Applied Knowledge Retention – Are Active Learning Tools the Solution?

Dr. Sushil Acharya, Robert Morris University

Acharya joined Robert Morris University in Spring 2005 after serving 15 years in the Software Industry. His teaching involvement and research interest are in the area of Software Engineering education, Software Verification & Validation, Data Mining, 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 member of Nepal Engineering Association and is also a member of ASEE, and ACM. Acharya was the Principal Investigator of the 2007 HP grant for Higher Education at RMU. In 2013 Acharya received a National Science Foundation (NSF) Grant for developing course materials through an industry-academia partnership in the area of Software Verification and Validation. Acharya is also the Associate Provost for Research, Graduate Study, and International Program.

Dr. Bruce R Maxim, University of Michigan, Dearborn

Bruce R. Maxim has worked as a software engineer, project manager, professor, author, and consultant for more than thirty years. His research interests include software engineering, human computer interaction, game design, social media, artificial intelligence, and computer science education. Dr. Maxim is professor of computer and information science at the University of Michigan—Dearborn. He established the GAME Lab in the College of Engineering and Computer Science. He has published a number of papers on computer algorithm animation, game development, and engineering education. He is coauthor of best-selling introductory computer science and software engineering texts. Dr. Maxim has supervised several hundred industry-based software development projects as part of his work at UM-Dearborn.

Jeffrey J. Yackley, University of Michigan, Dearborn

Jeffrey Jonathan Yackley is a doctoral student in computer information science and a graduate student instructor at the University of Michigan – Dearborn. His research focuses on computer science education and search-based software engineering in the sub-domains of software architecture, refactoring, and testing.

Applied Knowledge Retention – Are Active Learning Tools the Solution?

Abstract

Knowledge retention is a human issue requiring efficient effective teaching strategies to overcome. In today's deliverable driven economy members of the software development workforce are expected to bring in applied knowledge that allows them to perform at high levels from their first day on the job. Software Engineering students often do not have enough hands-on experiences to help them retain knowledge on the key software concepts that will allow them to make timely impact on their first job. Many engineering educators regard experiential learning as the best way to train the next generation of software engineers.

In this paper the authors discuss the effectiveness of Active Learning and describe Active Learning Tools developed to teach Software Engineering content in their Universities. The team created a mix of case studies, role play, trigger videos, prototype creation, and hands on work with software engineering tools and techniques. The investigators conducted several assessments of student learning during the course delivery, surveyed student reactions to the active learning tools, and collected student testimonials. Based on these assessments active learning students seem to improve their problem solving, communication, and team decision making skills during the semester. The authors note that active learning can be achieved by supplementing lecture material with case study discussions, in class exercises, and the use of video case studies.

1. Introduction

In today's competitive job market college graduates need to demonstrate they have the skills to perform from day one to land their first job. While the general belief is that new hires are trained on the job before requiring to perform, employers feel such training will require both time and resources and lean towards hiring experienced professionals to reduce both cost and time. In a report from Training magazine report in 2007, training turns out to be one of the costliest investments a company can make as companies spent an average of over $1,200 annually per employee for 32 hours of training per year (in 2005) [1]. For some companies, especially those noted for their high turnover clearly training cannot be justified as training an employee at $1,500 per year of training can be a major expense if the company's profit per employee is less than $1,500 [2]. To reduce costs Spark, (2018) suggests recruiting only the most skilled employees, retaining them for as long as possible and using performance reviews to identify training needs early on as some of the best methods to control the cost of new hires [3]. Some employers believe it is the worker's responsibility to acquire the skills necessary to do a job before getting hired [2].

In the manufacturing sector it is imperative that the industry work hand-in-hand with academia to properly train and educate both our current and future workforce. In addition to the traditional science-based theory courses in most academic programs, practice-oriented experiences which can and should be included more with the current curriculum [4]. According to CompTIA with employer demand for tech talent routinely outstripping supply, 2018 will force more organizations to rethink their approaches to recruiting, training, and talent management. Additionally, questions surrounding skills gaps (particularly soft skills and team problem solving), diversity, alternative education/career paths, and the future of work will demand more meaningful attention and resources [5].

The understanding in the industry, including software industry, is that new hires need to come with the skills that industry needs as training them is expensive and cannot be justified. To ensure our graduates are competitive in the job market the academia needs to step up and provide them the skills they need in addition to the degree they are awarded. Not doing so can hurt university reputation and affect incoming student pipeline. Such skills should provide real life industry examples, and students should be able to retain them so that they can speak confidently about them and demonstrate them to perform the job they will be hired to do. Knowledge retention is a human issue requiring efficient effective teaching strategies to overcome. In today's deliverable driven economy members of the software development workforce are expected to bring in applied knowledge that allows them to perform at high levels almost immediately. Software Engineering students often do not have enough hands-on experiences to help them retain knowledge on key software concepts to allow them to add value and make timely impact on their first job. Software Engineering programs need to go beyond simply offering industry-based capstone courses and internships.

Engineering educators regard experiential learning as the best way to train the next generation of software engineers [6]. It is the authors' belief that introducing active learning opportunities in Software Engineering courses can help students, bridge the experience gap between study and practice. Active learning is "embodied in a learning environment where the teachers and students are actively engaged with the content through discussions, problem-solving, critical thinking, debate, or a host of other activities that promote interaction among learners, instructors and the material" [7]. Prince [8] defines active learning as a classroom activity that requires students to do something other than listen and take notes. Active learning assists in knowledge retention by immersing students in hands-on activities whose purpose is to reveal the mapping

between theory and practice. Active learning is achievable by complementing lecture materials in the classroom by appropriate active learning tools (ALT).

The knowledge areas listed in the IEEE/ACM (2014) Software Engineering Curriculum Guidelines [9] encompass both theoretical and practical aspects pertinent to Software Engineering. These knowledge areas are essential for undergraduate education and a subsequent professional career in software engineering. Through a project supported by an NSF grant (#1245036) and a University mini grant the authors have developed delivered and disseminated a suite of ALTs in Software Engineering knowledge areas. These ALTs are in the form of case studies, class exercises, and case study videos.

Is section 2 of this paper the authors discuss active learning and its importance in imparting retainable knowledge in the academia. In section 3 the suite of software engineering active learning tools (ALTs) developed in authors' institutions is described. The authors discuss the assessment of ALTs in section 4 and present student testimonials in Section 5.

2. Why Active Learning?

Active learning helps students develop problem-solving, critical-reasoning, and analytical skills, all of which are valuable tools that prepare students to make better decisions, become better students and, ultimately, better employees [8]. Raju and Sankar undertook a study to develop teaching methodologies that could bring real-world issues into engineering classrooms [10]. The results of their research led to recommendations for funding agencies and educators on the importance of developing interdisciplinary technical case studies that allow engineering innovations to be communicated to students in the classroom.

Engineering education must strike a balance between the knowledge of theoretical concepts and the ability to apply the theory to solve real world problems [11]. Effective teaching requires effective teaching tools. Active learning tools complement lectures and make class delivery more interesting to the learners [12, 13]. Students in courses based on active learning techniques show better attendance, a higher sense of competence, and improved commitment to their studies [14]. Students taught using active learning gain confidence in their abilities and perform better on hands on tasks than students taught using lecture only and this learning is reflected in pre-test to post-test performance gains [15]. Active learning can increase student performance in science, engineering, and mathematics [16]. Active leaning is critical in developing the cognitive problem-solving skills used in synthesizing solutions to engineering problems [17]. Project-based learning increases student motivation in upper level courses [18].

