

What is the Return on Investment (ROI) of Agile Methods?

by David F. Rico

Abstract

The purpose of this paper is to investigate the return-on-investment (ROI) of agile methods. Agile methods are new product development processes for creating software-based goods and services. Agile methods are a lightweight alternative to traditional methods based on sequential product development processes created over the last four or five decades. The use of traditional methods is theorized to result in higher quality software products because of well-documented customer requirements and products that exhibit fewer problems over their life cycle. Agile methods on the other hand are used to achieve higher customer satisfaction and product quality through rapid implementation and early market testing. The ROI of agile methods is yet to be fully explored because of their newness, while the ROI of traditional methods is well-understood. Therefore, the purpose of this paper is to investigate and summarize the literature on the ROI of agile methods. These results show that the use of agile methods results in increased cost-effectiveness, productivity, quality, cycle-time reduction, and customer satisfaction ranging from 10% to 100%.

1. Introduction

An agile method is a contemporary new product development process for creating computer software such as operating systems, middleware, applications, and web-based technologies. A new product development process is a streamlined management and development methodology for quickly and efficiently creating innovative goods and services. New product development processes span the life cycle of a novel technology from its inception, concept, or idea-stage right on through its operations, market, or end-user stage. Agile methods are generally characterized by lightweight, informal, and highly-adaptable new product development processes. Agile methods are rooted in concept testing, rapid prototyping, and early market feedback, and are lightweight forms of overlapping, cross-functional, simultaneous, integrated-product, time-based, and concurrent development. Agile methods are also rooted in chaos theory, systems theory, systems thinking, systems dynamics, double-loop learning, learning organizations, organizational learning, and adaptable systems. Agile methods evolved directly from their earlier traditional cousins.

2. Agile Methods

Agile methods are different from traditional software development methods. Traditional methods are also product development processes based on the theory of sequentially building software goods and services. Traditional methods are a sequential product development process of systems planning, analysis, architecture, design, development, and testing. Traditional methods are based on rigidly defined policies, processes, procedures, documentation, and tools with rigid interfaces between them. Traditional methods are based on the theory that a customer's or end-user's requirements can be defined at the beginning of the process, sequentially transformed into a software product, and then delivered to the customer when it is complete. A traditional method may take years or even decades to cycle through its process. Traditional methods generally fail to deliver a product that satisfies its customer's requirements. Agile methods, on the other hand, may be better than traditional methods because they may be used to achieve customer satisfaction and software quality by soliciting customer feedback on a series of rapid product releases. Agile methods are based on four broad processes of using iterative development, customer feedback, small software development teams, and flexible software technologies. Here are some of the major forms of agile methods [14].

2.1 New Product Development Game

In 1986, two management scholars from the School of International Corporate Strategy at Hitotsubashi University in Tokyo, Japan, published a approach called the "new product development game" in the Harvard Business Review [28]. In their article, they argued that Japanese "companies are increasingly realizing that the old sequential approach to developing new products simply will not get the job done." They cited the sport of Rugby as the inspiration for the principles of their new product development game—In particular, Rugby's special play called the Scrum, when the players interlock themselves together as a tightly bound group to gain possession of the ball. The new product development game consisted of six major factors: (a) built-in instability, (b) self organizing project teams, (c) overlapping development phases, (d) multi-learning, (e) subtle control, and (f) organizational transfer of learning.

2.2 New Development Rhythm

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]. In their article, they stated that “user involvement programs yielded a product offering that met the user requirements with a significantly reduced development cycle.” The new development rhythm consisted of six major factors: (a) modularized software designs, (b) software reuse, (c) rigorous software reviews and software testing, (d) iterative development, (e) overlapped software releases, and (f) early user involvement and feedback.

2.3 Crystal Methods

In 1991, a software manager with IBM was asked to create an approach for managing the development of object oriented systems called “crystal methods” [5]. Crystal methods were piloted on a “\$15 million firm, fixed-price project consisting of 45 people.” Crystal methods are a “family of methods with a common genetic code, one that emphasizes frequent delivery, close communication, and reflective improvement.” The seven properties of crystal methods are: (a) frequent delivery; (b) reflective improvement; (c) close communication; (d) personal safety; (e) focus; (f) easy access to expert users; and (g) a technical environment with testing, configuration management, and frequent integration.

2.4 Scrum

In 1993, Jeff Sutherland of the Easel Corporation adapted the principles from the “new product development game” [28] to the field of computer programming management, explicitly calling it “scrum” [26]. 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 “the stated philosophy that systems development is a well understood approach that can be planned, estimated, and successfully completed has proven incorrect in practice.” Therefore, scrum’s creators set out to define a 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: (a) pre-sprint planning, (b) sprint, and (c) post-sprint meeting.

2.5 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 [16]. 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: (a) “the end user community must have a committed senior staff that allows developers easy access to end users,” (b) “the development team must be stable and have well established skills,” and (c) “the application area must be commercial with flexible initial requirements and a clearly defined user group.” These were expanded to functionality versus quality, product versus process, configuration management, business objectives focus, testing, risk management, and flexible requirements. DSDM consists of five major stages: (a) feasibility study, (b) business study, (c) functional model iteration, (d) design and build iteration, and (e) implementation.

2.6 Synch-n-Stabilize

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 “synch-n-stabilize” approach [6]. Experts on software management approaches for the mainframe market, their two year case study from 1993 to 1995 was more of a grounded theory or emergent research design. At one point in their textbook, they stated that “during this initial research, it became clear why Microsoft was able to remain on top in its industry while most of its contemporaries from the 1970s had disappeared.” Synch-n-stabilize consisted of six major factors: (a) parallel programming and testing, (b) flexible software requirements, (c) daily operational builds, (d) iterative development, (e) early customer feedback, and (f) use of small programming teams. This influential study was based on principles from [27].

2.7 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 [17]. In their textbook, they stated that “the bank had already made one attempt at the project

and failed, and the project had inherited a skeptical user community, wary upper management, and a demoralized development team.” Feature driven development consists of five phases: (a) develop an overall model, (b) build a features list, (c) plan by feature, (d) design by feature, and (e) build by feature. Feature driven development also consists of other best practices in software management and development such as domain object modeling, developing by feature, individual class ownership, feature teams, inspections, regular builds, configuration management, and reporting and visibility of results.

2.8 Open Source Software Development

The term “open source software development” or OSS was coined in 1997, though the practice of open source software started in 1970 [4]. Simply put, open source software is a “set of computer instructions that may be used, copied, modified, and distributed by anyone, anywhere, and for any purpose whatsoever” [12]. Another definition stated “open source software is labeled with free source, fast evolution, and extensive user collaboration” [31]. One study identified eight factors of open source software: (a) is parallel rather than linear; (b) involves large communities of globally distributed developers; (c) utilizes truly independent peer review; (d) provides prompt feedback to user and developer contributions; (e) includes the participation of highly talented developers; (f) includes increased user involvement; (g) makes use of extremely rapid release schedules; and (h) produces evolutionary designs [9]. One author wryly mused, “Internet time refers to something much faster, revolutionary, and more basic—It describes the process of developing open source software” [18].

