

A Model for Measuring Agile Methods and Website Quality

By: David F Rico, Hasan H Sayani, James J Stewart and Ralph F Field

Abstract

The purpose of this paper is to present a model for measuring the relationship between the use of agile methods to manage the development of Internet websites and their quality. Agile methods are a general purpose approach for managing the development of new products, which are often associated with Internet software. Agile methods may be used for improving website quality by obtaining early customer feedback on a large number of frequent software releases. Agile methods are characterized by early customer involvement, iterative development, self organizing teams, and flexibility. This model may help managers better understand the business effects of adopting or failing to adopt agile methods for the \$2 trillion US electronic commerce industry. Preliminary survey results from over 250 software developers shows that this new model for measuring compliance with agile methods is fairly accurate. A scholarly model for measuring the quality of e-commerce websites is also discussed.

1 Introduction

Does use of agile methods improve quality of Internet websites used for \$2 trillion electronic commerce industry by US firms?

Agile methods are a contemporary management approach used for developing Internet or electronic commerce websites by US firms. There is a well-documented debate among management scholars about the best approaches for managing the development of Internet websites. Some scholars believe traditional methods rooted in well-established scientific management principles lead to high quality Internet websites. However, contemporary management scholars believe agile methods have the characteristic of a job-shop or craft industry predating the scientific management era and are the best approach for managing the development of high-quality Internet websites.

Agile methods are an approach used for managing the development of Internet software, which aspire to improve website quality from early customer feedback on a large number of frequent releases.

Our challenge is to identify, survey, select, or develop a scholarly instrument for measuring the use of agile methods and the quality of Internet websites, and then collect data to determine whether their use is linked to higher quality websites for electronic commerce by US firms. Though agile methods are a general purpose approach for managing the development of new products, they are often associated with Internet software, which is the focus of this paper.

There seems to be no middle ground on this issue. Some management scholars firmly believe agile methods lead to lower quality websites and others believe agile methods lead to higher quality websites, with neither side offering much empirical evidence to support their claims. We hope to create the first in a long line of scholarly papers which test the relationships between the use of agile methods as a management approach for developing Internet websites and improved website quality for electronic commerce among US firms.

Internet websites are the fundamental tool for conducting business transactions on the Internet, otherwise known as electronic commerce. The US alone generates around \$1.95 trillion in revenues using Internet websites each year. Likewise, US firms commit between \$152 and \$231 billion in information technology expenditures each year to achieve these revenues. Currently at stake is a ten-fold return on investment in electronic commerce revenues to information technology expenditures by US firms each year (for example, revenues ÷ expenditures).

US managers have already committed between 50% and 67% of their information technology projects to the use of agile management methods^[1]. They may want to know whether the use of agile methods for managing

Editor's note:

>50% using Agile methods in the US, what are the proportions elsewhere in the world?

Let me know ...

the development of Internet websites leads to higher quality websites and what the potential consequences to lower electronic commerce revenues are if the detractors of agile methods are indeed correct – that is, whether traditional methods rooted in scientific management principles are superior to those of agile methods associated with the much earlier job-shop or craft industry tradition of the early industrial revolution in the US. The lessons associated with both sides of these debates will be addressed throughout this paper.

1.1 Rationale and Justification

Today, US firms generate more than \$2 trillion each year from electronic commerce and Internet websites are a big part of this massive revenue stream^[2]. As a result, the top 500 US firms spend more than \$231 billion per year on Internet related technologies in order to exploit this potential revenue^[3]. Furthermore, much of the annual \$400 billion US defense budget is devoted to information technology as well^[4], which may increase interest in the results of this paper. There are more than 250,000 software projects in the US of which more than 72% have failed or are failing^[5]. Therefore, executives and managers of US firms may benefit from the knowledge that agile methods may be linked to better website quality. There has never been a greater need for scholarly studies of agile methods.

1.2 Goals and Objectives

The research goals and objectives are to gather information that might determine if a link exists between the use of agile methods for managing the development of Internet software and website quality for electronic commerce. Therefore, the research goals and objectives are to examine the empirical links between the theoretical factors of agile methods and scholarly constructs of website quality in the field of electronic commerce. Conversely, the research goals and objectives of this paper are also to determine if early customer involvement, iterative development, self organizing teams, and flexibility are not linked to scholarly models of website quality for electronic commerce. Negative correlations between the factors of agile methods and website quality for electronic commerce are just as important as positive ones.

1.3 Research Questions

The basic research area to be explored is whether the use of agile methods by US firms is more effective than traditional approaches based on scientific management principles. A closely related question is how firms manage the development of Internet software for the \$2 trillion US electronic commerce industry? Another closely related question is what management approaches are linked to website quality for US firms? And, of course, what factors are motivating the use of new management approaches such as agile methods? Specific questions include whether the major factors or principles of agile methods are linked to website quality among US firms. Is the use of early customer involvement linked to website quality among US firms? Is the use of iterative development linked to website quality among US firms? Is the use of self organizing teams linked to website quality among US firms? Is the use of flexibility linked to website quality among US firms? Equally important is whether the answer to these questions is 'no'.