Sousa reports that students experiencing lecture only content delivery retain 5% of the material, when discussions are added to lecture presentations students retain 50% of the material covered, and when students teach each other the retention rate can be as high as 90% [19]. Students in active learning courses self-report that they are better able to retain the material covered [14]. Caicedo reports that active learning helps students to understand concepts better than lecture-based instruction [20].

Active learning helps students learn due to increased involvement in the process [21]. Active learning techniques help students to better understand the topics covered in the curriculum [13, 17]. Active learning helps students to be more excited about the study of engineering than traditional instruction [15]. The group work that often accompanies active learning instruction help students develop their soft skills and makes students more willing to meet with instructors outside of class [22]. Krause writes that engagement does not guarantee learning is taking place, but learning can be enhanced if it provides students with opportunities to reflect on their learning

activities [23].

Several best practices have been identified for making using of active learning. Prince reports that students retain knowledge better when lectures are interspersed with brief periods of activity [15]. Simply adding the use of clickers during lecture classes is not perceived as active learning by students [24]. Butler says that using active learning breaks during lectures can help students retain the concepts being studied [25]. Prince [8] also suggests that problem-based learning helps students retain information as well as developing critical thinking skills. Meier [22] suggests that up to 40% of the lecture material in many classes can be replaced by active learning and peer instruction. Active learning is supported by allowing students to schedule and report their progress privately [26]. Goof suggests that having students solve loosely structured problems leads them to inventing novel solutions [13].

Active learning makes use of applied classroom activities that engage students in the process and can be delivered in different formats like using a flipped classroom or peer instruction. Instructors in active classrooms make use of a variety of tools case studies, trigger videos, role play, prototype creation, and small group exercises. The authors' have embedded active learning in their software engineering courses through the use ALTs.

3. Software Engineering ALTs Developed at RMU and UMD

This paper focuses on three categories of ALTs namely *case studies, class exercises,* and *video case studies* that have been used to teach software engineering knowledge areas like Requirements Management, Software Reviews, Configuration Management, and Software Testing. The ALTs have been developed to teach software engineering students at the authors' institutions, Robert Morris University (RMU) and University of Michigan – Dearborn (UMD). Many of the ALTs have also been disseminated to 35 universities/colleges where they may be part of their software engineering curriculum.

- **Case Studies:** Case studies serve as useful tools to teach applications of science and engineering principles and can be used effectively to contextualize theoretical concepts [27]. It has been shown in many studies that benefits of case studies are derived from their interactive learning strategy and the shifting of emphasis from teacher-centered to more student-centered activities [28, 10, 29]. Case studies can result in different student interpretations of the same case. They are similar to open-ended questions.
- **Class Exercises:** Good questions raised in class invites student participation. Class exercises are designed to explicitly facilitate that. Woods and Howard [30] effectively used class exercises for information technology students to study ethical issues. Day and Foley [31] used class time exclusively for exercises, having their students prepare beforehand for class with materials provided online. Frydenberg [32] primarily used hands-on exercises to foster student understanding in data analytics. Class exercises should result in similar student responses.
- **Case Study Videos:** One commonly used technique to enhance the classroom learning experience is the use of video. Videos can reinforce reading and lecture material, help to develop common knowledge, enhance the quality of discussion and overall student comprehension, and accommodate students of different learning styles, increasing student motivation and teacher effectiveness [33]. Videos can aid in illustrating highly complex concepts and ideas in a short period of time, provoking meaningful discussion and analysis. Case study videos are similar to case studies and can result in different student interpretations of the video scenario.

3.1 ALT Development Methodology

Figure 1 depict the iterative development methodology used to design, develop and review ALTs [34]. The methodology ensures the modules reflect both academic research and industry best practices. The content development started with a meeting of the focus groups (comprised of academic and industry partners) at the author's institution. The focus groups drafted an ALT list of active learning content topics and delivery formats. The ALT list was reviewed by the PI and co-PIs and shared with the partners for further review. The finalized list was then used to guide the ALT development process. In this methodology, an industry partner or academic partner led the collaborative effort. Once the contents were ready for review, they were shared with focus group members and subsequently with all project partners. The finalized contents were then transferred to a shareable media where they became available for delivery, further reviews, and dissemination.

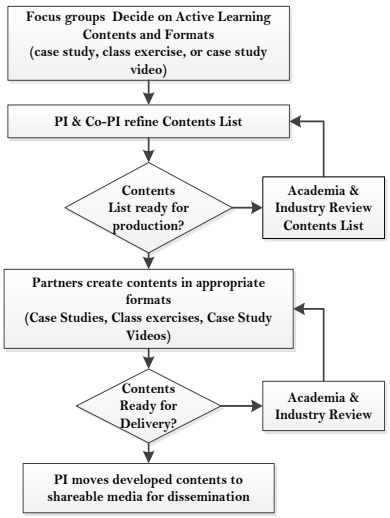

Figure 1: ALT Development Methodology

3.2 Descriptions of Software Engineering ALTs at RMU

44 contact delivery hours of ALTs developed through an NSF grant was developed at RMU through an academia-industry partnership. All of these were delivered in ENGR3400 Software Verification and Validation course over a period of 2 years. Currently a subset of these tools is the core of the ENGR3400 course. Tables 1, 2, & 3 list tools with description, estimated delivery time in minutes, and software engineering knowledge area. At RMU most Software Engineering courses (including ENGR3400) having three credits have 4 contact delivery hours. Two sessions each 2 hours long per week made it convenient to deliver many of the ALTs in one sitting. However, the class size and the length of discussions conducted affected delivery time.

Table 1: Case Studies at RMU

Case Study Modules	Case Study Description	SE Knowledge Areas
ALT: Understanding User Requirements *(50 mins)*	In this case study ■ The students discuss a set of given user requirements in smaller teams (usually of 2 or 3). ■ The students rewrite ambiguous user requirements. ■ The class discusses each requirement and students are asked to read out the rewritten requirements.	Requirements Management
ALT: Requirements from a Customer's Perspective *(250 mins)*	This case study has 5 parts. <u>Part 1</u> ■ Students are told that a brick and mortar store needs a full-fledged web presence. ■ The students create a requirements table for the landing page and a transaction page. ■ Students work in groups of 2 or 3. <u>Part 2</u> ■ Student groups are asked to design the landing page and the transaction page strictly following the table they created in Part 1. Color markers and large post-it sheets are provided and the	Requirements Management

