Student Engagement in Active Learning Software Engineering Courses

Bruce R. Maxim,
Computer and Information Science
University of Michigan-Dearborn
Dearborn, MI - USA
bmaxim@umich.edu

Adrienne Decker
Engineering Education
University at Buffalo
Buffalo, NY - USA
adrienne@.buffalo.edu

Jeffrey J. Yackley
Computer and Information Science
University of Michigan-Dearborn
Dearborn, MI – USA
jyackley@umich.edu

***Abstract*—Engineering instructors often rely on lectures as their primary mode of instruction even in project courses. In the lecture mode of instruction student engagement with the course material is often low or non-existent until the date of an assessment activity (assignment or exam) is near. In passive learning environments students often do not get many opportunities to develop their soft skills. Many engineering educators regard experiential learning as a better way to train the next generation of software engineers to do project work. This paper describes the authors' experiences introducing active learning opportunities in a junior level software engineering course and a senior level game design course. The materials created for these courses were developed using a variation of the ADDIE (analyze, design, development, implementation, evaluation) process model. The team created a mix of case-study review, role play, trigger videos, and hands-on exercises involving work with software engineering artifacts or tools to facilitate coverage of the topics. The investigators collected observational data on student engagement while they were involved in various class activities and found that they were least engaged when listening to even short lectures. Student feedback and the authors' own lessons learned are being used to plan the next iterations of these courses.**

Keywords—soft skills; active learning; student engagement; software engineering

I. Introduction

Many courses offered by the College of Engineering and Computer Science (CECS) at our university rely heavily on lectures as the primary vehicle of instruction. The authors have noticed gaps in students' software engineering skills when they begin their capstone design projects. It is the authors' belief that part of the reason for these gaps is that students were not asked to apply these skills in project settings in their previous courses. Often, instructors rely on just-in-time learning to fill in the knowledge gaps students may have when they begin their project work in their capstone courses. The purpose of the study reported in this paper is to examine the levels of student engagement during a variety of class activity types.

A. Active Learning

Many engineering educators regard experiential or active learning as the best way to train the next generation of software engineers [1]. Soft skills are increasingly important to the engineering profession and course modifications are often needed to ensure students have opportunities to practice them prior to graduation. The authors believe that introducing active learning opportunities prior to the senior year can improve the students' software engineering and game design skills.

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 and a host of other activities that promote interaction among learners, instructors and the material" [2]. Prince defines active learning as a classroom activity that requires students to do something other than listen and take notes [3]. Active learning tools complement or replace lectures and make class delivery more interesting to the learners.

Specifically, 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 [3]. Raju and Sankar undertook a study to develop teaching methodologies that could bring real-world issues into engineering classrooms [4]. The results of their research led to recommendations to engineering educators on the importance of developing interdisciplinary technical case studies that facilitate the communication of engineering innovations to students in the classroom.

Active learning helps students learn by increasing their involvement in the process [5]. Active learning techniques help students to better understand the topics covered in the curriculum [6]. Active learning helps students to be more excited about the study of engineering than traditional instruction [7]. 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 [8]. 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 [9].

B. Student Engagement

We often like to think of engagement as a student's active participation in our courses. In fact, there are various formal and somewhat conflicting definitions for engagement in the literature, but there is at least some agreement that it is a multidimensional construct [10]. There are also differing

This project was partially supported by the University of Michigan-Dearborn Creative Teaching Fund

numbers of dimensions discussed in the literature, but four common dimensions identified are academic, behavioral, cognitive, and psychological [10]. For purposes of our discussion here, we would like students engaged academically in our courses and exhibiting behaviors during the course that signal engagement.

One method that has been tried to help increase engagement in computing courses is the use of active learning. Some active learning techniques that have shown positive results are use of Think-Pair-Share exercises [11], pair programming [12], peer instruction [13], and active learning classrooms [14]. However, most of these interventions are used at the introductory level, primarily to address some of the issues of large classes [15] and broadening participation [16].

