13 HOW AGILE IS AGILE ENOUGH? Toward a Theory of Agility in Software Development

Kalle Lyytinen[1]
Case Western Reserve University
Cleveland, Ohio U.S.A.

Gregory M. Rose
Washington State University
Vancouver, Washington U.S.A.

Abstract *This paper outlines a theory of software development agility that draws upon a model of IT innovations. We examine how both exploration and exploitation impact software development agility. We propose a sequential model of learning in which agility is driven by exploration versus exploitation needs and development agility is influenced by learning focus. Organizations need to balance multiple conflicting goals including speed, quality, cost, risk and innovative content. The value of the model is illustrated by probing how software organizations controlled their agility in Internet computing between the years 1997 and 2003.*

1 INTRODUCTION

In software, *agility* can be defined as developers' ability to sense and respond nimbly to technical and business opportunities in order to stay innovative in a turbulent environment. An *agile* software development organization has the capability to respond to unexpected environmental changes and increase its process speed. In the past, the Information Systems literature sought to control the outcome quality and reliability by submitting to virtues of system engineering: the system must be flawless, user friendly,

[1]Author order is alphabetical; the authors contributed equally to this paper.

or scalable. This logic pervaded debates around the "software crisis" and motivated the development of approaches such as structured methodologies and process improvement frameworks.

This worldview faced a reality check when new economy rebels changed the idea of system development. Software had to be developed at, and for markets in, a fast pace (Baskerville et al. 2001; Carstensen and Vogelsang 1999; Cusumano and Yoffie 1999; Lyytinen and Rose 2003; Pressman 1998). The key to competitiveness was agility and this echoed well with research in strategy on dynamic capability (D'Aveni 1994; Teece et al. 1997) and rapid product development (Kessler and Chakrabarti 1996). However, it is not clear what agility in software means. Is it the speed at which some type of running system is available? Is it the change in ratio between delivered functionality and the elapsed time? Or is it the client's increased velocity? All these speeds are distinct aspects of *agility* and dictate different ramifications on how to improve it. Another issue relates to antecedents of agility, and to what extent the organizations can manipulate them. There is a huge difference in changing the speed in doing X when compared to changing the speed in which the organization moves from doing X to doing Z. Finally, we must better understand how agility relates to other process outcomes such as risk or how agility varies during technology diffusion (Baskerville et al. 2001, Lambe and Spekman 1997).

This paper develops a model that accounts for differences in the relative change and types of *agility* that organizations can achieve at different stages of technology diffusion. We show that the need for agility must be balanced with other desirable process features such as innovative content, risk, quality, and cost and how process outcomes are valued in competitive environments. We validate the model by a multisite case study of software development in seven organizations that adopted Internet computing over a 5-year period. The study illustrates how organizations changed and controlled their agility over the study period by changing their perceptions of agility and the need for it. These changes were outcomes of continued attempts to balance agility with other process features such as innovative content, cost, quality, and risk. The remainder of the paper is organized as follows. Section 2 formulates the development model and reviews the related literature. Section 3 describes the field study, while section 4 reports the main findings of the study.

2 RELATED LITERATURE AND SOFTWARE DEVELOPMENT AGILITY MODEL

The goal of the software development agility model is to detect dependencies between specific environmental, organizational, and market factors that affect how agility and other process factors relate to one another. The model draws on Swanson's (1994; see also Lyytinen and Rose 2003) model of IT innovation and March's (1991) exploration–exploitation dichotomy. According to the model, software organizations are engaged in both exploration and exploitation while innovating with information technology. During periods of fast transition (e.g., the shift to Internet computing), the exploration speed (absorptive capability of technical potential) and development speed

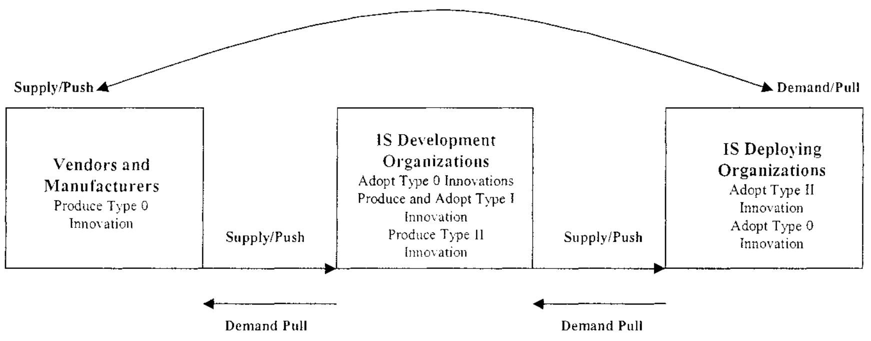
Supply/Push
Demand/Pull
Vendors and Manufacturers
Produce Type 0 Innovation
Supply/Push
IS Development Organizations
Adopt Type 0 Innovations
Produce and Adopt Type I Innovation
Produce Type II Innovation
Supply/Push
IS Deploying Organizations
Adopt Type II Innovation
Adopt Type 0 Innovation
Demand Pull
Demand Pull

Figure 1. IT Value Chain and Realms of IS Innovation

(fast exploitation) must be combined to harness the new technology. Yet, exploration and exploitation set up quite different demands and contexts for agility. To understand this process, the context of innovation must be understood.

2.1 Model of IT innovation

The concept of IT innovation has remained poorly developed despite the vast literature on IT-based innovation (Lyytinen and Rose 2003; Swanson 1994). IT innovation has multiple sources and a broad scope in the IT value chain (Swanson 1994). As a consequence, innovation within system development (such as agility) is not a singular event, but subsumes a chain of events which all portray significant departures from existing practices. An IT innovation normally traverses a complex ecology of innovative events (see Figure 1) (Lyytinen and Rose 2003; Swanson 1994).