	designs are posted around the classroom walls. Part 3 ■ The requirements tables created in Part 1 are given to student groups other than the owners. The groups are asked to design the landing page and the transaction page strictly following the requirements written on the table. They are asked not to assume missed or ambiguous requirements. Part 4 ■ Owners of the requirements tables are now asked to check if the new designers have incorporated all the requirements they created. The Owners are also asked to judge which design is better. Part 5 ■ The class reflects on the assignment through discussion questions like "why are the two designs different?", "were you tempted to assume missed or ambiguous information? Why?" etc. For homework the class writes a 500 words essay on the knowledge they gained.	
ALT: Continuous Integration *(100 mins)*	In this case study students work in groups of 2 or 3 to research on continuous integration tools. They are asked to define continuous integration, identify criteria to select a specific continuous integration tool, research both commercial and freeware tools and then present their findings to the class. To assist them with criteria the instructor provides them hints like features, cost, after sales service, learning curve, etc.	Configuration Management
ALT: Version Control Management System *(100 mins)*	In this case study students work in groups of 2 or 3 to research on Version Control Management System. They are asked to define version control, identify criteria to select a specific version control tool, research both commercial and freeware tools and then present their findings to the class. To assist them with criteria the instructor provides them hints like features, cost, after sales service, learning curve, etc.	Configuration Management
ALT: Importance of Reviews *(100 mins)*	In this case study students are introduced to a hypothetical software company SoftRight Inc. to understand the following: ▪ How are Inspections different from Walkthroughs? ▪ What are the key attributes of the Inspection process? ▪ Who decides what is to be inspected? ▪ How do you know if you are ready to perform an inspection? ▪ What material is required to conduct an inspection? ▪ How is this material disseminated? ▪ How many working days in advance should the materials be available to the inspectors? Students discuss these questions in groups of 2 or 3. Selected questions are then discussed in class.	Software Reviews
ALT: Peer Review Tools *(100 mins)*	In this case study students work in groups of 2 or 3 to research on Peer Review tools. They are asked to define Peer Review, identify criteria to select a specific Peer Review tool, research both commercial and freeware tools and then present their findings to the class. To assist them with criteria the instructor provides them hints like features, cost, after sales service, learning curve, etc.	Software Reviews

ALT: Test Case Development *(50 mins)*	In this case study students review the given Patient Registration System's SRS for a Hospital Management System in groups of 2 or 3. They then develop 5 test cases for the given functional requirements and identify test cases that can be automated. The developed test cases are first discussed in teams and then discussed in class.	Software Testing
ALT: Performance Testing/ Load Testing *(50 mins)*	In this case study students are asked to differentiate Performance testing from Functional testing? The students then research on Performance Testing tools and identify one that meets the given criteria such as creating baseline, support endurance testing, support spike testing, identify bottlenecks, etc.	Software Testing
ALT: Software Test Plan (STP) *(100 mins)*	In this case study the students review the IEEE 829 Test Plan Outline and Template. The different components of the Test Plan are first discussed in groups of 2 or 3 and then discussed in class. The students then discuss a real-world Software Test Plan (STP)	Software Testing
ALT: Liability for Bad Software and Support *(50 mins)*	In this case study the students read a paper on software liability as homework. First in teams of 2 or 3 and then in class the students discuss questions like: 1. What was the reason for the lawsuit? 2. In your view is the lawsuit justified? 3. How was it settled (if it was) or how would you settle it (if it wasn't)? 4. What were the lessons learnt from this lawsuit?	Additional topics
ALT: Software Legal Issues *(50 mins)*	In this case study the students read a paper on software legal issues as homework. First in teams of 2 or 3 and then in class the students discuss questions like: 1. Did the consumer do the right thing by filing a lawsuit? 2. If you were a member of the software development team would you ethically agree that you are not at fault? 3. Are lawsuits good for practitioners? 4. What were the lessons learnt from this lawsuit?	Additional Topics

Table 2: Class Exercises at RMU

Exercise Modules	Exercise Description	SE Knowledge Areas
ALT: Ambiguous Questions *(25 mins)*	In this exercise students answer a set of 10 ambiguous questions. Specific answers are then discussed in class. Some questions are silly but provide for good discussions. For example, *Where was the Declaration of Independence signed?* The emphasis is on removing ambiguity rather than accepting ambiguity by relating to software requirements and software development.	Requirements Management
ALT: Business Requirements and Functional Requirements *(50 mins)*	In this exercise students are given 10 requirements and are asked to identify which one's are BR (Business Requirements) and which ones are FR (Functional Requirements). The answers are then discussed in class.	Requirements Management
ALT: Clarifying User Requirements *(50 mins)*	In this exercise students identify what is wrong with each of the given SRS (5 sentences) and if they are wrong discuss in class how they may be corrected.	Requirements Management

ALT: Needs Statement to SRS *(50 mins)*	In this exercise students distinguish whether each of the given sentences is a "needs" statement, or a software requirement specification (SRS). If it is a needs statement, the students discuss how they may proceed to re-work the sentence into a SRS.	Requirements Management
ALT: Needs Statements to User Requirements *(50 mins)*	In this exercise students discuss how each of the given "needs" statement may be developed into a Use Case description for analysis and requirements specification. Students respond by saying the needs statement could be represented as a Decision Tree, a Decision Table (truth table) or a Use Case. The answers are then discussed in class.	Requirements Management
ALT: Requirement Ambiguity *(50 mins)*	In this exercise students Identify the ambiguity (or ambiguities) in the given statements. They then discuss and propose how the ambiguity may be corrected.	Requirements Management
ALT: Stated and Implied Requirements *(25 mins)*	In this exercise students read the given stated need and write down implied need(s). The answers are then discussed in class.	Requirements Management
ALT: Defect Lifecycle *(50 mins)*	In this exercise students review the Defect Lifecycle. Each component/status of the cycle is first discussed in groups of 2 or 3 and then discussed in class. Bugzilla is used to assist in the understanding	Configuration Management
ALT: Code Inspection *(150 mins)*	This exercise is on formal code Inspection and has both a homework component and a classwork component. *Homework* Following formal inspection guidelines, students are given the following documents 5 days in advance. 1. Code to be inspected 2. Inspection review sheet 3. Report recording sheet Students review code using review checklist. Students are informed that when the formal inspection takes place, they will play the role of inspectors, but some may have dual roles like reader, facilitator, author, and recorder. *Classwork* In class students are assigned roles. The facilitator leads the process and submits an inspection report.	Software Reviews
ALT: Review a given SRS with Checklist *(100 mins)*	This is a SRS review exercise. The students review a given SRS using a checklist. The checklist is designed for review of an SRS for the purpose of ensuring quality. The checklist is organized into four categories: Organization, Functional Requirements, Quality Requirements, and Completeness. Findings are discussed in class.	Software Reviews
ALT: Cost Effective Testing Approach *(50 mins)*	In this exercise, students are given 5 cases and are asked to discuss each to determine a cost-effective approach to testing the software. The discussion first takes place in groups of 2 or 3 and then the entire class participates.	Software Testing
ALT: Test Cases for a Given Requirement *(50 mins)*	In this exercise, students discuss appropriate Test Cases for the given five requirements. The discussion first takes place in groups of 2 or 3 and then the entire class participates.	Software Testing

ALT: Testing Tools *(50 mins)*	In this exercises students name and describe the kind of testing tools appropriate for the given cases, and discuss how to construct such a tool for testing	Software Testing

