Journal of Information Technology Education: Innovations in Practice **Volume 11, 2012**

Accelerating Software Development through Agile Practices - A Case Study of a Small-scale, Time-intensive Web Development Project at a College-level IT Competition

Xuesong (Sonya) Zhang
California State Polytechnic University, Pomona, CA, USA
xszhang@csupomona.edu

Bradley Dorn
California State University, Fresno, CA, USA
bdorn@csufresno.edu

Executive Summary

Agile development has received increasing interest both in industry and academia due to its benefits in developing software quickly, meeting customer needs, and keeping pace with the rapidly changing requirements. However, agile practices and scrum in particular have been mainly tested in mid- to large-size projects. In this paper, we present findings from a case study of agile practices in a small-scale, time-intensive web development project at a college-level IT competition. Based on the observation of the development process, the interview of the project team, and the study of relevant documents, we describe how agile practices, such as daily scrums, backlogs, and sprints, were successfully adopted to the project development. We also describe several supporting activities that the team employed, including cross-leveling of knowledge, socialization, and multiple communication modes. Finally, we discuss the benefits and challenges of implementing agile practices in the case project reported, as well as contribution and limitation of our findings.

Keywords: Agile, Scrum, Software development, Project management, Web application.

Introduction

Created to be a lightweight software development method by 17 software developers at a ski resort a decade ago ("Agile Software Development," 2011), agile development has received increasing interest both in industry and academia due to its benefits in developing software more quickly and at lower costs, meeting customer needs, and keeping pace with the rapidly changing requirements. Agile development aims for customer satisfaction through early and continuous delivery of useful software components developed by an interactive process with the design point that uses minimum requirements. Using agile methods helps refine feasibility and supports the process for getting rapid feedback as functionality is introduced. Developers can adjust as they better clarify unclear requirements. The rate of change in the business world has accelerated dramatically over the past decade. In order to remain competitive, companies developing software need a process that can help them to be more efficient and effective.

Traditional software development methodology such as waterfall is inflexible, expensive, and requires extensive plan-

Editor: Keith Willoughby

ning and rigid adherence to the sequentially based steps in the process (Boehm, 2002). In contrast, agile methods address the challenge of unpredictable and ever-changing needs and requirements in software development and deliver value to users as soon as possible (Nerur & Balijepally, 2007). Table 1 summarizes these differences.

Table 1. Traditional and Agile Perspectives on Software Development
(Nerur & Balijepally, 2007)

	Traditional view	Agile perspective
Design Process	Deliberate and formal, linear sequence of steps, separate formulation and implementation, rule-driven	Emergent, iterative and exploratory, knowing and action inseparable, beyond formal rules
Goal	Optimization	Adaptation, flexibility, responsiveness
Problem-solving process	Selection of the best means to accomplish a given end through well-planned, formalized activities	Learning through experimentation and introspection, constantly reframing the problem and its solution
View of the environment	Stable, predictable	Turbulent, difficult to predict
Type of learning	Single-loop/adaptive	Double-loop/generative
Key characteristics	Control and direction Avoids conflict Formalizes innovation Manager is controller Design precedes implementation	Collaboration and communication; integrates different worldviews Embraces conflict and dialectics Encourages exploration and creativity; opportunistic Manager is facilitator Design and implementation are inseparable and evolve iteratively
Rationality	Technical/functional	Substantial
Theoretical and/or philosophical roots	Logical positivism, scientific method	Action learning, John Dewey's pragmatism, phenomenology

Primary agile practices include:

- Rapid application development (RAD) methodology – RAD methodology emphasizes extensive user involvement in the rapid and evolutionary construction of working prototypes of a system to accelerate the systems development process.
- Extreme Programming (XP) methodology – XP methodology breaks down a project into small phases, and developers cannot continue on to the next phase until the first phase is complete.
- Rational unified process (RUP) methodology – owned by IBM, RUP methodology provides a framework for breaking down the development of software into four gates: Inception, Elaboration, Construction, and Transition. Each gate consists of executable interactions of the software in development. A project stays in a gate until the stakeholders are satisfied, and then moves to the next.

- Scrum methodology - Scrum methodology uses small, self-organizing teams to produce small pieces of deliverable software using sprints (usually 30-day intervals) to achieve an appointed goal, starting with planning and ending with a review. Features to be implemented in the system are registered in a backlog. Then, the product owner decides which backlog items should be developed in the following sprint. Team members coordinate their work in a daily stand-up meeting. One team member, the scrum master, is in charge of solving problems that stop the team from working effectively (Schwaber & Beedle, 2001).