Figure 1 shows three value activities in the IT domain: (1) *creation of IT base technologies* such as operating systems by vendors (we call this base innovation a *Type 0 innovation*), (2) *creation of processes, technologies and organizational arrangements* that enable better or more reliable delivery of software in organizational contexts (called a *Type I innovation*), and (3) *development and adoption of new types of IT solutions* (called a *Type II innovation).* The arrows in Figure 1 show how downstream organizations adopt innovations produced by companies upstream so as to increase their overall scope and quality of IT deployment. Hence, IT innovation means many things (Lyytinen and Rose 2003): breakthroughs in computing capability (Type 0 innovation), departure from current methods to develop applications (Type I innovation), or novel applications (Type II innovation). The connection is not causal: many Type II innovations do not necessarily affect other parts. The case for such Type 0 innovations is much rarer, but still possible. The value chain also suggests that innovations can take place in any part of the chain and by doing so they can affect other innovations upstream or downstream.[2]

[2]Swanson calls these strong and weak order effects.

Due to the technology dependent nature of software innovation, organizations adopting significant Type 0 and Type I innovations *together* can *produce* radically new applications (Type II) and thereby engage in *disruptive IT innovations* (Lyytinen and Rose 2003). These disruptions are outcomes of radical breaks in the IT base, where components in the computing base are reassembled (Henderson and Clark 1990). For example, Internet computing was a disruptive innovation created by (Type 0) architectural change (TCP/IP-based tools and n-tier computing), which was made radical with the addition of browsers, data formatting standards, and software platforms (J2EE, .Net, etc.). This enabled the development of radically new services (Type II) which were demanded by faster speed (Type I) (Lyytinen and Rose 2003).

We can now investigate the extent to which changes in Type 0 innovation *can* lead to innovations in Type I, such as agile development, and the consequent *fast* adoption of Type II innovations (business agility). We conjecture that the agile innovation is produced by two capabilities: (1) the capability of software organizations to adopt Type 0 innovations and (2) their capability to successfully *transform* and *hone* these capabilities into Type I innovations. This is dependent on the mobilization of two related capacities. The first capability—*technology absorption*—reflects an organization's ability to sense, acquire, and absorb new base technologies through *exploration*. The second capability reflects a software organization's (1) ability to use new IT deployments for process improvement and (2) to effectively learn from such occasions in order to formalize process knowledge. This latter process we call *exploitation*. Successful software innovators need to effectively and continuously identify and match strategic opportunities for their process improvement with emerging technical capabilities.

2.2 Exploration and Exploitation

In the management literature, *exploration* and *exploitation* have been established as two fundamental responses to environmental challenge (March 1991). These archetypes help distinguish two distinct modes in which organizations compete and adapt, and how they organize, strategize, and execute. Through exploitation, organizations refine by trial-and-error learning their competencies through repeated actions over of time. Exploitation is about harnessing "old certainties" through refinement, implementation, efficiency, production, and selection. Exploration, in contrast, is about discovering new opportunities where organizations create new competences through search, discovery, experimentation, risk taking, and innovation (Henderson and Clark 1990; March 1991; Tushman and Anderson 1986).

Exploration requires substantially different structures, processes, strategies, capabilities, and culture (Tushman and Anderson 1986). Exploration leans toward organic structures, loose couplings, improvisation, chaos, and emergence. Exploitation deals with mechanistic structures, tight coupling, routinization, bureaucracy, and stability. Returns with exploration are uncertain, highly variable, and distant in time, while exploitation yields returns that are short term, have higher certainty and lower variance (March 1991). Due to their fundamental differences, exploration and exploitation pose a continuous tension for management (Levinthal and March 1993). These tensions create dysfunctional learning outcomes when either exploration or exploitation is preferred (March 1991). Trial-and-error learning can bias management to focus too much on current capabilities—at the expense of new opportunities—thus causing capacities to become core *rigidities*, and creating learning myopias and competency traps (Levinthal and March 1993; March 1991). In contrast, when organizations engage in excessive exploration, continued "failure leads to search and change, which lead failure which lead to even more search and so on" (Levinthal and March 1993, p. 98). Organizations' learning becomes chaotic: managers love to explore but fail to allocate resources to exploit their new competencies.

This invites us to understand how organizations learn to tack between exploration and exploitation and consequently change their resource bases through acquisition, integration, recombination, and the removal of capabilities (Eisenhardt and Martin 2000). In doing so, they must relentlessly integrate, reconfigure, gain, and release resources as a response to changes (D'Aveni 1994; Teece et al. 1997). Such dynamic capability embodies a learning related meta-capability by which software organizations learn to blend exploration and exploitation across different stages of IT innovation.

2.3 Exploration and Exploitation in Software Development Organizations

The general logic of exploration and exploitation during IT innovation stages is depicted in Figure 2. Exploration processes result in IT development firms adopting

