Student Perspectives on Learning Through Developing Software for the Real World

Christopher Dean, Rajiv Ramnath, Thomas D. Lynch
The Ohio State University, dean.836@osu.edu, ramnath@cse.ohio-state.edu, lynch.268@osu.edu

Abstract **- From a student's perspective, the standard computer science curriculum can effectively develop fundamental software design principles and techniques, but may struggle to fully prepare students for professional practice. Real-world projects require many skills that are challenging to foster in the classroom, including the ability to implement large applications, interact professionally with others, and independently learn new concepts. Undergraduate programs have attempted to develop these abilities through capstone classes and by encouraging participation in co-ops and internships. At our university, nearly a dozen students have attempted to foster these abilities by doing long-term, real-world, large-scale, commercial-grade software development projects. The first such project recently released an iPhone-based, stadium-centric infotainment application to end-users in time for the 2010 football season. This paper, whose first author is an undergraduate computer science student, captures, from a student's perspective, the educational benefits of ongoing and real-world projects over the more traditional approaches. Following an examination of the educational impacts of these projects relative to the impacts of co-ops, internships and capstone classes, results suggest that long-term, real world projects are a valuable and synergistic component of an undergraduate education in computer science.**

Index Terms - Capstones, Comparative Study, Experiential Learning, Internships, Undergraduate Research

INTRODUCTION

From a student's perspective, the standard undergraduate curriculum in computer science may struggle to fully prepare graduates for professional practice. Research has shown that graduates often lack the practical skills and experience necessary to develop software for the real world or work in diverse teams [1]. Most commonly, the underdeveloped skills fit into two categories: technical and "soft" skills.

Technical skills in graduates are underdeveloped in the realm of practical applications [1]. Many students graduate with a solid foundation in computing theory, but without experience in large-scale software development. In undergraduate programs, students may not be exposed to real-world tradeoffs between clarity and efficiency, may not develop fluency in the necessary languages, frameworks, or platforms, and have little to no understanding of working with customers and designing software for usability. This deficit in practical abilities has led some employers to enroll graduates in a year-long training program in order to build these competencies [2].

Soft skills required for programming in the real world are also sparsely found in recent graduates [1]. In addition to inexperience with standard business practices, most graduates lack sufficient experience working in teams and communicating with others in a professional environment. These abilities are difficult to develop in the classroom, since the team diversity, motivation, and professional setting of real-world software development is usually very different from that encountered in the classroom.

Due to the difficulty in developing these practical skills inside the classroom, students are often encouraged to seek experiences outside of the traditional lecture format. Many programs develop these abilities through a capstone class, which allow students in their senior year to see practical applications of skills developed in previous classes. Many programs also encourage students to seek internships or co-ops in their field of choice. At our university, an average of 17% of engineering undergraduates participated in a co-op or internship between 1999 and 2005 [3].

Although capstones and internships are the most prevalent ways in which a student can develop the practical abilities necessary for real-world programming, they are not the only ways. Another option is through participating in software development projects that are real-world, long-term, and student-led. Over the past two years, nearly a dozen students at The Ohio State University have participated in several such projects.

The result of one of these projects is currently available in the iPhone app store, where it has received over 5,000 downloads and generally positive reviews. Students are currently involved in other projects of a similar nature.

The effectiveness of this experience was assessed using surveys, which were distributed to students within the project, as well as other students who had participated in capstones or co-ops and internships. This allowed the impacts of long-term, real-world experiential learning to be evaluated relative to the traditional methods using a common metric.

Students reported that participation in these real-world, long-term, student-led projects was a valuable supplement to the educational experience. Common outcomes included

improvement in technical skills, such as the ability to develop for large-scale software and an understanding of the engineering process, as well as 'soft' skills, including leadership and accountability. Overall, these benefits were comparable to those yielded by capstones and internships.

Due to inherent differences in the nature of these different experiences, they each provide unique benefits to students. This study suggests that, when implemented correctly, extracurricular projects can be a valuable and effective means of preparing students for the real world.

BACKGROUND

The traditional approaches for developing skills and abilities necessary for professional practice are capstone classes (including other project-based classes) and internships (including co-ops). Capstone classes attempt to simulate real-world conditions and constraints within the classroom. This approach is common in many computer science programs, but is even more widespread for software engineering and engineering as a whole. In fact, ABET requires, as part of accreditation criteria for software engineering programs, that "students must be prepared for engineering practice through a curriculum culminating in a major design experience based on the knowledge and skills acquired in earlier course work and incorporating appropriate engineering standards and multiple realistic constraints" [4].

Capstone courses incorporating strong real-world elements are especially beneficial to students, as they more realistically emulate the real world [5]. Real-world elements can be incorporated in a variety of ways, including the use of actual industry professionals as customers for the students.

Capstones typically have both technical and non-technical learning outcomes for students. Students who participate in capstones have a more solid understanding of technical topics, increased confidence in the ability to integrate multiple technologies, and experience working on large-scale software systems. Realistic capstones also produce students with a better understanding of the software development process, which leads to a more forward-thinking and thoughtful design. In addition to these technical benefits, students also develop stronger leadership and project management skills, become adept at working in teams, and communicate more effectively with customers and peers [5]-[7].

Due to the academic nature of capstone courses, there are also certain drawbacks to the experience. The most significant of these is the project duration, which is often limited by the length of the academic term. These are generally either 10 or 15 weeks in length, so students receive a limited view of the project lifecycle. Many capstone classes are able to provide an in-depth experience in only one or two areas, instead of exposing students to practical applications spanning the full breadth of computer science. Finally, since capstones are graded, the big-picture learning experience that these classes are meant to provide is often overshadowed by the task of getting the program to work [8].

