Understanding the Tenets of Agile Software Engineering: Lecturing, Exploration and Critical Thinking

Shvetha Soundararajan, Amine Chigani and James D. Arthur
Department of Computer Science
Virginia Tech
Blacksburg, VA 24061, USA
{shvetha, achigani, arthur}@vt.edu

ABSTRACT

The use of agile principles and practices in software development is becoming a powerful force in today’s workplace. In our quest to develop better products, therefore, it is imperative that we strive to learn and understand the application of agile methods, principles and techniques to the software development enterprise. Unfortunately, in many educational institutions, courses and projects that emphasize agile software development are minimal. At best, students have only limited exposure to the agile philosophy, principles and practices at the graduate and undergraduate levels of education. In an effort to address this concern, we offered an advanced graduate-level course entitled “Agile Software Engineering” in the Department of Computer Science at Virginia Tech. The primary objectives of the course were to introduce the values, principles and practices underlying the agile philosophy, and to do so in an atmosphere that encourages debate and critical thinking. The course was designed around three central components: (1) teaching the essentials of how one develops a product within an agile framework, (2) having invited talks by notable industry experts, and (3) having students present and discuss current agile research topics and issues. This paper describes our experiences during the offering of that course, and in particular, the unique perspectives of the class instructor, the teaching assistant and a student who was enrolled in the course.

Categories and Subject Descriptors

D.2 [Software Engineering], K.3.2 [Computer Science Education]: Computer and Information Science Education – computer science education

General Terms

Design, Documentation, Management, Theory

Keywords


1. INTRODUCTION

The software development process systematically begins by identifying, specifying and baselining user requirements and then proceeds to the architecting, design, implementation, testing and maintenance phases. Traditional software development methodologies such as the Waterfall model [1] and the Spiral model [2] have helped reduce the chaotic state of software development by focusing on extensive planning, comprehensive documentation and gathering and specifying user requirements upfront. In the last two decades, however, many development teams have realized that these traditional approaches to software development are not suitable for all contexts [3, 4]. The traditional approaches, which are process-centric, are found to be inadequate and expensive for development efforts that have to address rapidly changing requirements. Government and industry stakeholders are increasingly required to innovate and respond to emerging needs. Hence, requirements are subjected to constant change in the form of refining existing requirements or adding new ones [5, 6]. This phenomenon is called requirements drift [5, 6]. Conventional methodologies attempt to foresee requirements and create a system architecture upfront in order to accommodate current and future needs. However, when previously unidentified requirements surface, the baselined system architecture may no longer accommodate the implementation of those requirements [7]. The cost of modifying the architecture, and in turn the code, in order to accommodate requirements identified late during the development lifecycle is substantial [2].

To accommodate change and address uncertainty, teams involved in software development are embracing the notion of agility [6]. Currently, the number of organizations adopting agile methods is increasing - some of the reasons being (1) the ability to accommodate change throughout the development lifecycle, (2) improved quality, (3) greater return on investment, (4) shorter development periods, (5) improved customer satisfaction, (6) better team morale, (7) reduced waste and (8) better predictability [7-10]. Therefore, it is essential for software engineers to understand the tenets of the agile philosophy.

We, the members of the Department of Computer Science at Virginia Tech, are dedicated to grooming the next generation of Software Engineers. Hence, the department offers courses representing core areas in Software Engineering and current research foci to better prepare our students for success in the Software Engineering arena. With agile software development becoming
more widely used, we realize that instruction and exploration of agile tenets within an academic setting is essential. Hence, we offered an advanced graduate-level course entitled “Agile Software Engineering.”

1.1. Course Objectives

Instruction and exploration are integral components of an effective learning process. We designed our course, therefore, to place equal importance on both, and in particular, while addressing the many facets of agile software development. The principal objectives that guided course development were:

1. To introduce the agile philosophy, values, principles and practices
2. To develop a firm understanding of what constitutes agility, and how and why it differs from conventional Software Engineering
3. To think critically about the benefits and limitations in application of agile principles and practices.