2 Literature Review

2.1 History of Computers and Software

Modern electronic computers are characterized by four major generations: first generation vacuum tube computers from 1940 to 1950, second generation transistorized computers from 1950 to 1964, third generation integrated circuit computers from 1964 to 1980, and fourth generation microprocessor computers from 1980 to the present^[6, 7, 8]. By 1972, there were 170 programming languages in the US alone^[9] and today there are over 8,500 programming languages worldwide^[10]. First and second generation computers did not have any operating systems, third generation computers consisted of the first multiprogramming operating systems such as UNIX, and fourth generation computers were best characterized by personal computer operating systems such as MS-DOS and Microsoft Windows^[8]. The international software industry grew slowly in revenues for commercially shrink-wrapped software from about zero in 1964, to \$2 billion per year in 1979, and \$50 billion by 1990^[11, 12]. By 1990, there were over 20,000 commercial shrink-wrapped software packages on the market^[13]. And, the

international software industry grew to more than \$90 billion for pre-packaged software and \$330 billion for all software-related products and services by 2002^[3]. First and second generation computers were not known to have been networked together, third generation computers gave rise to packet switching theory, and fourth generation computers gave rise to the Internet as we know it today, when the number of computers on the Internet reached 110 million in 2001^[14, 15].

2.2 History of Electronic Commerce

Electronic commerce is as old as the computer and software industries themselves and predates the Internet era of the 1990s^[16]. There is no standard taxonomy of electronic commerce technologies, but they do include major categories such as magnetic ink character recognition, automatic teller machines, electronic funds transfer, stock market automation, facsimiles, email, point of sale systems, Internet service providers, and electronic data interchange, as well as electronic retail trade and shopping websites^[16]. Second generation computers were associated with electronic commerce technologies such as magnetic ink character recognition or MICR created in 1956^[17]. Third generation computers were associated with electronic commerce technologies such as automatic teller machines, electronic funds transfer, stock market automation, facsimiles, email, point of sale systems, electronic bulletin boards, and electronic data interchange. Early fourth generation computers were associated with electronic commerce technologies such as the vast automation of the stock market. Mid fourth generation computers were associated with electronic commerce technologies such as selected electronic services, electronic retail trade, and electronic shopping and mail order houses, which today garner more than \$1.95 trillion in revenues^[2].

2.3 History of Software Methods

Methodologies may be regarded as formal processes for managing the development of information systems, which standardize the steps of design with the intention of improving productivity and quality^[18]. The mainframe era of the 1960s included software methods such as database design, automatic programming, software project management, early user involvement, structured methods, and formal methods. The midrange era of the 1970s included software methods such as software life cycles, software reviews, object oriented methods, software testing, software environments, software quality assurance, and software processes. The microcomputer era of the 1980s included software methods such as rapid development, software reuse, and software architecture. Finally, the Internet era of the 1990s included software methods such as agile methods.

2.4 History of Agile Methods

Agile methods gained prominence in the late Internet and early personalized eras to accommodate the uniquely flexible nature of Internet technologies. Agile methods are an approach for managing the development of software, which are based upon obtaining early customer feedback on a large number of frequent software releases^[19]. The dynamic systems development methodology or DSDM has three broad phases, which consist of requirements prototypes, design prototypes, and then an implementation or production phase^[20]. Scrum is a light weight software development process consisting of implementing a small number of customer requirements in two to four week sprint cycles^[21]. Extreme programming or XP consists of collecting informal requirements from on-site customers, organizing teams of pair programmers, developing simple designs, conducting rigorous unit testing, and delivering small and simple software packages in short two-week intervals^[22]. Open source software development involves freely sharing, peer reviewing, and rapidly evolving software source code for the purpose of increasing its quality and reliability^[23]. Crystal methods involve frequent delivery; reflective improvement; close communication; personal safety; focus; easy access to expert users; and a technical environment with automated testing, configuration management, and frequent integration^[24]. Feature driven development involves developing an overall model, building a features list, planning by feature, designing by feature, and building by feature^[25].

3 Examples of Agile Methods

3.1 Scrum

In 1993, Jeff Sutherland of Easel adapted the principles from the ‘new product development game’^[26] to the field of computer programming, calling it ‘scrum’^[21]. In particular, scrum assumes that the ‘systems development process is an unpredictable and complicated process that can only be roughly described as an overall progression’. Furthermore, scrum’s creators believed ‘systems development is not a well understood approach that can be planned, estimated successfully’. Therefore, scrum’s creators set out to define the process as “a loose set of activities that combines known, workable tools and techniques with the best that a development team can devise to build systems”. Today, scrum is composed of three broad phases:

- pre-sprint planning,
- sprint, and
- post-sprint meeting.

During the pre-sprint planning phase, programmers gather to prioritize customer needs. During the sprint phase, programmers pretty much do whatever it takes to complete a working version of software that meets a small set of high priority customer needs. Finally, during the post-sprint meeting, programmers demonstrate working software to their customers, adjust their priorities, and repeat the cycle. Today, scrum is gaining in widespread popularity for its overall simplicity.

3.2 Extreme Programming

In 1998, 20 software managers working for the Chrysler Corporation published an article on how they devised a management approach called ‘extreme programming’ or XP to turn around a failing software project that would provide payroll services for 86,000 Chrysler employees^[22]. In their article, they stated that “extreme programming rests on the values of simplicity, communication, testing, and aggressiveness.” They also stated that the “project had been declared a failure and all code thrown away, but using the extreme programming methodology, Chrysler started over from scratch and delivered a very successful result.” Extreme programming consists of 13 principles:

- planning game,
- small releases,
- metaphor,
- simple design,
- tests,
- refactoring,
- pair programming,
- continuous integration,
- collective ownership,
- onsite customer,
- 40 hour work week,
- open workspace, and
- just rules.

What these 20 software managers did was start over, get an informal statement of customer needs, gradually evolve a simple system design using iterative development, apply rigorous testing, use small teams of programmers, and get early customer feedback on their evolving design. In the end, Chrysler was able to deploy an operational payroll system serving more than 86,000 employees. Today, extreme programming is the most widely used management approach for developing software.

