

Software Engineering and Quality Assurance Comparison of Tools and Techniques used by Universities

Srinivas Vittal, Raghvarma Basavaraju, Ancy Varghese, and Hanting Lin

Computer Science Department, Pace University
861 Bedford Road, Pleasantville, New York, 10570 USA

SVittal@iseoptions.com, raghuvarma.basavaraju@oracle.com, ancy_p_v@yahoo.com, hanting_lin@hotmail.com

Abstract

As the major focus of Pace University's CS615-616 Software Engineering Seminar, students form project teams to create real world systems for actual customers. Our team is dedicated solely to providing ongoing project support for the other project teams. The goal is to implement a new Utopia Project Server and manage the development and staging servers to show that providing quality assurance and testing early in a project will reduce the overall work effort required. This paper provides a comparative analysis of the project management and project support tools and methods used by Pace University and other universities who offer similar software engineering courses as part of their curriculum. The results achieved by this team and recommendations for improving the current curriculum of this course are also presented. Three universities were reviewed as part of this study – John Hopkins University, New York University and Southern California University.

Keywords

Software Engineering, Quality Assurance, University, Course work, Project management

1. Introduction

1.1 Software

Software encompasses programs that execute within a computer of any size and architecture, content that is presented as the computer programs execute, and documents in both hard copy and virtual forms that encompass all forms of electronic media.

Software has become the key element in the evolution of computer based systems and products and one of the most important technologies in the world stage.

1.2 Software Engineering

Software engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.

The intent of software engineering is to provide a framework for building higher quality software. The challenge of software engineers is to build applications that will facilitate mass communication and mass product distribution using concepts that are forming now and evolving.

1.3 Project and Project Management

A *project* is a temporary effort, undertaken with a defined start and end date, to create a unique product, service, or result.

The 3 characteristics, which define a project

- Scope/Objectives (products, deliverables)
- Time (project schedule)
- Cost (budget)

A project is implemented against:

- Resource constraints / availability
- Risks & issues
- Other competing priorities

Project Management is the application of knowledge, skills, tools, and techniques to project activities in order to meet or exceed the Sponsors needs and expectations. It is the art of balancing competing demands among scope, time, cost and quality.

The objective of Project management is

To enhance the value of the organization through the timely delivery of objectives, in the most cost effec-

tive way possible with the right resources providing subject matter expertise.

1.4 Quality and Quality Assurance

Software Quality is a characteristic or attribute of software such as cyclomatic complexity, cohesion, number of function points, lines of code etc. and Quality includes both quality of design and quality of conformance. A quality software product is one that satisfies the user requirements.

It can even be viewed as a *function of how much it changes the world for the better*.

Quality assurance consists of a set of auditing and reporting functions that assess the effectiveness and completeness of quality control activities. The goal of quality assurance is to provide management with the data necessary to be informed about product quality, thereby gaining insight and confidence that product quality is meeting its goals.

2 The Objectives

2.1 General Objectives

This is a yearly project that has the purpose of ensuring standardization and quality of our project systems. This project is ideal for those interested in or having expertise in QA and system maintenance.

This project will be responsible for the following:

- Update description of and monitor the project development infrastructure (initial task)
- Recommend, develop plans for, and oversee project reviews
- Establish the QA measures for the current projects
- Extend the QA and maintenance tools from previous years or develop new ones (optional)
- Clean up a few of the important projects of earlier years (optional, extra credit, see instructor)

2.2 Comparison of Practices followed by various Universities

In addition to the above general objectives, we have taken up a special task this year to gather information on practices followed by various universities in teaching the Software Engineering capstone course and the associated projects management and QA techniques.

2.3 The universities

We considered 3 universities for our comparative study are

1. New York University
2. Johns Hopkins University
3. University of Southern California

3. Universities – Comparison of the Software Engineering courses offered

We mainly depended on the web sites of the universities for information on the course work and practices. Wherever possible we also tried to gather information through networking of friends. However this source was limited due to practical reasons and we depended to a larger extent on the information posted on the web sites.

3.1 NYU

Founded in 1831, New York University is the largest private university in the United States. The University, which is composed of 14 schools, colleges, and divisions, occupies five major centers in Manhattan. It operates branch campus and research programs in other parts of the United States and abroad, as well as study abroad programs in more than 25 countries.

