The Agile Research Penultimatum

Thomas Way, Sandhya Chandrasekhar and Arun Murthy
Center of Excellence in Enterprise Technology
Department of Computing Sciences
Villanova University, Villanova PA 19085
thomas.way@villanova.edu

Abstract - *Agile software development is a widely used and successful methodology for organizing and managing software product development in an industry setting. For academic research projects in Computer Science, the development of software is often a major component and the use of agile methods may be appropriate. The fundamental difference in how software engineering is performed in an academic research setting as compared with that of industry, however, means that a strict application of agile approaches may not be ideal. In this paper, we adapt the Agile Manifesto to the needs of an academic research project. Four case studies are presented, two from industry, one from academia, and one from an industry-academia collaborative project to support the adaptations, motivating a set of proposed agile research policies called the Agile Research Penultimatum.*

Keywords: Agile methods, agile research, software engineering processes, research project management.

1 Introduction

Agile software development methods are currently popular in industry, and represent a natural evolutionary step in software engineering processes [5,6]. Although unlikely to be Brooks' elusive "silver bullet" [2], agile methods have been widely adopted and proven to be quite successful at improving productivity, and ultimately the bottom line, when used appropriately and consistently [4].

The production of professional quality software typically follows a disciplined approach [12], inspired by operations management and other well-defined management processes. The fundamental "Waterfall" approach considers a project as having a series of stages, where each stage must be accomplished and accepted before starting the next one. There are recognized disadvantages to this stepwise approach, with frequent changes to requirements among the most significant challenges [12]. Managing rapid change in a software project requires software processes to be adaptive.

This adaptability is accommodated in a refined framework for software development called Iterative development [7]. Iterative development defines the development lifecycle as consisting of several iterations, rather than a less adaptive stepwise or Waterfall approach [12]. Each iteration is a project cycle, with activities such as requirements analysis, design, implementation and testing performed incrementally. The goal for each iteration is a limited product "release" which consists of the most recent version, often partially completed, of the working software.

Agile methods are a more recent evolution of the iterative methodology and also employ other practices which are customized for specific projects [1,4]. Extreme Programming, Scrum, Agile Unified Process, Rational Unified Process, Agile Modeling, and most recently, Lean Software Development [10,11] are some of the more popular and well known Agile methodologies. These methodologies are used in a significant subset of the software industry for development of software [1,12]. For example, the widely adopted Scrum software project management process relies on monthly and daily iterations to drive progress (Figure 1). Scrum-managed projects maintain a "backlog" of goals or tasks and use brief, daily "Scrum" meetings and monthly "sprints" as a way to implement the agile methodology.

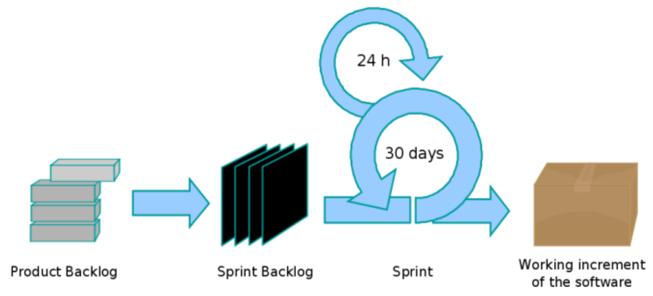

Figure 1. Diagram of Scrum agile method [13].

The goals of agile methods focus on rapid and seamless adaptation to change which is enabled by a management process that incorporates frequent, periodic and regular evaluations, and a team-oriented culture that encourages teamwork and self-organization. Agile methodologies emphasize continuous improvement, and continuously available and updated deliverables, always with the commitment to fulfilling customer needs. While effective for a wide range of industrial software projects, the applicability of these rapid and highly adaptive methodologies to research projects has not been established other than in limited studies constrained to specific variations of these techniques [8,14].

Research projects, even those which involve a significant software development component, have constantly evolving requirements as a result of the process of scientific inquiry, with new findings leading to new interpretations or complete redefinitions of the original goals. These new findings are unpredictable, both in their timing and their outcomes. This unpredictability of the course of a project and its defined goals makes the typical academic research process an inherently poor fit for the more rapid and rigid management approaches such as Scrum and the other agile methodologies.

