System Maintenance in Academia: Using Agile Model Driven Design

Joo Tan
Department of Computer Science
Kutztown University of Pennsylvania
Kutztown, PA

Abstract - This paper presents the experiences of student project teams as they embark on the process of enhancing an existing software system. Prior to this, a system has already been delivered to the client, a local ambulance organization. Two teams are assigned to work independently of each other on this project. The teams follow an agile driven model approach to design and development. This reactive approach is not perfectly suited to use in the academic environment and the problems encountered by the teams are discussed. The collaboration needed within each team during the lifecycle phases is described. Problems seen during the project are examined.

Keywords: Agile Model driven Design, System Maintenance, Enhancement, Collaboration.

Conference: International Conference on Software Engineering Research and Practice (SERP ’10)

1 Introduction

In Fall of 2009, XYZ Ambulance, a local organization requested assistance in creating an automated solution for managing their daily transport schedule. They had been using a white board and marker system as well as post-it notes to do the majority of their scheduling to that point. A student project team was assigned to design and implement an online solution for XYZ Ambulance. In December of 2009, an initial system was delivered to the client. However, this system has existing problems. As a result, it has not been used on a full time basis by XYZ Ambulance.

In Spring of 2010, XYZ Ambulance asked that existing problems be fixed and parts of the existing system be re-designed. Two new student project teams [4] from a software engineering course are assigned to work on this problem – to enhance the functionality of the system as well as fix any existing defects in the system.

The following sections explain the activities in the agile model driven approach that the teams take as they work with the client, student experiences as they collaborate within a team on the project, issues encountered by the teams, and lessons learned from this experience.

2 Current Problem

In January of 2010, a client who had been delivered an automated solution for a scheduling and management problem requested that existing defects be fixed and certain parts of the system be re-designed. The main functionality in this system includes keeping tracking of where ambulances are at a given time, updating pickup and drop off information, managing patient and insurance information, and the ability to add, edit, or cancel transports. This problem can be considered both a maintenance and enhancement issue that provides a great opportunity to introduce students to the problems of group work as well as aspects of software maintenance and enhancement [2]. Two project teams in a software engineering course are designated to work on this problem. The two teams function independently of each other in a competitive fashion, as they work towards resolving the issues echoed by the client. For ease of reference, they are named KATS1 and KATS2. Each team started with five members who are assigned specific project roles within the team – project leader, systems analyst, system designer, system developer, system tester, project manager. A student may assume more than one project role within a team. Note that besides the two KATS teams, two other student teams in the same course are assigned to work on a entirely different project. This paper focuses only on the KATS teams’ work.

2.1 Agile Design and Modeling

The course instructor advocates that the student teams follow an agile model driven approach [1] as they work on the project at hand. An iterative process with short phases (or timeboxes) is characteristic of the agile design and development. This is a reactive approach that allows for quick delivery of software
and accommodates changes quickly, while at the same time encouraging interaction and collaboration within a team. Rapid feedback from end users is another positive aspect of this approach. Since students had little prior experience with team projects, guidance is given to the teams as they attack the issues presented to them.

The software engineering course is 14 weeks long and the project covers the entire 14 weeks of the semester. A high level outline of the complete lifecycle is as follows:

- **Phase Zero (2 weeks)** – Planning, Client Contact, Requirements Envisioning.
- **Phase One (2 weeks)** – Information Gathering, Initial Model Storming and Design, Requirements Prioritization.
- **Phase Three (2 weeks)** – Client Acceptance Test, User’s Manual Documentation, and Deployment.

Note that a system test session in each cycle of phase two serve as major milestones during the project lifecycle. They also serve as a working version of the system at that particular milestone.

### 2.2 The Agile Phases

The approach taken by agile modeling design is to break the software development lifecycle into short iterations, planned with steady development velocity and frequent client feedback [3]. The primary goal of the agile approach is to produce high quality working software quickly that meets the needs of the project stakeholders in an effective manner.

During phase zero, students are assigned to two separate project teams on the first day of the semester. Students must have successfully completed a data structures course before they are allowed to take this software engineering course. The first task of the project teams during the first two weeks is to understand the existing system. They had to make contact with the client and arrange for an initial meeting to establish the overall goal and objectives of their project.

