AC 2008-1705: ENHANCING THE SOFTWARE VERIFICATION AND VALIDATION COURSE THROUGH LABORATORY SESSIONS

Sushil Acharya, Robert Morris University

Sushil Acharya, D.Eng. Assistant Professor of Software Engineering Acharya joined RMU in Spring 2005 after serving 15 years in the Software Industry. With US Airways Acharya was responsible for creating a Data Warehouse and using advance Data Mining Tools for performance improvement. With i2 Technologies he led the work on i2's Data Mining product "Knowledge Discover Framework" and at CEERD (Thailand) he was the product manager of three energy software products (MEDEE-S/ENV, EFOM/ENV and DBA-VOID) which are currently in use in 26 Asian and 7 European countries by both governmental and non-governmental organizations. Acharya has a M.Eng. in Computer Technology and a D.Eng. in Computer Science and Information Management with a concentration in knowledge discovery, both from the Asian Institute of Technology in Thailand. His teaching involvement and research interest are in the area of Software Engineering and Development (Verification & Validation), Data Mining, Data Warehousing, Neural Networks and Enterprise Resource Planning. He also has interest in Learning Objectives based Education Material Design and Development. Acharya is a co-author of "Discrete Mathematics Applications for Information Systems Professionals- 2nd Ed., Prentice Hall". He is a life member of Nepal Engineering Association and is also a member of ASEE, ACM and IEEE. Acharya is a recipient of the "Mahendra Vidya Bhusak" a prestigious medal awarded by His Majesty the King of Nepal for academic excellence. He is a member of the Program Committee of WMSCI, MEI, and CCCT and is also a Member of the Editorial Advisory Board of the Journal of Systemics, Cybernetics and Informatics of the International Institute of Informatics and Systemics. Acharya is also the Principal Investigator of the 2007 HP grant for Higher Education at RMU.

Enhancing the Software Verification and Validation Course through Laboratory Sessions

1. Introduction

Many engineering courses are taught through lecture-only sessions and students garner experiences through course based projects and internships. However these methods alone do not suffice to place fresh engineering graduates at a competitive advantage in the job market. In good and bad times employers look for engineers with job-related experience since such engineers require less training and provide faster results. In view of this, course enhancements and delivery with focus on real-life work experience needs to be embraced by engineering programs. Software Engineering (SE) is one such engineering discipline where curriculum enhancement through laboratory sessions will highly benefit its graduates.

Software Engineering (SE)

Prior to the 1990s SE was not an explicit engineering discipline. Computing Curricula 2005[1] states that during the 1990's SE began to develop as a discipline unto itself. Since then this discipline has been playing an important role in the multibillion dollar software industry. SE is defined as the discipline of developing and maintaining software systems that behave reliably and efficiently, are affordable to develop and maintain, and satisfy all the requirements that customers have defined for them[1]. Figure 1 depicts the conceptual territory occupied by SE in the problem space of computing. Unlike other computing disciplines (like IS, IT, CS and CE), SE domain spans on all directions. This is because software engineers fill a wide range of needs in large-project software expertise[1].

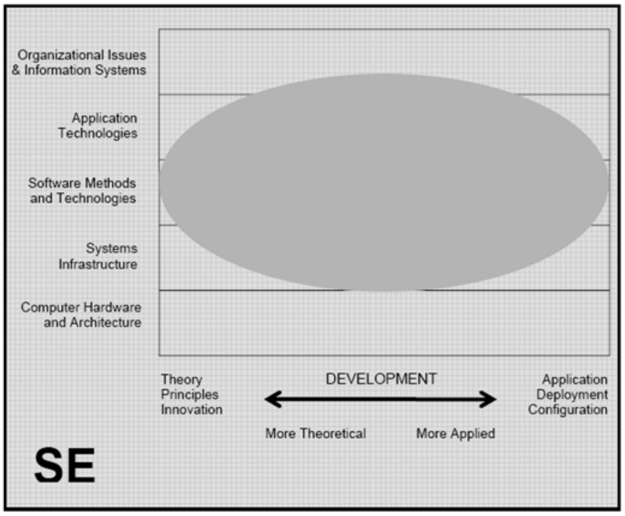
Organizational Issues & Information Systems
Application Technologies
Software Methods and Technologies
Systems Infrastructure
Computer Hardware and Architecture
Theory Principles Innovation
DEVELOPMENT
Application Deployment Configuration
More Theoretical
More Applied
SE

Figure 1: Software Engineering Conceptual Territory[1]

SE's main goal is to develop systematic models and reliable techniques for producing high-quality software on time and within budget, and these concerns extend all the way from theory and principles to daily practice[2]. This goal indicates that software engineers need both theoretical and applied knowledge to perform. Lecture-only courses fulfill some aspects of this goal. The remaining aspects should be covered through laboratory sessions where students are provided hands-on experience on methods and tools.