The first objective focuses on the motivation, rationale and fundamentals underlying the agile approach. The second objective emphasizes a more in-depth study of agility and the rudiments of critical thinking within an agile framework. Finally, the third objective stresses exploration and reasoning about the raison d'être of the agile approach to software development.

1.2. Course Design

The advanced graduate-level Agile Software Engineering course was designed for graduate students who already understood conventional software engineering models and practices. To our delight, more than half of the students had been or were currently working in industry. The course was designed around three principal components: (1) the essentials of how one develops a product within an agile framework, (2) in-class presentations by industry experts, and (3) presentations by students to showcase current research topics and issues. The former two components focused more on instruction and the latter on exploration. During all three of the above, discussion, questions and debate were highly encouraged. In addition to the presentations, the students were also assigned research papers and in-class ‘mini-exams.’

Table 1 outlines additional course details, e.g., pre-requisites, enrollment, background knowledge of the students, and other details.

The remainder of this paper is organized as follows. Section 2 provides an overview of experiences from the perspectives of the instructor, the teaching assistant and a student who took the course. Section 3 summarizes students’ responses to prominent issues that emerged during class discussions. To encourage critical thinking and exploration, student responses had to be prepared in the form of research papers. In Section 4, we present the results from the end-semester course evaluation surveys. Finally, Section 5 summarizes our efforts and observations.

2. EXPERIENCES

In this section, we have included the experiences of the course staff (the instructor and the teaching assistant) and a student who participated in the course. These experiences, recorded from three different perspectives, discuss the extent to which the course objectives were achieved and outline the lessons learned.

2.1. Instructor’s Perspective

The overarching objective of the course was to introduce the values, principles and practices underlying the agile philosophy, and to do so in an atmosphere that encourages debate and critical thinking.

The Basics: The goal for the initial portion of the course was to introduce the students to the agile tenets. More specifically, we wanted the student to (a) develop an understanding of what comprised the agile philosophy and software development approach, (b) gain an appreciation as to why it is needed, and (c) evolve a questioning mindset that encouraged critical thinking. The latter was particularly important because we wanted the students to develop their own perceptions as to how agile differed from the more conventional software development models and processes. We began by introducing the Agile Manifesto [11] and discussing the values it espoused. From this point, I provided presentations outlining Scrum [12], eXtreme Programming (XP) [12, 13] and Lean [14]. For each, the underlying principles, practices and activities were introduced. Throughout the presentations, questions were asked, and discussions ensued, that focused on comparisons among the methods, and on their similarities and differences with the more conventional development methodologies. To further immerse the students in the ‘ways of agile,’ each was required to identify, investigate and present an additional aspect of the agile software development approach. It is my opinion that while the presentations provided foundational knowledge, the discussions, debates, and student presentations were the tools that began cultivating their (the students) aptitude for critical thinking.

Invited Speakers: Three speakers from industry were invited to present and discuss their experiences with agile software development. The first speaker was Keith Lang who manages the development of Vanguard’s Investment Product website. Of particular interest to the students was how they (the Vanguard Agile development teams) were able to have four concurrent agile development efforts, all of which were synchronized to produce a single product. Mr. Lang recounted his experiences in managing
various software development efforts. The second speaker was Dr. Jason Lee who works for Meridium Corporation, and in particular, has helped integrate their agile practices with those practices fundamental to usability engineering [15]. A substantial amount of discussion revolved around how two seemingly independent processes could be integrated into a single set of complementary practices. The students were skeptical, but Dr. Lee was convincing. The third invited speaker was Dr. Ahmed Sidky, an Agile Consultant at TenPearls. Having consulted around the world, Dr. Sidky introduced the class to cultural issues and how they can and do impact the Agile Coaching process [16]. His presentation and insights provided a glimpse of agility rarely found in books or papers. Finally, we were fortunate to have Dr. Todd Stevens ‘attend’ many class periods, and provide his comments on the topic ‘de jour.’ Dr. Stevens, a Blacksburg resident, is also a Meridium employee and helps manage their agile development processes. Throughout the semester, he would provide a ‘real world’ perspective on issues and questions, and often challenged the students to think more critically.