Research Method: Case Study

Case study research is appropriate to investigate a phenomenon in its real-life context and to answer how and why questions (Yin, 2002). Case study method is widely used in Software Engineering and Information Systems (IS) research since it studies contemporary phenomena in its natural context (Runeson & Höst, 2009). This case study examines the value of agile practices in a group-based, small-scale, time-intensive web development project through discovering the following:

1. The particular agile practices adopted by the project team in a small-scale, time-intensive web development project, and the manner in which they were used;
2. Additional supporting activities;
3. Benefits and challenges.

Data were collected using a qualitative, interpretive approach, including observation, interview, and documentation. Team meetings and project development life cycle were observed. Meeting notes, Skype (daily scrum) messages, and project documentation were reviewed and analyzed. Informal interviews were conducted to collect related activities or tasks and further clarify issues.

IT Competition Project Case and Rules

The Information Technology Competition (ITC) hosted by the Management Information Systems Student Association (MISSA) at California State Polytechnic University, Pomona is an annual event where students from California universities and colleges gather to demonstrate and compete their expertise in the Information Technology field. Teams of 3-5 students are given a real-life business project created by a company in the business industry to work on within a two-week time period. Their work is then analyzed, judged, and critiqued by industry professionals. There are five case categories in the competition:

- Business Systems Analysis
- Computer Forensics
- Computer Programming
- Web Applications Development
- Telecommunications

This case study focuses on a web project in the Web Application Development category. Creating the next generation of high-impact, effective web systems requires technical expertise combined with artistic talent. This event encompasses elements of visual design, functionality, usability, creativity, and engineering. This category requires a combined knowledge and skills of both web design and application development. The project assignment is to develop a community oriented web site that allows registered users to submit source code for review by other registered users. The high-level system features and workflow for the web site is illustrated in Figure 1.

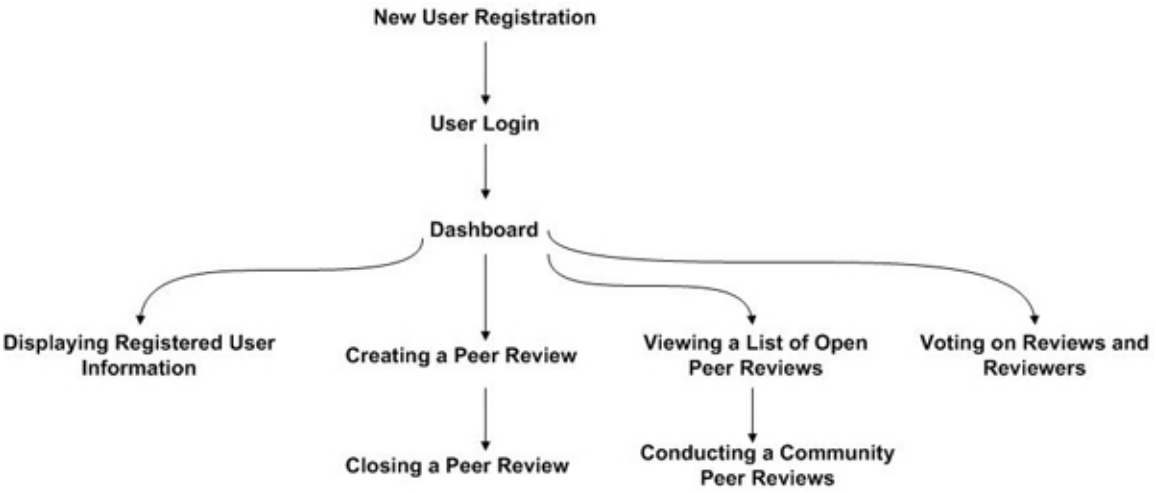

Figure 1. High-level System Features and Workflow

Forming and Structuring Project Team

Team composition in an agile project is usually self-organizing and cross-functional. The project team is self-organizing in that the team leader does not decide which person will do which task or how a problem will be solved; the team as a whole makes such decision instead. The team is cross-functional so that everyone necessary to take a task from idea to implementation is involved. The self-organizing and cross-functional characteristic of agile teams also makes the teams capable of great speed and agility and especially good at socialization and communication. The team is usually supported by two specific individuals or roles: a scrum master and a product owner. The scrum master can be thought of as a project manager for the team, maintaining the processes and helping team members use the scrum framework to perform at their highest level. The product owner represents the business and stakeholders, such as customers or users, and guides the team toward building the right product. In this case study, four students from a California State University majoring in Information Systems volunteered, formed the project team, and represented the university in the competition.

