CAN AGILE AND TRADITIONAL SYSTEMS DEVELOPMENT APPROACHES COEXIST? AN AMBIDEXTROUS VIEW

Vishnu Vinekar, Craig W. Slinkman, and Sridhar Nerur

Emerging evidence seems to indicate that most systems development organizations are attempting to utilize both agile and traditional approaches. This study aims to understand the reasons organizations feel the need for this unlikely juxtaposition and the organizational challenges in sustaining the opposing cultures. Drawing on the extensive literature in organizational theory and management, we advocate ambidexterity as a viable solution to systems development organizations attempting to harness the benefits of both agile and traditional development.

VISHNU VINEKAR is a doctoral candidate in information systems at the University of Texas at Arlington. His research interests include systems development, IT value, collaborative work processes, and knowledge management. He can be reached at vvinekar@uta.edu.

CRAIG W. SLINKMAN is an associate professor of information systems at the University of Texas at Arlington.

SRIDHAR NERUR is an assistant professor of information systems at the University of Texas at Arlington.

THE LOW RATE OF SUCCESS IN THE FIELD of systems development (Standish Group, 1994, 1999, 2001, 2003, 2004) provided the impetus for the development of several new methods and practices. These methods include eXtreme Programming (Beck, 1999), Scrum (K. Schwaber & Beedle, 2002), Dynamic Systems Development Method (Stapleton, 1997), Adaptive Software Development (Highsmith, 2000), Crystal (Cockburn, 2002), and Feature-Driven Development (Palmer & Felsing, 2002). The Agile Manifesto articulates the common principles and beliefs underlying these methods (Cockburn, 2002). The fundamental notions behind the manifesto are:

1. The ingenuity and competence of people as well as their interactions and collaborations are of greater value than tools and processes.
2. Delivering a high-quality working system to the customer is more important than producing copious documentation.
3. The active participation and constant involvement of the customer in systems development yields greater benefits than the fulfillment of predetermined requirements specified in a contract.
4. Recognizing the inevitability of change and embracing it, rather than attempting to cope with it through extensive planning, provides the nimbleness needed to survive in a turbulent business world.

The early adopters of agile methods believe that their use may positively affect their success rate (Berinato, 2001; Larman, 2004; Lindstrom & Jeffries, 2004). Followers of more traditional methodologies believe that agile methods are chaotic and lack the formal procedural rigor that the former possess. One of the most important differences is that traditional development attempts to minimize change in the course of the project through rigorous upfront requirements gathering, analysis, and

Although the majority of adopters believe that the adoption of agile systems development has improved productivity, quality, and business satisfaction, they also feel that other methodologies are necessary.

design; the intent is to attain higher quality results under a controlled schedule. Agile methods, on the other hand, assume that change over the development process is not only inevitable, but also necessary, and aim at achieving innovation through individual initiative (Cockburn & Highsmith, 2001; Highsmith, 2003; Zhiying, 2003). The focus, therefore, is on adaptation and innovation rather than prediction and control.

Agile methods emphasize short iterations, in which the entire project is broken up into several small projects, each lasting between one and six weeks (Larman, 2004). Each iteration deals with only a few prioritized features and ends with a working system as a deliverable. The end of each iteration provides an opportunity to elicit user feedback, to assess the suitability of the delivered solution, and to reflect on what worked and what did not. Developers and users dynamically prioritize features at the beginning of each iteration, and the project grows through evolutionary development (Gilb, 2004). In agile development, the requirements for each iteration are primarily client-driven priorities. Therefore, the requirements that the client perceives as having the highest business value for the project will be included in the first iteration; the remaining features will be reassessed and prioritized for inclusion in future iterations. Although agile methods have been in use for over a decade, substantial debate exists in the industry regarding their effectiveness.

Adoption of agile systems development is increasing in the industry (C. Schwaber & Fichera, 2005). A few industry surveys seem to indicate that most systems development organizations are trying to use both approaches. For example, even though 96.4 percent of Shine Technologies (2003) respondents intend to adopt or continue to use agile methods, only 16 percent believe those methods are suitable for all projects. Although the majority of adopters believe that the adoption of agile systems development has improved productivity, quality, and business satisfaction, they also feel that other methodologies are necessary. Shine Technologies concludes, "[agile methods] should be applied only where it will deliver benefit. . . . Agile processes should be used only for the right projects, and that there is room for other methodologies to sit along side Agile and be used on a project-by-project basis as appropriate." Similarly, the *Methods & Tools* (2005) survey shows that, of the organizations that have adopted agile approaches, only 17 percent are using them for all new projects. This raises our first research question:

> If agile adopters believe that adoption has improved productivity, quality, and business satisfaction (Shine Technologies, 2003), why do they still feel the need for other approaches?

The results of the surveys mentioned above strongly suggest the need to balance agile methods with other approaches. However, both academics and practitioners agree that agile systems development requires a suitable organizational culture (Boehm & Turner, 2004; Lindvall et al., 2002; Nerur, Mahapatra & Mangalaraj, 2005), and changing an organizational culture takes several years (Adler & Shenhar, 1990). This raises our second research question:

> How can organizations overcome the obstacle of changing their organizational culture to sustain both agile and traditional systems development?

This article addresses these questions and makes the following four contributions: First, we examine why the capabilities of both agile and traditional development are needed by systems development organizations. Second, we elaborate on the organizational challenges in sustaining these two opposing cultures. Third, we propose an ambidextrous form of organization as a viable solution to balance agile and traditional systems development while maintaining the necessary organizational cultures for each approach. Fourth, we examine how the ambidextrous structure can dynamically address key IS project characteristics as well as the client organization's characteristics.

