Software Development as Service to the Student Community: An Experiential and High Student Involvement Approach to Software Engineering Education

John C. Georgas
Department of Electrical Engineering and Computer Science
Northern Arizona University
John.Georgas@nau.edu

Abstract

While a common expression of experiential learning in software engineering is the industry-sponsored project, it suffers from key shortcomings at the introductory level. These center on the fact that projects are neither drawn from nor relevant to the everyday lives and communities of their student-developers. To address these challenges, we focus on casting project development in terms of service to the student community, with the involvement of our university's ACM Student Chapter. We discuss the pedagogical foundations of our work, present specific course organization issues, expand on a number of specific challenges with our approach, offer concrete project examples, and outline future work and evaluation.

1. Introduction

As with many other engineering disciplines, software engineering education has been significantly influenced by the experiential and situated learning paradigms, which focus on contextualizing educational activities within settings that are a close match to those in which learners will actually apply the knowledge they gain. A very common application of these paradigms in teaching software engineering is to have students work on real-world software development projects in collaboration with industry partners.

While this approach is most commonly found in final-term capstone projects, it is also applied in introductory software engineering courses. Within this introductory context, however, the industry-partnered approach suffers from a number of shortcomings. Since students at this level lack the context to fully appreciate the learning experiences involved, they often focus on the nature of the industry-sponsored projects themselves. However, these projects are not directly tied to their everyday activities and community, results and benefits from the software systems they develop are reaped by the industry customer and not the students themselves, and the involvement of students with these projects ends the moment their final deliverable is produced. We have found that these challenges mean that the intended increased level of student involvement does not actually materialize.

In order to address these shortcomings, we re-designed the project component of our introductory software engineering course with a focus on increasing the relevance of the projects used to the everyday lives of students, and therefore increasing the degree of interest and investment students exhibit. The core insight is to cast the software development activities of students as *service* to the student community itself: By structuring our course so that it involves the solicitation of projects drawn from the needs of the everyday operations of the Northern Arizona University (NAU) ACM Student Chapter and using its members as student-clients, we aim to give student-developers the opportunity to work on projects that have potentially long-term impacts on their community.

While we are working on the repeated application and further evaluation of our approach, our work to date contributes a conceptualization of the difficulties in applying experiential

learning at the introductory software engineering level, a substantiation of these difficulties based on learning theory foundations, an outline of the process through which we solicited community service-oriented projects, and a set of specific example projects we have used.

2. Learning Theory Foundations

Our overall approach is strongly informed by insights from the *learn by doing* philosophy [2], which stresses the centrality of the social and interactive aspects of learning. Building on this philosophy, *situated learning* [7] espouses a focus on learning activities taking place in a context that is similar to that in which the knowledge will be eventually applied. Another well-known expression of the learn by doing philosophy is *experiential learning* [6], which emphasizes the grounding of learning activities in problems that are fundamentally similar to tasks which students will encounter outside the educational setting. An application of these principles is *problem-based* [9] learning, which stresses activities that are open-ended and encourage self-directed learning. By focusing on authentic tasks – such as the projects drawn from the student community that are the focus of our work – this problem-based approach encourages learners to take a greater degree of personal ownership of their own learning activities, which in turn is intended to result in greater investment and quality of work.

Our work is also significantly influenced by findings related to motivating students, a concern separate from specific educational methods adopted. The ARCS motivational model [4] outlines four key factors in maintaining a high degree of learner motivation: *Attention* (supported by appealing to learners' curiosity and problem-solving abilities), *relevance* (promoted by relating tasks to learner interests), *confidence* (fostered by ensuring that learners have a realistic expectation of at least some success), and *satisfaction* (supported by showing learners the value of learning outcomes achieved). Our work is intended to foster a higher degree of motivation in order to further increase student involvement and commitment into their development projects by focusing on projects that would have an impact in their everyday lives, which is a critical factor in promoting learner success [1]. Finally, in the categorization found in [8], our work falls under the "real project/customer" grouping, primarily supporting learn by doing and situated learning.

3. Software Development as Service: Course Design

Rather than adopting projects sponsored by industry, we instead propose a course structure that involves the direct solicitation of projects from the student community itself. By doing so, we cast the development efforts of students as *service* to their own student community: Student work becomes focused on projects that are relevant to their lives as students, have inherent value to their community, and entail a long-term engagement with their own products by virtue of their products' adoption by the central social entity of their community.

