Using Software Engineering Principles to Develop a Web-Based Application

Cynthia Y. Lester
Department of Computer Science
Tuskegee University
Tuskegee, Alabama, USA
cylester@mytu.tuskegee.edu

Abstract – **In the United States, the software development industry is about a $220 billon industry. Therefore, the need to produce accessible, reliable and trustworthy software that is within budget and that also meets the demands of the heterogeneous user can sometimes become an overwhelming task by organizations. Since its inception in the late 1960s, software engineering has used software processes as systematic approaches that lead to the development of software applications. However, with the introduction of the Internet and the World Wide Web, there have been changes to the way that software is produced. The aim of this paper is to present the results of an inquiry that introduced to undergraduate software engineering students an approach to developing a web-based application. Traditionally, specialized web engineering courses are offered at the graduate level or as an elective course in an undergraduate curriculum. The paper presents a semester-long project that combines some traditional software engineering processes with web engineering processes. The results from the study suggest that introducing a hybrid approach inclusive of traditional software processes and web engineering techniques can be done successfully at the undergraduate level but not without certain challenges.**

Keywords – ***software development; software engineering; web-based application; web engineering.***

I. INTRODUCTION

Software engineering is defined as a discipline that is concerned with all aspects of software production from the early stages of inception and specification to the maintenance of the system when it has gone into use. It has often been seen as the “cradle-to-grave” approach for producing reliable, cost-efficient software that is delivered in a timely manner, under given budget constraints, that meets the client’s needs.

The concept of software engineering was first introduced in 1968 at the NATO Science Engineering conference held in Garmisch, Germany, to discuss the ominous “software crisis [1].” The software crisis was a result of informal development practices used to meet the needs of a rapidly changing hardware industry. Software applications were often noted as being unreliable, complex, expensive and sometimes delivered years after the deadline. Since the inception of software engineering, tremendous strides have been made to develop effective strategies to deliver reliable, cost-efficient software.

Yet, with the advent of the World Wide Web, the topic of how to deliver trustworthy, cost-efficient web applications has become one of increasing importance. Software applications no longer run on a local machine and are accessed by only those who are part of the organization. Now, users demand that software be accessible wherever they are which brings a whole new notion to the delivery of reliable, cost-efficient software. Consequently, the role of software engineering is changing to meet the demands of the heterogeneous user.

The aim of this paper is to present the results from a semester-long project in which software engineering concepts were modified to develop a web-based application. Typically undergraduate students enroll in a general software engineering course which is part of the required curriculum. This course can be taught from many different perspectives. However, to gain specific knowledge in web engineering, students must often enroll in a separate course, if one is offered. However, a survey of various undergraduate curricula did not find many web engineering courses at the undergraduate level. Consequently, if educators want to introduce to the next generation of technologists these concepts, a traditional software engineering course serves as the best vehicle.

The paper begins by presenting several time-honored software development methodologies discussed in a traditionally taught software engineering course. This discussion provides the impetus for an introduction to web engineering concepts. Additionally, the paper presents a semester-long project in which students were engaged which focused on the practical implementation of software engineering concepts for a web-based application. Lastly, challenges and future work are discussed.

II. TRADITIONAL SOFTWARE DEVELOPMENT METHODOLOGIES

Since its inception, there have been many methodologies that have emerged that lead to the production of a software product. The most fundamental activities that are common among all software processes include [1]:

- *Software specification* – engineers and customers define the software and its constraints
- *Software development* – the software is design and programmed
- *Software validation* – the software is checked to ensure that it meets the customer requirements
- *Software evolution* – the software changes to meet the changing needs of the customer

Typically, students are introduced to these activities in the undergraduate computer science curriculum through a software engineering course. This course is often a survey course which exposes students to a variety of life cycle models. The course is frequently taught from a systems approach which places an emphasis on creating requirements and then developing a system to meet the requirements. In the traditional view of software development, requirements are seen as the contract between the organization developing the system and the organization needing the system [2].