THE NEED FOR SIMULTANEOUSLY MANAGING AGILE AND TRADITIONAL SYSTEMS DEVELOPMENT

Agile systems development has attracted a lot of attention in recent times. The Agile 2005 conference was replete with positive experiences organizations have had with agile methods (Agile Alliance, 2005). The benefits reported, such as increased productivity, faster turnaround, shared learning, and higher developer satisfaction, are consistent with earlier evidence cited by proponents such as Cockburn (Cockburn & Williams, 2001). These make for a compelling case for the adoption of agile methods. Nevertheless, does this mean that traditional development built around a tradition of predictability and control has to give way to

It may be difficult and inefficient to strictly adhere to all agile practices in projects that have stable requirements.

agile methods that assume an unanalyzable world fraught with uncertainties? We contend that there is a need to maintain "dual structures" that accommodate both approaches because they each have their benefits, and practical considerations may preclude the simple replacement of one by the other.

Organizations engage in a wide variety of systems development projects. The variations in these projects could be considerable, and the degree of variety is a function of many factors. Boehm and Turner (2004) convincingly argue that there is a pragmatic need to balance stability and agility. They analyze the "home grounds" of agile and traditional approaches based on application characteristics, management characteristics, technical characteristics, and personnel characteristics. Further, they assert that the choice of traditional or agile methods for a given project is largely contingent on five factors:

- The size of the systems development project and team
- The consequences of failure (i.e., criticality)
- The degree of dynamism or volatility of the environment
- The competence of personnel
- Compatibility with the prevailing culture

In essence, they offer a strategy for choosing a particular approach for a particular project based on the risks posed by the five factors mentioned above, implying the simultaneous pursuit of agile and traditional development approaches.

It may be difficult and inefficient to strictly adhere to *all* agile practices in projects that have stable requirements, involve a lot of legacy code and heterogeneous tools/languages (e.g., COBOL, Java, C++), and mainly comprise maintenance tasks with little or no need for search and discovery. Boehm and Turner (2004) point out that traditional development is desirable when the requirements are stable and predictable and when the project is large, critical, and complex. Agile development, on the other hand, is suitable when there is a high degree of uncertainty and risk in the project, arising from frequently changing requirements and/or the novelty of technology used (Boehm & Turner, 2004; Highsmith, 2003). Large and complex projects more suited to the traditional approach may impede the transfer of tacit knowledge as well as entail significant rework, making agile development a suboptimal choice.

Stability brings with it the advantages of discipline and automation and the disadvantage of being overly restrictive. Agility, on the other hand, brings with it the advantages of flexibility and human initiative while inhibiting repeatable processes perceived to contribute to the maturity of an organization (Zhiying, 2003). By systematically codifying the various artifacts of systems development, the traditional approach allows for the exploitation of existing knowledge. On the other hand, much of the knowledge in agile development is tacit (Boehm, 2002). Therefore, an essential tension exists between the goal of optimization to which the traditional orthodoxy of systems development has aspired and the objective of learning and adaptability that agile approaches emphasize. Although agility is necessary for organizational adaptation, stability is necessary for organizational optimization, which results in higher assurances. Hence, systems development organizations need to strike a balance between the two conflicting interests: agility and stability.

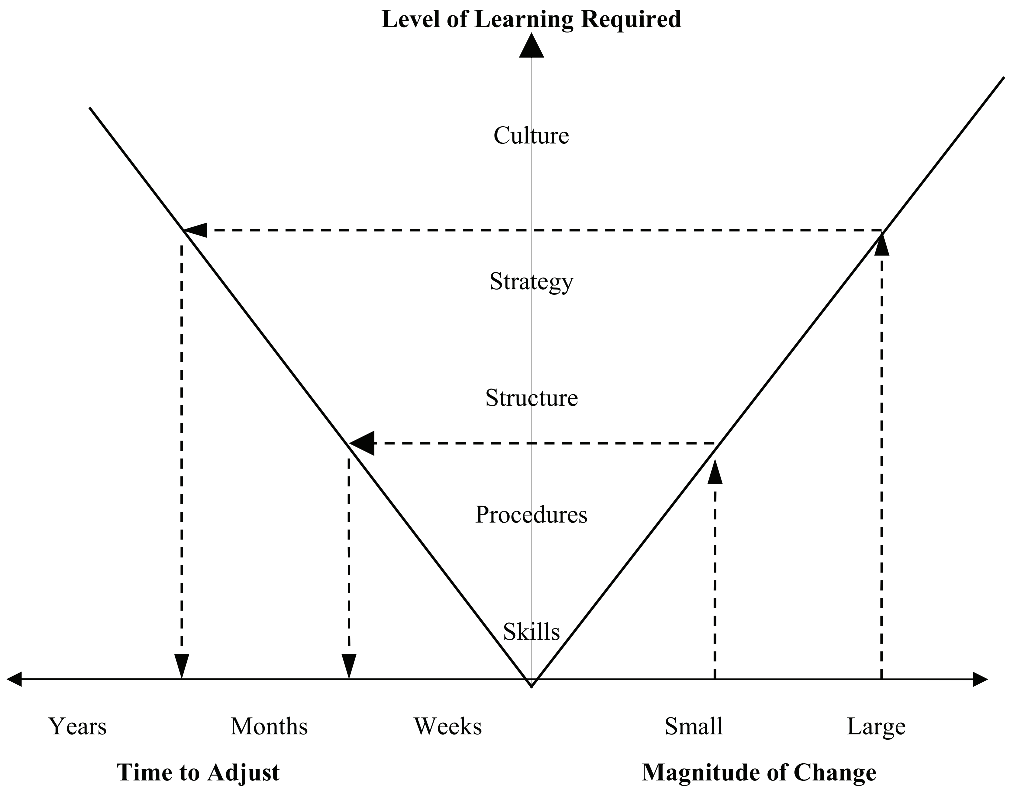
In this section, we argued for the simultaneous pursuit of traditional and agile system development approaches. However, the two approaches differ in many respects and entail conflicting organizational, people, and technical demands that might make the concurrent practice of these methods problematic. The next section discusses the challenges that confront organizations that plan to follow both approaches at the same time.