The competitive software job market demands engineers to be experienced in different aspects of SE. At all times software engineers must ensure that the resulting program (software) satisfies its specification and that the program as implemented meets expectations of the stakeholders[2]. For the resulting program to meet such stringent requirements software engineers must be well versed in the theoretical and practical aspects of Software Verification and Validation. Software Verification and Validation is also identified by ACM/IEEE Software Engineering 2004 as one of the ten Knowledge Areas (KA) that make up the SEEK (Software Engineering Education Knowledge) [2].

Software Verification & Validation

In every aspect of our lives software systems are an important partner providing us service efficiency and service reliability. Our growing dependence on software systems demands that the software industry deliver quality software products. Erroneous software continues to be costly. We have learned valuable lessons from the Ariane 5 Bug, the NASA English/Metric System Bug, the Air Traffic Control System Bug, the Therac-25 Accidents, the Patriot Missile Bug and the Therac-25 Accidents[3].

Software Verification and Validation, also known as Verification and Validation (V&V), is that important activity in software development which is used to ensure that software conforms with its' specifications and meets customer requirements[4]. Verification deals with processes that ensure that the software is being built correctly i.e. the development conform to its specifications based on all the assumptions made. The question that is asked is "*Does the software do what I want it to do?*" or "*Is the product being developed right?*" Validation deals with processes that ensure that the correct software is being built i.e. the development conform to the needs of the customer. The question that is asked is "*Does the software do what the customer wants it to do?*" or "*Is the right product being developed?*" Graduates seeking work in the software industry need to be taught both theoretical and applied concepts of V&V.

This paper discusses a successful enhancement of a SE course namely "ENGR3400: Software Verification and Validation" through laboratory sessions. Section 2 describes the pre-enhanced and enhanced course syllabi. Key employment skill areas are also discussed in this section. Section 3 presents an outline of the enhanced course. Course delivery and outcome assessment analysis are discussed in section 4 and section 5 respectively. Finally conclusions and recommendations are presented in section 6.

2. Course Enhancement

ENGR3400: Software Verification and Validation is a 3 credits required course offered by the Software Engineering Program at Robert Morris University. This course addresses ABET Derived Outcomes 1, 2, 3, 4, 5, 6, 7, 8, 9, 11 and ABET Software Engineering Track Outcomes: S1, S2. This course also addresses the "NRC (National Research Council) challenge to effective undergraduate education in STEM (Science, Technology, Engineering, Mathematics) disciplines", namely the challenge of providing engaging laboratory, classroom and field experiences.

Prior to any enhancement this course was taught as three 50 minutes lecture only sessions per week. The course objectives were:

- To understand the concepts of Software Testing.
- To gain understanding of basic and advance Software Testing processes.
- To introduce tools and techniques for System Testing.
- To understand the importance of V&V.

The focus of this pre-enhanced course was primarily on software testing and covered topics such as test outlines, test cases, test tables, test results analysis, testing of web applications and testing of Object Oriented software. The course lacked hands-on exercises and did not require a student project. One other important issue with this course was the contents covered could have been easily delivered in 9 weeks (instead of the 14 weeks that was allotted) if time was effectively utilized.

The enhanced course provides an in-depth understanding of V&V. The enhancement incorporates the inclusion of a 2½ hour laboratory session and the reduction of lectures to two 50 minutes sessions per week. To shift the focus of the course towards theoretical and practical aspects of V&V and to be able to deliver the identified contents within the allotted time a tradeoff in course contents was necessary. Only 40% of the pre-enhanced course contents have been retained and are covered either as lectures or as discussions during project time. Some contents have been kept in their original form whereas other contents such as web testing, OO testing, test case reduction methods and test tables are retained in a condensed form. The reduction of lecture sessions has resulted in certain contents being delivered as collaborative learning exercises during lab sessions. The enhanced course objectives are:

- To understand the importance of V&V.
- To gain understanding of the V&V processes and industry best practices.
- To gain hands-on experience in V&V methods and tools.
- To understand the Software Testing process.