Traditional software development methodologies usually have very specific, laid out development plans. Because of the nature of the team and limited project time frame in this case study, it was not practical for the project team to spend a great deal of time on project management details in advance, such as which team member would develop which aspect of the project; instead, the project team decided to adopt a more flexible approach - agile practices. Scrum, in particular, was used because of its proficiency in developing in small, self-organizing, and cross-functional groups, as well as its best practices for rapid delivery of high-quality software. The formed ITC team self-structured and defined new roles for each member, including a Team Leader, a Database Administrator, a Web Design Specialist, and an Implementation Lead. While these roles are assigned based upon each team member's assessed strength and expertise, it does not translate to exclusive responsibilities. For example, the team leader also participated in web design and coding, and the database admin also participated in coding and testing, and both contributed to the interface design and documenting. Also, decision-making and task assignment were made via mutual agreement among the team members. The team leader acted as a scrum master, and the competition's case document writer (an industry professional) served as the product owner.

Implementing Effective Scrum Practices

Daily scrum (via Skype)

Daily scrum meetings are typically held at the same time everyday. During a daily scrum meeting, each team member answered the following questions:

- What have you done today?
- What are you planning to do tomorrow?
- Do you have any problems preventing you from accomplishing your goal?

By focusing on what each member accomplished that day and will accomplish the next day the team gains an excellent understanding of what work has been done, what work remains, and makes commitments to each other. This makes it possible for all members of the team to efficiently gain a perspective of how the system works and to quickly adapt to developing in an area where another team member left off. Traditional methods usually do this using extensive documentation, planning, and rigid work distribution.

The team chose Skype as the daily scrum media because it allowed team members to meet online regardless of their physical location, to easily collaborate via instant messaging, voice, and video conference calls, and to share documents via file transfer - all free of charge. Every day at a specific time (usually 10:30pm) the team met online via Skype for a short period of time (15 - 30 min) and discussed the project status (Figure 2).

Team members usually stayed online after the daily scrum to extend discussion or solve problems together. The team also used Email to clarify questions and answers before and after the daily scrum meeting. In addition, the team held face-to-face meetings once every few days to do in-depth analysis and discussion.

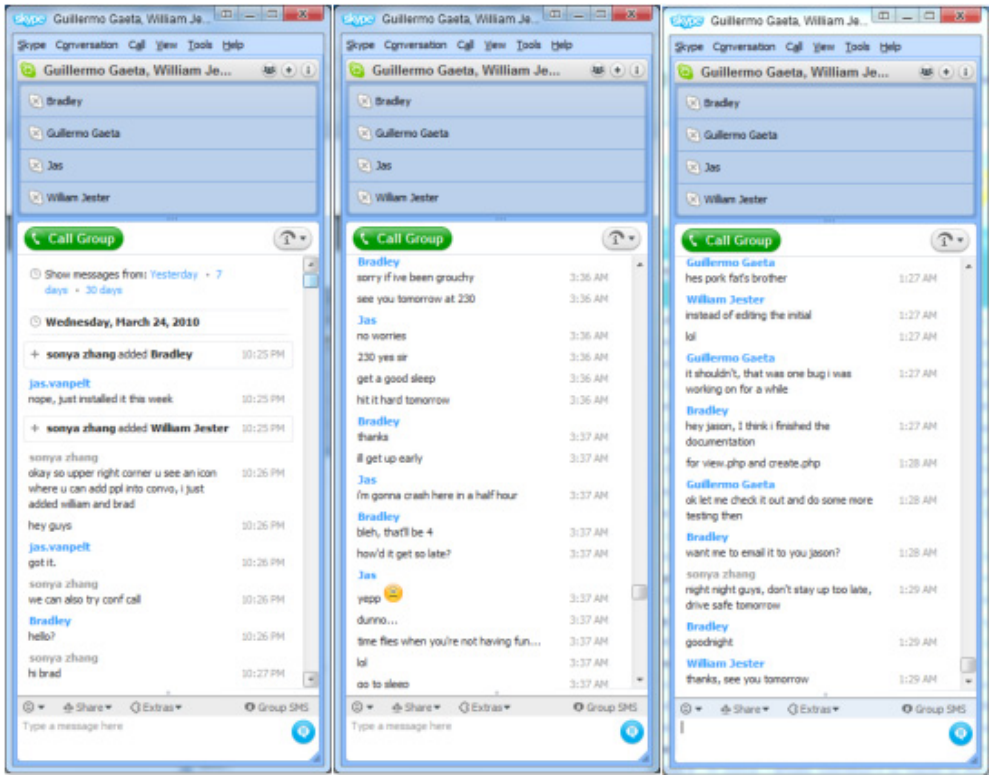

Figure 2. Daily Scrum via Skype

Backlogs

The scrum method uses two types of backlogs to keep a record of the list of work throughout the entire project life cycle. The Product backlog is a high-level, master list of all functionality desired in the working product. Creating a product backlog usually involves the process of transforming user centric specs to technical tasks. The product backlog is allowed to grow and change as more is learned about the product and its customers. In this case study, while the content of the Product backlog and business value of each listed item were derived from the project case requirement document provided by the ITC organizer (written by the product owner) and maintained and updated by the team leader, the associated development effort is set by the team as a whole.