2.9 Judo Strategy

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” [7]. The more notable characteristics of Netscape’s judo strategy included: (a) design products with modularized architectures; (b) use parallel development; (c) rapidly adapt to changing market priorities; (d) apply as much rigorous testing as possible; and (e) use beta testing and open source strategies to solicit early market feedback on features, capabilities, quality, and architecture.

2.10 Internet Time

In 1998, a management scholar from the Harvard Business School conducted a study on how U.S. firms manage the development of websites, referring to his approach as “Internet time” [15]. His study states that “constructs that support a more flexible development process are associated with better performing projects.” He surveyed 29 projects from 15 Internet firms such as Microsoft, Netscape, Yahoo, Intuit, and Altavista. He set out to test the theory that website quality was associated with three major factors: (a) greater investments in architectural design, (b) early market feedback, and (c) greater amounts of generational experience.

2.11 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 [3]. Extreme programming consisted of 13 factors: (a) planning game, (b) small releases, (c) metaphor, (d) simple design, (e) tests, (f) refactoring, (g) pair programming, (h) continuous integration, (i) collective ownership, (j) onsite customer, (k) 40 hour workweek, (l) open workspace, and (m) just rules.

3. ROI of Agile Methods

The purpose of ROI studies is to illustrate the business-value of using agile methods. There is the notion of soft-side ROI and hard-side ROI. Soft-side ROI refers to qualitative benefits such as improved morale or attitudes towards agile methods. While this is a legitimate form of ROI, this paper examines hard-side ROI. Hard-side ROI refers to the quantitative benefits of agile methods, often expressed in economic terms. For instance, if the use of agile methods takes half the time of traditional methods, then there is a direct economic benefit to increased productivity. That is, using agile methods may cost half as much as traditional methods. Below are only 11 major studies on the ROI of agile methods. There are many studies of agile methods ROI not mentioned here. However, these studies only looked at one or two techniques of agile methods such as pair programming. It is the intent of this paper to investigate the ROI of using agile methods in their entirety. In other words, what is the ROI of iterative development, early customer feedback, small teams, and flexible software technology?

3.1 Harvard Business School

In 1998, two management scholars from the Harvard Business School conducted a survey of 391 respondents to test the effects of flexible versus inflexible product technologies [29]. What they found was that projects using inflexible product technologies required over two times as much engineering effort as flexible product technologies (e.g., 17.94 vs. 8.15 months).

3.2 Harvard Business School

In 1998, a management scholar from the Harvard Business School conducted a survey of 29 projects from 15 U.S. Internet firms to test the effects of flexible software development management approaches on website quality [15]. What he found was that flexible product architectures and customer feedback on early beta releases were correlated to higher levels of website quality.

3.3 Boston College Carroll School of Management

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 iterative development on project success [10]. 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.

3.4 Reifer Consultants

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 [20]. 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.

3.5 Shine Technologies

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 [13]. 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.

3.6 CIO Magazine

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 [19]. 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.

3.7 Digital Focus

In 2006, Digital Focus conducted a survey of 136 respondents to determine the effects of using agile methods to manage the development of software [8]. 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.

3.8 Version One

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 [30]. 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.

3.9 AmbySoft 2006

In 2006, Ambysoft conducted an international survey of 4,232 respondents to determine the effects of using agile methods to manage the development of software [1]. 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.

3.10 AmbySoft 2007

In 2007, Ambysoft conducted another international survey of 781 respondents to further determine the effects of using agile methods to manage the development of software [2]. What they found was that 69% of organizations had adopted agile methods, 89% of agile projects had a success rate of 50% or greater, and 99% of organizations are now using iterative development.

3.11 UMUC

In 2007, a student at the University of Maryland University College (UMUC) conducted a survey of 250 respondents to determine the effects of using agile methods on website quality [21], [22], [24], [25]. What he found was that: (a) 70% of all developers are using many if not all aspects of agile methods; (b) 79% of all developers using agile methods have more than 10 years of experience; (c) 83% of all developers using agile methods are from small to medium-sized firms; (d) 26% of all developers using agile methods have had improvements of 50% or greater; (e) developers using all aspects of agile methods produced better e-commerce websites.

Year	Source	Findings	Responses
1998	Harvard (Thomke et al., 1998)	50% reduction in engineering effort 55% improvement in time to market 925% improvement in number of changes allowed	391
1998	Harvard (MacCormack, 1998)	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 (Fichman et al., 1999)	38% reduction in time to produce working software 50% time to market improvement 50% more capabilities delivered to customers	28
2003	Reifer Consultants (Reifer, 2003)	20% reported productivity gains 10% reported cost reductions 53% reported time-to-market improvements	78
2003	Shine Technologies (Johnson, 2003)	49% experienced cost reductions 93% experienced productivity increases 88% experienced customer satisfaction improvements	131
2004	CIO Magazine (Prewitt, 2004)	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 (Digital Focus, 2006)	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 (Version One, 2006)	86% reported time-to-market improvements 87% reported productivity improvements 92% reported ability to dynamically change priorities	722
2006	AmbySoft (Ambler, 2006)	41% of organizations used agile methods 44% reported improved productivity, quality, and costs 38% reported improvements in customer satisfaction levels	4,232
2007	AmbySoft (Ambler, 2007)	69% of organizations had adopted agile methods 89% of agile projects had a success rate of 50% or greater 99% of organizations are now using iterative development	781
2007	UMUC (Rico, 2007)	70% of developers using most aspects of agile methods 26% of developers had improvements of 50% or greater Agile methods are linked to improved website quality	250

4. Conclusion

The purpose of this article was to examine and identify the ROI of agile methods. More specifically, its purpose was to identify the ROI of using agile methods in their entirety, not just some of the tools and techniques of agile methods like pair programming. This paper looks at the ROI of using all of the major factors of agile methods including iterative development, early customer feedback, small software development teams, and flexible software technologies that enhance productivity. The studies examined here identify many hard-side ROI benefits for using agile methods. The benefits of using agile methods range from 10% to 100% for increased cost-effectiveness, productivity, quality, cycle-time reduction, and customer satisfaction. The use of agile methods as a new product development approach does result in increased ROI. This begins to dispel the notion that agile methods result in lower ROI than traditional methods. However, it is important to note that these are only early studies and further study of the ROI of agile methods is necessary to make better conclusions. Promising new studies are starting to emerge based on more sophisticated approaches to measuring the ROI and more extensive historical data [11], [23].

5. References

- [1] Ambler, S. W. (2006). Agile adoption rate survey: March 2006. Retrieved September 17, 2006, from <http://www.ambysoft.com/downloads/surveys/AgileAdoptionRates.ppt>
- [2] Ambler, S. W. (2007). Agile adoption survey: March 2007. Retrieved July 23, 2007, from <http://www.ambysoft.com/downloads/surveys/AgileAdoption2007.ppt>
- [3] 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.
- [4] Bretthauer, D. (2002). Open source software: A history. *Information Technology and Libraries*, 21(1), 3-10.
- [5] Cockburn, A. (2002). *Agile software development*. Boston, MA: Addison Wesley.
- [6] 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.