Although research projects often follow a less predictable course, there are many aspects of agile methodologies that are well suited to managing such a project. For example, an iterative refinement approach that embraces frequent evaluation, regular meetings, and an explicit (albeit malleable) “backlog” of goals and tasks, can be used in a research project to move towards a specific solution in a positive way. Because the customer of a research project tends toward the extremes, either as an intimately involved researcher or a remote, grant-funding entity, the customer orientation of most agile methodologies requires the most adaptation. In addition, although the need for continuous refinement of research goals is less clear, the core concepts of an agile approach hold promise for software-based research projects, provided they are judiciously adapted and applied.

In our experience, research projects that have a software development component tend to be managed in an ad hoc manner. This extemporized approach arises of necessity due to the discontinuous nature of an academic researcher’s daily schedule, but also for lack of a properly designed and well suited management process. The result is often a less efficient project and a less effective research program than if an appropriate management process was followed. In this paper, we present an adaptation of some of the more commonly used agile software development techniques and methodologies, closely following the framework put forth in the Agile Manifesto. Based on our experiences in industry, academic research, and a combined industry-academic research setting, we have developed this adaptation of the agile process suited to the needs of a software oriented research project.

2 Agile Software Development

Although software development projects long have made use of selected ideas that are now defined as “agile methodologies,” the drafting of the Agile Manifesto by 17 experienced engineers and managers in 2001 marks a turning point in software engineering process management [5]. Although its ideas have been adopted in a variety of agile methodologies since, the Agile Manifesto in its original form (Figure 2) provides a guide which is both breath-taking in its simplicity, yet powerfully adaptable in its aims.

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Figure 2. The Agile Manifesto [5].

The Agile Manifesto provides a high-level process management philosophy in active language as a guide to effectively manage a software project. The original authors provided depth and motivation through 12 accompanying principles (Figure 3) that extend and support each of the four primary value statements in the Manifesto.

Principles behind the Agile Manifesto

We follow these principles:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Figure 2. Twelve Principles of Agile Software [5].

3 Project Case Studies

In order to identify, and eventually formulate, our agile research guidelines, we briefly examine four case studies. Each are based on the experience of members of our faculty and graduate students working on a number of industry or academic research projects, including as part of outside consulting and resulting internal technical reports [3,9]. Where data was available from multiple case studies with similar scenarios, it has been combined here to create a case study for a representative, composite company.

3.1 Study 1: Small Company A

In this study, a small software development company, Company A, of approximately 30 employees made use of an Extreme Programming (XP) methodology to manage the development process. At Company A, which typically provided software services on projects of limited scope and duration to the financial and medical industries, as well as some in-house new product development, selected XP and Scrum practices were adopted, although with considerable accommodation for the individual working styles and expectations of their software engineers.

Test-driven development and Pair Programming were the norm, yet a handful of experienced programmers were allowed to work independently due to personal preference. Often the scope of a given project was very small, with a project team consisting only of one or two developers. The variations were viewed by all involved as productive, although some of the individual Pair Programmers expressed a belief that they would be more productive coding alone. While the value of working in pairs is well noted [1], this example illustrates the importance in accommodating individual programmer work styles. A programmer or engineer is likely to be more productive when comfortable with the working environment, and a significant percentage of programmers and engineers feel most comfortable and productive coding alone.

Daily Scrum meetings were not regularly held at Company A, although a regular weekly engineering Scrum meeting was held that served the same purpose. Because many team members were juggling multiple projects and responsibilities, including meeting with external and remote customers, it was often not possible for all members of a given team to be available and present at the same time each day, but a weekly meeting was more practical. Ad hoc meetings were often held in smaller groups or pairs, and generally this approach worked well in keeping members connected and "in the loop." In situations where all team members were assigned to a single team this was not an issue, but is a reminder that a rigid meeting schedule is not always a practical or desirable expectation.

3.2 Study 2: Large Company B

Some of the issues of individual variability were seen to be reduced in the larger Company B study. Company B served primarily as a defense software contractor, with offices at multiple geographic locations throughout the United States. As a result of the larger scale of projects and their corresponding budgets, with the exception of higher level managers, team members were generally assigned to a single project team. Projects at Company B were managed using a strict implementation of the Scrum agile methodology, which enabled Company B to successfully manage multiple, simultaneous, high-profile projects with multiple industry and military stakeholders.

