# Teaching Software Testing as a Problem-Based Learning Course

Stephanie Ludi
Software Engineering Department
Rochester Institute of Technology
Rochester, NY 14623
salvse@rit.edu

**Abstract**

Teaching software testing has many facets, including the use of tools and techniques. In addition instructors need to "sell" the need for testing the complex systems that our students will labor on as future software developers. Many perspectives exist in terms of schools of thought when planning for, conducting, and managing the testing activities conducted on a varied array of systems. Instructors wrap these facets of testing into small modules that are integrated into overview courses in Software Engineering or into its own course for either undergraduate or graduate students.

While these facets are not disputed, the mechanism of structuring the course that encompasses that is an issue that should be discussed further. The traditional, lecture-based course format still reins, particularly in undergraduate courses. Some instructors utilize team projects to simulate the "real-world" working environment for course projects. Among the reasons for utilizing the team approach, students can share skills and perspectives with one another. Such collaboration offers students the opportunity to apply course concepts to a project that has meaning to them, thus enabling the student to internalize the concepts. The internalization of such knowledge provides a much stronger bond between the student and the material than mere lecture and recitation can provide. How can this be leveraged further in the teaching of testing?

This paper provides insight into the design and execution of an undergraduate course in Verification and Validation, utilizing a strong problem-based approach that resonates throughout the course. The intent is for students to work through exercises and project "problems" continuously in order to provide a more active learning environment. Another effect of this format is to provide more time for the instructor to interact with the students directly. Thus specific questions can be answered or mini-lectures and discussions can be conducted to address specific issues that have more meaning to students. Students can read definitions and basic concepts on their own. Using class time for more directed and complex activities allows the instructors the time to provide more value-added interaction. In addition to course content, the paper addresses issues regarding student attitude and course expectation. During the pilot offering of the Verification and Validation course student feedback was gathered. The lessons learned by the author, including future directions for the course, will be provided.

## Introduction

Students enrolled in the undergraduate Software Engineering program at the Rochester Institute of Technology are exposed to many facets of Software Engineering. The Software

Engineering program was the first in the country and is conducted under the 10-week quarter system. The Verification and Validation course provides students with the opportunity to learn about the role of verification and validation in the different stages of software development. Techniques are learned, in addition to such topics as planning, tools, and metrics. The course is an overview while also reinforcing links to other courses (e.g. Metrics, Process, Requirements and Specification). As the goal of the program is to enable graduates to be productive members of a development team, each course has been designed to support this overall goal. To facilitate these goals, all courses offered by the Software Engineering project contain a strong team component.

The Verification and Validation (V&V) course is not a required course, but is instead a Process elective whereby students complete 2 of the 3 courses in the set. The other courses are Metrics and Process although seminars are periodically offered which apply to process as well. Students enrolled in the course are typically in their 3rd, 4th, or 5th year in the program. The result is that students have at minimum a foundation is general Software Engineering concepts and Design in different contexts. Traditionally the course was offered in the lecture format, meeting 4 hours per week. The typical class consists of 25 students.

The potential of applying active learning techniques resulted in the new rendition of the course. Specifically, the course reflects Problem-Based Learning (PBL), which has been successfully applied to education in other fields [1, 3, 4]. Rather than the instructor serving as the nexus of learning, the student takes an active role. Specifically PBL uses the solution to problems posed by the instructor to guide student learning. In this arrangement, the students have greater ownership over his/her learning and are more active in the process than watching a PowerPoint presentation. PBL contains attributes that represent its goals. These characteristics include the use of small teams of students working on a set of open-ended problems and the use of reference materials identified by the instructor [2]. The instructor serves as a resource as well. During the course students need to further extend their knowledge and skills in order to complete the problems.

In order to accomplish PBL the notation of the lecture-format plays a small role in the course. The role of the instructor is strong, in terms of course planning and student interaction. The amount of time allocated for student interaction increases dramatically in terms of providing feedback, guidance, and instruction (as needed). Lecture is provided when students ask for clarification or when the instructor observes that students need perspective on a topic or concept at the beginning of a lesson or after an exam (if performance was not satisfactory). Rather than lecturer, the instructor is more of a resource and mentor. The exercises themselves (and the project), along with the planning needed to successfully complete them, are a more realistic dynamic than the traditional lecture format.