Table 3: Video Case Studies at RMU

Video Case Study Modules	Video Case Study Description	SE Knowledge Areas
ALT: Requirements Elicitation *(100 mins)*	The scenes in this video case study portray brief dramatizations in a requirements elicitation process. The 5 scenes present industry best practices and problems that can occur during the process. The scenes also demonstrate appropriate and inappropriate conducts during requirements analysis process ▪ The instructor stops the video after each scene to discuss what just happened. ▪ Sample questions like "Do you think Mike is asking the right questions? "Did the conversation between Yang and Mike happen in a professional manner?" i.e. "were the words used by Yang and Mike appropriate?" ▪ Instructors are also encouraged to come up with their own questions appropriate for the scenes.	Requirements Management
ALT: V&V in Scrum *(50 mins)*	The scenes in this video case study portray brief dramatizations in a Scrum process. The 4 scenes present industry best practices and problems that can occur during the process. The scenes also demonstrate appropriate and inappropriate conducts during Scrum. ▪ The instructor stops the video after each scene to discuss what just happened. ▪ Sample questions like "Why is the Sprint Planning is important?", "Why should the team use Test Driven Development?" are discussed. ▪ Instructors are also encouraged to come up with their own questions appropriate for the scenes.	Requirements Management
ALT: Code Inspection *(100 mins)*	The scenes in this video portray brief dramatizations in a Formal Inspection process. The 7 scenes present industry best practices and problems that can occur during the process. The scenes also demonstrate appropriate and inappropriate conducts during formal inspection process ▪ The instructor stops the video after each scene to discuss what just happened. ▪ Sample questions like "What module is being inspected?" "Is it important to budget the inspection in the plan? Why?" ▪ Instructors are also encouraged to come up with their own questions appropriate for the scenes.	Software Reviews
ALT: Testing and Security *(50 mins)*	The scenes in this video portray brief dramatizations in a Security Inspection case study. The 5 scenes present industry best practices and problems that can occur during the process. The scenes also demonstrate appropriate and inappropriate conducts during formal inspection process ▪ The instructor stops the video after each scene to discuss what just happened. ▪ Sample questions like "Based on the scene, were the developers using proper inspection procedures to check over the code?" "What aspects of the inspection process were not followed in this scenario?" are discussed in class.	Software Testing

	▪ Instructors are also encouraged to come up with their own questions appropriate for the scenes.	

These ALTs are available through the project website www.rmu.edu/nsfvv and ENSEMBLE, a computing portal connecting computing educators, accessible through www.computingportal.org/. The tools and supporting documents are organized based on software engineering knowledge areas. Folders are provided for activities related to requirements management, software reviews, configuration management, and software testing. Underneath each of these folders are folders for the three categories of ALTs: case studies, class exercises, and case study videos. There is also a folder for topical assessments. For greater and easier availability, the videos have been uploaded to YouTube.

3.3 Descriptions of Software Engineering ALTs at UMD

AT UMD two of the authors used many of the ALT's described in section 3.2 as the basis for the activities used in their junior level software engineering course, CIS 375 (Software Engineering 1), offered by the Computer and Information Systems (CIS) department. CIS 375 is organized as a four credit-hour course that meets for 14 weeks. As shown in Table 4 several additional modules were developed to cover all topics offered in CIS 375. This course is required of all computing majors: Computer and Information Systems (CIS), Software Engineering (SE) and Cybersecurity and Information Assurance (CIA) and is taken prior to working on their capstone design projects. The capstone projects completed by UMD students involve working with external clients for eight months as part of a four-person team to develop software solutions to small industrial problems.

The term project for CIS 375 was the creation of a small web-based software engineering tool. Each team created a different tool (e.g., cost estimation tool using use cases or a risk table editor). The students were asked to take an agile approach to the design and development of their tools. A design document and a test plan were developed initially and evolved as the implementation code was created. While teams were allowed to use any agile approach they wished, most teams used a variant of the scrum framework to manage this project. This was not too surprising since the scrum framework was the first agile approach they experienced in detail. The teams did not have a scrum master assigned to assist them. The teaching assistants played the role of customers or product owners.

Table 4: Course Modules

Modules	Description	SE Knowledge Areas
Software Engineering Process Models *(200 mins)*	This unit focuses on four software process models (waterfall, prototyping, spiral, and scrum).	
ALT: Build a Tower	Part 1 After a brief introduction, student groups are asked to build a paper tower using the waterfall process model. ▪ Teams of 4 are given 20 minutes to build the tallest freestanding tower possible using 2 pieces of paper, scissors, and tape. ▪ No materials are provided until design is complete and approved, no testing is allowed until tower is built. ▪ Any design changes may trigger a complete restart.	Process Models

ALT: Land an Airplane	Part 2 Students also complete an exercise, which requires the use of incremental prototypes to complete a mission involving landing paper airplanes on a table. ▪ Teams of 4 are given 8 minutes to design a paper airplane capable of landing on a 4-foot long table. ▪ Teams are required to sketch an initial design but are free to test them and modify them as often as they wish. ▪ During the second round, students have 6 minutes to improve their first design using as many prototypes as they wish. ▪ During the third round, students are given 6 minutes to complete a y new airplane design.	Process Models
ALT: Scrum Processes	Part 3 The class participate in a large group discussion of the video titled "Scenes from Scrum" [35] which focuses on dos and do nots for scrum teams as a way to introduce agile process frameworks.	Process Models
ALT: Play a Card Game	Part 4 The students play a card game [36], which simulates the decision-making processes found in the scrum framework. ▪ Teams of 4 complete 3 simulated 3-day sprints. ▪ Each sprint consists of a planning session, 3 daily standups during the days worked, and sprint review. ▪ Teams use a simplified Kanban chart for planning and tracking purposes. ▪ A burndown chart is used to plot velocity during each sprint.	Process Models
Requirements Modeling (150 mins)	This unit focuses on activities that have teams modeling the requirements for a hypothetical ATM system based on user stories and formal use cases provided to the students. Students are introduced to both CRC (class-responsibility-collaborator) cards and UML as requirements modeling tools.	
ALT: Perform a Grammatical Parse	Part 1 In the first session teams of 4 persons: ▪ Perform a grammatical parse of the user stories to create CRC cards. ▪ Distribute the CRC cards to 3 team members and pass a token to card holders while each user story is read to review them. ▪ Revise the CRC card and create new cards as needed.	Requirements Management
ALT: Prepare a Requirements Document	Part 2 In the second session 4-person student teams are provided with a set of formal use cases for the ATM system, a UML object model for the ATM system, and a Use Case diagram for the ATM systems, teams are asked to create a UML: ▪ Sequence diagram for the Withdraw Cash use case. ▪ Communication diagram for the Withdraw Cash use case. ▪ Activity diagram for the Withdraw Cash use case. ▪ State diagram for a complete ATM session. These activities are used to prepare teams of students to undertake the creation of the requirements document for a small commercial system (e.g. point of sale system (POS) or online music library) using the teaching assistants as simulated customers.	Requirements Management

Project Management (300 mins)	Prior to writing a project management plan for a small commercial system, a set of three activities are introduced.	Management
ALT: Estimate Time & Cost	Part 1 In the first activity 4-person groups estimate the time and cost of building a system based on the formal use cases for the ATM system ▪ Students use a proxy-based procedure to estimate the lines of code needed to estimate the size of the complete ATM system ▪ Students use function point estimation with COCOMO to determine person months required to complete the ATM system ▪ Groups are asked to reconcile their estimates with another group and make adjustments to their estimates	Management
ALT: Create a Schedule	Part 2 In the second activity 4-person groups are given person month estimates for the ATM system and asked to create a schedule that would allow a team of four to complete the systems in four months. Each team is asked to: ▪ List the project deliverables. ▪ Select a process model and determine their project milestones (use of user stories and sprints is suggested). ▪ Assign durations to the completion of each milestone and create and activity graphs for the project. ▪ Determine the critical path through the activity graph. ▪ Create a Gant chart for the project.	Management
ALT: Identify Risks	Part 3 In the third activity 4-person groups are asked to determine the project risks, the technical risks, and the business risks for the ATM system. The groups are asked ▪ To determine the consequences of each risk given a table of operational definitions. ▪ Determine the likelihood of each risk occurring. ▪ Create a risk table. ▪ Create risk information sheet describing monitoring, management, and mitigation procedures for each significant risk. ▪ To present their risks table and one of their risk information sheets to the class as a whole.	Management
Software Design (400 mins)	Four additional active learning modules are used as the students begin the design work for their term project.	
ALT: Assess Architectures	Part 1 Student groups are asked to propose and assess three candidate architectures for the ATM system. They share their tradeoff analyses during a whole class discussion.	Design
ALT: Assess Quality	Part 2 Groups of 4 students use a usability questionnaire to assess quality of the campus homepage and propose suggestions to at least one problem area they uncover.	Design
ALT: Requirements Analysis	Part 3 Student groups develop a set of requirements for the web site for a small clothing store whose owners wished to create a web store. ▪ The student groups trade requirements and create a paper prototype of the web site proposed by another group.	Design

