4-11-2008

How Agile is Agile Enough? Towards A Theory of Agility in Software Development

Kalle Lyytinen
Case Western Reserve University, kalle@case.edu

Gregory M. Rose
Washington State University

芽|Sprouts Working Papers on Information Systems ISSN 1535-6078

How Agile is Agile Enough? Towards A Theory of Agility in Software Development

Kalle Lyytinen
Case Western Reserve University, USA

Gregory M. Rose
Washington State University, USA

Abstract
One poorly investigated issue in organizational agility is the question how organizations change their speed while adopting and exploiting new IT capability. In this paper we outline a theory of software development agility that draws upon a model of IT innovations by Swanson and on Marchâ s learning theory and in particular on his concepts of exploration and exploitation. We explore how both exploration and exploitation as organizational learning modes can software development agility. We propose a sequential model of organizational learning in which agility is driven by different factors during different stages â exploration vs. exploitation- of organizational learning. We show that software development agility is influenced by the external demands, the diffusion level and rate of the IT innovation, its radicalness, and the organizationsâ needs to balance multiple conflicting process goals including speed, quality, cost, risk and innovative content. We illustrate the value of the model by exploring how seven software organizations controlled the demands for increased agility i.e. their development speed or over a period of five years (1999-2004), and how they balanced the need for the increased agility with other critical development criteria like cost, risk, quality and innovative content. In conclusion, we discuss the implications of our findings for future research on agility and related management practices.

Keywords: Agility, IT innovation, Radical nature, Exploration, Exploitation, Ambidexterity, Process features

How Agile is Agile Enough?
Towards A Theory of Agility in Software Development

Introduction

Agility can be defined as the quality of being quick-moving and nimble. In software development *agility* can be defined as the ability of a system developer to sense and respond to new technical and business opportunities in order to stay innovative and competitive in a turbulent and quickly changing business environment. An *agile* software development organization (one that demonstrates *agility*) has thus the capabilities and processes in place to respond to unexpected environmental changes both in the technology and in the business environment.

In the past the software and system development literature has mainly sought to control and explain the outcome quality / reliability of software processes. The main driver in this endeavor was to submit to virtues of good system engineering: the final technical system must be flawless, user friendly, scalable or portable. This logic pervaded debates around "software crisis" in the early 70's and has since then motivated the development of rigorous software development approaches including structured programming, and structured methodologies (Lyytinen 1987). The downside of these approaches is that they incur high coordination, documentation and learning costs. The same logic motivated later on much of the process improvement research in which the low level of errors and process repeatability were the driving goals (Humphrey 1989, Curtis et al 1992). Most researchers in these camps approached development speed with an assumption that it was relatively constant while the outcome quality mattered. Therefore, agility was not a goal to aspire in software development.[1]

This model faced significant challenge when the rebels of the new economy and skunk works of web development in the 90's changed the idea of "good" system development: software had to be developed in markets and for the markets with extremely fast pace. "Internet speed" became *mot dú jour* (see .e.g., Cusumano and Yoffie 1999, Lyytinen and Rose 2003a, 2003b, Baskerville et al. 2001, Pressman 1998, Carstensen and Vogelsang 1999). For the first time the focus in software development was on agility- speed truly mattered. This was viewed to be a necessity in order to harness the disruptive potential of Internet computing (Lyytinen and Rose 2003a). A great portion of process research in software development during the last five years assumes that speed is desirable (see e.g. Cusumano and Yoffie D. 1999, Turk et al 2004, Henderson-Sellars and Serour 2004): the key to competitiveness as it enables to compete in time. Moreover, such focus on agility echoes well with pivotal research in strategy on hyper-competition and dynamic capability (Cohen and Levinthal 1990, Eisenhardt and Martin 2000, Eisenhardt and Tabrizi 1995, D'aveni 1994) as well as studies in rapid product development (Menon et al 2002, Kessler and Bierly 2002, Kessler and Chakrabarti 1999, Kessler and Chakrabarti 1996).

Though the connection of increased speed to stronger competitiveness appears to be clear during hyper-competition (D'Aveni 1994) it is not clear what agility in software development truly means: is it the speed at which some type of running system is available for evaluation?; or is it the change in ratio between delivered functionality (in LOC, function points etc) and the elapsed time?, or is it the increased velocity of the client to adopt the software? It is clear that all these speeds are distinct aspects of "agility" in relation to

[1] At the same time the research on programmer productivity and even team productivity showed significant variation in individual and team productivity that was due to individual differences and team organization.

software. They also have quite different ramifications for how to increase agility. Another set of research issues relate to the organizational and technological antecedents of agility, and to what extent the organization can truly manipulate them. For example, there is a huge difference in changing the speed in doing X faster when compared to changing the speed in which the organization moves from doing X to doing Z. Researchers and managers alike must also understand how agility (if and when being manipulated) relates to process outcomes like *innovative content*, *risk*, *quality* and *cost* (e.g. Baskerville et al 2001). Finally, there is a dearth of knowledge how agility varies during technology diffusion and maturation. For example, much of the hype around internet speed was explained by the revolutionary technological capability- increased agility followed as the new technology enabled to do things faster. Yet, it remains unclear are such relative changes in speed sustainable, or will our perception of agility change just when the technology capability changes dramatically (Lambe and Spekman 1997).

In this paper our goal is to address some of these issues. We will develop a model that seeks to explain differences in types of *agility* that software organizations seek to achieve at different stages of technology diffusion. We show that the need to agility must be balanced in relation to other desirable process features like innovative content, risk, quality and cost. We will seek to validate the developed model by analyzing a longitudinal data set (5 year period) describing software development practices and outcomes in seven software development organizations that adopted Internet computing[2]. Findings show that studied organizations changed their perceptions of agility and their need for it as they sought to balance agility in relation to innovative content, cost, quality and risk. The remainder of the paper is organized as follows. Section 2 formulates the development agility model and reviews the related literature in software agility research and organizational learning. Section 3 describes the field study, while section 4 reports main findings of the longitudinal field study. The paper will conclude by noting remaining research challenges and discussing managerial implications.

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 relates to other process factors and how it can be manipulated by software organizations. The model draws on Swanson's model of IT innovation (Swanson 1994, Lyytinen and Rose 2003b) and March's exploration / exploitation dichotomy (March 1991) According to the model software organizations engage both in exploration and exploitation while innovating with IT. During periods of fast technological transition (e.g. shift to Internet computing) the exploration speed (absorptive capability of technical potential) and development speed (fast exploitation) must be combined to harness the new technology. Yet, exploration and exploitation set up contradicting demands for agility. Before embarking to develop the model we will shortly review the current state of the art in agile software development.

Research on Development Speed and Agility

During the current years we have seen the advent of a significant stream of research on agile software development methods advocated under such acronyms as Extreme

[2] The concept of Internet computing involves a relatively broad and evolving set of distributed computing models and solutions that rely on open, ubiquitous networks and associated sets of protocols and services. It draws upon models of computing that operate within open, heterogeneous, and distributed computing environments. (see Lyytinen and Rose 2003a, 2003b)

Programming, SCRUM, adaptive software development, or Agile Unified process among others (Turk et al 2004, Henderson-Sellars 2004). These methods assume that software processes can be organized for faster and more economic delivery of high quality software that meets the needs of the customer. The main focus here is on adopting process and organizational changes that improve communication, decrease administrative overhead, and submit to early and consistent focus on developing code by a team of software developers often organized in pairs (Agile alliance 2002). Empirical studies show that software processes gain agility in two ways: 1) developers shorten times between systems releases and get the running system up fast (e.g. Extreme programming Turk et al. 2004), and 2) they deliver faster the same functionality (Baskerville et al. 2001; Lyytinen and Rose 2003a). There are however, few studies which explore how faster product releases and faster development times affect, or are influenced by other types of agility in software development organizations created by dynamic capability (see Lyytinen et al 2004).