A traditional view of software development is the waterfall method. The waterfall method was the first published software development process and forms the basis for many life cycles. It was noted as a great step forward in software development [3]. The method has stages that cascade from one to the other, giving it the "waterfall" name. Figure 1 is an example of the waterfall life cycle [4].

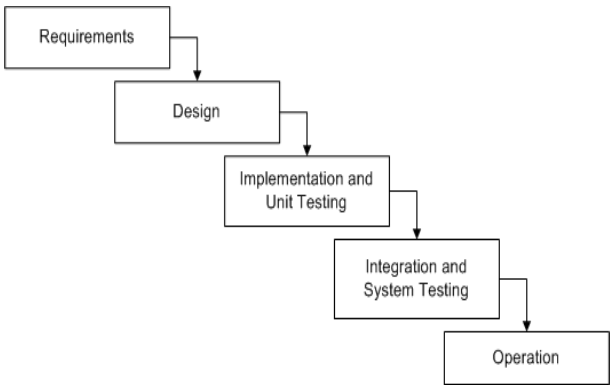
Requirements
Design
Implementation and Unit Testing
Integration and System Testing
Operation

Figure 1. Waterfall model

It has been noted that the method might work satisfactorily if design requirements could be addressed prior to design creation and if the design were perfect prior to implementation [3]. Consequently, one of the main disadvantages of this model is that requirements may change accordingly to meet the needs of the customer and the change is difficult to incorporate into the life cycle. As a result of this shortcoming, additional life cycles emerged which allowed for a more iterative approach to development.

Software prototyping is based on the idea where a version or sample of the software is developed to test requirements and design feasibility [1]. Figure 2 is a modified version of Sommerville's process of prototype development [1]. A reason why this particular model was introduced by the author to the undergraduate software engineering students is because it serves as a precursor to web-based application development. More specifically, software prototyping provides an effective way to gain understanding of requirements, provides early testing of system design, and reduces challenges during implementation. It has been stated that the advantages of software prototyping over its predecessor include that it accommodates change easier, it allows users to see how well the system can be integrated into current work activities, and it reveals errors and oversights early on in the process [1].

However, this approach to software development is not without some concerns. For example, the prototype may not be used in the same manner as the final system, the skill level of the testers of the prototype may differ from the users of the system, and because prototypes lack full functionality rapid changes may be made without proper documentation [1]. Consequently, these concerns provide an impetus to review the spiral model and agile methods, which are presented in the next.

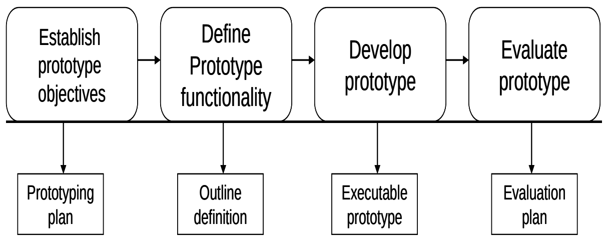

Figure 2. Prototype development

The spiral development model is also an example of an iterative process model that represents the software process as a set of interleaved activities that allows activities to be evaluated repeatedly. The model was presented by Barry Boehm in his 1988 paper entitled *A Spiral Model of Software Development and Enhancement* [5]. The spiral model is shown in Figure 3 [1]. The spiral model differs from the waterfall model in one very distinct way because it promotes prototyping; and, it differs from the waterfall and incremental development method because it takes into consideration that something may go wrong which is exercised through risk analysis.