This enhanced course has been delivered since spring 2006. The lecture sessions of this course focuses on theoretical understanding of V&V processes, methods and tools. In the lab sessions V&V methods and tools are introduced with emphasis on Requirements Definition and Refinement, Peer Reviews, and Software Testing. Students also carry out individual projects in which they go through a software testing process by creating 50 test cases for a program they developed in a prerequisite SE course (ENGR3410: Fundamentals of Software Engineering).

The course enhancement involved understanding industry requirements and designing the course to implement these requirements. Discussion with prospective employers on their requirements, research of employment listings on job sites (www.monsters.com and www.careerbuilder.com), and discussion with software consultants on the field were used to understand industry requirements. Discussions with text authors, understanding of IEEE/ACM Curriculum Guidelines recommendations[2], the author's 20 years software development experience and the industry requirements were used to enhance the course. During the process, skill areas sought after by employers were studied. Four key skill areas were identified and used to drive the enhancement.

Key Skill Areas

i. **Communication Skills (C):** Students will gain experience in technical communication skills through role-plays, collaborative learning and technical presentations. This is in line with IEEE/ACM Curriculum Guideline # 8[2].

ii. **Applied Knowledge of Methods (M):** Students will be further exposed to V&V methods in the laboratory sessions. With practical exercises, case studies, case-based videos and through role-plays students will be able to map theory to practice. This is in line with IEEE/ACM Curriculum Guideline # 4[2].

iii. **Applied Knowledge of Tools (T):** Students will be exposed to V&V Tools. These tools will encompass software for version control and bug management. This is in line with IEEE/ACM Curriculum Guideline # 12[2].

iv. **Research Capability (R):** Students will understand the implementation of industry best practices through research and apply the same to their industry. Research assists in students making educated decisions.

In this paper a meaningful comparative study of the enhanced and pre-enhanced course outcome assessments could not be made as the objectives of the course contents are different. However outcome assessments for 2006 and 2007 are discussed.

3. Enhanced Course Outline

This course enhancement makes an attempt to cover relevant V&V topics with adequate laboratory sessions. Week by week lecture and lab coverage is presented in Table 1. The "Skills Area Focus" column makes an attempt to map the four *Key Skill Areas* listed in the previous section.

Table 1: Course Outline

Week	Topic (s)	Skills Area Focus
W1	Lecture: Relationship of Software V&V to software development, Historical Perspective of V&V, S/W Quality, Software Quality Assurance Lab 1-1: Business paper analysis: Kmart Sues i2 over Software. Lab 1-2: Business paper analysis: Bad Software: A Consumer Protection Guide.	R, C
W2	Lecture: Quality Methods in other Industries, Overview of Software Development Lifecycle Models, Formal Proof of Correctness, Clean-room Process for Software Development Lab 2-1: Research paper analysis: Bugs. Lab 2-2: Role Play – Manager Director – Justification for budget	R, C, M