OBSTACLES TO BEING BOTH AGILE AND TRADITIONAL

An organization that is attempting to use both agile and traditional development on different projects faces several challenges. This is because although project characteristics and client characteristics vary, it may be very difficult to change the systems development organization's own characteristics from project to project. These challenges can be viewed as occurring at four levels: management and organizational, people, process, and technology (Nerur et al., 2005).

The framework for organizational change articulated by Adler and Shenhar (1990) is useful for assessing the effort required to meet these challenges (see Figure 1). Of the four levels that Nerur et al. (2005) discuss, technological and process changes occur at the skills and procedures levels, where, relatively speaking, the magnitude of change is small, the level of learning needed is low, and the time to adjust

FIGURE 1 Framework for Organizational Change (Adler & Shenhar, 1990)



is short. However, the people and management/organizational changes occur at the levels of culture, strategy, and structure, where the magnitude of change is relatively large, the level of learning required is high, and the time to adjust is long. Therefore, we focus below on the challenges that these latter two levels present to systems development organizations that are pursuing agile methods for some projects and traditional methods for others.

People Level

Agile systems development places a premium on people and their interactions. The emphasis is on teams and on the intense dynamics of team interactions. The roles of agile team members are interchangeable, and developers often choose roles that are not in their area of specialty (Martin, 2003). Self-organization is one of the key traits of such systems development teams. The traditional role of a project manager as planner, organizer, and controller disappears, and the role of a facilitator or coach who effectively manages the collaborative efforts of team members without stifling their creativity takes its place (Highsmith, 2003). Proponents of agile methods argue that processes should be flexible and dynamic enough to mold around the competencies of people, and not the other way around (Cockburn & Highsmith, 2001). This focus on people is a significant departure from the traditional systems development's focus on processes.

Management and Organizational Level

Agile and traditional systems development have conflicting organizational cultures, management styles, organizational forms, and reward systems (Nerur et al., 2005). The influence of organizational culture on shaping the assumptions and biases of its employees is well documented (Charette, 2003). In fact, departments within companies may develop their own "mini-cultures" that become ingrained in the work habits and actions of their personnel (Cockburn, 2002). Organizational forms that have supported a culture of hierarchical control for several years may find it particularly difficult to accommodate some of the characteristics of agile development, such as self-organizing teams, pluralistic decision-making contexts involving stakeholders with

Agile systems development places a premium on people and their interactions.

diverse interests and goals, and a collaborative environment fostered by strong leadership rather than strict authority. Organizations also need to reevaluate their reward systems to encourage agile practices, where collective goals supersede individual accomplishments.

It is apparent from the preceding discussions that factors related to people, organization, and management pose significant challenges to the simultaneous pursuit of agile and traditional development. Systems development organizations face the methodological dilemma of achieving optimization while fostering an environment of agility to respond quickly to changes. However, the problem that confronts systems development organizations is not unique. There is a large body of literature in organizational management that has extensively studied the essential tension and trade-offs between stability and agility. These include the conflicting demands and organizational dilemma related to exploitation and exploration (March, 1991; Benner & Tushman, 2003; Gibson & Birkinshaw, 2004; He & Wong, 2004).

More specifically, organizations generally pursue two types of innovation behaviors; namely, *exploitation* and *exploration* (March, 1991; Katila & Ahuja, 2002). Exploitation behaviors focus on core competencies, efficiency, routines, incremental changes, and the like (He & Wong, 2004), which are often associated with alignment with a stable environment. Exploration behaviors, on the other hand, assume a changing environment that demands frequent experimentation, learning by doing, risk-taking propensity, innovative behaviors, and so forth. Exploitation and exploration behaviors are not mutually exclusive, and it is considered detrimental to pursue one at the expense of the other (March, 1991). Rather, these two behavioral dimensions *together* enable an organization to be innovative, flexible, and effective without losing the benefits of stability, routinization, and efficiency (Katila & Ahuja, 2002).

He and Wong (2004, p. 481) make the following observation:

> In general, exploration is associated with organic structures, loosely coupled systems, path breaking, improvisation, autonomy and chaos, and emerging markets and technologies. Exploitation is associated with mechanistic structures, tightly coupled systems, path dependence, routinization, control and bureaucracy, and stable markets and technologies.

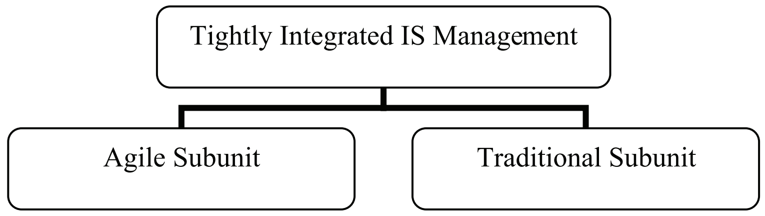
These apparent differences between exploitation and exploration are also consistent with the contrasts between agile and traditional systems development presented by Nerur et al. (2005). The insights into reconciling the differing structures, cultures, strategies, work habits and roles of people, information scanning and decision-making processes, tools and techniques, and inherent processes associated with exploitation and exploration (He & Wong, 2004) are valuable to organizations endeavoring to pursue both traditional and agile systems development methods simultaneously.