Though agile method studies offer useful guidelines how to increase software development speeds and increase its economic impact (i.e. the software meets better customers needs) these analyses are limited in explicating how increased process speed and changes in organizational and technological environments are related. In most studies this relationship looks like a form technological determinism: because we have better and more flexible technology this will automatically result in faster development (Carsten and Vogelsang 1999). Sometimes claims are made that new markets demand faster development irrespective of the technological environment (Cusumano and Yoffie 1999). As a result we currently have poor theoretical explanations how software agility has changed over time, and how varying speeds in different processes relate to technological / environmental change. For example to what extent observed changes are strategic choices made by the management to adapt to a market niche / competitive ecology? To what extent the speed is an outcome of better technological capability? In order to analyze these issues we need to carefully analyze how IT drives innovation in software development and how such innovation relates to agility. To this end we will next formulate fragments of a theory of software development agility. It draws on the concept of IT innovation and March's model of organizational learning.

Model of IS Innovation

In the IS field, the concept of IT innovation is poorly developed despite a huge literature on IT **based** innovation (Swanson 1994. Lyytinen and Rose 2003b). One reason for this is the inherent difficulty IS scholars face in addressing what innovation means in the IS field. Currently, it means many things to many men. IT innovation has multiple sources and a such broad scope that covers a broad range of activities in the IT value chain (Swanson 1994). As a consequence, innovation within system development (like agility) is not a singular event, but subsumes a causal chain of events along the value chain which portray significant departures from existing practices. An IS innovation must often traverse through a complex ecology of multiple types of innovative events (Figure 1) (Swanson 1994, Grover et al 1997, Lyytinen and Rose 2003a,b).

Figure 1 shows three primary value adding activities in the IT domain: 1) *creation of IT base technologies* like operating systems, middleware, databases systems etc. by vendors and manufacturers (e.g. Intel, Microsoft, IBM). This is called here base innovation (*Type 0 innovation*), 2) *development of processes, technologies and organizational arrangements by software developers* that enable better or more reliable delivery of software systems in organizational contexts (called *type I innovation*), and 3) and *development and adoption of new types of IT solutions* by IT deploying companies (e.g. Amazon.com). This is called *type II innovation.* Arrows in Figure 1 show thus how downstream organizations adopt innovations produced and innovated by companies in the upstream as to increase their overall scope and quality of IT deployment.

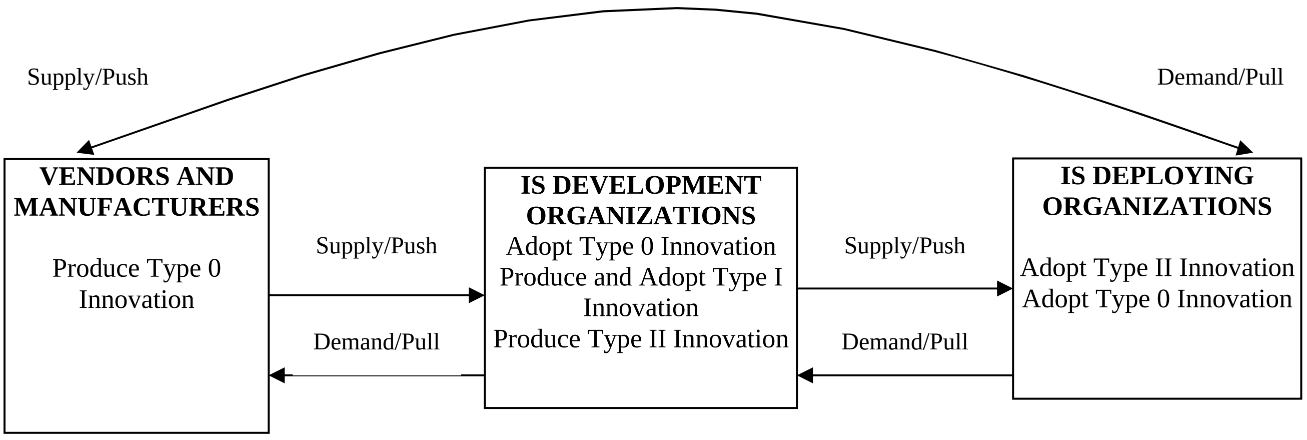
Supply/Push
Demand/Pull
VENDORS AND MANUFACTURERS
Produce Type 0 Innovation
Supply/Push
Demand/Pull
IS DEVELOPMENT ORGANIZATIONS
Adopt Type 0 Innovation
Produce and Adopt Type I Innovation
Produce Type II Innovation
Supply/Push
Demand/Pull
IS DEPLOYING ORGANIZATIONS
Adopt Type II Innovation
Adopt Type 0 Innovation

Figure 1. IT value chain and realms of IS innovation

IT innovation in light of this model means multiple things (Lyytinen and Rose 2003): breakthroughs in computing capability and architectures (Type 0 innovation), departures in current ways to develop and design computing applications (Type I innovation), or novel applications and/or ways of applying them (Type II innovation). For example, most of the past software agility research has focused on formulating and adopting type I innovations as a result of demand for increased speed from IS deploying organizations (pull) and availability of new technological capability (type 0 innovations) (push). This connection is not causal in a sense that many innovation in type II do not necessarily affect other parts of the value chain. The case for such type 0 innovations is much rarer but still possible. The value chain model also suggests that innovations can take place in any part of the value chain and by doing so they can affect other innovations upstream or downstream[3].

Due to the technology dependent nature of IS innovation, software organizations adopting significant Type 0 and I innovations *together* can often *produce* radically new applications (Type II). In situations like this, software organizations take part in *disruptive IS innovation* (Lyytinen and Rose 2003a) where change both in processes and development outcomes is pervasive and radical. These disruptions are by necessity outcomes of radical breaks in the IT base, when components in the computing base are re-assembled in novel ways (Henderson and Clark 1990). As identified in Lyytinen and Rose (2003a, 2003b), Internet computing was an example of 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 in faster speed (Type I).

This model helps investigate the extent to which changes in computing capability (Type 0) *can* and *will* lead to innovations in the development activities (Type I) like agile development, and the consequent *fast* adoption of novel applications (Type II innovation/agility). We conjecture that the innovation capability in and for agility is produced by two combined capabilities: 1) the capability of software organizations to adopt Type 0 innovations, and 2) their capability to successfully *transform* and *hone* these new capabilities into Type I innovations like *agile* development. This transformation is dependent

[3] Swanson (1994) calls these strong and weak order effects.

on the mobilization of two capacities. The first capability -- *technology absorbtion* -- reflects an organization's ability to efficiently sense, acquire and absorb new base technologies and to deploy them effectively (Srinivasan *et al.* 2002) through *exploration*. The second capability of *process trransformation* is reflected in a software organization's: (1) ability to use occasions of new IT deployment for process improvement; and (2) to effectively learn from such occasions as to standardize and formalize process knowledge into complementary assets that can be mobilized for fast development. This latter process we call *exploitation.* In other words, succesful software process innovators need to effectively and continuously identify and match the strategic opportunities for their process improvement with the emerging computing capabilities.[4]

Exploration and Exploitation

In the management literature, the concepts of *exploration* and *exploitation* have been established as two fundamental modes organizational response to environmental challenge (March 1991). These archetypes of organizational learning help managers and scholars alike to distinguish two modes in which organizations compete and adapt, and which draw upon very distinct logics of how to organize, strategize or execute. Through exploitation organizations garner and refine in trial and error learning their competencies through repeated actions over extended periods of time (Eisenhardt and Martin 2000, Nelson and Winter 1982, Nonaka and Takeuchi 1995, March and Levinthal 1993). Exploitation is thus about harnessing "old certainties" and implies behaviors that are labeled as refinement, implementation, efficiency, production and selection. Exploration, in contrast, is about discovering new opportunities whereby organizations search and create new competences by engaging in second loop learning (Christensen 1997, Eisenhardt and Tabrizi 1995, Henderson and Clark 1990, March 1991, Tushman and Anderson 1986, Winter and Szulanski 2001). Exploration involves behaviors labeled as search, discovery, experimentation, risk taking and innovation. Exploration and exploitation are like water and fire (Brown and Eisenhardt 1998, Tushman and Anderson 1986): they require substantially different structures, processes, strategies, capabilities and culture. Exploration leans towards 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, Levinthal and March 1993).

