Technology Enhanced Active Learning in Software Engineering

Muthu Ramachandran
School of Computing, Creative Technologies, and Engineering, Headingley Campus, Leeds Beckett University, Leeds LS6 3QS, U.K.

Keywords: Software Engineering Management, Active Learning, Group Project, Project Supervision, Collaborative Learning.

Abstract: Educating software engineers in software management have long been hard both in academia and in industry. It is extremely difficult to educate software engineering management techniques actively. Historically, we have been quite used to educate in programming in a classroom and in a lab with instructions Teaching any management aspects has been traditionally based on instructions and case studies. We have adopted an active based learning approach to teach final year BSc students in Software Engineering. We let the final year students manage group projects carried out by level 5 students. Mainly, we don't come across a large real-world case study. This work on active learning has changed our way of teaching software engineering and it has made a significant impact on the way the students learning and have been taught traditionally. This research has also proposed an information system model for Technology Enhanced Active Learning and Teaching (ALT) with emphasis on three key principles for teaching Software Engineering: divergent thinking, collaborative learning, and learning through differentiated assessments. More than 90% of students felt they had gained knowledge more quickly with active learning. The ALT model is part of the large scale technology enhanced learning for future learning environments which has been developed adopting most of computer science courses and specialist module.

1 INTRODUCTION

Software Engineer (SE) education is highly essential in this era of high demand for software systems that has become part of our everyday life. It has been hard to train high level of software engineering concepts, methods, techniques, tools, and managing large scale projects. It is even harder to train software engineering management practices in academia as it is often self-learned in industrial practice or at work place. Historically, we have been quite used to educate in programming and related science. It is extremely difficult to educate software engineering management techniques actively. Mainly we don't come across a large real-world case study. This we have seen being criticized as methods that have not been demonstrated in real work. It is important to educate SE management techniques in this global world with some real practical aspects. Software Engineering itself is increasingly a management discipline of any education and in the real world. Due enhancement and innovation in software technology, we are in a much better position to capture, resolve requirements, intuitively make our design solution with the use of a wide range of design notations and CASE tools, develop code which is much more efficient, and test it all again with a wide range of tools.

One of our main aims of this module (managing software development) is blending of formal academic methods with knowledge of industrial development practices. Consequently, an objective of the course is to provide a balance between the science of management and the practice of management. This emphasis on the 'hands-on' approach to management required a considerable rethinking of this and other modules of the course.

There are several definitions of active learning used worldwide across different disciplines from social sciences to school teaching. *The aims of this approach are to make students learn themselves by doing it rather than only by reading and instructing them in a classroom or in a lab*. There is an immediate change needed in Software Engineering

Ramachandran, M.
Technology Enhanced Active Learning in Software Engineering.
DOI: 10.5220/0006257602420248
In *Proceedings of the 9th International Conference on Computer Supported Education (CSEDU 2017) - Volume 1*, pages 242-248
ISBN: 978-989-758-239-4

Education whether it is taught in the classroom or on online (E-Learning) by applying active and other practical learning types (should not totally be lab based on the other hand as seen in Japan). Offutt (2013) argues that "Software engineering isn't a branch of computer science; it's an engineering discipline relying partly on computer science, in the same way that mechanical engineering relies on physics." Similarly, David Parnas (1999) asserts that process must be shown, not taught, by mentoring young engineers through actual projects. Therefore, this research, have considered more practical and engineering approach to teach Software Engineers in a more active manner with emphasis on professional values and consultancy approach.

IT courses, in particular, Software Engineering courses have seen as difficult to complete in the social inclusion context as well as normal perception in schools. Yet, a shortage for these skills remains high across the globe. One of the main reasons for this situation is the lack of teaching practices and methods adopted in the computing curriculum as we tend to focus more on the new technologies to practice ourselves and quickly convert them into courses. This situation has to change. Therefore, we have adopted an active learning technique of practicing level 6 final year BSc (Hons) students to supervise level 5 (pre-final year/2nd year of BSc (Hons)) group project students in making them to apply the key software engineering principles as well as making them to progress. In this context, this paper is divided into a number sections: section 1 provides an introduction to this work, section 2 provides a brief survey on active learning, section 3 provides the case study of the group project management, and finally section 4 discusses the results and analysis.