Further, the literature on organizational theory and learning also provides a persuasive reason for reconciling the differences between the two approaches, and for providing an organizational climate conducive to the *simultaneous* practice of agile and traditional development (He & Wong, 2004; Tushman & O'Reilly, 1996). Recent empirical evidence suggests that the notion of *ambidexterity*, which permits the simultaneous pursuit of these contrasting approaches, is an effective and viable solution to the stability–agility dilemma (Katila & Ahuja, 2002; He & Wong, 2004; O'Reilly & Tushman, 2004). The ability to be both *aligned* with the existing environment as well as *adaptive* to cope with the dynamics of a changing world is positively associated with superior performance (Gibson & Birkinshaw, 2004; He & Wong, 2004).

SUSTAINING THE DUAL CULTURES OF STABILITY AND AGILITY THROUGH AMBIDEXTERITY

Exploitation and exploration are among the many paradoxes organizations need to manage in order to survive (Gibson & Birkinshaw, 2004). Indeed, the truly successful organizations foster an environment that encourages the simultaneous presence of paradoxical and contradictory forces, such as differentiation versus integration (Lawrence & Lorsch, 1967), stability and change, alignment and adaptability (Gibson & Birkinshaw, 2004), variation-reducing versus variation-increasing (Burgelman, 1991, 2002), loose and tight coupling (Weick, 1976; Zaltman, Duncan, & Holbeck, 1973), and so forth. The acceptance of paradox as a ubiquitous organizational phenomenon has led some theorists to recognize that it is detrimental to view opposing elements such as exploitation and exploration as being mutually exclusive. Rather, organizations that facilitate their simultaneous presence have much to gain

FIGURE 2 The Ambidextrous Organization



(Cameron & Quinn, 1988; Gibson & Birkinshaw, 2004).

The fact that different structures influence different innovative behaviors (such as exploitative and explorative) has long been recognized in the management literature. For example, Burns and Stalker (1961) characterize structures as mechanistic and organic, the former being suitable for the attainment of goals related to stability, routinization, and efficiency, whereas the latter is necessary if flexibility and adaptability are the primary concerns. The origins of ambidexterity may be traced to the work of Duncan (1976), who proposed a dual structure to deal with the paradox of stability and change. However, the recent resurgence of interest in ambidexterity as well as the elucidation of its form and characteristics in an organization may be largely attributed to the works of Tushman and O'Reilly (O'Reilly & Tushman, 2004; Tushman & O'Reilly, 1996). In light of recent empirical evidence on the efficacy of ambidexterity and its positive influence on an organization's performance (He & Wong, 2004; Jansen, Van den Bosch, & Volberda, 2005), we focus here on the ambidextrous form proposed by Tushman and O'Reilly (1996).

The ambidextrous organization has subunits that are highly coupled within subunits and loosely coupled across subunits but are tightly integrated at the senior executive level (O'Reilly & Tushman, 2004). The task, culture, individuals, and organizational arrangements are highly consistent within each subunit and highly differentiated from the other subunits (Benner & Tushman, 2003). Case studies suggest that ambidextrous organizations (such as USA Today and Ciba Vision) can be superior to other organizations (O'Reilly & Tushman, 2004). Empirical evidence has also demonstrated that the interaction between explorative and exploitative strategies positively affects performance, whereas an imbalance between the two strategies negatively affects performance (He & Wong, 2004).

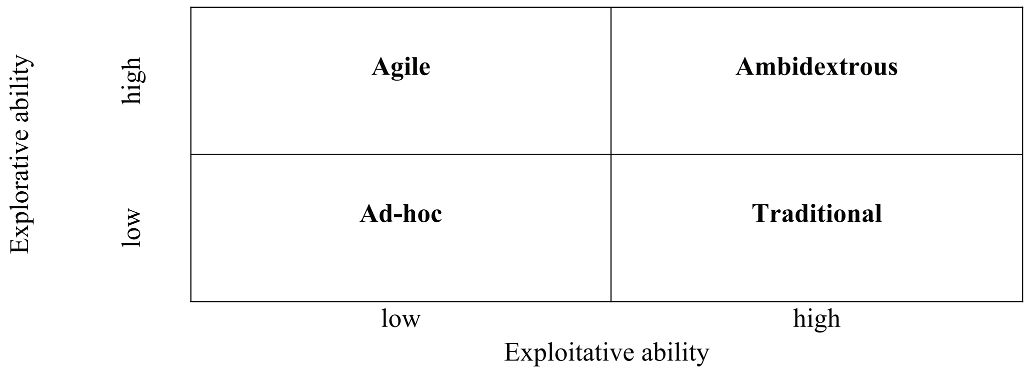
This suggests that an ambidextrous IS development organization may be able to simultaneously pursue and reap the benefits from both traditional and agile development. Such an organization would consist of at least two subunits: an agile subunit and a traditional subunit. The traditional subunit would have a more hierarchical structure, with the project manager as the planner, segregating and delegating responsibility between a large team of specialized developers working individually. The agile subunit, on the other hand, would have a more decentralized, flexible structure, in which small teams of developers with multidisciplinary skills work closely with customers and diverse stakeholders. Separating the two units and buffering them from each other would ensure that they preserve their own cultures. This may also be necessary because having collocated agile and traditional systems development teams may create elitist attitudes between the teams and hinder progress (Nerur et al., 2005). However, a single, tightly integrated IS governance structure above both subunits is also needed (Figure 2). This IS management team assesses the relative feasibility of both development methodologies for each project and delegates project work accordingly. This structure also allows IS management to learn from the successes and failures of both subunits and to modify them as needed.