3.1.1 Course Description

An intense hands-on study of practical techniques and methods of software engineering. Topics include: software processes and management, requirements engineering, software evolution and configuration management, advanced object-oriented design, design patterns, code construction techniques, verification and validation techniques, and code optimization and tuning. All topics are integrated and applied during the semester-long group project. The aim of the project is to prepare students for dynamics in a real workplace. Members of the group will meet on a regular basis to discuss the project and to assign individual tasks. Students will be judged primarily on the final project presentations.

3.1.2 Prerequisites

Familiarity with C++, experience with at least one large software project.

3.1.3 Textbooks

Sommerville, Ian. *Software Engineering*, 7th Edition .
Meyers, Scott. *Effective C++*, Second Edition

3.1.4 Lectures

Two 1.25 hr sessions per week.

3.1.5 Sections

There will be two one hour discussion sections per week, where students attending can receive additional help from the TA. Sections are intended to be more interactive than lectures.

3.1.6 Grading

Final grades will be based on the following:

30% Weekly Assignments

30% In Class Presentations

40% Final Project Report

3.1.7 Academic Honesty

In this course, you are encouraged to work together on the project and assignments. However, any help you receive must be clearly explained. Also, you should consult the instructor before using materials or code other than that provided in class. Copying without giving appropriate acknowledgement is a serious offense with consequences ranging from no credit to potential expulsion.

3.2 John Hopkins University

Founded in 1916, the Johns Hopkins University has been offering engineering education to working adults. The Whiting School of Engineering's Engineering and Applied Science Programs for Professionals provides continuing education to the technology workforce. Students can take courses at any of seven locations throughout central Maryland, although not all programs and courses are available at all sites.

3.2.1 Seminar in Software Engineering

- Instructor Information

The instructor is a software engineer who has built software for defense, medical data, biomedical analysis, and signal processing systems. Dr. Grossman's work areas of interest include software design for real-time and distributed systems, database applications, human machine interfaces, hardware interfaces, and architectural system design.

3.2.2 Course Information

This course examines the underlying concepts and latest topics in software engineering. Potential topics include effective software development techniques such as agile and extreme programming; use of UML to define testing strategies; useful development tools

and environments; patterns; metrics issues in system generation; teamwork in successful developments; and training aspects of CMM. Each student will select and report on a software engineering topic, perform independent reading, and prepare a paper describing a major software engineering issue. The course is taught using a seminar format in which significant portions of the class period are set aside for students to lead and actively participate in discussions.

3.2.3 Prerequisite: One software engineering course beyond 605.401 Foundations of Software Engineering.

3.2.4 Course Goal

To provide an open, informed forum for discussing and understanding some of the key methods and approaches in use by practicing software engineers.

3.2.5 Course Objectives

- Research current software engineering topics
- Understand software engineering methods and approaches to a deeper level than allowed for in other courses
- Share knowledge with other practicing software engineers
- Learn to openly discuss technical ideas with other software engineers

This course is typically offered during the spring semester at the APL Center.

Syllabus - Topics covered

What is software engineering?

What is a professional software engineer?

Student selected discussion topics

Student selected topics continue: several weeks

Coverage of topics not selected

3.2.6 Student Assessment Criteria

Class Participation	20%
Class Preparation	20%
Class Presentation	20%
Term Paper(about12-15pages)	40%

This course is a seminar in which all students study, learn and discuss various topics associated with software engineering. Most topics discussed are selected by the students in the class. Everyone is expected to read, participate in and lead discussions.

3.2.7 Computer and Technical Requirements

Students should have a practical and working knowledge of software engineering. This can come from working as a software engineering professional or by taking at least one course beyond Foundations of Software Engineering (605.401).

3.2.8 Participation Expectations

Since one of the major learning components of this course is participation in discussions, students are expected to attend class. Generally missing more than two class discussions is detrimental to one's grade.

3.2.9 Textbooks

There is no required textbook for this course. Reading will be done in various texts and journals. Please call or email the teacher if you would like to have any of the above explained in more detail.

3.3 University of Southern California

USC was founded in Los Angeles in 1880 with 53 students and 10 teachers. USC's University Park Campus, located in the heart of Los Angeles' Downtown Arts and Education Corridor, is home to the USC College of Letters, Arts and Sciences and many professional schools.

3.3.1 Course Description

This is the first of two courses in the Software Engineering sequence, which forms the core of the Master of Science in Computer Science with specialization in Software Engineering. Software Engineering I focuses on software plans, processes, requirements and architectures. Software Engineering II focuses on software product creation, integration, test and maintenance with an emphasis on quality software production. Much of the content is organized around the key practices in the SEI Integrated Capability Maturity Model (CMMI).