During phase one, the teams had to arrange more meetings with the client to gather exact requirements for the system. These two weeks are spent trying to get the system up and running on their local test environment, prioritizing requirements obtained from the client, doing model storming, and creating an initial design for the intended system. Active client contact is critical at this stage as the teams attempt to capture an early model of the system design. A snapshot of the dispatcher transport tab in the system is shown in Figure 1. The KATS team is able to redesigns screens such as this and gets feedback from the end users as often as possible.

![Dispatcher Transport Details Tab](image)

**Figure 1: Dispatcher Transport Details Tab**

Phase two is divided into two distinct cycles of four weeks each. At the end of each cycle, a major milestone is set at system testing time for a complete working version of the system. Clients and end users of the project are invited to participate in the system testing session. Furthermore, at the end of each cycle, the two teams are expected to give a formal presentation on the status of their project.

The last phase (two weeks) of the project involves preparation for client acceptance testing and system deployment. The teams are expected to coordinate with their client to set up their completed system at the client site. An acceptance criterion that must be approved by the client is established prior to the start of acceptance test. The teams’ systems will be evaluated during acceptance testing and the result will be classified into three possible outcomes – full acceptance, conditional acceptance, rejection. In the event of a conditional acceptance, the team is given one week to fix defects and have their system re-tested by the client.

Another important task required of the teams at this time is the creation of a user’s manual documentation. This is an important component of the agile methodology as enabling the next effort is often considered a secondary goal [1] of this approach. A project may still be considered a failure if the team delivers a working system but it is not robust enough to be extended (maintained) over time.
2.3 Modeling Techniques

The KATS teams use an object-oriented approach while performing agile modeling and design. The Unified Modeling Language (UML) notation is used extensively during this process. Models introduced in class include the event table, use case diagram, domain model class diagram, system sequence diagram, sequence (interaction) diagram, design class diagram, and package diagram. Moreover, user stories are also recommended for modeling purposes. In fact, the teams are encouraged to use one or more of the models listed. They are also advised to use any given model (post-it notes, a sketch on paper or whiteboard) that might help them to iteratively derive a working design quickly. After checking with the client, the features modeled would be then implemented and tested.

An outline of the UML models discussed is as follows:
- **Phase Zero** – None used at this time
- **Phase One** – Event table, use case diagrams, and user stories.
- **Phase Two**, Cycle 1 – Domain model class diagrams (iterative), user stories, system sequence diagrams, design class diagrams (iterative), and package diagrams.
- **Phase Two**, Cycle 2 (4 weeks) – Design class diagrams (iterative), sequence diagrams (iterative), and package diagrams.
- **Phase Three** (2 weeks) – As applicable.

Since students are new to both the agile approach as well as the use of UML analysis and design models, they are not able to create accurate descriptions for them in the early phases. As the project progresses, the models are revised iteratively and the models get more accurate and refined.

2.4 Collaboration

The project teams in the course are required to meet at least once a week outside of class time. This allows the students in a team to discuss models, design, and any outstanding issues each week. The teams are required to write agenda and minutes (details) of their meetings.

To allow for better collaboration, each team is instructed to post their work (models, designs, technical documents, meeting agenda and minutes) on Google Docs [5], a free, Web-based word processor, spreadsheet, presentation, form, and data storage service offered by Google. It allows the teams to create and edit documents online while collaborating [8] in real-time with each other.

In previous years, the instructor had used Basecamp [7] to monitor the level of communication within projects team. Basecamp is a subscription-based collaboration and communication tool for project management. This year, the students are encouraged to use Google Wave [6], an online tool for real-time communication and collaboration. A wave can be described as both a conversation and a document where people can discuss and work together using richly formatted text, photos, videos, maps, and more. Note that Google Wave is currently in beta release. The KATS1 team uses Wave extensively in their daily communication about topics in their project.

3 Issues Encountered

Given that working software is the primary goal of the agile approach and is often considered a measure of progress, one would expect teams to collaborate effectively within a team. Unfortunately, things do not always work out that way. The following sub-sections elaborate on some of the issues encountered during the project lifecycle.