- [7] 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.
- [8] Digital Focus. (2006). *Agile 2006 survey: Results and analysis*. Herndon, VA: Author.
- [9] Feller, J., & Fitzgerald, B. (2002). *Understanding open source software development*. London, England: Pearson Education.
- [10] Fichman, R. G., & Moses, S. A. (1999). An incremental process for software implementation. *Sloan Management Review*, 40(2), 39-52.
- [11] Fichman, R. G., Keil, M., & Tiwana, A. (2005). Beyond valuation: Options thinking in IT project management. *California Management Review*, 40(2), 74-96.
- [12] Fink, M. (2003). *The business and economics of linux and open source*. Upper Saddle River, NJ: Prentice Hall.
- [13] Johnson, M. (2003). *Agile methodologies: Survey results*. Victoria, Australia: Shine Technologies.
- [14] Larman, C. (2004). *Agile and iterative development: A manager's guide*. Boston, MA: Pearson Education.
- [15] MacCormack, A. (1998). *Managing adaptation: An empirical study of product development in rapidly changing environments*. Unpublished doctoral dissertation, Harvard University, Boston, MA, United States.
- [16] Millington, D., & Stapleton, J. (1995). Developing a RAD standard. *IEEE Software*, 12(5), 54-56.
- [17] Palmer, S. R., & Felsing, J. M. (2002). *A practical guide to feature driven development*. Upper Saddle River, NJ: Prentice Hall.
- [18] Pavlicek, R. C. (2000). *Embracing insanity: Open source software development*. Indianapolis, IN: Sams Publishing.

- [19] Prewitt, E. (2004). The agile 100. CIO Magazine, 17(21), 4-7.
- [20] 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.
- [21] Rico, D. F. (2007). Effects of agile methods on electronic commerce: Do they improve website quality? Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS 2007), Waikaloa, Big Island, Hawaii.
- [22] Rico, D. F. (2007). Effects of agile methods on website quality for electronic commerce. Unpublished doctoral dissertation, University of Maryland University College, Adelphi, MD, United States (<http://davidfrico.com/rico07.pdf>).
- [23] Rico, D. F. (2007). ROI of technology readiness assessments using real options: An analysis of gao data from 62 u.s. dod programs. Retrieved November 28, 2007, from <http://davidfrico.com/rico07o.pdf>
- [24] Rico, D. F. (2008). Effects of agile methods website quality for electronic commerce. Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008), Waikaloa, Big Island, Hawaii.
- [25] Rico, D. F., Sayani, H. H., Stewart, J. J., & Field, R. F. (2007). A model for measuring agile methods and website quality. TickIT International, 9(3), 3-15.
- [26] 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.
- [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] Takeuchi, H., & Nonaka, I. (1986). The new product development game. Harvard Business Review, 64(1), 137-146.
- [29] Thomke, S., & Reinertsen, D. (1998). Agile product development: Managing development flexibility in uncertain environments. California Management Review, 41(1), 8-30.
- [30] Version One. (2006). The state of agile development. Apharetta, GA: Author.
- [31] Zhao, L., & Deek, F. P. (2004). User collaboration in open source software development. Electronic Markets, 14(2), 89-103.

WHAT IS THE ROI OF AGILE VS. TRADITIONAL METHODS? An analysis of XP, TDD, Pair Programming, and Scrum (Using Real Options)

Dr. David F. Rico, PMP, CSM

Abstract

Little is known about the costs and benefits of Agile Methods since their popularization in 1999, though 67% of projects use them and 75 books and 100s of papers have been written about them. The purpose of this article is to analyze the costs and benefits reported in studies of new product development approaches such as Agile Methods as compared to those of Traditional Methods. Over 300 articles on Agile Methods were examined; cost, schedule, productivity, quality, and customer satisfaction data were found in 69 studies; and ROI data were identified in 29 studies. Agile Methods ROI was four times more than expensive Traditional Methods, two times less than inexpensive ones, and the best Agile and Traditional Methods had equal ROI (see Figure 1). However, it may not be proper to compare Traditional Methods optimized for productivity and quality to Agile Methods optimized for customer satisfaction, project success, and risk reduction.

Introduction

The U.S. and worldwide information technology industry continues to grow at an amazing rate. In 2006, software industry revenues reached \$393 billion, and business-to-consumer (B2C) and business-to-business (B2B) electronic commerce revenue reached \$220 billion and \$2.7 trillion. Likewise, the number of Internet websites now exceeds 136 million, the number of U.S. Internet shoppers is in excess of 147 million, and the number of Internet users is greater than 1.3 billion.¹ Accordingly, information technology is the second leading contributor to the U.S. economy and contributes to more than 50% of labor productivity growth in the top 10 industrialized nations. Also in 2006, U.S. firms spent over \$251 billion in information technology investments and the U.S. Department of Defense used \$447 billion to acquire information technology based systems. This flurry of activity led to more than 6 million U.S. information technology jobs, 450,000 projects, 265,000 certified project managers, and 36,000 Scrum masters to help manage them. Finally, 900,000 firms used ISO 9001 for quality management, 300,000 projects used Agile Methods for software design, and 840 firms used CMMI[®] for process improvement in 2006.

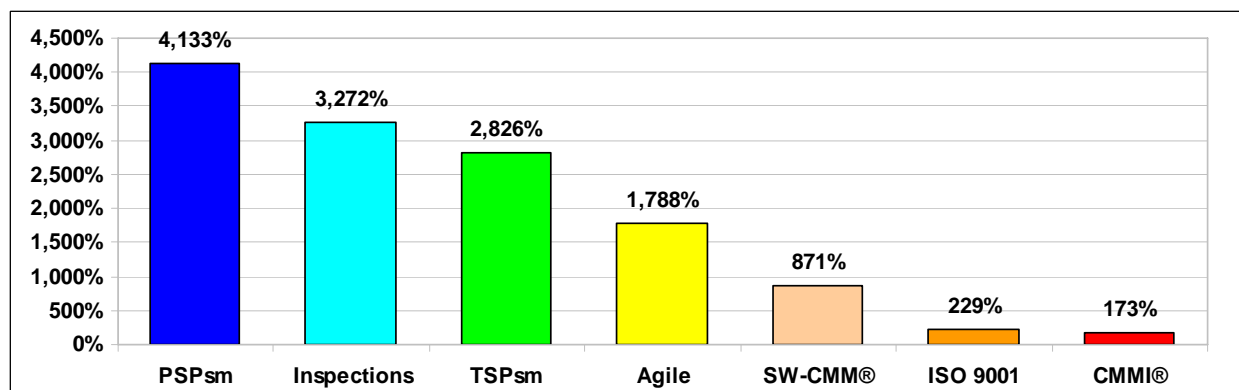


Figure 1. Methods for Managing Information Technology Projects (with decreasing ROI from left to right)

[®] Personal Software Process (PSP), Team Software Process (TSP), Software-Capability Maturity Model (SW-CMM), and Capability Maturity Model Integration (CMMI) are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University (CMU).

Agile Methods

Agile methods are lightweight software design processes based on small teams using flexible technologies to iteratively improve software using customer feedback to converge on solutions. Kent Beck is credited with creating Agile Methods by devising Extreme Programming in 1998, though XP was just one in a long line of hundreds of software methods dating back to 1968.² According to the Agile Manifesto, the major factors of Agile Methods are: (1) early customer involvement, (2) iterative development, (3) self-organizing teams, and (4) adaptation to change. Early customer involvement was known as top-level commitment, management involvement, user involvement, user participation, lead users, and participatory design from 1950 to 1980. Iterative development was known as concept testing, beta testing, and probing in marketing and iterative, incremental, evolutionary, spiral, and time-boxed development in the software field. Self organizing teams were known as self organizing dynamic teams, self determined groups, small decision-making groups, task oriented groups, and autonomous groups up to the 1960s. Adaptability came from organismic biology, cybernetics, systems theory, systems dynamics, double loop learning, adaptive organizations, learning organizations, and systems thinking.