	▪ A representative from another group reviews the paper prototype and requests a new requirement be accommodated in the existing prototype. ▪ The original student groups critique the final paper prototype using their original requirements.	
ALT: Reusability	Part 4 Student groups are asked to create a reusable design pattern based on their analysis of the ATM case study system.	Design

4. Assessment of Active Learning Tools at Implementing Universities

4.1 Implementation of ALT at RMU

One of the authors has been delivering a Software Verification and Validation (SV&V) course since 2005 and is required to perform an ABET Criterion 3 outcomes assessment after every offering. Applicable ABET Criterion 3 Learning Outcomes is listed in Table 5.

Table 5: Applicable ABET Criterion 3 Learning Outcomes for Software V&V course at author's institution

b. An ability to design and conduct experiments, and analyze and interpret data
c. an ability to design a system, component or a process to meet desired needs
e. An ability to identify, formulate, and solve engineering problems
f. An understanding of professional and ethical responsibilities
g. An ability to communicate effectively
h. Broad education necessary to understand the impact of engineering solutions in a global and societal context
i. Recognition of the need for and an ability to engage in life-long learning.
j. A knowledge of contemporary issues
k. An ability to use the techniques, skills, and modern engineering tools necessary for engineering practice

Figure 2 depicts a graphical display of the class assessment performed in Spring 2013 when the ALTs were not incorporated as a pedagogical approach. The Spring 2013 class had seven software engineering junior level students (all males) and all of them were considered for this study. This chart presents percentages of students scoring 80% or better on a variety of assessment tasks. The student performance in each assessment task was measured and regrouped in terms of ABET outcomes to calculate percentage of students that scored within certain levels of assessment vector as detailed in Table 6 given below.

Table 6. Descriptors of ABET Outcomes Assessment Vector

% of students with at least 80% or better score in assessment tasks	**Descriptor of the Resulting Proficiency Status**
90% – 100%	Excellent (E)
80% – 89%	Proficient (P)
70% – 79%	Adequate (A)
60% – 69%	Concern (C)
< 60%	Weakness (W)

It is seen from Figure 2 that there was a weakness associated with learning outcome '***e***' (an ability to identify, formulate, and solve engineering problems), where less than 60% of the students scored better than 80% on the assessment tasks, causing ABET outcome '***e'*** to be identified as a 'weakness'. One of the main reasons for the lower outcome percentage is because the student performance data was obtained through exams, which may not be the best suited tools for assessing outcome '***e***'.

From the 2013 ABET Outcome Assessment report this instructor realized a need for more applied, higher level learning tools. In the 2015 delivery of ENGR3400 ALTs were incorporated as a pedagogical approach and relevant outcomes assessment was performed. The Spring 2015 class had twelve software engineering junior level students (all males) and all of them were considered for this study. This time the student performance data for outcome '***e***' was obtained assessing student performance in ALTs tasks. The results of this evaluation are presented in Figure 3.

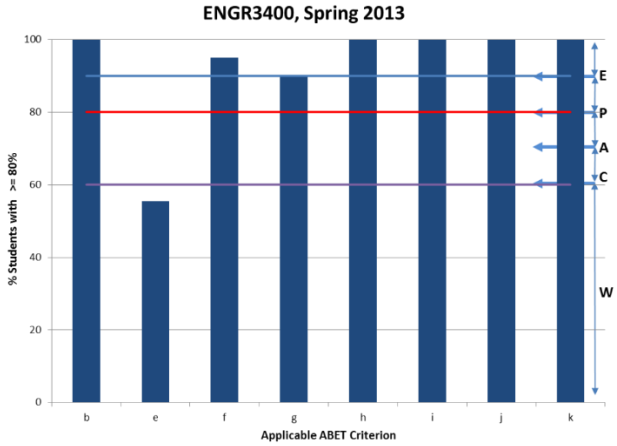

Figure 2: Student outcomes assessment with respect to the specified ABET criteria in Spring '13 term – case studies were not available for this class. (E – Excellent, P – Proficient, A – Adequate, C – Concern and W – Weakness)

It can be seen clearly that the student performance related to outcome '***e***' is now in the excellent range (>=90%) as compared to being an area of concern (<60%) in Spring 2013. This presents clear evidence that the ALT based teaching method is more effective in delivering an ability to identify, formulate, and solve engineering problems to the students.

Figure 4 depicts the assessment of the Spring 2018 class which had twenty-one software engineering junior level students (19 males, 2 females) and all of them were considered for this study. The student performance related to outcome '***e***' (5) continues to be above 80%.

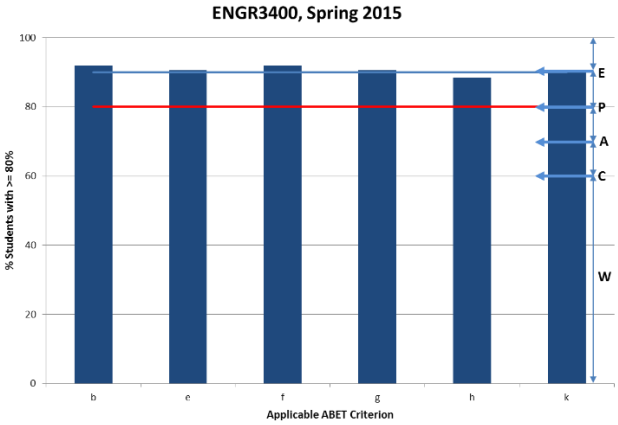

Figure 3: Student outcomes assessment with respect to the specified ABET criteria in Spring '15 term – case studies were delivered in the class.

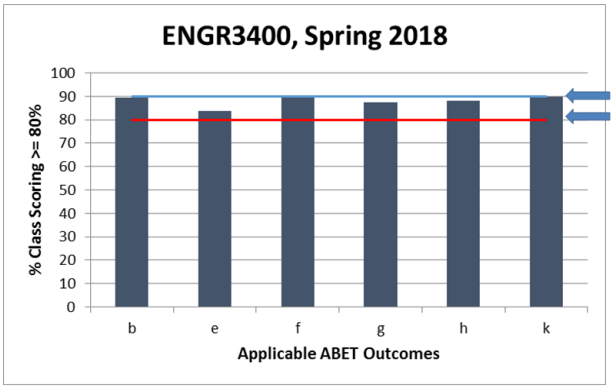

Figure 4: Class performance with respect to ABET outcomes. (The current RMU - designated benchmark for class performance is 80% or B-).