From a pedagogical perspective, these people reinforced the theory of Agility through discussions of actual industry practices. The students expressed their appreciation of this fact in the many complimentary comments that followed.

Research Presentations: As a third component of the course, students were required to identify, present and critique published research papers focusing on some critical aspect of Agility. The Research Presentations helped expand the students’ perspectives as to what are the important issues facing the agile community, and forced them (or provided them with the opportunity) to analyze current research efforts, critique the results, and then defend their assessment.

Research Papers: On a final note, throughout the semester, when (selected) questions arose for which there was no apparent answer, the students were asked to compose a 3-4 page research paper that responded to each question. Part of that task was to identify and cite supporting documents that confirmed their reasoning. This, too, was an activity intended to broaden each student’s understanding of the agile philosophy, and to encourage critical thought.

2.2. Teaching Assistant’s Perspective

I am a graduate student researcher in the field of Agile Software Engineering and I worked as the Teaching Assistant (TA) for the advanced graduate-level Agile Software Engineering course. My responsibilities as the TA included (1) instruction, (2) assisting with grading assignments, and (3) facilitating and leading class discussions.

The course was an excellent opportunity for me to hone my teaching skills. I delivered two lectures to (1) introduce the agile philosophy, its values and principles, and (2) provide an overview of some of the agile methods endorsed by the agile community. Additionally, I facilitated class discussions and offered insights and opinions about topics including, but not limited to, agile methods, current research trends and issues faced by the agile community.

I am a proponent of the agile philosophy. When I started my research in the field of agile software development nearly five years ago, I embraced the agile philosophy almost immediately. I expected the same from my fellow graduate students enrolled in the course. I was surprised to know that most of them were reluctant to accept the notion of agility. Though more than half of the number of students enrolled in the course had prior Software Engineering industry experience, only two of them had worked for organizations that had transitioned to an agile environment. This contributed significantly to their skepticism. For example, the students found the concept of minimal documentation hard to believe and accept. The class discussions at the beginning of the semester were almost confrontational. More than once, the whole class period was consumed by the discussions that would leave no time for the instructor to proceed with his topic for the day. As the semester progressed, however, the students began to appreciate and embrace the agile philosophy. I could observe that the discussions became more complimentary. The transition was reflected not only in the class discussions but also in their answers to questions in exams and assignments.

Dr. Ahmed Sidky was invited to present his experiences as an agile coach. During his presentation, he mentioned that the people of an organization ready to transition to agile are usually skeptical about the concept of agility to start with, and embrace the idea gradually. It was surprising to see the same trend in an academic setting.

My doctoral research involves assessing the collective ‘goodness’ of agile methods adopted by organizations [17]. We have developed an underlying structured framework to guide the assessment process. The design and construction of the framework revolves around the identification of agile objectives, principles that govern the achievement of those objectives, and practices that help implement those principles. Our discussions in class regarding Capability Maturity Model Integration (CMMI), the difficulties in applying it in an agile environment, the advantages and disadvantages of Capability Maturity Model (CMM) and CMMI with respect to agility, provided excellent insights and helped refine the design of the framework mentioned above. I presented my research to the class and received constructive feedback.

At Virginia Tech, students are required to evaluate their course instructors and teaching assistants at the end of each semester. An analysis of the student responses revealed that they found my insights valuable. What gives me greater pleasure than positive student evaluation results is that the students now embrace agility and have begun applying the agile philosophy in their work.

With the number of organizations transitioning to agile increasing, there is a need for agile education in colleges and universities. Courses such as these expose students to the concept of agility and prepare them to adopt and apply the agile philosophy in industry settings.

2.3. A Student’s Perspective

The primary goal of taking the course was to learn the methods agile practitioners prescribe for developing software systems. As a doctoral student conducting research in architectures for network-centric systems of systems, which require process-oriented, plan-driven approaches in their development, I came into the course with two presumptions. First, I was doubtful about the ability of agile methods to scale to accommodate the needs of large-scale
systems. Second, I was unsure how *agilists* claim to develop quality software without architecting for these qualities early on in the life cycle.