	allocation for V&V activities. Lab 2-3: Exercise on Deming's 14 Points - System of Profound Knowledge (SoPK).	
W3	Lecture: Requirements, Overview of Software Verification Lab 3-1: Article Read & Discuss - Fear of Rejection. Lab 3-2: Article Read & Discuss - Requirements from the Customer Perspective. Lab 3-3: Exercise on Requirements – Error Check, Rewrite. Lab 3-4: Article Read & Discuss - Understanding User Requirements.	R, C, M
W4	Lecture: Verification Activities, Formal Inspection Process. Lab 4-1: Article Read & Discuss - Post Delivery Maintenance. Lab 4-2: Article Exercises on Inspection. Lab 4-3: Exercise on Pair Programming – Driver and Partner Coordination, Cooperation. Lab 4-4: Exercise on Reviews.	R, C, M
W5	Lecture: Applying the Formal Inspection Process Lab 5-1: Understanding the SRS Document – Cafeteria Ordering System. Lab 5-2: DVD- Scenes of Software Inspection.	R, C, M, T
W6	Lecture: Configuration Management, Defect Management Lab 6-1: Team exercise on Configuration Management - Analysis of existing systems and use of SubVersion.	R, T
W7	Lecture: Overview of Software Validation Lab 7-1: Team exercise on Defect Management – understanding the features of Bugzilla.	R, T
W8	Lecture: Overview of Software Validation Lab 8-1: Project related exercise – Requirements Definition.	M
W10	Lecture: Software Testing Lab 10-1: Project related exercise – Requirements Refinement. Lab 10-2: Project Work, Project Progress Presentation.	C, M
W11	Lecture: Software Testing Lab 11-1: Project related exercise – Test Outline. Lab 11-2: Project Work, Project Progress Presentation.	C, M
W12	Lecture: Predictable Software Development Lab 12-1: Project related exercise – Designing Test Cases. Lab 12-2: Project Work, Project Progress Presentation.	C, M
W13	Lecture: Software Reliability Modeling Lab 13-1: Project related exercise – Executing Test Cases. Lab 13-2: Project Work, Project Progress Presentation.	C, M
W14	Lecture: Software Standards Lab 14-1: Project related exercise – Preparing Test Report. Lab 14-2: Project Work, Project Progress Presentation.	C, M
W15	Lecture: Miscellaneous Topics in V&V Project Final Presentation	M

Software Engineering 2004[2] recommends 42 contact hours for V&V Knowledge Area (VAV KA) and 16 contact hours for Software Quality Knowledge Area (QUA KA) for a total of 58 contact hours. This course enhancement makes an attempt to integrate the essential (E) components of both of these knowledge areas resulting in approximately 23 contact hours of lecture and 35 contact hours of lab for a total of 58 contact hours.

4. Course Delivery

Both lecture and lab sessions are delivered in the Software Design Studio. The studio is equipped with a large meeting table at the center and workstations near the walls. This setup enables an emulation of a work environment; both a meeting room and a cubicle pod. The setup also enables student easy access to their workstations.

The studio is equipped with software tools to facilitate the delivery of laboratory sessions. The workstations are installed with Zend Studio (PHP development environment), TortoiseSVN (used with Subversion), HeidiSQL (MySQL client), Apache 2.2 (Web Server), PuTTY (terminal emulator for SSH) and WinSCP (secure file transfer). The studio has a Gentoo Linux Server with SubVersion, SSH (Secure Shell), SFTP (used with WinSCP), Apache 2.2 and MySQL (Open Source Database). Most of these tools are open source or freeware and the remaining are purchased under academic license agreements. Work is in progress to acquire testing software. As more enhancements are implemented additional tools will be added to the studio.

The lectures are delivered as 50 minute sessions on Mondays and Fridays. The laboratory sessions take place on Wednesdays for 2 ½ hours and are used for garnering experience in the four key skill areas listed above:

i. **Communication Skills (C):** All lab sessions begin with a discussion on the accomplishments of the previous lab session and the topics covered during the preceding two lecture sessions. Students are asked to actively participate in collaborative learning for which they are also graded. To emulate a real working environment students participate in role plays. Students are asked to prepare for their roles before participating in role plays. Each role play is analyzed and the dos and the don'ts discussed. During the second half of the term students are assigned an individual project. Each week students are required to make a brief project progress presentation. During the final week the students are required to make a detailed final project presentation. The presentations are orally made using audio video devices. Students are made aware of the importance of professional presentations and are graded in areas like: content (50%), organization/structure (20%), style/presentation/appearance (8%), use of visual aids (10%), audience participation (10%) and adherence to time limit (2%).

ii. **Applied Knowledge of Methods (M):** Laboratory sessions are used to map theory to practice. Case-based video analysis, role-plays (inspection meetings, pair programming), and expert talk sessions enforce further understanding of V&V methods. Videos developed by the Software Engineering Institute (SEI) are used for case based video analysis. Video clips are shown and specific points are discussed. Exercises are conducted to practice formal inspections and pair programming. Three inspection meetings are conducted assigning the students to play different roles (moderator, author, recorder, inspector, etc.). During these meetings students are professionally taken through the entire inspection process. For pair programming exercises students are grouped in two and given coding assignments. Student's take turns being the driver and the observer. Industry partners, alumni and in-house IT professionals deliver expert talk sessions. These talk programs help reinforce concepts discussed during the lab and lecture sessions. Hearing from people “who have been there and done that” makes it easier for student’s to understand V&V in the workplace.