2 ACTIVE LEARNING IN SOFTWARE ENGINEERING MANAGEMENT PRACTICE

Active learning helps to facilitate re-enforced learning, participation, and communication by actually doing it rather than just listening. In addition, educating management aspects can't be just done in a classroom setting. As it involves managing and communicating with people. Johnson (1998) explains active learning as how college faculty can use cooperative learning to increase student achievement, create positive relationships among students, and promote healthy student psychological adjustment to college. Johnson (1998) and Walsh and Inala (2010) discuss several approaches to teach librarians with practical examples of active learning style. They have also proposed a number active learning characteristics:

- Students are involved in more than just listening.
- Less emphasis is placed on transmitting information and more on developing students' skills.
- Students are involved in higher order thinking (analysis, synthesis and evaluation). Cognitive and meta-cognitive activities.
- Students are engaged in activities (e.g. reading, discussing and writing).
- Greater expectation is placed on the students' exploration of their attitudes and values.

This highlights the importance of active learning method as opposed to passive learning methods that we are used to in higher education sectors. In addition, in computer science education, Hamada (2007) reports to have successfully adopted a web based active e-learning strategy. "With the advance in applying technology in education, the traditional lecture-driven teaching style is gradually replaced by a more active teaching style where the students play a more active role in the learning process. Hamada (2007) has introduced a set of web-based tools for active (e-)learning in Automata theory and related fields in addition to experimental evaluation of its use in context".

Meyers and Thomas (1993) explain classroom experiences and faculty suggestions in providing a practical guide to teaching strategies to encourage active learning in the college classroom. A wide range of teaching tools which ask students to apply what they are learning are considered, including problem-solving exercises, cooperative student projects, informal group work, simulations, case studies, and role playing.

Silberman (1996) explains in 101 principles of active learning as specific, practical strategies that can be used for almost any subject matters to promote active learning. It brings together in one source a comprehensive collection of instructional strategies, with ways to get students to be active from the beginning through activities that build teamwork and get students thinking about the

subject matter. The 101 strategies are grouped into the following areas: (1) "Introducing Active Learning"; (2) "How To Get Students Active from the Start";

Barnet and Coate (2005) suggest we need change the way we teach and to re-consider our strategies. Furthermore, they argue that "the test of an effective curriculum is 'engaged': Are the students individually engaged? Are they collectively engaged?" Their main argument is that a complex and uncertain world requires curricula in which students as human beings are placed at the centre of quality of learning experience. Brew (2006) discusses the need for initiatives to strengthen the relationship between teaching and research as steps on the path to the development of a new higher education. Using examples, conversations and critical inquiry, it suggests that the establishment of inclusive scholarly knowledge-building communities of both students and staff should result from the development of a stronger relationship between research and teaching.

Offutt (2013) has proposed three principles for teaching Software Engineering: ***divergent thinking, collaborative learning, and learning through differentiated assessments***. Typically, Convergent thinking meaning of the traditional believes that computer science and mathematical problems, in general, **always or most likely** ***have one correct answer*** and successful students should tend toward that answer. Engineering, however, especially software engineering, on the other hand needs *divergent thinking*, where multiple answers are possible and the most successful students should find a solution that's unique when compared with other students solutions (Offutt, 2013). Computer science projects and homework assignments tend to be assessed on a uniform scale that measures every student's work with the same yardstick. But in engineering, especially software engineering, the notion of what will succeed often varies depending on the context, including users, market, platform, and release date. This suggests that we, as educators, should use differentiated assessments.

Therefore, in summary, Offutt (2013) emphasis is on, when we teach software engineering, we must remember that *divergent thinking and collaborative learning are essential abilities for practicing engineers, and differentiated assessment is essential for teaching software engineering*.

The current research study has learned that "the feedback and our interaction with our learners via our teaching has a strong and positive effect on the achievement of our learners" (Hendry et al., 2016; Bonwell, 1991). Other active learning adopted in software engineering courses and in computer science courses include (Boud and Soloman, 2001; Spicer, 1983; Huynth et al., 2016; Krusche et al., 2017; Manohar et al., 2015; Lutz et. al., 2014; Exposito, 2014). However, they mostly mean active learning is a way of interacting with computer based assessment, short cycle exercises, etc. These are one of the traditional forms of interactive learning techniques with the use of a computer. In our work, we use the term active learning to include face to face meeting, interaction and social activities with a group project students, self-learning, etc.

