development teams using plan-driven methodologies perform elaborate requirements gathering to develop complete, consistent, traceable and testable specifications. Big upfront planning is used to control change. User involvement is mainly in the initial phases of development. While plans are evolved to guide development and control change, communication among developers is primarily through extensive documentation. Thus the traditional plan driven methodologies are predisposed to producing explicit knowledge artifacts such as designs, plans and documentation that serve as the means of communication. Creation of these artifacts results primarily from combination and externalization processes. The traceability and verifiability imperatives of plan-driven development entail externalization of tacit knowledge of the developers into explicit forms. Correspondingly, there is less emphasis on tacit knowledge with fewer opportunities for facilitating processes such as internalization and socialization that are essential for creation of new tacit knowledge.

Agile methodologies follow a minimalist approach to requirement gathering and documentation. Only some architectural designs and project vision documents are prepared in advance that guide developers. Participating users articulate the requirements in the form of user stories and prioritize them. The competence and knowledge of team members are used to cope with emergent problems (Highsmith 2003). This is consistent with the idea that "communities of interaction contribute to the amplification and development of new knowledge" (Nonaka 1994). With the primacy accorded to delivery of working code at frequent intervals, shared vision and shared understanding of project goals and objectives is emphasized to incorporate changes as they occur (Boehm and Turner 2004). With collocated team members and a readily accessible user representative, the need for externalization processes to convert tacit knowledge into more explicit forms is reduced. Instead a climate for generating higher levels of tacit knowledge is fostered through processes such as socialization and internalization. There is thus substantially higher level of tacit knowledge generated in agile teams as compared to plan-driven software development teams. These arguments yield the following propositions.

> *Proposition 3a: The human capital and social capital embedded in a software development team produce higher levels of explicit knowledge when using plan-driven methodologies than when using agile methodologies*
>
> *Proposition 3b: The human capital and social capital embedded in a software development team produce higher levels of tacit knowledge when using agile methodologies than when using plan-driven methodologies*

IT Value Creation through Generative Learning

The knowledge outcomes in the software development team underpin the organizational IT value created. The knowledge when it undergoes a process of recombination and integration with the existing knowledge and capabilities, leads to valued outcomes for the team and the organization (Grant 1996a; Kogut and Zander 1992). The IT value created may be conceptualized in terms of generative learning. Generative learning typically occurs when, based on exigencies, people start questioning the preset conditions and taken for granted assumptions. It involves changes in the underlying governing values or master programs that lead to changes in action (Argyris 1977; Argyris 1996; Senge 1990). Such learning outcomes enhance the human and social capital of the team and strengthen organizational capabilities for the future. We conceptualize that the generative learning for the team and the organization is reflected in the strengthened internal IT partnership and IT enabled innovation (Ye and Agarwal 2003). We argue that knowledge outcomes interact with the combinative capabilities of the team in creating IT value through generative learning.

Team Knowledge Outcomes and IT Enabled Innovation

IT enabled innovation is the deployment of IT for the creation of new products and services, new forms of organization, or for exploring new market opportunities (Agarwal and Sambamurthy 2002). Innovation, IT enabled or otherwise, is an important organizational capability that contributes to competitive advantage enjoyed by a firm. As any organizational innovation provides competitive advantage for a finite period before competitors catch up, continuous innovation through cannibalization of existing products and technologies is the mantra for organizations to survive and thrive in a competitive marketplace. Contribution of knowledge to creating and sustaining organizational innovation is well documented in knowledge management literature (Grant 1996a; Grant 1996b).

The knowledge outcomes of a software development team could spur IT-enabled innovation in complex ways. The IT artifact which personifies explicit knowledge may be used in novel ways not originally conceived by the software team. The

knowledge gained during a systems development effort could help create new software products or services by the focal unit and the IT. For example, a software team based on its knowledge gained from a system development/implementation effort could provide consulting services on related matters to other teams within or even outside the organization. In the presence of generative learning, when a software team is able to come up with new ways of applying the knowledge gained to current or new problems, such efforts are expected to result in greater IT-enabled innovation. Thus,

> *Proposition 4: The knowledge outcomes of software development team are positively related to its IT enabled innovation*

Team Knowledge Outcomes and Internal IT Partnership

Having an effective relationship between IT and business in organizations is considered a key requirement for deriving IT value (Reich and Benbasat 2000). Appreciating the complementarity of mutual contributions to the organizational value is a key to nurturing such partnerships. Internal partnership between line managers and IT managers and specialists helps an organization in deriving higher strategic benefits from IT resources and investments. Such partnerships foster harmonious relationships between IT and business which enhances mutual trust and confidence and even improves the overall standing of IT within the organization. Having internal IT-business partnership constitutes IT value as the success of any IT initiative is critically dependent upon the convergence of business and technology (Nambisan et al. 1999).