Daily "stand-up" Scrum meetings were held, usually with excellent attendance and within the designated timebox. A certified Scrum Master was shared among projects, and was frequently present to assist with meeting and project management as needed. Notably, the Scrum Master would sternly enforce adherence to the Scrum methodology at the first hint of deviation, particularly when there was any appearance of a tendency toward a Waterfall approach [12]. A designated Customer served as the advocate for the remote client or customer as needed. New hires tended to have the most difficulty meeting the expectations of the Scrum approach, but quickly conformed and projects tended to run smoothly.

Monthly "sprints" culminated with "sprint reviews," followed by a "sprint retrospective and planning" meeting. The regularity of the Scrum approach offered a comfortable predictability to the external customers (military, industry), which is believed among management to be a primary reason for Company B's continued success in acquiring defense contracts. Newer employees were initially hesitant to provide WAGs (Wild-Assed Guesses) when committing to stories in the project backlog, possibly due to lack of specific project experience. The strictness of the Scrum approach used, and the implication of failure (usually minor) of not completing a story in the predicted time, often seemed to be an additional cause among all team members for hesitancy in producing WAGs. However, projects were generally successful and the Scrum approach continued to delivery quality software that kept the customers and other stakeholders satisfied.

3.3 Study 3: Academic Research Project

Software was developed for an academic research project at a large, research university by a team consisting of a faculty advisor, three graduate and two undergraduate students. Management was very ad hoc, although weekly research meetings were well attended and served as motivation for continued progress for each team member. Tasks tended to be assigned by matching project needs with the individual capabilities of team members. The research focus of the project was determined by the dissertation goals of one of the graduate students, who in an industry software setting would be the Customer.

Student schedules are unpredictable, as is the research process in general, so the project proceeded in a very bursty fashion. Short, intensely productive periods were interspersed with long stretches of minimal progress, as code was refactored, literature reviews were performed, or outside commitments were fulfilled. There was neither the expectation nor the possibility of a daily research meeting, as student and faculty schedules and responsibilities were mostly divergent.

Electronic communication (email, IM) and spontaneous small meetings were facilitated by the research lab office facilities. Some students produced their best work during late night hours, while others kept to a more work-a-day schedule, yet found common ground in the shared laboratory facilities. Occasional discussions were the norm, but never among more than two or three team members at a time. In spite of the lack of predictability of communication, development efforts produced excellent results as evidenced by a number of published conference papers and journal articles.

3.4 Study 4: Industry & Academic Partnership

As the pace of change in technology has increased, so, too has the desire to see academic research be transferred to industry in a timely fashion. To support this goal, a defense contract was awarded to a collective of faculty researchers at a balanced teaching-research university with the obligation to partner with a large, defense software contractor (in this case, Company B, above). The logistics of organizing a large scale research and software development project were enormous, with a total of eight faculty, two research programmers, 12 students, and 10 industry partners comprising the project team. The project involved developing modeling and simulation techniques, conducting research on various implementations of the techniques being modeled, and then implementing selected techniques in a final software product.

The Scrum approach was applied, and worked very well for the industry-centered team members, as it was already an ingrained part of their development culture. For the academic side of the team, the Scrum approach was not ideal. Because of the unpredictable and intermittent schedule of the faculty involved, it was very difficult to coordinate meeting times. Scrum emphasizes the daily, in-person meeting, typically of no more than 15 to 20 minutes in length. Because the driving distance between the industry and university meeting locations was 30 minutes, the decision was made to attempt to hold a daily, teleconferenced, Scrum meeting with all available partners dialing in. This was initially successful, but attendance dwindled over time and dropped off significantly during peak academic life-cycle activities (exams, mid-term, etc.).

In response, the project was reorganized as two, parallel projects. The academic side of the project was responsible for “getting out ahead” of the industry side to conduct research, develop models, and suggest implementation strategies. This enabled the faculty, research programmers and students to follow a more typical academic research project schedule that accommodated the scheduling peculiarities of the academic world. The industry side of the team continued to follow a strict Scrum method, and was able to accomplish a significant amount of well-tested development of the original models and concepts conceived by the academic research partners.

At one noteworthy point in the project, the Customer determined that it was abandoning a significant implementation goal in favor of a new goal. This dramatic change of direction caused some degree of angst among the industry team members as they revised the project backlog and reorganized team members to take on new tasks. The impact to the academic research side of the team was less dramatic, and some of the original research was concluded, written up and published, since the results were valid and significant regardless of the software product status.

