2004

The High Speed Balancing Game: How Software Companies Cope with Internet Speed

Jan Pries-Heje
The IT University of Copenhagen, jph@itu.dk

Richard L. Baskerville
Georgia State University, baskerville@acm.org

Linda Levine
Carnegie Mellon University, ll@sei.cmu.edu

Balasubramaniam Ramesh
Georgia State University, bramesh@gsu.edu

Follow this and additional works at: http://aisel.aisnet.org/sjis

The High Speed Balancing Game

How Software Companies Cope with Internet Speed

Jan Pries-Heje,

The IT University of Copenhagen, Denmark
jph@itu.dk

Richard Baskerville

Georgia State University, Atlanta, USA
Baskerville@acm.org

Linda Levine

Software Engineering Institute, Pittsburgh, USA
ll@sei.cmu.edu

Balasubramaniam Ramesh

Georgia State University, Atlanta, USA
bramesh@gsu.edu

Abstract. This paper compares and contrasts the practices in developing application software for the Internet in 2000 and 2002. We identify key organizational and technical factors that facilitated or impeded implementation of Internet applications. The identification is done through a grounded theory analysis of data collected from ten companies. A comparison of the 2000 and 2002 data shows how major factors, such as market environment and lack of experience emerged to change the software process and the attitude toward

quality. The comparative study shows how the interrelationship between the core factors of speed and quality, together with the other major factors, unfolded in a decision process wrought with trade-offs and balancing decisions at multiple levels in the software organization. These trade-offs and balancing decisions – a high speed balancing game – are taking place at three different levels: the market, the portfolio, and the project. Balancing game theory extends existing theories about software development trade-off decisions, providing new insights into how such trade-off decisions are translated into software practices. The paper includes a comparative analysis of the findings from 2000 and 2002.

1 Introduction

This paper compares and contrasts the practices in developing application software for the Internet during two time periods: during the peak of the dot.com boom in 2000 and just after this economy collapsed in 2002. The distinctions are interesting because the fast-moving nature of Internet-connected global marketplaces creates a different environment for software development. It requires the ability to rapidly develop and deliver software systems. Formal knowledge in this fast paced development began to emerge at the turn of the millennium (e.g., Cusumano & Yoffie 1999a). Traditional software development approaches failed to match the needs of this new world, and new practices emerged. These practices centred on the development of good teamwork, the flexibility to adapt to rapidly changing environments, and the ability to compensate for the inevitable, but unpredictable consequences of software process innovation (Baskerville et al. 2003). These new principles fit well in the context of a software development culture closely coupled to fast changing markets, and led to a remarkably well-defined suite of practices. These practices included parallel development, a release-driven orientation, a dependence on development tools, customer involvement, prototyping, defined architecture, component oriented designs, and tailored methodologies that often hold maintenance in disregard (Ramesh et al. 2002).

These high-speed approaches to software creation have variously been called agile methods (Aoyama 1998), light methods (Beck 2000), short-cycle-

time methods (Baskerville and Pries-Heje 2004), Internet-time methods (Cusumano and Yoffie 1999a, 1999b) and Internet speed software development (Levine et al. 2002). We adopt the last convention in this paper. Internet speed means that software products are developed very quickly, perhaps taking less than a month to raise from an initial concept to an operational system. Internet speed methods are more formal than hacking but less formal than traditional, rigorous methodologies (Highsmith 1999b). Internet speed development is "adaptive rather than predictive" (Beck 2000) and emphasizes improvisation (Miner et al. 2001). Examples include spiral models of development (Boehm 1998; Boehm, et al. 1998), Agile Software Process (Aoyama 1998), Scrum (Rising and Janoff 2000) and eXtreme Programming (XP) (Beck 2000; Beck and Fowler 2001; Jeffries et al. 2001).

Existing research has established ways in which Internet speed development is different from traditional development (Baskerville & Pries-Heje, 2004). There is a package of at least five systems development practices that distinguish this new form of systems development. These practices included a focus on completion speed, release-oriented parallel prototyping, architecture, negotiable quality and an ideal workforce. While no single practice may be entirely new, the packaging of these practices is somewhat revolutionary. This package has been shown to be consistent with amethodical development approaches that facilitate emergent systems and thereby organizational emergence. Such systems are not built as a monolithic project that completes with a delivery, but rather are continually growing (or being grown) to adapt to the emergent organization (Truex et al. 1999, 2000).

Critics have pointed to some problems with Internet speed development, including the potential loss of control in project management and the potential for inefficient and chaotic software design and architecture that require re-factoring to be rationalized (Fowler 2000). The approach is characterized as somewhat unstable, unpredictable and lacking in repeatability. Internet speed development is a much less defined process than traditional software engineering approaches, embodying a different kind of discipline.

A grounded theory analysis of data collected from ten companies suggests the following: changes in practices during these two time periods occur both within and outside the software development units. External to these organizations, fundamental changes in the economic conditions affected the resources available for development. Expectations of success from Internet speed software projects have changed dramatically in three key ways. First, the IT economy underwent a major upheaval as revenue fell; productivity rose and budgets were slashed. Second, business expectations changed. Rather than an obsession with fast software delivery, customer concerns shifted to emphasize

J. Pries-Heje, R. Baskerville, L. Levine & B. Ramesh • 13

both the quality and speed. Third, the economy forced an emphasis on the business case for software projects; the concerns of the project managers expanded to encompass the value to the enterprise, including development of more complex, mission critical software systems.

The study finds an emerging set of practices followed in 2000 and 2002, before and after the boom for Internet software development. In other words, a similar set of technical solutions was in use in both periods although these practices continue to evolve. For example, in 2002, we found greater reliance on hard-won experience and the application of business solutions.

2 Related Research

There is increasing evidence showing that current approaches for process improvement in software engineering (e.g., ISO9000-3, CMM, SPICE, Bootstrap) are effective in large-scale software engineering efforts of a long duration (Harter et al. 2000; Krishnan et al. 2000). Such approaches are focused on optimization and disciplined control of software engineering processes. However, software development in high velocity organizations is characterized by rapid changes in requirements and unpredictable product complexity. Agility in software development is increasingly recognized as essential for survival under such conditions (Thomke & Reinertsen 1998). Aoyama (1998) states that "to be agile, the process must be flexible enough to adapt smoothly to changes in requirements and delivery schedule." Agility provides the ability to reconfigure a system quickly and cheaply and helps achieve responsive and economical development (Newman et al. 2000). Firms that compete in volatile and unpredictable market situations must employ different product development methods (Cusumano & Yoffie 1999a, 1999b).

Software developing organizations have been struggling to find the appropriate methods, tools and practices to meet the challenges imposed by Internet software development. Most traditional software development methods and tools deal with the needs of large systems developed by multiple large teams. Such methods and tools have difficulty keeping up with the increasing size and complexity of large-scale systems, but more importantly in the context of high-speed software development, they are also not easily downward scalable (Laitinen et al. 2000). It is unclear whether existing methods and tools can be adopted in Internet software development (Fayad et al. 2000). Traditional software development methodologies are often criticized as bureaucratic and slowing down the pace of the development process. In contrast to 'heavy methodologies' or 'monumental methodologies' (Highsmith 1999a), there is

growing interest in 'light methodologies' that are more formal than hacking and less formal than the popular rigorous methodologies. These methodologies are claimed to be "adaptive rather than predictive" (Beck 2000). Unlike heavy methodologies that plan in detail weeks or months ahead, light methods "adapt and thrive on change, even to the point of changing themselves." The tension between structure and creativity manifests countervailing forces that both spark invention and introduce the structure necessary for transforming creative inventions into marketable products (Brown & Duguid 2001). Agile methods emphasize improvisation, a form of real-time, short-term learning (Miner et al. 2001) over highly structured approaches to software development. While improvisation may interfere with aspects of long term learning processes, it can also play a positive role in long-term trial-and-error learning (Miner et al. 2001). There is tension between the agility advocated by light methodologies and the definition advocated by process improvement frameworks (e.g., CMM). For quality software development, this tension poses interesting challenges. Evaluating XP from a CMM perspective, Paulk (2001) suggests that several ideas in XP contribute to achieving CMM levels 2 and 3 practices. However, he cautions that organizations using XP should carefully address management and infrastructure issues that are not adequately addressed in XP.

Recognizing the need to address the concerns of Internet speed software development, attempts have been made to develop methodologies that can be tailored to the needs of particular kinds of projects. For example, Conallen (1999) presents a methodology by refining, extending and conflating the Rational Unified Process (RUP) (Kruchten 1998) and the ICONIX process (Rosenberg & Scott, 1999) as a part of the Web Application Extension for UML (WAE) framework. WAE is an attempt to use RUP as a lightweight methodology. Spiral models of development (Boehm 1998; Boehm et al. 1998) and eXtreme programming (XP) (Beck 2000; Beck & Fowler 2001; Jeffries et al. 2001) are among those practices gaining attention for quick development in an environment that is ill-understood and in which requirements are rapidly evolving. Proponents of XP suggest that XP can be used in longer projects by breaking them into a sequence of self-contained, one- to three-week mini-projects.

Though agile development methods offer many commonsense solutions, they are not without critics. For example, one of the problems associated with XP is the potential loss of control in project management. In addition, developing applications in small pieces that evolve into a larger system can result in an inefficient and chaotic software design and an architecture that requires "re-factoring" to be rationalized (Fowler 2000). Further, as the methodologies

themselves are not very stable, predictability and repeatability (the hallmarks of a defined process) cannot be guaranteed.

The use of concepts like Internet speed and Internet time has triggered some fundamental rethinking of our perception of time. For example, driven by the confusion of time zones for the globalization of telecommunications-based contracts, Swatch proposed defining a new international time standard, Internet time. Under Internet time, a day would be fixed worldwide without time zones (eg, midnight would occur simultaneously worldwide), and would consist of 1000 "beats" instead of hours, minutes and seconds (Lee and Liebenau 2000). Such proposals have brought attention to the issue of time itself. Hall (1966) distinguished between two ways of organizing time, M-time and P-time (monochronic and polychronic time), where M-time involves doing things one by one and P-time involves doing things in parallel. Barley (1988) studied computer-support for radiologists and found that the computer enforced M-time work at the cost of P-time. To the contrary, Lee (1999) found that information systems enhanced the organization of work according to P-time. Looking at work itself Lee & Sawyer (2002) found that work varies by two characteristics: the level of worker interdependence and the degree of work autonomy. They also discovered that M-work and P-work can be related to either processes or structures. In settings where high-speed software developers have a lot of autonomy and require a lot of interaction, Lee & Sawyer would use a "Club" metaphor to describe their tempo-spatial relationships because both the structures and the processes are polychronic.