The composition of the two subunits, agile and traditional (plan driven), would differ fundamentally along the four levels of management/organizational, people, process, and technology (Nerur et al., 2005). Table 1 summarizes the opposing characteristics of these two subunit types.

TABLE 1 The Ambidextrous Systems Development Organization

	Agile Subunit	Traditional/Stable Subunit
Management and organizational	Leadership and collaboration	Command and control
	Cooperative	Autonomous
	Flexible	Disciplined
	Manager as facilitator	Manager as planner
	Tacit knowledge	Explicit knowledge
	Team reward systems	Individual reward systems
People	Collaborative work	Individual work
	Multidisciplinary skills	Specialized skills
	Pluralist decision making	Managerial decision making
	High customer involvement	Low customer involvement
	Small teams	Large teams
Process	People centric	Process centric
	Speculative	Standardized
	Assess progress	Measure progress
	Evolutionary development	Life-cycle development
	Write tests prior to code	Write code prior to tests
	Individual approach to projects	Unified approach to projects
	Adaptable	Preplanned
	Iterative	Linear
	Short durations	Long durations
Technology	Object oriented	Structured or object oriented
	Tools for iteration	Standardized tools

FIGURE 3 The Relationship between Explorative Abilities and Exploitative Abilities for Systems Development Organizations



Further, an ambidextrous form may be well suited to enable a traditional systems development organization to incorporate agile development without disrupting its existing organization. Instead of changing the management style of the entire organization, it could create a structurally buffered, agile subunit. Employees who are more tolerant of ambiguity as well as those who work better in teams could be part of the agile subunit. Leaders, collaborative decision makers, and developers with multiple skills potentially perform to their fullest capability in such units. Managers and employees long accustomed to a hierarchical structure that emphasizes command and control, role specialization, solitary work habits, and a high degree of formalization may be more comfortable and effective in a traditional subunit. Thus, an ambidextrous organization promises to be an effective way to achieve the benefits of stability without compromising its ability to dynamically respond to changes in the environment (Figure 3).

An ambidextrous organization may be particularly well suited to use agile methods for larger projects.

DYNAMICS OF THE AMBIDEXTROUS ORGANIZATIONAL STRUCTURE

Emerging evidence seems to indicate that agile systems development is gaining acceptance among systems development organizations (*Methods & Tools*, 2005; C. Schwaber & Fichera, 2005; Shine Technologies, 2003). Some proponents of agile development claim universal applicability of their methods, and others believe that it is only suitable in particular situations. Boehm and Turner (2004) identify five critical factors to assess the suitability of agile or traditional development: size, criticality, dynamism, personnel, and culture. However, the effectiveness of their framework depends on the extent to which the systems development organization understands how its personnel and cultural factors differ from those of the client environment. Therefore, we analyze the dynamics of the ambidextrous structure by addressing these five factors as relevant across three levels:

1. Systems development organization: These include the systems development organization's culture and personnel, which it has direct control over.
2. Information systems project: These include the information systems project's size, criticality, and dynamism, which the systems development organization attempts to control through its actions.
3. Client organization: These include the client's culture and its ability to communicate requirements for systems functionality, which the systems development organization has no direct control over.

Systems Development Organization Factors

Without an ambidextrous structure, the systems development organization's culture and personnel are critical constraints (Boehm & Turner, 2004) that limit the organization's ability to use agile or traditional development. With an ambidextrous structure, the systems development organization turns these constraints into strengths. Through its pursuit of dual cultures and personnel, the ambidextrous organization provides enhanced capabilities to handle critical factors of the information systems project and of the client organization.

Information Systems Project Factors

The ambidextrous organization increases the system development organization's flexibility to address three project factors associated with the choice between agile and traditional methods: size, criticality, and dynamism.

Size. Whereas some developers report difficulties with using agile approaches for large development teams (Boehm & Turner, 2004), others report successes, including a 40-person team using agile modeling, distributed teams of 80 people, a 150-person team divided into smaller teams each with its own methodology, and an 800-person team organized into a "scrum of scrums" (Lindvall et al., 2002). These experiences may indicate that it is not that agile development does not scale to large projects, but that very few developers have attempted this.

An ambidextrous organization may be particularly well suited to use agile methods for larger projects. Because experience with using agile development for large projects is limited, the organization may continue to assign its largest projects to its traditional development subunit but assign incrementally larger projects to its agile subunit. Using agile methods for large projects essentially involves breaking down the project into smaller development teams. This raises the issue of coordination between teams. Scaling agile development therefore involves learning to use coordination mechanisms between systems development teams (Lindvall et al., 2002), including regular team leader meetings and core teams to handle coordination issues. The agile subunit with an explorative culture (March, 1991) and double-loop learning (Argyris, 1977), can gradually learn to scale agile methods to larger projects. The agile subunit can apply its learning to increasingly larger projects until it feels that it has reached a comfortable upper limit, which may vary among organizations.