3.5 Discussion

What these four projects reveal is that there is a diversity of needs across industry and academia. Large organizations with multiple projects and large teams require different organization than small, academic research or industry development teams.

Significantly, the differing degrees of predictability of scheduling and progress provide significant contrast between the industry software development project and its counterpart, the academic research project involving software development. Academic schedules for faculty and students are inherently variable, while schedules for individuals working in industry are much more regular.

In the next section, we propose a series of recommendations for adapting various aspects of agile development to be more suitable for the typical software-based academic research project or small industry development project.

4 Agile Research Penultimatum

Motivated by our experiences in industry and academic software development projects, we discovered that there were a number of features of an agile approach that were well suited to a research project, with others that were not well suited. Over the course of a three year study, we drafted and refined our Agile Research Penultimatum. Although a major motivation of software engineers is the idea of reuse, we did not feel that our principles were worthy of the term “manifesto,” which is defined in the dictionary as “a public declaration of intentions, opinions, objectives, or motives, as one issued by a government, sovereign, or organization,” and thus intended for widespread reuse. Rather, we chose the root term “penultimate” as more appropriate, meaning “next to the last.” Our goal for this Agile Research Penultimatum (Figure 3) is to codify a starting point of workable practices (not necessary “best practices”) that, as with research, can be revised as the needs of an individual project warrant.

Penultimatum for Agile Software Research

We are uncovering better ways of conducting software-based research by doing it and helping others do it. Through this work we have come to value:

- Individuals and interactions over processes and tools
- Research process over comprehensive documentation
- Collegial collaboration over endless paperwork
- Adapting to discovery over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Figure 3. The Agile Research Penultimatum, parts taken from or based upon the Agile Manifesto [5].

The Penultimatum and its principles (Figure 4) are intended as a starting point for managing research projects using an agile approach, while acknowledging the less predictable path which research projects can often follow.

Principles behind the Agile Research Penultimatum

We follow these principles:

1. Our highest priority is to perform quality research through consistent effort and regular publication.
2. Welcome the unexpected, although better early than late in the process. Agile research processes enable early discovery, so care should be taken to minimize change later.
3. Maintain research documentation continually, updating a notebook or wiki as work is done and discoveries are made.
4. Faculty advisors and graduate students must meet weekly or regularly throughout the project.
5. Build projects around shared research goals that motivate faculty and student alike. Establish a work environment to support their individual needs, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a research team is face-to-face conversation, but email, instant-messaging, wikis and blogs are essential media, as well.
7. Published papers, technical reports, literature surveys and working research software are the primary measures of progress.
8. Agile research processes adapt to the highly variable nature of the academic schedule, recognizing that the pace will vary dramatically over the course of a project.
9. Continuous attention to proper citation, short- and long-range planning, and maintaining forward momentum enhances research productivity.
10. Simplicity is a worthwhile goal, yet recognizing when complex problems often require complicated solutions is essential.
11. The best research emerges from self-motivated, highly-organized teams of one or more, yet chaos also can be an ally to discovery.
12. At regular intervals, individuals reflect on the effectiveness of their contribution to the team and adjust their behavior accordingly.

Figure 4. Twelve Principles of Agile Research, with significant inspiration from the AgileManifesto [5].

Both of the lists of value statements and principles are clearly inspired by the original Agile Manifesto, borrowing heavily from their intention, tone and phrasing [5]. This similarity is intended as an homage to the significance of the original, and as a way to reinforce both those original concepts and our new research-oriented principles as tools for good research project management. Because research projects, particularly those in an academic setting, are often much less compatible with a more rigid agile software approach, we justify the adaptation and revision of each of the original statements from the Agile Manifest and the Twelve Principles as follows:

4.1 Value Statements

Individuals and interactions over processes and tools

This first value statement survives in tact from the original Agile Manifest. Academic research is personal and collaborative, yet often the collaborations are through the research process and use or development of software tools.

Research process over comprehensive documentation

Producing the output of research, the papers, surveys, notes, and bibliographies, are only more important than comprehensive documentation because they combine to form that documentation, which is the end goal.

Collegial collaboration over endless paperwork

The ability of a researcher to interact and form partnerships, rather than attempting to conduct research in an intellectual or collegial vacuum, are the smoothest paths to successful research.