iii. **Applied Knowledge of Tools (T):** The course enhancement focuses on exposing students to commonly used tools for V&V. Tools installed on the studio are selected based on cost and ease of installation and usage. Students make use of SubVersion (freeware) for configuration management exercises. Establishing connections to the server, students learn to check-out and check-in program modules. Bugzilla, a server-based popular industry software, is used for exercises on defect management. Students are taken through the required steps to effectively file, assign, and close a defect.

iv. **Research Capability (R):** Research activities are carried out in three ways. The first activity involves discussions on short case studies. Students first read the case studies and discuss within their teams before participating in the larger class discussions. The second activity involves the study of research papers. Throughout the term the students are given five research papers for independent study and analysis. They are also required to answer five questions per paper and to discuss it in class. The paper analysis accounts for 10% of the total grade. The third activity involves the understanding of standard documents such as the IEEE SE standards for V&V.

5. Outcome Assessment Analysis

For the Faculty Course Analysis Report (FCAR), ABET Derived Outcomes: 1, 2, 3, 4, 5, 6, 7, 8, 9, 11 and ABET Software Engineering Track Outcomes: S1, S2 are assessed after the completion of the course. Student outcome assessments are based on *analysis of examination questions (*EQ*), research paper reviews (*RP*), classroom discussions (*CD*), hands-on exercises (*HE*)* and *project deliverables (*PD*).* Applicable ABET outcomes as well as evaluation tools, assessed tasks and average assessment scores for spring 2006 and Spring 2007 are presented in Table 2.

Table 2: ABET Outcomes, Evaluation Tools and Assessed Tasks

Applicable ABET Outcomes	Evaluation Tools/ Assessed Tasks/ Assessment Scores
Outcome 1: An ability to apply knowledge of mathematics, science and engineering.	Evaluation Tools: PD, EQ Assessed Tasks: Test case design, Calculation of flow chart complexities, RMMM. Assessment: 100% of the class scored 82.5% or more
Outcome 2: An ability to design and conduct experiments, as well as to analyze and interpret data.	Evaluation Tools: PD, EQ, HE Assessed Tasks: Requirements interpretation and refinement, Test case reduction, Test report analysis. Assessment: 100% of the class scored 85% or more
Outcome 3: An ability to design a system, component or process to meet desired needs.	Evaluation Tool: PD Assessed Tasks: Analyze, design and develop test cases. Assessment: 100% of the class scored 80% or more
Outcome 4: An ability to function on multi-disciplinary teams.	Evaluation Tools: CD, HE Assessed Tasks: Collaborative problem solving, Role plays, Inspection meetings, Pair programming. Assessment: 100% of the class scored 80% or more
Outcome 5: An ability to identify, formulate, and solve engineering	Evaluation Tools: PD, EQ, HE Assessed Tasks: Test case outline iterations, Research on

problems.	requirements management and configuration management tools, Analysis of case studies. Assessment: 100% of the class scored 82.5% or more
Outcome 6: An understanding of professional and ethical responsibilities.	Evaluation Tools: CD, HE Assessed Tasks: Collaborative problem solving, Role plays, Inspection meeting, Pair programming. Assessment: 100% of the class scored 80% or more
Outcome 7: An ability to communicate effectively.	Evaluation Tools: CD, PD, HE, RP Assessed Tasks: Class participation, Project report and presentation, Role Plays, Inspection meetings, Pair programming, Research paper analysis. Assessment: 100% of the class scored 80% or more
Outcome 8: The broad education necessary to understand the impact of engineering solutions in a global and societal context.	Evaluation Tools: CD, EQ Assessed Tasks: Discussions on V&V topics of interest. Assessment: 100% of the class scored 87.5% or more
Outcome 9: Recognition of the need for and an ability to engage in life-long learning.	Evaluation Tools: CD, FE Assessed Tasks: Discussions on V&V topics of interest, Answers to questions on life-long learning. Assessment: 100% of the class scored 80% or more
Outcome 11: An ability to use the techniques, skills, and modern engineering tools necessary for engineering practice.	Evaluation Tools: PD, HE Assessed Tasks: Use of bug tracking, configuration management and requirements management tools. Use of standard office tools. Assessment: 100% of the class scored 80% or more
Outcome S1: The ability to analyze, design, verify, validate, implement, apply and maintain software systems.	Evaluation Tools: PD, RP Assessed Tasks: Software testing process, Research paper answers, Analysis of open source V&V tools. Assessment: 100% of the class scored 80% or more
Outcome S2: The ability to appropriately apply discrete mathematics, probability and statistics and relevant topics in computer science and supporting disciplines to complex software systems.	Evaluation Tool: PD Assessed Tasks: Test case development, Project management, Role play on budget justification for V&V. Assessment: 100% of the class scored 80% or more