The Sprint Backlogs on the other hand prioritize and expand each Product backlog item into one or more detailed tasks that the team can effectively share and commit to completing within the sprint iteration. In this case study, the project team leader (also the scrum master) maintained the backlog and updated it to reflect the task status and progress (e.g., complete, testing, or implemented). The team added or removed tasks during the iteration when necessary.

Sprints

A sprint is a basic unit in the scrum development methodology and other agile development methodologies. Sprints usually last between one week to one month and are restricted to a specific duration of a constant length. There are usually two types of sprint meetings. Sprint planning

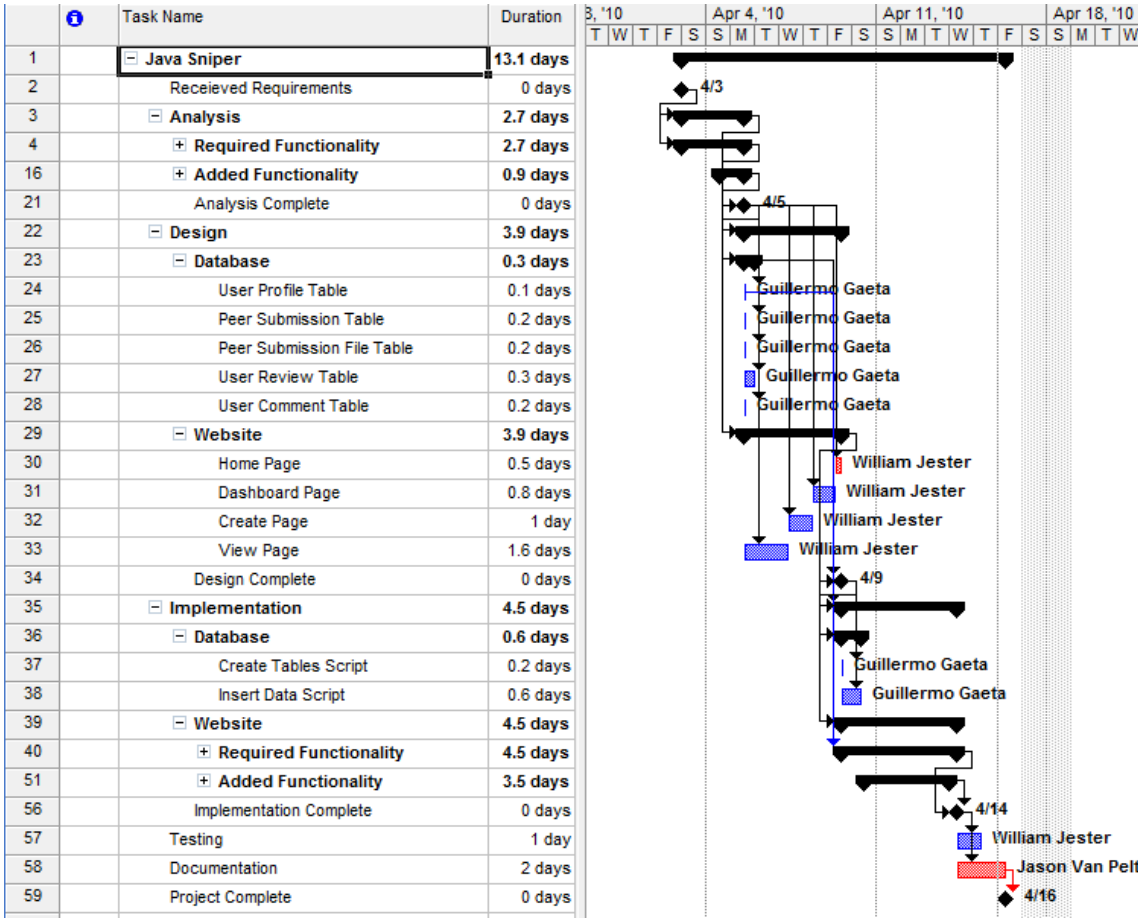

Figure 3. Project Management in a Microsoft Project Gantt chart

meetings are usually held at the beginning of each sprint to decide desired outcome (a commitment to set of features to be developed) in the iteration. A sprint review meeting is usually held after each sprint, when newly developed functionality is presented/demonstrated and reviewed. Modifications are noted and added to the future Sprint backlog.