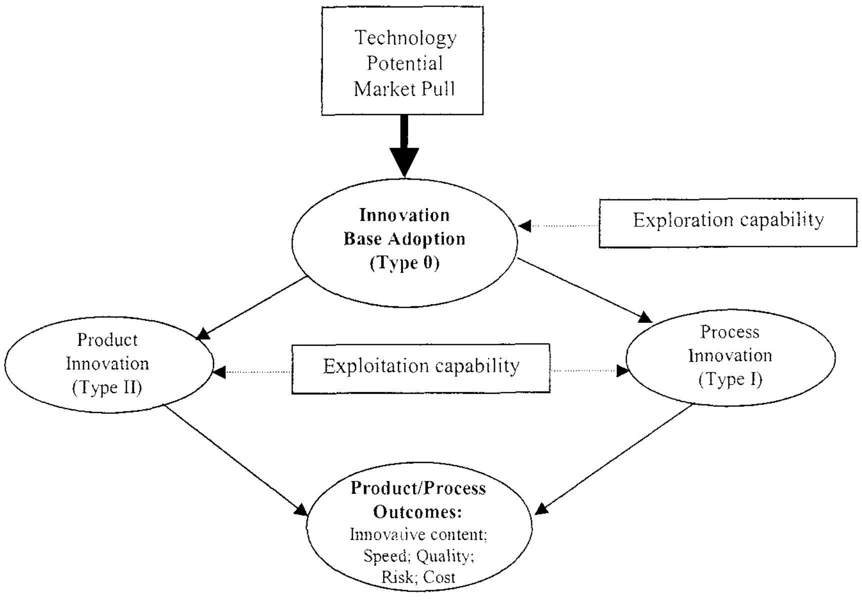

Figure 2. A General Model of IT Innovation as Exploration and Exploitation

Type 0 base innovations that lead to production of new Type II and Type I innovations (Lambe and Spekman 1997; Lyytinen and Rose 2003). An example of Type II innovations would be the organizations' ability to create a capability to produce totally new types of applications, while the innovation of Type I would be adopting new process technologies that help deliver the same software functionality in half of the time. Exploration agility as *absorptive capacity* (Cohen and Levinthal 1991) means two things: (1) the software organization must adopt new Type 0 and Type I technologies faster than its peers, and (2) it must use these technologies to develop Type II innovations (explorative process innovation) faster. If the organization is successful, this will change the organizations' innovations in its products (Type II innovations) and processes (Type I innovations). The more the former deviate from the current product mix, the more *innovative* and *agile* is product innovation. The more the latter deviates from the status quo, the more *innovative process* is instantiated—and the more agile is process change.

Software organizations need also to exploit when technologies mature by streamlining, standardizing, automating, and scaling up their processes for exploitation capability. This can be defined as the organizations' learning capability to improve and change their delivery processes over time in order to maximize process outcomes such as speed, quality, risk, or cost. Clearly, this learning mode is distinct from exploration and agility in exploitation can be viewed as lubricating a well-defined process.

Lambe and Spekman (1997) describe how exploration and exploitation are temporally organized across different phases of IT innovation (adapted for Figure 3). We later use this model to explore how each phase affects process features such as agility. Type 0 innovations can be regarded as offering general *technology push* to improve both software products and processes. Growth in the innovation base can lead to **radical** IT innovations (significant departures of existing behaviors and solutions) covering both development outcomes (new *kinds of* systems—i.e., product innovations) and development process (new *ways* of developing systems) that enable new innovative solutions and processes. Such explorations take place in short and intense periods during which hyper-competition and fast learning are valued.[3] When main features of the new product family have been fixed and become more or less standardized, organizations move to product exploitation by incrementally adding new features to the developed product platform. When such a stage is achieved (or sometimes when product explorations are being conducted), organizations move on to discover significant and radical ways to improve their product delivery processes. We call this stage *process exploration* or Type I radical innovation. Such innovations can include investments in better cross-product platforms or development of innovative process technologies (CASE tools, software libraries, collaborative tools). When the radical innovation potential in process improvements is mostly exhausted, organizations will move to what we call *process exploitation* or incremental Type I innovation.

[3]This is called *hyperlearning* in Lyytinen et al. (2004).

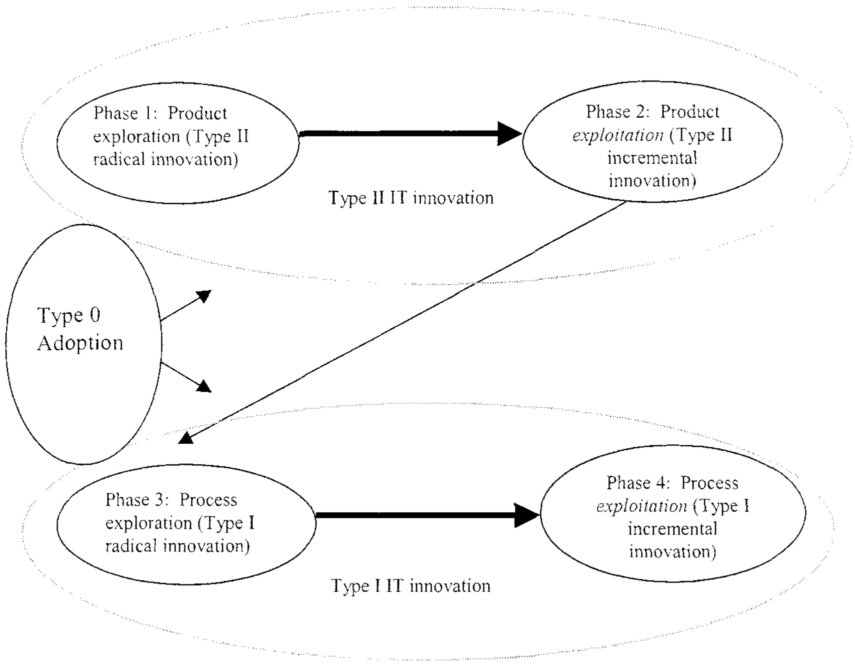
Phase 1: Product exploration (Type II radical innovation)
Phase 2: Product *exploitation* (Type II incremental innovation)
Type II IT innovation
Type 0 Adoption
Phase 3: Process exploration (Type I radical innovation)
Phase 4: Process *exploitation* (Type I incremental innovation)
Type I IT innovation

Figure 3. Organizing Logics for Exploration and Exploitation Across Different Types of IT Innovations

2.4 A Model of Process Features During Exploration and Exploitation

Relationships between process features (innovative content, speed, cost, quality, and risk) are complex. It is impossible to optimize them all simultaneously. Relationships between them vary depending on whether new Type II innovations are discovered or incremental Type I innovations are proposed. We model process goals as directed graphs where each process goal is depicted as a separate vector and its relative size shows to what extent this process feature is being maximized.[4] An illustration of such a graph for Phase 1 is shown in Figure 4. In Phase 1, software organizations maximize innovative content, they tolerate relatively high risks, expect relatively fast product development and medium cost, but do not expect high quality. To speed up exploration, their capability to deliver any workable solution may be slowed down. Likewise, if they want to be more nimble, they may have to paradoxically sacrifice their innovativeness.