3.3 Feature Driven Development

In 1997, three software managers and five software developers created a software development approach called ‘feature driven development’ to help save a failed project for an international bank in Singapore^[25]. In their textbook, they stated that “the bank had already made one attempt at the project and failed, the project had inherited a sceptical user community, wary upper management, and a demoralized development team.” Furthermore, they stated that “the project was very ambitious, with a highly complex problem domain spanning three lines of business, from front office automation to back-end legacy system integration.” In order to address this highly complex problem domain that had already experienced severe setbacks, they created an agile and adaptive software development process that is “highly iterative, emphasizes quality at each step, delivers frequent tangible working results, provides accurate and meaningful progress, and is liked by clients, managers, and developers.” Feature driven development consists of five overall phases or processes:

- develop an overall model,
- build a features list,
- plan by feature,
- design by feature, and
- build by feature.

Feature driven development also consists of other best practices in software management and development such as domain object modelling, developing by feature, individual class ownership, feature teams, inspections, regular builds, configuration management, and reporting and visibility of results.

	FDD	Extreme Programming	DSDM	Scrum
Practices	<ul style="list-style-type: none"> • Domain object modeling • Developing by feature • Class (code) ownership • Feature teams • Inspections • Regular build schedule • Configuration management • Reporting/visibility of results 	<ul style="list-style-type: none"> • Planning game • Small releases • Metaphor & Simple Design • Tests • Refactoring • Pair programming • Continuous integration • Collective ownership • On-site customer • 40-hour weeks • Open workspace & Just rules 	<ul style="list-style-type: none"> • Active user involvement • Empowered teams • Frequent delivery • Fitness (simplicity) • Iterations and increments • Reversible changes • Baselined requirements • Integrated testing • Stakeholder collaboration 	<ul style="list-style-type: none"> • Product backlog • Burndown chart • Sprint backlog • Iterations and increments • Self managed teams • Daily scrums
Processes	<p>Develop an Overall Model Form the Modeling Team Conduct a Domain Walkthrough Study Documents Develop Small Group Models Develop a Team Model Refine the Overall Object Model Write Model Notes Internal and External Assessment</p> <p>Build a Features List Form the Features List Team Build the Features List Internal and External Assessment</p> <p>Plan by Feature Form the Planning Team Determine Development Sequence Assign Features to Chief Coders Assign Classes to Developers Self Assessment</p> <p>Iteration (1) Design by Feature Form a Feature Team Conduct Domain Walkthrough Study Referenced Documents Develop Sequence Diagrams Refine the Object Model Write Class/Method Prologue Design Inspection Build by Feature Implement Classes/Methods Conduct Code Inspection Unit Test Promote to the Build</p> <p>Iteration (2) Design by Feature Form a Feature Team Conduct Domain Walkthrough Study Referenced Documents Develop Sequence Diagrams Refine the Object Model Write Class/Method Prologue Design Inspection Build by Feature Implement Classes/Methods Conduct Code Inspection Unit Test Promote to the Build</p> <p>Iteration (n) Design by Feature Form a Feature Team Conduct Domain Walkthrough Study Referenced Documents Develop Sequence Diagrams Refine the Object Model Write Class/Method Prologue Design Inspection Build by Feature Implement Classes/Methods Conduct Code Inspection Unit Test/Prime the Build</p>	<p>User Stories Requirements Acceptance Tests</p> <p>Architectural Spike System Metaphor</p> <p>Release (1) Release Planning Release Plan Iteration (1) Iteration Planning Iteration Plan Daily Standup Collective Code Ownership Create Unit Tests Unit Tests Pair Programming Move People Around Refactor Mercilessly Continuous Integration Acceptance Testing</p> <p>Iteration (2) Iteration Planning Iteration Plan Daily Standup Collective Code Ownership Create Unit Tests Unit Tests Pair Programming Move People Around Refactor Mercilessly Continuous Integration Acceptance Testing</p> <p>Iteration (n) Iteration Planning Iteration Plan Daily Standup Collective Code Ownership Create Unit Tests Unit Tests Pair Programming Move People Around Refactor Mercilessly Continuous Integration Acceptance Testing</p> <p>Release (2) Iteration (1) Iteration (2) Iteration (n)</p> <p>Release (n) Iteration (1) Iteration (2) Iteration (n)</p>	<p>Feasibility Study Feasibility Report Feasibility Prototype (optional) Outline Plan Risk Log</p> <p>Business Study Business Area Definition Prioritized Requirements List Development Plan System Architecture Definition Updated Risk Log</p> <p>Functional Model Iteration Functional Model Functional Prototype (1) Functional Prototype Functional Prototype Records Functional Prototype (2) Functional Prototype Functional Prototype Records Functional Prototype (n) Functional Prototype Functional Prototype Records Non-functional Requirements List Functional Model Review Records Implementation Plan Timebox Plans Updated Risk Log</p> <p>Design and Build Iteration Timebox Plans Design Prototype (1) Design Prototype Design Prototype Records Design Prototype (2) Design Prototype Design Prototype Records Design Prototype (n) Design Prototype Design Prototype Records Tested System Test Records</p> <p>Implementation User Documentation Trained User Population Delivered System Increment Review Document</p>	<p>Iteration (1) Sprint Planning Meeting Product Backlog Sprint Backlog Sprint Daily Scrum Shippable Code Sprint Review Meeting Shippable Code Sprint Retrospective Meeting</p> <p>Iteration (2) Sprint Planning Meeting Product Backlog Sprint Backlog Sprint Daily Scrum Shippable Code Sprint Review Meeting Shippable Code Sprint Retrospective Meeting</p> <p>Iteration (n) Sprint Planning Meeting Product Backlog Sprint Backlog Sprint Daily Scrum Shippable Code Sprint Review Meeting Shippable Code Sprint Retrospective Meeting</p>