This course will focus on the application of software engineering process models and management approaches for design and architecture of large software systems. Students will work in teams and be required to understand and apply the Win-Win spiral model and Model Based System Architecting and Software Engineering guidelines for software engineering to real-world projects. Students will also be expected to

understand and apply quality management approaches to their projects.

Past projects have included development of e-commerce systems for commercial entities, projects for various needs of the Center of Software Engineering and, projects for USC's Information Services Division (ISD) . During this course, the student team members will formulate operational concepts, requirements specifications, architectures, prototypes, life cycle plans, and integrating rationale for the proposed capabilities. In CSCI 577b, student teams develop Initial Operational Capability products based on the best results from CSCI 577a.

The DEN students will be responsible for "Independent Verification and Validation" (IV & V). Verification and validation is one of the software engineering disciplines that helps build quality into the software. V & V is a collection of analysis and testing activities across the full life cycle and complements the efforts of other quality-engineering functions. It determines that the software performs its intended functions correctly, performs no unintended functions and measures the quality and reliability of the software. The DEN students will each be assigned a single project for which they will play the role of IV & V.

Course Prerequisite: Graduate standing

Lecture Time and Location: MWF, 2:00-3:20 pm

3.3.2 Assignments and Grading (Approximate Distribution)

Homeworks		23%
Class	Assignments	16%
Project package	(Team assignments)	47%
Individual Project	Critique	11%
Individual Project Contribution		3%

Submissions are due at or before 2:05 pm, they will be picked up in class, unless specified. For late submissions a penalty of 15% for each day the assignment is late. The student will be in charge of making sure the assignment is then correctly handed to the TA's in the case of a late submission.

3.3.3 Textbooks

Balancing Agility and Discipline: A Guide for the Perplexed by Barry Boehm and Richard Turner, Addison Wesley Professional

Barry W. Boehm, et al, Software Cost Estimation With COCOMO II, Prentice Hall, New Jersey, 2000

3.3.4 ASSIGNMENTS

All assignments are due at the start of class (on or before 2:05pm). There is 15% penalty per late day.

- INDIVIDUAL Assignments
- INDIVIDUAL CLASS EXERCISES: Pre-class (To be done at home to submit in class); Questions for in-class to be handed during the lecture.
- TEAM Assignments
- WEEKLY Assignments

3.4 Pace University

3.4.1. Textbooks:

Software Engineering, Roger Pressman, 6th Ed., McGraw (2004), ISBN 007301933x

The Mythical Man Month, Fred Brooks, Addison (1995), ISBN 0201835959

3.4.2. Course Prerequisite:

Twelve credits of 600-level course work.

3.4.3 Course Description:

CS615 : This is the first semester of the two-semester capstone course on software engineering for CS majors. The goals of the course are to understand what a Computer Information System (CIS) is, the importance of a systematic approach in CIS design, and how to conduct requirements analysis and develop a real-world CIS. This semester we will focus on the software engineering process and practice, and next semester on managing software projects and the more advanced topics in software engineering.

CS616 : This is the second semester of a two-semester course on software engineering. The goals of the course are to understand what a Computer Information System (CIS) is, the importance of a systematic approach in CIS design, and how to develop a real-world CIS. This semester you will reengineer and add functionality to your first semester system and work more closely with your customers, the QA team, and your instructor.

This course uses an extensive course website to present the course information: course requirements and grading system, projects, current grades, syllabus, student and team information, and links to related websites, to papers in PDF, and to related conferences/exhibits.

3.4.3 Graded Events and Grade Scale

Ten open-book quizzes. Quizzes are to be taken via Blackboard with a 15 minute time limit. Occasionally, however, a quiz may be given during the class session (students absent for such a quiz receive a 4-point penalty when taking the make-up). The quizzes are designed not only to check that you read the assignments

but, more importantly, to increase your capability to quickly process and comprehend software engineering and related computer science material.

Open-book Midterm and Final Exams are based on the material covered and typically consist of multiple choice questions, design/calculation problems, and short answer questions.