Thomas Edison's success is attributed to the use of agile, new product development processes, along with Lockheed's SR-71, NASA's Apollo program, and the Jet Propulsion Laboratory.³ But, direct antecedents of Agile Methods include Joint Application Design, Rapid Application Development, Participatory Design, Synch-and-Stabilize, Judo Strategy, and Internet Time.⁴ Agile methods include Extreme Programming, Scrum, Feature Driven Development, Dynamic Systems Development, Lean Development, Crystal Methods, and Adaptive Software Design.⁵ By 2003, 66% of the world's projects were using Agile Methods and 90% of those were using Extreme Programming (XP),⁶ although the number of projects using XP has declined to 23%.⁷ The number of software projects using Scrum is increasing and it has caught the fancy of big firms like Google, Yahoo, and Microsoft, and as many as 50,000 projects may be using Scrum. The latest trend is to mix-and-match Scrum and XP to tap into practices like Pair Programming (PP) and Test-Driven Development (TDD) to increase productivity and quality (see Figure 2).⁸ Agile Methods have capabilities beyond Traditional Methods—That is, the ability to successfully deliver results quickly and inexpensively on complex projects with ill-defined requirements.

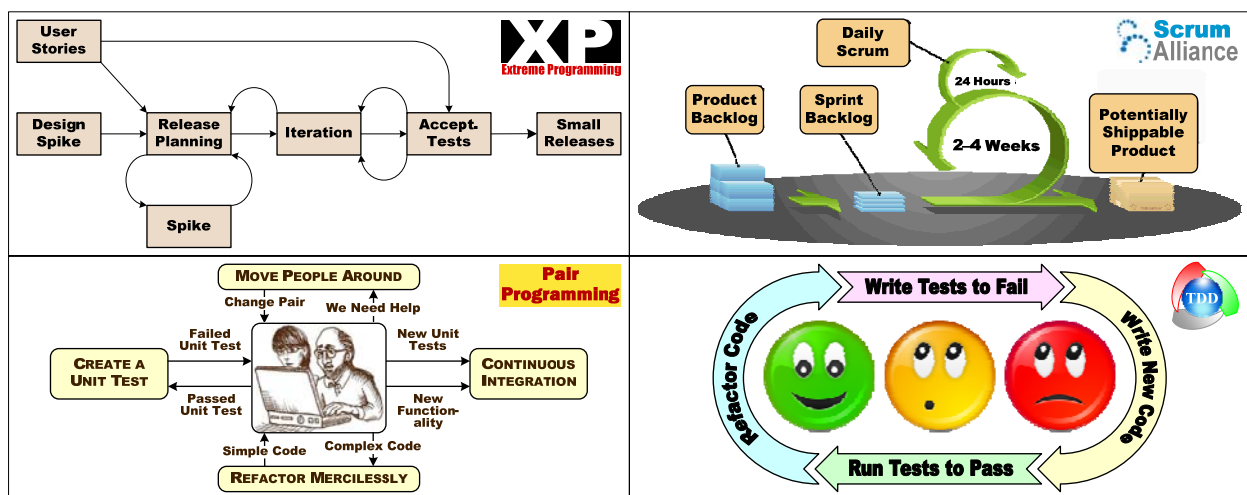


Figure 2. Agile Methods and Practices (with often-reported costs and benefits)

Agile Methods Costs and Benefits

A primary goal of this study was to examine scholarly studies of Agile Methods and survey the range of quantitative costs and benefits associated with the use of Agile Methods (see Table 1). Data were compared to costs and benefits of Traditional Methods such as CMMI[®] (see Table 2).⁹ Agile Methods emphasize teams, working software, customer collaboration, and responding to change, while Traditional Methods focus on contracts, plans, processes, documents, and tools.¹⁰ The SEI study identified 99 data points on cost, schedule, productivity, quality, satisfaction, and ROI gains from 25 organizations as reported by CMMI[®]-related literature from SEI conferences. It's important to note that CMMI[®] data are optimistic and often come from CMMI[®] proponents, rather than scholarly research studies such as experiments, surveys, or other scientific methods. Oftentimes, the percentages are only relative proportions and do not state the actual costs and benefits (e.g., large CMMI[®] initiatives cost millions of dollars and oftentimes do not succeed). Some of these data came from mixing and matching Traditional Methods such as Inspections, PSPsm, TSPsm, Six Sigma, and others to gain synergy not possible within a CMMI[®] environment. Nonetheless, these data represent a major milestone in the research on Traditional Methods for software process improvement, software development, and information systems (IS) research. Two similar studies on the costs and benefits of SW-CMM[®] were gathered by the Data and Analysis Center for Software (DACs)¹¹ and software development researchers in Israel.¹²

Table 1. Agile Methods Costs and Benefits

No.	Category	Low	Median	High	Points
1.	Cost	10%	26%	70%	9
2.	Schedule	11%	71%	700%	19
3.	Productivity	14%	122%	712%	27
4.	Quality	10%	70%	1,000%	53
5.	Satisfaction	70%	70%	70%	1
6.	ROI	240%	2,633%	8,852%	29

Table 2. Traditional Methods Costs and Benefits

No.	Category	Low	Median	High	Points
1.	Cost	3%	20%	87%	21
2.	Schedule	2%	37%	90%	19
3.	Productivity	9%	62%	255%	17
4.	Quality	7%	50%	132%	20
5.	Satisfaction	-4%	14%	55%	6
6.	ROI	200%	470%	2,770%	16

Using the SEI cost and benefit summary as a framework, cost, schedule, productivity, quality, satisfaction, and ROI data were gathered from over 300 scholarly articles about Agile Methods. In Table 1 and Table 2, the category represents the benefits of Agile and Traditional Methods, while the low, median, and high represent the range of reported benefits within each category. This was a laborious process, because relevant articles on Agile Methods had to be identified and categorized, and then cost and benefit data had to be extracted and normalized for comparison. The original goals were limited in scope and consisted of gathering a small amount of data in order to gain an appreciation for the range of costs and benefits possible with Agile Methods. However, this quickly blossomed into a two-month long effort due to the number of studies on Agile Methods, the amount of data, and the process of data cleansing for comparative analysis. In the end, cost, schedule, productivity, quality, and satisfaction data from 69 scholarly studies were utilized, consisting of 36 experiments, 25 cases, 6 surveys, and 2 simulations (see Table 3). On average, studies of Agile Methods reported 29% better cost, 91% better schedule, 97% better productivity, 50% better quality, 400% better satisfaction, and 470% better ROI than CMMI[®]. The complete results were compiled into an ROI spreadsheet model on the costs and benefits of Agile Methods and represent one of the largest collections of data on Agile Methods to-date.¹³ Several good studies of Pair Programming and Test Driven Development also served as an inspiration for this study as well as sources of additional cost and benefit data on Agile Methods.