Table 1: Summary of Practices and Processes of Agile Methods

3.4 Dynamic Systems Development

In 1993, 16 academic and industry organizations in the United Kingdom banded together to create a management approach for commercial software called the ‘Dynamic Systems Development Method’ or simply DSDM^[20]. Their goal was to “develop and continuously evolve a public domain method for rapid application development” in an era dominated by proprietary methods. Initially, DSDM emphasized three success factors:

- the end user community must have a committed senior staff that allows developers easy access to end users,
- the development team must be stable and have well established skills, and
- the application area must be commercial with flexible initial requirements and a clearly defined user group. These success factors would later be expanded to include functionality versus quality, product versus process, rigorous configuration management, a focus on business objectives, rigorous software testing, risk management, and flexible software requirements. DSDM consists of five major stages:
 - feasibility study,
 - business study,
 - functional model iteration,
 - design and build iteration, and
 - implementation.

The goal of DSDM is to explore customer requirements by building two full scale prototypes before the final system is implemented.

4 Studies of Agile Methods

4.1 Early Case Studies

In 1989, three managers from IBM in Rochester, Minnesota, published an article on how IBM devised a management approach called the ‘new development rhythm’, to bring the AS/400 midrange computer to market in only two years^[27]. The new development rhythm consisted of six major principles:

- modularized software designs,
- rigorous software reviews and software testing,
- overlapped software releases, and
- software reuse,
- iterative development,
- early user involvement and feedback.

IBM’s new development rhythm was a remarkable feat of management science, and boasted a long list of accomplishments: (a) time-to-market improvement of 40%, (b) development of seven million lines of operating system software in 26 months, (c) compatibility with 30 billion lines of commercial applications, (d) \$14 billion in revenues, and (e) the IBM corporation’s first Malcolm Baldrige National Quality Award. In 1995, two management scholars from MIT’s Sloan School of Management published a textbook on how Microsoft managed the development of software for personal computers, dubbed the “sync-n-stabilize” approach [28].

The sync-n-stabilize approach consisted of six major principles:

- parallel programming and testing,
- daily operational builds,
- early customer feedback, and
- flexible software requirements,
- iterative development,
- use of small programming teams.

Microsoft’s success was indeed remarkable, and their sync-n-stabilize approach did indeed help them create more than 20 million lines of code for Windows and Office 95, achieve customer satisfaction levels of 95%, and maintain annual profit margins of nearly 36%. In 1998, two management scholars from both the Harvard Business School and MIT’s Sloan School of Management published a textbook on how Netscape managed the development of software for the Internet, dubbed the ‘judo strategy’^[29]. Some of the more notable principles characteristic of Netscape’s judo strategy included:

- design products with modularized architectures;
- rapidly adapt to changing market priorities;
- apply as much rigorous testing as possible; and
- use beta testing and open source strategies to solicit early market feedback on features, capabilities, quality, and architecture.
- use parallel development;

The judo strategy helped Netscape achieve \$1.4bn in revenues, a 168% growth rate, and capture 90% of the market share for Internet browsers.

4.2. Early scholarly studies

In 1998, a management scholar from the Harvard Business School conducted a study on how US firms manage the development of websites, referring to his approach as ‘Internet time’^[30]. He set out to test the theory that website quality was associated with three major principles:

- greater investments in architectural design,
- greater amounts of generational experience.
- early market feedback, and

Correlational analysis supported two of the hypotheses, e.g., greater architectural resources and early market feedback were associated with higher website quality, but not the third, e.g., greater experience among computer programmers is associated with higher website quality. In 1999, two management scholars from Boston College’s Carroll School of Management conducted a case study of 28 software projects to determine the effects of Results driven incrementalism on project success^[31]. Their approach consisted of the use of targeted business results, a series of non-overlapping increments, increments implementing everything required to produce desired results, 90 day iteration periods, and feeding back the results of each increment into subsequent iterations. What they found was that software projects that use iterative development deliver working software 38% sooner, complete their projects twice as fast, and satisfy over twice as many software requirements. In 2003, two management scholars conducted a study of 42 software projects, 19 implementing agile methods and 23 of them implementing more traditional waterfall methods^[32]. They measured actual effort in person hours, estimated effort in person hours, balanced relative error bias, actual schedule performance in calendar days, estimated schedule performance in calendar days, and the amount of functionality implemented. What they found was that agile projects have fewer overruns due to better requirements from early user involvement.

4.3 Early surveys

In 2003, Reifer Consultants conducted a survey of 78 projects from 18 firms to determine the effects of using agile methods to manage the development of software^[33]. What they found was that 14% to 25% of respondents experienced productivity gains, 7% to 12% reported cost reductions, and 25% to 80% reported time-to-market improvements. In 2003, Shine Technologies conducted an international survey of 131 respondents to determine the effects of using agile methods to manage the development of software^[34]. What they found was that 49% of the respondents experienced cost reductions, 93% of the respondents experienced productivity increases, 88% of the respondents experienced quality increases, and 83% experienced customer satisfaction improvements. In 2004, CIO Magazine conducted a survey of 100 information technology executives with an average annual budget of \$270 million to determine the effects of agile management on organizational effectiveness^[35]. What they found was that 28% of respondents had been using agile management methods since 2001, 85% of the respondents were undergoing enterprise wide agile management initiatives, 43% of the respondents were using agile management to improve organizational growth and market share, and 85% said agile management was a core part of their organizational strategy.