In upper division courses, engagement is still important, but can take on different forms. Ham and Myers brought Process Oriented Guided Inquiry Learning (POGIL) into a computer organization course [17] In software engineering courses, the use of real-world, community-based projects has been recognized as a good way to engage students with a meaningful problem while teaching them software engineering concepts [18]. Students often become more invested in their project when they see that their products are more than simply a paper design.

An important aspect of software engineering education is the development of soft skills such as communication and project management. These skills are hard to practice without a long-term project to manage. There are a number of examples of courses that use their projects to help students with soft skills simultaneous with their software development skills [19]. Some very good on-line simulations have been created in which the computer allows the users to manage simulated projects. These simulations are often structured as role-plays in which users are project managers who are able to see the consequences of their decisions [20, 21].

Decker and Simkins [22] introduced the use of an extended role play approach in a game development processes class where the students were not assessed solely on the artifacts they produced, but rather the processes by which they created the artifacts. Their role-play activities emphasize industry best practices for both technical and soft skills (project management, communication, marketing, and interdisciplinary design). In their course, student teams were discouraged from simply “finishing the product” using a crunch mentality. The goal of their activities was the process itself and student engagement with it.

C. Gamification

Gamified learning or the gamification of learning has been defined as the use of game design elements in non-game settings in order to increase motivation and attention on a task [23]. James Gee has identified thirty-six learning principles that are present in good games [24]. These learning principles provide the backbone for good game design and, in turn, can be used as guiding principles when designing a gamified learning environment. For instance, good games provide players with information when they need it and within the context in which the information will be used [25]. Effective game design includes challenging players so they are routinely working at the edge of their abilities and knowledge, also known as their zone of proximal development [23]. Having students, or players, operate within this optimal learning zone helps keep them engaged and encourages them to learn more in order to meet the demands of the next challenge.

Gee contends that well designed games are motivational specifically because of the different learning principles outlined previously [26]. Working at the limits of their abilities keeps players engaged as they continue to take on new challenges [27]. Gee refers to this process as a cycle of expertise, which requires players to constantly learn, act, revise and learn again in order to demonstrate mastery and be successful in a game [26].

Lee and Hammer suggest that the social and emotional aspects of rewards and consequences earned in gaming environments contribute to student motivation as well [28]. The key is finding a balance between the positive and negative outcomes so that players remain motivated to proceed and don’t become overwhelmed or discouraged [23]. A well-designed game can also motivate players to stay engaged by enhancing the value of the task or tasks being completed [29]. This is particularly beneficial with educational games focused on school related subjects that students might not otherwise choose to immerse themselves within.

Allowing students to negotiate the nature of their activities and rewards up front often goes a long way to ensuring that all students are engaged for the entire semester. A gamification framework was created for a senior level game design course (CIS 487) and was used within the course to allow students to customize their course participation. Gamification was introduced to the junior level software engineering course (CIS 375) to reward differential student contributions of team members to the team project work.

II. Course Sturcture

A. Course Activities

The junior level software engineering course, CIS 375 (Software Engineering 1), offered by the Computer and Information Science (CIS) department meets twice a week for 2 hours each class period for 14 weeks. This course is required of all computing majors (CIS, Software Engineering, Cybersecurity and Information Assurance, Data Science) at the University of Michigan-Dearborn (UMD) prior to working on their capstone design projects. The capstone projects completed by our students involve working with external clients for eight months as part of a four-person team to develop software solutions to small industrial problems.

During the Fall 2018 semester, CIS 375 was taught using a flipped classroom model. Most of the four weekly contact hours were used for engaged/experiential learning. This class covers the full software development life cycle with an emphasis on agile software engineering practices. The students worked on a team-based term project outside of the regular class meeting time. The project required them to propose and create a web-based software engineering tool. A detailed description of the class content and activities appears in Maxim, et al. [30].