Due to their fundamental differences (March 1991, Mezias and Glynn 1993), exploration and exploitation pose a continuous tension for management (Gibson and Birkshaw 2004, He and Wong 2004). On one hand, exploitation fosters inertia and reduces capacity to adapt and seize new opportunities. On the other hand, exploration slows down the speed in which existing competencies can be improved (March 1991). These tensions create often dysfunctional learning outcomes when either exploration or exploitation is one-sidedly preferred (March 1991, Levinthal and March 1993). Trial and error learning and successful adaptation through exploitation can bias management to focus too much on current capabilities- at the expense of new opportunities- thus causing current capacities to become core "rigidities". Such constant search for short term efficiencies leads to learning myopias and competency traps (Levitt and March 1988). In contrast, when organizations engage in excessive exploration continued "failure leads to search and change, which lead failure which

[4] Note that Figure 2 emphasizes that radical innovation is not a product of one-way communication from ISD organizations to their clients. Our model recognizes that there is a combination of supply/push and demand/pull mechanisms and that ISD organizations engaged in radical innovation can be effective at either or both active pushing or reactive sensing of client demands and environmental pressures.

lead to even more search and so on" (Levinthal and March 1993). In such situations organizations' learning becomes random and chaotic: managers love to explore and but fail to allocate resources to exploit their new competencies.

Due to the threat of competency traps an increased attention has been paid in understanding how organizations learn to tack effectively between exploration and exploitation by changing their resource bases through acquisition, integration, re-combination, and the removal of capabilities (Eisenhardt and Martin 2000, Lant and Mezias 1992). In doing so (software) organizations must relentlessly integrate, reconfigure, gain and release resources in order to respond to swift changes (D'aveni 1994, Eisenhardt and Martin 2000, Teece et al. 1997). A (software) organization's dynamic capability thus embodies a learning related meta-capability by which this organization learns to effectively blend exploration and exploitation across different stages of technology innovation. In other word, software organizations must learn to both explore and exploit with multiple IT innovations across the IT innovation value chain. These distinct moments of innovation have differential impacts how agility is perceived and defined.

Exploration and Exploitation in Software Development Organizations

In the context of IT innovation software organizations' *agility* is determined by their capability to *both explore* and imagine emerging needs and match them with the observed technology potential, *and then* to *exploit* these product innovations by improving their product delivery capability. The general logic of exploration and exploitation during IT innovation stages is depicted in figure 2. Exploration processes result in adopting new type 0 base innovations that enable organizations to produce both type II and type I innovations. An example of type II innovations would be the organizations' capability 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, or to integrate the customers better into the process.

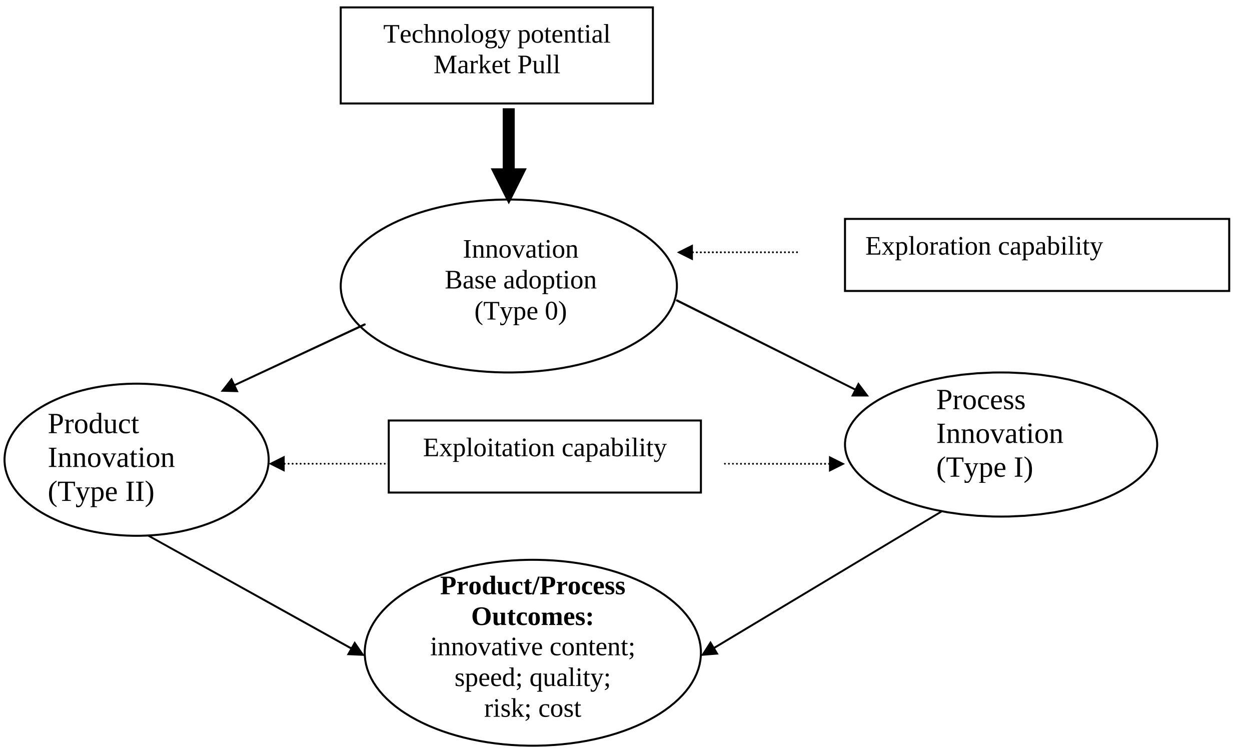

Figure 2. A general model of IT innovation as exploration and exploitation

During exploration agility means two things which both must be managed well: 1) the software organization must adopt new type 0 and I technologies faster than its peers, and 2) the organization must use these technologies to develop faster type II innovations, if such need arises in the market (explorative process innovation). The organizations' capability to address these needs is clearly dependent on the firm's *absorptive capacity* (Cohen and Levinthal 1991). If the organization is successful in the exploration this will lead to changes in the organizations' innovation base in that both its product (the type of applications, type II innovations) and the process (the way in which it develops these type II innovations) will change. The more the former deviates from the current product mix, the more *innovative in content* is the change- one can say that organizations is agile in its product innovation. The more the latter deviates from the current process the more *innovative process* is instantiated for software delivery- one can say that the software organization is agile in its process improvement. Normally, changing the process in itself is not a goal *per se* but it is evaluated in terms of how it contributes to improvements in the *innovative content, quality, risk or speed*.

Software organizations need also exploit while products and technologies mature by streamlining, standardizing, automating, scaling up their processes as to gain better control. They must rely on their exploitation capability, which can be defined as the organizations' learning capability to improve and change their delivery processes over time as to maximize desired process outcomes including speed, quality, risk or cost. Clearly, this learning mode and associated organizing logics are distinct from exploration tasks that focus on innovative content. Agility can be rather defined as lubricating a well-defined process- not how fast such processes can be revamped and replaced.

We can now formulate a model how exploration and exploitation are organized across different phases of IT innovation (figure 3). We will use this model later to explore how each phase (1-4) affects process features like agility or cost. In this model Type 0 innovations can be regarded as "technology push" which seeks to improve and expand both software products and processes. Growth in this 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 way of developing systems) that enable new* innovative solutions and processes. Such explorations take place in quite short and intense periods during which hyper-competition and fast learning are valued[5] (Phase 1). 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 (Phase 2). When such a stage is achieved organizations (or sometimes when organizations are doing product explorations) move to discover significant and radical ways to improve their product delivery processes. This stage we call process exploration or type I radical innovation (Phase 3). 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 the process improvements is mostly exhausted the organizations will move to process exploitation what we call process exploitation or incremental type I innovation (Phase 4).