4.3 Newest Surveys

In 2006, Digital Focus conducted a survey of 136 respondents to determine the effects of using agile methods to manage the development of software^[36]. What they found was that 27% of the respondents were adopting agile methods for a project, 23% of the respondents were adopting agile methods company wide, 51% of the respondents wanted to use agile methods to speed up the development process, 51% of the respondents said they lacked the skills necessary to implement agile methods at the project level, 62% of the respondents said they lacked the skills necessary to implement agile methods at the organization level, and 60% planned on teaching themselves how to use agile methods.

In 2006, Version One conducted an international survey of 722 respondents to determine the effects of using agile methods to manage the development of software^[37]. What they found was that 86% of the respondents reported time-to-market improvements, 87% of the respondents reported productivity improvements, 86% of the respondents reported quality improvements, 63% of the respondents reported cost reductions, 92% of the respondents reported the ability to manage changing priorities, 74% of the respondents reported improved morale, 72% of the respondents reported risk reductions, 66% of the respondents reported satisfaction of business goals, and 40% were using the scrum method. In 2006, Ambyssoft conducted an

international survey of 4,232 respondents to determine the effects of using agile methods to manage the development of software^[38]. What they found was that 41% of organizations were using agile methods; 65% used more than one type of agile method; 44% reported improvements in productivity, quality, and cost reductions; and 38% reported improvements in customer satisfaction.

Year	Source	Findings	Responses
1998	Harvard [30]	48% productivity increase over traditional methods 38% higher quality associated with more design effort 50% higher quality associated with iterative development	29
1999	Boston College [31]	38% reduction in time to produce working software 50% time to market improvement 50% more capabilities delivered to customers	28
2003	Simula Research [32]	5% shorter schedule estimates 9% shorter schedules 33% improvement in overall estimate accuracy	42
2003	Reifer Consultants [33]	20% reported productivity gains 10% reported cost reductions 53% reported time-to-market improvements	78
2003	Shine Technologies [34]	49% experienced cost reductions 93% experienced productivity increases 88% experienced customer satisfaction improvements	131
2004	CIO Magazine [35]	28% had been using agile methods since 2001 85% initiated enterprise-wide agile methods initiatives 43% used agile methods to improve growth and marketshare	100
2006	Digital Focus [36]	27% of software projects used agile methods 23% had enterprise-wide agile methods initiatives 51% used agile methods to speed-up development	136
2006	Version One [37]	86% reported time-to-market improvements 87% reported productivity improvements 92% reported ability to dynamically change priorities	722
2006	AmbySoft [38]	41% of organizations used agile methods 44% reported improved productivity, quality, and costs 38% reported improvements in customer satisfaction levels	4,232

Table 2: Summary of Recent Studies and Surveys of Agile Methods

5 Agile Manifesto

In 2001, the central figures responsible for creating agile methods convened a council to analyze the similarities between their approaches, the goals of agile methods, and how agile methods differed from the traditional methods. The manifesto for agile software development states, “we are uncovering better ways of developing software by doing it and helping others do it”^[39]. They devised four broad values:

- individuals and interactions over processes and tools,
- working software over comprehensive documentation,
- customer collaboration over contract negotiation, and
- responding to change over following a plan.

Furthermore, they stated, “while there is value in the items on the right, we value the items on the left more.” In other words, they value individuals and interactions, working software, customer collaboration, and responding to change much more than processes and tools, comprehensive documentation, contract negotiation, and following a plan. They also devised 12 broad principles, which can be summarized in four:

- early customer involvement,
- self organizing teams, and
- iterative development,
- flexibility.

Early customer involvement provides feedback on working software releases; iterative development means programmers produce frequent small releases of working software; self organizing teams means programmers are free to manage themselves as they see fit; and flexibility means use of informal processes, freely accepting

changing requirements, and structuring the software itself to accommodate change. The values opposed software methods characterized by processes, tools, documentation, contract negotiation, and following plans.

6 Conceptual Model of Agile Methods

Following an analysis of literature on agile methods, a conceptual model of agile methods was created. The conceptual model consisted of five broad factors:

- early customer involvement,
- self organizing teams,
- flexibility, and
- iterative development,
- website quality.

While a factor analysis of numerous textbooks and articles was conducted over a 2.5 year period,^[39] and ^[40] served as the primary sources of the four principal factors of agile methods. An analysis of 76 research studies from 44 scholarly journals covering 138 authors, 200 factors, 564 variables, 2,550 instrument items, and 30,797 respondents was then conducted to identify and select operational variables and measures for each of the five factors comprising the model. 20 variables and 20 instrument items were chosen that were: (a) conceptually aligned with the factors of agile methods and website quality, (b) used within the same study to ensure inter-item reliability, and (c) proven to exhibit high levels of validity with the fewest number of instrument items. It is important to note that this conceptual model and the items below are highly original and are unique to this study.

6.1 Early Customer Involvement

Do we seek customer feedback on our software iterations, increments, or demonstrations?

Do we receive customer feedback on our software iterations, increments, or demonstrations?

Do we receive timely customer feedback on our software iterations, increments, or demonstrations?

Do we receive a lot of (detailed) customer feedback on our software iterations, increments, or demonstrations?

Do we incorporate customer feedback into our software iterations, increments, or demonstrations?

6.2 Iterative Development

Do we develop software using time-based iterations, increments, or demonstrations?

Do we develop software using operational iterations, increments, or demonstrations (working code vs. documentation)?

Do we develop software using small iterations, increments, or demonstrations?

Do we develop software using daily, weekly, bi-weekly, or monthly iterations, increments, or demonstrations?

Do we develop software using multiple (several) iterations, increments, or demonstrations?

Editor's note:

Try asking these questions of your organizations using Agile methods and see how you shape up.