Table 3. Agile Methods Costs and Benefits (identified from an analysis of over 300 studies)

No.	Author(s)	Year	Tech	Cost	Sched	Prod	Quality	Satis	Method	N
1.	Abrahamsson	2003	XP			88%			Case	4
2.	Abrahamsson	2007	General	70%	700%		250%		Case	1,800
3.	Al-Kilidar et al.	2005	PP				13%		Exp	121
4.	Arisholm et al.	2007	PP		11%		23%		Exp	295
5.	Back, Hirkman, & Milovanov	2004	XP				87%		Exp	8
6.	Bhat & Nagappan	2006	TDD				71%		Case	12
7.	Bipp, Lepper, & Schmedding	2008	PP				62%		Exp	95
8.	Canfora et al.	2006	PP			14%	20%		Exp	70
9.	Canfora et al.	2007	PP		39%		39%		Exp	18
10.	Cohn	2008	Scrum			405%	71%		Case	7
11.	Dalcher, Benediktsson, & Thorbergsson	2005	XP	21%		384%			Exp	55
12.	Damm & Lundberg	2006	TDD				56%		Case	100
13.	Drobka, Noftz, & Raghu	2004	XP			289%	63%		Case	29
14.	Erdogmus, Morisio, & Torchiano	2005	TDD			28%			Exp	24
15.	Fitzgerald, Hartnett, & Conboy	2006	Scrum				700%		Case	45
16.	Flohr & Schneider	2006	TDD			27%			Exp	18
17.	George	2002	TDD				16%		Exp	138
18.	George & Williams	2003	TDD				18%		Exp	24
19.	George & Williams	2004	TDD				18%		Exp	24
20.	Heiberg et al.	2003	PP				16%		Exp	100
21.	Huang & Holcombe	2008	TDD			172%			Exp	274
22.	Hulkko & Abrahamsson	2005	PP			18%	46%		Case	18
23.	Ilieva, Ivanov, & Stefanova	2004	XP	12%		41%	13%		Exp	8
24.	Janzen & Saiedian	2008	TDD				34%		Exp	64
25.	Jensen	2003	PP			127%	1,000%		Case	10
26.	Jones	2008	Scrum			74%			Case	5
27.	Kaufmann & Janzen	2003	TDD			50%	50%		Exp	8
28.	Kuppuswami et al.	2003	XP	28%					Sim	n/a
29.	Layman	2004	XP			61%	48%		Case	21
30.	Lui & Chan	2004	PP		24%				Exp	3
31.	Lui & Chan	2006	PP		23%				Exp	40
32.	Lui, Chan, & Nosek	2008	PP		70%				Exp	15
33.	Madeyski	2006	PP				14%		Exp	188
34.	Madeyski & Szala	2007	TDD			18%	45%		Case	1
35.	Mann	2004	TDD				81%		Case	7
36.	Maurer & Martel	2002	XP			66%			Case	9
37.	Maximilien & Williams	2003	TDD				50%		Case	9
38.	McDowell et al.	2003	PP				27%		Exp	555
39.	McDowell et al.	2006	PP				27%		Case	486
40.	Melis et al.	2006	TDD				36%		Case	4
41.	Mendes, Al-Fakhri, & Luxton-Reilly	2005	PP				10%		Exp	300
42.	Molokken-Ostvold & Jorgensen	2005	General		12%				Survey	42
43.	Muller	2005	PP				29%		Exp	38
44.	Muller	2006	PP		29%		11%		Exp	18
45.	Muller	2007	PP				50%		Exp	21
46.	Muller & Padberg	2003	XP	20%					Sim	n/a
47.	Nawrocki & Wojciechowski	2001	PP		25%		15%		Exp	21
48.	Nosek	1998	PP		29%		36%		Exp	15
49.	Pandey et al.	2003	PP		40%	20%	40%		Exp	10
50.	Phongpaibul & Boehm	2006	PP		24%		34%		Exp	104
51.	Reifer	2003	XP	10%	53%	20%			Survey	18
52.	Rico	2007	General	51%	65%	56%	63%	70%	Survey	122
53.	Saff & Ernst	2004	TDD				16%		Exp	39
54.	Sanchez, Williams, & Maximilien	2007	TDD				40%		Case	17
55.	Schatz & Abdelshafi	2005	Scrum			29%	30%		Case	90
56.	Schatz & Abdelshafi	2005	TDD				75%		Case	90
57.	Sutherland	2007	Scrum			712%			Case	5
58.	Talby et al.	2006	TDD			90%			Case	60
59.	Van Schoonderwoert	2006	XP			192%	89%		Case	4
60.	Vanhanen & Lassenius	2005	PP				42%		Exp	20
61.	Version One	2006	General	10%	18%	17%	17%		Survey	722
62.	Version One	2007	General	11%	16%	17%	17%		Survey	1,681
63.	Williams	2001	PP		47%		15%		Exp	41
64.	Williams et al.	2003	PP				16%		Exp	575
65.	Williams, Maximilien, & Vouk	2003	TDD				40%		Case	14
66.	Wilson, Hoskin, & Nosek	1993	PP				38%		Exp	34
67.	Wolf & Roock	2008	General		72%	78%	74%		Survey	200
68.	Xu & Rajlich	2006	PP		48%	201%	21%		Exp	12
69.	Ynchausti	2001	TDD				153%		Case	5

ROI Metrics and Models

A significant concept or principle within Agile Methods is the notion of creating business value, which often means delivering working software through the process of iterative development. This is clearly evident by analysis of the first principle of the Agile Manifesto, “Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.” This stands in opposition to the central concept or principle within some Traditional Methods in which creating processes and documentation is considered the main measure of business value.¹⁴ Within some Traditional Methods, writing documentation is considered paramount to the quality, maintainability, reliability, and safety of mission critical systems such as aviation electronics.¹⁵ While Agile Methods use programming for creating business value, some equate them with hacking, ill-conceived prototypes, and coding without documented requirements and design.¹⁶ The advent of Agile Methods was a return to fundamentals—That is, software craftsmanship versus documentation, which has been a mantra of the commercial software industry for years.¹⁷ Traditional Methods are usually used on extraordinarily large systems, in which public funds are necessary to pay for Acquisition Category I programs (e.g., spacecraft, aircraft, missiles, etc.).