[4]Van Kleijnen (1980) calls these Kiwiat graphs.

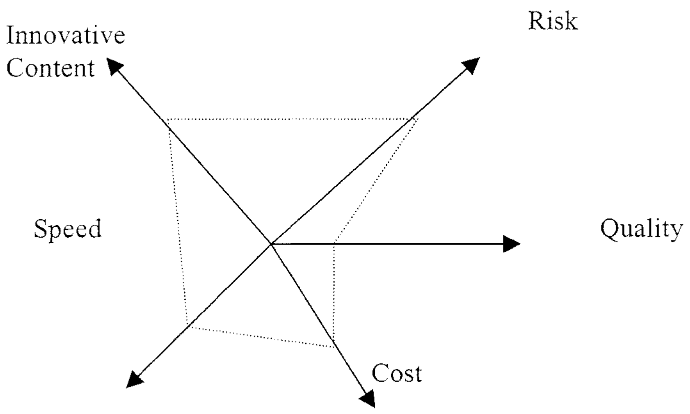

Figure 4. Desirable Process Features for Product Exploration

These features are causally related. During product exploration, we suggest the following relationships:

For **Innovative Content**[5]:

(1) + Innovative Content → + Risk (i.e., when innovative content increases risk increases)
(2) + Innovative Content → + Cost
(3) + Innovative Content → – Quality
(4) + Innovative Content → – Speed

If speed is a requirement, it must come at the expense of other outcomes, giving the following relationships:

For **Speed**:

(1) + Speed → + Risk
(2) + Speed → + Cost
(3) + Speed → – Quality
(4) + Speed → – Innovative Content

As can be seen during Phase 1, speed and innovation take precedence. However, both cannot be optimized simultaneously, and an increase in one counteracts the other.

[5]These causal dependencies were derived through content analysis from our interview data, which will be discussed in more detail in the next section.

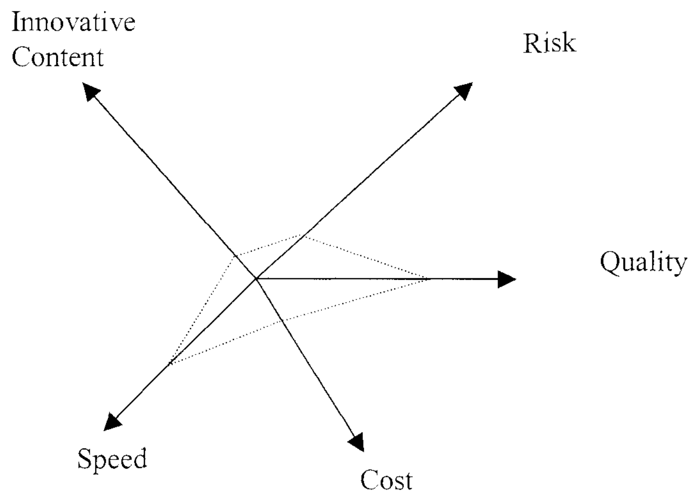

Figure 5. Process Features in Type I Incremental Innovation

In a similar fashion, we can model the process features for Phase 4[6] (Figure 5). Software delivery is faster as no effort is wasted to explore products or architectural solutions. The focus is on incremental innovations through economies of scale and scope where organizations maximize quality and speed while minimizing cost and risk by fixing product and process features. This has been assumed in the process improvement research (Humphrey 1989). The following dependencies can be observed:

(1) – Innovative Content → – Risk (i.e., when innovative content decreases risk decreases)
(2) – Innovative Content → – Cost
(3) – Innovative Content → + Quality
(4) – Innovative Content → + Speed

2.5 Some Implications for the Study of Agility

If an organization engages in radical Type II innovation, it will decrease its opportunity for incremental process innovation due to their contradicting logic. Likewise, increases in organizations' exploration will decrease their exploitation capability. Therefore, organizations that focus either on exploration or exploitation—although in

[6]We could similarly model the two other phases but for the brevity they are omitted here as they are not as distinct as the two extreme cases.

both modes they view agility as a desirable feature—have different mindsets about agility. During exploration, the desire to explore fast dominates, while during exploitation, the main focus is to remove friction from well-defined processes. However, the new technology (Type 0 innovation) *per se* can dramatically increase the speed by offering higher granularity (e.g., ERP parameterization), powerful abstraction mechanisms (e.g., Web services), standardized functionalities (e.g., browsers), or architectural integration mechanisms (e.g., architectural patterns). Improvements here can be dramatic and as important as radical innovations in products. When an organization shifts its focus away from radical exploration, it must increase its exploitation by fixing the product and, later, the process. It must change process measures as its focus is now on efficiency, economies of scale, and quality. This shift leads to increased trial-and-error learning (March 1991).

Software organizations need thus to innovate in a lumpy manner by balancing trade-offs between innovative content, cost, speed, quality, and risk. Over time they must exploit technologies, organize, and control in contradicting ways. Therefore, IT innovations will be appropriated through multiple innovation paths. As the contrast between early exploration and late exploitation is stark, organizations can only entertain a certain amount of transformations over a time period. They increase their innovative agility first by adopting radically new technologies (Type 0), but later shift their focus on exploitation by stabilizing product features. At the same time, they engage in other exploration–exploitation cycles, thus organizing in an ambidextrous manner (Tushman and Anderson 1986). The impacts of this stepwise transformation on process features (innovative content, quality, risk, and cost) are significant, and organizations locate themselves into alternative regions with different idea configurations of process features (see Table 1).