4.2 Implementation of ALT at UMD

Traditionally, students in CIS 375 are required to complete two written examinations and four software engineering documents (software requirements specification, project management plan, software design document, and a test plan). Students also make oral presentations of these documents as they implement a team-based software development project. Each of these assignments is evaluated by rubrics designed by the instructor for each type of submission. Typically, these rubrics contain eight to ten criteria scored 1 to 5 for each. These instruments were used as the primary means of assessing student learning in this course.

No statistical comparisons were made between student performance in the active learning delivery of the CIS 375 and a lecture-based delivery of CIS 375. Three different instructors teach CIS 375 as a single course section three times a year at UMD, which makes statistical comparisons between treatments difficult.

The students provided informal feedback on the active learning modules at the end of each class period. They were asked to reflect on what they liked, what they did not like, and what they learned from the day's activities. In addition, systematic feedback on the class and learning was attained in a formal mid-term and end of term assessments. A mid-term assessment was conducted to evaluate student perceptions of the active learning elements of the class in both the Fall semester 2016 and 2018, which had the same instructor and syllabus. Of the 36 students enrolled in the course during Fall 2016, 34 completed the individual survey and likewise participated in the small group and whole class evaluation while 43 of 49 of the students enrolled in Fall 2018 completed the survey. Table 8 includes survey responses from individual students gathered at the midterm of each class.

As shown in Table 7, most students agreed or strongly agreed that the instructor provided opportunities for active learning and the application of knowledge in class. This was reinforced through open-ended responses on the survey was well. Students in Fall 2016 were asked what things about the class made it easy for them to learn and 28 (82% Fall 2016) mentioned the in-class activities and hands-on learning opportunities provided in the class. This was reiterated in the small group feedback. The groups were in 100% agreement that the instructor provided opportunities to become actively involved and engaged with real-world applications of knowledge. Each of the groups listed the hands-on activities when asked what made it easy for them to learn in class.

Table 7: Student Perceptions of Active Learning: Midterm Fall 2016 and 2018

Survey Statement	Cis 375 Fall 2016 Midterm Survey					CIS 375 Fall 2018 Midterm Survey				
	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
The instructor communicates the course material clearly.		1 (3%)	1 (3%)	18 (53%)	14 (41%)		2 (5%)	9 (21%)	17 (40%)	15 (35%)
The instructor uses effective examples or applications.			5 (15%)	16 (47%)	13 (38%)			3 (7%)	24 (56%)	16 (37%)
The instructor answers questions clearly.			2 (6%)	13 (38%)	20 (59%)	1 (2%)	2 (5%)	6 (14%)	20 (47%)	14 (33%)
The instructor helps the students feel free to ask questions.				9 (26%)	25 (74%)		1 (2%)	2 (5%)	10 (23%)	30 (70%)
The instructor provided students with opportunities to become actively involved in their learning.			1 (3%)	4 (12%)	29 (85%)			5 (12%)	11 (26%)	27 (63%)
The instructor facilitated the application of knowledge through relevant class activities.		1 (3%)	3 (9%)	11 (32%)	19 (56%)			5 (12%)	9 (21%)	29 (67%)

Students were also asked what changes could be made to the course to make learning easier for them in Fall 2016 after the initial redesign of the course to include ALT. On the individual response surveys 23 (68% Fall 2016) mentioned they would appreciate having clearer instructions for the in-class activities and more time to complete them. It seems clear that while the students enjoyed the activities and learned from them, they felt they needed more guidance and time to fully benefit from each ALT experience.

Students were surveyed individually again at the end of the term in both 2016 and 2018. Several of the same questions from the midterm survey were used again in the end of term assessment to see if student opinions changed as they became more experienced with the subject matter. In addition, questions were added to get more specific feedback on the active learning component of the class. There were 27 students who completed the end of term survey in Fall 2016 and 35 students in Fall 2018. Table 8 shows that student opinions about the attempts the instructor made to create active learning opportunities in the class didn't vary from the midpoint of the term to the end of class. Most the class either agreed or strongly agreed with statements pertaining to the instructor's effectiveness in providing active learning opportunities through hands-on class activities. This is consistent with data collected during the midterm survey. In addition, 23 (85% Fall 2016) and 27 (77% Fall 2018) of the respondents felt this class was an effective example of active learning.

Based on the survey responses collected both at the midterm and end of term of both semesters, it appears that students recognized and appreciated the instructor's attempts to create active learning opportunities within the class. Students valued the chances they had to work in small groups and deal with real-world problems. However, these hands-on experiences might have been even more effective if more explicit instruction had been provided and if students were given enough time to fully complete the activities.

Table 8: Student Perceptions of Active Learning: End of Term Fall 2016 and 2018

Survey Statement	Cis 375 Fall 2016 End of Term Survey					CIS 375 Fall 2018 End of Term Survey				
	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
Class activities were a useful way for me to learn.			3 (11%)	14 (52%)	10 (37%)	3 (9%)	2 (6%)	2 (6%)	17 (49%)	11 (31%)
The instructor uses effective examples or applications.		2 (7%)	1 (4%)	7 (26%)	17 (63%)	3 (9%)	1 (3%)	2 (6%)	13 (37%)	16 (46%)
The instructor made clear connections between class activities and the real world.		1 (4%)	1 (4%)	7 (26%)	18 (67%)	3 (9%)	2 (6%)	1 (3%)	8 (23%)	21 (60%)
The in-class activities were designed in a way that allowed me to directly apply what I was learning in class.		1 (4%)	1 (4%)	3 (11%)	22 (81%)	3 (9%)	2 (6%)	3 (9%)	14 (40%)	13 (37%)
The instructor provided students with opportunities to become actively involved in their learning.		1 (4%)	1 (4%)	3 (11%)	22 (81%)	3 (9%)			9 (26%)	23 (66%)
The instructor facilitated the application of knowledge through relevant class activities.		1 (4%)	1 (4%)	12 (44%)	13 (48%)	3 (9%)	1 (3%)	1 (3%)	11 (31%)	19 (54%)
I consider this class to be an effective example of active learning.		2 (7%)	2 (7%)	12 (44%)	11 (41%)	3 (9%)	2 (6%)	3 (9%)	13 (37%)	14 (40%)

Student performance (from CIS 375 Fall semester 2016) on selected activities was tracked in their two-course senior design classes during the two years since they completed the active learning version of CIS 375. Since senior design project teams often included students from both CIS 375 delivery modes any observations on the long-term effects of active learning would be anecdotal at best. It was clear that, as a whole, the active learning students did not receive lower grades than students from lecture-based CIS 375 courses. Figure 5 includes a snapshot of a major assignment in each course and the student outcomes labeled with respect to their status as students who did not take CIS 375 with ALT (labeled Non-ALT) and students who did take CIS 375 with ALT (labeled ALT). The assignments selected for each course, Software Requirements Specification (SRS) from Senior Design 1 and the Software Design Specification (SDS) from Senior Design 2, represent major milestones for student progress in each course respectively and both follow closely from instruction in CIS 375 without regards to the teaching methods (ALT vs. Non-ALT) used in the course. In Figure 5 the vertical axis is the number of students attaining the score shown on the horizontal axis.

5. Student Testimonials

The ALTs have been disseminated to 35 universities/colleges. Student testimonials from 4 universities that have integrated the ALTs in their curriculum is presented below.