Teaching Software Engineering at Leeds Beckett at the graduate level include taking two semester foundation modules in the first year and moving on to second year where they learn project management, group projects, and software design module. In their final year, they take a two semester module on software engineering with emphasis on software reuse, component-based software engineering (CBSE), software cost estimation models, model-driven development, software process improvement, quality models and testing techniques. As part of their assessment in the final year they are required to manage a second year group project with a size of 5 maximum. The main idea is that students learn more effectively when they are faced with situation to introduce and teach someone else and it is dynamic as they judged based on actively self-engaged as well to make sure the managed group is also actively engaged in their project. This is the rational for adopting active learning techniques into software engineering classrooms. The dynamic nature of the arrangement makes them to collaborate (achieving collaborative learning), socialise, and to think differently (divergent thinking) as they needed to meet outside the normal classroom hours, often encouraged to meet in the social areas such as cafeteria, etc.

This section outlined some of the existing literature on good learning practices. However, it is not clear how we can teach those practices and therefore, we need efficient teaching strategies along with those learning strategies. The following section presents our work on applying some of the learning strategies adopted through active learning methods

as Offutt (2013) and others have not presented how we can teach those techniques in practice.

3 ACTIVE LEARNING & TEACHING MODEL BASED ON SPECIFIC LEARNING OUTCOME (ALT MODEL)

It is important to have an efficient teaching and learning model which should consist of a clear learning objectives, learning outcome, proposed assessment, and clear marking criteria. However, for an active learning technique this can be quite hard, especially during the first time around. Making clear assessment criteria can also be quite difficult as it should be based on observation, feedback from the students themselves to each other, group dynamics, etc. In our approach to active learning is specifically tuned for learning outcome so the effectiveness of the learning method can easily be monitored and can also see how well students have progressed with the work. Our proposed learning model is shown in Figure 1 which consists of a list of learning objectives and expected outcomes and the assessment criteria are largely based around meeting those expected results/achievements by the groups.

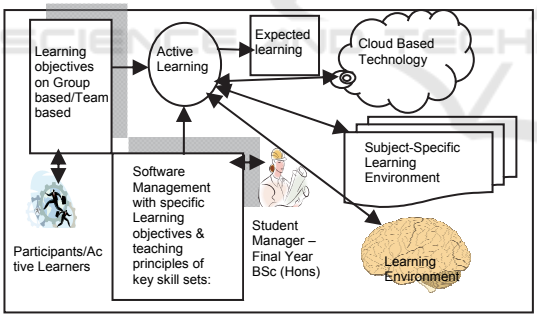

Figure 1: Technology Enhanced Active Learning Environment.

In the final year software engineering module, our course structure has the following learning outcomes:

1. Practice of diverse Requirements Engineering Methods (**Divergent Thinking**)
2. Practice of applying diverse software cost estimation techniques (**Divergent thinking**)
3. Practice of Software Project Management Techniques such as scope management, communication management, project planning and scheduling (**Differential Assessment**)
4. Application of software metrics measuring the productivity of group projects (emphasis is on **Collaborative Learning**)

This model has been implemented in a group project that level 5 students take which was managed by level 6 students. This has been demonstrated in the following section with details of data observed. As discussed, this paper has enforced three types of learning practices that is proposed by Offutt (2013) more efficiently with our approach to ALT model. The divergent thinking was enforced through application of variety of requirements, design, and cost estimation approaches consulted and trained Level 5 students, collaborative learning has been enforced and supervised by making sure smooth and successful delivery of level 5 group project students and their product deliverable in a timely and in a more engineered manner with professional values and dignity of others (assessed through acting as professional consultant) in the group is also monitored. The successful implementation of differential assessment is implemented and monitored through varying degree of assessment mark sheets, feedback from the groups, and interview with consultant engineers with the module tutors.