Table 4. ROI Metrics (showing simplicity of return on investment formulas and their order of application)

Metric	Definition	Formula
Costs (sum of costs)	Total amount of money spent on Agile Methods	$\sum_{i=1}^n Cost_i$
Benefits (sum of benefits)	Total amount of money gained from Agile Methods	$\sum_{i=1}^n Benefit_i$
B/CR (benefit to cost ratio)	Ratio of Agile Methods benefits to costs	$\frac{Benefits}{Costs}$
ROI% (return on investment)	Ratio of adjusted Agile Methods benefits to costs	$\frac{Benefits - Costs}{Costs} \times 100\%$
NPV (net present value)	Discounted cash flows of Agile Methods	$\sum_{i=1}^{Years} \frac{Benefit_i}{(1 + Discount\ Rate)^{Years}} - Costs_0$
BEP (breakeven point)	Point when benefits exceed costs of Agile Methods	$\frac{Costs}{NPV} \times 60\ Months$
ROA (real options analysis)	Value realized from strategic delay due to risk	$N(d_1) \times Benefits - N(d_2) \times Costs \times e^{-Rate \times Years}$

$$d1 = [\ln(Benefits \div Costs) + (Rate + 0.5 \times Risk^2) \times Years] \div Risk \times \sqrt{Years}, \quad d2 = d1 - Risk \times \sqrt{Years}$$

However, Agile Methods elevate business value beyond just the activities of creating working software at regular intervals—Agile Methods go on to define business value in terms of ROI.¹⁸ This is clearly evident within Agile Methods such as Extreme Programming and Scrum, where user stories (requirements) are “prioritized” based on business value (e.g., ROI, NPV, etc.).¹⁹ With this second definition of business value in-mind, the most often cited measure of business value for prioritizing requirements is ROI, or any closely related family of business metrics.²⁰ ROI metrics are used to evaluate the economic value of one or more investments in information technology and are often expressed as simple ratios of benefits to cost, less the costs of course.²¹ Seven metrics were used for valuation of Agile Methods: Costs, Benefits, Benefit to Cost Ratio, Return on Investment%, Net Present Value, Break Even Point, and Real Options (see Table 4).²² ROI metrics are slight variations created over the last 100 years (e.g., benefits relative to costs) and each is good for measuring the business value of Agile Methods with increasing accuracy.

Agile Methods Costs

As shown in Table 4, the first basic input necessary to estimate the ROI of Agile Methods is cost, so it was necessary to identify studies of Agile Methods with cost measures for estimating ROI. Therefore, software productivity and quality measurement data such as lines or code or function points and quality measures such as defect density had to be identified in order to estimate ROI. This data could then serve as the basis for establishing the empirical cost estimating relationships necessary to design top down parametric models for estimating the costs of using Agile Methods.

Table 5. Agile Methods Productivity and Quality Data (identified from an analysis of over 300 studies)

No.	Author(s)	Year	Tech	LOC/Hour	Def/KLOC	Method	N
1.	Abrahamsson	2003	XP	19.2550	2.1450	Case	4
2.	Abrahamsson & Koskela	2004	XP	16.9000	1.4300	Case	4
3.	Back, Hirkman, & Milovanov	2004	XP	8.0000	0.7000	Exp	8
4.	Bowers et al.	2002	XP	18.1731	0.0325	Case	???
5.	Dalcher, Benediktsson, & Thorbergsson	2005	XP	14.8667		Exp	55
6.	Hashmi & Baik	2008	XP	16.8420		Case	19
7.	Ilieva, Ivanov, & Stefanova	2004	XP	20.2030	0.0032	Exp	8
8.	Layman	2004	XP	9.1154	0.6250	Case	21
9.	Layman et al.	2006	XP	13.3846	1.6200	Case	8
10.	Manzo	2002	XP	43.0000	0.5000	Case	17
11.	Maurer & Martel	2002	XP	17.0000		Case	9
12.	Van Schoenderwoert	2006	XP	3.5000	0.1700	Case	4
13.	Williams, Layman, & Krebs	2004	XP	9.8077	0.2400	Case	19
14.	Huang & Holcombe	2008	TDD	12.3800		Exp	274
15.	Madeyski & Szala	2007	TDD	46.1800		Case	1
16.	Maximilien & Williams	2003	TDD		3.7000	Case	9
17.	Williams, Maximilien, & Vouk	2003	TDD		0.6100	Case	14
18.	Baheti, Gehringer, & Stotts	2002	PP	16.6370		Exp	132
19.	Erdogmus & Williams	2003	PP	43.4780	5.8500	Case	41
20.	Hulkko & Abrahamsson	2005	PP	15.6667	4.1500	Case	18
21.	Nawrocki & Wojciechowski	2001	PP	49.2500		Exp	21
22.	Pandey et al.	2003	PP	22.4462	2.3900	Exp	10
23.	Vanhanen & Korpi	2007	PP	15.4667	0.5500	Case	4
24.	Vanhanen & Lassenius	2005	PP	17.8403	0.3250	Exp	20
25.	Xu & Rajlich	2006	PP	86.4502	0.8651	Exp	12
26.	Cohn	2008	Scrum	5.9050	2.9000	Case	7
27.	Jones	2008	Scrum	5.7400	8.5000	Case	5
28.	Schatz & Abdelshafi	2005	Scrum		0.4350	Case	90
29.	Sutherland	2006	Scrum	4.6858		Case	5

Data from Table 5 were averaged to establish the cost estimating relationships to design top down parametric models used to estimate the ROI of Agile Methods (see Table 6 and Table 7). An average programming productivity measurement was taken of the 26 data points in Table 6 and was used to construct an empirical cost model called ‘Agile Methods’ for the entire data set. The cost and quality models in Table 6 and Table 7 were then be used to estimate the software development and maintenance costs of Agile Methods along with their benefits (hence, ROI). The method for estimating the ROI of Agile Methods will be explained in the next section.

Table 6. Agile Methods Cost Models

No.	Tech	Low	Median	High	Pts	Cost Model
1.	XP	03.5000	16.1575	43.0000	13	LOC ÷ 16.1575
2.	TDD	12.3800	29.2800	46.1800	2	LOC ÷ 29.2800
3.	PP	15.4667	33.4044	86.4502	8	LOC ÷ 33.4044
4.	Scrum	04.6858	05.4436	05.9050	3	LOC ÷ 05.4436
5.	Agile	03.5000	21.2374	86.4502	26	LOC ÷ 21.2374

Table 7. Agile Methods Quality Models

No.	Tech	Low	Median	High	Pts	Quality Model
1.	XP	0.0032	0.7466	2.1450	10	0.7466 × KLOC × 100
2.	TDD	0.6100	2.1550	3.7000	2	2.1550 × KLOC × 100
3.	PP	0.3250	2.3550	5.8500	6	2.3550 × KLOC × 100
4.	Scrum	0.4350	3.9450	8.5000	3	3.9450 × KLOC × 100
5.	Agile	0.0032	1.7972	8.5000	21	1.7972 × KLOC × 100

Agile Methods Benefits

There are two ways to increase business value or ROI: (a) increasing volume and revenue while maintaining current costs or (b) reducing costs while maintaining current volume and revenue.²³ This study uses the latter (e.g., reduce costs while maintaining volume and revenue), which is known as cost of quality (CoQ), total cost of ownership (TCO), and total lifecycle cost (TLC).²⁴ Unless previously stated, we can't predict the business value or ROI of an Agile Methods study; However, we can predict costs of software development and maintenance given the right data. This is especially true for software maintenance costs, which can be predicted using software quality measurements from the software development phase such as defect density (Def/KLOC). Together, the software development and maintenance costs constitute the CoQ, TCO, and TLC; That is, cradle-to-grave costs of software analysis, design, development, test, and maintenance.