In this case study, due to the 2-week time constraint for the entire project, each sprint had duration of only 3-5 days. Both sprint planning and review meetings are held via face-to-face meetings. The team also tried to clarify as many issues as possible via Email before the meeting to keep it short and concise.

The team used Microsoft Project as the project management tool to maintain the sprint backlogs and to facilitate sprint planning, monitoring, and reviewing. A common project management tool provided by Microsoft Project is the Gantt chart. A Gantt chart is a simple bar chart that depicts project tasks against a calendar. On a Gantt chart, tasks are listed vertically and the project's time frame is listed horizontally. A Gantt chart works well for representing the project schedule. It also shows actual progress of tasks against the planned duration. Figure 3 shows the project management using a Gantt chart.

As shown in the Gantt chart in Figure 3, the team divided the project into 3 sprints:

Analysis Sprint - In the analysis sprint, the team applied Use case (user centric specs in product backlog) to describe system behavior from an actor's point of view as scenario-driven threads through the functional requirements. Figure 4 shows an example of the Use case for the scenario where a user comments on a post.

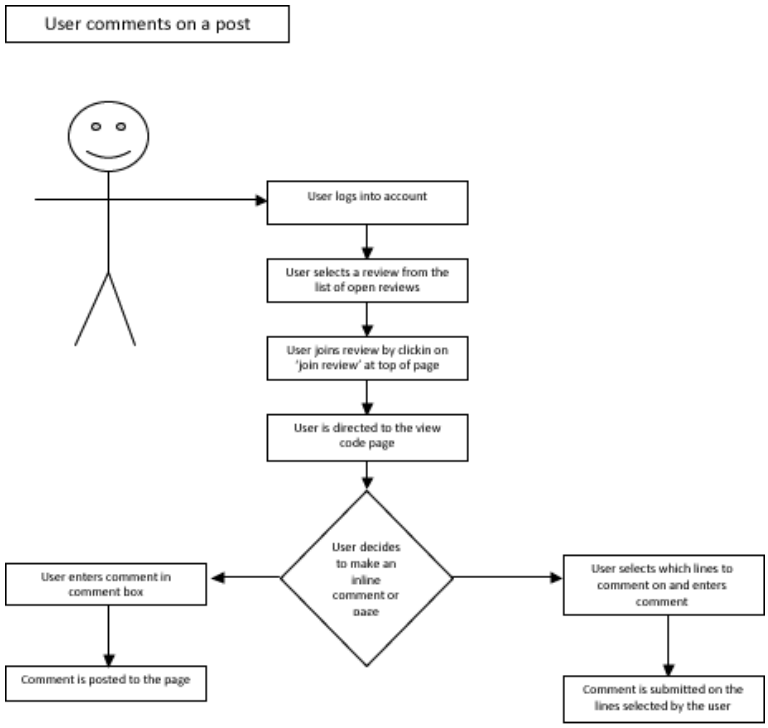

Figure 4. Use Case for a User Comments on a Post

Design Sprint - The design sprint includes database design and web interface design. The database design (technical tasks in product backlog) in this sprint is mainly a conceptual design. The process includes creating the data requirement document (i.e., describe what data items will be stored in the database and how the various data items relate to one another), analysis (i.e., define detailed attributes of the data and constraints), and constructing ERD (conceptual data model) and

Data dictionary (logical schema). After the database was structured, the web interface was prototyped and designed to coordinate the functionalities and tasks specified in the sprint backlog, also to be understandable, intuitive, and simple to users. Figure 5 shows the Entity-Relationship Diagram for the project.