Lyytinen and Rose (2003) discuss Internet computing as a disruptive IT innovation in systems development. Their research illustrates how a major shift in the underlying base technologies (such as internetworking) may demand transformational innovations in both systems development and the products of systems development. Such shifts are characterized as architectural innovations that are necessary (but not sufficient) to enable radical and pervasive kinds of changes in software development.

While the current literature recognizes many challenges, and new ideas are emerging, much room remains for better work toward solutions and frameworks for Internet software organizations. Shenhar (2001) argues that management of some kinds of software projects requires a new approach. In his study of 127 projects, he identifies four levels of technological uncertainty and three levels of systems complexity that affect the appropriate type of project management. Our research question is designed to add in a significant way to this body of work, primarily through investigating, first-hand, the innovative practices actually used by high velocity software development organizations, further illuminating how Internet speed software product development is car-

16 • J. Pries-Heje, R. Baskerville, L. Levine & B. Ramesh

ried out in practice. This paper offers insights into the ways in which firms are succeeding with Internet speed software development and why they choose to approach their problems in these particular ways.

3 Research Study and Questions

The many unknowns surrounding Internet speed and fast-paced software development were the initial drivers for this exploratory case study. In 2000, researchers and Internet strategists alike were claiming that an Internet year was roughly equivalent to one month; however, no empirical studies on Internet software development had been reported or published to corroborate these claims. A rich description of Internet speed did not exist. In response, we set out to study the phenomenon of Internet speed, formulating several high-level research questions: are Internet software development processes different from traditional development processes? If so, how? What improvement approaches might be a match for this development environment? Generally speaking, we were testing the hypothesis that working on Internet time would necessarily cause changes in the way software development processes were organized. Beyond these high-level questions, no hypotheses had been formulated in advance.

We investigated business strategies, product priorities, and Internet application development processes in ten companies in various regions of the United States, through initial interviews conducted in the fall and winter of 2000-2001, and a second round of interviews in 2002.

The firms ranged in size from 10 employees to more than 300,000 employees, in different industries in the private and public sectors including: financial services, insurance, business and consulting services, courier services, travel, media, utilities, and government services. Some of the firms were new Internet application start-up companies while others were brick and mortar companies with new Internet application development units.

At the time of the second round of interviews, only five of the original nine companies remained in business or were available to participate in the study. Only one of the small Internet software houses had survived. To maintain the representative nature of the selection of companies, we added a tenth company - a small innovative Internet software house. A brief description of each firm is provided in Table 1. Several of the companies asked to remain anonymous. As pseudonyms for 2000-2001 companies, we chose the names for The Nine Muses; the ancient Greek goddesses of inspiration, learning, the arts, and culture. The pseudonym for the tenth company is Deca, the Greek number ten.

J. Pries-Heje, R. Baskerville, L. Levine & B. Ramesh • 17

Name	Industry and what offered? When founded, size?	Number of people interviewed in each round and their organizational roles
Calliope	• Creates weather derivates based on customer demand. Utility companies who are interested in learning about their customers' consumption habits purchase the weather derivatives. • Mid 1990s. 20 employees when interviewed	2000: 3 people interviewed: VP Operations, Project Manager, Software Developer 2002: Not interviewed
Clio	• Exploit the benefits of purchasing goods, such as utilities, in large quantities. The fact that customers can receive a lower price for utilities if they were to purchase them in large quantities became the basis of the firm's business model, offering a buying pool online. • Late 1990s. 35 employees when interviewed.	2000: Six people interviewed: President & CEO, VP Technology Operations, Director of Marketing Research, Chief Information Officer, two developers 2002: Not interviewed
Deca	• Develops and markets a platform of E-business software modules allowing users more control when doing business online. • Late 1990s. Approximately 10 employees when interviewed	2000: Not interviewed 2002: Four people interviewed: CEO, developer, QA specialist and marketing manager
Erato	• Creates e-commerce sites for traditional brick-and-motor companies. The firm develops the personification of the client's brand name and brand image on the web. • Late 1990s. 55 employees when interviewed.	2000: Four people interviewed: Director, Chief Financial Officer, Chief Operations Officer, and developer 2002: Not interviewed

Table 1. Facts about the ten companies interviewed

18 • J. Pries-Heje, R. Baskerville, L. Levine & B. Ramesh

Euterpe	• Provides its clients with the technology for searching and indexing videos. Was the sole competitor in this market, and enjoyed many benefits of being in a market niche. • Mid 1990s. 80 employees when interviewed.	2000: Four people interviewed: Project managers, marketing specialists, senior web developers 2002: Not interviewed
Melpomene	• Offers services to SMEs for managing and communicating human resources, employee benefits and payroll information. The systems it creates reduce the administrative costs of its clients. • Mid 1990s. More than 100 employees when interviewed.	2000: Seven people interviewed: Project managers, process improvers, architects, user interface designers, web developers 2002: 6 of 7 people interviewed. Process improvement person had left company
Polyhymnia	• Provides travel arrangements for mid-sized corporations. Because the firm focuses only on corporate travel, it can better understand the needs of its clients. • Early 1990s. More than 1000 employees when interviewed.	2000: Six people interviewed: Senior managers, Project managers, QA manager, lead developers, web developer 2002: Seven people interviewed: Same distribution of roles as in 2000
Terpsichore	• Offers various types of insurance such as automobile, motorcycle watercraft, etc. Successfully launched a website in mid-1990s, where customers can get quotes or buy insurance online. • First half of 20th Century. More than 10000 employees when interviewed.	2000: Three people interviewed: Human Resources Manager, Internet site manager and Internet site developer. 2002: Not interviewed.

Table 1. Facts about the ten companies interviewed

J. Pries-Heje, R. Baskerville, L. Levine & B. Ramesh • 19

Thalia	• Delivery provider. The firm plays a significant role in the delivering of products to end-consumers. The website offers functionality such as the ability for customers to track their packages on-line. • First half of 20th Century. More than 100000 employees when interviewed.	2000: Six people interviewed: CIO, Senior manager, project managers, architects, senior developers, web developers 2002: Three of the six people interviewed: CIO, senior and project manager
Urania	• Provides information technology consulting and solutions to different types of companies in many different industries. We focused on internet applications in the telecommunication sector. • Last half of 20th Century. The part we looked at was founded in the 1980s. More than 100,000 employees when interviewed	2000: Six people interviewed: Senior manager, Project managers, quality assurance manager, QA specialist, Web developers 2002: Six people interviewed. Same roles. But only three were the same people.

Table 1. Facts about the ten companies interviewed

4 Research Method

Our approach in both the 2000 and the 2002 studies was empirical and qualitative. We used semi-structured interviews as a forum for collecting data, following the same study guide in both studies. The data was analyzed using grounded theory techniques to develop a central story line or core category. This approach uses several coding stages to rigorously link the story line to the underlying data. See the Appendix for details about the research method, including the interview guide and an example of the coding.

5 A High Speed Balancing Game

During the frenzied 2000 dot-com expansion, time-to-market, inspired by the need to be first-to-market, was prevalent. A second major factor at this time was the lack of experienced developers (a central resource). These two con-

certed factors formed the major causal factors in the 2000 story. With the economic pullback of 2002 and the changing economy as a major influence, the cases re-centred on the need to optimize sets of business and technical solutions in order to maximize speed and quality.

The business solution set comprised models to select the most lucrative projects affordable. The technical solution set and the experience of the developers played an important role in achieving maximum speed and quality for a given project. However, this simple, single perspective on speed and quality is deceptive. In fact, trade-offs and decisions (on business and technical issues) were performed at multiple levels—at the organizational level (*the portfolio level balancing game)* and at the level of individual projects *(the project level balancing game)*. Beyond the project and organization, there was also a *market level balancing game* that encompassed tradeoffs and decisions by customers and suppliers in the marketplace. We will see how each of the constants of time, resources, scope, and quality are interpreted differently at each level.

5.1 Portfolio Level Balancing Game

At the portfolio level, the constants or values (time, resources, scope, and quality), are interpreted in terms of the organization's identity, the nature of its market place and competitors and translated into the portfolio concerns—mission, core competencies, risk, revenue streams, capital, branding, product strategy, staffing, etc. Senior management responds to these demands by balancing needs, constraints and trends to gain profit, market share and satisfy customers. The set of business solutions, as defined by the business models, helps the senior manager select specific projects and set their scope accordingly in the context of the portfolio. Customer expectations, market conditions, and the changing IT economy all play important roles achieving maximum speed, quality and profitability.

5.2 Project Level Balancing Game

These portfolio-level tradeoffs and decisions, in turn, influence the balancing game at the project level. Here, project-level values and constants (time, resources, scope, quality) are interpreted in terms of project cost and schedule, project scope, resources, functionality, and product quality. The use of tools and techniques that comprise the set of technical solutions, along with the seasoning that arises from experience, helps the project manager to achieve speed

and quality in the specific project. The project manager, making decisions and tradeoffs at this level, is engaged in the project-level balancing game.

5.3 Market Level Balancing Game

All of the internal tradeoffs and decisions will directly or indirectly reflect the organization's positioning in its customer and supplier market places. Time is relevant in the competitive demands to respond as a first-to-market innovator, or a market-follower, etc. Competitive pressures often drive resources as well. Such commodities as skills, knowledge, tools, may be more or less expensive or available. Scope issues can be regarded as decisions about the breadth or depth of market penetration by various organizations in the marketplace (the product dimension), as well as customer and supplier demands (the demand dimension). The levels of quality, expected and supplied, are also part of the balancing game.
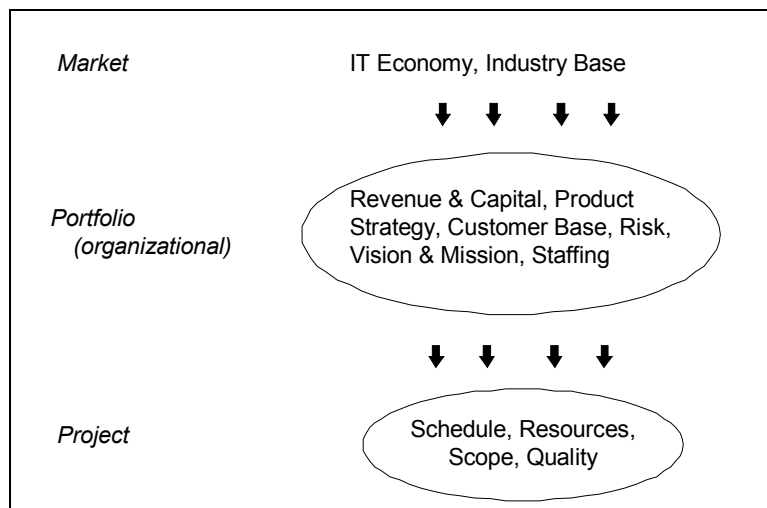

Figure 1. The High Speed Balancing Game

5.4 Three Balancing Games

In Figure 1 we illustrate the project, portfolio, and market levels. While each area of play is often viewed as a set of independent decisions, our research suggests that the games are interactive. Decisions made by one player to

22 • J. Pries-Heje, R. Baskerville, L. Levine & B. Ramesh

change the balance in the market level game may affect the decisions made by multiple organizations to achieve balance at the portfolio and project levels. In the next sections of this paper we summarize the results of our two-part case study on the phenomenon of Internet Speed, to illustrate how this research has contributed to our current representation of the high speed balancing game.