The experience from the course was both challenging and rewarding at the same time. The main challenge I faced consisted of my inability to reconcile my training and experience in conventional software engineering with the practices of agile methods. More specifically, one of the issues raised in many discussions in class was that of scalability. Being aware of the complexity of large-scale software development efforts, which is often characterized by multi-team, multi-system, and multi-year development aspects, it was problematic for me to see the applicability of agile practices such as small teams, non-emphasis on documentation and short life cycles in a large development effort.

In spite of this challenge, the experience in the course was stimulating in many ways. The course proved to be an excellent opportunity to widen my understanding and increase my appreciation of agile methods.

First, assignments were structured to engage our analytical thinking to explore the opportunities and the limitations of agile methods. They focused on wide areas of agile for which we prepared a short paper and a presentation to discuss our findings. This component of the course allowed me to reconcile my architecture experience with some of the agile practices and incorporate these new findings into my own work. This component of the course also encouraged me to write a position paper about agile and architecture that I presented in the Software Architecture Technology UserR Network (SATURN) Conference 2010 [18].

The second reason why this course was a successful experience is the fact that it changed my perspective about agile. In fact, I currently apply basic agile values and principles in conducting my own research work. Participating in this seminar made me understand that agile is a way of thinking rather than it is a set of practices. As a result, I have since become engaged within the architecture community in discussions about how we can bring the two domains together. Specifically, I advocate the use of four agile practices into architecture development. These practices include:

- User stories, which describe the functional capabilities that the system architecture must provide and which can be mapped to quality scenarios;
- Quality scenarios, which describe one or more quality characteristics of the system (e.g., maintainability, interoperability and usability);
- The “architecture wall” concept, which is literally the wall used to post and organize index cards and post-it notes describing the architecturally-relevant capabilities and their corresponding architectural blueprints; and
- Architectural refactoring, which refers to the way index cards and post-it notes are shifted around in a structured manner to ensure the integrity of the existing relationships among the user stories, among quality scenarios and between stories and quality scenarios.

Finally, my experience in this course gave me the chance to realize that, although practitioners from both sides of the fence (conventional and agile) see software development form different prisms, the two approaches can be complimentary. This is evidenced by the effect this course had on my own work, and by the emergence of several initiatives to reconcile between extreme interpretations of agile philosophy and traditional software development approaches.

The potential of agile methods is promising, as they have shown value and wide adoption in practice. Therefore, incorporating these methods in the computing curriculum, by offering courses such as this one, is fundamental to the success of next generation software engineering practitioners and researchers.

### 3. PROMOTING CRITICAL AND CREATIVE THINKING

As mentioned previously, class discussions formed an integral part of the course throughout the semester. The students were encouraged to critique agile methods, analyze their current state, discuss research issues and suggest solutions. During these discussions, some issues were brought to the forefront time and time again. These were important issues that the students knew were addressed in the conventional Software Engineering approaches.

Therefore, the discussions were often about “How are these issues addressed in the agile approach?” In order to urge on the students’ critical thinking, we selected three of the major issues and assigned them as research papers. The students were expected to write a short paper (3-4 page) addressing the issue. Critical thinking also enabled them to be creative in their solution approaches. In the following paragraphs, we discuss the assigned questions, the underlying issues, the students’ thought process in outlining solutions and our observations.

Question 1: Government and contracting organizations expect agile organizations to meet the same Capability Maturity Model Integration (CMMI) [19] measurement standards as organizations employing a plan-driven development process. If we wanted a CMMI-like approach to measuring the capability of an agile organization [20], what Key Process Areas (KPAs) and measurement indicators would you define?

**Issue:** CMMI focuses on assessing an organization’s process and process artifacts. On the other hand, agile Software Development places emphasis on assessing the working software or product.

**Students’ thought process:** The students understood the essence of CMMI. They performed a comparative analysis of CMMI and the agile philosophy. Ways to strike a balance between the two were described.