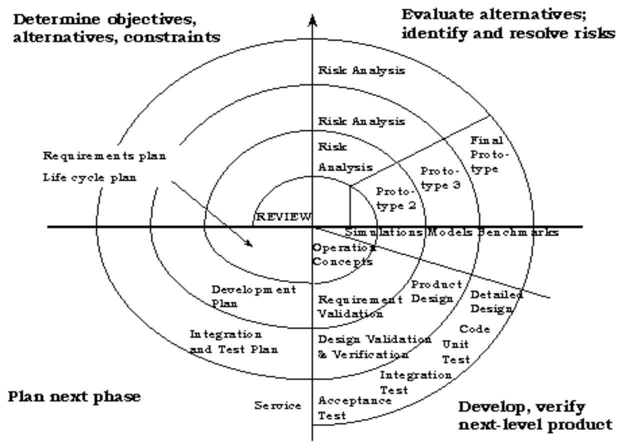

Figure 3. Spiral model

It is noted that this life cycle provides more flexibility than its more traditional predecessors. Further, this method produces a preliminary design. This phase of the life cycle was added specifically in order to identify and resolve all the possible risks in the project development. Therefore, if risks indicate any kind of uncertainty in requirements, prototyping may be used to proceed in order to determine a possible solution.

However, in these approaches, as with many of the other approaches to software development that are taught in traditional software engineering courses, a true focus on the software is mostly absent from the process. Hence, the increasingly important need to include a discussion of how to deliver working software to customers quickly. The next section explores agile methods and its life cycle.

III. AGILE METHODS

In an effort to address the dissatisfaction that the heavy-weight approaches to software engineering brought to small and medium-sized businesses and their system development, in the 1990s, a new approach was introduced termed, "agile methods." Agile processes are stated to be a family of software development methodologies in which software is produced in short releases and iterations, allowing for greater change to occur during the design [6]. A typical iteration or sprint is anywhere from two to four weeks, but can vary. The agile methods allow for software development teams to focus on the software rather than the design and documentation [1]. The following list is stated to depict agile methods [1], [6]:

- *Short releases and iterations* - allow the work to be divided, thereby releasing the software to the customer as soon as possible and as often as possible
- *Incremental design* – the design is not completed initially, but is improved upon when more knowledge is acquired throughout the process
- *User involvement* – there is a high level of involvement with the user who provides continuous feedback
- *Minimal documentation* – source code is well documented and well-structured
- *Informal communication* – communication is maintained but not through formal documents
- *Change* – presume that the system will evolve and find a way to work with changing requirements and environments

More specifically, the agile manifesto states [1]:
"We are uncovering better ways of developing software by doing it and helping others to do it.
Through this work we have come to value:
Individuals and interaction over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan
That is, while there is value in the items on the right, we value the items on the left more."

While agile methods are considered as light-weight processes as compared to the traditional software processes, they too are not without some controversy. For example, it is sometimes difficult to keep the customer involved after software delivery; there may be resistance to change and tool integration; as well as teaming issues. Therefore, in an effort to address these concerns as it relates to developing web-based applications, new models in web engineering emerged.

IV. WEB ENGINEERING

The World Wide Web can be described as a multimedia environment that allows documents to be seamlessly linked over the Internet [7]. It was developed by Tim Berners-Lee with help from Robert Cailliaua, both researchers at the European Laboratory for Particle Physics (CERN) [8]. The basic idea was that documents stored on computer that were linked by a network could be accessed by an authorized individual using the network. This idea however, relied on two types of software, a web server and a web browser. The web server stores the documents and "serves" them to other computers who desire access to the documents [7]. The web browser allows user to request and view the documents [7]. These ideas became common place in the 1990s and provide the foundation of today's Web and its many uses.

In 1990, it was reported that there were less that 50 million users of the Internet in the U.S. However, by 2008 the U.S. reported approximately 230,630,000 Internet users [9]. Therefore, it stands to reason that with more users and more advanced systems, the user population of today's technology would be more technically savvy than those user groups of yesteryear. However, the average user is now less likely to understand the systems of today as compared to the users of a decade ago. Consequently, the designers and developers of these applications must ensure that the software is designed with the three “use” words in mind so that the user experience is successful. Hence, the application must be useful, usable, and used [2].