6   Results From 2000 Study

Figure 2 illustrates the Grounded Theory core category (the story line) from the 2000 study. This story line emerged from the coding analysis of the interview data. The analysis led us to three major categories of observations *causing* a change, and three major categories *resulting* from the changing causes. Furthermore, one of the resulting categories (New Software Process) was easily sub-categorized. Each of the categories is described below, along with examples of how the companies manifested these properties.
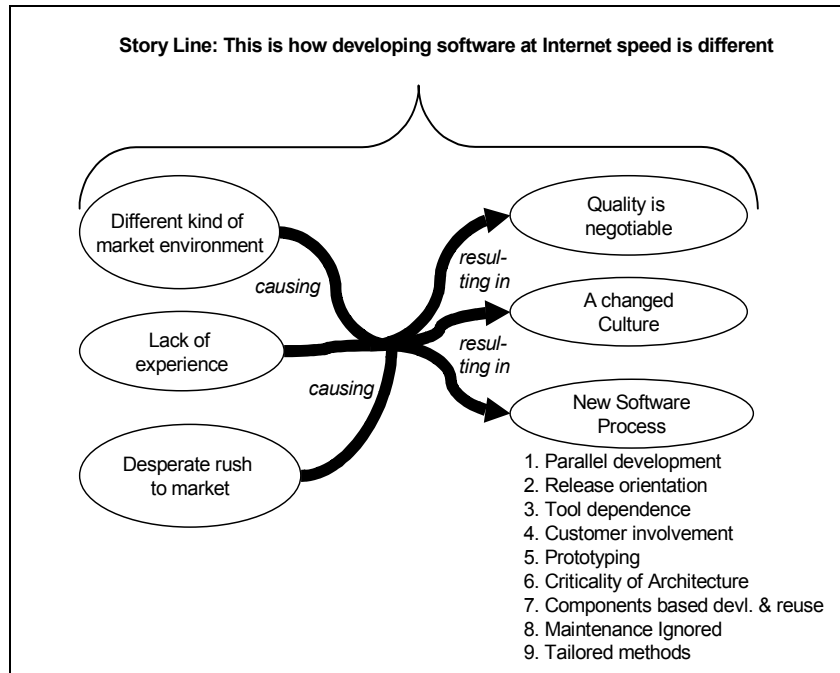

Figure 2. Results from 2000 study

J. Pries-Heje, R. Baskerville, L. Levine & B. Ramesh • 23

6.1 Across all levels

Software product quality was definitely lower in our 2000-study, a function of implicit negotiations in the market between software competitors and customers. Creative people, operating under a different set of constraints, were willing to rethink the meaning of quality. Customers and users appreciated immediate functionality and were willing to defer a certain amount of reliability and performance. Developers were willing to rebuild badly designed or coded features later when their deferment runs out. "It is different working at Internet speed. Compressed cycles mean that quality suffers. With speed we are sending less quality out the door," they told us at Polyhymnia.

6.2 The market level

"Time-to-market—I hear that constantly. Bigger, faster, better. Everything is very rush, rush, rush," they told us at Polyhymnia. The standardised architecture around the World Wide Web browser meant that deployment of Internet software occurred instantly. The Internet, the Web and the browser created an international market with a breadth and scale of nearly inestimable potential. Venture capitalists, particularly in the US, were pouring vast amounts of money into small companies with high Internet market potential. In the 2000, time was key in the market level balancing game, driving many decisions in the games at the other levels.

Different kind of software market environment. The characteristics of the Internet and WWW environment represented a large and unique marketplace for software products—one that was flexible in terms of requirements and quality. Demands for quality were less stringent, especially in the early stages of a product's introduction. Requirements were negotiable from release-to-release in a market-defined process where pragmatics intervene to limit the features in each release. Quality factors, such as scalability and maintainability could be postponed for later releases. In 2000, scope and quality were deemphasized in the market-level game, reinforcing a similar pattern in the project and portfolio balancing games.

Lack of experience developing software under these conditions. In 2000, there were too few knowledgeable developers with experience to meet the speed and market challenges above. Furthermore a lot of the experience from old style development organisations was not useful. For example, a manager from Melpomene told us "lots of people [in our organisation] came from more corporate environments where it took forever to get things out the door." Much of this prior experience was a hindrance rather than a benefit in the new

environment. This shortage of experienced professionals made the market-place for developers tight and expensive. Decisions at the other levels reflected this supply-bound market level game.

6.3 The Portfolio Level

We found that Internet software development organizations had a distinct culture that appreciated informal structure, smaller teams, and diverse team compositions. Individuals were not replaceable programmers in a sea of developers.

At the same time, there seemed to be a tight bond among Internet software developers, a sense of belonging with others who share the same values. We encourage people to "take a risk, make a mistake. But, be smart not to make the mistake twice," as they said at Thalia, "We are not 9 to 5 people down here. We are more dynamic … There is lot more excitement and enthusiasm here." The portfolio included Internet speed projects that were assigned to Internet teams and traditional projects assigned to traditional teams. The portfolio was rebalanced with an emphasis on one resource—people.

6.4 The Project Level

We identified nine distinct characteristics. Although no single characteristic was unique to this new software process, the collection of characteristics was distinctive. Table 2 outlines these characteristics, illustrating how the market level balancing game changed the project processes. The emphasis on time at the market level percolated down to an emphasis on time in balancing project decisions.

Characteristic of the new software process	Description and examples
Parallel development	To achieve high-speed we found that companies compressed development into a time frame where only overlapping, parallel development could meet the demands. Sometimes, coding begins even before the requirements have been fully understood. "In these projects, even during the requirements phase, our teams will begin coding, expecting what will be in the final requirements and in various releases"(Urania).

Table 2. Nine characteristics of the new software process in the 2000 study.

J. Pries-Heje, R. Baskerville, L. Levine & B. Ramesh • 25

Release orientation	"People have a perception of Internet speed. They expect it. So we've had to scope our delivery or deliver a smaller set of features. Thereby releasing more often," said a manager from Euterpe. Clio said: "Development cycles last from 2 to 15 days… timing is important. Features that cannot be completed in time can slip from one release to the next. The fast cycle time softens the penalty from slipping a feature."
Tool dependence	Urania estimated that "fifty percent of development is already taken care of by tools we use such as iPlanet or WebSphere. The APIs to these tools gives a lot of functionality." Many Internet software development organizations made heavy use of development tools and environments that could speed up the design and coding process. Furthermore new tools also helped to create well modularised and architected systems.
Customer involvement	When requirements are fuzzy it helps having close access to customers. Thus intimately involving customers to cope with evolving and unstable requirements was typical. Our line of enquiry into aspects of development teams revealed that customers were often co-located with these teams, and participated closely in all phases of development. Most projects relied on such involvement rather than a formalised requirements management process.
Prototyping	Instead of using formal requirements documents, most projects use prototyping as a way to communicate with their customers to validate and refine requirements. Customers describe the basic functionality for new or changed features and these are quickly prototyped for demonstration and experimentation. "We are supposed to have a full [requirements and design documents] but a lot of programmers use the prototype and go back and forth to check, or go back and ask: 'What was this supposed to do?'"(Melpomene)

Table 2. Nine characteristics of the new software process in the 2000 study.

26 • J. Pries-Heje, R. Baskerville, L. Levine & B. Ramesh

Criticality of architecture	A well-planned architecture enabled each release to be developed with some similarity, and the largest possible reuse of components. A three-layer architecture was common: (1) Database layer, (2) Business logic layer, the detailed processing code, and (3) User interface layer. Architecture was also a framework for the division of labour in some firms, e.g., a database team, a business logic team and a user interface team.
Components based development and reuse	Internet speed can be achieved by software assembled with as many reusable components as possible, rather than crafted from scratch. "Internet speed needs reuse. We need to take components or assets and know how to put them together", said a Thalia developer.
Maintenance ignored	The short life span of Internet software meant that maintenance often was not given serious consideration. "Products are not documented. No design document, no requirements specification. The person who did it is gone. It takes much longer time. Often we can start from scratch. It leads to a throw away mentality"(Polyhymnia).
Tailored methodology	The processes and methods used in Internet software development varied considerably depending on the composition of the project team and the nature of the product. "We have an overall methodology. But we have to tailor processes for individual teams." (Urania) Just "enough process to be effective," added Euterpe.

Table 2. Nine characteristics of the new software process in the 2000 study.

6.5 Results From 2002 Study

Figure 3 illustrates the core category (the story line) that emerged from the 2002 study. We can compare this result with that from the 2000 study (Figure 2) in order to reveal major changes in these software development settings. This comparative analysis reveals that two such major changes have taken place in high-speed Internet software development from 2000 to 2002. First, quality is no longer being treated as a disadvantaged stepchild. Speed and quality must be balanced for companies to survive in the newer market. Second, related monetary factors have been reversed: the unending supply of money characteristic of the boom has dried up; and good people are no longer such scarce resources.

J. Pries-Heje, R. Baskerville, L. Levine & B. Ramesh • 27

6.6 Across All Levels

Speed and quality. According to our 2002 results, the core factors of speed and quality are interrelated in a way we had not seen previously. Since these factors were often discussed in association, we treat them together. Individuals in all of the companies commented explicitly about these factors and the struggle to balance needs for speed and quality.   Consider, for example, some of the statements from the various companies we interviewed: "E-speed and e-haste are just normalcy now. Now you just know that you have to go that way and balance for quality." (Urania) "Going slower also means an improvement in quality. And we develop with that in mind." (Polyhymnia).