**Course Structure**

The previous presentations of the Verification and Validation course were traditional in nature. The class met for 4 hours each week, and the format was lecture-based. The author taught the course for the first time during the Fall 2003 quarter. The successful offering of other Software Engineering course using a problem-based learning format was intriguing [6]. The use of active learning in a Verification and Validation course seemed plausible if the course and materials were carefully designed.

Several weeks were spent designing the overall course structure down to the assignments and readings used therein. The learning objectives were reviewed and organized into groups, which later became learning units. The objectives, based on Bloom's Taxonomy, are provided to the students at both the course level and at the unit level. Within each unit, additional objectives are provided to direct student learning. The learning units and objectives are presented in Table 1.

| Learning Unit | Learning Objectives. After completing the unit, the student will be able to: |
|---|---|
| Unit 1: The Role of Verification & Validation in the SE Process (2 weeks) | • Describe the difference between software verification and software validation.<br>• Use proper terminology in verbal and oral communication regarding the concept and execution of verification and validation.<br>• Describe the software engineering process in terms of the interrelationship of development, test, and support.<br>• Summarize the advantages and disadvantages, with an emphasis on testability, of each of the software development lifecycle models.<br>• Describe the role of risk in approaching verification and validation activities.<br>• Describe the roles and responsibilities people involved in verification and validation.<br>• Prepare an overall test strategy, which reflects the role of risk in the testing process. |
| Unit 2: Types of Testing (3 weeks) | • Distinguish among important test methodologies - black box, white box, and gray box and their applicability to each phase of the development process (e.g. unit test, integration test, system test and acceptance test).<br>• Generate test cases with well-defined inputs, actions, and expected outcomes.<br>• Develop a set of comprehensive equivalence classes for a set of functionality.<br>• Contribute clear test cases to complete the tests needed for a system feature.<br>• Critique test cases and revise them to test the feature(s) more effectively or efficiently.<br>• Describe how to utilize appropriate testing techniques consisting of dynamic and static methods.<br>• Describe the methods of validating each deliverable of the software engineering process: Requirements, Design, Implementation, User Interface, Configuration, and Backward/Forward Compatibility considerations.<br>• Determine the appropriate verification or validation technique applicable to given goals<br>• Analyze the results of the application of a developed test suite. |
| Unit 3: Planning and Measuring for Success (3 weeks) | • Track defects in terms of identifying, classifying by severity and tracking to resolutions.<br>• Describe the role of testing tools in the testing process.<br>• Evaluate and utilize appropriate test tools.<br>• Evaluate the cost/benefit/risk of identifying, resolving, and working around software defects.<br>• Objectively determine the implemented functionality and reliability of an end-to-end software system.<br>• Describe why the investment of testing planning and management are beneficial to software development.<br>• Estimate the resource requirements for each phase of the test cycle.<br>• Formulate test plans and strategies for each component and the full product.<br>• Describe the components of a test plan for a given project. |

| | |
|---|---|
| | • Select test cases and prepare a test plan for a software module/feature.<br>• Analyze measurement and measurement criteria for software<br>• Select appropriate metrics to assess the testing effort of a project.<br>• Compare and contrast available automated support tools and their applications |
| Unit 4: Verification & Validation Perspectives (2 weeks) | • Compare the principles behind test-first development with those of traditional testing.<br>• Describe the process utilized to conduct test-first development.<br>• Discuss the relationship between regression testing and testing for system security and safety. And automated testing and testing for system security and safety.<br>• Given a project scenario, describe the test process (including test cases) needed to effectively verify specified safety and security attributes.<br>• Describe the concept of designing for testability. |

Table 1. Decomposition of course into learning units and associated learning objectives.

Unit 1 is designed as an introductory unit where students learned about the basic concepts of V&V and the relationship it has in Software Engineering. Units 2 and 3 are the bulk of the course. Unit 2 presents the various V&V techniques that students can apply. Unit 3 targets test planning, defect tracking, tools, and measuring the effectiveness of testing in both the product and the process. The goal of unit 4 is to present different views of testing such as test-first development, designing for testability, and testing for security.