5.1 UMD

- **Case Studies:** The student performance on the case studies was best measured in their term project work, especially the software requirements document and the test plan document. The instructor felt their work was superior to what had been observed in the past from students in this class, but no statistical tests were performed. The student comments indicated that they enjoyed the case study activities and felt that they prepared them for work required to create the term project deliverables. They also felt that sharing ideas and insights with other students during class discussions helped them learn. They enjoyed being able to apply the material covered in the textbook to solve typical work place problems.

Figure 5: Student Outcomes for students from ALT and Non-ALT CIS 375 Courses for One, Major Assignment in each Senior Design Course.

- **Class Exercises:** Student performance on class exercises was assessed using two 90-minute written examinations. The class average for the first exam was 83.9% and for the second exam was 83.6%. The students liked the use of multiple diagrams to represent requirements. They also enjoyed writing meaningful questions during requirements engineering activities to help resolve the ambiguities inherent in working with customers. They felt these activities were more engaging than just listening to a lecture accompanied with slides. The students liked the redundancy that was built in the activities that often had them look at different facets of the similar problems. They felt the group work and subsequent presentation summaries help them to improve their communications skills. They enjoyed the group work and loved using the active learning classroom. They liked being able to critique testing artifacts created by others. Occasionally they would have liked more time to complete an activity.

- **Video Case Study:** Video case studies was used as trigger films to provide a context for class discussions. The students appreciated the irony and humor presented in the videos. In many ways, the videos showed why it is important to do things right the first time and the students thought that was good. By the end of the semester, they made many comments about understanding the importance of quality as guiding principle that needs to be pervasive thought a software development organization.

5.2 RMU

- **Case Studies:** Teamwork, discussion participation, presentation, and work products were used to measure student learning. All students liked the hands-on experience, the teamwork and the team presentations. They felt they remained alert and were not distracted. Students felt they communicated well in their team settings and thought critically about content while completing the assignments. Students also felt the presentations of their work to the class gave them a sense of achievement and helped them polish their communication skills. Most important it was understood that answers to questions could vary as software development problems are not like calculus problems i.e. everyone arrives at the same answer. Selected student comments on the case studies were:
    - *It gave real world examples of the legal aspects of the software which gave me a better understanding of the field.*
    - *Was a good introduction for real life application*
    - *The activity allowed my group member and I to share thoughts. It was to the point and educational.*
    - *The scenario was setup in a way that was easy to understand.*
    - *Very creative and effective. It helps to learn more than the old way.*

- **Class Exercises:** Teamwork and discussion participation were used to measure student learning. All students liked the hands-on experience and the work carried out in small teams of 2 or 3. They felt they understood what was expected and made use of lecture slides and past discussions to answer the questions. Given the opportunity to discuss their findings many felt it was good to share their thoughts and hear the thoughts of others. Most important it was understood that answers to exercises had to be similar like calculus problems i.e. everyone arrives at the same answer. Selected student comments on the class exercises were:
    - *It made me feel like I am in the industry. Also, it opens my mind to imagine my future career.*
    - *Small time constraint, couldn't see all major defects*.
    - *This was good instruction into the relationship between developers and customers after deployment.*
    - *Great activity that made me look at different situations from real life.*
    - *I thought it was a good activity to get me to start thinking critically about the subject.*

- **Video Case Studies:** Discussions after each scene were used to measure and understand student learning and perspective. The students felt that scenes and the dramatization of the software engineering process helped them relate to industry best practices. The questions asked, the responses from their classmates, and the thoughts of the professor helped them understand why certain things would happen at work. With the understanding of the “dos” and “don’ts” as depicted in the video students felt confident on being able to execute in the real world should situation arise. Selected student comments on videos shown in class were as following:

- *It did a good job at demonstrating requirement analysis while dealing with a client.*
- *The video was professional a real-life situation. Maybe include more group thinking.*
- *Gives you a good picture of the real-world. Work place is not always friendly. However, you must be a professional.*
- *It was a descent representation of a how a real client interaction takes place.*
- *I thought it was an effective way to facilitate a discussion*

5.3 Embry Riddle Aeronautical University

Five ALTs (3 Case Studies, 2 Class Exercises) were incorporated in SE 420/625 Software Quality Assurance course at Embry Riddle Aeronautical University. Selected student feedback were as following:

- *Very informative and good process to go through. I believe it will be a positive application to a real work scenario.*
- *I understand the importance of meeting customer's requirements. The badly performed software will lead to lawsuit.*
- *The article was difficult to understand (I thought it was poorly written) but otherwise it was good.*
- *This was a good exercise to judge understanding of business requirements and their differences from functional requirements.*
- *I thought this exercise was a bit challenging as there was lots of documentation and picking out the specifics was not straight forward.*
- *Very interesting experience for me. Learned to do research under pressure of teammates.*
- *Learned a lot about different types of testing just from this short activity.*
- *Allowed time constrained research and forced utilization of all available team resources to finish task in given time*
- *The case study was not technical enough.*
- *I was unsure about open source and commercial software. This exercise cleared my understanding.*

5.4 MSOE

One ALT (1 Video Case Study) was incorporated in SE4930 Secure Software Development course at Milwaukee School of Engineering. Select student feedback were as following:

- *The video was well put together.*
- *The video did facilitate small and large group discussions. I thought the acting in the video was actually pretty decent and it portrayed event that could have actually happened.*
- *I Found this actually suitable for this class.*
- *It provided a nice intro in this class.*
- *Cheesy but effective.*

6. Conclusions

Active learning helps students develop problem-solving, critical-reasoning, and analytical skills, all of which are valuable tools that prepare students to make better decisions, become better students and, ultimately, better employees. Software Engineering students often do not have enough hands-on experiences to help them retain knowledge on key software concepts to allow them to add value and make timely impact on their first job. To ensure our graduates are competitive in the job market the academia needs to step up and provide them the skills they need in addition to the degree they are awarded. Active learning assists in knowledge retention by immersing students in hands-on activities whose purpose is to reveal the mapping between

theory and practice. Active learning is achievable by complementing lecture materials in the classroom by appropriate active learning tools (ALT) which could be in the form of case studies, class exercises, and case study videos. ALTs implemented in two universities and disseminated to 35 other universities/colleges have been described and student assessment and testimonials have been presented. ALTs assist software engineering programs go beyond simply offering industry-based capstone courses and internships by providing much needed retainable applied knowledge in classroom setting involving teamwork, communication and research resulting in competitive graduates.

7. Acknowledgement

The authors acknowledge the mini grant awarded by UMD to one of the authors. The authors also acknowledge the support of NSF through a grant entitled "Collaborative Education: Building a Skilled V&V Community", NSF-TUES Award # 1245036 awarded to another author. An author is thankful to his project co-PIs Dr. Priya Manohar and Dr. Peter Wu. The input of industrial partners for this project including Eaton Electrical Corporation, PNC Bank, ANSYS, ServiceLink, and JDA Software Group is gratefully acknowledged. The support of the academic development partners Virginia State University, and Milwaukee School of Engineering is highly appreciated. In addition, input from academic implementation partners including Embry-Riddle Aeronautical University, Montana Technical University, University of Michigan - Dearborn, Fairfield University, Auburn University, East Carolina University, Kennesaw State University (Georgia), Bowie State University, and Clarion University is appreciated.