The first contingency presented in Table 1 is rare and can be mostly observed in bureaucratic environments. For R&D software development (pre-competitive phase),

Table 1. Contingencies for Organizational Learning in Software Development

Exploration Focus/ Exploitation Focus	**Low**	**High**
Low	Normally natural monopoly: Little impact on any process features	Pre-competitive product development: Innovative content dominates, other features tangential *Internet computing around 1993-1997*
High	Process competition in established markets: Incremental changes in speed, efficiency focus in reducing risk, quality *Internet computing 2001–*	Hypercompetition: Fluid technology and markets, speed dominates, necessary to meet minimal process/product features *Internet computing 1996–2001*

only exploration focus is high. When both exploration and exploitation are high (i.e., organizations are fast oscillating between two phases of product innovation in Figure 3), this can be regarded as hyperlearning–hypercompetition as has been observed in software development organizations between the years 1997 and 2000 (Lyytinen et al. 2004). The push toward higher exploitation comes normally from competitive demands created by the growing market size, stiffer competition, and new value propositions. The organization tilts toward process improvement and starts to compete based on process integration. Agility in software thus relates to capability to be a fast explorer or to be an effective integrator. The jump between these positions takes place when organizations recognize that the emerging technology has become mainstream and they must decide whether they will keep their focus on markets that value exploration or specialize on exploitation and start to manage process features such as quality and cost.

3 RESEARCH METHOD AND RESEARCH SITES

3.1 Research Goals and Design

We wanted to explore the following questions: Do perceptions of and need for agility change during different phases of IT innovation? How software organizations manage contradicting demands of exploration and organize their innovation for agility? Does the IT innovation model predict how agility relates to other process features? To address these questions, we conducted a 5-year longitudinal field study (Yin 1994) in Web development software companies (see Table 2). We chose multisite case study as it allowed a replication by which we could test emerging theoretical insights and triangulate both theory and data (Eisenhardt 1989). To minimize bias, we sought to maximize the variations in order to improve external validity (Yin 1994). Companies had different sizes and operated in many industries. They had experience using Web-based technologies in several domains. The geographical scope of their operations varied, as some were local while others were part of global companies. The firms also had large variation in their development experience, ranging from as few as 4 years to 40+ years.

3.2 Data Collection and Analysis

The data were gathered between June 2000 and April 2003 at three different time points (2000, 2001, 2003-2004). The exact times of data collection are shown in Table 3. For all companies, the data is not complete due to mortality (some of the companies went out of the business or were bought or sold). For some data we had problems with poor tape quality and were unable to transcribe them verbatim so only collected the main facts. We organized the data into three different temporal periods—pre-2000 (Period 1), 2000-2001 (Period 2), and 2002-2004 (Period 3)—that align with the different stages of the dot-com boom. Here pre-2000 stands for market growth and period of fast innovation, Period 2000-2001 stands for the recession and crisis, and 2002-2004 stands for the recovery.

Table 2. Firm Characteristics

FIRM	Firm 1	Firm 2	Firm 3	Firm 4	Firm 5	Firm 6	Firm 7
Division Focus	Custom e-business applications.	B2B e-Business consulting.	Spin-off. Web-based ASP for parent company's customers.	E-Business consulting in mobile computing.	Systems integrators to upgrade legacy systems with Web and mobile solutions.	E-Business solutions in mobile computing. Assembling of components and applications.	Consulting, development, product, networking and hosting services.
History	15 year old mainframe and client server shop 500 employees in 4 locations.	Part of a large, multinational consulting company.	Part of a large financial company with several thousand employees.	Multinational e-business consulting. Founded in 1995 with several thousand employees	e-Commerce development firm. Founded in 1996.	Large multinational e-business consulting and software development firm.	Mature, large, multinational development and IT service firm.
Number of Employees in Division	Several hundred	Several hundred	70	100+	200+	700+	Several hundred
Typical work week	40 hours	50 hours	50 hours	60 hours	37.5 hours	Varies	37.5 hours
Employee turnover per year	18-30%	15-30%	< 10%	3%	3%	Uncertain	Uncertain
Organizational Structure	• President • Branch manager • Field manager • Project manager	• Partner • Director • Project and technical managers	CIO, then flat	• Client manager • Project manager	Entirely flat except for salary issues.	Rigid vertical hierarchy with formalized methodologies for all aspects of business.	Company is divided into autonomous units based on market sector of client.
Project Team Characteristics	15-20 people including business analysts, architects, lead developer, other developers, QA person	Architects, analysts, expert developers, rookie developers	Informal	Flat with the following roles: project assistant, technical lead, designer, information architect	Informal	Rigid vertical hierarchy.	Broken down by customers (approximately 50/ customer) and subsequently by teams (of 10 each).

Table 3. Data Collection Summary

FIRM	Firm 1	Firm 2	Firm 3	Firm 4	Firm 5	Firm 6	Firm 7
Interview Date 1	**June 2000**	**June 2000**	**June 2000**	**October 2000**	**September 2000**	**September 2000**	**November 2000**
Interviewees in Time 1	Six senior employees including an executive, managers, and software architects.	A senior manager of an development group and one of his key developers.	The CIO, and five key senior technologists were responsible for the creation of the spin-off.	Five senior employees including ISD project managers, developers, and the senior technology architect.	One of the founding executives who was responsible for development of business processes.	One senior manager of IT development services.	Four senior employees including a systems architect, manager, and software engineer.
Interview Date 2	**October 2001**	**October 2001**	**October 2001**	**August 2001**	**August 2001**	**August 2001**	**August 2001**
Interviewees in Time 2	One software architect from first interview.	A senior manager of an IS development group and one of his key developers from first interview.	Two technologists from first interview.	Two employees from first interview.	Same interviewee from first interview.	Manager who replaced manager in first interview.	One manager from first interview.
Interview Date 3	**March 2003**	**March 2003**	**No interview**	**No interview**	**March 2003**	**April 2003**	**April 2003**
Interviewees in Time 3	An architect from Time 1, the replacement of executive in Time 1, and a developer not in Time 1.	Developer from Time 1.	Firm absorbed by parent company and IT employees reassigned. Interviews not available. Information gathered via e-mail with one of the original interviewees and review of online documentation of parent firm in March 2003.	Finnish office closed. Interviewees not available.	Same interviewee from first interview.	Manager who replaced manager in first interview.	One manager from first interview.