Knowledge outcomes of software development contribute to fostering IT-business partnerships for the focal team in important ways. Higher levels of tacit and explicit knowledge outcomes enhance the visibility of the team and its contributions to the IT unit and the business. The explicit knowledge artifacts such as documentation help ease communication gaps between the team, IT unit and the business. The business customers carrying rich tacit knowledge from their involvement with the software development team become ambassadors of IT in nurturing and cementing the IT-business partnership. The boundary spanning activities of these business customers not only provides valuable resources to the focal unit and IT during the systems development effort, but also opens up innovative collaboration possibilities. The communication and interaction developed with business during the systems development effort also increases the responsiveness of focal unit and the IT to future business requirements. Thus,

> *Proposition 5: The knowledge outcomes of a software development team is positively related to its internal IT business partnership*

Interaction of Team Knowledge Outcomes and Combinative Capabilities

Argyris and Schon originally enunciated that a performance gap is a necessary condition for first and second order organizational learning (Argyris and Schon 1978). Other researchers identified two additional necessary conditions (Duncan and Weiss 1979; Huber 1991). First, the organizational members should have the motivation, ability, and opportunity to resolve the perceived performance gap for the organization. Second the first and second order learning by individual members must be externalized or translated from the individuals' tacit knowledge into an appropriate useful form (Arthur and Aiman-Smith 2001). Combinative capability of software development team is a factor that could potentially moderate the relationship between knowledge outcomes and IT value created by the team.

Combinative capability is the intersection of an entity's capability to exploit its current knowledge to create new opportunities (Kogut and Zander 1992). Innovation and new learning are results of an entity's combinative capabilities to create new knowledge and applications from its existing pool of knowledge. Kogut and Zander provide illustrations of such abilities at the individual level and demonstrate its utility at the organizational level (Kogut and Zander 1992). We conceptualize combinative capabilities in our model at the team level to show the generative learning outcomes of its interaction with knowledge outcomes of software team. The combinative capabilities comprise system capabilities, coordination capabilities, and socialization capabilities (Van Den Bosch et al. 1999). System capabilities include the various explicitly laid down policies, procedures, manuals that provide direction in routine situations and help in the integration of explicit knowledge. They help in reducing the need for communication and coordination while tackling every day problem situations. Creation of higher levels of new knowledge and innovation is determined by the combinative abilities of the focal unit and existing knowledge stock. In software development the combinative capability of the focal team include system capabilities to codify knowledge and experience to provide future direction, coordination capabilities to manage the project and socialization capabilities to interact with the stakeholders within and outside the focal unit. In software development teams high levels of this ability could determine whether the knowledge outcomes created by the team result in higher IT value for the team and by extension to the organization. This leads us to the last proposition:

Proposition 6: The strength of the positive relationship between knowledge outcomes of a software development team and IT value is positively moderated by the focal team's combinative capabilities[1]

Conclusion

Theoretical foundations of empirical research in software development are not very robust. Given the changing nature of the field, there is ample opportunity to draw on the extensive corpus of knowledge in organizational and management theory to address myriad issues. This research examines the applicability of three very popular theoretical expositions to understand the benefits of social capital that accrue to software development teams using agile versus traditional plan-driven methodologies. We argue that software development is a knowledge creating activity where human capital and social capital embedded in a software team work to create knowledge outcomes, both tacit and explicit. The relative extent of knowledge outcomes is contingent upon the methodology used, with agile teams producing higher tacit knowledge and traditional plan-driven teams creating higher explicit knowledge outcomes. Drawing from the organizational learning literature, we argue that the adaptive learning of the software team results in knowledge outcomes, while generative learning leads to IT value for the team and the organization. We have articulated a research model anchored in some well-known theories. In addition to developing a robust model to investigate software development using different methodologies, we have demonstrated the value of applying widely accepted theories to software development.

References

Agarwal, R., and Sambamurthy, V. "Principles and Models for Organizing IT Function," *MIS Quarterly Executive* (1:1), 2002, pp. 1-16.

Argyris, C. "Double Loop Learning in Organizations," *Harvard Business Review* (55:5), 1977, pp. 115-125.

Argyris, C. "Unrecognized Defenses of Scholars: Impact on Theory and Research," *Organization Science* (7:1), 1996, pp. 79-87.

Argyris, C., and Schon, D.A. *Organizational Learning: A Theory of Action Perspective*, Addison-Wesley, Reading, MA, 1978.

Arthur, J.B., and Aiman-Smith, L. "Gainsharing and Organizational Learning: An Analysis of Employee Suggestions Over Time," *Academy of Management Journal* (44:4), 2001, pp. 737-754.

Balijepally, V., Mahapatra, R., and Nerur, S. "Social Capital: A Theoretical Lens for IS Research," Americas Conference for Information Systems, New York, NY, 2004, pp. 1585-1592.

Benbasat, I., Dexter, A.S., and Mantha, R.W. "Impact of Organizational Maturity on Information System Skill Needs," *MIS Quarterly* (4:1), 1980, pp. 21-34.

Boehm, B., and Turner, R. *Balancing Agility and Discipline: A Guide to the Perplexed*, Addison-Wesley, Boston, MA, 2004.