One manager elaborated upon the need to seek a balance between speed and quality, showing how the element of risk was factored into the speed-quality equation. "With every quality issue, we put the risk on the table. For example we tell the customer that if we do this fast there may be bugs in the 20% of the code. On the other hand the 80% would earn you so many millions in revenue from week one. What do you say?" (Urania) Typically, the need for speed concerns product development and delivery to the customer and market but it can also relate to behaviour among competitors. At one company, a staff member remarked: "Another driver for speed is competition from other consultancy firms. We regularly have fights with other consultancy companies to get projects here." (Urania).

Results form 2002 study. The need for speed appears to be as constant in 2002 as it did in 2000. Only one company (Polyhymnia) acknowledged a possible slowing, and in doing so they also observed that their projects were larger, more complex and now oriented to needs of the enterprise. "Pace may have slowed a little. We used to work so many hours, we had people working 70 hours to get something out the door … [But now] most projects are moving to enterprise releases twice a year … so projects are larger and more of a long term thing."

The customers had grown accustomed to high-speed development and were expecting it in every project. Quality, however, was viewed as having greater importance than previously seen. Quality is associated with the number of defects, with the overall success, and with customer satisfaction. Many of our interviewees highlighted the need for quality in both the product and the process. As the products and the markets mature, the need for quality also appears to get significant attention. "If you didn't follow your processes, or do your documentation, that is not quality. Quality is overall success—of
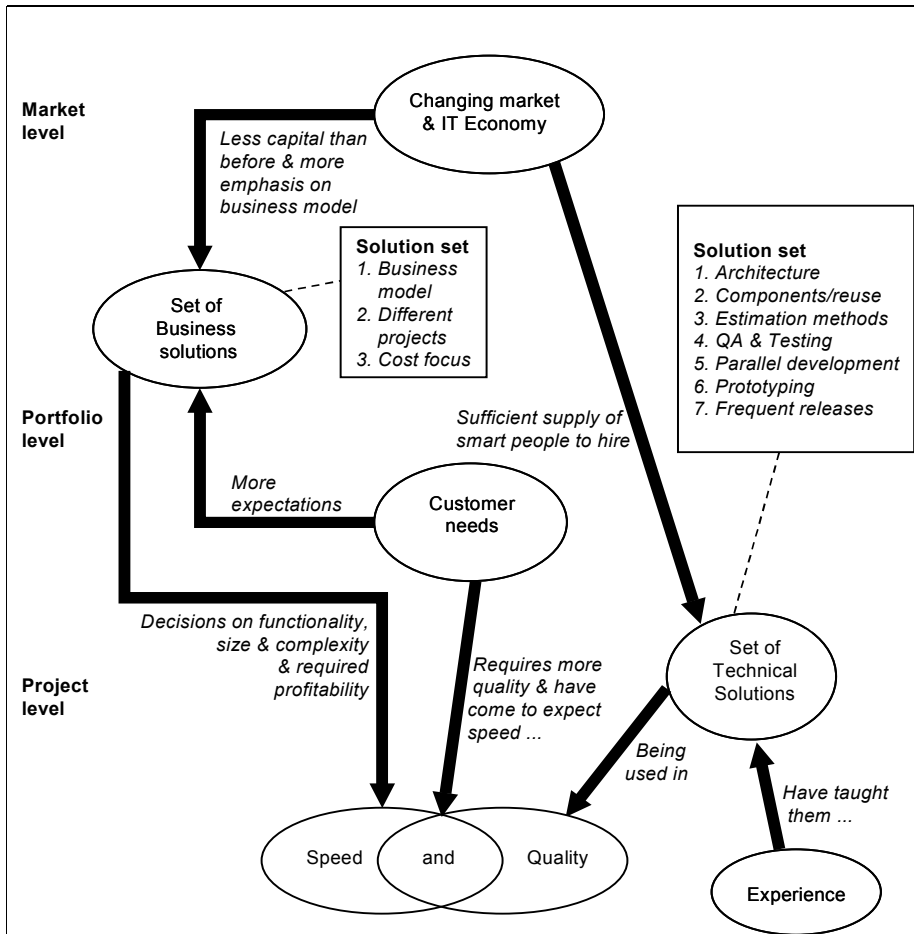
28 • J. Pries-Heje, R. Baskerville, L. Levine & B. Ramesh

Figure 3. Story line: Balancing game at three levels

the team, and the deliverable." (Urania) "I measure quality in terms of pro-
duction defects … They'll forget you're late but they won't forget if it's bad"
(Polyhymnia).

6.7 The Market Level

Changing market and IT economy. Disappearing venture capital along with
tight budget constraints have forced companies to focus on cost and business
models. Companies have stopped making new hires, have made staff redun-
dant, and many have virtually eliminated contractor positions. A ready supply

of talent is now available. The need for speed, expressed in terms of customer expectations, remains as critical as the developer's toolbox of technical solutions applied to achieve high-speed development. What is new is the employment of a set of business solutions at the project-portfolio level, carefully coordinated with use of a set of technical solutions at the project level.

The desperate rush to market, in conjunction with the need to operate in a different kind of market environment and the lack of experience developing such products (Ramesh et al. 2002) evident in the 2000 data have been significantly transformed. The changing market and IT economy are having visible impact on the firms that we visited, all of which are still under pressure to develop and deliver software at Internet speed. These firms are operating with significantly lower capital than before and are no longer finding it difficult to hire or retain the talent they need. In fact, most of the companies described their situations as either holding back or cutting down on their employees. Polyhymnia observes: "We're in a holding pattern. We still have a lot of projects but we're trying to weather the storm." Melpomene remarks they are "now holding steady with their head count until we're on even financial footing. We had a 20% cut in the summer. Some folks leaving are not being replaced or another part of the company may get the slot. Layoffs were across the board."

Changes in the IT economy have been especially noticeable with respect to personnel and staffing. While it was extremely difficult to find and keep experienced staff in 2000, now "Java developers are a dime a dozen," said a manager at Melpomene.

Experience. Also contributing to the current state of the practice in Internet software development is the hard-won experience gained during both the dot-com boom and the dot-com bust. Development staff at the companies we visited in 2000 expressed boundless energy and excitement (and to an extent a degree of confusion) about *what* they were supposed to be doing and *how* they were actually supposed to do it. By 2002 a more mature and reflective perspective was evident. One member at Melpomene suggests that caution contributed to the company's survival: "I am critical of the phrase 'Internet speed'—about the dot-com craze and demise. At that time, there was excitement. All the successful practices were successful because they were good practices. We made conservative decisions and are still here."

Significant learning has taken place in these companies, across a wide range of business and technical topics, over the last eighteen months. This experience has been gained in a variety of areas, including technology, tools, software development process and project management. A member of Thalia sums up the learning when he remarks: "Products are getting better due to more experienced developers."

6.8 The Portfolio Level

There are three categories associated with the Business Solution Set. These include business models, different projects, and a cost focus.

Business model. As indicated, the changing market and the IT economy have increased attention on business models with somewhat less emphasis placed on the urgency of being first to market. One organization illustrates the shift in matter of fact terms: "What has changed? We don't waste time on things that don't generate revenue." (Thalia) New business constraints have affected staff and contract personnel in a number of ways: "The personnel situation has also been quite frustrating recently. First we were hiring then firing." (Urania) "Over 2001 we have weaned ourselves off consulting groups. We've converted consulting requisitions to full time positions. The market has changed. You can get as good experience in a full time person." (Polyhymnia)

Market fluctuations have demanded that companies continue to define and refine their business offerings. Some companies have harkened back to more conventional service models, recognizing "Success from now on depends on being a software and services company rather than an Internet company." (Melpomene) Other companies acknowledge market trade-offs and volatility, including growth in some areas and slowing in others. A manager in Polyhymnia says: " [our] income depends on travel transactions. Agency business [is] not picking up but ecommerce is growing." For many of these companies, benefits have been accrued by sheer survival; for example, Melpomene states, "Several competitors have disappeared over the last year or so."

Different projects. The criteria by which the organizations select and fund projects have undergone a major change. As opposed to the days of abundant resources where risky projects even with faint hopes of success were entertained, organizations are more and more concerned about funding projects that do not have a clear business case. "We have to balance the need to do things fast and [our] desire to do it right – you need to have a business case." (Urania). Projects that do not demonstrate a clear contribution to the bottom line no longer have any chance of winning any support. In Urania, "For new projects there is more drive to prove your business case and show your value." Even on-going projects are not exempt from this requirement. If they cannot clearly demonstrate adequate returns, they are unlikely to be continued. A manager at Thalia laments that his "Product is expected to generate revenue. That is different than before. Now we need to make a business case for each project." This situation has encouraged project managers to clearly articulate the rationale for their projects, position them appropriately in alignment with organizational needs and requirements and in addition, market them to relevant

J. Pries-Heje, R. Baskerville, L. Levine & B. Ramesh • 31

decision makers. In Deca, this new reality was recognized as a serious challenge "Business side – positioning the products in the niche – marketing properly is a big challenge."

Cost focus. All of the companies are refining their identities and offerings through the creation and use of new business models. These models are affecting strategy and product decisions and are placing an increased emphasis on expenditures (e.g. cost) and revenue generated. Thus, one of the most dramatic results of the changing market and IT economy is the companies' new focus and adaptation of a set of business solutions, including development and use of business models, exploitation of multiple and different projects, and emphasis on cost. These challenges are being tackled by different mechanisms. For example, forming business partnerships with external development organizations is one way to keep the costs under control. Or as they said at Deca: "Teamwork with our partners is pretty important to us."

Take customer needs into account. Customer needs and customer expectations—for speed and for higher quality—carry increasing clout in 2002. The voice of the customer is ever present, expressed through product strategy concerns, relationships, and ongoing contact: "Product strategy gets bombarded with customer requests." (Melpomene) **"**Keep the customers happy, and more than that, we are a partner, not a vendor." (Urania) "Many clients appreciate us talking to them. They feel good about having influence on the product. We try our best to act on what they want." (Melpomene) "In the last project I was on we had several checkouts throughout the project, where we taking things out and showing them to customer: 'Is this what you want?'" (Urania)

Customer expectations are growing and the voice of the customer is louder than we have seen earlier. One organization—Urania—reported, "The speed hasn't changed. If anything it gets faster and faster as customer expectations grow. … [The biggest challenge] is meeting your customer's expectations." Another (Thalia) observed the effects of product maturity and customer volume: "…we have more customers now. Our product is gaining popularity. Sales force is hit with more complaints and this means more changes." Yet another (Melpomene) remarked on outspoken customers: "Our customers have no choice on upgrades. We upgrade and they get it. If there are problems, you will know it because they are very vocal."