Prior to coming to class students were expected to read the sections of the course textbook [31] assigned for class that day. The lecture slides and class handouts were available on UMD's course management system (Canvas). The 2-hour class periods followed the same pattern. The instructor spent the initial 30 minutes introducing the day's topics and activities. During most class periods, students worked in small groups (3 or 4 students) to complete the day's active learning tasks. The learning tasks consisted of team building activities, software artifact construction, discussing video case studies [32] presenting good and bad industry practices, or creating informal presentations summarizing their small group activities. The instructor led the class in a debriefing of the daily activities when time allowed. Students completed a brief feedback survey at the end of each class period.

The purpose of CIS 487 is to introduce students to the technology, science, and art involved in the creation of computer games. The course meets once a week for three hours over a fourteen-week semester. The weekly class session was split in to three components. The first component was a 30-45 minute interactive lecture on the game design material for the week. This presentation was followed by activities designed to engage the students more deeply with the material. Finally, the third component was a 30-minute, live, Unity engine tutorial on a particular topic usually related to the game design content for the week. These live demonstrations provided the opportunity for student interaction and questions not possible when watching video tutorials outside of class. A detailed description of the class content and activities appears in Yackley, et al. [33]. Students completed a brief feedback survey at the end of each class period.

The students completed five projects, four of which were team-based assignments. Gamification was used to reward differential contributions on team projects. The team projects required the delivery of incremental 2D and 3D game prototypes. All assignments contained both required and elective components each with their own point values. Students could select any number of electives from the assignment to complete and attempt to earn the maximum number of assignment points.

B. *Gamification*

One problem the authors have observed on many student team projects was that some students provide little effort toward the final product or they feel their contributions to the final product were not rewarded by their grade. In both classes, students worked in teams to create the milestone documents and prototypes as part of their work outside class. In the past, the course instructor asked each student to grade the participation of each team member (including themselves) using a score of 0 to 5. The average of these scores was added to the team score. The instructor penalized people who failed to make significant contributions. Often the loss of 2 or 3 points on an assignment was not enough to encourage students to be active team contributors.

For the major project portions of each of the two courses, a gamification framework was created, where the points for the team artifacts became part of the core or required work for everyone and the individual work products became part of the elective work. The individual work included peer evaluations of their classmates' work, attendance, programming, testing, project management, and document creation. This framework is described in more detail in Yackley, et al. [33].

The points assigned to the various tasks were mapped to a time card where the maximum points the students earned for their individual prototype work matched the maximum number of points awarded the team documents submitted for that turn in. The students were required to earn at least 10% of their time card points from the activities in the programming category. Pair programming was allowed, but each member of the pair split the points earned for completing a user story. The completed time cards were submitted to the instructor for grading following review and approval by all team members.

In some cases, these activities were further refined. Test engineers were rewarded for writing test cases, executing test cases, and documenting test results. Programmers were not credited with work completed unless a user story satisfied its acceptance criteria. Some tasks, such as management tasks, were better rewarded on an hourly basis. Typically, one point per hour was awarded for these tasks.

The gamification framework for the game design class was implemented using the Gradecraft class management system [34]. This allowed an easy implementation of a leaderboard and provided access to a grade predictor tool. A badge system was also initiated to recognize outstanding achievement in many categories (leadership, game development, marketing, and creative activities). We made use of the time cards in the software engineering class but did not make use of a more formal gamification structure.

III. Evaluation

Student engagement in both classes was measured by observations made every 10 minutes during each class period by counting the number of students who were engaged during the class activity and those who were not engaged. The observers counted students as not engaged if they were using their cell phones, looking at web pages not related to class activities, doing work for other courses, or otherwise not participating in the current activity. The observers considered students who were paying attention to instructions, taking notes, organizing material for the activity, forming groups, discussing the current activity with group members, or looking at related material on their electronic devices as being engaged with that activity. These counts were performed by two graduate student research assistants so that the course instructor could focus on teaching and allowed an accurate count of student engagement to be made at the required time intervals.

A summary of the geometric means of the percentage of students actively engaged in each course is provided in Table I and Table II for each activity category. Course activities were categorized in two major categories: active learning (design, discussions, games, peer-reviews, and worksheets) and non-active learning (watching lectures or videos).

