SOFTWARE PROJECT DRIVEN ANALYSIS AND DEVELOPMENT OF PROCESS ACTIVITIES SUPPORTING WEB BASED SOFTWARE ENGINEERING TOOLS

Shriram Sankaran, Joseph E. Urban
Department of Computer Science and Engineering, Arizona State University, Tempe, Arizona 85287 USA

Keywords: Software engineering methodologies, Web based, Software engineering tool, Software process

Abstract: The field of software engineering has seen the development of software engineering tools that allow for distributed development of software systems over the web. This paper covers analysis and process activities for a web based software design tool that served as the basis for software requirements formulation of a software process tracking tool. These software tools are an outgrowth of a software engineering project capstone. The discussion focuses on those development activities that assisted the front end of the development through software requirements formulation. This paper describes the background for the software engineering projects, software tool development processes, and the developed software tools.

1 INTRODUCTION

Software applications on the Internet have been proliferating. However, there is a need to provide effective software engineering tool support for the benefit of software engineers, project managers, and end users involved in the development of these web based applications. For the most part, software engineering tools have migrated to the web either as standalone applications or for use in distributed development environments.

On an international level, there are two efforts that have provided major contributions to software engineering methodologies, tools, and processes. The IEEE Standards Activities has developed a comprehensive set of software engineering standards. The Software Engineering Institute's Capability Maturity Model has been beneficial in assisting organizations with providing a means to baseline efforts in key process areas, as well as providing the means for process improvement (Paulk, 1993).

There is an upper level undergraduate software engineering project capstone course sequence that is offered as a set of technical electives in the Department of Computer Science and Engineering at Arizona State University. The Software Engineering Project I and Software Engineering Project II courses are referred to below as SEP1 and SEP2, respectively. Starting in the Spring 1999 semester, these courses have been taught by the second author on a Spring / Fall sequence. These course sequences have focused on the development of web based software tools to support software engineering standards (Ahamed, 2000).

The remainder of this paper includes a description of both the SEP1 and SEP2 software development activities. One particular instance of software engineering standard tool development was selected for inclusion in this paper (IEEE, 1997c, 1997d). These activities lead to software process improvement and software requirements for a process tracking tool, which is described next and then a summary and future research.

2 BACKGROUND

A comprehensive set of software engineering standards has been developed, maintained, and continues to evolve through the IEEE Standards Activities. These software engineering standards have served as the basis for development of a set of web based software engineering tools that was the outcome of a two course capstone sequence in an undergraduate computer science degree program. This paper describes experience with the development and implementation of one of six IEEE Software Engineering Standards that were developed as web based software tools.

The first course, SEP1 focused on software requirements formulation / analysis, specification, and architectural design. The software design and implementation was conducted in SEP2 with testing activities integrated throughout the software development. Experience with two different development methods occurs over the two semester capstone sequence. The concept of a software engineering standards based project in a capstone course or course sequence is a novel approach for applying software engineering concepts in software engineering tools. More typically, research papers show that upper level undergraduate computer science and engineering project courses develop systems in a wide variety of application domains.

During the first semester of the two course sequence, the students follow a waterfall process model, which proceeds from software requirements formulation and analysis though overall architectural design with the detailed design and implementation of one module. A majority of the students only take one semester of the two semester capstone sequence, which facilitates carrying out the development effort in this narrowing of scope manner.

The software engineering standards that were used over the past six years were: ANSI/IEEE Std. 830-1993, IEEE Recommended Practice for Software Requirements Specifications (IEEE, 1997a); ANSI/IEEE Std. 1058.1-1987 (Reaffirmed 1993), IEEE Standard for Software Project Management Plans (IEEE, 1997b); ANSI/IEEE Std. 1016-1987 (Reaffirmed 1993) IEEE Recommended Practice for Software Design Descriptions (IEEE, 1997c) combined with ANSI/IEEE Std. 1016.1-1993, IEEE Guide to Software Design Descriptions (IEEE, 1997d); IEEE Std. 829-1998, IEEE Standard for Software Test Documentation (IEEE, 1998a); IEEE Std. 1219-1998, IEEE Standard for Software Maintenance (IEEE, 1998b); and IEEE Std. 1012-1998, IEEE Standard for Software Verification and Validation (IEEE 1998c) that was developed by the Life Cycle Data Harmonization Working Group of the Software Engineering Standards Committee of the IEEE Computer Society. The first software tool developed through these courses was in support of the IEEE Standard 830-1993 – Software Requirements Specifications (Ahamed, 2000).

In the Spring of 2001, SEP1 began software tool development in order to support the ANSI/IEEE standard 1016-1987 and 1016.1-1993 (IEEE, 1997c, IEEE, 1997d). The developers were formed into six groups of five to six members in each group. Each group selected a different software design method as the basis for tool development to meet the standard.