Dynamism. Adopters of agile development believe that its most useful feature is its ability to respond to change (Shine Technologies, 2003). However, some "heavy" approaches, such as the Rational Unified Process (RUP), are also iterative, incremental, and evolutionary and can allow a traditional development culture to handle project dynamism. The main difference between agile development and RUP is that the latter is more dependent on up-front, explicit, and documented plans (Cantor, 2001). RUP sacrifices some speed and flexibility so that changes in the system can be explicated, agreed on, and finalized formally before implementation. Agile development gains speed by

The client's culture may be the deciding factor in using agile or traditional methods for a project.

depending less on procedure and more on the on-site customer. However, this dependence on the on-site customer may cause project failure if the on-site customer is misaligned with the stakeholder goals. Similarly, RUP's dependence on following procedure and formal, documented changes may result in development failure if the client is unable to articulate needs through formal documents. Therefore, we see that dynamism is less of a constraint relative to the client's ability to communicate requirements through formal documents vis-à-vis its ability to provide a CRACK customer (collaborative, representative, authorized, committed, and knowledgeable). We elaborate this issue under client organization factors.

Criticality. Boehm and Turner (2004) believe that agile development may not be suitable for safety-critical projects. However, other adopters of agile approaches believe that some of agile development's methods make it easier to address critical issues (Lindvall et al., 2002; Turk, France, & Rumpe, 2002). First, test-first development ensures rigorous testing of safety-critical features. Second, continuous client feedback may refine safety-critical features that were poorly defined earlier. Third, the practice of pair programming may help catch deficiencies that formal reviews may miss. The difference between these opposing views may be that Boehm and Turner (2004) believe that the client is able to specify safety-critical requirements up-front, whereas Turk et al. (2002) believe that the on-site customer is able to refine safety-critical requirements through iterative feedback. As with dynamism, we see that the choice to use agile or traditional development is less dependent on the criticality of the project and more dependent on the ability of the client to communicate safety-critical requirements through explicit specifications vis-à-vis tacit collaboration. We elaborate this issue below.

Client Organization Factors

From the above, we see that an ambidextrous systems development organization can use the culture and personnel characteristics of its agile and traditional units to mitigate the constraints imposed by the project characteristics of size, criticality, and dynamism. Therefore, it has the added flexibility of addressing the client's organizational culture and abilities to improve customer satisfaction.

Client Culture. The client's culture may be the deciding factor in using agile or traditional methods for a project. First, clients may be uncomfortable with agile systems development's flexible budgets and schedules and may prefer an up-front contractual obligation to specific features, deadlines, and costs. Second, using an agile approach entails formidable responsibility on the client's part. Agile methods require identifying and prioritizing features and continuous, active collaboration throughout the development. The client may be unwilling to take on this amount of responsibility. Third, the client's management and IT departments may dislike having their organization constantly interrupted by frequent implementations of deliverables for user feedback. Similarly, it may also be possible that the client organization has a highly flexible, adaptive culture that is uncomfortable with the up-front, explicit, formal, and detailed specification that traditional development entails. In these cases, an ambidextrous system development organization may be able to derive greater customer satisfaction by assigning the project to the subunit more aligned with the client organization's culture.

Client's Ability to Communicate System Functionality. Agile methods are highly dependent on the on-site customer to identify and prioritize features, provide feedback, and guide change through the course of the development. This reliance on the customer can fail if the on-site customer goals are misaligned with other stakeholders' goals. For example, the Chrysler Comprehensive Compensation (C3) System, which was developed using eXtreme Programming (Paulk, 2001), went over time and over budget and was canceled before it was completed. The project was reportedly canceled because the on-site customer kept tweaking the existing payroll system, whereas the goal of other stakeholders was to replace the existing legacy systems handling the company's payroll with a new, integrated payroll system (Cunningham & Cunningham, Inc., n.d.; Hendrickson, 2001; Deursen, 2001). This example indicates how pivotal the on-site customer is to agile systems development. Agile methods require a CRACK customer (Boehm & Turner, 2004; Nerur et al., 2005) to succeed. The client organization may not have such a person who is expendable enough to be continuously involved with the development (Highsmith, 2004). Traditional development, on the other hand, is highly dependent on the clients' abilities to specify requirements

New organizational structures are needed to sustain these opposing cultures so that systems development organizations can reap the full benefits of both agile and traditional systems development.

up-front. This can cause development failure because the clients may be unable to articulate their needs clearly (Ackoff, 1967). Therefore, we see that the clients' ability to specify system functionality becomes a critical factor in deciding between agile and traditional development. If the client is able to articulate functionality through formal requirements, traditional development is more suitable. If the client is able to provide a CRACK customer, agile development is more suitable. However, if the client does not have either ability, the project is highly likely to fail.

IMPLICATIONS

The vast majority of adopters of agile systems development methods believe that their adoption has resulted in increased business satisfaction, increased quality, and increased productivity (Shine Technologies, 2003). Yet, while indicating that they will continue using these methods, these organizations also believe that agile methods are not suitable for all projects (*Methods & Tools*, 2005; Shine Technologies, 2003). Instead, the majority of organizations are attempting to use both agile and traditional systems development, on a case-by-case basis.

There is consensus among academics and practitioners that agile systems development needs a suitable organizational culture to sustain it, one that is very different from the organizational culture needed for traditional systems development (Boehm & Turner, 2004; Lindvall et al., 2002; Nerur et al., 2005). Yet changing an organizational culture is extremely difficult and may take several years (Adler & Shenhar, 1990), and these opposing cultures cannot coexist within the same organizational structure (March, 1991). Therefore, new organizational structures are needed to sustain these opposing cultures so that systems development organizations can reap the full benefits of both agile and traditional systems development.

