Balancing Quality and Agility in Internet Speed Software Development

Richard Baskerville
Georgia State University

Linda Levine
Software Engineering Institute

Jan Pries-Heje
IT University of Denmark

Balasubramaniam Ramesh
Georgia State University

Sandra Slaughter
Carnegie Mellon University

BALANCING QUALITY AND AGILITY IN INTERNET SPEED SOFTWARE DEVELOPMENT

Richard Baskerville
Georgia State University
Atlanta, GA USA
baskerville@acm.org

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

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

Balasubramanian Ramesh
Georgia State University
Atlanta, GA USA
bramesh@gsu.edu

Sandra Slaughter
Graduate School of Industrial Administration
Carnegie Mellon University
Pittsburgh, PA USA
sandras@andrew.cmu.edu

Abstract

This research seeks to discover how quality and agility can be achieved in Internet speed software development. The study is being conducted in multiple phases. During phase 1, detailed case studies of Internet software development were conducted with 10 companies. In phase 2, a Discovery Colloquium was held to synthesize knowledge on innovative practices for quality and agility in Internet software development. During phase 3, the objective is to develop a contingency framework that characterizes choice and effectiveness of the practices based upon the insights from the prior two phases and through a second round of interviews with the original companies (still in business) and with selected additional companies from phase 2. This paper reports on findings on the use of agile methodologies that have emerged from phase 2, the Discovery Colloquium, as presented within the larger context of the study.

1 INTRODUCTION

The explosion of electronic commerce on the Internet and the rapid rate at which corporations are reinventing themselves into e-businesses have created a radically new environment for software development. To be competitive in the digital economy requires the ability to develop high quality software systems at "Internet speed." However, the widespread diffusion of quality software development practices has met with serious obstacles. Software has been credited with contributing to some of the most spellbinding advances of the 20th century, but it also "easily rates among the most poorly constructed, unreliable, and least maintainable technological artifacts invented by man." (Gross et al. 1999). The problems with software development are exacerbated in the Internet environment, with the emphasis on reduced cycle times. Existing software development approaches may not be a match to the needs of this world. Such approaches can be oriented toward a large scale, can make cumbersome project configuration assumptions, and are based on assumptions that represent a single-goal dimension.

Beyond the high visibility Internet software leaders such as Microsoft and Netscape, little is known about how Internet software product development is carried out. How do software developers actually build their fast cycle time software? What is the impact of these development practices on the quality of the software? Anecdotal evidence suggests that quality assurance practices are candidates for compression in cycle time reduction (Wetherbe and Frolick 2000). However, there is a need to understand how firms are developing fast cycle time software, what quality processes are used, and how software quality is retained in this environment. Software development organizations are seeking appropriate methods, tools, and practices that meet the challenges imposed by Internet speed software development. Current methods for software development and models for process improvement (e.g., ISO 9000-3, CMM, SPICE, Bootstrap) are effective in large-scale, long-term development efforts that employ stable and disciplined processes (Harter et al. 2000; Krishnan et al. 2000). However, Internet software development is characterized by rapid changes in requirements and unpredictable product complexity. Approaches that achieve a balance between flexibility and disciplined methodology are necessary in such environments (Cusumano and Yoffie 1999a, 1999b; Iansiti and MacCormack 1997; Thomke and Reinertsen 1998). However, most current methods and tools address the needs of large systems developed by multiple large teams and are not easily downward scalable to Internet speed software development (Fayad et al. 2000; Laitinen et al. 2000).

In contrast to *heavy* or *monumental* methodologies (Highsmith 1999), 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) and emphasize improvisation (Miner et al. 2001). 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) are among such methodologies. XP, which is the most popular in this environment, emphasizes planning, analysis and design throughout the software development life cycle (Beck et al. 1999) and relies on some simple ideas that are put to extreme practice. The agile methodologies are not without critics. Among the problems associated with light methods are the potential loss of control in project management and the potential for inefficient and chaotic software design and architecture that require "refactoring" to be rationalized (Fowler 2000). Further, as the methodologies themselves are not very stable, predictability and repeatability, hallmarks of a disciplined process, cannot be guaranteed.