**Observations:** We identified three approaches that the students suggested in order to nurture a symbiotic relationship between the agile philosophy and CMMI. They are outlined below:

- Identify KPAs from CMMI that are relevant to the agile environment, e.g., the Project Planning KPA. Modify the assessment criteria and the expected artifacts such that agility is not compromised.
- Attempting to force-fit agile to CMMI, or watering down CMMI to suit an agile environment, is not an effective solution. Develop a completely new set of KPAs that are modeled along the lines of the KPAs in CMMI, but that reflect the agile approach to Software Engineering.
In analyzing the advantages and disadvantages of both CMMI and agile methods, it is imperative to develop a hybrid approach that provides the best of both worlds. More specifically, we should develop new KPAs and also modify existing KPAs that are suitable to an agile environment.

**Question 2:** How do we scale agile methods to fit larger scale systems development? Or more specifically, what are some of the issues that would have to be addressed to achieve scalability?

**Issue:** Agile methods were intended for small-scale systems.

**Students’ thought process:** For larger scale systems, the students had a mental model of a conventional Software Engineering approach like the Waterfall model. They compared the agile approach to software engineering to the conventional approach and identified areas that they thought were crucial to scalability.

**Observations:** The students most often identified the need for comprehensive documentation, team size and system complexity as being at odds with existing agile practices. They also suggested ways to scale the identified factors.

**Question 3:** To what extent do agile methods address the maintenance activity?

**Issue:** Very little information is revealed concerning to what extent the agile Methodologies address the post-development maintenance activity.

**Students’ thought process:** In conventional software engineering approaches, maintenance is portrayed as a downstream development activity. Some of the students perceived maintenance in this conventional sense. Others envisioned maintenance as a part of the agile development process itself.

**Observations:** The students focused primarily on the two approaches given below:

- Maintenance is integrated within the agile software development process. Certain artifacts developed during the process assists with maintenance within agile software development.

- Maintenance should be considered as a downstream development activity. After the product is developed, maintenance on that product should be supported.

### 4. ASSESSMENT

Students at Virginia Tech are requested to complete end semester course evaluation surveys to evaluate the courses, the course instructors and the teaching assistants. These surveys help course staff gauge the extent to which course objectives are met. In this section, we summarize the results for some of the questions from the surveys.

90% of the students enrolled in the course completed the surveys. The survey questions primarily involve providing descriptive answers or Likert scale [21] measures.

We introduced the agile philosophy, the values and principles stated in the manifesto, and provided an overview of some of the agile methods endorsed by the agile community. These introduc-

![Figure 1. Summary of survey results for questions pertaining to course gains.](image)

More than 70% of the students who completed the surveys had found the degree to which the subject matter was made stimulating or relevant to be “excellent.”

Most of the respondents (71%) recognized the values of the assignments to be “excellent”.

In summary, we received significantly positive feedback from the students. Based on the survey responses and the students’ feedback, we are of the opinion that we have largely achieved what we
set out to do (with respect to the course objectives) when we designed the course.

5. CONCLUSIONS
The intent of the course was to provide an understanding of Agile Software Engineering by studying the basics, questioning the principles and practices, and formulating reasoned opinions through critical thinking. We designed the course to include research presentations, talks by industry experts, class discussions and research paper assignments. We could observe the evolution of critical thought as the semester progressed.

The students were required to complete the end-semester student evaluation surveys. Nine out of ten students in the class completed the surveys. They all indicated that the course objectives were met. They also found the class discussions and research papers to have been the most valuable components of the course.

The agile philosophy has emerged from the experiences of several software engineering practitioners, and hence, its adoption has been predominantly in industry settings. Research in academia involving agile software development has been minimal. Therefore, students are receiving only minimal exposure to agile methods in the classroom environment. We conjecture that courses such as the Agile Software Engineering class we offered can bridge the gap between industry and academia, and prepare budding software engineers to adopt and effectively apply agile concepts when they enter the software and systems development industry.

REFERENCES