[5] We call this hyperlearning in Lyytinen et al (2004).

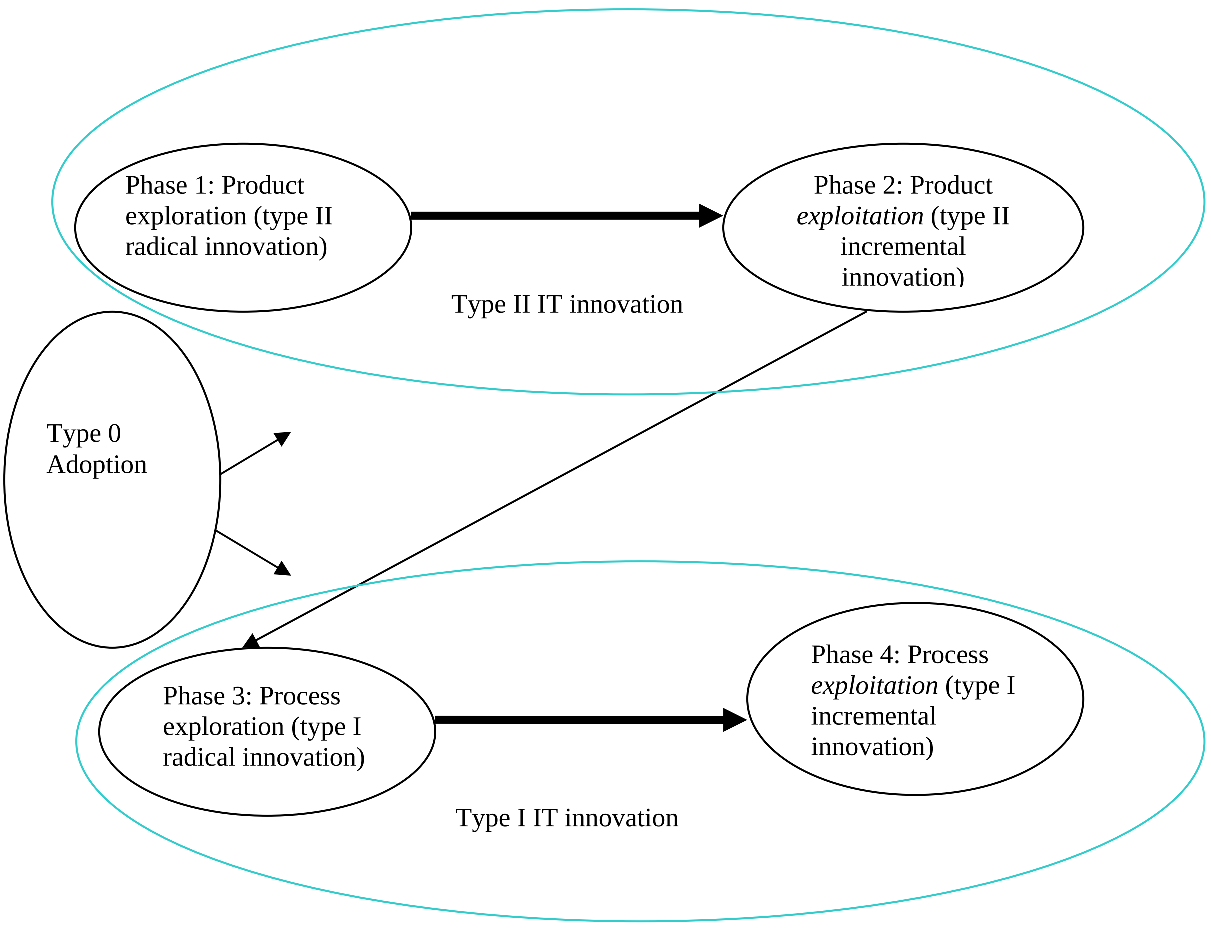
Phase 1: Product exploration (type II radical innovation)
Phase 2: Product *exploitation* (type II incremental innovation)
Type II IT innovation
Type 0 Adoption
Phase 3: Process exploration (type I radical innovation)
Phase 4: Process *exploitation* (type I incremental innovation)
Type I IT innovation

Figure 3. Organizing logics for exploration and exploitation across different types of IT innovations

A Model of Process Features During Exploration and Exploitation

Relationships between desirable process features like innovative content, speed, cost, quality and risk are complex across innovation phases. Clearly these dependencies vary across different phases of IT based exploration and exploitation. In general it is impossible to maximize all of them simultaneously and the relationships between them are different during exploration phase where new type II innovations are discovered or incremental type I innovation where additional process steps are proposed. In general, process features across stages (or organizations) can be modeled as directed graphs where each process feature is depicted as a separate dimension to be optimized and where the relative size of the vector shows to what extent this process feature is being maximized during this stage[6]. An example of such a graph is shown in figure 4 below for a situation that is typical for radical product exploration (phase1). It suggests that during product exploration software organizations seek to maximize innovative content, they tolerate relatively high risks, expect relatively fast product development and medium cost, but do not aspire for high quality. In short, the software processes in which organizations engage maximize innovative content (exploration depth and breadth) and speed, while tolerating higher risks and costs by sacrificing quality.

[6] In van Kleijnen (1980) these are called Kiwiat graphs.

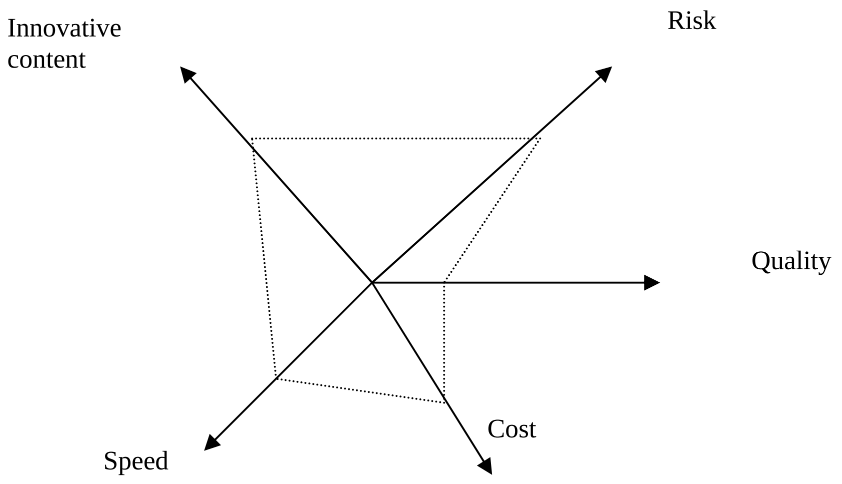

Figure 4. Desirable process features for product exploration

The features influence causally one another during each stage- but differently. During product exploration organizations seek to maximize innovative content and exploration processes thus suggesting the following relationships:

For Innovative Content[7]:
(1) + Innovative Content → + Risk (i.e. when innovative content increases risk increases)
(2) + Innovative Content → + Cost
(3) + Innovative Content → - Quality
(4) + Innovative Content → - Speed

Simultaneously, 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 from these relationships, speed and innovation take precedence over the other factors. However, both cannot be optimized simultaneously and an increase in one counteracts the other.

To phrase these observations in other way: during product exploration organizations must be willing to face higher risks and cost. They must also speed up their exploration speed but despite this improvement their capability to deliver any workable solution may be slowed down. If they want to be more nimble they will also incur higher cost, face higher risks and may have to sacrifice their innovativeness.