Each unit is organized in a consistent manner, with a set of learning objectives, topic and reading outline, guided questions, and unit exercises. These unit components are made available to the students at the beginning of each unit [5]. The learning objectives enable the students to gauge the extent of learning that is expected and to provide a means of presenting what the outcomes of the unit are. An outline is also provided that organizes and breaks down the unit topics into an outline format that corresponds to the unit readings. A sample of a unit outline is in Table 2.

| **Unit 2: Types of Testing** | |
|---|---|
| Testing Type Concepts | 1. Black box, White Box (Patton: pp. 55-56)<br>2. Static Testing, Dynamic Testing (Patton: p. 56) |
| Black box | 1. Static: Examining the Specification (Patton: pp. 57-61)<br>2. Dynamic<br>1. Concept (Patton: pp. 64-68)<br>2. Equivalence Partitioning (Patton: pp. 68-70)<br>3. Data Testing (Patton: pp. 70-80)<br>4. State Testing (Patton: pp. 80-88) |
| White box | 1. Static: Examining the Design and Code (Patton: Ch. 6)<br>2. Dynamic: (Patton: Ch. 7) |
| Integration | 1. Unit Testing (Rakitin: Ch. 9)<br>2. Integration Testing (Rakitin: Ch. 9)<br>3. Validation (Rakitin: Ch. 9 )<br>4. Acceptance Testing (Perry: Ch. 13 )<br>5. Regression Testing (Perry: Ch. 4 --> Functional System Testing Techniques --<br>>Regression Testing Techniques ) |
| Usability Testing | Patton: Ch. 11 |

Table 2. Excerpt from the Unit 2 Topic and Reading Outline.

The guided questions are intended as study aids, whereby students can prepare for the unit exams and the final exam. The unit exercises are a set of exercises that provide an opportunity for the students to apply their new knowledge and skills in a variety of contexts. Some questions require students to conduct additional research while others require concepts to be analyzed and applied to a scenario. In addition, some questions refer to the course project. Sample questions from Unit 2 are presented in Table 3.

| **Unit 2 Questions** |
| --- |
| Create a table of equivalence classes for each of the following problems. You may need to do a bit of research.<br><br>• dialing a phone number from an RIT phone [single-input]<br>• a credit card number (e.g. for an e-commerce site) [dual input: type of card and card number] |
| Describe the role that equivalence classes play in both black box testing and white box testing. |
| Select 2 methods from your project that are non-trivial. For each method, outline the white box test cases required to attain line coverage, branch coverage and condition coverage. In the table listings of the tests cases required, add a column associating the test case with being with line, branch, or condition coverage (it may be associated with more than one type). Since most teams have people from different projects, select one of the projects (or you can do more than 1 project if you wish). |
| You are in charge of managing the testing process for the next version of Microsoft Word for Windows. Describe to what extent each of the following types of testing is important to the product's goals. Also give an example of how you would approach the testing task required for each type.<br><br>a. Configuration testing<br>b. Compatibility testing<br>c. Foreign-language testing<br>d. Usability testing<br>e. Documentation testing<br>f. Installation testing<br><br>Assume the testing goals are:<br><br>• The software will work on computers supported under the system requirements required to run Windows XP in all countries supported by Microsoft.<br>• The documents created/modified by the software can be opened with MS Word 2000 and higher.<br>• The software can open documents created/modified with MS Word 2000 and higher.<br>• The phrases and icons used will have the same meaning in a foreign country as they do in the US.<br>• The data formats will be localized for the target country.<br>• The Microsoft user interface guidelines will be followed.<br>• The software will be accessible to the disabled. |

Table 3. Sample questions from Unit 2 Types of Testing.

The unit problems, outline, and guided questions correspond to the unit outline. The exams reflect the unit objectives and content. As such the exams are meant to assess the individual competence. The unit exercises are submitted at the team-level on the last day allotted to the unit on the course schedule. During the initial offering of the course, the typical format of a unit had the following flow:

- Day 1 of Unit X: Instructor briefly introduces the unit concepts. The unit objectives, outline, guided questions, and exercises are also introduced.
- Day 2 to N-1 of Unit X: For concepts that the students ask for a lecture or when the instructor feels that lecture is needed, a brief 30-40 minute lecture is presented. The remaining time is used to meet with teams as they work on unit exercises.
- Day N of Unit X: Students submit exercises electronically and as a hard copy. The students also have their responses available for reference as a class discussion is conducted regarding the exercise responses and related topics.