Table I (CIS 375) shows the active learning tasks were superior to lectures at holding students' attention. We note that the games and peer reviews seemed to have the strongest engagement of the active learning tasks. These are activities requiring students to pay attention to be able to provide meaningful feedback to other students.

Table II (CIS 487) shows the biggest differences between active and non-active learning tasks. It also shows large differences among the types of active learning tasks. CIS 487 students we more engaged with game and design activities than discussion activities. Game and design activities typically require greater focus often requiring more synthesis of theory, creativity, and soft-skills. Discussions typically have a lower number of students contributing at any one time leaving open the potential for easy distraction.

TABLE I. CIS 375 ENGAGEMENT DATA

CIS 375 – Software Engineering I – Fall 2018		
Classification	***Class Activity***	***% Average Number of Students Engaged***
Non-Active Learning	Lectures	82.52% (n=43)
	Videos	89.04% (n=43)
Active Learning	Discussions	89.59% (n=43)
	Games	96.99% (n=43)
	Peer-Reviews	92.68% (n=43)
	Worksheets	91.92% (n=43)

TABLE II. CIS 487 ENGAGEMENT DATA

CIS 487 – Game Design and Implementation I – Fall 2018		
Classification	***Class Activity***	***% Average Number of Students Engaged***
Non-Active Learning	Lectures	58.03% (n=30)
Active Learning	Design	91.74% (n=30)
	Discussions	81.72% (n=30)
	Games	97.57% (n=30)

We surveyed the students at the end of the Fall 2018 to check student perceptions of their classroom engagement. The survey findings appear in Tables III and IV. The pattern of student responses in the two classes was similar. Six students took both classes during Fall 2018. Most students indicated their satisfaction and preference for active learning course delivery based on their responses. The data for both classes indicate that students felt more engaged with their course content using active learning methods. They also felt better able to apply their knowledge. The data in Tables III and IV also show that there are some students who do not enjoy active learning or flipped instruction at all.

Informal student comments made during the course suggest that they embraced the use of the time cards as a way to document the individual effort on software projects. It is expected that this time card data may provide a rich set of historic data to help students estimate the effort required to complete future software projects.

IV. FUTURE WORK

A more meaningful process for conducting both formative and summative assessment of both CIS 375 and CIS 487 is planned for their next offerings. We plan to track the performance of these students as they complete their capstone project courses. It seems that the use discussions and trigger videos will need to be rethought if we want to increase student engagement. We may need to explore some type of clicker device to engage students during the short introductory lectures.

TABLE III. STUDENT PERCEPTIONS OF ENGAGEMENT

CIS 375 – Software Engineering I – Fall 2018					
Survey Statement	***Strongly Disagree***	***Disagree***	***Neutral***	***Agree***	***Strongly Agree***
There were opppertunities for me to actively engage in learning	3 (9%)	0	3 (9%)	11 (31%)	18 (51%)
Course activities were useful way to learn	3 (9%)	2 (6%)	2 (6%)	17 (49%)	11 (31%)
Course activites let me apply what I learned	3 (9%)	2 (6%)	3 (9%)	14 (40%)	13 (37%)
Course is an example of active learning	3 (9%)	2 (6%)	3 (9%)	13 (37%)	14 (40%)
I was actively engaged in my learning	3 (9%)	2 (6%)	5 (14%)	13 (37%)	12 (34%)
I applied the course material to real world sittations	3 (9%)	3 (9%)	6 (17%)	12 (34%)	11 (31%)
I felt more engaged during activites than lecture	4 (11%)	2 (6%)	5 (14%)	10 (29%)	14 (40%)
I prefer use of activitieis and discussion to lecture only content	3 (9%)	3 (9%)	5 (14%)	9 (26%)	15 (43%)