Bourdieu, P. "The Forms of Capital," in: *Handbook of Theory and Research for the Sociology of Education,* J.G. Richardson (ed.), Greenwood, New York, 1986, pp. 241-258.

Burt, R.S. "The Contingent Value of Social Capital," *Administrative Science Quarterly* (42:2), 1997, pp. 339-365.

Coleman, J.S. "Social Capital in the Creation of Human Capital," *American Journal of Sociology* (94), 1988, pp. S95-S120.

Crossan, M.M., Lane, H.W., and White, R.E. "An Organization Learning Framework: From Intuition to Institution," *Academy of Management Review* (24:3), 1999, pp. 522-537.

Duncan, R.B., and Weiss, A. "Organizational Learning: Implications for Organizational Design," in: *Research in Organizational Behavior,* B.M. Staw (ed.), JAI Press, Greenwich, CT, 1979, pp. 75-123.

Fiol, C.M., and Lyles, M., A. "Organizational Learning," *Academy of Management Review* (10:4), 1985, pp. 803-813.

Gilb, T. *Principles of Software Engineering Management*, Addison-Wesley, Reading, MA, 1987.

Grant, R.M. "Prospering in Dynamically-competitive Environments: Organizational Capability as Knowledge Integration.," *Organization Science* (7:4), 1996a, p. 375.

Grant, R.M. "Toward a Knowledge-Based Theory of the Firm," *Strategic Management Journal* (17), 1996b, pp. 109-122.

Green, G.I. "Perceived Importance of Systems Analysts' Job Skills, Roles, and Non-Salary Incentives," *MIS Quarterly* (13:13), 1989, p. 2.

Hannay, J.E., Sjoberg, D.I.K., and Dyba, T. "A Systematic Review of Theory Use in Software Engineering Experiments," *IEEE Transactions on Software Engineering* (33:2), 2007, pp. 87-107.

[1] The propositions for moderating effect of combinative capabilities on the effect of knowledge outcomes on individual dimensions of IT value have not been shown for the sake of brevity.

Highsmith, J. "Agile Project Management: Principles and Tools," *Agile Project Management Advisory Service* (4:2), 2003, p. 37.

Huber, G.P. "Organizational Learning: The Contributing Processes and the Literatures," *Organization Science* (2:1), 1991, pp. 88-115.

Kogut, B., and Zander, U. "Knowledge of the Firm, Combinative Capabilities, and the Replication of Technology," *Organization Science* (3:3), 1992, p. 383.

Lindblom, C.E. "Alternatives to Validity: Some Thoughts Suggested by Campbell's Guidelines," *Knowledge Creation, Diffusion, Utilization* (8), 1987, pp. 509-520.

Moran, P., and Ghoshal, S. "Value Creation of Firms," in: *Academy of Management Best Paper Proceedings,* J.B. Keys and L.N. Dosier (eds.), 1996, pp. 41-45.

Nahapiet, J., and Ghoshal, S. "Social Capital, Intellectual Capital, and the Organizational Advantage," *Academy of Management Review* (23:2), 1998, pp. 242-266.

Nambisan, S., Agarwal, R., and Tanniru, M. "Organizational Mechanisms for Enhancing User Innovation in Information Technology.," *MIS Quarterly* (23:3), 1999, pp. 365-395.

Nerur, S., Mahapatra, R., and Mangalaraj, G. "Challenges of Migrating to Agile Methodologies," *Communications of the ACM* (48:5), 2005, pp. 73-78.

Nonaka, I. "Dynamic Theory of Organizational Knowledge Creation," *Organization Science* (5:1), 1994, pp. 14-37.

Polanyi, M. *The Tacit Dimension*, Anchor Day Books, New York, 1966.

Putnam, R.D. "The Prosperous Community: Social Capital and Public Life," *American Prospect* (13), 1993, pp. 35-42.

Reich, B.H., and Benbasat, I. "Factors that Influence the Social Dimension of Alignment between Business and Information Technology Objectives," *MIS Quarterly* (24:1), 2000, p. 81.

Senge, P.M. "The Leader's New Work: Building Learning Organizations." *Sloan Management Review* (32:1), 1990, p. 7.

Spender, J.C. "Making Knowledge the Basis of a Dynamic Theory of the Firm," *Strategic Management Journal* (17:S2), 1996, pp. 45-62.

Todd, P., McKeen, J.D., and Gallupe, R.B. "The Evolution of IS Job Skills: A Content Analysis of IS Job Advertisements from 1970 to 1990," *MIS Quarterly* (19:1), 1995, pp. 1-27.

Van Den Bosch, F.A.J., Volberda, H.W., and De Boer, M. "Coevolution of Firm Absorptive Capacity and Knowledge Environment: Organizational Forms and Combinative Capabilities." *Organization Science* (10:5), 1999, p. 551.

Ye, F., and Agarwal, R. "Strategic Information Technology Partnerships in Outsourcing as a Distinctive Source of Information Technology Value: A Social Capital Perspective," Twenty-Fourth International Conference on Information Systems, Seattle, WA, 2003, pp. 304-315.