Clearly, customer needs and expectations are growing and are being explicitly expressed. While speed is not the most dominant issue, customers are an important concern, "Our customer is looking for us to go as fast as we can." (Urania) Customer needs remain a challenge to satisfy, and are sometimes difficult to discern, however customer expectations are significantly higher than we saw in 2000. A manager in Euterpe observed the following about customer

expectations and satisfaction: "Customers do not explicitly ask for quality but want us to 'bring me a rock,' so [we] have to discuss how to handle errors – if it meets the customer's original need, then they're happy. Doesn't matter what other features are there, as long as what they want is there."

6.9 The Project Level

The project level categories are manifested by the set of technical solutions. There are seven categories in this set.

Stick to a standard architecture. Architecture is one element of the technical solutions set. The criticality of architecture in developing stable, scalable and evolving applications is well recognized. The development of a standardized architecture across applications enables the development of a family of products that have common features, and the reuse of components across the members of this family. Also, a standardized architecture is helpful in integrating various parts of a system. In Deca "work products are sold as a suite: Seamless interfacing and integration. Custom developed current architecture is being extended so that products have similar design." In the absence of common architecture it may not be feasible to integrate components. Further, there was recognition that the maintenance of systems developed without a standardized architecture can pose serious difficulties.

Many organisations use standardised tools and development environments to implicitly impose an architecture and to achieve scalability. For example, in Thalia the system had to be re-architected such that the "New user interface has made things simpler. … We broke it into a front end and ASP, separating presentation layer from business logic." Urania is focused on integration of components and systems developed that requires an open architecture: "We develop a lot based on out-of-the-box applications. It has changed the way we develop systems. We probably spend more time adapting and for example making them more open that we did before."

In our study of team factors, we sometimes found that teams were constructed according to the architecture in place. For example, Melpomene organized around two team functions. One team handled the development of user interfaces, while another team worked the business logic and technical issues such as database changes. In such cases, we can see that architecture provided a framework for the division of labour in these development projects.

Use components and reuse. Component use is another element of the technical solutions set. The speed with which Internet software has to be developed makes the assembly of components rather than developing products from scratch the only viable option. A well developed architecture makes integrat-

J. Pries-Heje, R. Baskerville, L. Levine & B. Ramesh • 33

ing components at all levels of the architecture (business logic, interfaces and back-end infrastructure) feasible. "We do use components — web logic server and web logic commerce server," remarked a developer at Polyhymnia.

Products can be differentiated for different market or customer segments using add-on components. For example at Melpomene: "We have one product and only one instance of the software running at any time. … Core product is for the HR user. Then add-ons for employees and managers."

In many cases, the very nature of software development just involves achieving interoperability and integration among components developed elsewhere or purchased off the shelf. A software development manager in Thalia notes that they are simply "acting as a middleman. Putting components together to create systems is fast."

The development of generic components that can be used across multiple products is more expensive in the short run even though it may be more economical and efficient in the long run. In addition, development of generic components rather than products meeting immediate needs is likely to require more time, a scarce resource in Internet Speed development. However, this constraint is balanced against the flexibility offered by component-based development, as well as the ability to reuse the components across a variety of products.

Carry out estimation. Estimation is part of the set of technical solutions. A major difference between the first and second rounds of our study was the recognition of the need for good estimation methods to track and improve performance. The organizations that have been involved in Internet software development for a few years declare that they are more mature in their estimation of effort and schedule: "We know what its like to develop in this environment" (Polyhymnia).

The ability to make better estimates on costs and schedules helps in making decisions about the feasibility of projects. A manager in Polyhymnia said, "We are getting better at doing estimates. We used to just do projects. … Now, there are projects that we don't do." This ability to estimate has helped development organizations to convince their customers to develop realistic expectations.

"We used to have much greater problem when product strategy will ask for 50 things and we will say we can do five. Doing detailed analysis was a waste when we have that many things. Now they ask for six or seven things. They have learned a lesson. As a result, they have a better sense of cost of what they are asking for," they told us at Melpomene.

The variables that are estimated and tracked vary from organization to organization. For example in Deca, "We are starting to measure cycle time and

34 • J. Pries-Heje, R. Baskerville, L. Levine & B. Ramesh

cost. They are focused on KPAs [CMM key process areas] at level 2 that need the most attention—requirements management and project planning & tracking, and at level 3—inspections." Some organizations have well developed practices that enable them to estimate and monitor variables at a finer level of detail. At Melpomene for example, a manager said, "I have been tracking for several releases. I do estimation based on screens, wizards involved etc. And I rely heavily on past performance."

Improve and involve QA and testing. Quality and testing comprise another element of the technical solutions set. In Internet speed software development projects where the customers were willing to accept less quality for speed, the role of QA and testing were minimized. With markets and products maturing, there is more and more recognition for ensuring quality during development. QA and testing are beginning to get more attention from development teams and are gaining recognition as important aspects of software development. A senior manager in Polyhymnia notes, "As a QA organization, we are gaining respect across the company. Historically the QA was positioned differently. Initially there was no QA manager. In 1998 the QA manager was made director of QA, a direct report to the VP. The challenge is to keep that up, and keep from being bypassed."

The newfound respect for QA in Internet speed development is not without problems. Tension that exists between QA and development teams in traditional development environments, in fact, may get accentuated in the time-constrained setting experienced by the organizations we studied. In Melpomene, for example, "There is not a close relationship between QA lead and design lead. There is some interworking between departments. The two job descriptions are adversarial. The QA's job description is to break what we build." QA may be seen as a process that slows down development, and the resources allocated to QA (in relation to development) may be seen as unnecessary overhead. "The QA phase is longer than development phase and has more people." (Melpomene)

To make the process of QA more efficient in a time-constrained setting, the concerns of the QA process are beginning to be addressed early in the design process itself. In organizations such as Thalia, "The QA group has a representative on each project give their input like a test plan." However, in other organizations the tensions that exist between development and QA may prevent this from happening.

In a time-constrained environment, the need to make QA more efficient, especially to complete the process in a timely manner is highlighted. "If I look at a project timeline, a lot of it is in QA testing. We need to improve and automate and create scripts to drive that down" (Thalia).

J. Pries-Heje, R. Baskerville, L. Levine & B. Ramesh • 35

Parallel development. Parallel development is another element of the technical solutions set. Competitive pressures force release cycles to be compressed to such an extent that overlapping, parallel development is necessary. Also customers are accustomed to fast release cycles and they expect them even though the complexity of the products (and hence the complexity of development) has increased significantly. Fast release cycles can be achieved by developing entire releases in parallel or development staged such that design, implementation, and quality assurance, are all taking place simultaneously, but sequentially on different releases. This may follow a staged waterfall model with different releases at different phases of the lifecycle. "Basically we have a waterfall model. … In a project we were trying to overlap phases. We work in parallel across releases. We have large teams where one part is in construction, another is in planning and analysis" (Urania).

Handling multiple releases simultaneously causes some interesting problems. Staffing multiple releases is a challenge especially when adequate numbers of personnel with critical skills are not available. Though some organizations find it effective to keep personnel working on a version from start to finish, others are organized around areas of expertise where the people move in and out of the project depending on when their skills are needed, as for example at Melpomene: "We have the two teams working in parallel. However, there is lot of fluidity between teams. Someone can move from one release to another."

Managing the dependencies across the code base in different releases is another challenge, especially when these releases are not developed from scratch. A manager in Thalia observes, "dependencies are difficult to manage." A strategy they use to keep the dependencies low is to keep the changes localized: "Try to focus releases on certain functionality. We are pulling code in and out of releases. We have parallel development." Another strategy used to manage parallel development is to integrate QA with development: *"*Developers do more unit testing to free QA from doing this and have more time to attend to integration tests" (Melpomene).

Another problem created by working on several versions at the same time is that some teams working on early releases of the life cycle may not even see their work realized in production versions of the product. In Melpomene, a user-interface developer was frustrated: "Some stuff we were working on two cycles ago is not yet in the product," and that lack of feedback from customers due to this delayed feedback may in fact make some of their work irrelevant. At the extreme, coding may even take place before the requirements have been fully understood, with the anticipation that the users will indeed request the features that are being implemented. However, it is quite possible for a feature

36 • J. Pries-Heje, R. Baskerville, L. Levine & B. Ramesh

that is almost fully implemented is not required at all or phased out to a later release to meet the current needs of the customer.

Prototyping. Prototyping, an element of the technical solutions set, is used as an efficient way to understand user requirements and get immediate feedback from customers as the development proceeds. Prototypes that implement the basic functionality or features that are demanded by the customers are used to validate and refine requirements. In some cases, these prototypes are focused only on the user interfaces. A Urania manager commented, "We use prototypes to get a project going. … Bringing it out to the customers to see what they think. Mock ups, smoke and mirrors used to get customers interested. Prototypes are more for GUI purposes. The functionality isn't emphasized."

To the extent possible, throwaway prototypes are avoided. Operational prototypes are evolved into production prototypes by refining the code created for experimentation with necessary features. The rush to market and the ability to replace products when problems are discovered encourages a tendency to deploy the prototypes rather than wait for robust implementations of the desired functionality. However, in many instances it is just not feasible to reuse prototypes created due to considerations of efficiency, scalability and performance. "We try to build on top of the prototype and not make it a throwaway—we use at least the web page designs in the final system." (Urania) One problem with the use of prototyping is the redundancy between prototyping and development, when the prototypes cannot be readily evolved into the final product. In Melopomene, for example, this problem is mitigated by using a common platform and technology for both prototyping and production code. "User interface programmers may start to implement production interfaces."

Frequent releases. Frequent releases represent another element of the technical solutions set. The requirements can be kept fluid because of the constant monitoring and prioritization of the features that will be included in the product in successive releases. Often customers need to wait only a few weeks before their next important feature is supported and the development process is "flexible enough to make changes multiple time a year." (Melpomene) The close involvement of the customers with the project facilitates such dynamic scoping of releases. The frequent release cycles are considered an important characteristic of operating in Internet speed by some organizations. "We do patches outside usual release, once a month or so. Compared to others in this space, we are doing sizable releases two to three times a year. It is Internet speed" (Melpomene).