As the main contribution of this article, we suggest an ambidextrous form of organization (Tushman & O'Reilly, 1996) to overcome the challenges of sustaining the dual cultures. The ambidextrous organization essentially has two types of subunits — one with an organizational culture that sustains agile systems development and another with an organizational culture that sustains traditional systems development. These two subunits of the organization diverge along four dimensions: management, people, process, and technology. To maintain these two opposing cultures, the two subunits are structurally buffered from each other, allowing each to coexist separately. However, IS management over the two subunits needs to be tightly integrated to enable a common organizational vision and to prevent the subunits from working against each other's interests.

We describe how such an organizational form overcomes the critical factors that have been argued to constrain other systems development organizations (Boehm & Turner, 2004). An ambidextrous organization benefits from its own organizational characteristics of culture and personnel. It is less constrained by the IS project characteristics of size, criticality, and dynamism. It is therefore able to maximize customer value by choosing agile or traditional systems development based on the client's characteristics — namely, the client organization's culture and the client's ability to specify requirements through formal means vis-à-vis its ability to provide a CRACK customer for a project.

For researchers, there are several avenues of future work. Aside from anecdotal evidence and consensus among small groups of practitioners, we found no empirical studies addressing the challenges that traditional systems development organizations face in adopting agile methods. Future research needs to identify factors that may affect organizations that have attempted this adoption. In addition, the efficiency of the ambidextrous form of organization in easing the adoption of agile systems development needs to be tested empirically.

CONCLUSION

Although agile methods are gaining acceptance among traditional systems development organizations, the majority of these organizations seem to indicate a preference to sustain both forms of development. In the rhetoric of the superiority of one development methodology over the other, very little has been learned about the challenges faced by organizations that attempt both and even less about any successes in sustaining the opposing cultures. This article prescribes adopting an organizational form that may enable this duality. We detail the form that this organization should take and how it can dynamically address differences in key IS project characteristics as well as the client organization's characteristics. Through an ambidextrous organizational structure, systems development organizations can reap the benefits of both agile and traditional systems development. ▲

References

Ackoff, R. L. (1967). Management misinformation systems. *Management Science*, (14)4, 147-156.

Adler, P. S., & Shenhar, A. (1990). Adapting your technological base: The organizational challenge. *Sloan Management Review,* (32)1, 25-37.

Agile Alliance (2005). Agile 2005 conference. Retrieved February 23, 2006 from http://www.agile2005.org/track/experience_reports

Argyris, C. (1977). Double loop learning in organizations. *Harvard Business Review* (55)5, 115-125.

Beck, K. (1999). *Extreme programming explained: Embrace change*. Reading, MA: Addison-Wesley.

Benner, M. J., & Tushman, M. L. (2003) Exploitation, exploration, and process management: The productivity dilemma revisited. *Academy of Management Review* (28)2, 238-256.

Berinato, S. (2001). The secret to software success. *CIO,* July 1, pp. 76-83

Boehm, B. (2002). Get ready for agile methods, with care *IEEE Computer* (35)1, 64-69.

Boehm, B., & Turner R. (2004). *Balancing agility and discipline: A guide for the perplexed,* Boston: Addison-Wesley.

Burgelman, R.A. (1991). Intraorganizational ecology of strategy making and organizational adaptation: Theory and field research. *Organization Science*, 2, 239-262.

Burgelman, R. A. (2002). Strategy as a vector and the inertia of coevolutionary lock-in. *Administrative Science Quarterly,* 47, 325-357.

Burns, T., & Stalker, G. M. (1961). *The management of innovation,* London: Tavistock.

Cameron, K. S., & Quinn, R. E. (1988). Organizational paradox and transformation. In R. E. Quinn, and K. S. Cameron, (Eds.) *Paradox and Transformation,* Cambridge, MA: Ballinger Publishing Company, 1-18.

Cantor, M. (2001). The Rational Unified Process for systems engineering. *The Rational Edge*. Retrieved February 1, 2006 from www.ibm.com/developerworks/rational/library/content/RationalEdge/dec01/RUPSEDec01.pdf

Charette, R. (2003). Challenging the fundamental notions of software development. *Agile Project Management*, Arlington, MA: Cutter Consortium.

Cockburn, A. (2002). *Agile software development*. Boston: Addison-Wesley.

Cockburn, A., & Highsmith, J. (2001). Agile software development, the people factor. [Electronic version]. *IEEE Computer* (34)11, 131-133. Retrieved October 18, 2004, from the IEEE Xplore database.

Cockburn, A., & Williams, L. (2001). The costs and benefits of pair programming. In *Extreme programming examined*. Boston, MA: Addison-Wesley.

Cunningham & Cunningham, Inc. (n.d.) Cthree project terminated. Retrieved from Wiki Wiki Web on February 17, 2006. http://c2.com/cgi/wiki?CthreeProjectTerminated

Deursen, A. V. (2001). Customer involvement in extreme programming: XP2001 Workshop Report. *ACM SIGSOFT Software Engineering Notes*, Nov. 2001, pp. 70-73.

Duncan, R. B. (1976). The ambidextrous organization: Designing dual structures for innovation. In R. H. Kilmann, L. R. Pondy, & D. Slevin (Eds.), The management of organization, vol. 1: 167-188. New York: North-Holland.

Gibson, C. B., & Birkinshaw, J. (2004). The antecedents, consequences, and mediating role of organizational ambidexterity. *Academy of Management Journal*, 47(2), 209-226.

Gilb, K. (2004). Evo — Evolutionary project management & product development Retrieved on October 4, 2004 from http://www.gilb.com/Pages/2ndLevel/gilbdownload.html#Whirl-Wind

He, Z., & Wong P. (2004). Exploration vs. exploitation: An empirical test of the ambidexterity hypothesis. *Organization Science,* (15)4, 481-494.