While the current literature recognizes that many challenges and new ideas are emerging, the need for solutions and frameworks that an Internet software development organization can readily adopt remains unfulfilled. Our research study addresses this void. It is conducted using multiple phases, each of which is designed to accomplish a particular objective. Specifically, this study seeks to: (phase 1) understand how and why the development of Internet software is different from traditional software development; (phase 2) discover innovative practices used to achieve both quality and agility in Internet software development; and (phase 3) discern the contingencies motivating the choice and effectiveness of these practices (Shenhar 2001). Note that this paper describes phase 2 of the study.

2 METHODOLOGY

The study uses a mixed methods research design (Tashakkori and Teddlie 1998) involving the collection of multiple kinds of data. During phase 1, in Fall 2000, detailed case studies of Internet software development were conducted with 10 companies in two major metropolitan areas. The objective of phase 1 was to understand how and why Internet software development differs from traditional software development. Phase 1 identified practices for Internet speed development (Baskerville et al. 2001) and explored the role of quality in this fast cycle environment (Ramesh et al. 2002).

The objectives of phase 2 are to synthesize knowledge on best practices for quality and agility in Internet software development. To help achieve these objectives, in October 2001, we held a one-day Discovery Colloquium on Innovative Practices for Speed and Agility in Internet Software Development (Levine et al. 2002). The colloquium activities were designed to expand understanding of the interface between business issues and Internet software development through dialogue; explore promising practices for Internet software development; and engage participants in a vision-based approach to identify emerging models, strategies, and directions to address challenges in Internet software development.

The Colloquium represented the second phase of our larger study and was designed to benefit from the findings from phase 1. Interviewees from the phase 1 companies, as well as selected experts were invited. Participants included software practitioners from entrepreneurial small firms as well as large "brick and mortar" companies, Internet business strategists, and leading software development experts. Software development methodological perspectives also ranged from adherents of agile methodologies to adherents of more traditional software process disciplines, representing a broad spectrum of stakeholders.

The event was designed to capitalize on the rich mix of participants and allow for maximum exchange of ideas. Our unconventional research process leveraged forward-looking approaches including search approaches to system learning, difference questioning, and creative abrasion. Grounded in Kurt Lewin's action research model, search conferences seek to "bring the whole system in the room" to exchange views and learn from one another (http://futuresearch.net/). The search conference design utilizes difference questioning, in the belief that "when members of a group question differences [they] generate new information" (Goldstein 1994). This is done through creative abrasion—a term coined by Dorothy Leonard (1995) to describe a process that "creates a collision of ideas within a climate of social cohesion," resulting in original and innovative ideas.

The Colloquium opened with an inclusion exercise focusing on the use of metaphors to illustrate participants' images about the current Internet software development environment. This exercise was followed with a series of active-listening exercises conducted in context-based groups, wherein participants discussed their own experiences. The exercises used "fishbowl" discussions as applied in education and human development, in combination with a model drawn from Zuni tribal council "talking circles." A series of small participant groups (or subgroups) were centered in the larger group, discussing issues pertinent to quality software development. While each subgroup engaged in dialogue, participants in the rest of the room reflected and listened to the discussion without comment. The fishbowls created an environment where thoughtful listening is as important as talking and sharing, and framed core questions for further consideration. The segment concluded with a full group discussion and selection of core issues for further analysis.

In the next session, participants joined one of several breakout groups, dedicated to exploration of a core issue. The first part of this exploration involved *hypothesis testing*, wherein groups identified observations relating to their core issue, and then developed hypotheses about the associated factors and dynamics. The purpose of this work was to begin the process of deep, thoughtful examination of an important trend or issue with Internet software development at the present time. The approach helped ensure that the groups avoided polarization in their views, but continued to carry on fruitful dialogue, embracing differences as well as convergence.