Though during early stages of introduction of a product, releases are made very frequently, as the product matures, release cycles get longer to be able to

J. Pries-Heje, R. Baskerville, L. Levine & B. Ramesh • 37

accommodate the need for quality in the product. Customer requirements for robustness and stability appear to be related to the maturity of the product and the size of the customer base. When the product is more mature and has attracted a large client base, the need for quality factors such as security, stability and robustness seems to overtake the need for speed. The release cycle times vary widely, for example in Thalia there is "generally one enhancement release every month" interspersed with one defect release. There were much fewer in other instances. In fact, in Polyhymnia the product releases have to be synchronized with the development of backend systems, causing the frequency of releases to be reduced dramatically: "Most projects are moving to enterprise releases twice a year." Urania has developed a portfolio of projects with different needs to develop different release strategies. Here 'Type 2 releases can take from one to three or four months. Type 3 work is typically six months to a year. Type 4 can be anywhere from six months to more than two years."

7 Discusion

It is possible to logically link any of the categories above to the major balancing game factors of time, speed, scope and resources. However, in this analysis we would like to draw out certain prevalent effects that demonstrate the interdependence of the changing values of the factors at all three levels.

7.1 Market Level Values

During the period of the two studies, we saw values shift from a fever-pitched struggle for first-mover market advantage to a slower, less intense consolidation of best practices. Notably the changing market and IT economy eased the interest in IT products, while at the same time easing the intense competition for human resources necessary for wide-scale software development. With the market focused on a narrowed scope of Internet applications, competition remained intense but concentrated. Resources disappeared from all but a few revenue points within this narrow scope. With the focused competition on these points, the market began to value quality more highly than before. Development activity began to fall, and more experienced personnel became available. Indeed, organizational and individual experience improved and higher quality developers were more accessible, providing access to quality. In this analysis, we can determine major changes in the market values for Internet application quality, scope, and resources.

7.2 Portfolio Level Values

During the period of the two studies, we detected a shift in these values from a resource-rich, build-everything blast to a resource-constrained, tightly managed and well-organized stable of ideal jobs. As a result of the changing values in the market, the companies in our study began to make major adjustments to their project portfolios. The *business case* became the primary mode for apportioning resources and selecting projects for inclusion or continuation within the scope of the portfolio. With falling resources, managers began to "cherry pick" the most ideal projects to meet their customer's needs. At the portfolio level, the scope of the portfolio narrowed without necessarily eliminating the importance of quick and timely delivery of new features within the narrow scope. At the portfolio level, the demand for Internet speed development remained strong. It simply became focused in on a limited range of important projects. The ability to transfer development experience and knowledge across projects became valuable for improving all of the projects remaining in a smaller portfolio with more limited resources. As a result of the change in market values, available resources declined, the portfolio's scope focused, yet the remaining projects still had to respond rapidly to the customer needs in an continuingly intense but focused market.

7.3 Project Level Values

Because many of our respondents were operating at the project level (project managers and developers), we have more detailed data and analysis at this level. Here we see the project values move from speed-at-all-costs to an economized scope. Denied the resources to build products without a clear economic justification, project managers began to consolidate the product development to embrace the competitive construction of fewer products. Thus we see the major Internet speed development values continuing, such as parallel development, limited maintenance and documentation, frequent releases, etc. These factors are still necessary to maintain customer satisfaction and compete in the (more focused) marketplace. However, the Internet speed development values take on new importance as the project values shift. Most apparent in factors like standard architectures and component reuse, the factors are also noted for enabling quick, economical products. The cost-savings and responsiveness are valuable in order to keep the project costs down. The lower project costs make the business case workable at the project portfolio level and help keep the software product viable. Also notable was the increased emphasis on quality assurance and testing. While not readily appar-

ent at the portfolio level, the customer contact at the project level still triggered a small rise in this value as a direct impact of the market on the project level. At the project level, the time value remains strong, the importance of quality rises slightly, the feature scope narrows, and the available resources are more limited.

7.4 Changing Software Development Practices

The shifting values in the balancing games at the three levels affected the practices we found in 2000 and 2002. Certainly some of the Internet speed software development practices continued through both periods. Other innovative practices seemed to migrate toward their more traditional forms that predated Internet speed development. Some practices appeared on the 2002 Internet speed scene that were not part of the central story line in 2000.

Table 3 details a comparison of the 2000 Internet speed software process and the 2002 technical solution set. Five elements are common in both sets of practices including parallel development, release orientation, components/reuse, prototyping, and the use of stable architectures. This set of practices preserves the polychronicity in the software process and demonstrates, as discussed above, that the balancing game at the project level has not entirely changed.

Four 2000 practices disappeared from the central story line in 2002. While this disappearance does not imply that the practices have entirely disappeared, it does mean that the practices have become marginalized in the reflections of Internet speed developers. These practices are much less important in 2002. Included here are tool dependence, customer involvement, maintenance ignored, and tailored methodologies. The shift in tool dependence as a central practice at the project level can be explained by the shifts in the market level that made more experienced developers available for projects. Experienced and well-trained developers will certainly take advantage of tool suites, however, the dependence is reduced with their capability to choose when and how to use the tools. Tools are more taken-for-granted at the project level because of the shifts at the market level.

The focus on tailored methodologies in the 2000 story line may have similarly disappeared. In 2002, changes in the market level balancing game led to more experienced developers being provided at the project level. Experience lessens the dependence of developers on methodologies. While methodologies were certainly still in use in these cases, the centrality of practices for tailoring the processes was relaxed through the higher degree of experience found in the 2002 Internet speed project developers.

2000 Characteristics of the new software process	2002 Technical Solution Set	Project Level Comparison
Parallel Development, Release Orientation and Components based development and reuse and Prototyping and Criticality of Architecture	Parallel Development, Frequent releases and Components/reuse and Prototyping and Architecture	These 5 solutions were commonly found in both 2000 and 2002
Tool dependence		Less dependence on tools in 2002. More developer experience reduced the importance of tool suites found in 2000.
Customer involvement		Received a lot of focus in 2000. Was less distinct in 2002. Probably because of the increasing customer influence in project selection
Maintenance ignored		The market place changed from 2000 to 2002. The focus on speed shifted to a focus on both speed and quality. Companies could not completely ignore maintenance for quality reasons.
Tailored methodology		Improving developer experience reduced the importance of methodology, and consequently the importance of tailoring methodology.
	Estimation methods	The 2002 portfolio level focus on cost and benefits lead to much more emphasis on project management – and especially estimation – in 2002.

Table 3. Comparison of the 2000 Internet speed software process and the 2002 technical solution set

J. Pries-Heje, R. Baskerville, L. Levine & B. Ramesh • 41

	QA and Testing	In 2000 it was possible to "negoti-ate" low quality in the marketplace. That was not as acceptable in 2002. Thus QA and Testing at high speed was of distinct importance.

Table 3. Comparison of the 2000 Internet speed software process and the 2002 technical solution set

The disregard for maintenance also disappeared from the central story line in 2002, and indeed we find the need for QA and testing has replaced this practice. Shifts in the balancing game at the market level, viz., the increasing importance of quality in the competitive marketplace, certainly helped to drive this shift in the balancing game at the project level. The negotiation of quality in relation to development speed gave way to a marketplace that demanded both quality and speed.

Customer involvement ceased as a central element in the 2002 practice set. We find this interesting because the data and our observations clearly indicated that customers were still brought into the development process. However, it appeared that part of the customer involvement had shifted to the portfolio level. As 2002 organizations rebalanced their portfolios in light of business cases, the customers exercised considerable influence at the portfolio level in shaping these business cases, decreasing the somewhat strategic impact of their involvement in the project level (as we had seen in 2000).

Similarly to QA and testing, estimation methods appeared in the 2002 story line, and had not been a central part of the practices in 2000. One of the major changes in the portfolio level balancing game, the emphasis on business cases, has likely influenced the increased attention to estimating the parameters of projects. Business cases usually compare costs and benefits, increasing the role of practices in making estimations at the project level. This shift at the portfolio level affected the balancing game at the project level.

7.5 Relationships to Other Trade-Off Theories

The comparison of changing practices represented in the 2000 and 2002 story lines illustrates the way in which the balancing games interact across the three levels. Changes in the balance in the market and portfolio games affect the way factors are balanced in the project game. As a result, practices are directly affected and shift as the project balances shift. This comparison provides insight into how organizations balance values in games at three levels. The levels operate in a consequential fashion, with the market level balancing

game constraining the operation of the portfolio balancing game, which in turn constrains the project level balancing game. Balancing game theory helps to explain the mechanisms by which software marketplaces affect software processes. Balancing game theory extends a number of previous theoretical views that explained trade-offs between values at one level or another, but are less effective at explicating the multiple layers of such trade-offs.

A well known theory regards the balance between exploration and exploitation as strategies for organizational learning (March 1991). *Exploration* is a strategic posture that promotes research and experimentation. Exploration strategies place high values on creating variation in an organization's products, and accept high risk in regard to the potential returns. *Exploitation* is a strategic posture that promotes the use and development of products that are becoming familiar and popular (Levinthal and March 1993). Balancing game theory helps us to understand the mechanisms by which software organizations manifest the selection of exploration or exploitation. In the portfolio level balancing game, managers assign values to projects depending on whether they want to create value (exploration) or appropriate value (exploitation). An inclination to exploration would be reflected by values in the portfolio balancing game that promoted experimental projects (Sutton 2004). An inclination to exploitation would be reflected by values in the portfolio balancing game that promoted software process improvement (Benner and Tushman 2003). In light of the balancing game theory, it seems obvious that strategic decisions regarding the value placed on exploration versus exploitation are responding to shifts in the marketplace, and imply shifts in project management values. However, these extra-portfolio relationships are much clearer for the development of this theory when placed in the context of a multi-layered balancing game that includes portfolio decisions.

Another theory related to trade-offs between software development projects is strategic alignment theory. Alignment theory regards the synchronization of organizational strategy and IT strategy, for example, in terms of how the nature of the product fits with the process for producing this product (Hayes & Wheelwright 1979). The theory reveals how IT strategies that are misaligned with organizational strategy are less effective and less efficient at achieving organizational goals. When considered in light of balancing game theory, it is clear that alignment theory operates across all three levels of the balancing game. The strategy for the product is clearly derived from a market level balancing game, while the strategy for the process is largely determined in a project level balancing game. Balancing game theory helps us to understand that, in terms of software development, there is also a mediating game at the portfolio level in which the product and resource marketplaces affect a

portfolio decision about which products are to be selected for development. This portfolio balancing game mediates the alignment between the product characteristics (the market-level balancing game) and the process characteristics (the project level balancing game).