4 CASE STUDY – GROUP PROJECT MANAGEMENT

The group project module is the appropriate one selected for Software Engineering Consultancy and Management (SECM). The aim of the group project is to develop a web-based video rental agency. This module has its own learning outcome such as to work as a team, attend meetings regularly, record and monitor progress, meet the deadlines, show individual contribution to the group, know your strength and weakness when working as a team. Managing Software Development (MSD) module has its own learning outcomes such as to be able to consult group project, write a reflective report, knowledge transfer skills, group communications, technical consultancy on software engineering methods and management. Our aim is to achieve both, but at the same time encourage active learning. The aim is to marry and synchronise expected learning outcome and deliverable between MSD and

the group project (year 2). The MSD coursework descriptions are:

- To supervise a Level 2 group project (1 or 2 groups will be allocated to each student in ASE module).
- Their role is to act as a project manager to monitor, measure, advice, and measure project metrics and to facilitate communication amongst the group members. This should be about processes and metrics for managing quality in software development, and practicing professional facets of software engineering.
- They are expected to apply methods, tools, techniques, and metrics where possible. Apply knowledge from Managing Software Development module (taught in the autumn term –Semester A) and to use their own experience as a Software Engineer.

The expected outcome is a reflective managerial report consists of assessment of group dynamics, role analysis and allocation, metrics collection, observation, facilitating communication, conflict resolution, and assessing product deliverables such as requirements, project planning, cost estimation, and implementation. The reflective report was marked against the following criteria:

- Log book – activities, issues, solutions. Should reflect phases of development process. Must incorporate application of selected metrics
- Reflective report on what could be done to improve the process of managing projects, ethical issues about conflict between getting it done and quality, cost estimation success, usefulness of selected metrics, post mortem analysis of the project, etc.
- A report consisting of both technical merits that are introduced and management summary
- A range of metrics
- Assessment of the project process
- Evidence of facilitating the Group communication and management
- Range of requirements engineering technique introduced (e.g. use case diagram)
- The cost estimation technique used
- SQA technique used
- Metrics collected (using Together CASE tool as well as ASE module)
- Complexity analysis of the code
- Application of MSD module techniques
- Test plan, Test case design, and test implementation techniques for web based projects
- Feedback form from your group (Hint format)
  - How useful was your participation?
  - Did they learn new techniques?
  - Did they feel their development was speeded up?
  - What did you introduce?
  - What were their problems and issues, and have they been solved?
  - How well you managed communication and management?
  - Will they feel they can work with you on another project?

Each year we are continuing to update on each of the above activities and expected outcome.

5 RESULTS AND ANALYSIS

It is relatively hard to make any clear observation in action research type of project due their nature of personality and communication issues. However, we have made some quite interesting observation. From the past three years of this approach we have identified a number of key outcomes:

- Students manager/consultant need to be assigned to a group by the instructor instead of leaving them to choose their preferred group didn't work well in the first year of this approach
- Step in, when necessary, to resolve conflicts between the consultant and the group early on during their involvement
- Improvise the student manager now and then during their discussion. This was made easy due a specific allocated hour for group projects.
- We found that the 90 % of the student manager expressed this assignment as very interesting and different to their usual assignments.
- More than 90% expressed more than satisfactory to the feedback questionnaire completed by the group project team members.

- More than 60% of the student managers/consultants felt this has increased their learning and deeper understanding of some of the SE issues as it was questioned by group project students
- 2nd year students felt they had learned something additional to their normal way of working on a group project on a software development project and hence speeded up their work. For example, they have learned effort estimation and structuring, questioning and rationalising of requirements earlier on before starting their development.

We are hoping to continue this approach for coming years, but hopefully with more assessment of the group communication and conflict resolution conducted by the student managers. Figure 2 summaries the feedback received on the application of ALT based learning. The group project students (2nd year BSc SE students) expressed overall 90% satisfaction with active learning based approach to their assessment and the help they received from their final year students as compared to just by themselves and the final year students expressed nearly 100% and felt how much they enjoyed in supervising and applying software project management, learned people's management, and have also improved their communication skills.

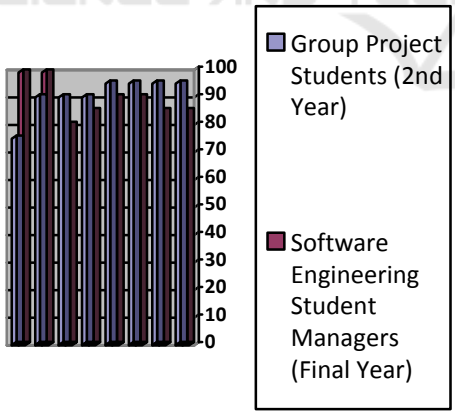

Figure 2: Results and Analysis on SE Practices.