Let me know what results you achieve ...

6.3 Self Organizing Teams

Do our software teams have clear administrative or technical leaders?

Do our software teams have clear visions, missions, or strategies?

Do our software teams have clear goals or objectives?

Do our software teams have clear schedules or timelines?

Do our software teams have a small size with no more than 10 people?

6.4 Flexibility

Is our software designed to be as small as possible?

Is our software designed to be as simple as possible?

Is our software designed to be modular or object-oriented?

Is our software designed to work on multiple operating systems?

Is our software designed to be changed, modified, or maintained?

6.5 Website Quality

Website design: to what extent do websites:

- provide in-depth information?
- avoid wasting time?
- make it quick and easy to complete transactions?
- allow for personalization?
- have a good selection of products, services, and information?^[41]

Fulfillment/reliability: to what extent do websites:

- accurately represent products to be delivered?
- provide products ordered from them?
- satisfy the product delivery times they promise?^[41]

Security/privacy: to what extent do websites:

- provide the feeling that privacy is protected?
- provide the feeling that transactions are safe?
- have adequate security features?^[41]

Customer service: to what extent are websites:

- designed to respond to customer needs?
- designed to show a sincere interest in solving problems?
- designed to answer inquiries promptly?^[41]

6.6 Preliminary Survey Results

The preliminary survey results look quite promising. Results from 100 respondents show that the conceptual model of agile methods is a good predictor of agility. In most cases, the responses range from somewhat agree to strongly agree. Only in a few exceptions, do the respondents favour the range strongly disagree to somewhat disagree. They are feedback frequency, feedback quality, small size, and portable design. However, these preliminary survey results may not be necessarily indicating a weakness with the measurement model. The respondents may simply be saying that their customers rarely provide them with timely feedback, and certainly not with high quality feedback. This is not surprising. In the case of small size, this item may need to be corrected. Software size isn't important as the simplicity of the iteration, increment, demonstration, or release. In other words, as long as it is simple, the release may be large. Portable design may simply be a misunderstanding. Anyone designing web applications is developing a portable design, because browsers enable applications to run on any type of computer. The next step is to correlate these results to our model of website quality^[41].

7 Summary

Similar, but not identical, papers have empirically examined the use of agile methods for managing the development of Internet software and website quality. These papers have implied or even shown limited and indirect linkages between the use of agile methods and primitive constructs of Internet software quality, reliability, and robustness. Some have even gone as far as to link the use of agile methods to basic notions of website quality and project success. In doing so, these papers have demonstrated some empirical evidence of the reliability and validity of these hypothesized inferences and knowledge claims. These papers may reduce the risks associated with attempting to link the use of agile methods to website quality for electronic commerce. The uniqueness of this paper is a holistic focus on all of the major tenets of agile methods. This paper proposes to link the use of agile methods to scholarly models of website quality for electronic commerce. Thus, this may be one of the first holistic examinations of agile methods for the \$2 trillion US electronic commerce industry. (We wish to thank Jon Erickson of Dr. Dobbs Journal and Scott Ambler for their assistance in collecting this data.)

Editor's note:

Try asking these questions of your organization's web applications and see how you shape up.

Let me know what results you achieve ...