Our work involves a re-design of our introductory software engineering course, which is focused on basic concepts and techniques, such as processes, requirements, architecture, implementation methodologies and testing, while also incorporating cross-cutting concerns such as configuration management tools and processes. The course also serves as the first curricular point where students are exposed to team-based learning in the form of a semester-long project, enabled by the fact that the course rarely exceeds 30 students in size.

3.1. Pedagogical Challenges

The situated and experiential learning models have fundamentally influenced modern software engineering education: It is very common for curricula to include a substantial, hands-on and team-based software development project in the senior year, while a more modest project frequently grounds introductory software engineering courses. A popular approach is to collaborate with industry partners that provide problems from their respective

domains as the basis for these projects. The fundamental premise is that the authenticity of these projects motivates students to produce higher-quality deliverables [3] than they otherwise would. These same industry partners act as clients and have frequent interactions with students, particularly during requirements elicitation.

While the inclusion of authentic, industry-sponsored projects is extremely valuable, a key consideration is student maturity: More knowledgeable students are better positioned to value the learning experiences of an experiential project. For beginning students, a number of challenges complicate achieving the intended goals of this industry-sponsored approach:

- *Context of origin*: Industry-sponsored projects are not drawn from the everyday life of learners nor are they related to the activities that form their community;
- *Lack of perceived value*: Since learners are not themselves the intended end-users of the software they develop, there is little perceived value inherent to their product; and,
- *Short-lived involvement*: The involvement of learners with these projects tends to be short-term, ending upon delivery of the final product to the industry partner.

These challenges go against three key factors contributing to student motivation (discussed in the preceding background section): The *relevance* and *satisfaction* factors of the ARCS model, and the high degree of student involvement identified as critical to student success [1]. We do not intend to appear as contradicting the value of experiential learning: Instead, we view our work as a refinement of the generally accepted method for applying experiential learning in software engineering education, particularly with students in earlier stages of the curriculum. Our approach is intended to foster an increase in the relevance and satisfaction factors identified in the ARCS model, and to promote a high level of involvement by students – all concerns critical to increasing student motivation, investment and therefore learning.

3.2. Project Solicitation

Key partners in this work are the officers and members of the NAU ACM Student Chapter, who serve as facilitators for the solicitation of projects and act as student-clients. In the summer before the beginning of the course, the instructor initiates the project solicitation effort. During a six week time period and through iterative discussions, a number of best-candidate projects emerge: The selection process focuses in the importance of the project to the operations of the ACM Student Chapter, the rigor of the development tasks involved, and the feasibility for the work to be completed in a single term.

Each project is described by student-clients in a document that outlines an overview of the work to be performed and its importance to the student community, the objectives to be achieved by the final system deliverable, the challenges and risks that are important to address, and the scope of the development work anticipated. The scope of anticipated work and early elaboration of project risks and challenges are particularly important, as they directly support the confidence factor of the ARCS model, and partly address the challenges with minimally guided experiential activities [5]. A summary of the specific projects selected for the Fall 2009 semester course offering appears in Table 1: Each of these systems supports an important activity in the social fabric of the student community.

3.3. Overall Project Development Process

Based on the project descriptions prepared by the student-clients, each student-developer ranks projects by preferences while also providing a short description of their strengths and weaknesses as a developer, including specific technical skills that may be of relevance to specific projects. The instructor then uses this information to assign team members to each project, attempting to respect preferences while balancing teams in terms of size and skill distribution. Each team is then allowed to self-organize and select members for specific roles, while all teams are required to have a team-leader. Remaining activities involve iterating

Table 1. An overview of example projects, outlining background and objectives for each.

	Background and Objectives
Tournament Manager	A system intended to assist with the management of computer game tournaments, supporting the acceptance of tournament entries, generating tournament trees, and providing real-time tournament progress information.
Snack Inventory	Computerized support for providing up-to-date information on current inventory, income, and expense levels of the snack table managed by the ACM Student Chapter, as well as automated generation of shopping lists.
LOUISE	Building on NAU's course management system named LOUIE, LOUISE is intended to provide computer science specific course and degree program management functionality, including personalized student degree plans.
Achievement Points	Inspired by achievement systems from the online multiplayer game milieu, this system is intended to create and manage such a system for the ACM Student Chapter, awarding points for event participation and contributions.
Slide Display	A system intended to manage the creation and storage of publicity slides for ACM Student Chapter events, while also wirelessly managing an LCD screen deployed in the building's lobby for the display of these slides.

through requirements elicitation, design, implementation, and testing. Parallel to these software engineering efforts, student teams also deliver two oral presentations covering their requirements and design, as well as a final review of project-related efforts. Throughout this process, the student-clients are deeply involved in the effort and provide frequent clarifications, feedback, as well as a final evaluation of the team's efforts.