3.4.4. Project

A *Team Project* focuses on developing a CIS that meets a real customer's real needs. Although the requirements for the projects come from the customers, the course instructor is the "boss" or "Chief Executive Officer" of the project teams – that is, the person who makes all the major decisions. Most of the systems will involve one or more of the following: programming, a database, a computer network, a Web interface. Java is the preferred language for projects that require programming. A team consists of 3-5 students as follows:

Teams choose an Architect-Designer, one or two Implementers, a Quality Officer, and a team Coordinator-Liaison. For small teams several team member functions can be combined. At least one team member, usually the Coordinator-Liaison, must be a good communicator for client and instructor interactions.

Once the project is underway, teams should meet once a week in addition to project work time.

Except for extenuating *circumstances communication between your team and instructor must be through your team leader.* Communication between your team and your client must be done as a group or with the approval of your team leader.

Teams may be reconstituted over the course of the academic year due to problems with student class schedules, in response to the needs of the project, or for other extenuating circumstances. However, every effort will be made to keep the same students together on a project over the entire academic year.

Each team will deliver a prototype system that performs the basic required functions to their client at the end of the first semester. This should be possible since, according to the 80-20 rule, 80% of the project can be completed in 20% of the time it would take to deliver the complete 100% system. A complete, high-quality system will be delivered at the end of the second semester. In the spring semester we anticipate that technical papers related to most of these projects will be presented at a student conference such as MASPLAS or Pace University's CSIS Research Day, or will be published as Pace University CSIS Technical Reports.

An instructor provided grade (IPG) is for your individual class participation, contribution, attendance, positive influence, etc.

Project Support and Demos

Most of the projects systems use the client/server architecture. The application systems use a variety of database-related software, including different scripting languages, such as Cold Fusion, PHP, Perl, and JSP, to communicate with backend databases in Microsoft Access, MySQL, MS-SQL Server, and Oracle. Some projects also use Java-related software, such as Java servlets, and Tomcat is installed to handle the processing of that code.

Project Development Servers (Internal access only)

Three Win2000 servers and three RedHat Linux 9 servers located in Graduate Center room 408, having access only within the university, provide development platforms for the student systems. These servers initially contain no development software, so the students must install software on these hosting platforms as required by their projects. All project files must be tested here before moving to the staging server.

Project Staging Server (outside access)

After the development phase, the students move their application systems to a staging server having access from outside the university. Students must FTP their project files to the staging server for further system development and testing. Although all project systems should be finalized on the staging server, some may be moved to production servers within or outside the university. The development and staging servers are independent and separate from the CSIS production servers so that students cannot corrupt data or interfere with operations on those servers.

4. Results and Conclusions

The QA team has completed the following 3 major deliverables as part of this project:

1. Implementing a new utopia project staging server with many additional feature enhancements. The Utopia Server upgrade Project brought the following benefits to the student community and the University:

- a. Improved hardware and operating system.
- b. Latest versions of all the software required for student projects.
- c. Improvements in disk layout in a reducing the fragmentation of file system
- d. Less complicated security for Matlab and Terminal Services
- e. Secure FTP connections to avoid any hacker attacks

- f. Avoidance of scope creep resulting in a big ball of mud
- g. Removal of large amounts of data with little connection to Pace coursework

2. A technical paper comparing three major Universities offering a Software Engineering Seminar

We have compared information on the software engineering capstone course teaching and associated project implementation/QA practices followed by 3 universities: New York University, Southern California University and John Hopkins University. These universities like Pace appear to be successfully using software engineering as a capstone course – not just another theoretical computer science course. All universities are encouraging students to do practical projects/assignments and are providing the necessary infrastructure. Overall pace University has very good infrastructure to support student projects. Students can develop the projects in a separate Development server and then they can move their applications to a Staging or Production servers. We upgraded the Production Server Utopia to support complex projects which require high performance hardware to support.

3. Updating all the project support web pages with new staging server information. Finally to finish the project we are updating with all the latest project support information on the Course web pages for the new students starting from next semester.

To conclude, this was a great project with lots of support from Pace University DoIT for the server upgrade project. Our team did a wonderful job delivering the project on time.

References

1. John Hopkins University web site
http://www.epp.jhu.edu/pages/viewpage.php?homepage_id=2568
2. Roger S. Pressman, *Software Engineering, A Practitioner's Approach*, Sixth Edition, McGraw-Hill, 2005.
3. New York University web site
<http://www.nyu.edu/>
4. University of Southern California
<http://www.usc.edu/academics/schools/>
5. Introduction to Testing Object-Oriented Software by John D. McGregor, David A. Sykes Addison Wesley (Oct 19, 2001)