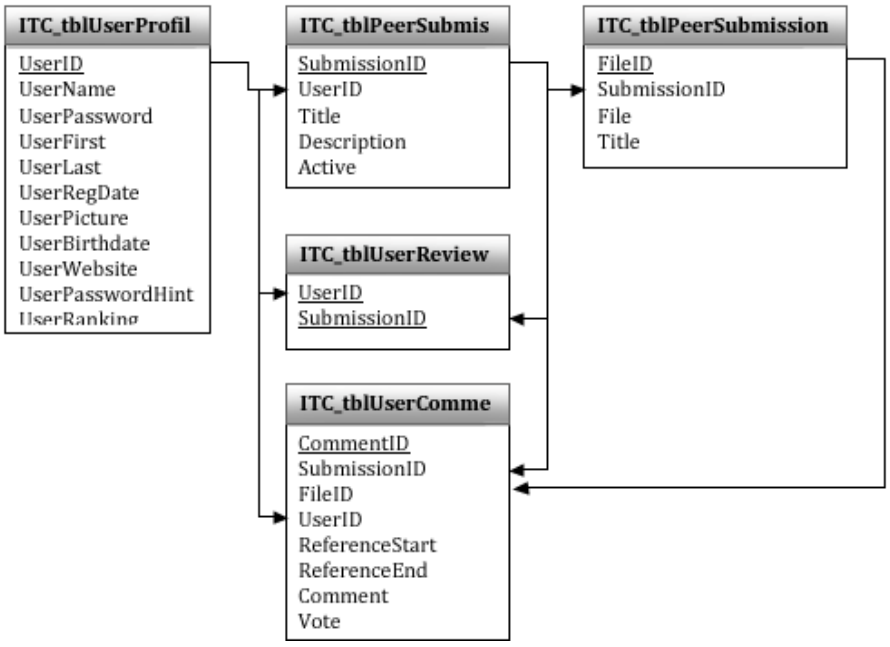

Figure 5. Entity-Relationship Diagram

Implementation Sprint - The implementation sprint includes database implementation and functionality implementation. The database implementation was accomplished using SQL statements. The process includes implementing the database (e.g., create database, set users and privileges, create tables, set constraints, etc.), populating each table with specific data (e.g., configuration and testing data), and optimizing the database structure and performance (e.g., normalize, index, cache, monitor). After the database was implemented and populated with data, functionalities shown in Figure 1 (High-level System Features and Workflow) were coded, tested, and implemented.

In this case study, the software product (called "Java Sniper" because it provides community-based review service for source codes written in Java) was developed using an open source LAMP (Linux, Apache, MySQL, and PHP) environment. The base user interface, dynamic page contents, and reusable functions of Java Sniper were created using HTML, CSS, Javacript, AJAX, and PHP with Adobe Dreamweaver CS4 and Adobe Photoshop CS4. The data is stored in a MySQL database and accessed via embedded SQL within PHP pages. The working product is cross-platform – it can be hosted on various operating systems including Linux and Windows, it can also be implemented in various web servers including Apache and Microsoft IIS. Users can access the web site via various web browsers, including IE, Firefox, and Safari. Figure 6 shows the system architecture.

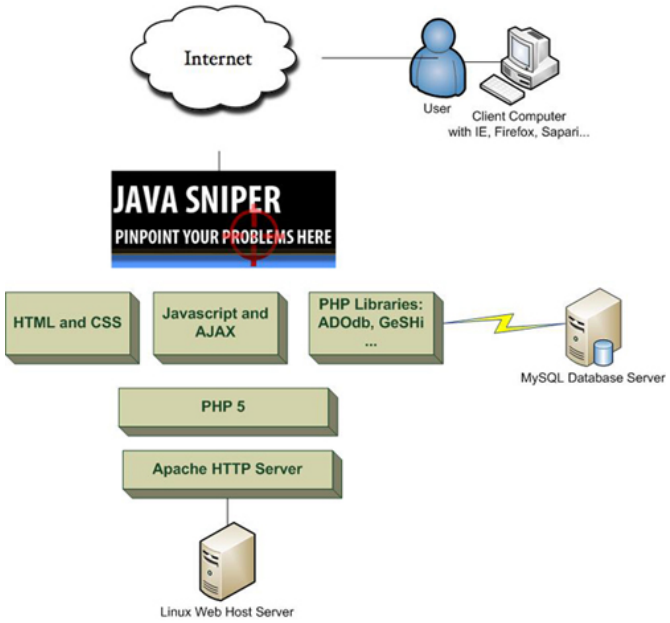

Figure 6. System Architecture

Supporting Activities

In addition to the common scrum practices, the team also employed several supporting activities:

Cross-leveling of knowledge

As opposed to a traditional development team, a scrum team is by nature self-organizing and cross-functional. Although different roles are assigned based upon the team member's strength and expertise, each member acts as a middle manager to bridge the difference between the theory (what ought to be) and the reality (what can be done) to each other. They translate theory into practical requirements, which are then tested in the reality. Any contradiction is communicated and resolved. In this case study, team members shared knowledge and skills in web design and development, database design and management, and other relevant areas to help each other accomplish the tasks.