Hendrickson, C. (2001). Will Extreme Programming kill your customer? Position Paper, OOPSLA 2001. Retrieved February 17, 2006 from http://www.coldewey.com/publikationen/conferences/oopsla2001/agileWorkshop/hendrickson.html

Highsmith, J. (2000). *Adaptive software development: A collaborative approach to managing complex systems*. New York: Dorset House.

Highsmith, J. (2001). Order for free: An organic model for adaptation. In L. L. Constantine (Ed.), *Beyond chaos: The expert edge in managing software development* (pp. 251-257). Boston: Addison-Wesley.

Highsmith, J. (2003). Agile project management: Principles and tools. *Agile Project Management,* (4)2. Cutter Consortium.

Highsmith, J. (2004) Objections to agile development. *Agile Project Management,* May 2004. Cutter Consortium.

Jansen, J. J. P., Van den Bosch, F. A. J., & Volberda, H. W. (2005). Exploratory innovation, exploitative innovation, and ambidexterity: The impact of environmental and organizational antecedents. *Schmalenbach Business Review,* 57, 351-363.

Katila, R., & Ahuja, G. (2002). Something old, something new: A longitudinal study of search behavior and new product introduction. *Academy of Management Journal*, 45(6), 1183-1194.

Larman, C. (2004). *Agile and iterative development: A manager's guide*. Boston: Addison-Wesley.

Lawrence, P. R., and Lorsch, J. W. (1967). *Organizations and environment.* Homewood, IL: Irwin.

Lindstrom, L., & Jeffries, R. (2004) Extreme programming and agile software development methodologies. *Information Systems Management*, (21)3, 41-52.

Lindvall, M., Basili, V., Boehm, B., Costa, P., Dangle, K., Shull, F., et al. (2002). Empirical findings in agile methods. Presented at the Extreme Programming and Agile Methods — XP/Agile Universe, Chicago, IL, USA.

March, J. G. (1991). Exploration and exploitation in organizational learning. *Organization Science*, (2)1, 71-87.

Martin, R. C. (2003). *Agile software development: Principles, patterns, and practices*. Upper Saddle River, NJ: Prentice Hall.

Methods & Tools (2005). Software development poll archives. Retrieved February 13, 2006 from http://www.methodsandtools.com/dynpoll/oldpoll.php?Agile

Nerur, S., Mahapatra, R., & Mangalaraj, G. (2005). Challenges of migrating to agile methodologies. *Communications of the ACM*, (48)5, 73-78.

O'Reilly, C. A., III & Tushman, M. L. (2004). The ambidextrous organization. *Harvard Business Review* (82)4, 74-81.

Palmer, S. R., & Felsing, J. M. (2002). *A practical guide to Feature-Driven Development*. Upper Saddle River, NJ: Prentice Hall PTR.

Paulk, M. C. (2001). Extreme programming from a CMM perspective. *Paper for XP Universe, Raleigh, NC, 23-25 July 2001.*

Schwaber, C., & Fichera, R. (2005). Corporate IT leads the second wave of agile adoption. Forrester, November 30, 2005. Retrieved February 7, 2006 from http://www.forrester.com/Research/Document/Excerpt/0,7211,38334,00.html

Schwaber, K., & Beedle, M. (2002). *Agile software development with Scrum*. Upper Saddle River, NJ: Prentice Hall.

Shine Technologies (2003). *Agile methodologies survey results.* Retrieved November 21, 2004 from http://www.shinetech.com/download/attachments/98/ShineTechAgileSurvey2003-01-17.pdf

Standish Group (1994). The CHAOS report. Retrieved February 17, 2006 from Sample Research from The Standish Group Web site: http://www.standishgroup.com/sample_research/PDFpages/chaos1994.pdf

Standish Group (1999). CHAOS: A recipe for success. Retrieved February 17, 2006 from Sample Research from The Standish Group Web site: http://www.standishgroup.com/sample_research/PDFpages/chaos1999.pdf

Standish Group (2001). Extreme CHAOS. Retrieved February 17, 2006 from Sample Research from The Standish Group Web site: http://www.standishgroup.com/sample_research/PDFpages/extreme_chaos.pdf

Standish Group (2003). *CHAOS Chronicles Version 3.0,* Yarmouth, MA: The Standish Group International.

Standish Group (2004). CHAOS demographics and project resolution. Retrieved February 17, 2006 from Sample Research from The Standish Group Web site: http://www.standishgroup.com/sample_research/PDFpages/q3-spotlight.pdf

Stapleton, J. (1997). *DSDM, Dynamic Systems Development Method: The method in practice*. Harlow, Eng: Addison-Wesley.

Turk, D., France, R., & Rumpe, B. (2002). Limitations of agile software processes. Proceedings of the Third International Conference on eXtreme Programming and Agile Processes in Software Engineering, pp. 43-46, May 26-29, 2002, Alghero, Sardinia, ITALY.

Tushman, M. L., & O'Reilly, C. A., III (1996). Ambidextrous organizations: Managing evolutionary and revolutionary change. *California Management Review* (38)4, 8-30.

Weick, K. E. (1976). Educational organizations as loosely coupled systems *Administrative Science Quarterly*, (21)1, 1-19.

Zaltman, G., Duncan, R., & Holbeck, J. (1973). *Innovations and organizations.* New York: Wiley.

Zhiying, Z. (2003). CMM in uncertain environments. *Communications of the ACM* (46)8, 115-119.