The data were obtained through semi-structured interviews with senior management and senior developers who managed the organizational knowledge bases and skills needed to execute the technology and business strategy. We also examined the archives of company documents, including systems development documentation and technology strategies and made notes. A range of one to six individuals participated from each company. A total of 19 interviews were conducted with a typical interview time of approximately 2 hours. The transcribed data currently covers about 700 pages of interviews. Specifically, we asked the firms to clarify the extent, scope, depth, and speed of change in their software development during the Web development adoption.

Data analysis was done using the inductive method (Yin 1994). The transcripts of each company for each period were subject to a within-case analysis that involved repeatedly reading the transcript and taking thorough notes about the firms' perceptions of agility, its antecedents and resulting process outcomes. After each individual case, we began cross-case comparisons that involved listing the similarities and differences among the firms in their process outcomes at each period of time. Two researchers coded the transcripts individually. Coding was compared for inter-coder reliability and differences in interpretation were identified and discussed until consensus could be found. Data codes within cases were then converted into tabular form and again analyzed by both researchers to confirm findings within and across cases and to identify any gaps or contradictions in the original models identified. Any discrepancies or contradictions were scrutinized and the original transcripts revisited for clarification. Tables were iteratively modified until both researchers were satisfied with the validity of the findings. Once the model was formally developed, a summary was written and presented for external review by participants of the study. Phone interviews were conducted with individuals from three different firms that had participated in the longitudinal study. For each of the three follow-up interviews, the models identified in the analyses were confirmed.

4 RESEARCH FINDINGS

4.1 Changes in Agility During Exploration and Exploitation

Table 1 and Figure 3 show a movement from product exploration to process exploitation. A related summary of organizational change in organizations through periods 0-3 is given in Tables 4 and 5. Overall, the tables show that the firms organized their perceptions of agility and concerns for exploration and exploitation as recommended by the model. Each firm in the early stages of Internet computing (e.g., the periods between 1995 and the first interview) were engaged in radical innovation product when compared to Period 1.

Six of the seven firms created their own product innovations and before the first interview time were regarded as radical product innovators (Phase 1). They then moved sequentially to Phase 4. One firm (Firm 6) in our data set did not conduct their own product innovation at all. Instead, it formed alliances with other radical product innovators (thus outsourcing that activity) and focused all of its time on exploitative process innovations. It sought to deploy its existing product bases quickly and thus was already

Table 5. Innovation Summary: Finnish Firms

FIRM	Firm 4	Firm 5	Firm 6	Firm 7
Time 0	**Phase 1: Product Exploration Phase**	**Phase 1: Product Exploration Phase**	**Phase 3: Process Exploration Phase**	**Phase 1: Product Exploration Phase**
Time 1	**Phase 2: Product *Exploitation* Phase *then* Phase 3: Process Exploration Phase.** Recognized the need for speed. Believed a methodology would help. Began work on formalization of process at end of Time 1.	**Phase 2: Product *Exploitation* Phase.** Indicated there is a high risk associated with speed. Used light methods with unknowable outcomes (because of less quality testing, less needs analysis) & limited scope of product innovations.	**Phase 4: Process *Exploitation* Phase.** Limits to a fixed set of product innovations. Process innovations early. Reuse of components, affiliation with partners, and reusable methodologies. Higher quality, lower costs, and low risks because of fixed solutions.	**Phase 2: Product *Exploitation* Phase.** Speed increases; risk increases; costs increase. Technology change is inherently faster. Certain amount of freezing in product innovation had begun and allowed increased speed via reuse.
Time 2	**Phase 4: Process *Exploitation* Phase *then* a new Phase 1: Product Exploration Phase .** Methodologies implemented were for projects of Time 1 type. Time 2 projects were radically different and methods were inappropriate. Cost rose and speed declined as a result of product exploration.	**Phase 3: Process Exploration Phase.** Moved to incremental product innovation. Return of methodologies.	**Phase 4: Process *Exploitation* Phase.** Nothing different from Time 1 except even more focused on exploitation of product and process.	**Phase 3: Process Exploration Phase.** Moved to incremental product innovation stage. Coincides with return of methodologies.
Time 3	Closed Finnish office.	**Phase 4: Process *Exploitation* Phase *then* a new Phase 1: Product Exploration Phase .** Speed slower again as product innovation becomes more radical. Problem when encounter changes in Type 0 innovations. Counteracted with process innovations.	**Phase 4: Process *Exploitation* Phase.** Nothing different from Time 1 except perhaps more focused on exploitation of process.	**Phase 4: Process *Exploitation* Phase.** Focused on reuse to increase speed and decrease risk. Freezing product innovation allows process innovation.

in Phase 3 at Period 0. It achieved this at the cost of radical innovation. Not surprisingly, by Period 1, Firm 6 was already engaged in process exploitation (Phase 4).

While each firm moved eventually to Phase 4, some of them moved beyond Phase 4 (or back) to a new Phase 1, thus demonstrating ambidexterity. These organizations found that they could not be successful in engaging solely in process exploitation. In two cases (Firms 1 and 5) we observed that new product innovations made their previous process innovations less effective. These firms experienced their process agility decreasing and they needed to reevaluate tradeoffs between speed and other features. Likewise, Firm 4 found that it now incurred higher costs and slower speed. The firm found this by Period 2 and subsequently went out of business as a result of declining market demands and having the wrong capability.