Table 8. Total Lifecycle Costs

No.	Tech	Total Lifecycle Cost Model	Costs
1.	XP	$(10,000 \div 16.1575 + 0.7466 \times 10 \times 100) \times 100$	\$136,548
2.	TDD	$(10,000 \div 29.2800 + 2.1550 \times 10 \times 100) \times 100$	\$249,653
3.	PP	$(10,000 \div 33.4044 + 2.3550 \times 10 \times 100) \times 100$	\$265,437
4.	Scrum	$(10,000 \div 05.4436 + 3.9450 \times 10 \times 100) \times 100$	\$578,202
5.	Agile	$(10,000 \div 21.2374 + 1.7972 \times 10 \times 100) \times 100$	\$226,805

In order to estimate total lifecycle costs, both software development and maintenance costs have to be estimated and then added together using cost and quality models from Table 6 and Table 7. First, software development costs are estimated using the cost models from Table 6 and then the software maintenance costs are estimated utilizing the quality models from Table 7 (see Table 8). A baseline size of 10,000 lines of code is used for software development and a baseline effort of 100 hours is used for software maintenance (along with a conversion rate of \$100 U.S. dollars). The software development cost model is a simple linear model based on productivity measures, but maintenance cost is based on 100 hours of effort for each defect which escapes development. This methodology assumes a ratio of 1:10:100 ratio for pre-test, test, and maintenance effort.²⁵

Table 9. Total Lifecycle Benefits

No.	Tech	Total Lifecycle Benefit Model	Benefits
1.	XP	$(10,000 \times 10.51 - 6,666.67 \times 9) \times 100 - \text{TLC}$	\$4,373,449
2.	TDD	$(10,000 \times 10.51 - 6,666.67 \times 9) \times 100 - \text{TLC}$	\$4,260,344
3.	PP	$(10,000 \times 10.51 - 6,666.67 \times 9) \times 100 - \text{TLC}$	\$4,244,560
4.	Scrum	$(10,000 \times 10.51 - 6,666.67 \times 9) \times 100 - \text{TLC}$	\$3,931,795
5.	Agile	$(10,000 \times 10.51 - 6,666.67 \times 9) \times 100 - \text{TLC}$	\$4,283,192

In order to estimate total lifecycle benefits, the total lifecycle costs of using Agile Methods were subtracted from the estimated total lifecycle costs of Traditional Methods (as shown in Table 9). Some assumptions were that the total lifecycle costs of Traditional Methods exceeded the total lifecycle costs of Agile Methods (and Agile Methods don't exceed costs of Traditional Methods). The major terms of the benefit models represent the total lifecycle costs of a 10% defect rate and a 0.51 LOC/hour productivity rate (less the benefits of finding 66.67% of the defects by testing). The TLC methodology used here to estimate the costs, benefits, and ROI has been outlined in a number of publications²¹ and the complete results are available in an ROI spreadsheet model.¹³

Agile Methods Return on Investment

The total lifecycle cost and benefit models for each of the Agile Methods from Table 8 and Table 9 were combined with the ROI metrics from Table 4 to estimate the ROI data shown in Table 10. Extreme Programming had the lowest overall total lifecycle cost at \$136,548, followed by Test Driven Development, Pair Programming, and Scrum around \$249,653, \$265,437, and \$578,202. As a result, Extreme Programming had the highest return on investment at 3,103%, followed by Test Driven Development, Pair Programming, and Scrum at around 1,607%, 1,499%, and 580%. Pair Programming had the highest overall average productivity at 33 LOC/Hour, followed by Test Driven Development, Extreme Programming, and Scrum around 29, 16, and 5 LOC/Hour. Extreme Programming had the highest overall quality at 0.8 Defects/KLOC, followed by Test Driven Development, Pair Programming, and Scrum at around 2.2, 2.4, and 4 Defects/KLOC. Extreme Programming had half the productivity of Pair Programming; however it had six times better quality than all the other methods combined leading to lower total costs and higher ROI.

Table 10. Agile Methods Return on Investment (estimated from productivity and quality data)

No.	Tech	Prod.	Quality	Costs	Benefits	B/CR	ROI%	NPV	BEP	Cost/Per	Risk	ROA
1.	XP	16.1575	0.7466	\$136,548	\$4,373,449	32:1	3,103%	\$3,650,401	\$4,263	\$34,137	21.23%	\$4,267,105
2.	Agile	21.2374	1.7972	\$226,805	\$4,283,192	19:1	1,788%	\$3,481,992	\$12,010	\$56,701	62.27%	\$4,110,308
3.	TDD	29.2800	2.1550	\$249,653	\$4,260,344	17:1	1,607%	\$3,439,359	\$14,629	\$62,413	67.95%	\$4,074,506
4.	PP	33.4044	2.3550	\$265,437	\$4,244,560	16:1	1,499%	\$3,409,908	\$16,599	\$66,359	71.30%	\$4,050,918
5.	Scrum	5.4436	3.9450	\$578,202	\$3,931,795	7:1	580%	\$2,826,320	\$85,029	\$144,551	100.00%	\$3,660,805

The ROI data for Agile Methods in Table 10 were combined with prior ROI data for Traditional Methods²² in order to compare the ROI of Agile vs. Traditional Methods (as shown in Table 11). Some Traditional Methods were expected to top the list in this analysis (which they did), because the ROI methodology used in this study rewards methods with high quality (low defect density). Most Agile Methods were expected to rank better than expensive Traditional Methods (which they did), because the costs of implementing expensive Traditional Methods tends to be high. Although, Extreme Programming was expected to top the list of Agile Methods (which it did), Extreme Programming ranked third ahead some of the industry's premier Traditional Methods. Extreme Programming ranked almost second on the strength of quality rather than productivity, which was half its nearest competitors, because total lifecycle cost rewards quality handsomely. The best traditional methods remove defects before testing to minimize total lifecycle costs.

Table 11. Agile vs. Traditional Methods Return on Investment (estimated from productivity and quality data)

No.	Method	Costs	Benefits	B/CR	ROI%	NPV	BEP	Cost/Per	Risk	ROA
1.	PSP sm	\$105,600	\$4,469,997	42:1	4,133%	\$3,764,950	\$945	\$26,400	6.44%	\$4,387,756
2.	Inspection	\$82,073	\$2,767,464	34:1	3,272%	\$2,314,261	\$51,677	\$20,518	26.78%	\$2,703,545
3.	XP	\$136,548	\$4,373,449	32:1	3,103%	\$3,650,401	\$4,263	\$34,137	30.78%	\$4,267,105
4.	TSP sm	\$148,400	\$4,341,496	29:1	2,826%	\$3,610,882	\$5,760	\$37,100	37.33%	\$4,225,923
5.	Agile	\$226,805	\$4,283,192	19:1	1,788%	\$3,481,992	\$12,010	\$56,701	61.83%	\$4,110,118
6.	TDD	\$249,653	\$4,260,344	17:1	1,607%	\$3,439,359	\$14,629	\$62,413	66.13%	\$4,073,167
7.	PP	\$265,437	\$4,244,560	16:1	1,499%	\$3,409,908	\$16,599	\$66,359	68.67%	\$4,048,404
8.	SW-CMM [®]	\$311,433	\$3,023,064	10:1	871%	\$2,306,224	\$153,182	\$77,858	83.51%	\$2,828,802
9.	Scrum	\$578,202	\$3,931,795	7:1	580%	\$2,826,320	\$85,029	\$144,551	90.38%	\$3,622,271
10.	ISO 9001	\$173,000	\$569,841	3:1	229%	\$320,423	\$1,196,206	\$43,250	98.66%	\$503,345
11.	CMMI [®]	\$1,108,233	\$3,023,064	3:1	173%	\$1,509,424	\$545,099	\$277,058	100.00%	\$2,633,052

Conclusion

The main purpose of this article was to identify, analyze, and summarize the costs and benefits of Agile Methods found in the best possible literature (e.g., experiments, surveys, and case studies). Not only did we find 69 studies with cost and benefit data, but we found more, better quality studies with an average of 200% better performance than big and expensive Traditional Methods. We also found 29 studies of Agile Methods with the productivity and quality data necessary to estimate ROI using metrics that would enable the comparison of Agile vs. Traditional Methods. This analysis showed that Agile Methods are almost as good as the best Traditional Methods under the light of total lifecycle cost analysis, which tends to reward methods with high quality.