Difference questioning flowed from the hypothesis testing and created a process for groups to develop a set of assumptions that underlay each of their hypotheses. Once that was complete, the groups scanned their exploratory work to that point to identify emerging *principles and practices* that held potential for further investigation. The purpose of this work was to further build the community dialogue from generative ideas, and to ensure that that dialogue fairly represented all strongly held viewpoints. A further purpose was to allow for both divergence and convergence of insights, with maximum cross-fertilization across differences. Each group was provided worksheets to guide them through the process.

The session concluded with breakout groups reporting to the full gathering the key elements of their analysis. Finally, the full group engaged in "proto-scenario development" through identification of emerging conditions and impacts. Several scenarios were selected for analysis of factors enabling or constraining their emergence. This exercise highlighted evolving dynamics in Internet software development.

To illustrate the nature of the outputs of the Colloquium, we present initial findings on one core issue investigated, i.e., *agile methodologies.*

3 INITIAL FINDINGS: ARE AGILE METHODOLOGIES DIFFERENT?

The participants observed that the effectiveness of agile methods appears to lie in people not in process. The discussion revealed the belief that basic software development principles do not change. However, we learned that these principles are implemented through specific practices that evolve and are matched to environmental contingencies. The questions driving discussion included the following: What agile methodologies are being used in software development? What distinguishes agile methodologies from traditional software development methodologies? When are agile methodologies more effective?

Several general observations about agile methodologies emerged:

(1) People claim that agile methods are the best way to develop information systems, and assert that agile methods are a "new and radically different methodology."

(2) Three key elements of agile methods include: collaborative work; incremental, evolutionary life cycles within a strategy; and strong customer communication.

(3) Larger questions on agile methodologies can be explored by examining the basic practices of XP and other agile practices. There was general agreement that XP embodied most of what is seen as good about agile methods.

(4) The effectiveness of agile methods appears to lie in people *not* process. Agile methods reward heroes.

As part of the research and discovery process, the group generated rival hypotheses and underlying assumptions for these hypotheses about agile methodologies (Figure 1).

H1	Agile methods are more effective (than traditional software development methods)
H2	Agile methods are not fundamentally different from traditional software development methods
H3	Partially implementing key agile practices will lead to project failure
H4	Agile methods require good people to be successful
H5	Internet speed development is fundamentally different from traditional development; thus, agile methods are needed
H6	Agile methods are effective when the time horizon is short term, and not as effective over the long term
H7	Agile methods aren't really new per se, but their implementation is extreme

Figure 1. Hypotheses About Agile Methodologies

A number of key insights emerged. First, the basic principles of software development exist, are known, and are immutable. Second, there is more than one way to implement a software development principle. While everyone agrees on software development principles, some practices are controversial. It is important to describe what the principles are and to discern how practices differ from one methodology to another. Third, for every agile method practice, an analog in traditional software development methods can be found. For example, consider the following principles: good teamwork is critical; requirements change; and talking to the customer is good. These hold for agile and traditional methods. Fourth, some combinations of practices are superior in specific environments. This means that different environments require different implementations for effectiveness. Environments dictate practices, not principles. Fifth, software development is linear and time bounded at an atomic level. Development approaches are bounded by lifecycle choices. Thus, the choice of software development lifecycle dictates the choice of practices such as waterfall, spiral, and prototyping.

Based upon these insights, the group members generated meta-principles underlying software development and methodologies. The meta-principles are presented in Figure 2.

P1	All software processes require good teamwork regardless of methodology
P2	Good software development methodologies engage the customer
P3	The project environment constrains what software development practices will be effective
P4	Practices interact often in unforeseen ways. The implication is that you can mix and match methodologies, but sometimes this has unintended consequences
P5	Current agile methods are a simple instantiation of a set of practices that seem to work

Figure 2. Meta-Principles Underlying Software Development Methodologies

Finally, the group members identified promising practices in agile methodologies, including: short development iterations to explore unknown problem spaces effectively without wasting time, accountability and discipline in programming, minimal essential documentation, onsite customer liaison, pair programming, frequent peer and customer reviews, frequent regression testing, configuration management, and detailed planning for the immediate future (up to six months) with "fuzzier" planning beyond six months.