Socialization

The effects of socialization depend upon the team members' interpersonal skills, trust, and interaction with each other. In this case study, all team members came from the same class, therefore they were somewhat familiar with each other before the project began. However, none of the team members had any sort of social or professional relationship with any of the other team members prior to the start of the course. The team members did not interact with one another specifically, though they were familiar with each other's appearance and course work. These relationships are a good representative for team members that do not face significant cultural or communicative barriers but have not worked together before.

Throughout the project, the team members got to know each other more and became friends. In addition to working on the project together, they also shared common values and interests as college students. The team commented that the establishment of successful and satisfying relationship during socialization helped improving collaboration and productivity.

Multiple communication modes

In agile practices, usually several different kinds of communication are available that can also be applied in parallel, i.e., individual and conference telephone, teleconference, videoconference, email, instant messaging, blog, wiki, and desktop sharing. In this case study, the team utilized multiple communication channels including face-to-face meetings, Email, Skype-supported instant messaging and teleconference, and file transfer.

The team recognized the importance of the fact that each communication avenue provided a valuable way for the team to exchange information. Each communication avenue had its strong points, and weak points. For example, though face-to-face communication can convey many ideas quickly, it is ineffective when describing code, syntax, or sharing files.

Specifically, the team pointed out that face-to-face meetings helped the team to focus on the discussion without other disruptions and also provided more information in terms of body language, subtle personality and demeanor, and lively exchange of ideas or conversation; while online communications such as Skype meetings provided more flexibility including meeting late night or geographically separated, looking up information online, or sharing important files and documents while chatting. Email and chatting was the least invasive and most flexible, but was usually not as effective as other methods. Therefore, both communication channels facilitated teamwork in various aspects.

Discussion

The project team successfully delivered a working software product (i.e., Java Sniper) at the end of the 2-week time frame. The team demonstrated the final product to the judges (industry professionals) and won second place in the web development category.

The team also compared the efficiency of Java Sniper with other popular code review systems on the market (e.g., Codestriker and Reviewboard) by conducting three different types of testing – installation, code submission, and code review tests. The installation test involved creating a database, installing and configuring scripts, modules, and libraries, and getting the system up and running. The code submission test involved navigating to the submission page, loading the page, filling in the information required, and submitting four files containing Java source code designed to output the text "Hello World!" for review. The code review test involved accessing the submitted files, reading through it (the files were all very small and reading time was almost negligible), and writing and submitting the review (time for writing review was recorded separately because writing time should be independent of all three systems). The testing results (Table 2) show that Java Sniper is the most efficient of all three systems in terms of installation, code submission, and code review.

Table 2. Efficiency Testing

Code Review Software	Language	Installation Test	Code Submission Test	Code Review Test (writing time excluded)	Allow submitting multiple source files as one project
Codestriker	Python	45 min	12 min 20 sec	5 min 32 sec	No
Reviewboard	Perl	20 min	2 min 58 sec	2 min 4 sec	No
Java Sniper	PHP	15 min	47 sec	35 sec	Yes

During both project presentation and interview, the team affirmed that the agile practices employed were a critical success factor for them to achieve the goal of developing a good quality

software product that satisfied the case requirements within limited time frame; the product would have been completed with fewer features or less quality otherwise. The team stated that the agile practices benefited the project in the following aspects:

Increased quality of the deliverables

Agile technologies feature more frequent delivery of smaller, valuable increments and build quality in rather than add it in at the end of the project. Because of the strict time and scope limit, quality is more easily monitored, managed, and achieved by the end of the iteration. In this case study, team members often reminded each other during daily scrums and sprint meetings to stay on track, prioritize, and focus on the main features specified in the requirement document. As the result, the team was able to deliver a satisfying software product within the limited time frame.

Better change management

A project's requirements will change at some point of time. In scrum, change is embraced and new requirements will be evaluated against existing ones when planning the next sprint. The business and product owner are actively involved in this process, making sure the delivered features are actually useful and valuable to end users and business. Even though in this case study, the project requirements were mostly defined in the case document in advance, during the analysis sprint the team still found some issues that were unclear, such as the expectation of utilizing open source components and services and the weight of additional features that is not explicitly specified in the case document. After clarifying these issues with the case writer the team quickly changed the technical specs accordingly before moving into the design sprint.

More efficient workflow

By using short iterations (i.e., daily scrum and sprints), the team was able to calculate/estimate the time and resources needed and track the development progress for each task in a clearer and more accurate way. As the result, the team was able to be more in control of the project schedule and status and work more efficiently. In this case study, the team used Microsoft Project to plan and manage the time and resources needed for each critical project task and made sure that the tasks were accomplished by the end of the assigned iteration.