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

We can in a similar fashion model the desirable process features for incremental process innovation (pure focus on exploitation, phase 4)[8]. Its goal graph is shown in Figure 5. The graph shows that organizations in this stage seek to maximize quality and speed while minimizing their cost and risk. They do so by fixing most of their product and process features. Such process goal combination is typical for relatively mature and well-defined software markets. This goal structure has been assumed in a majority of the process improvement research (see e.g. Humphrey 1989). In this situation cost reduction, increased quality, risk avoidance, and increased speed dominate at the expense of innovation. We observe the following causal dependencies as a result:

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

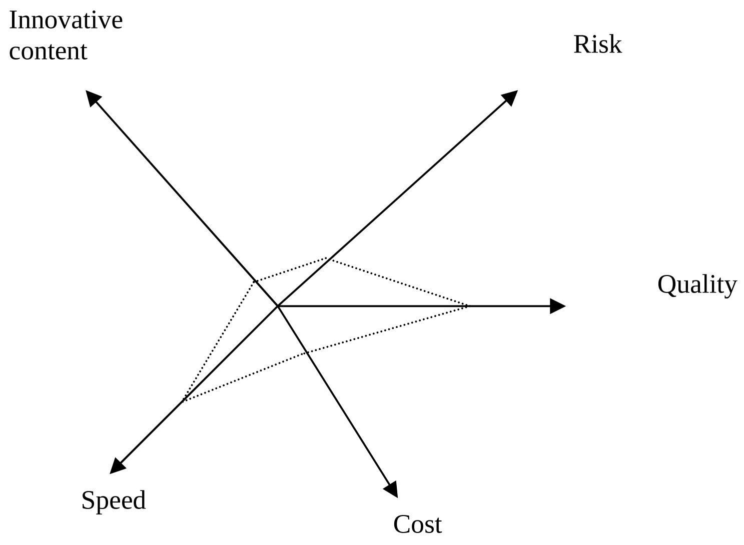

Figure 5. Process features in type I incremental innovation

In general software delivery speed in exploitation is faster than in exploration as time is not wasted to explore product features or architectural solutions. Instead, the focus is on incremental innovations that can deliver significant improvements through economies of scale and scope.

Some Implications for The Study of Agility and Process Improvement

Per March's theory, if an organization engages in radical type II innovation, it will decrease its opportunity for incremental process innovations (exploitation) due to their contradicting logics. Likewise increases in organizations' exploration efforts will decrease their current exploitation capability. From this follows that organizations focusing primarily either on exploration or exploitation- though both view agility as a desirable feature – have quite different mindsets about agility.

During exploration an organization's desire is to explore fast and build up new products that are shown to work, while during the exploitation their main focus is to remove friction from their well-defined processes and to utilize current process capabilities to the

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

fullest. It is important to note, however (as shown in figure 3) that the new technology potential (type 0) *per se can dramatically increase* the speed of development by offering tasks with higher granularity (e.g. ERP parametrization vs. implementing code to meet specific requirements), using powerful abstraction mechanisms (e.g. Web services), utilizing highly standardized functionalities (e.g. relational databases systems or browser based user interface), standardizing architectural integration mechanisms (E.g. solution and architectural patterns). Improvements through such mechanisms can be dramatic and be as important as radical innovations in products. Yet, in the early stages of exploration the process improvement potential is never exhausted and the fast product innovations take place with the additional cost of decreased quality and higher cost/risk.

When an organization shifts its focus from radical IT innovation it must increase its exploitation effort. It will rather emphasize fixing the product qualities and later on the process qualities. This requires it to change key process measures as its main focus will now be on efficiency, economies of scale, and standardized development outcomes (quality control). Hence organizations' interest changes over time from innovative content and exploration speed to process cost, speed and quality. This shift will lead to increased reliance on trial and error learning in contrast to generative learning that enables to establish causally effective routines to utilize technology potential (March and Levinthal 1993).

As shown in figure 3, due to the structural and time disjuncture between exploration and exploitation and due to their contrasting logics (the left hand side vs. the right hand side of figure 3) software organizations innovate with IT in lumpy manner. They need to continually balance tradeoffs between innovative content, cost, speed, quality, and risk and arrange accordingly in contradicting ways how they exploit technologies, organize processes and control outcomes. The IT innovations are moreover influenced by different learning outcomes at each phase, and they are "pulled" by diverse market signals. Therefore, each innovation in the IT base will be "appropriated" through multiple "different" innovation paths as shown in figure 3. During these processes the organizations will adapt their strategies and goals while the technology changes leading to diverse product innovations and different market environments, which gradually shift their focus from exploration to exploitation. Because the contrast between early exploration and late exploitation is deep and stark software organizations must transform themselves while innovating in a stepwise manner. They can only entertain only a certain amount of transformations over a time period (if they want to do so). As a result organizations normally follow staggered adoption and learning patterns: they continue to increase their innovative agility first by adopting radically new technologies (type 0), but later on shift their focus on exploitation by stabilizing product features and then seeking to stabilize process features. At the same time they may engage already in other exploration / exploitation cycle thus organizing their capabilities in ambidextrous manner (Tushman and Anderson 1986).

The impact of organizational transformation on critical process features- innovative content, quality, risk and cost- is significant. The concern for agility is not the same at all times. Agility in absorbing technologies dominates in the early exploration as a result of radical innovation, and later on it is about introducing incremental changes in the product delivery process. Overall software organizations need to locate themselves into alternative exploration / exploitation regions during the diffusion. Each region offers a different concept of agility. The main positions of alternative regions are outlined in table 1 depending on whether the software organization places its efforts on the left or right side of the figure 3 (exploration vs. exploitation focus).

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*

Table 1. Contingencies for organizational learning in software development

The first contingency where both types of learning is low is rare and can be mostly observed in state controlled and bureaucratic software development environments. When only exploration focus is high this can be considered typical R & D software development (pre-competitive phase) where the development speed for the final product does not count much (most academic software development). When both exploration and exploitation is high (i.e. organizations are fast oscillating between two phases of product innovation in figure 3) this can be regarded a case of hyper-competition. This was typically observed in new economy software development organizations between1997-2000 and the related learning condition is defined as hyper-learning in Lyytinen *et al* (2004). The simultaneous push to both high exploration and high product exploitation arose from huge unexplored technology potential and its potentially disruptive nature. It was expected to change businesses based on new software products and processes. The push towards higher levels of exploitation comes normally from heightened competitive demands created by the growing market size stiffer competition, alternative economies of scope, and new value propositions. When this happens the organizations focus tilts towards process improvements and organizations start compete based on their process integration and management capability. In general, we can observe that agility in software organizations relates either to organizations' capability to be fast explorers and innovators or to be effective process integrators i.e. to improve their capability to integrate and manage delivery processes within expected cost and quality. The shift between these positions happens when software organizations recognize that the so-far "emerging" technology potential has become mainstream, and its impacts on the market structure is significant. In this situation they must decide whether they will shift their focus on specific new markets that value exploration (increase innovative content) or whether they need to specialize on exploitation, where they need to control process features like reliability and cost efficient delivery.

In light of this model we wanted to explore the following questions: 1) do perceptions of and need for agility change during different phases of IT innovation?; 2) how software organizations manage contradicting demands of exploration and organize their innovation for agility? 3) does the IT innovation model help to predict how agility is perceived in relations to other process features?

Research Method and Research Sites

Research Goals and Design

We conducted a five-year longitudinal field study (Yin 1994) in a theoretical sample of Web development software companies (March et al 2001) to address the above questions. We chose multi-site case study as it allowed a replication logic by which we could test emerging theoretical insights in different contexts and triangulate both theory and data (Eisenhardt 1989, Strauss and Corbin 1990)[9].