An understanding of the mediating balancing game can help illuminate seemingly irrational strategic misalignments. For example in 2000 Thalia had a large customer base and a cost leadership strategy, which shaped its market level balancing game decisions. However, its processes were semi-customized, and theoretically misaligned with large scale cost leadership. Our understanding of the rationale for this seemingly misaligned strategy is improved in the light of the portfolio balancing game at the time. The firm had decided to promote innovation in order to compete head on with a smaller firm that was rapidly nibbling away market share through customized products. To inhibit this competitor, the values in the portfolio balancing game were shifted to privilege small and nearly customized products. At the project level, this decision was most evident in the lack of a coherent, standard and modular architecture for its Internet applications. In effect, this mediating project level balancing game ignored (assigned no value to) architecture projects. In this setting, the mediating balancing game at the portfolio level had substantial impact on the project level balancing game that appeared out of sync with the market level balancing game. Alignment theory suggests that the strategy is misaligned. Balancing game theory helps explain why the misalignment was needed, and how the misalignment affected practice.

A third theoretical framework that has been useful for understanding innovative software practice regards the operation of disruptive innovations, such as Internet speed development, on different aspects of the systems development arena (Lyytinen and Rose 2003). For example, a major shift in a base technology like the advent of commercial availability of the Internet, leads to transformational innovations in systems development and the consequent systems being developed. While this work recognizes the interactivity between these three aspects of the innovation arena (base technology, systems development, and systems), the perspective is chiefly focused at the project level and does not extend explicitly to balancing games at higher levels. When regarded from the viewpoint of balancing game theory, it is clear that innovation in systems development is being modelled as the consequence of either innovations in systems applications or innovations in base technologies. Changes in the base technology may indeed drive part of the innovation. However, changes in the market and the business model of the organization may also drive the way the organization balances its activities and resources with regard to development projects. The enabling of such innovations through allocation values

44 • J. Pries-Heje, R. Baskerville, L. Levine & B. Ramesh

at the portfolio level (or the market level) is part of the assumption space for disruptive innovation theory. Balancing game theory reveals the mechanisms that shape these values and thereby reform the assumptions on which the existing theory is based.

In addition, balancing game theory contributes additional insights into the way in which organizational software development practices can recycle in response to market shifts. As Internet speed software development emerged during 2000-2002, we documented the arrival and departure of certain important practices, such as disregarded maintenance and negotiable quality. Like other Internet speed practices that arrived and remained, these transient practices were not mistakes, but necessary elements that responded quite well to the balancing games. Disruptive innovation theory helps explain how these practices arrive, while balancing game theory adds additional insight into how the practices can arrive, and sometimes remain or depart. Balancing game theory helps understand why some clearly needed and advantageous new practices might eventually, and happily, recycle back to their predecessors.
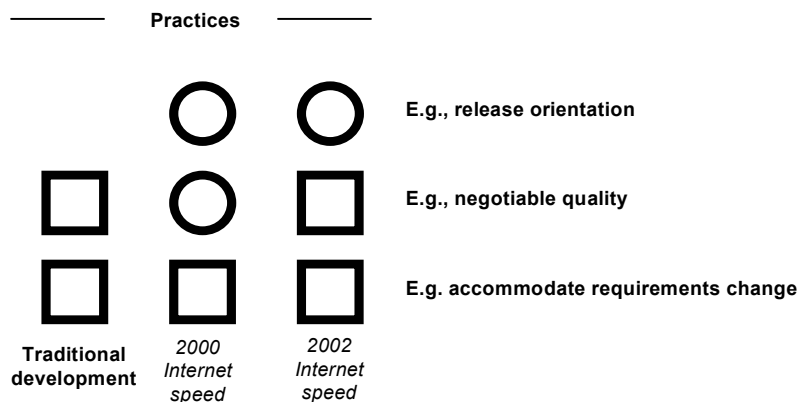

Figure 4. Change and recycle framework; *key:* square: traditional practice; circle: new form of practice

Figure 4 illustrates this change and recycle framework. Balancing game theory helps explain why and how some innovative practices arrive and remain. An example is release orientation, which was shown to be prevalent in the 2000 story line, and remained as an important practice in 2002. Before release orientation, traditional practice developed software against complete specifications. It also helps explain why other innovative practices arrived, but recycled back to the original practices. An example is the disregard for technical quality (negotiable quality) that was important in 2000, but disappeared as

J. Pries-Heje, R. Baskerville, L. Levine & B. Ramesh • 45

the focus on QA and testing had returned to central practice by 2002. While our focus in this article is not on the many traditional software practices that have been preserved in Internet speed development, an example of a practice common to Internet speed development and more traditional forms of development is the accommodation of requirements change (see Baskerville, et al, 2003). Disruptive innovation theory helps explain transitions such as release orientation. Balancing game theory not only provides additional insight into how these transitions unfold, but also provides insight into how other practices, such as negotiable quality, recycle to earlier forms.

8 Conclusion

IS development organizations are operating under the constraints of multiple dimensions including quality, speed, and cost. For example, in many instances customers are willing to accept lower quality for innovation and speed, especially during the early stages of development of innovative products or services. However, even when user expectations are initially low, as the products mature these attract a larger customer base. The nature of the intense competition in a large market place demands that additional quality features be incorporated in the product. Thus, IS organizations are forced to constantly juggle multiple priorities and constraints to achieve the agility required to meet competing demands.

In this study, we have identified a balancing game theory that explains how some of the key organizational and technical factors affect IS organizational decisions meant to achieve success in such environments. Our study suggests that trade-offs and balancing decisions—which we call balancing games—are taking place at three different levels: the market level, the portfolio level, and the project level. The nature of the balancing game has evolved with the evolution of the market and organizational environments over recent years. The peak of the dot-com boom was characterized by very few constraints on financial resources, but severe constraints on availability of qualified personnel and very tight deadlines. At this peak, the balancing game was focused more towards achieving speed, often at increased project costs and lower levels of quality. This situation later evolved into market conditions that expect higher levels of product quality and lower costs while still demanding product development agility. The research reveals how several marketplace developments help organizations manage these high demands, such as the availability of qualified personnel who have more experience with fast paced development methods and tools. As a result of these marketplace changes, the balancing

46 • J. Pries-Heje, R. Baskerville, L. Levine & B. Ramesh

games at the organizational and portfolio levels have grown in importance compared to the dominance achieved by the project balancing game in 2000.

While balancing game theory does not refute major theoretical propositions about the way organizations make trade-off decisions relevant to software development (such as exploration-exploitation theory, strategic alignment theory, and disruptive innovation theory), it does extend the operation of these propositions into several levels, and provides additional insights into exactly how these trade-offs are executed in software companies. Balancing game theory helps us to understand how strategic trade-offs get translated into changes in software development practices.

The development of balancing game theory raises new questions and opens new research opportunities. Such research opportunities arise in potentially valuable insights that the application of balancing game theory could develop in other application arenas. The focus of the study described in this paper was commercial products and services that evolved following the commercial availability of the Internet. Many of these practices are only just reaching public arenas in the form of e-Government. It is not clear that the marketplace balancing game would operate in the same fashion in public services. Additionally, new technologies are driving m-Commerce (mobile information systems). There are opportunities for confirmatory studies in this arena. Are the balancing games operating in the same fashion for new systems arising in the wake of rapidly advancing and revolutionary mobile information technologies?

There would be benefit from extending the empirical work about the various levels in the balancing game. Our findings are grounded in data that is primarily collected from respondents who operate at the project and portfolio management levels. The market impact is represented in this data because it could be perceived by many of the respondents at the project level. Additional empirical work that directly assesses the market level balancing game in relation to the portfolio and project balancing game could be achieved with a larger set of respondents at the executive level of software development organizations. Such assessments would more solidly confirm the balancing game theory at the market level. At this level, it may be possible to detect additional levels of the balancing game that exist beyond the marketing level balancing game. Along these lines, we should note that the levelled balancing game only became recognizable when our research reached across two time periods. The 2000 one shot study alone did not reveal the levels in the balancing game. More extensive longitudinal research might also reveal other levels or other dimensions in the balancing game. Our research, however, does not reach beyond the three levels discussed above.

J. Pries-Heje, R. Baskerville, L. Levine & B. Ramesh • 47

Acknowledgment

References

Aoyama, M. "Web-based agile software development," *IEEE Software* (November/December) 1998, pp 56-65.

Barley, S. "On technology, time, and social order: Technologically induced change in the temporal organization of radiological work," in: *Making Time: Ethnographies of High-Technology Organizations,* F.A. Dubinskas (ed.), Temple University Press, Philadelphia, PA, 1988, pp. 123-169.

Baskerville, R., and Pries-Heje, J. "Racing the e-bomb: How the Internet is redefining information systems development methodology," in: *Realigning Research and Practice in IS Development: The Social and Organisational Perspective,* B. FitzGerald, N. Russo and J. DeGross (eds.), Kluwer, New York, 2001, pp. 49-68.

Baskerville, R., Levine, L., Pries-Heje, J., Ramesh, B., and Slaughter, S. "Is Internet-speed software development different?," *IEEE Software* (20:6) 2003, pp 70-77.

Baskerville, R., and Pries-Heje, J. "Short Cycle Time Systems Development," *Information Systems Journal* (14:2) 2004, pp 237-264.

Beck, K. *Extreme Programming Explained: Embrace Change* Addison-Wesley, Boston, 2000.

Beck, K., and Fowler, M. *Planning Extreme Programming* Addison Wesley Longman, New York, NY, 2001.

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

Boehm, B. "A Spiral Model of Software Development and Enhancement," *IEEE Computer* (21:5) 1998, pp 61-72.

48 • J. Pries-Heje, R. Baskerville, L. Levine & B. Ramesh

Boehm, B., Egyed, A., Kwan, J., Port, D., Shah, A., and Madachy, R. "Using the win-win spiral model: A case study," *IEEE Computer* (31:7) 1998, pp 33-44.

Brown, J.S., and Duguid, P. "Creativity versus structure: A useful tension," in: *Mit Sloan Management Review*, 2001, pp. 93-94.

Conallen, J. *Building Web Applications with UML* Addison-Wesley, Boston, 1999.

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

Cusumano, M.A., and Yoffie, D.B. "What Netscape learned from cross-platform software development," *Communications of the ACM* (42:10) 1999, pp 72-78.

Fayad, M.E., Laitinen, M., and Ward, R.P. "Software engineering in the small," *Communications of the ACM* (43:3) 2000, pp 115-118.

Fowler, M. *Refactoring: Improving the Design of Existing Code* Addison-Wesley, Boston, MA, 2000.