Adapting to discovery over following a plan

Having well-defined short-term and long-term plans are vital, yet the vagaries of research require that the researcher be ready to change course as new discoveries are made.

4.2 Principles

Our highest priority is to perform quality research through consistent effort and regular publication.

An excellent way to focus research and generate quality results is to identify one or more publication outlets, and target short-term research efforts to those outlets, making sure that they support the longer-term research goals of the project.

Welcome the unexpected, although better early than late in the process. Agile research processes enable early discovery, so care should be taken to minimize change later.

Many years will be invested in a research project and the further along a project is the less welcome are dramatic changes of course. Early in a project, when the direction and goals are less well known, unexpected discoveries are a necessary part of the process.

Maintain research documentation continually, updating a notebook or wiki as work is done and discoveries made.

Good researchers maintain a note book or other continually updated log of research efforts as an important way to document discoveries, identify new avenues for research, and justify a project budget.

Faculty advisors and graduate students must meet weekly or regularly throughout the project.

There is great motivational value in a weekly, or as frequent as practical, research meeting where each team member reports on recent progress, short-term plans and anticipated roadblocks. Because of the variance in the day to day schedule of a faculty member or student, having a weekly accountability session is essential to maintaining progress for all but the most motivated and organized researchers.

Build projects around shared research goals that motivate faculty and student alike. Establish a work environment to support their individual needs, and trust them to get the job done.

Academic research often is subject to the whims of grant funding organizations, yet it is quite possible for funded students to conduct research that supports the individual goals of a dissertation while supporting the larger goals of a project. Identifying projects that are interesting to the individuals involved can be the best way to maintain motivation over the course of a multi-year project.

The most efficient and effective method of conveying information to and within a research team is face-to-face conversation, but email, instant-messaging, wikis and blogs are essential media, as well.

The weekly or regularly scheduled, face-to-face research meeting is essential, but other forms of time-shifting communication can be equally effective at maintaining research progress, trading discoveries, answering questions and maintaining a solid collaboration.

Published papers, technical reports, literature surveys and working research software are the primary measures of progress.

If it isn't published, it didn't happen. The ultimate goal of academic research is almost always the publication of research results in a journal, conference proceedings or other outlet.

Agile research processes adapt to the highly variable nature of the academic schedule, recognizing that the pace will vary dramatically over the course of a project.

The academic year has its own pattern, which differs greatly from that of an industry software development schedule. From semester to semester, season to season, and even monthly, weekly and daily, the schedule in academia varies greatly. By acknowledging this unpredictability, unrealistic expectations for meetings and results can be avoided.

Continuous attention to proper citation, short- and long-range planning, and maintaining forward momentum enhances research productivity.

Keeping good notes as data is gathered is critical, as are identifying tasks to be undertaken in the short-term and determining the long-range goals for a research project. It is never too early to try to project a year or two ahead, and craft a plan backwards from "release dates" (i.e., submission deadline, thesis defense, graduation, etc.).

Simplicity is a worthwhile goal, yet recognizing when complex problems often require complicated solutions is essential.

Unnecessary simplification can be the bane of research progress at times. Research often involves tackling unsolved problems with unknown solutions, and the process of conducting such research can involve great complexity which should not be eschewed in favor of false simplicity.

The best research emerges from self-motivated, highly-organized teams of one or more, yet chaos also can be an ally to discovery.

Years and years of hard effort, detailed data gathering, and meticulous note taking sometimes yields uninteresting results. Yet, after all of those years, usually a moment of clarity arises that cannot be planned, organized, or devised. This productive chaos is captured in the famous quote from Isaac Asimov, who said, "The most exciting phrase to hear in science, the one that heralds new discoveries, is not 'Eureka!' but 'That's funny.'"

At regular intervals, individuals reflect on the effectiveness of their contribution to the team and adjust their behavior accordingly.

Members of a research team in an academic setting are not always following a shared schedule. Therefore, frequently it is up to each individual team member to self-analyze and reflect on the effectiveness of his or her own efforts in relation to the research goals of the team. Upon identifying an area of improvement, the individual's needs should be communicated in the hopes that others can adapt, although the individual must also be willing to adjust personal behavior to better support overall team goals.