Lecture time is significantly reduced, as virtually no class time is spent going over basic concepts and definitions that the students can read about on their own time. Instead lecture is targeted towards specific concepts or topics that go beyond the Knowledge or Comprehension levels of understanding. Whether as a class or within a team (of 3-5 students), students can ask for targeted instruction and feedback. In regard to student-initiated lecture requests, the author's experience in the two course offerings has shown that team-level discussion and supplemental explanation have occurred more than class-level discussion/lecture. Different teams have different strengths and weaknesses and the comfort level is higher at the team level than at the class level.

*Issues*. An issue that was noticeable during the initial offering of the course was in regards to keeping the students on track. When the students are working in teams, it is easy for them to be distracted. To address this issue, the use of unannounced quizzes has been utilized in the second offering of the course. As each unit consists of several readings, the author divides the readings for each unit into reasonable, conceptually grouped "chunks". Each of these "chunks" is listed in the course schedule as a reading assignment for the next class session. This has been quite successful as the students have been on task nearly all of the class time. Another potential solution is to always quiz the students at the beginning of every class session.

Another related issue is providing an environment where students can discuss the exercises in a constructive manner. In the initial offering, many teams split up the questions and then discussed them at the end of the unit in order to consolidate the responses. The result was that the discussion was sometimes rushed, which was not the goal of the course. During the subsequent offering, some students also split up questions but met several times to discuss the answers at the draft stage. Other students simply worked on the questions together from start to finish. The result was deeper discussion and more participation. Such discussion also provided more opportunities for the instructor to offer feedback.

**Course Project**

Rather than devise a course project that the students implement and test, each student team selects a project delivered in a previous Software Engineering course. The intent is for students to assess the quality of a previous project in terms of the product and related artifacts. The project selected does need to be approved by the instructor in order to ensure that the size of the project is appropriate for the team size and that minimal documentation exists to support subsequent assignments. As a minimum a requirements document or statement of needs must be present, in addition to a design document. The students in the team who did not work on the project also need these documents in order to understand the system. The majority of projects

used are either from the Introduction to Software Engineering course or from the Software Architecture course. Examples include a bowling alley management system, a chess game, and a battleship-like game.

The project is used throughout the course, providing students the opportunity to apply different techniques and tools and analyze the results. The assignments include:

1. Ad-hoc testing: Students test the system using ad-hoc methods that are documented. Defects are documented for future reference and analysis. The assignment duration is about 2 weeks.
2. Black box testing: Students test the system using black box testing techniques, including the use of equivalence classes. Test cases and any defects found are documented for future reference and analysis. The assignment duration is about 2 weeks.
3. White box testing: Students test a subset of the system by developing a test suite for the purpose of attaining line, branch, condition and path coverage. The use of test tools has become more pronounced as the course has evolved. Resulting tool reports, code, and defect documentation is provided as well. The assignment duration is about 3 weeks.
4. Fixing Defects: In the first presentation of this course, students were allowed to fix defects within the assignment that the defect was detected in. The current presentation of the course will direct students to fix defects and the relevant artifacts in a more disciplined manner. The assignment duration is about 2 weeks.
5. Comparative analysis: A final presentation outlines the effectiveness of the different techniques applied to the system. The assignment duration is about 1 week.

Overall, the project is well received by the students. The notion of utilizing a previous project is new. Traditionally students submit an assignment and forget about it as soon as the grade is noted or at best the course is completed. The project is successful in several areas:

- Students apply a variety of techniques and analyze the results on an ongoing basis.
- Students apply techniques on a project that is of appropriate size and complexity. The learning curse is low, which allows students to spent more effort on the course.
- Students learn the value of documentation. All too often the task of documenting code and maintaining artifacts is not taken seriously. Many see no value in such an activity or justify not doing so due to the lack of time. During the project, students see the consequences of such negligence.

While some time may be used for the course project, most of the project assignments are completed outside of class.

*Issues*. In order for such a course project to be successful, the students need to have saved their old projects. This has not a given. While students were interested in utilizing their old projects, some students were not able to do so as old projects were deleted upon completion

of the course. While this issue did arise, all teams have been able to locate an old project for use in the course.

Since the projects utilized differ in quality, the student teams encounter different types of defects and different amounts of defects. While experience has shown that students make an effort to find a project that has several defects, some teams have more defects than others. To address this inconsistency, the analysis of results and of the process is emphasized more so than the number of defects found. In addition the use of the final presentation enables the teams to share the different issues and experiences encountered during the course.