A major finding is that agile methods may appear to be different in implementation, but are not really different in terms of software development principles. Additionally, while software development principles don't change, different implementations (or practices) may be more effective in different environments. No "one size fits all" approach works for all projects.

4 DISCUSSION

We are currently analyzing our findings from the Colloquium. Further, we are synthesizing knowledge on best practices for quality and agility that have emerged from our phase 1 case studies, the Colloquium, and our literature review. Based upon this synthesis, we are developing an initial framework that characterizes the situational factors motivating the choice of these practices and their effective use. At the conference, we will present key findings from phase 1 and phase 2. Phase 3 of the study is planned to validate and refine the framework based upon a second round of interviews with the original companies (still in business) and with selected additional companies from Phase 2.

5 REFERENCES

Aoyama, M. "Web-Based Agile Software Development," *IEEE Software*, November/December 1998, pp. 56-65.

Baskerville, R., Levine, L., Pries-Heje, J., Ramesh, B., and Slaughter, S. "How Internet Software Companies Negotiate Quality," *IEEE Computer* (34:5), 2001, pp. 51-57.

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

Beck, K., and Fowler, M. *Planning Extreme Programming*. Reading, MA: Addison Wesley, 2001.

Beck, K., Hannula, J., Hendrickson, C., Wells, D., and Mee, R. "Embracing Change with Extreme Programming," *IEEE Computer* (32:10), 1999, pp. 70-77.

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

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.

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

Cusumano, M. A., and Yoffie, D. B. "What Netscape Learned from Cross-Platform Software Development," *Communications of the ACM* (42:10), 1999b, 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*. Reading, MA: Addison-Wesley, 2000.

Goldstein, J. *The Unshackled Organization*. Portland, OR: Productivity Press, 1994.

Gross, N., Stepanek, M., Port, O., and Carey, J. "Software Hell," *BusinessWeek Online*, December 6, 1999.

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.

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

Iansiti, M., and MacCormack, A. "Developing Products on Internet Time," *Harvard Business Review*, September-October 1997, pp. 108-117.

Jeffries, R., Anderson, A., and Hendrickson, C. *Extreme Programming Installed*. Reading, MA: Addison-Wesley, 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.

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

Leonard, D. *Wellsprings of Knowledge: Building and Sustaining the Sources of Innovation*. Cambridge, MA: Harvard Business School Press, 1995.

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. Pittsburgh, PA: Software Engineering Institute, 2002 (available online at http://www.sei.cmu.edu/publications/documents/02.reports/02tr020.html).

Miner, A. S., Bassoff, P., and Moorman, C. "Organizational Improvisation and Learning: A Field Study," *Administrative Science Quarterly* (46:2), Jun 2001, pp. 304-337.

Ramesh, B., Baskerville, R., and Pries-Heje, J. "Internet Software Engineering: A Different Class of Processes," *Annals of Software Engineering* (14), 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.

Shenhar, A. J. "One Size Does Not Fit All Projects: Exploring Classical Contingency Domains," *Management Science* (47:3), 2001, pp. 394-414.

Tashakkori, A., and Teddlie, C. *Mixed Methodology: Combining Qualitative and Quantitative Approaches*. Thousand Oaks, CA: Sage Publications, 1998.

Thomke, S., and Reinertsen, D. "Agile Product Development: Managing Development Flexibility in Uncertain Environments," *California Management Review* (41:1), 1998, pp. 8-30.

Wetherbe, J., and Frolick, M. "Cycle Time Reduction: Concepts and Case Studies," *Communications of the AIS* (3:Article 13), 2000, pp. 1-42.