Measuring Agile Methods

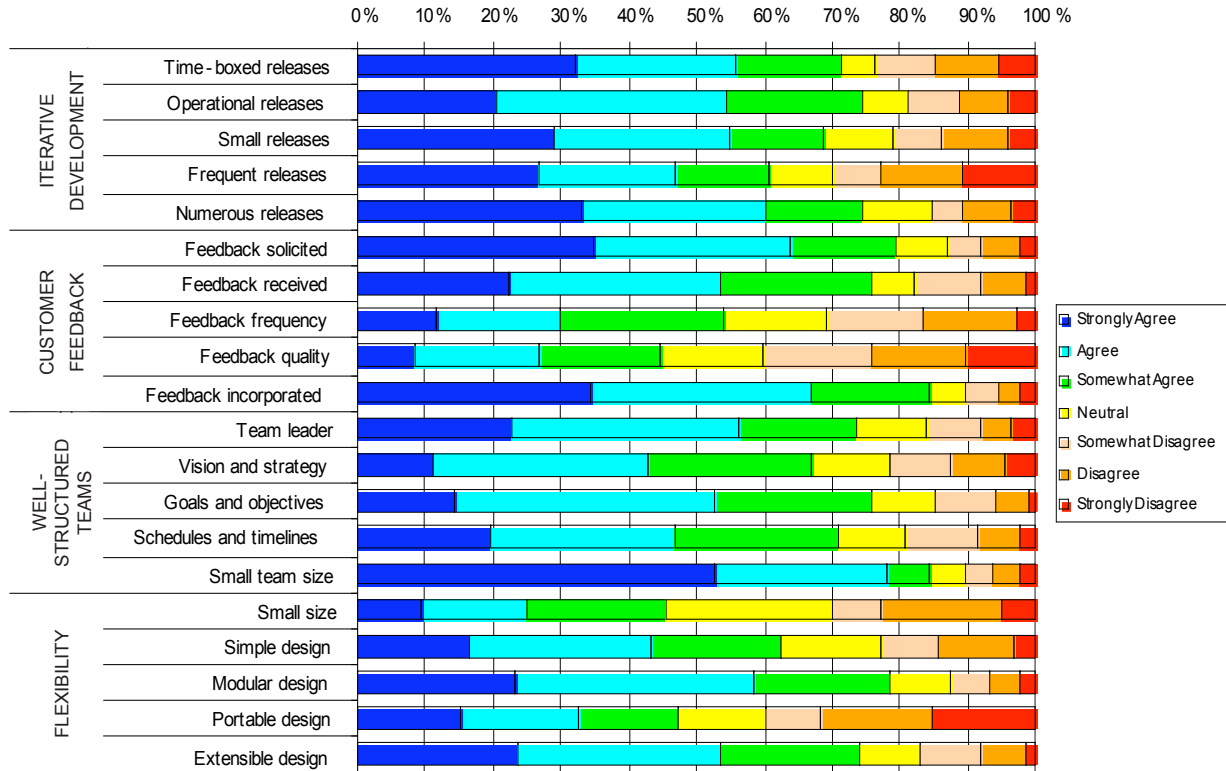


Figure 1. Results of Preliminary Survey of Agile Methods

Factor	Item	1	2	3	5	6	7
Fulfillment and reliability	You get what you ordered from this site						
	The product is delivered by the time promised by the company						
	The product that came was represented accurately by the website						
Website design	The website provides in -depth information						
	The site doesn't waste my time						
	It is quick and easy to complete a transaction at this website						
	The level of personalization at site is about right , not too much or too little						
	This website has good selection						
Privacy and security	I feel like my privacy is protected at this site						
	I feel safe in my transactions with this website						
	The website has adequate security features						
Customer service	The company is willing and ready to respond to customer needs						
	When you have a problem , the website shows a sincere interest in solving it						
	Inquiries are answered promptly						

Figure 2. Model of Website Quality for Future Analysis of Survey Results

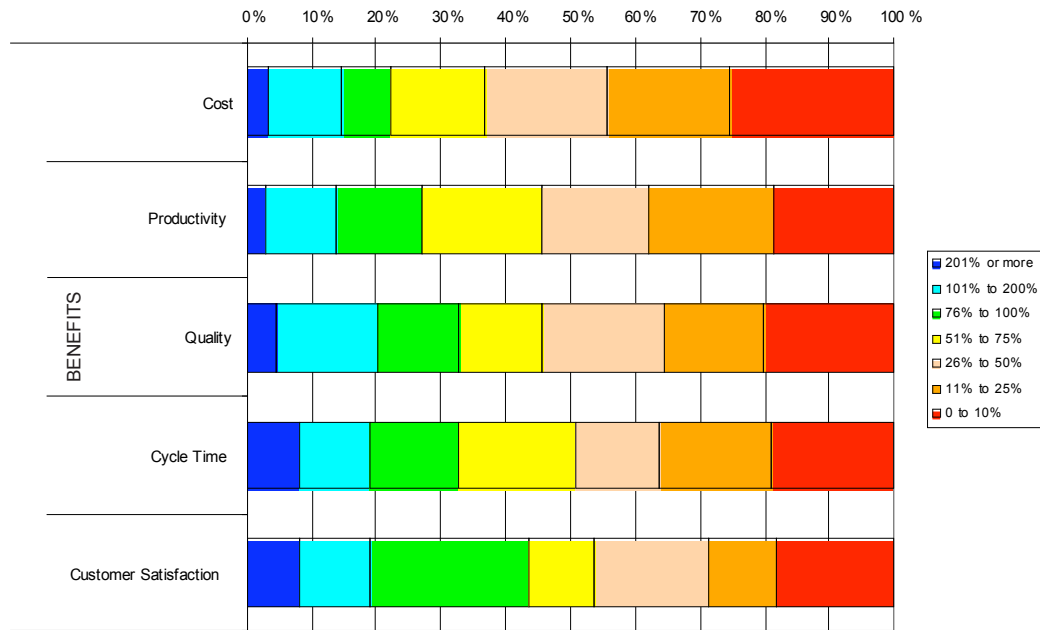


Figure 3. Preliminary Benefits from Survey of Agile Methods

8 References

- [1] Sliwa, C. (2002). Users warm up to agile programming. *Computer World*, 36(12), 8-8.
- [2] US Census Bureau. (2006). *E-stats*. Washington, DC: Author.
- [3] US Department of Commerce. (2006). *Information and communication technology: 2004*. Washington, DC: Author.
- [4] Fulghum, D. A., & Wall, R. (2004). Is this war? *Aviation Week and Space Technology*, 160(6), 22-24.
- [5] Standish Group. (2000). *Extreme chaos*. West Yarmouth, MA: Author.
- [6] Denning, J. (1971). Third generation computer systems. *ACM Computing Surveys*, 3(4), 175-216.
- [7] Rosen, S. (1969). Electronic computers: A historical survey. *ACM Computing Surveys*, 1(1), 7-36.
- [8] Tanenbaum, A. (2001). *Modern operating systems*. Englewood Cliffs, NJ: Prentice Hall.
- [9] Sammet, J. E. (1972b). Programming languages: History and future. *Communications of the ACM*, 15(7), 601-610.
- [10] Pigott, D. (2006). HOPL: An interactive roster of programming languages. Retrieved October 21, 2006, from <http://hopl.murdoch.edu.au>
- [11] Campbell-Kelly, M. (1995). Development and structure of the international software industry: 1950-1990. *Business and Economic History*, 24(2), 73-110.
- [12] Steinmueller, W. E. (1996). The US software industry: An analysis and interpretive history. In D. C. Mowery (Ed.), *The international computer software industry* (pp. 25-52). New York, NY: Oxford University Press.
- [13] Middleton, R., & Wardley, P. (1990). Information technology in economic and social history: The computer as philosopher's stone or Pandora's box? *Economic History Review*, 43(4), 667-696.
- [14] Leiner, B., Cerf, V. G., Clark, D. D., Kahn, R. E., Kleinrock, L., Lynch, D. C., et al. (1997). The past and future history of the internet. *Communications of the ACM*, 40(2), 102-108.
- [15] Mowery, D. C., & Simcoe, T. (2002). Is the internet a US invention? An economic and technological history of computer networking. *Research Policy*, 31(8/9), 1369-1387.
- [16] Kalakota, R., & Whinston, A. (1996). *Electronic commerce: A manager's guide*. Reading, MA: Addison Wesley.
- [17] Mandell, L. (1977). Diffusion of EFTS among national banks. *Journal of Money, Credit, and Banking*, 9(2), 341-348.
- [18] Hardgrave, B. C., Davis, F. D., & Riemenschneider, C. K. (2003). Investigating determinants of software developer's intentions to follow methodologies. *Journal of Management Information Systems*, 20(1), 123-151.
- [19] Beck, K. (1999). Embracing change with extreme programming. *IEEE Computer*, 32(10), 70-77.
- [20] Millington, D., & Stapleton, J. (1995). Developing a RAD standard. *IEEE Software*, 12(5), 54-56.
- [21] Schwaber, K. (1995). Scrum development process. *Proceedings of the 10th Annual ACM Conference on Object Oriented Programming Systems, Languages, and Applications (OOPSLA 1995)*, Austin, Texas, USA, 117-134.