TABLE IV. STUDENT PERCEPTIONS OF ENGAGEMENT

CIS 487 – Game Design and Implementation I – Fall 2018					
Survey Statement	***Strongly Disagree***	***Disagree***	***Neutral***	***Agree***	***Strongly Agree***
There were opppertunities for me to actively engage in learning	1 (4%)	0	2 (8%)	10 (42%)	11 (46%)
Coure activities were useful way to learn	1 (4%)	3 (13%)	2 (8%)	7 (29%)	11 (46%)
Course activites let me apply what I learned	3 (13%)	0	4 (17%)	6 (25%)	11 (46%)
Course is an example of active learning	1 (4%)	3 (13%)	4 (17%)	5 (21%)	11 (46%)
I was actively engaged in my learning	1 (4%)	0	5 (21%)	9 (38%)	9 (38%)
I applied the course material to real world sittations	3 (13%)	1 (4%)	9 (38%)	2 (8%)	9 (38%)
I felt more engaged during activites than lectures	0	3 (13%)	2 (8%)	12 (50%)	7 (29%)
I prefer use of activitieis and discussion to lecture only content	0	3 (13%)	3 (13%)	7 (29%)	11 (46%)

References

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

[2] Promoting Active Learning, https://utah.instructure.com/courses/148446/pages/active-learning, retrieved February 25, 2016.

[3] M. Prince. "Does active learning work? A review of the research," Journal of Engineering Education, Vol. 93, 2004, pp. 223-231.

[4] P. K. Raju and C. S. Sanker. "Teaching real-world issues through case studies," Journal of Engineering Education. Vol. 88 No 4 pp501-508.

[5] K. M. Nickels, "Do's and don'ts of introducing active learning techniques," Proceedings of the 2000 Annual Meeting of the American Society for Engineering Education, St. Louis, Missouri, June 2000.

[6] K, Wood, D, Jensen, A. Dutson, and M. Green, M., "Active learning approaches in engineering design courses," Proceedings of the 2003 Annual Meeting of the American Society for Engineering Education, Nashville, Tennessee, June 2003.

[7] S. Acharya, and W. Schilling, "Effective active learning approaches to teaching software verification," Proceedings of the 2012 Annual Meeting of the American Society for Engineering Education, San Antonio, Texas, June 2012.

[8] R. D. Meier, "Active learning in large lectures," Proceedings of the 1999 Annual Meeting of the American Society for Engineering Education, Charlotte, North Carolina, June 1999.

[9] R. Krause, A. C. Hayton, J. Wonoprabowo; and L. Loo, Lawrence, "Is engagement alone sufficient to ensure "active learning"?," Loma Linda University Student Journal, Vol. 2 No. 1, 2017.

[10] J. Appleton, S. Christenson, and M. Furlong, "Student engagement with school: Critical conceptual and methodological issues of the construct," Psychology in the Schools, Vol.. 45, No. 5, 2008, pp. 369 – 386.

[11] A. Kothiyal, R. Majumdar, S. Murthy, and S. Iyer, "Effect of think-pair-share in a large CS1 class: 83% sustained engagement,:, In Proceedings of the ninth annual international ACM conference on International computing education research (ICER '13). ACM, New York, NY, USA, 2013, pp. 137-144. DOI: https://doi.org/10.1145/2493394.2493408

[12] N. Nagappan, L. Williams, M, Ferzli, E., Wiebe, K, Yang, C,arol Miller, and S. Balik, "Improving the CS1 experience with pair programming," In Proceedings of the 34th SIGCSE technical symposium on Computer science education (SIGCSE '03). ACM, New York, NY, USA, 2003, pp. 359-362. DOI: https://doi.org/10.1145/611892.612006

[13] L. Porter, D. Bouvier, Q. Cutts, S. Grissom, C. Lee, R. McCartney, D. Zingaro, and B. Simon, "A multi-institutional study of peer instruction in introductory computing," In Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16). ACM, New York, NY, USA, 2016, pp. 358-363. DOI: https://doi.org/10.1145/2839509.2844642

[14] T. Greer, Q. Hao, M. Jing, and B. Barnes, "On the effects of active learning environments in computing education," In Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19), February 27-March 2, 2019, Minneapolis, MN, USA. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3287324.3287345

[15] M. Minnes, C. Alvarado, and L. Porter, "Lightweight techniques to support students in large classes," In SIGCSE '18: The 49th ACM Technical Symposium on Computer Science Education, Feb. 21–24, 2018, Baltimore, MD, USA. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3159450.3159601

[16] B. Hoffman, R. Morelli, and J. Rosato, "Student engagement is key to broadening participation in CS," In Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19), February 27-March 2, 2019, Minneapolis, MN, USA. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3287324.3287438

[17] Y. Ham and B, Myers, "Supporting guided onquiry with cooperative learning in computer organization," In Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19). ACM, New York, NY, USA, 2019, pp. 273-279.