4.3 Summary

Clearly, there are many similarities between the needs of industry and research projects. Yet, it is the inherent unpredictability of the research process and the academic schedule that normally drives it, that necessitates a modified approach from mainstream agile methods. By drafting these value statements and principles, it is hoped that they can be used to guide an academic research project, particularly ones that involve the production of software, to benefit the organization of research teams and to begin a discussion on how to adapt agile methodologies to best serve the needs of research projects in general.

5 Conclusion

Agile development methods such as Scrum, Extreme Programming and others are widespread among software development projects in industry. They share advantages such as high customer satisfaction, reduced risk and predictability of progress. The typical academic research project does not fit well into the constraints of an agile approach. By examining a variety of case studies on the use of agile methods, and other complementary project management structures, we identify key issues to fitting an agile process to a research project.

The most significant factors that contrast an academic research project that involves software development with an industry software development project are unpredictability of progress and variability of schedule. An industry software project has the benefit of a known level of resource availability, and can spend significant, coordinated time organizing those resources to produce regular deliverables to support a timely release of software. In the academic setting, resources are highly variable, so planning and organization must be highly flexible and accommodating, and the end results are often very difficult to predict. The day to day schedule in industry is very predictable, with the expectation that team members are available 40 hours per week with no other encumbrances. The schedule in academia varies widely among semesters, months, weeks and days, making the use of agile methods impractical without modification.

The wide acceptance of the Agile Manifesto and its principles as a set of guidelines for successful software project management is well known. With some modification, we have adapted these guidelines to suit the needs of an academic research project. These guidelines evolved over the course of three years of an industry-academia research and software development collaboration, and the result is presented as our Agile Research Penultimatum. Designed to serve as a set of guidelines for organizing a research project, and a means to further conversation in this area, the four value statements and twelve principles of this Penultimatum are intended to be non-final and open to further modification, debate, revision and refinement.

6 Acknowledgment

This research was supported in part by the Air Force Materiel Command (AFMC), Electronic Systems Group (ESG) under contract number FA8726-05-C-0008. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of USAF, AFMC, ESC, or the U.S. Government. Thanks are due to the faculty, staff and student researchers in the Villanova Center of Excellence in Enterprise Technology.

7 References

[1] K. Beck. Extreme Programming Explained: Embrace Change. 2nd Edition, Addison-Wesley, 2004.

[2] Frederick P. Brooks. No silver bullet - essence and accidents of software engineering. Computer, pages 10-19, April 1987.

[3] S. Chandrasekhar. Agile Software Development: A Case Study. Technical Report, Department of Computing Sciences, Villanova University, 2007.

[4] T. Chowa and D.-B. Cao. “A survey study of critical success factors in agile software projects.” Journal of Systems and Software, 81(6), Pages 961-971, June 2008.

[5] M. Fowler and J. Highsmith. “The Agile Manifesto.” Software Development, 9(8), pages 28-32, August 2001.

[6] J. Highsmith and A. Cockburn, “Agile Software Development: The Business of Innovation,” Computer, pages 120-122, September 2001.

[7] C. Larman and V. Basili. 2003. “Iterative and incremental development: a brief history.” Computer 36(6), pages 47-56, June 2003.

[8] M. Marchesi, K. Mannaro, S. Uras and M. Locci. “Distributed Scrum in Research Project Management.” Lecture Notes in Computer Science, Springer Berlin / Heidelberg, Vol. 4536, pages 240-244, 2007.

[9] A. Murthy. Agile Methods Applied to Research Projects. Technical Report, Department of Computing Sciences, Villanova University, 2007.

[10] M. Poppendieck and T. Poppendieck. 2003. Lean Software Development: An Agile Toolkit. Addison-Wesley, 2003.

[11] M. Poppendieck. “Lean Software Development.” Proceedings of the 29th International Conference on Software Engineering, pages 165-166, May 2007.

[12] I. Sommerville. Software Engineering: 8th Edition, International Computer Science Series, Addison Wesley, June 2006.

[13] Diagram of the Scrum process management method, Wikimedia Commons, accessed March 15, 2009, http://en.wikipedia.org/wiki/File:Scrum_process.svg.

[14] W. A. Wood and W. L. Kleb. “Extreme programming in a research environment.” Lecture Notes in Computer Science, Springer Berlin / Heidelberg, Vol. 2418, pages 89-99, 2002.