- [22] Anderson, A., Beattie, R., Beck, K., Bryant, D., DeArment, M., Fowler, M., et al. (1998). Chrysler goes to extremes. *Distributed Computing Magazine*, 1(10), 24-28.
- [23] O'Reilly, T. (1999). *Lessons from open source software development*. Communications of the ACM, 42(4), 32-37.
- [24] Cockburn, A. (2002). *Agile software development*. Boston, MA: Addison Wesley.
- [25] Palmer, S. R., & Felsing, J. M. (2002). *A practical guide to feature driven development*. Upper Saddle River, NJ: Prentice Hall.
- [26] Takeuchi, H., & Nonaka, I. (1986). The new product development game. *Harvard Business Review*, 64(1), 137-146.
- [27] Sulack, R. A., Lindner, R. J., & Dietz, D. N. (1989). A new development rhythm for AS/400 software. *IBM Systems Journal*, 28(3), 386-406.
- [28] Cusumano, M. A., & Selby, R. W. (1995). *Microsoft secrets: How the world's most powerful software company creates technology, shapes markets, and manages people*. New York, NY: The Free Press.
- [29] Cusumano, M. A., & Yoffie, D. B. (1998). *Competing on internet time: Lessons from netscape and its battle with microsoft*. New York, NY: The Free Press.
- [30] MacCormack, A. (1998). *Managing adaptation: An empirical study of product development in rapidly changing environments*. Unpublished doctoral dissertation, Harvard University, Boston, MA, United States.
- [31] Fichman, R. G., & Moses, S. A. (1999). An incremental process for software implementation. *Sloan Management Review*, 40(2), 39-52.
- [32] Molokken-Ostfold, K., & Jorgensen, M. (2005). A comparison of software project overruns: Flexible versus sequential development models. *IEEE Transactions on Software Engineering*, 31(5), 754-766.
- [33] Reifer, D. J. (2003). The business case for agile methods/extreme programming (XP). *Proceedings of the Seventh Annual PSM Users Group Conference, Keystone, Colorado, USA*, 1-30.
- [34] Johnson, M. (2003). *Agile methodologies: Survey results*. Victoria, Australia: Shine Technologies.
- [35] Prewitt, E. (2004). The agile 100. *CIO Magazine*, 17(21), 4-7.
- [36] Digital Focus. (2006). *Agile 2006 survey: Results and analysis*. Herndon, VA: Author.
- [37] Version One. (2006). *The state of agile development*. Apharetta, GA: Author.
- [38] Ambler, S. W. (2006). *Agile adoption rate survey: March 2006*. Retrieved September 17, 2006, from <http://www.ambysoft.com/downloads/surveys/AgileAdoptionRates.ppt>
- [39] Agile Manifesto. (2001). *Manifesto for agile software development*. Retrieved November 29, 2006, from <http://www.agilemanifesto.org>
- [40] Highsmith, J. A. (2002). *Agile software development ecosystems*. Boston, MA: Addison Wesley.
- [41] Wolfenbarger, M., & Gilly, M. C. (2003). Etailq: Dimensionalizing, measuring, and predictingetail quality. *Journal of Retailing*, 79(3), 183-198.

David F. Rico is a SPI consultant specializing in cost and benefit analysis. He helped design a \$250M software engineering toolset and the spacecraft software for NASA's \$20B space station in the 1980s, performed graduate studies under SEI Level 5 space shuttle managers, helped a \$40B Japanese corporation design a CMM self assessment tool in 1993, designed a software cost model for 37 kinds of US Navy fighter aircraft, helped reengineer 36 logistics depots for America's largest foreign military customer, played key roles in the design of US military intelligence satellites, and has supported 15 software engineering process groups (SEPGs) over the last decade. He's been an international keynote speaker, published numerous articles, and holds a B.S. in Computer Science and Master's Degree in Software Engineering (with 19 years of experience).

dave@davidfrico.com <http://davidfrico.com>



TickIT+ Achieves Initial Support

It is good to report that the evolutionary development of an improved and enhanced TickIT scheme has achieved initial financial support and will therefore be going ahead under the stewardship of the Joint TickIT Industry Steering Committee.

It is now important for us to gather more supporters and financial backers to ensure that this important progression is implemented in a timely manner so that we can all benefit ASAP.

So if you are aware of any sources of support, either expertise or finance, please make the TickIT International editorial team aware and we will pass the contacts on to the development team.