Increased innovation

Whenever a new and innovative idea was discovered, the team could quickly share and communicate with the business and production owner and possibly build it straight into the next sprint. In this case study, when the team found out that open source components and services were encouraged in the analysis sprint, they decided to implement an open source tool called GeSHi (Generic Syntax Highlighter) to process each source code file when it was submitted for review and then saved in the database to be displayed on the Source Code Viewing page later in different colors and fonts according to the programming syntax. This feature provides the reviewers with a consistent code formatting and/or syntax highlighting and helps them better recognize problem points in the code. And because the tool was open source, the team saved significant time and effort compared to developing it from scratch.

Like any other agile practitioner, the team in this case study also encountered several challenges. Because the size of the team was fairly small and all the members were college students, the daily scrum or sprint meetings were informal and often involved a lot of socialization. Also because the team members were amateur in software development, they spent a lot of time collaborating and helping each other in cross-functional problem solving. These challenges may be less significant in the teams consisting of professional software engineers/developers; however, other challenges may present, such as recruitment of agile staff, training, motivation, performance evaluation,

communications, adapting agile for distributed teams (different culture, background, language, time zone, etc.) and large-scale enterprise projects (more complicated technical specs and rapidly changing requirements).

Conclusion

This paper presents findings from a case study on agile practices in a small-scale, time-intensive web development project at a college-level IT competition. Based on the observation of the development process, the interview of project team members, and the study of relevant documents, we describe how agile practices, such as daily scrums, backlogs, and sprints, were successfully applied to the project development. We also describe several supporting activities that the team employed, including cross leveling of knowledge, socialization, and multiple communication modes. Finally, we discuss the benefits and challenges of implementing agile practices in the case project reported. Our study found that agile practices were a success in this case study, which confirms that agile methodology is suitable for voluntary, self-organized, cross-functional teams developing small-scale, time-intensive software development projects.

Case studies in software engineering and information systems research often test theories and collect data through observation of a project and other qualitative methods such as interview and documentation. Each team and project characteristics are unique to each case study; thus comparisons and generalizations of case study results are difficult and are subject to questions of external validity (Kitchenham, et al., 2002). The Java Sniper project described in this case study is a reasonable representative of a class of small-scale, time-intensive software development projects in computer science or software engineering courses (with similar number of developers, developers' background and experience, time, and project scope). However, such group projects can still differ in terms of size (software requirements, the number of lines of code), design pattern, type of software developed, language used, etc. It would be interesting to analyze the degree to which agile practices in projects that differ along these dimensions resemble the findings of this case study.

References

Agile Software Development. (2011). *Wikipedia.* Retrieved August 26, 2011, from Wikipedia: http://en.wikipedia.org/wiki/Agile_software_development

Boehm, B. (2002). Get Ready for Agile Methods, with Care. *Journal Computer , 35* (1), 64-69.

Kitchenham, B. A., Pfleeger, S. L., Pickard, L. M., Jones, P. W., Hoaglin, D. C., Emam, K. E., et al. (2002). Preliminary Guidelines for Empirical Research in Software Engineering. *IEEE Transactions on Software Engineering , 28* (8), 721-734.

Nerur, S., & Balijepally, V. (2007, March). Theoretical reflections on agile development methodologies. *Communications of the ACM , 50* (3).

Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Journal of Empirical Software Engineering , 14* (2), 131-164.

Schwaber, K., & Beedle, M. (2001). *Agile Software Development with Scrum.* Prentice Hall.

Yin, R. K. (2002). *Case study research, design and methods.* Sage Publications.

Biographies

Sonya Zhang is an Assistant Professor of Computer Information Systems at The College of Business Administration at Cal Poly Pomona. Sonya received a PhD in Information Systems and Technology from Claremont Graduate University, a Master of Science in Computer Science and a Master of Business Administration from Illinois State University. Sonya was an Assistant Professor at Fresno State prior to joining Cal Poly; she also worked as a software engineer in health informatics and higher education. Sonya's research interests focus on social learning and web intelligence. She has published in leading IS journals and conferences including Journal of Information Technology Education, Journal of Information Systems Education, International Journal of E-Learning, HICSS, AMCIS and ACM SIGMIS.

Bradley Dorn is a MBA student at California State University, Fresno (i.e., Fresno State). Bradley received a Bachelors of Science in Business Information Systems (major) and Mathematics (minor) from Fresno State. Bradley also worked as a teaching associate and student assistant there. Bradley's primary research interests pertain to Internet applications, their design and development, and development process.