FIRM	Firm 1	Firm 2	Firm 3	Firm 4	Firm 5	Firm 6	Firm 7
Division Focus	Custom software development. Primarily e-business applications	B2B e-Business consulting solutions	Small spin-off of parent. Is Web-based ASP for parent company's customers	E-Business consulting specializes in mobile computing	Systems integrators to upgrade legacy systems to include Web and mobile solutions	E-Business solutions Specializes in mobile computing Need-based assembling of components and applications	Management consulting, development IS products, networking and hosting services
History	15 year old firm- had been mainframe and client server shop with 500 employees in 4 locations	Part of a large, multinational business consulting company	Part of a large financial company with several thousand employees	Multinational e-Business consulting firm founded in 1995 with several thousand employees	e-Commerce development firm founded in 1996 starting with 6 employees	Large multinational e-Business consulting and software development firm	Mature, large, multinational development and IT service firm
# 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 / 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 2. Firm characteristics

[9] The "Web- development" refers to computing applications that utilize Internet browsers, such as Netscape or Microsoft Internet Explorer, and a set of open standards and protocols that include XML, HTML, http, URL, TCP/IP, combined with the extensive use of middleware architectures in leveraging the computing service.

We identified seven companies that met the following criteria: (1) they were developing Web-based systems; (2) they were recognized by their peers as fast adopters of the most advanced technologies available; and (3) they worked mostly for outside and leading edge clients through contractual relationships. To minimize bias we sought to maximize the variations in our sample that would improve external validity (Yin 1994). Companies had different sizes and operated in different industry sectors in terms of the services provided (ranging from manufacturing, financial services and public administration to retail and transportation). They had experience using Web-based technologies in several application domains (back office, front office, and inter-organizational applications). The geographical scope of their operations varied largely as some were local software firms while others were part of large global companies. The firms also had large variations in their software development experience, ranging from as few as four years to 40+ years. A summary of the firms' characteristics is included in Table 2.

Data Collection and Analysis

The data was gathered primarily between June 2000 and April 2003 at four different time points (2000, 2001, 2003-2004) from seven companies. The exact times and periods of data collection are shown in table 3. For all seven 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 sets we had problems with the poor quality of the tapes and could not transcribe them verbatim but instead just collect main facts. We organized the interview data into three different temporal periods: pre 2000 (Time 1), 2000-2001 (Time 2), and 2002-2004 (Time 3) that align well with the different stages of the dot-com boom. Here pre 2000 stands for market growth and period of fast innovation driven by the ideas of new economy, period 200-2001 stands for the recession and crisis, and 2002-2004 stands for the new recovery and modest growth of the markets.

The data was obtained through semi-structured tape-recorded interviews with senior management and senior developers who made decisions about technology investments and were in charge of business strategy. The interviewees managed the organizational knowledge bases and skills needed to execute a chosen technology and business strategy. We also examined the archives of the company documents, including systems development documentations and technology strategies and made notes during the visits concerning their physical sites, personnel age, general atmosphere etc. A range of one to six individuals from each company participated in the study. A total of 19 interviews were conducted with a typical interview time of approximately two hours each. The transcribed data covers currently c.a. 2000 pages of interviews. The interviews reviewed the applications and solution portfolios provided by these companies to their clients and examined how they delivered these applications. We probed for changes taking place in the business and the technology domains of their operations as a result of Internet computing. Specifically, we asked the firms to clarify the extent, scope, depth and speed of changes in their software development practices when compared with their situation prior to the Web development and during different stages of its deployment. We further examined how these firms coped with changes in technology and markets. The interviews were transcribed and the summaries of these transcripts were sent to the companies for correction and validation.

Data analysis was done using inductive method (Boyatzis 1998, Glasser and Strauss 1967, Strauss and Corbin 1990). The transcripts of each company for each time period was subject to a within-case analysis that involved repeatedly reading the transcript and taking thorough notes about firms' perceptions of the competitive environments and their reactions to the environmental changes and their perceptions of agility and process features. After each individual case had been analyzed, we began cross-case comparisons that involved listing the similarities and differences among the firms in process outcomes at each period of time. The results of the cross-case and cross-period comparisons provide a vivid picture of the ways in which companies tacked between exploration and exploitation and accordingly changed their process features. Furthermore, the results provide a consistent picture of how these organizations controlled agility in order to speed up either exploitation or exploration and how they configured accordingly their processes.

FIRM	Firm 1	Firm 2	Firm 3	Firm 4	Firm 5	Firm 6	Firm 7
Interview Date 1	Jun. 2000	Jun. 2000	Jun. 2000	Oct. 2000	Sep. 2000	Nov. 2000	Sep. 2000
Time 1	Six senior employees including an executive, managers, and software architects	A senior manager of an IS development group and one of his key developers	The CIO, and the five key senior technologists who 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	Four senior employees including a systems architect, manager, and software engineer	One senior manager of IT development services
Interview Date 2	October 2001	October 2001	October 2001	August 2001	August 2001	August 2001	August 2001
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	One manager from first interview	Manager who replaced manager in first interview
Interview Date 3	March 2003	March 2003	No interview	No interview	March 2003	April 2003	April 2003
Time 3	An architect from Time 1, the replacement of executive in Time 1, and a developer not in Time 1 interview	Developer from Time 1	Firm absorbed by parent company and IT employees reassigned. Interviewees not available. Time 3 information gathered via email 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	One manager from first interview	Manager who replaced manager in first interview

Table 3. Data collection summary

Research Findings

Changes in Agility During Exploration and Exploitation

Table 1 and Figure 3 suggest a movement from a product exploration to process exploitation among software organizations as they adopt a new technological base. A summary of the changes in focus in exploration v.s. exploitation in our firms through three observation points is given in Tables 4a and b. As indicated therein, each of the firms in the early stages of Internet computing (e.g., the periods between 1995 and the first interview [noted as Time 0 and Time 1]) were engaged in radical innovation product when compared to period 2.

FIRM	Firm 1	Firm 2	Firm 3
Time 0	Phase 1: Product Exploration Phase	Phase 1: Product Exploration Phase	Phase 1: Product Exploration Phase
Time 1	Phase 2: Product *Exploitation* Phase Speed increase; risk increase; costs increase. Base technology is inherently faster in internet frame. Certain amount of freezing product innovation had already begun and allowed time reduction via reuse. Projected in Time 2 that they would focus on more reuse to increase time and decrease risk. Indication was that they planned to freeze product innovation to allow this process innovation to occur.	Phase 2: Product *Exploitation* Phase No rigid methodologies. Have faster development than before Internet development but deemed to be with poor quality.	Phase 2: Product *Exploitation* Phase Spun off by parent company that focuses on methodologies and process exploitation. Recognized they were in a new environment and required radical product innovation. Quality required. Speed faster than in the parent firm, but at 24 months to get a working product, it was considered unacceptably slow in "new economy." No formal process used. Lots of experimentation with product. Product innovation was high, risks were high, quality was lower than the parent company standards. Just at that time planning on freezing product innovation and beginning radical process innovation. Goal was to increase speed and quality. Did so at high cost by buying outside help (grafting). Goal is to stabilize and get speed up/quality up/risk down.
Time 2	Phase 3 Process Exploration Phase and then Phase 4: Circumstantial Process *Exploitation* Phase Speed not a problem because project scopes were smaller as client demand for product innovation stopped. Business dropped off significantly and required elimination of many programmers. Fired all slowest programmers and kept those with tacit understanding of effective methods (a Darwinian model). Combination of things allowed for rapid development, but not by design. They had methods as a result of market requirements and a natural selection of developers, not because they looked to have methods.	Phase 3: Process Exploration Phase Compared to Time 1- faster still; costs lower; quality higher. Fast learning stopped. Process innovation occurred in Type I as a result of stability in Base IT and incremental innovation in base.	Phase 3: Process Exploration Phase Product innovation frozen. Process innovation slowing down to almost frozen as well. Development speed was way up. Quality was better than in Time 1. Stable base technology. Stabilized product. Stabilized process. Goal to shore up process methods to maintain high quality and increased speed for future rollouts.
Time 3	Phase 1: New Product Exploration Phase Speed for ISD still an issue but urgency around it gone. Looking into exploration in radical new Base and product innovations. Inherent in their mission statement (a single sentence about meeting client needs with leading-edge IT innovations).	Phase 4: Process *Exploitation* Phase Technology Base, processes and product solutions have matured and stabilized since Time 1 and Time 2. Speed up still as a result of: new Type I innovations; stability of solutions and knowledge sets; and reuse (stable base tools and incremental innovation in base).	Phase 4: Process *Exploitation* Phase Innovation frozen entirely. Quality very hi risks in ISD and costs in ISD approach zerc ISD firm is swallowed up by the parent company and leadership given to marketing team instead of it team. Originally spun out of parent to innovate rapidly and radically. Team disbanded in 2003 and frozen products and processes (methods) swallowed up.