The use of such a project does have a distinct limitation. The project is not successful in demonstrating the use of verification and validation through the lifecycle. Since the initial delivery of the project has already transpired, this aspect of the course is lost insofar as having students applying the techniques on a project as it is being developed. To address this, other course assignments have been developed to address this gap.

**Administrative Issues**

As with the other Software Engineering courses at the Rochester Institute of Technology, a significant portion of the overall grade is derived from team projects. The V&V course is not exception, but individual assessment is also a significant portion of a student's overall grade. The overall course grade break down is presented in Table 4.

| Grading Category | Percent of Grade | Work Based on |
|---|---|---|
| Course Project | 25% | ***Team*** |
| Exam 1 | 15% | ***Individual*** |
| Exam 2 | 15% | ***Individual*** |
| Final Exam | 15% | ***Individual*** |
| Unit Questions | 20% | ***Team*** |
| Individual Activities (incl Quizzes) | 10% | ***Individual*** |

Table 4. Course grade components

In order to assess the participation of each team member, all students submit a peer evaluation for each unit exercise set and project assignment. Students who are noted as having done especially superior or unsatisfactory work have the score for the respective exercise set/project assignment adjusted up or down, respectively.

The grading breakdown is one of the many informative sections in the syllabus. The syllabus is discussed with the students on the first day of class as some orientation is needed for the course structure. Some students have completed a similarly designed course, but many have not. Along with a discussion of the course structure and objectives, is a discussion of the course expectations. In order for the course to work, students need to understand their more active role in approaching the course. As such, a list of both student and instructor rights and responsibilities is provided (see Table 5). This approach is less dictatorial and shows the students that they have ownership in their learning and that such learning also involves a partnership between students and the instructor.

| | |
|---|---|
| Student Rights | *You have the right to:*<br><br>• Ask the instructor for clarification of project, activity, quiz and exam instructions and evaluation.<br>• Ask the instructor for clarification or a supplemental explanation of course concepts (e.g. lecture). Some notice may be required for an adequate preparation of the explanation.<br>• Contribute to team and class discussions.<br>• Be treated in a professional manner by your peers. |
| Student Responsibilities | *You have the responsibility to:*<br><br>• Be a self-starter and take ownership of your learning.<br>• Take initiative in the project and other activities, including the maintenance of group dynamics.<br>• Come to class prepared to ask questions, participate in discussions, and participate in activities. This includes conducting any needed reading in the text or other sources.<br>• Act professionally towards your peers. This includes expressing opinions and offering ideas without "putting down" anyone.<br>• Take an active role in your team, in and out of class.<br>• Contribute to the overall team effort as a cooperative team member.<br>• Behave in an academically and intellectually honest manner in class, examinations, and learning activities. |
| Instructor Rights | *The instructor has the right to:*<br><br>• Expect that you will read instructions for project deliverables, activities, and exams.<br>• Expect that you will be prepared for class and any activities and discussions therein (including the project).<br>• Offer individual and team feedback during activities, exams, project deliverables, and other assessment items.<br>• Guide student and team learning by facilitating discussions and activities.<br>• Expect that you will take an active role in your learning, including your participation in your team and roles therein.<br>• Expect that you will treat your peers in a professional manner.<br>• Be treated in a professional manner by the class, and to treat you in a professional manner. |
| Instructor Responsibilities | *The instructor has the responsibility to:*<br><br>• Provide you an environment where you can take an active role in your learning and skill acquisition.<br>• Encourage you to utilize your skills and creativity to shape the project in a manner that will deliver the best solution possible.<br>• Provide fair individual and team feedback in a timely manner.<br>• Work with you and your team to address your questions regarding the project and course material.<br>• Address your concerns in the course, including the course content, project, assessment, and team dynamics. |

Table 5. Student and Instructor Rights and Responsibilities.

*Issues*. The design and execution of a problem-based course is very time consuming. The time to conduct the structure and materials is comparable to that of other courses. The difference is in the time required to grade materials. While students are not continuously

submitting materials for grading, the unit exercise sets are substantial. On average, the each exercise set required about 90 minutes to grade. Also the exercise sets are submitted just before the unit exam is given. A possible solution is to have the students submit the exercises earlier than is currently done. This would also allow for more student feedback before the unit exam.