In addition outcome assessment analysis is also carried out in the following areas:

- **Student Confidence in Methods and Tools:** Throughout the term students are graded on reactive and proactive class participation. In the first case students participate in collaborative learning only when asked to whereas in the second they show enthusiasm to participate. Class discussion takes into account research paper reviews and discussions on methods and tools. Hands-on exercises and project deliverables are used to assess student confidence in methods and tools. At the beginning of the term reactive class participation was the norm. As the term progressed proactive class participation was observed. It was observed that students participated with confidence in the methods they had learnt and the tools they had mastered. For both delivery years (spring 2006 and 2007) similar trends were observed.

- **Exam Grades:** The mid-term exam accounts for 15% and the final exam accounts for 15% of the course grade. Both exams have two sections: a set of multiple choice questions and a set of subjective questions. The exams are closed book for the former and open book for the latter. After the first delivery of the enhanced course (spring 2006), 63% of the students received a grade of A- or higher. After the second delivery (spring 2007), 78% students received a grade of A- or higher. It was observed that majority of the answers were better formulated and justified indicating good understanding of the subject matter. This is also attributable to the fact that the enhanced course had gone through a revision after the first cycle of delivery. Figure 2 depicts student grades for 2006 and 2007.

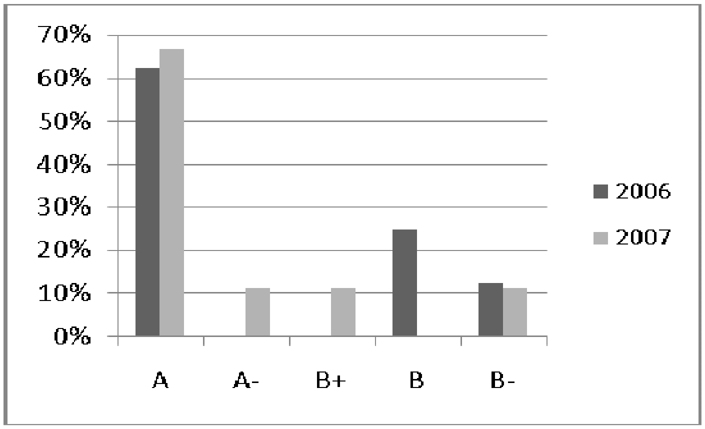

Figure 2: ENGR3400 Student Final grade distribution for Spring 2006 and Spring 2007 Terms

- **Feedback from Employers:** Issues in software development are every developer's nightmare. Customers demand bug free, on-time, and within budget delivery of software. This is only possible through employees that are well versed in V&V. Prospective employers were pleased to know that a course on Software Verification and Validation is offered at Robert Morris University. The course outline was shared with prospective employers who concurred with the contents and the way it was being delivered. After the first delivery (spring 2006), 25% of the students were employed as interns based on them having taken this course. Subsequently these interns were hired as full time employees based on their work performance in large part by the V&V area. After the second delivery spring 2007), 75% of the seniors that were employed as interns were employed based on the V&V course. Companies have been constantly approaching the program for graduating seniors.

  An example feedback from an employer to the question "*During the interview for the internship position was our student able to answer questions related to V&V? In your view did he have adequate knowledge of S/W V&V?*" reads:

  > **Employer >** *Yes. I asked him some fundamental process questions and IV&V basics. He did fine for the internship.*

- **Feedback from Students/Alumni:** Student feedbacks through course assessment and course reflection have been crucial to the ongoing enhancement of this course. End-of-term student course and instruction assessment were conducted using ***Student Instructional Report II*** (SIR II). Table 3 depicts the mean score for selected assessment items for spring 2006 and spring 2007 terms.