Table 4a. Innovation summary (USA firms)

Six of the seven firms were responsible for their own product innovations and began period 0 (before the first interview time) as radical product innovators (Phase 1). They then moved sequentially to Phase 4 during our observation period. One firm (Firm 6) in our data set did not conduct their own product innovation though it engaged in Internet computing explorations. Instead it formed alliances with other radical product innovators (thus outsourcing that activity) and focused all the time on process innovations. They sought to deploy the existing product bases as quickly as possible and thus were already in Phase 3 at Time 0. This they did at the cost of radical product innovation and they emphasized the need to control that activity. Not surprisingly, by period 1, Firm 6 was already engaged in process exploitation (Phase 4).

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 get them there. 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 First firm of the seven to limit to a fixed set of product innovation solutions. Therefore, had process innovations possible earliest. Mostly reuse of components, affiliation with partners, and reusable methodologies. Very high quality, low costs, and low risks because of fixed solutions.	Phase 2: Product *Exploitation* Phase Speed increase; risk increase; costs increase. Base technology is inherently faster in internet frame. Certain amount of freezing product innovation had already begun and allowed time reduction 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 wrong for them. Cost rose and speed declined as a result of moving again toward product exploration phase	Phase 3: Process Exploration Phase Moved to incremental product innovation stage. Coincides with return of methodologies.	Phase 4: Process *Exploitation* Phase Nothing different from Time 1 except perhaps 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 again. Further problem when encounter changes in base innovations. Counteracted with process innovations.	Phase 4: Process *Exploitation* Phase Nothing different from Time 1 except perhaps even more focused on exploitation of product and process.	Phase 4: Process *Exploitati* Phase Focused on more reuse to increase speed and decrease risk. Freezing product innovation allowed this process innovation to have occurred.

Table 4b. Innovation summary (Finnish firms)

While each firm moved eventually to Phase 4, some of them moved beyond Phase 4 (or back) to a new Phase 1 by starting to seek new technology and solutions thus demonstrating the need for ambidexterity. Each of the organizations found that they could not successfully continue to engage solely in process exploitation. In the case of Firms 1 and 5, they found that adopting radical Base innovations and creating radical product innovations made the process innovations they had developed during the previous cycle less effective. Notably, these two firms saw their development agility decrease notably and they needed to evaluate constantly tradeoffs between speed and other features. Likewise, Firm 4 found that it now incurred higher costs and slower speeds. This firm found this by period 2 and subsequently went out of business as a result of declined market demand and wrong capability set. Overall our data set shows that organizations organized their perceptions of agility and concerns for exploration and exploitation as suggested by the IT innovation model.

Impact on Process Features and Speed

Figures 4 and 5 highlight critical interrelationships between ISD process features at different stages of exploration and exploitation. As indicated in the figures, organizations must continuously control five interrelated and contradictory process features: speed; innovation; cost; risks; and quality. Of the 19 interviews, we found strong evidence that these five factors were heeded by the organizations during each of the periods (see Tables 5a and b, and 6a and b for details).

FIRM	Firm 1	Firm 2	Firm 3
Time 1	Evidence of tradeoffs between speed and quality, costs, and risks	Evidence of increases in speed and decline in quality Evidence of tradeoffs between speed and quality, costs, and risks	Evidence of tradeoffs between speed and quality, costs, and risks
Time 2	Evidence of tradeoffs between speed and quality, costs, and risks	Evidence of tradeoffs between speed and quality, costs, and risks	Evidence of tradeoffs between speed and quality, costs, and risks
Time 3	Evidence of tradeoffs between speed and quality, costs, and risks	Evidence of tradeoffs between speed and quality, costs, and risks	

Table 5a. Tradeoffs between speed vs. quality, costs, and risks summary (USA firms)

FIRM	Firm 4	Firm 5	Firm 6	Firm 7
Time 1	Evidence of tradeoffs between speed and quality, costs, and risks	Evidence of tradeoffs between speed and quality, costs, and risks	Evidence of tradeoffs between speed and quality, costs, and risks	Evidence of tradeoffs between speed and quality, costs, and risks
Time 2	Evidence of tradeoffs between speed and quality, costs, and risks			Evidence of tradeoffs between speed and quality, costs, and risks
Time 3		Evidence that slowing down development reduces costs and improves quality		Evidence of tradeoffs between speed and quality, costs, and risks

Table 5b. Tradeoffs between speed vs. quality, costs, and risks summary (Finnish firms)

Within the data set, we also found strong evidence for the types of interrelationships between the five goals as noted in Figures 4 and 5. Specifically, we evinced that when organizations wanted to increase speed in innovation, they faced a tradeoff of increased risk, increased cost, and decreased quality (see Tables 5a and b). In 16 of the 19 interview transcripts, these tradeoffs were clearly observed by as the price for increased speed. For example, a developer in Firm 1 indicated *"because you(are) moving so fast through the whole thing, if you mess up somewhere it impacts you a whole lot more than it would have impacted you in a slower process."* Likewise, a participant from Firm 7 noted, *"you have less time to think and you don't have the time to think of everything."* The dominating desirable process feature in period 1 is innovation and we also observed that speed and innovation are inversely related. Again, in 16 of the 19 interviews, evidence was found for the negative relationship between speed and innovation (as can be seen in bold in Tables 6a and b). For example, Firm 3 finished their proof in concept stage of development and subsequently stopped radical product innovation. As a result of moving to incremental innovation in Time 1, they were going to be able to formalize a methodology to enable the *"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 in all three innovation types in Figure 1. 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]."*

Likewise, the remaining relationships between innovation and the features of risks, cost, and quality were observed (as noted in the non-bolded data summaries in Tables 6a and b). For example, in period 1, a member of Firm 7 referred to the period before the radical innovations associated with Internet development as *"the good old days"* and noted that lower risks were *"old fashioned."* Similarly, Firm 5 noted when it began adopting radical base innovations for creating radical innovations in period 3 (and thus entered a new phase 1 period), development was slower, more resources needed to be allocated, and quality declined:

> *"I already did miss the deadline and the resource allocation [target]… when there [have been] only just a couple of experts in certain [new base technology] and we needed to share the knowledge by allocating the people that were not in so big in that technology, it meant that also the amount of time and the amount of work used were exceeded but also the qualities probably not the best possible one when looking back at the acts or work."*

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 6a. Tradeoffs of innovation vs. speed, quality, risk, or cost summary (USA 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

Table 6b. Tradeoff of innovation vs. speed, quality, risk, or cost summary (Finnish firms)

Collectively, the directional interrelationships of the five factors shown in Figures 4 and 5 are supported. With regards to phases, the primary relationships shown in Figures 4 and 5 are also supported. Figure 4 represents the desired tradeoffs made by a firm in product exploration (phase 1) and Figure 5 the desired tradeoffs made in the process exploitation phase (phase 4). As can be seen in the data in Tables 4a and b, in earlier phases quality was lower, risks were higher, and costs were higher. In later phases the opposite was true (note in all phases, speed was deemed as important and as such it does not show a differences between the figures, but the type of speed in some sense was different).

The tradeoffs between innovation and the other factors are most evident when period 1 is considered across the firms. In period 1, Firm 6 was already in phase 3 (product exploitation). They were already reaping the rewards of this phase and noted that quality was higher, costs were lower, and risks were lower because they had purposefully frozen innovation and were primarily engaged in assembly of *"ready made components"* and they had *"a set of solutions that [they knew] how to give and [could] give them quickly."* In contrast, the other firms were basically beginning phase 2 and all saw increased risks and costs, with decreased quality.