4. Challenges

While we have found the notion of leveraging the student community as a source of projects to be powerful in providing an easily relatable context to junior software engineering students, we also identify some important challenges:

- *Technically knowledgeable clients*: By using students-clients that are themselves knowledgeable software developers, it becomes difficult to insulate developers from the client's envisioned technical solution, which serves to preserve the opportunity of student-developers to be innovative in their designs. While we partially address this through advising of student-clients on the matter, the approach is not wholly successful and we intend to explore more structured measures, such as instructor-monitored requirements elicitation interviews.
- *Lack of a reward structure for student-clients*: While student-developers are tangibly rewarded for their efforts though curricular rewards and learning experiences, there exists no such explicit reward structure for student-clients – the time they spend in their client roles is entirely voluntary. While the development of a software system needed by the student community provides some degree of motivation, this applies more to the community as a whole rather than the individual. While this has only been an issue once (one student-client was unreachable for a period of three weeks), we are investigating practical avenues for better rewarding student-clients for their help, such as offering a small amount of curricular credit for their participation.
- *Personal familiarity with student-clients*: Since our student-clients are drawn from the student community of the department, some student-developers are familiar with them through extra-curricular interactions. These personal acquaintances make it more difficult to maintain professional decorum between developers and clients. While we

strive to ensure a certain degree of detachment through team assignments, we attribute a higher level of informality during client meetings to this personal familiarity.

5. Conclusion and Future Work

One of the key insights that drives our work is that adopting experiential learning projects drawn from the everyday lives and communities of students has the potential to strengthen the levels of relevance and satisfaction that students perceive, therefore increasing their investment in their work. While we have found this approach effective, we are working toward addressing the challenges outlined in the previous section, while also focusing our efforts toward more concrete evaluations of the merits of this approach. Some critical questions to investigate include: How measurably does our course design improve on its intended increase in student motivation? Are the core educational outcomes of the course better served by our re-design? Is the quality of the deliverable artifacts developed by students improved, therefore validating the hypothesis that increased motivation results in higher quality? Finally, of how much benefit to the student community are the systems developed by student-developers within the context of this course, and for how long are they used?

6. Acknowledgements

The author thanks the NAU ACM Student Chapter, and the students and student-clients of CS386 in Fall 2009. This work sponsored in part by the National Science Foundation through NSF Grant CCF-1017408.

7. References

[1] Astin, A.W., What Matters in College? Four Critical Years Revisited, Jossey-Bass, 1993.

[2] Dewey, J., Democracy and Education: An Introduction to the Philosophy of Education, The Macmillan Company: New York, USA, 1916.

[3] Hayes, J.H., "Energizing Software Engineering Education through Real-World Projects as Experimental Studies", in Proceedings of the 15th Conference on Software Engineering Education and Training (CSEET), IEEE Computer Society, Covington, KY, USA, 2002, pp. 192-206.

[4] Keller, J.M. and Suzuki, K., "Use of the ARCS Motivation Model in Courseware Design", Instructional Designs for Microcomputer Courseware, Jonassen, D.H. (Ed), Lawrence Erlbaum: Hillsdale, NJ, USA, 1988.

[5] Kirschner, P.A., Sweller, J., and Clark, R.E. "Why Minimal Guidance During Instruction Does Not Work: An Analysis of the Failure of Constructivist, Discovery, Problem-based, Experiential, and Inquiry-based Teaching", Educational Psychologist, 41(2), Routledge, 2006, pp. 75-86.

[6] Kolb, D.A., Boyatzis, R.E., and Mainemelis, C., "Experiential Learning Theory: Previous Research and New Directions", Perspectives on Thinking, Learning, and Cognitive Styles, Sternberg, R.J and Zhang, L.F. (Eds), Lawrence Erlbaum: Hillsdale, NJ, USA, 2001, pp. 227-247.

[7] Lave, J., Cognition in Practice: Mind, Mathematics, and Culture in Everyday Life, Cambridge University Press: Cambridge, UK, 1988.

[8] Navarro, E.O. and van der Hoek, A., "On the Role of Learning Theories in Furthering Software Engineering Education", Software Engineering: Effective Teaching and Learning Approaches and Practices, Ellis, H.J.C., Demurjian, S.A., and Naveda, J.F. (Eds), IGI Global, 2008, pp. 38-59.

[9] Savery, J.R and Duffy, T.M., "Problem Based Learning: An Instructional Model and its Constructivist Framework", Constructivist Learning Environments: Case Studies in Instructional Design, Wilson, B. (Ed), 1996, pp. 135-148.