[18] J.A. Stone, E. Madigan, "Experiences with community-based projects for computing majors," Journal of Computer. Science in the Colleges, Vol. 26, No.6, June, 2011, pp.64-70.

[19] Y. Kharitonova, Y. Luo, and J. Park, "Redesigning a software development course as a preparation for a capstone," An Experience Report. In Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19), February 27-March 2, 2019, Minneapolis, MN, USA. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3287324.3287498

[20] T. Nakamura, H. Maruyama, A. Takashima and Y. Sambe, "Role-play exercises for project management education that incorporate a software agent," Teaching, Assessment and Learning for Engineering (TALE), 2012 IEEE International Conference on, Hong Kong, 2012, pp. W2A-8-W2A-14.

[21] B.R. Maxim, R. Kaur, C. Apzynski, D. Edwards, and E. Evansm "An agile software engineering process improvement game," Proceedings of 46th IEEE Annual Frontiers in Education Conference, Erie, PA, October 2016, pp. S3F1-S3F5.

[22] A. Decker and D. Simkins "Leveraging role play to explore the software and game development process," Proceedings of 46th IEEE Annual Frontiers in Education Conference, Erie, PA, October 2016, pp. S3F6-S3F10.

[23] J. Domínguez Saenz-de-Navarrete, L. de-Marcos, L. Fernández-Sanz, C.A. Pagés, and J.J. Martínez-Herráiz, "Gamifying learning experiences: Practical implications and outcomes," Computers & Education, Vol. 65, 2013, pp. 380–392.

[24] J.P. Gee, What Video Games Have to Teach Us About Learning and Literacy, Second Edition. St. Martin's Press, 2014.

[25] L.S. Vygotsky, Mind and society: The development of higher mental processes, Harvard University Press, 1978.

[26] J.P Gee, "What video games have to teach us about learning and literacy," Computers in Entertainment, Vol. 1, No. 1, October 2003, pp.1-4.

[27] M. Ott and M. Tavella, "A contribution to the understanding of what makes young students genuinely engaged in computer-based learning tasks," Procedia - Social and Behavioral Sciences, Vol. 1 No. 1, 2009, pp. 184–188.

[28] J. J. Lee and J. Hammer, "Gamification in education: what, how, why bother? Definitions and uses," Exchange Organizational Behavior Teaching Journal, Vol. 15 No. 2, 2011, pp.1–5.

[29] Y.T.C Yang, "Building virtual cities, inspiring intelligent citizens: Digital games for developing students' problem solving and learning motivation," Computers and Education, Vol. 59 No. 2, 2012, pp. 365–377.

[30] B. R. Maxim, S. Acharya, S. Brunvand, and M. Kessentini, M. "WIP: Introducing active learning in a software engineering course", Proceedings of the 2017 Annual Meeting of the American Society for Engineering Education, Columbus, OH, June 2017, pp.1-12.

[31] R. S. Pressman, and B. R. Maxim, Software Engineering; A Practitioner's Approach, McGraw-Hill, 2015.

[32] Robert Morris University Software V&V Fundamentals, https://sites.google.com/a/rmu.edu/rmu-nsf/v-v-fundamentals, retrieved February 3, 2017.

[33] J. J. Yackley, B. R. Maxim, A. Decker "Active learning and gamification in game design courses," Proceedings of Meaningful Play 2018 Conference, East Lansing, MI, October 2018, pp. 1-23.

[34] Gradeccraft Home Page, https://www.gradecraft.com/, retrieved April 11, 2017.