3.1 Database Connectivity

During phase zero of the project, the two KATS teams were given a DVD which was supposed to contain all technical artifacts (documents) that was previously written, a copy of the source code for the existing system as well as a copy of the database. In phase zero of the project, the two KATS teams discovered that the database saved onto the DVD was incomplete. The two KATS teams had difficulty establishing a connection from the user interface of the system to the backend database. Many hours (more than 20) of debugging the problem was reported by both teams. Eventually, KATS1 sought help from the developer of the previous semester’s project team in phase one of the project. This unexpected delay in system setup caused a noticeable amount of delay in kick-starting the project. The teams were only able to successfully establish connection to the database and unit test their systems at the beginning of phase two.

3.2 Effort (or Lack of it)

The KATS teams started slowly in phase zero of the project. They did not meet with the client for the first time until the second week of the project. In phase one of the project, the performance and participation levels of the KATS2 team leader dropped dramatically. He had decided to withdraw from the course but did not inform either his team
members or the course instructor. His decision to drop out of the course was only made known at the end of phase one. This caused a lot of stress on the remaining team members.

At about the same period of time, two other members (analyst and designer) of KATS2 had put very little effort into the team project. Their non-contribution had placed extra burden on the remaining two members (developer and tester) in the team. At the start of cycle 1 in phase two, the KATS2 team was disbanded. The developer joined KATS1 while the tester joined one of the other teams that did not work on the XYZ Ambulance project. The analyst and designer became individuals without a team. According to the course syllabus, individuals with no team cannot earn any credit during the semester until another team is willing to accept them.

3.3 Academia Limitations

Since this is the first attempt by the course instructor to have student teams use the agile approach for modeling, design, and development in an academic environment, problems are expected.

Doing a project in academia is different from working as a full time employee in industry. The incentives are different in industry—job security, incentive for promotion, having 8 or more hours a day to work on the project, the availability of fellow employees to discuss work are characteristics of job requirements in real companies. In academia, a majority (over 90%) of the people enrolled in our software engineering course are full time students. As such, they are required to enroll for a minimum number of credits – 12 hours at our university – to be considered full time. Hence, the amount of time that the students are able to spend on the project is limited and is dependent on the number of courses currently enrolled. Unless a team is truly motivated, anything over 2 hours a day spent working on the project would be a tall order for the students. One would also expect the level of contribution to the project to fluctuate depending on the amount of work is required from the other courses. It is therefore not reasonable to assume a constant level of high quality result every week.

Since the amount of time between an action and the feedback on that action is critical in the agile approach, any delay would have undesirable consequences. Besides the final grade for the course, there is no extra incentive to spend an enormous amount of time on a project. This may lead to a working system at the end but all too often with various limitations such as poor design or low system performance.

4 Lessons Learned

Some students in the KATS team did not put in as much effort as other student on the project. This causes some annoyance to other project members as the burden of responsibility and completion shifts to a selected few students. For future project team courses, it would be important to allocate a portion of the course grade for participation and contribution.

Quite often, the students did not complete work as required in a specific time period. It is important that they deliver models and design as specified, especially when following the agile approach. Any unnecessary delays push the entire schedule back and jeopardize the entire project.

There should be constant communication with the client and end users when using the agile approach. Without frequent feedback from the end users of the system, the models and design might not be accurate. Consequently, implementations would not reflect the expected functionality at the end.

If possible, the choice of project leaders and other members must be considered carefully prior to the start of the semester. The project leader must be technically strong and be motivated to lead the team on the project. The skills and knowledge of other team members should match the requirements of the project as close as possible. An example would be the knowledge of HTML and JavaScript for user interface scripting, whereas knowledge of PHP and MySQL would be needed for backend implementation and connectivity to the database.

5 CONCLUSION

We presented a software engineering course where students work in small project teams to maintain and enhance an existing system that was already delivered to a client. The stakeholders and end users are from a local ambulance organization. They work closely with the students to add new functionality and fix existing defects in the system.

An agile driven design approach is followed by the students during the process as they investigate, model, design and implement the system. Being new to the agile approach, students face some inherent problems that are discussed in this paper. Additional demands required from the students in the course are also described. At the writing of this paper, the students are working in phase two of the project. They are about a week away from meeting the major milestone of system testing in phase two. Progress has not been as rapid as anticipated in the current cycle of phase two of the agile model. We expect better progress in the next cycle as the KATS team
get familiarize with the process. The final results for the project will be reported at a later date.

6 REFERENCES