An evaluation was conducted in SEP2 of the software projects developed in SEP1. An object oriented tool was selected by the developers of SEP2 for complete software development of detailed design through implementation. This tool essentially consisted of four modules: server, graphics, user interface, and database.

3 PROCESS ACTIVITIES

The developers were given the outline of a project that had to be developed in terms of the use cases, design, algorithms, test data, and also code one of the modules. The groups had to choose one programming model and then incorporate the model in the design of the tool.

3.1 Process Activities for SEP1

There were eight major process activities that had an impact on each student. These activities included meetings, tasks, file management, resource allocation, project management, tools, communication, and evaluation.

The groups met twice weekly in class during part of the class hours and then outside class depending upon the need and the schedule of the members. The activities for the upcoming week were charted out in advance by the leader of each group and then divided among the members based on expertise.

The documents and files were made available to every member of the group by using a web-based file management system, such as, the course management tool provided by Blackboard or Yahoo! Groups. The files were available to every member of the group for read-only purposes. Modifications to a file could only be carried out with the consent of the author of the file. In case of major modifications, the version of the file was incremented. Every member of the group had authority to download and upload files. Email messages were sent out to inform the group on each uploaded file.

The meetings were expected to be conducted in a professional manner in the meeting rooms made available to the students by the University. The students had access to computer facilities in the University. Some students made use of computers outside the University campus.

Confidential weekly status reports were required of all members of the groups to be submitted to the course instructor. These reports had provisions to evaluate the status of the project, other members of the group, time sheets and also a self-evaluation. These reports gave the instructor sufficient data to analyze the performance of the groups and take any necessary action. Early on in the course, the groups had to come up with their own risk management plans based on their perceived risks. This task was useful later in the course especially in scenarios where the group had predicted the risk.

The following tools were made available by the instructor for use to the members of the groups: Microsoft Word – used to create documents; Microsoft PowerPoint – used to create slideshow presentations; Yahoo! Groups / University – file management system; Yahoo!/University email – used for intra-group email communications; Rational RequisitePro – used to create templates of specific project documents; and Rational Rose - used to create the class and state diagrams.

The communications among the members of the group were conducted using meetings, phone and email. All email communication was saved for future uses. The grading of the groups was based on published criteria. The final demonstration of the coded module with the test data was conducted in conjunction with the delivery of the project.

3.2 Software Process for SEP2

In SEP2, a unique voting mechanism was used whereby the developers formulated the process and evaluation method for determining which software project to use for completion of the software design tool. Certain criteria were identified to be used in the evaluation of projects and a voting process determined their importance in the evaluation process. After detailed analysis of the initial design of the project produced during SEP1, the volunteer reviewers considered a few changes necessary for the detailed design. These changes were carried out and the new design was incorporated after approval by the whole class.

In order to be able to manage the project efficiently, the project was split into four modules: server, user interface, graphics and database. Each module was to be assigned to a team of students.

The twenty students in the class were divided into four teams based on their expertise and personal preference with six students in the server team, five students in user interface, and four students each in the graphics and the database teams. Each group nominated a liaison to support the lines of communication with the other teams and was also responsible for team leader dutires.

One of the students volunteered to act as the project manager for the project. The liaisons of the teams reported on a weekly basis to the project manager who in turn was responsible to report the weekly activities to the faculty member.

The eight process activities used in SEP1 were modified to accommodate the incremental build model in SEP2. Unlike SEP1, the students were guided through the control over the project management activities, under the supervision and advice of the faculty member. A process was designed for tracking wherein members of each team filed weekly timesheets with their respective liaisons. Four releases of the project were created for easier project tracking. Similar to SEP1, risk management plans were devised.

The class decided to publish deadlines for the four versions. This plan gave the teams an opportunity to incorporate buffer time in the deadlines and also to schedule their activities correspondingly. The same tools as used in SEP1 were used. Together Control Center was a new tool that was used for design and implementation. The student grades were based on the criteria of quality of work submitted, participation in the project , and compliance to the SEI-CMM levels. The final demonstration of the project was conducted on the day of the scheduled final exams.

4 SOFTWARE PROCESS IMPROVEMENT AND TRACKING TOOL REQUIREMENTS FORMULATION

An initial self-evaluation was conducted within SEP2 in order to determine compliance with each of the goals in each of the key process areas as provided in the SEI-CMM (Paulk, 1993).