4.2 Impact on Process Features and Speed

Figures 4 and 5 highlight critical interrelationships between ISD process features at different phases of innovation. Accordingly, organizations have to control interrelated and contradictory process features: speed, innovation, cost, risks, and quality. Among the data set (all 19 interviews), we found strong evidence that managers heeded these five factors (Tables 6, 7, 8, and 9).

We also found strong evidence for the types of dependencies as noted in Figures 4 and 5. Specifically, we found that organizations increased speed in innovation in Period 1, but faced a tradeoff of increased risk, increased cost, and decreased quality (Tables 6 and 7).

Likewise, Firm 7 noted, ***"you have less time to think and you don't have the time to think of everything."*** The dominating process feature in Period 1 was *innovation content*. We also observed that speed and innovation were inversely related. Again, in most (16 of 19) interviews, evidence was found for this inverse relationship (as can be seen in bold in Tables 8 and 9). For example, Firm 3 finished their proof of concept stage and subsequently stopped radical product innovation. As a result of moving to incremental innovation in Period 1, they were able to formalize a methodology for ***"rapid software development and rapid implementations that we have to do."*** Similarly, Firm 2 attributed increased speed in Period 3 to the shift to incremental innovation. Specifically, increased speed was a function of stabilization in ***"methodology [PROCESS], a function of increased skill sets [BASE], and a function of using packaged product type solutions [PRODUCT]."***

In addition, the other relationships between innovation and risks, cost, and quality were observed (Tables 8 and 9). For example, in Period 1, a member of Firm 7 referred to the period before Internet development as ***"the good old days"*** and noted that lower risks were ***"old fashioned."*** Similarly, Firm 5 noted, when it began adopting radical Type 0 innovations for creating product innovations in Period 3, that development was slower, more resources were needed, and quality declined. Overall, the interrelationships of the five goals in Figures 4 and 5 were supported. With regard to phases, the primary relationships between Figures 4 and 5 were also supported when a firm is involved in product exploration (Phase 1), or in the process exploitation phase (Phase 4). As can be seen in Tables 4 and 5, during earlier phases, quality was lower and risks

Table 8. Tradeoffs of Innovation versus Speed, Quality, Risk, or Cost Summary: United States Firms

FIRM	Firm 1	Firm 2	Firm 3
Time 1	Evidence that more radical innovation increases **risk**	Evidence that more radical innovation increases **risk**	Evidence that if you freeze innovation, speed increases Evidence that more radical innovation increases **risk**
Time 2	Evidence that if you freeze innovation, speed increases	Evidence that if you freeze innovation, speed increases	Evidence that if you freeze innovation, speed increases Evidence that if you freeze innovation, **cost** decreases Evidence that stopping radical innovation allows improved **quality** deliverables
Time 3	Evidence that if you freeze innovation, speed increases	Evidence that if you freeze innovation, speed increases Evidence that stopping radical innovation allows improved **quality** deliverables	

Table 9. Tradeoff of Innovation versus Speed, Quality, Risk, or Cost Summary: Finnish Firms

FIRM	**Firm 4**	**Firm 5**	**Firm 6**	**Firm 7**
Time 1	Evidence that if you freeze innovation, speed increases Evidence that more radical innovation increases **risk**	Evidence that if you freeze innovation, speed increases Evidence that more radical innovation increases **risk**	Evidence that if you freeze innovation, speed increases Evidence that moving to incremental innovation improves quality Evidence that incremental innovation decreases **risk**	Evidence that if you freeze innovation, speed increases Evidence that more radical innovation increases **risk**
Time 2		Evidence that if you freeze innovation, speed increases Evidence that stopping radical innovation allows improved quality deliverables	Evidence that if you freeze innovation, speed increases Evidence that moving to incremental innovation improves **quality**	Evidence that if you freeze innovation, speed increases
Time 3		Evidence that if you freeze innovation, speed increases Evidence that if you freeze innovation, **costs** decrease Evidence that moving to incremental innovation improves **quality**	Evidence that if you freeze innovation, speed increases Evidence that moving to incremental innovation improves **quality**	Evidence that if you freeze innovation, speed increases

and costs were higher. In later phases, the opposite was true, although in all phases, speed was deemed important. As such, the concern for speed did not diminish between phases, as the idea of speed was different.

The tradeoffs between innovative content and the other factors are most visible when Period 1 is considered. In Period 1, Firm 6 was already in Phase 3. They were already reaping the rewards of this and noted that their quality was higher, costs were lower, and risks were lower as they had frozen innovation and assembled ***"components"*** for ***"a set of solutions that [they knew] how to give and [could] give them quickly."*** In contrast, other firms, while moving to Phase 2, saw increased risks and costs, with decreased quality.

As each firm moved into other phases, its market matured and stabilized. Their methodologies became refined while their risk, costs, and quality moved to a new trade-off pattern (Figure 5). For example, Firm 2 entered Phase 4 during Period 3. The interviewee noted that their ***"methodologies and strategies are now mature"*** and that quality was improved as ***"a function of better trained people, a methodology...and less innovation."***

4.3 Discussion and Conclusions

Software agility is affected by the scope and depth of innovative activity in base technologies as well as in continued process innovations in complementary assets. We explored the concept of agility in terms of the following questions: Do perceptions of and need for agility change during different phases of IT innovation? How do software organizations manage contradicting demands of exploration and organize their innovation for agility? Does the IT innovation model predict how agility relates to other process features? We observed the following: (1) concern for both exploration speed and process development speed changed significantly over the period of study, (2) software organizations tended to organize themselves differently during different innovation periods while they decide either to explore fast or deliver fast (process integrators), and (3) the variance in process features emphasized varied across phases and also between companies due to the varying focus on exploration or exploitation. Software organizations controlled their concern for agility in how good they wanted to become in managing technologies during different innovation phases. In doing so, they had to trade agility against other criteria including innovative content or risk. How these trade-offs were made depended on competencies, managerial focus, and competitive demands.