A. Web Engineering as a Multi-disciplinary Field

Web engineering is an emerging multi-disciplinary field that is concerned with the development of web-based applications and systems [11]. As stated by Ginige and Murugesan, the premise of web engineering is a proactive approach taken to successfully manage the diversity and complexity of web application development and to avoid potential failures [11]. Consequently, web engineering encompasses not only the technical aspects of software engineering and its traditional software processes, but also the business-related area of project management, and the humanistic side of computer science, human-computer interaction. Figure 4 is a representation of the many disciplines that provide the foundation for web engineering [12].

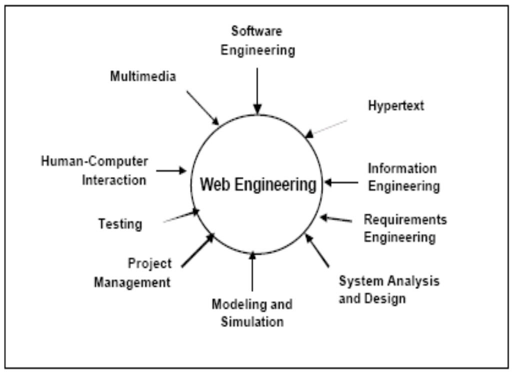

Figure 4. Web engineering

B. Web Engineering Activities

Since web engineering deals with all aspects of web-based system development, it too has many different activities akin to software engineering. These activities begin with specification, and continue with development, validation and evolution. However, specific web engineering activities include [12], [13]:

- Requirements engineering for web applications
- Techniques and methodologies for modeling web applications
- Design of functionality and interaction
- Implementation using a language for web-based applications
- Performance evaluation including verification and validation
- Operation and maintenance

Even more specific to web engineering activities are those that focus on the interaction between the application and the user. Activities that focus on the humanistic side of web development include [12]:

- Human and cultural aspects
- User involvement and feedback
- End-user application development
- Education and training
- Team and staff development

The next section presents a semester-long project that introduces to undergraduate software engineering students an approach to developing a web-based application. The project combines the concepts of traditional software processes with web engineering.

V. DEVELOPING A WEB-BASED APPLICATION

It was the anticipation of the author that through the hands-on experience of developing a web-based application, students would gain an understanding of the software engineering process, various process models and how they could be manipulated and combined to develop a web-based application. It was also the anticipation of the author that students would understand that the fundamental ideas of software engineering are still relevant in the development of today’s software applications as well as those web-based applications.

A. The Project

The semester-long project selected for the fall 2010 semester was to develop a web-based application that could be used during the academic advising process by students and faculty in a medium-sized computer science department. The application was to replace a paper-based method used by students and faculty. The department has approximately 100 undergraduate majors which are advised by roughly seven faculty members. Student advisees are paired according to last name with a faculty advisor.

Currently, the department uses a paper-based method on which students and faculty record by hand the student’s courses, credit hours, grade earned, and semester in which a course is taken. The recording of this information is either done during the University’s registration period or during the academic advising timeframe which happens twice within the academic year, once during the fall semester and another in the spring semester. During the meeting, the student advisee discusses with the faculty advisor the progress made toward the completion of the computer science degree. At the meeting the student advisee and the faculty advisor review the paper. The paper is called a *Provisional Sheet*. The *Provisional Sheet* is updated by hand by the student and the faculty member. At the completion of the meeting, each has a copy of the updated *Provisional Sheet.*

However, problems arise if the *Provisional Sheet* is lost; if updates are made on one sheet and not the other; or if a grade, course number, or course credit is written incorrectly.

At the end of a student's matriculation, the updated *Provisional Sheet* which should now contain all the courses, grades, and credit hours in which the student was enrolled is given to the Office of the Registrar for graduation preparation. It is, at this time, if an error has occurred that it is identified. While both the student and the faculty member have sufficient time to correct an error that might have happened, the paper-based method does not truly permit for a check-and-balances procedure.