In addition, the students need to have consistency in the team membership. In the initial offering, the author wanted to have students work in different teams for the different units. The idea was to enable students to work with different people and hear new perspectives. However since the project team needed to stay the same, several issues arose. Students had a difficult time organizing their time both in and outside of class. The stress and dissatisfaction was escalated when other courses using a similar problem-based format also changed the teams. Students had trouble keeping straight what teams they were on and had a difficult time scheduling meetings outside of the course. By Unit 3 and onward through the subsequent offering of the course, student teams remain constant throughout the course.

**Feedback**

Due to the fact that the V&V course has only been offered in the Problem-Based format, no comparison is available to any prior offerings of the course. However, student feedback was gathered from the initial offering. In addition, some students have offered comments during the current offering of the course.

In terms of the structure of the course, the feedback from the initial offering offered some validation of the PBL style. The course structure was evaluated on many levels including the course materials. The most helpful aspects of the course, in terms of enabling students to meet the learning objectives were the unit outlines and the unit exercises. In both areas over 70% of the students rated these course components as being Helpful or Extremely Helpful. The project assignments also received high ratings from the students (62%). Although the course had a small lecture component, the 75% of students rated the amount of lecture to be an appropriate amount of class time. Some students did comment that they learned from lecture, and that it should have a place in the course. In terms of the overall pace of the course, nearly all of the students rated the pace of the course to be just right (as opposed to too fast or too slow). When asked if the V&V course should continue to be offered in the PBL format, 62% of students agreed that the course should continue in the PBL format. These results were promising, and resulted in the continued offering of V&V in the new format.

The feedback collected also supported the instructor's impression that many students were not consistently prepared for class. In the initial course offering, 24% of the students responded that they were prepared for class on a regular basis. This low level of preparation inspired the instructor to revise the pace of the course and use unannounced quizzes to motivate the students to prepare for class. As such, some structural changes were made such as utilizing unannounced quizzes and revising the course resources. In addition, most students acknowledge that exercises were not being completed as a team in many instances. In the subsequent offering of the course, this issue has been addressed for the most part.

The biggest complaint was in regard to the team assignment changes that occurred during the midpoint of the initial course offering. As mentioned previously, the students had an adverse reaction to the communication and scheduling overhead that resulted in not only the V&V course but in the other courses that tried to regroup students as well. The reaction was so strong during

the course that the teams reverted to their original composition and stayed constant throughout the course.

## Future Directions

The next offering of the course will build upon the current offering. Student feedback will be analyzed alongside the instructor's notes. Many of the currently identified revisions reflect issues that have been identified. For example, the need to keep students on task in terms of reading is making progress. The unannounced quizzes have had some success. Conducting a quiz at every class meeting will be piloted.

Another potential means of reinforcing the concepts that the students are learning about in a class session is the use of a 'Problem of the Day.' Students' reading would continue to be paced, but the first hour of the class session would require students (in teams) to research, discuss and devise a response to specified problem. Such an activity would further reinforce new concepts and encourage students to keep on task. Such problems could be in addition to or taken from the Unit exercise set.

The course resources are in continuous flux as new texts and papers are published in the area of V&V. The use of respectable online resources need to be utilized and a short lesson on imparting the need to filter information (particularly that which is online) would be useful as students are bombarded with information of varying quality.

Just as the new PBL offering of V&V allows students to actively participate in the learning process, the course also enables the instructor to more actively engage the students and gauge their progress.

## References

[1] H. S. Barrows and T. R. N, *Problem-based Learning: An Approach to Medical Education*. New York: Springer, 1980.

[2] K. M. Edens, "Preparing Problem Solvers for the 21st Century through Problem-Based Learning," *College Teaching*, vol. 48, pp. 55-68, 2000.

[3] J. J. Kellar, W. Hovey, M. Langerman, S. Howard, L. Simonson, et al., "A Problem Based Learning Approach for Freshman Engineering," ASEE/IEEE Frontiers In Education Conference, Kansas City, 2000.

[4] M. C. LaPlaca, W. C. Newstetter, and A. P. Yoganathan, "Problem- Based Learning in Biomedical Engineering Curricula," ASEE/IEEE Frontiers In Education Conference, Reno, pp. 16-21, 2001.

[5] S. Ludi, SE 452 Verification and Validation unit course materials for Units 1-3, Available at: http://www.se.rit.edu/~sal/se452

[6] J. Valino, "Design Patterns: Evolving from Passive to Active Learning," ASEE/IEEE Frontiers In Education Conference, Boulder, pp. 19-24, 2003.