There are several avenues for future research in this fascinating area. First we need to generalize the findings here with a better and more representative sample of organizations. There is also a need to develop more careful constructs for agility and other process features. We need to explore other factors than just the organizations' learning focus to establish causal explanations of agility in organizational contexts. Finally, it needs to be seen if these findings are generalizable beyond Internet computing, and if so, when and where.

REFERENCES

Agile Alliance. "Manifesto for Agile Software," 2001 (available online at http://www.agilemanifesto.org/; accessed September 4, 2004).

Baskerville, R., Levine, L., Heje, J-P, Balasubramarian, R., and Slaughter, S. "How Internet Software Companies Negotiate Quality," *IEEE Software*, May 2001, pp. 51-57.

Carstensen, P, and Vogelsang, L. "Design of Web-Based Information Systems: New Challenges for System Development," in *Proceedings of the 9th ECIS*, Bled, Slovenia, June 27-29, 2001, pp. 536-547.

Cohen, W. M., and Levinthal, D. A. "Absorptive Capacity: A New Perspective on Learning and Innovation," *Administrative Science Quarterly* (35), 1990, pp. 128-152.

Cusumano M., and Yoffie, D. "Software development on Internet Time," *IEEE Computer* (32:10), 1999, pp. 60-69.

D'Aveni, R. A. *Hypercompetition: Managing the Dynamics of Strategic Maneuvering*, New York: The Free Press, 1994.

Eisenhardt, K. M. "Building Theories from Case Study Research," *Academy of Management Review* (14:4), 1989, pp. 532-550.

Eisenhardt, K. M., and Martin, J. A. "Dynamic Capabilities: What Are They?," *Strategic Management Journal* (21), 2000, pp. 1105-1121.

Henderson, R. M., and Clark, K. B. "Architectural Innovation: The Reconfiguration of Existing Product Technologies and the Failure of Established Firms," *Administrative Science Quarterly* (35:1), 1990, pp. 9-30.

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

Kessler E., and Chakrabarti, A. "Innovation Speed: A Conceptual Model of Context, Antecedents and Outcomes," *Academy of Management Review* (21:4), 1996, pp. 1143-1191.

Lambe C., and Spekman, R. "Alliances, External Technology Acquisition, and Discontinuous Technological Change," *Journal of Product Innovation Management* (14), 1997, pp. 102-116.

Levinthal, D., and March, J. "The Myopia of Learning," *Strategic Management Journal* (14), 1993, pp. 95-112.

Lyytinen, K., and Rose, G. "The Disruptive Nature of Information Technology Innovations: The Case of Internet Computing in Systems Development Organizations," *MIS Quarterly* (27:4), 2003, pp. 557-595.

Lyytinen, K., Rose, G., and Yoo, Y. "Exploring and Exploiting in High Gear: Hyper-learning in Seven Software Firms," under review, 2004.

March, J. G. "Exploration and Exploitation in Organizational Learning," *Organization Science* (2:1), 1991, pp. 71-87.

Pressman, R. "Can Internet-Based Applications Be Engineered?," *IEEE Software* (15:5), September-October 1998, pp. 104-110.

Swanson, E. B. "Information Systems Innovation Among Organizations," *Management Science* (40:9), 1994, pp. 1069-1088.

Teece, D. J., Pisano, G., and Shuen, A. "Dynamic Capabilities and Strategic Management," *Strategic Management Journal* (18:7), 1997, pp. 509-533.

Tushman, M. L., and Anderson, P. "Technological Discontinuities and Organizational Environments," *Administrative Science Quarterly* (31), 1986, pp. 439-465.

Van Kleijnen, J. *Computer and Profits: Quantifying Financial Benefits of Information Systems*, Englewood Cliffs, NJ: Prentice-Hall, 1980.

Yin, R. K. *Case Study Research: Design and Methods*, Thousand Oaks, CA: Sage, 1994.

ABOUT THE AUTHORS

Kalle Lyytinen is Iris S. Wolstein Professor at Case Western Reserve University. He currently serves on the editorial boards of several leading information systems journals including *Journal of the AIS* (Editor-in-Chief), *Journal of Strategic Information Systems, Information & Organization, Requirements Engineering Journal, Information Systems Journal, Scandinavian Journal of Information Systems,* and *Information Technology and People.* He is the former chairperson of IFIP WG 8.2 and is a member of WG 8.6. He has published over 150 scientific articles and conference papers and edited or written eight books on topics related to system design, method engineering, implementation, software risk assessment, computer-assisted cooperative work, standardization, and ubiquitous computing. He is currently involved in research projects that look at the IT-induced innovations in the software development, architecture, and construction industries, the design and use of ubiquitous applications in health care, high-level requirements models for large-scale systems, and the development and adoption of broadband wireless standards and services, where his recent studies have focused on South Korea and the United States. Kalle can be reached atkalle@po.cwru.edu.

Gregory M. Rose is an assistant professor at Washington State University. He received his Ph.D. in the CIS Department at Georgia State University, an MBA from Binghamton University, and a B.S. in business administration from the University of Vermont. Gregory has more than 20 publications including those in journals such as *MIS Quarterly, IEEE Transactions on Engineering Management, Accounting, Management and Information Technologies, Information Systems Journal, Journal of Global Information Management, Psychology and Marketing,* and *Communications of the AIS.* A 1998 ICIS Doctoral Consortium fellow, he has won multiple teaching awards, a post-doctoral fellowship from the University of Jyväskylä (Finland), and was an invited scholar at the University of Pretoria (South Africa). He is currently working on research projects involving electronic commerce, innovation theory, organizational learning, and global issues in IT. He also serves on the editorial board of *Journal of Global Information Management.* Prior to entering the doctoral program at Georgia State, he worked as a systems integrator. Greg can be reached at grose@wsu.edu.