Therefore the focus of the semester-long project was to develop a web-based application that would alleviate some of the difficulties as seen with the paper-based Provisional Sheet. The goal of the project was to design the *Provisional Sheet* so that it could be completed and updated on-line by authorized users only, and be cross-checked with the University's student information system.

B. Learning Outcomes

Students were part of a team which was expected to meet with the customer (or representative) so that each phase of the process could be implemented. The team was also expected to produce a deliverable by the set deadline for each phase of the process and to also deliver it and make presentations to the customer (or representative).

The learning outcomes of the semester-long project included that after the completion of the project students would:

- Have a working knowledge of software engineering principles
- Understand how software engineering principles could be applied to the semester-long project
- Identify activities and implement strategies that were germane to the development of a web-based application
- Work effectively and efficiently in a team environment to produce the semester-long project

C. Project Requirements

Students were given basic requirements from the instructor for the web-based application; however, the majority of the requirements were gathered from stakeholders. Since the project was a web-based application, students had to consider basic requirements found in traditional software process models, as well as those that are part of the web engineering process.

D. Project Deliverables

Each item that the student team submitted was considered a deliverable. The project had four deliverables which were the requirements document, design document, implementation, and the test plan. The following is an overview of the project deliverables (i.e., models for web-based applications, language for web-based applications).

1) Requirements Document. The first document students were required to submit was the requirements document. The requirements document was considered the official statement of what the students would implement. It included both the stakeholder requirements for the software application, which students named the *Computer Science Provisional System (CSPS),* and a detailed specification of system requirements.

To capture the requirements students engaged in a modified version of the requirements engineering process as presented by Sommerville [1] and by Kappel et al. [13]. During the requirements engineering process, students met with stakeholders who included faculty members and students in the computer science department. Additionally students met with staff members in the Office of the Registrar as well as identified faculty and students in other academic units on campus in order to complete the elicitation and analysis phase of the requirements engineering process.

The document was meant to get the students actively involved in the planning and development of the application. Consequently, after the completion of the requirements document, students had an idea of the system architecture, functional and non-functional requirements, external interface specifications, how the application would be accessed, and by whom. Moreover, because this was to be a web-based application and not a traditional desktop application, students were charged with identifying varying levels of risk which included defining in the requirements document the method to secure student information and how change was to be accommodated.

2) Design Document. The design document was meant to be an in-depth description of the system design. The design showed how data flowed between system components and the trust relationships between components. Since the application was a web-based application, both the system and security requirements were described and explained how they would be implemented. Further since the *CSPS* would contain personal and confidential information, the design document elaborated on the requirements document risk analysis of critical characteristics of information that was presented in a module of the course designed by Lester on software security [14].

The team was required to use one of the decomposition strategies discussed in the course. The design document was required to have an introduction, an overview of the design strategy chosen, and the diagrams, charts, and/or details required as part of the decomposition strategy chosen. Additionally, the design document was to be based on one of the architectural design patterns which were discussed in class. The team chose the Model-View-Controller (MVC) pattern because of the stated advantages: team members had varying technical skill levels, MVC separates design concerns, and there is the likelihood of less code duplication [15].

3) Implementation. Students were required to implement the project based on the requirements and design documents. To implement the project students chose the PHP scripting language for use in a Linux environment. The student team chose this language and the environment based on familiarity and accessibility to the MySQL server database.

4) Testing. Students were required to develop a test plan which required them to perform development testing and end-user testing. The test plan was based on Sommerville's structure of a software test plan for large and complex systems but modified to be less formal and represent the smaller nature of the *Computer Science Provisional System* [1]. The modified version of the software test plan included: the testing process, test case design, hardware and software requirements and constraints.

VI. CONCLUSION

The aim of this paper was to present the results of an inquiry that introduced to undergraduate software engineering students an approach to developing a web-based application. The paper discusses traditional software engineering principles typically introduced in undergraduate software engineering courses which provides a case for introducing in these same courses some of the topics found in web engineering.