A final SEI-CMM evaluation was undertaken after the third version release in order to determine the compliance of the process with the CMM levels 1 through 5 of initial, repeatable, defined, managed, and optimizing, respectively. The teams were asked to report the results of their self-evaluations. The teams complied with the goals of the process activities for levels 1 through 3. The teams also satisfied most of the level 4 activities and some of the level 5 activities. Based on this evaluation, the process was modified to satisfy the completion of level 4 certification. Evaluations conducted at the end of the semester determined that the class as a whole had performed at a CMM level 4. The project would benefit from an independent validation of the self-assessment. There is currently not enough time for the validation. However, by moving these concepts earlier in SEP1, then feedback could be obtained from industry volunteers.

During the discussions in SEP2, especially during the reviews of the process documents, a need was determined to automate the process of document reviews. This result started out as the motivation for the Software Process Automation and Workflow System (SPAWS) project (Urban, 2002).

For a group of software engineers to work on a software project efficiently and effectively, a software process is considered necessary. In order for the software engineers to follow the process consistently and without any place for ambiguity, the process has to be well documented, reviewed, and automated. The SPAWS software currently enables users to follow the process change management activities - document review, code review, and code inspections (Urban, 2002).

During the initial stages of the project, discussions were conducted with individuals involved in the software engineering courses to refine the requirements for the software. Four releases of the project were created for easier project tracking. This approach gave an opportunity to incorporate buffer time in the deadlines and also to schedule activities correspondingly. The same tools as used in the SEP1 and SEP2 courses were used.

Subsequent offerings of SEP2 have also been driven by the SEI-CMM self-assessment. These self-assessments have identified the need for a standard process to be implemented earlier than in SEP2. The Unified Process for EDUcation (UPEDU) has been the basis for process modelling (École Polytechnique de Montréal, 2004).

5 SUMMARY AND FUTURE RESEARCH

This paper discussed capstone sequence software engineering courses and the analysis and process activities associated with software engineering tools that were developed in support the software engineering projects. An outgrowth of this paper is planned for presenting the multi-year experience with this approach to web based software engineering standard tool development.

SPAWS is now available to the students of the Department of Computer Science and Engineering. The students of the future offerings of SEP2 will be using SPAWS in order to gather some experience before requiring use of the tool in SEP1.

Addition or deletion of features and process activities, data storage enhancements, and interfacing with other software engineering tools are some of the future research that could be conducted regarding this project. Finally, additional software engineering tool support will continue to be developed in conjunction with the software engineering project capstone sequence. A software engineering tool for group member scheduling is under development as an outgrowth of this effort.

REFERENCES

Ahamed, S. I., Ali, S., Bingham, D. G., Dawra, A., Ha, L. T., Luong, T. M., Martinez, D. M., Morris, J., Palangala, S. A., and Urban, J. E., 2000. "Software Requirements on the Web," In *Proceedings of the 4th International Conference on Business Information Systems (BIS'2000)*, Poznan, Poland, April 12-13, 2000, Springer Verlag London Ltd., pp. 133-144.

École Polytechnique de Montréal, 2004. *UPEDU*, http://www.upedu.org/upedu/index.asp

IEEE, 1997a. IEEE Std. 830-1993, IEEE Recommended Practice for Software Requirements Specifications, In *IEEE Standards Collection: Software Engineering*, IEEE, New York.

IEEE, 1997b. ANSI/IEEE Std. 1058.1-1987 (Reaffirmed 1993), IEEE Standard for Software Project Management Plans, In *IEEE Standards Collection: Software Engineering*, IEEE, New York.

IEEE, 1997c. ANSI/IEEE Std. 1016-1987 (Reaffirmed 1993), IEEE Recommended Practice for Software Design Descriptions, In *IEEE Standards Collection: Software Engineering*, IEEE, New York.

IEEE, 1997d. ANSI/IEEE Std. 1016.1-1993, IEEE Guide to Software Design Descriptions, In *IEEE Standards Collection: Software Engineering*, IEEE, New York.

IEEE, 1998a. IEEE Std 829-1998 Standard for Software Test Documentation, IEEE, New York.

IEEE, 1998b. IEEE Std. 1219-1998 Standard for Software Maintenance, IEEE, New York.

IEEE, 1998c. IEEE Std. 1021-1998 Standard for Software Verification and Validation, IEEE, New York.

Paulk, M. C., Curtis, B., Chrissis, M. B., and Weber, C. V. 1993. *Capability Maturity Model for Software Version 1.1*, Software Engineering Institute, Technical Report, CME/SEI-93-TR-024, ESC-TR-93-177, 82 pp.

Urban, J. E. and Sankaran, S., 2002. "Supporting Software Process Tracking Through the Internet," In *Proceedings of the 2002 IFIP Workshop on Internet Technologies, Applications, and Societal Impact (WITASI'02)*, Wroclaw, Poland, October 10-11, 2002, Kluwer Academic Publishers, Norwell, Massachusetts, pp.243-254.