- Agile Methods weren't born yesterday. Agile Methods are based on early customer involvement, iterative development, self organizing teams, and adaptability to change, which originated from agile, new product development approaches dating back to the 19th century.
- Agile Methods scale up to large problems. Agile, new product development methods have been used for many large-scale, complex research and development projects such as Lockheed's SR-71, NASA's Apollo, and the Jet Propulsion Laboratory's deep space probes.
- Agile Methods may not be in use by very large organizations. Agile Methods are used by 70% of small to medium-sized projects; however, larger projects use Traditional Methods, so the relevance of Agile Methods to large, complex projects needs to be convincingly made.
- Agile Methods can learn something from traditional methods. Agile Methods should apply traditional quality and reliability theory, which holds that defects are less expensive to eliminate early in the lifecycle and late defect removal has a negative, multiplicative effect.
- Agile Methods hybrids are the latest trends. Agile Methods are being combined with one another to gain synergies not possible with any one approach, such as XP and Scrum. Agile and Traditional Methods are also being combined to tap into one another's capabilities.
- Agile Methods require non-traditional measures. Traditional Methods were optimized for productivity and quality, which rewards them using total lifecycle cost analysis; but Agile Methods should focus on project success and customer satisfaction where they shine best.
- Agile Methods lend themselves to advanced economic models. Agile Methods lend themselves to valuation methods such as real options; therefore researchers should focus on real options as a way of explaining the superiority of Agile Methods on complex projects.
- Agile Methods adoption involves traditional critical success factors. Executive commitment, resources, leadership, strategy, culture, incentives, training, tools, execution, consulting, measurement, and improvement are vital to the adoption of Agile Methods.
- Agile Methods adoption also involves non-traditional critical success factors. Lest we forget the Agile Manifesto, emphasis on individuals and interactions, working software, customer collaboration, and responding to change are non-traditional critical success factors.

In conclusion, not all Agile and Traditional Methods are created equal, there are pitfalls for using any method with a low ROI, and there are lessons to be learned from the best software methods. However, it may not be fair to compare methods optimized for productivity and quality to those optimized for speed, satisfaction, project success, and optimal ROI in the face of increasing risk. It's important to note that the power of Agile Methods is not in minimizing lifecycle costs, but maximizing business value through successful delivery of working software in the face of risk. Agile Methods are a unique paradigm, which cannot be easily grasped through traditional means.

References

1. Rico, D. F. (2007). *Effects of agile methods on website quality for electronic commerce*. Retrieved August 10, 2008, from <http://davidfrico.com/rico07q.pdf>
2. Rico, D. F., Sayani, H. H., & Field, R. F. (2008). History of computers, electronic commerce, and agile methods. In M. V. Zelkowitz (Ed.), *Advances in computers: Emerging technologies, Vol. 73*. San Diego, CA: Elsevier.
3. Thomke, S. (2003). *Experimentation matters: Unlocking the potential of new technologies for innovation*. Boston, MA: Harvard Business School Press.
4. Rico, D. F. (2008). Effects of agile methods on website quality for electronic commerce. *Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008), Waikalooa, Big Island, Hawaii*.
5. Highsmith, J. A. (2002). *Agile software development ecosystems*. Boston, MA: Addison Wesley.
6. Johnson, M. (2002). *Agile methodologies: Survey results*. Victoria, Australia: Shine Technologies.
7. Ambler, S. W. (2006). *Agile adoption rate survey: March 2006*. Retrieved September 17, 2006, from <http://www.ambysoft.com/downloads/surveys/AgileAdoptionRates.ppt>
8. Fitzgerald, B., Hartnett, G., & Conboy, K. (2006). Customising agile methods to software practices at intel shannon. *European Journal of Information Systems, 15*(2), 200-213.
9. SEI. (2005). *SEI performance results*. Retrieved August 10, 2008, from <http://www.sei.cmu.edu/cmmi/2005results.html>
10. Agile Manifesto. (2001). *Manifesto for agile software development*. Retrieved November 29, 2006, from <http://www.agilemanifesto.org>
11. Rico, D. F. (2000). *Using cost benefit analyses to develop software process improvement (SPI) strategies* (Contract Number SP0700-98-D-4000). Rome, NY: Air Force Research Laboratory—Information Directorate (AFRL/IF), Data and Analysis Center for Software (DACS).
12. Galin, D., & Avrahami, M. (2006). Are CMM program investments beneficial? Analyzing past studies. *IEEE Software, 23*(6), 81-87.
13. Rico, D. F. (2008). *What is the ROI of agile vs. traditional methods? An analysis of extreme programming, test-driven development, pair programming, and scrum (using real options)*. Retrieved June 28, 2008, from <http://davidfrico.com/agile-benefits.xls>
14. International Standards Organization. (2008). *Systems and software engineering: Software life cycle processes (ISO/IEC 12207)*. Geneva, Switzerland: Author.
15. Radio Technical Commission for Aeronautics. (1999). *Software considerations in airborne systems and equipment certification (DO-178B)*. Washington, DC: Author.
16. McCormick, M. (2001). Programming extremism. *Communications of the ACM, 44*(6), 109-111.
17. McBreen, P. (2001). *Software craftsmanship: The new imperative*. Boston, MA: Addison-Wesley.
18. Agile Project Leadership Network. (2008). *Declaration of interdependence*. Retrieved August 10, 2008 from, <http://www.pmdoi.org>
19. Beck, K., & Fowler, M. (2001). *Planning extreme programming*. Upper Saddle River, NJ: Addison-Wesley.
20. Schwaber, K. (2004). *Agile project management with scrum*. Redmond, WA: Microsoft Press.
21. Rico, D. F. (2004). *ROI of software process improvement: Metrics for project managers and software engineers*. Boca Raton, FL: J. Ross Publishing.
22. Rico, D. F. (2007). *Practical metrics and models for ROI with real options*. Retrieved November 28, 2007, from <http://davidfrico.com/rico07b.pdf>
23. Garrison, R. H., & Noreen, E. W. (1997). *Managerial accounting*. Boston, MA: McGraw-Hill.
24. Campanella, J. (1999). *Principles of quality costs: Principles, implementation, and use*. Milwaukee, WI: Quality Press.
25. McGibbon, T., Ferens, D., & Vienneau, R. L. (2007). *A business case for software process improvement (2007 update): Measuring the return on investment from software engineering and management* (Contract Number SP0700-98-D-4000). Griffiss AFB, NY: AFRL/IFT, DACS.