The study revealed that it is difficult to implement the entire software development life cycle. While students have a very good understanding of the requirements engineering process and developed a well-planned requirements document and design document, the implementation and testing phases proved to be challenging. The student team indicated that with the fixed timeframe of a sixteen week semester, it was difficult to fully implement the phases of the life cycle. Also, because students were under the impression that developing a web-based application would be similar to developing a web page or web site with which they had previous experience, they underestimated the time needed to produce the required documentation. They also underestimated the understanding of client/server network technology and the time needed to conduct accessibility and usability testing. Consequently, based on these observations, future work for the author is to re-design the semester-long from a heavy-weight process which is plan-driven to a more light-weight process focuses more on the software (i.e., the use of agile methods with a focus on extreme programming).

In conclusion, as the users of today's Web demand more from their web applications, it is the responsibility of educators to train the technologists of tomorrow to meet those demands. Consequently, the traditional software engineering practices that rely on well established development processes must also embrace change in order to continue to produce reliable, trustworthy software applications specifically developed for the Web.

ACKNOWLEDGMENTS

The author thanks the students enrolled in the CSCI 430-Software Engineering course, fall 2010 semester.

REFERENCES

[1] I. Sommerville. (2011). *Software Engineering 9th Ed*. Addison Wesley, 13:978-0-13-703515-1, Boston, MA.

[2] C. Angelov, R.V.N. Melnik, and J. Buur. (2003). The synergistic integration of mathematics, software engineering, and user-centered design: exploring new trends in education. *Future Generation Computer Systems.* Vol. 19, 299 – 1307.

[3] B. K. Jayaswal and P.C. Patton (2007). Design for trustworthy software: Tools, techniques for developing robust software. Prentice Hall, 0-13-187250-8, Upper Saddle Rover, NJ.

[4] Codebetter.com http://codebetter.com/blogs/raymond. lewallen/downloads/waterfalllModel.gif. (Accessed on October 10, 2009.

[5] B. Boehm. (1988). A Spiral Model of Software Development and Enhancement. *IEEE Computer 21*, 5, 61-72.

[6] F. Tsui and O. Karam. (2011). *Essentials of Software Engineering 2nd Ed*. Jones and Bartlett Publishers, 13:978-0-7637-8634-5.

[7] D. Reed (2008). A Balanced Introduction to Computer Science 2nd Ed. Pearson Prentice Hall, 13:978-0-13-601381-5.

[8] A Short History of the Web, Text of a speech delivered at the launching of the European branch of the W3 Consortium Paris, 2 November 1995. http://www.netvalley.com/archives /mirrors/robert_cailliau_speech.htm (Accessed May 31, 2011).

[9] Internet users as percentage population. http://www. geohive.com/ charts/ec_internet1.aspx (Accessed December 20, 2010).

[10] A. Dix, J. Finlay, G.B. Abowd, and R. Beale. (2004). *Human-Computer Interaction.* Prentice Hall, Boston, MA.

[11] A. Ginige and S. Murugesan. (2001). Web Engineering: A Methodology for Developing Scalable, Maintainable Web Applications. *Cutter IT Journal.* Vol. 14, No. 7, 24-35.

[12] S. Murugesan, Y. Deshpande, S. Hansen, and A. Ginige. Web Engineering: A New Discipline for Development of Web-based Systems. (2001). *Proceedings of Web Engineering*, pp.3-13.

[13] G. Kappel, B. Pröll, S. Reich, and W. Retschitzegger (eds), (2006). *Web Engineering - The Discipline of Systematic Development of Web Applications.* John Wiley & Sons.

[14] C. Lester and F. Jamerson. "Incorporating software security into an undergraduate software engineering course," in *Proc. of the Third International Conference Emerging Security, Information Systems and Technology*, 2009, pp. 161-166.

[15] Designing Enterprise Applications with J2EE Platform 2nd ed, http://java.sun.com/blueprints/guidelines/designing _enterprise_applications_2e/web-tier/web-tier5.html (Accessed August 4, 2011).