References

[1]. Mueller, A. (2019) "The Cost of Hiring a New Employee," *Investopedia*, Web Link https://www.investopedia.com/financial-edge/0711/the-cost-of-hiring-a-new-employee.aspx. Retrieved 2/1/2019

[2]. Reh, F.J. (2018) "Is New Employee Training Worth the Investment?," Web Link https://www.thebalancecareers.com/new-employee-training-is-it-worth-the-investment-2275305 . Retrieved 2/1/2019

[3]. Taylor, T.C. (2018) "The Costs of Training New Employees, Including Hidden Expenses," Web Link https://www.adp.com/spark/articles/2018/10/the-costs-of-training-new-employees-including-hidden-expenses.aspx . Retrieved 2/1/2019

[4]. Song, J. (2014) "A New Educational Paradigm to Train Skilled Workers with Real World Practice," Paper presented at 2014 ASEE Annual Conference & Exposition, Indianapolis, IN. https://peer.asee.org/19969

[5]. IT Industry Outlook 2019 (2019) Web Link https://www.comptia.org/resources/it-industry-trends-analysis . Retrieved 2/1/2019

[6]. Promoting Active Learning. Web Link https://utah.instructure.com/courses/148446/pages/active-learning . Retrieved 2/1/2019

[7]. Samavedham, L. & Ragupathi, K., "Facilitating 21st century skills in engineering students," *The Journal of Engineering Education,* Vol. XXVI No. 1, 2012, pp.38-49.

[8]. Prince, M., "Does Active Learning Work? A Review of the Research," *Journal of Engineering Education*, Vol. 93, 2004, pp. 223-231.

[9]. Software Engineering 2014 IEEE/ACM, Software Engineering 2004, *Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering*, a Volume of the Computing Curricula Series, February 23, 2015. Joint Task force on Computing Curricula IEEE Computer Society and Association of Computing Machinery

[10]. Raju, P. K. and Sanker, C. S. (1999): "Teaching Real-World Issues through Case Studies," *Journal of Engineering Education*. Vol. 88 No 4 pp. 501-508.

[11]. Bertha, C. (2010): "How to teach engineering ethics course with case studies," *Proceedings of 2010 ASEE Annual Conference*, Louisville, Kentucky.

[12]. Sivan, A., Wong, L. R., Woon, C. and Kembler, D. (2001): “An implementation of active learning and its effects on the quality of student learning,”’ *Journal of Innovations in Education and Training International*, Vol. 37, No. 4, pp. 381 – 389.
[13]. Goff, R. (2002, June), “The Art Of Creating An Active Learning Environment,” Paper presented at 2002 Annual Conference, Montreal, Canada. https://peer.asee.org/10408
[14]. Thatcher, T. (2007, June), “Incorporating Active Learning into Environmental Engineering,” Paper presented at 2007 Annual Conference & Exposition, Honolulu, Hawaii. https://peer.asee.org/2816
[15]. Acharya, S., & Schilling, W. W. (2012, June), “Effective Active Learning Approaches to Teaching Software Verification,” Paper presented at 2012 ASEE Annual Conference & Exposition, San Antonio, Texas. https://peer.asee.org/21258
[16]. Anthony, G. “Active learning in a constructivist framework,” *Educational Studies in Mathematics* (December 1996) Vol. 31, No. 4: pp. 349-369.
[17]. Wood, K., & Jensen, D., & Dutson, A., & Green, M. (2003, June), “Active Learning Approaches In Engineering Design Courses,” Paper presented at 2003 Annual Conference, Nashville, Tennessee. https://peer.asee.org/11908
[18]. Zhan, W., & Wang, J., & Vanajakumari, M. (2013, June), “High impact activities to improve student learning,” Paper presented at 2013 ASEE Annual Conference & Exposition, Atlanta, Georgia. https://peer.asee.org/19675
[19]. Sousa, Davis (1995) *How the Brain Learns*, The National Association of Secondary School Principals, Reston, VA ISBN 0-88210-301-6
[20]. Caicedo, J. M. (2013, June), “Active Learning Activities in Structural Model Updating,” Paper presented at 2013 ASEE Annual Conference & Exposition, Atlanta, Georgia. https://peer.asee.org/19149
[21]. Nickels, K. M. (2000, June), “Do's And Don'ts Of Introducing Active Learning Techniques,” Paper presented at 2000 Annual Conference, St. Louis, Missouri. https://peer.asee.org/8316
[22]. Meier, R. D. (1999, June), “Active Learning In Large Lectures,” Paper presented at 1999 Annual Conference, Charlotte, North Carolina. https://peer.asee.org/8125
[23]. Krause, Reed; Hayton, Amy C. MD; Wonoprabowo, Jeff; and Loo, Lawrence (2017) "Is Engagement Alone Sufficient to Ensure Active Learning?" *Loma Linda University Student Journal*: Vol. 2: Issue 1, Article 13.
[24]. Catton, K. B., & Galang, A., & Bulk, A. T. (2016, June), “Disruption in Large Classes during Active Learning Sessions,” Paper presented at 2016 ASEE Annual Conference & Exposition, New Orleans, Louisiana. 10.18260/p.26856
[25]. Butler, S. R., & Dee, K. C. (2013, June), “Active Learning Requires Learning - Not Just Activity,” Paper presented at 2013 ASEE Annual Conference & Exposition, Atlanta, Georgia. https://peer.asee.org/19152
[26]. Birmingham, W., & DiStasi, V. (2008, June), “Active Learning Across the Computer Science Curriculum,” Paper presented at 2008 Annual Conference & Exposition, Pittsburgh, Pennsylvania. https://peer.asee.org/3950
[27]. Davis, C., & Wilcock, E. (2003). *Teaching materials using case studies*. UK Center for Materials Education.
[28]. Grant, R. (1997). “A claim for the case method in the teaching of geography,” *Journal of Geography in Higher Education*, *21*(2), 171–185. http://dx.doi.org/10.1080/03098269708725423
[29]. Sivan, A., Wong, L.R., Woon, C., & Kembler, D. (2001). “An implementation of active learning and its effects on the quality of student learning,” *Journal of Innovations in Education and Training International*, *37*(4), 381–389. http://dx.doi.org/10.1080/135580000750052991

[30]. Woods, D., & Howard, E. (2014). “An Active Learning Activity for an IT Ethics Course,” *Information Systems Education Journal,* 12(1) pp.73–77. http://isedj.org/2014- 12/ ISSN: 1545-679X.
[31]. Day, J.A. and Foley, J.D. (2006) “Evaluating a web lecture intervention in a human - computer interaction course,” *IEEE Transactions on Education*, 49(4):420–431, 2006.
[32]. Frydenberg, M. (2013). “Flipping Excel,” *Information Systems Education Journal, 11*(1) pp.63–73. http://isedj.org/2013-11/ ISSN: 1545-679X
[33]. Corporation for Public Broadcasting. (2004). “Television Goes to School: The Impact of Video on Student Learning in Formal Education”.
[34]. Acharya, S., Manohar, P., Wu, P, and Schilling , W. (2017). “Using Academia-Industry Partnerships to Enhance Software Verification & Validation Education via Active Learning Tools”, *Journal of Education and Learning*, Vol. 6, No. 2, ISSN 1927-5250. http://dx.doi.org/10.5539/jel.v6n2p69
[35]. Robert Morris University Software V&V Fundamentals, https://sites.google.com/a/rmu.edu/rmu-nsf/v-v-fundamentals, retrieved February 3, 2017.
[36]. Yevgrashyn, Timofey, SCRUM Card Game, https://scrumcardgame.com/, retrieved January 27, 2018.