Table 3: Selected Assessment Items from SIR II Report

Item	**Spring 2006**	**Spring 2007**	**Comparative Mean for 4 Year Institutions**
Course Organization and Planning	4.31	4.34	4.23
Communication	4.20	4.06	4.28
Faculty/Student Interaction	4.27	4.26	4.27
Effectiveness of Student Assessment Tasks	4.22	3.88	4.02
Supplementary Instructional Methods	Very Effective	Very Effective	NA
Course Outcomes	3.20	3.63	3.65
Overall Evaluation	3.75	3.86	3.99

At the end of course delivery students are asked to reflect upon the course in relation to the text used, course materials provided, projects assigned, quality of lab/lecture delivery, allocation for contents and areas of improvement. For example the following enhancements have been made based on student feedback:

- Using software development projects carried out in a prerequisite course for the project in this course.
- Performing a formal inspection review with assistance from an industry partner.

Occasional feedback like *"Most of my job here is verification and validation just like what you taught me one year ago"* (received from an alumnus) indicate that enhancements are being made in the right direction.

The assessments discussed above are based on an average class size of nine. Due to lab space constraints maximum class size is capped at 10.

6. Conclusions and Recommendations

The Software V&V course was enhanced to provide experience for SE students. The enhancement focused on four key Employment Skill Areas. The course was designed to balance lecture and lab delivery and provides 58 contact hours in V&V and Software Quality knowledge areas. A Software Design Studio was setup with necessary hardware and software to deliver both lecture and lab sessions. Besides the ABET based outcome assessments four additional areas were also analyzed. Completing this course has helped students secure internship and full time employment positions.

The enhancement and delivery made in the past two years has been able to meet the enhancement objectives. Continuing enhancement is important to further meet the expectations of all stakeholders in the software industry. As enhancements are identified and incorporated it is important to understand the important roles of faculty, industry, laboratory and students and how they should be re-strengthened.

- **Role of Faculty:** In this course the faculty is both an instructor and a facilitator. The faculty is the instructor of methods but a facilitator of tools. Faculty should be open to industry feedback and incorporate enhancements as necessary. Ideally the faculty would have real industry exposure as it is easier to draw upon one's own experience. Course delivery should take into consideration real world cases studies, presentations should be professional and tools selected for use should be industry acceptable. However as processes, methods and tools change existing faculty members should also gain knowledge in these changes.

- **Role of Industry:** For SE industry-academia partnership is vital. Industry should provide insightful input and assist by providing tools (hardware and/or software), sharing experiences (through talk programs) and advising the academia when necessary (being an active member of the advisory board). The industry has to understand that their contributions will result in highly qualified graduates whom eventually they may employ. Industry may also benefit by using academic facilities for their employee training in this field. At the moment only a few industry partners have expressed their requirements. More industries should be invited to advise academia fostering a strong sense of partnership between the two.

- **Role of Laboratory:** Physical laboratories are important entities in academia where hands-on experience is imparted. It is important to keep this entity up-to-date. At the moment a Software Design Studio has been setup. However more hardware and software tools need to be installed. More operating system platforms and case-based videos are necessary to strengthen the activities conducted in this studio.

- **Role of Students:** It is important for students to understand that enhancements are being made to provide them a competitive footing in the job market. To benefit from the enhancements students need to become active participants of classroom activities.

References

[1] Computing Curricula 2005, The Overview Report covering undergraduate programs in Computer Engineering, Computer Science, Information Systems, Information Technology, Software Engineering, A volume of the Computing Curricula Series, The Joint Task Force on Computing Curricula, IEEE Computer Society, The Association for Information Systems (AIS), The Association for Computing Machinery, September 30, 2005.

[2] Software Engineering 2004, Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering, A volume of the Computing Curricula Series August 23, 2004, The Joint Task Force on Computing Curricula, IEEE Computer Society, Association for Computing Machinery.

[3] A Bug's Life , Food for Thought: An e-newsletter published by Software Quality Consulting, Inc., November 2005, Vol. 2 No. 10, http://www.swqual.com/newsletter/vol2/no10/vol2no10.html

[4] S. Acharya, V. Harvey, E.G. Holdan, M. Maxwell, D.F. Wood and P.Y. Wu, Discrete mathematics Applications for Information Systems Professionals 2/e, Pearson Publications, 2005, ISBN: 1580532969

[5] Rakitin, S.R., Software Verification and Validation for Practitioners and Managers, 2nd Edition, Artech House, 2001, ISBN: 1580532969