Hall, E.T. *The Hidden Dimension* Anchor Press, New York, 1966.

Harter, D., Krishnan, M., and Slaughter, S. "Effects of process maturity on quality, cycle time, and effort in software product development," *Management Science* (46:4) 2000, pp 451-466.

Hayes, R., and Wheelwright, S. "Link Manufacturing Process and Product Life Cycles," *Harvard Business Review* (57:1) 1979, pp 133-140.

Highsmith, J. *Adaptive Software Development* Dorest House, New York, 1999.

Highsmith, J. *Beyond RAD: Reducing cycle time through innovative management* Cutter Information Corp, Arlington, MA, 1999.

Jeffries, R., Anderson, A., and Hendrickson, C. *Extreme Programming Installed* Addison-Wesley, Boston, 2001.

Krishnan, M., Kekre, S., Kriebel, C., and Mukhopadhyay, T. "An empirical analysis of productivity and quality in software products," *Management Science* (46) 2000, pp 745-759.

Kruchten, P. *The Rational Unified Process: An Introduction* Addison-Wesley, Reading, MA, 1998.

Laitinen, M., Fayad, M.E., and Ward, R.P. "The problem with scalability," *Communications of the ACM* (43:9) 2000, pp 105-107.

Lee, H. "Time and information technology: Monochronicity, polychronicity and temporal symmetry," *European Journal of Information Systems* (8:1) 1999, pp 16-26.

J. Pries-Heje, R. Baskerville, L. Levine & B. Ramesh • 49

Lee, H., and Liebenau, J. "Time and the Internet at the turn of the millennium," *Time & Society* (9:1) 2000, pp 43-56.

Lee, H., and Sawyer, S. "Conceptualizing time and space: Information technology, work and organization," The International Conference on Information Systems, ACM Press, Barcelona, Spain, 2002, pp. 279-286.

Levine, L., Baskerville, R., Loveland Link, J.L., Pries-Heje, J., Ramesh, B., and Slaughter, S. "Discovery colloquium: Quality software development @ Internet speed," SEI Technical Report CMU/SEI-2002-TR-020, ESC-TR-2002-020, Software Engineering Institute, Pittsburgh, PA.

Levinthal, D.A., and March, J.G. "The myopia of learning," *Strategic Management Journal* (14) 1993, pp 95-112.

Lyytinen, K., and Rose, G.M. "The disruptive nature of information technology innovations: The case of Internet computing in systems development organizations," *MIS Quarterly* (27:4) 2003, pp 557-595.

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

Miner, A.S., Bassoff, P., and Moorman, C. "Organizational improvisation and learning: A field study," *Administrative Science Quarterly* (46) 2001, pp 304-337.

Newman, W.S., Podgurski, A., Quinn, R.D., Merat, F.L., Branicky, M., Barendt, N., Causey, G., Hasser, E., Kim, Y., Swaminathan, J., and Velasco, V. "Design lessons for building agile manufacturing systems," *IEEE Transactions on Robotics and Automation* (16:3) 2000, pp 228-238.

Orlikowski, W. "CASE Tools as Organizational Change: Investigating incremental and Radical Changes in Systems Development," *MIS Quarterly* (17:3) 1993, pp 309-340.

Paulk, M. "Extreme Programming from a CMM perspective," *IEEE Software* (18:6) 2001, pp 19-26.

Ramesh, B., Baskerville, R., and Pries-Heje, J. "Internet software engineering: A different class of processes," *Annals of Software Engineering* (14:1-4) 2002, pp 169-195.

Rising, L., and Janoff, N.S. "The Scrum software development process for small teams," *IEEE Software* (17:4) 2000, pp 26 -32.

Rosenberg, D., and Scott, K. *Use Case Driven Object Modeling with UML: A Practical Approach* Addison-Wesley, Reading, MA, 1999.

Shenhar, A.J. "One size does not fit all projects: Exploring classical contingency domains," *Management Science* (47:3) 2001, pp 394-414.

Strauss, A., and Corbin, J. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory* Sage Publications, Beverly Hills, CA, USA, 1990.

Strauss, A., and Corbin, J. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*, (2nd ed.) Sage Publications, Beverly Hills, CA, USA, 1998.

Sutton, R. "Why these ideas work, but seem weird," *Design Management Journal* (15:1), Winter 2004 2004, pp 43-49.

Thomke, S., and Reinertsen, D. "Agile product development: Managing development flexibility in uncertain environments," *California Management Review* (41:1), Fall 1998, pp 8-30.

Truex, D.P., Baskerville, R., and Klein, H.K. "Growing systems in an emergent organization," *Communications of The ACM* (42:8) 1999, pp 117-123.

Truex, D., Baskerville, R., and Travis, J. "Amethodical Systems Development: The Deferred Meaning of Systems Development Methods," *Accounting, Management and Information Technology* (10) 2000, pp 53-79.

Urquhart, C. "Exploring analyst-client communication: Using grounded theory techniques to investigate interaction in informal requirements gathering," in: *Information Systems and Qualitative Research,* A.S. Lee, J. Liebenau and J.I. DeGross (eds.), Chapman and Hall, London, 1997, pp. 149-181.

Urquhart, C. "Strategies for conversation and systems analysis in requirements gathering: A qualitative view of analyst-client communication," *The Qualitative Report (On-line serial)* (4:1) 2000.

Appendix: Research Methodology

Our approach follows established qualitative research methods. Semi-structured interviews were conducted with senior managers, project managers, quality assurance personnel, and software developers in the firms to elicit development methods and tools, quality issues, product issues, and issues surrounding development teams and people. An excerpt (in detail) from our interview guide is provided in Table 4.

A minimum of two and usually three researchers conducted all interviews. One concentrated on interviewing while one captured responses in short-hand notes. (Eliminating tape recorders from the interview setting encouraged frank

J. Pries-Heje, R. Baskerville, L. Levine & B. Ramesh • 51

Internet Speed

What does "Internet Speed development" mean to you? Are you engaged in it? How is it different/similar? Where are you spending your energies?

Product Issues

What primary products & services do you produce at 'internet speed'? Do they differ from other products in terms of complexity, scalability, connectivity or interfaces, or other characteristics? If so, how? Do you build individual products or product lines or families?

Quality Issues

What is quality to you? What are the most important quality dimensions for your products & processes? What QA processes are used for Internet development? Are these different from traditional development?

Team and People Issues

How is people/team management different at Internet speed? What roles are typical? How does knowledge transfer occur? What assignment rotation and coordination strategies are used?

Issues of Development Methods and Tools

Do your development methods, tools or platforms differ from traditional software development? How?

Timing Issues

Is planning for Internet products different? How? How is 'time-to-market' important?

Table 4. Excerpt from semi-structured interview guide

responses.) Immediately after conducting the interview, the notes were expanded in full text, consolidated, and reviewed by all the participant researchers. Typically the notes from one 2-hour interview came to 10-15 pages.

We used Grounded Theory methodology to analyze the interview data. Grounded Theory (GT) is a qualitative research methodology that takes its name from the practice of discovering theory that is grounded in data. GT is best used in research where one has relatively "uncharted land," as was the case with the notion of Internet Speed. Grounded theories are inductively discovered by careful collection and analysis of qualitative empirical data. That is, this method does not begin with a theory, and then seek proof. Instead, it begins with an area of study and allows the relevant theory to emerge from that area (Strauss and Corbin 1990). Use of GT in IT research is exemplified by a landmark paper by Orlikowski (1993) on CASE tools and organizational change, over explorations on software requirements (Urquhart 1997; Urquhart 2000), and in a study from Denmark on how the Internet is redefining information systems development methodology (Baskerville and Pries-Heje 2001).

52 • J. Pries-Heje, R. Baskerville, L. Levine & B. Ramesh

After collecting our data, we applied the three coding procedures of GT (Straus and Corbin, 1998) called open, axial and selective coding. These procedures do not occur entirely as a sequence, but each overlaps the others, making for iteration throughout the analysis. The coding procedures are described below, along with examples of what we did.

The goal of open coding is to reveal the essential ideas found in the data. Open coding involves two essential tasks. The first task is labelling phenomena. This task involves decomposing observations into discrete incidents or ideas. Each discrete idea receives a name or label that represents the phenomenon. These names represent a concept inherent in the observation. An example of our coding is shown in Table 5 below.

Example excerpt from interview at Urania	Open Coding
Person 1: "Also, following our methodology in the project is also important to ensure quality. Especially in a high speed environment, we may not have the time to document a few things."	Development: well-defined methodology exists Methodology requires quality
Person 2: "Also, make sure that the application in the business sense is of high quality, besides looking at system quality.	Two types of quality: Business and System
"In the business sense, make sure the application works properly, meets needs, and that you simplify the users experience with the application."	Quality also implying user experience
Person 1: "More thing to worry about is: when changes comes along you are able to accommodate by the quality of the design." Person 2: "Quality of environment is also important: meeting performance, stability of requirements.	Quality of environment important
"Quality of the design is key to having success in the application in the long term. We could have built things quickly cutting corners. Then, going forward, maintenance will be a nightmare and any changes will break the system in unexpected places."	Long term success of application important Developing too fast leads to nightmare during maintenance

Table 5. Example of Open Coding

The second essential open-coding task is discovering categories. Categorising is the process of finding related phenomena or common concepts and themes in the accumulated data in order to group them under joint headings, thus identifying categories and sub-categories of data. In our analysis we iden-

J. Pries-Heje, R. Baskerville, L. Levine & B. Ramesh • 53

tified more than fifty categories in both the 2000 and the 2002 studies. Three examples of categories from 2000 are: (1) A changed culture, (2) Different kind of market, and (3) Criticality of architecture.

Developing a deeper understanding of how the identified categories are related is the purpose of axial coding. Axial coding involves two tasks further developing the categories and properties. The first task connects categories in terms of a sequence of relationships. For example, a causal condition or a consequence can connect two categories, or a category and a sub-category. The second task turns back to the data for validation of the relationships. This return gives rise to the discovery and specification of the differences and similarities among and within the categories. Thus discovery adds variation and depth of understanding.

Selective Coding involves the integration of the categories that have been developed to form the initial theoretical framework. First, a story line is generated or made explicit. A story is simply a descriptive narrative about the central phenomenon of study; the story line is the conceptualisation of this story (abstracting). When analysed, the story line becomes the core category, which is related to all the categories found in axial coding, validating these relationships, and elaborating the categories that need further refinement and development. In this investigation, the story line that best suited the data was a story about the changes that had happened from 2000 to 2002.