In addition, the final year project managers have also expressed how efficient they had this opportunity to learn by teaching, coaching, and monitoring the group project students in terms of individual topics such as software projects management practice, cost estimations, etc, and as follow:

SPM: Effective Application of Software Project Management Practices (over 90%)
U: Deeper Understanding of SPM Techniques (>85%)
TC: Team Communications & Dealing with Team Issues
CE: Application of SPM Cost Estimation Techniques (FP, COCOMO)
RE&D: Application of Requirements Engineering and Design Methods
Ref: Reflective Report Writing Skills
QM: Application of software quality assurance techniques & software metrics techniques
PM: Improved Presentation & Communication Skills

These are the practices asked in the feedback form for which we have obtained more than 90% satisfaction with effective application of SE practices on SPM, CE, Ref, QM, and PM by group project students as well as the student managers have improved their understanding of the SE practices and their leadership and communication skills. Compared to other approaches in software engineering education in particular they all refer to computer based interaction with some team project. In our project, we used a meaningful active learning that developed group communication and management practices early in their educational career.

6 CONCLUSION

This research on Active Teaching and Learning in Software engineering Education looked at existing approaches to teaching and learning practices in computer science and software engineering. We have adopted our own teaching and learning model based on active learning strategies that are successful in school and social science education. We have also have discovered three key learning approaches have also been successfully adopted: divergent thinking, collaborative learning, and differential assessment. The ALT model has also been evaluated by students and found to be 100% more effective in their learning.

REFERENCES

Offutt, J (2013) Putting the Engineering into Software Engineering Education, IEEE Software, Jan-Feb 2013, 30(1):96

Parnas, D (1999) “Software Engineering Programs Are Not Computer Science Programs,” IEEE Software, vol. 16, no. 6, 1999, pp. 19–30.

Johnson, D. et al (1998) Active Learning: Cooperation in the College Classroom, Interaction Book Company, USA

Walsh, A and Inala, P (2010) Active Learning Techniques for Librarians: Practical examples, Chandos Publishing

Hamada, M (2007) Web-based Active e-Learning Tools for Automata Theory, Seventh IEEE International Conference on Advanced Learning Technologies (ICALT 2007)

Meyers, Chet Jones, Thomas B (1993). Promoting Active Learning. Strategies for the College Classroom, Jossey-Bass Inc Publishers, USA

Silberman, L (1996) Active Learning: 101 Strategies To Teach Any Subject, Prentice Hall

Barnett, R. and Coate, K. (2005) Engaging the curriculum in higher education. Maidenhead: Open University Press and McGraw Hill.

Brew, A. (2006) Research and teaching: beyond the divide. London: Palgrave, Macmillan.

Hendry, G.D, White, P, and Herbert (2016) Providing exemplar-based ‘feedforward’ before an assessment: The role of teacher explanation, Active Learning in Higher Education Journal (SAGE Publication) July 2016 17: 99-109, first published on March 18, 2016 doi:10.1177/1469787416637479

Bonwell, C (1991) Active Learning: Creating Excitement in the Classroom, www.active-learning-site.com

Boud, D, Ed. and Soloman, N. Ed (2001) Work-Based Learning: A New Higher Education?, Taylor & Francis Inc, USA.

Spicer, C. J (1983) A Spiral Approach to Software Engineering Project Management Education, ACM SIGSOFT SOFTWARE ENGINEERING NOTES Vol 8 No 3 Jul 1983 Page 30

Huynh, Trongnghia; Hou, Gene; Wang, Jin (2016) Communicating Wave Energy: An Active Learning Experience for Students, American Journal of Engineering Education, v7 n1 p37-46 Jun 2016. 10 pp.

Krusche, S. et al. (2017) Interactive Learning: Increasing Student Participation through Shorter Exercise Cycles, The proceeding of the 2016 Australasian Computing Education Conference, ACM Digital Library 2017.

Manohar, P. A., et al. (2015) Case Studies for Enhancing Student Engagement and Active Learning in Software V&V Education, Journal of Education and Learning, v4 n4 p39-52 2015. 13 pp.

Lutz, Michael J., Naveda, J. F., and Vallino, James R (2014) Undergraduate Software Engineering, Communications of the ACM. Aug2014, Vol. 57 Issue 8, p52-58. 7p.

Exposito, E (2014) yPBL: An Active, Collaborative and Project-Based Learning Methodology in the Domain of Software Engineering, Journal of Integrated Design & Process Science. 2014, Vol. 18 Issue 2, p77-95. 19p