As time moved on, and each of the firms moved into later phases their market needs matured (in terms of Type II innovations and Type 0 innovations). Likewise their methodologies became more refined and they were implemented, while the risks, costs, and quality moved to the pattern of tradeoffs as shown in Figure 5. For example, Firm 2 entered phase 4 in during period 3. The interviewee noted in that interview that their *"methodologies and strategies are now mature"* and that quality was greatly improved as *"a function of better trained people, a methodology,…and less innovation."*

Discussion and Conclusions

In this paper we have developed a model of IT innovation which views it as an extended process of both exploration and exploitation across IT value chain. The model predicts that software development speed/agility is not a universal goal within all phases of IT innovation. In contrast it is affected by the scope and depth of innovative activity in base technologies that offer new radical products and new types of software processes as well as in process innovations when complementary assets are garnered. To address the issue of development agility more thoroughly we conducted a multi-site longitudinal field study in seven software development organizations to address the following questions: 1) do perceptions of and need for agility and the type of agility change during different stages of IT innovation?; 2) how do software organizations manage contradicting demands of exploration and exploitation and how they organize their innovation for agility accordingly?; 3) does the model of exploration and exploitation help to predict how agility is perceived in the context of other process features?. We observed within our study: 1) concern for both exploration speed and process development speed changed significantly over the period of study; 2) software development organizations tended to organize themselves differently during the study period and they changed from fast explorers to process integrators as technologies matured. Some organizations also positioned themselves as general explorers (agility in exploration) or generic process integrators (agility in integrating new technological solutions into existing generic development routines); and 3) the variance in agility in relation to process features varied across innovation phases and also between companies due to their differentiated focus on exploration or exploitation.

In general software organizations control their agility in terms of how good they become in adopting disruptive technologies during different stages of innovation. They must constantly trade off agility against other criteria like risk, quality or innovative content. How these trade-offs are made depend on garnered organizational competencies, shifting managerial focus and new competitive demands. Over time organizations learned to locate themselves differently in this space (explorative players: agility primary focus, exploration main capability; process orchestrators: exploit new technology in large scale fast, use alliances, source knowledge, focus on process and integration skills). These competitive niches had quite different business logics and managers interpreted differently what “agility” meant.

There are several avenues for future research in this fascinating area. First we need to generalize the findings here with 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 also explore other factors than just the organizations’ learning focus to establish causal explanations of agility in organizational contexts.

References

Agile Alliance (2002): http://www.agilemanifesto.org/, visited 09/04/04

Attewell, P. (1992) “Technology diffusion and organizational learning: The case of business computing.” *Organization Science*, 3, 1, pp. 1-19.

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

Boyatzis, R. 1998. *Transforming Qualitative Information: Thematic Analysis and Code Development*, Sage, Thousand Oaks, CA.

Brown, S. L., K. M. Eisenhardt. 1997. The art of continuous change: Linking complexity theory and time-paced evolution in relentlessly shifting organizations. *Administrative Science Quaraterly* **42** (1-34).

Carstensen P, Vogelsang L. (2001), “Design of Web based information systems; new challenges for system development”, in Proceedings of the 9th ECIS, Bled Slovenia, 27-29-2001, pp. 536-547.

Christensen, C. M. 1997. *The innovator's dilemma: When new technologies cause great firms to fail*, Harvard Business School Press, Boston, MA.

Cohen, W. M., D. A. Levinthal. 1990. Absorptive capacity: A new perspective on learning and innovation. *Administrative Science Quarterly* **35** 128-152.

Curtis, B., M. Kellner & J. Over (1992): Process Modeling. *Communications of the ACM*, Vol. 35, No. 9 (75–90).

Cusumano M., Yoffie D. 1999: "Software development onInternet Time", *IEEE computer*, 32, 10, pp. 60-69.

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

Eisenhardt, K. M. 1989. Building theories from case study research. *Academy of Management Review* **14** (4) 532-550.

Eisenhardt, K. M., J. A. Martin. 2000. Dynamic capabilities: What are they? *Strategic Management Journal* **21** 1105-1121.

Eisenhardt, K. M., B. N. Tabrizi. 1995. Accelerating adaptive processes: Product innovation in the global computer industry. *Administrative Science Quaraterly* **40** 84-110.

Fichman, R.G., and Kemerer, C.F. "The Assimilation of Software Process Innovations: An Organizational Learning Perspective," *Management Science* (43:10) 1997, pp 1345-1363.

Gibson, C. B. and J. Birkinshaw, 2004. The Antecedents, Consequences, and Mediating Role of Organizational Ambidexterity, *Academy of management Journal*, **47** (2) 209-226.

Glasser, B. G., A. L. Strauss. 1967. *The Discovery of Groupded Theory: Strategies for Qualitative Research*, Aldine Publishing Company, New York, NY.

He, Z., P. Wong. 2004. Exploration and Exploitation: an Empirical Test of the Ambidexterity Hypothesis, *Organization Science*, **15** (4) 481-494.

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

Henderson-Sellars B. Serour M. (2004): Creating a dual agility method – the value of method engineering, forthcoming *Journal of Database Management* 2004.

Humphrey, W. (1989): *Managing the Software Process*. Reading, Massachusetts: Addison-Wesley.

Kessler E., Bierly P. (2002): “Is faster really better: an empirical test of the implications of the innovation speed”, *IEEE Transactions on Engineering Management*, 49, 1, pp. 2-12.

Kessler E., Chakrabarti A. (1996): “Innovation speed: a conceptual model of context, antecedents and outcomes”, *Academy of Management Review*, 21, 4, pp. 1143-1191.

Kessler E., Chakrabarti A. (1999): “Speeding up the Pace of New Product Development”, *Journal of Porduct Innovation Management*,, 16, pp. 231-247.

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

Lant, T. K., S. J. Mezias. 1992. An Organizational Learning Model of Convergence and Reorientation. *Organization Science* **3** (1) 47-71.

Lyytinen K, (1987) "Different Perspectives on Information Systems: Problems and their Solutions", *ACM Computing Surveys*, 19, 1, pp. 5-44.

Lyytinen K., Rose G. (2003a), “Disruptive Information System Innovation: the Case of Internet”, *Information Systems Journal*, 13, 4, pp. 301-330.

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

Lyytinen K., Rose G., Yoo Y. (2004): “Exploring and Exploiting in High Gear: Hyper-learning in Seven Software Firms”, submitted to a special issue on exploration and exploitation in *AMJ*.

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

Menon A., Chowdhury J., Lukas B. (2002): “Antecedents and outcomes of new product development speed: an interdisciplinary conceptual framework”, *Industrial Marketing Management,* 31, pp. 317-328.

Mezias S., Glynn M. 1993. The Three faces of corporate renewal: institution, revolution and evolution, *Strategic Management Journal*, 14, 1, pp. 77-101.

Nelson, R. R., S. G. Winter. 1982. *An Evolutionary Theory of Economic Change,* Belknap Press, Cambridge, MA.

Nonaka, I., H. Takeuchi. 1995. *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation*, Oxford University Press, New York.

Pressman, R. 1998. Can internet-based applications be engineered? *IEEE Software* (Sept-Oct) 104-110.

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

Teece, D. J., G. Pisano, A. Shuen. 1997. Dynamic capabilities and strategic management. *Strategic Management Journal* **18** (7) 509-533.

Turk D, France R., Rumpe B. (2004): “Assumptions Underlying Agile Development Processes”, forthcoming *Journal of Database Management* 2004

Tushman, M. L., P. Anderson. 1986. Technological discontinuties and organizational environments. *Administrative Science Quaraterly* **31** 439-465.

Winter, S. G., G. Szulanski. 2001. Replication as strategy. *Organization Science* **12** (6) 730-743.

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