In addition to capstone classes, many students obtain real-world experience through internship programs (which also include co-ops). Through internships, students gain real-world skills and experience that is (understandably) great preparation for real-world software development. Internships are so valuable that some schools, including the University of Cincinnati, require students to alternate two quarters of classes and two quarters of co-op experience throughout their entire degree program, starting as early as their freshman year [9].

Just as capstones yield both technical and non-technical benefits to students, so do internships. Students strengthen their technical skills, such as the ability to independently learn and use new technologies, work on large-scale or pre-existing systems, as well as legacy systems, and evaluate alternative choices for optimal cost and efficiency. Non-technical skills are also strengthened, including leadership and interpersonal skills, the ability to interact professionally with team members and customers, and accountability and other job-holding skills.

Furthermore, since internships are generally in the student's intended field, they allow students to network within their field and evaluate potential employers. Students often become more mature throughout these experiences, since it is their first glimpse at adulthood. Additionally, internship experience makes students more attractive to potential employers and increases the likelihood of job placement [9]-[12].

Internships are not without drawbacks. Currently, participation in internships and co-ops is relatively low overall, averaging near 17% of engineering students [3]. Many students that would benefit from internship experience are persuaded against it, since internships often require students to relocate or take a break in taking classes. There is also a danger that students may not obtain the "complete experience" from their internship if they are placed in ineffectual roles, or if their internship is too short.

Although capstones and internships are the most common means by which students obtain the skills and experience necessary for professional practice, they are not the only options. Some universities, like the University of South Australia, have integrated real-world experiential learning projects throughout the curriculum [13]. The real-world relevance of these projects leads to students exhibiting increased motivation and retention during the rest of their studies.

Other programs featured a more radical transformation of the curriculum. In 1992, Washington State University began offering a team-oriented software practicum to eligible undergraduates [1]. For their first two years, students focused on developing teamwork and technical skills through a variety of labs and hands-on activities. The cornerstone of the practicum was the long-term, real-world software design project that occupied the second two years

of the degree program. Students were divided into teams and exposed to a variety of professional issues that utilized class material and common development practices. The program was successful, as students developed a very solid foundation in teamwork skills and gained experience in all stages of the software development process.

INNOVATIVE SOLUTION

The successes of programs like that of Washington State University, coupled with the demands of industry, make it very clear that students should enrich their education through experiential learning. At our university, students have been participating in extracurricular software development projects. The nature of these projects is defined by three primary criteria: real-world projects used by real-world users, durations greater than one term, and student leadership.

The projects are real-world and are intended for real users. As indicated by the program at the University of South Australia, students are generally more motivated and even excited to work on these real-world projects. This also leads to the software being much larger in scale and more robust than what students have previously worked on.

The projects and their level of student involvement are intended to be long-term. Students are encouraged to begin participating as early as their freshman year or as late as their senior year, leading to a diverse team of students with varying skill levels. Participation typically occurs alongside taking classes, allowing students to gain this valuable experience without having to halt their academic program.

The projects are also student-led, with minimal oversight from faculty or graduate student mentors. Students manage the project using an agile software development methodology that strongly encourages both open communication between team members and direct interaction with customers and project mentors.

The first project was the development of an iPhone application to deliver up-to-date statistics and play-by-play information for the university's football team. It was completed in time for the 2010 football season, during which it was downloaded over 5,000 times. Other mobile-app development projects are currently in the works, involving a greater number of students.

These mobile applications meet the criteria by being real-world, in that they are intended for actual users, long-term, in that the typical development length is nearly half a year, and student-led, in that faculty and graduate student mentors aim to provide only general guidance.

RESULTS

The outcomes of these projects, as perceived by the participants, were determined through a survey. The project leader from the first application is also the lead author on this paper, offering particular insights on the impacts of real-world projects. The survey was also distributed to students who had taken a capstone classes or have been on internships to allow for comparison to traditional methods of gaining real-world experience as a student.

Project Outcomes

Although the first project attained some commercial success, the true benefits were educational in nature. The project also posed some significant challenges, many of which stemmed from the three necessary project criteria.

As a real-world application, the design needed to be easily scalable, since the final product was quite large. Additionally, the application needed to integrate with another large, pre-existing system. This led to a considerable amount of effort going into the design and requirements-building stage. Working on systems of this scale justified the curriculum's emphasis on practices including proper documentation, automated testing, and version control.

Because real users would use the software, it needed to be robust against runtime errors and a large number of simultaneous users. Validating and ensuring this robustness was one of the most technically challenging aspects of the project, as there are few available means of automating testing procedures with user interfaces on the iPhone. Also, since we were designing the application for real users, usability was a primary concern throughout development. Although the application was usable under typical network conditions, there were legitimate challenges in maintaining usable levels of responsiveness during the heavy load of a football game. Considerations such as these are undoubtedly present in professional software development, but they are entirely absent from the undergraduate curriculum.

Since the development occurred over a long period of time, students were able to observe and partake in all stages of the software development lifecycle. In a typical class, which may last as few as ten weeks, students are hardly able to experience these stages in a way that allows them to draw meaningful conclusions. Through this experience, students develop confidence in developing software in all possible stages, including design, primary development, testing and refinement.

Furthermore, incorporating students of all experience levels, some only in their first collegiate quarter, posed a particular challenge. More experienced team members were able to mentor less experienced team members, leading by example and teaching through pair programming. By joining this project at an early point in their academic career, the younger students obtain context and justifications for the concepts and practices taught in their classes. This not only makes their classes more interesting and relevant, but also leads to more meaningful learning.

Over the long-term, these students became valuable members of the team. In doing so, they developed an understanding of the engineering process, became comfortable working on large-scale software systems, and

developed confidence in their abilities as an engineer.

By placing the leadership role on undergraduates instead of faculty or graduate students, projects of this sort can offer undergraduates a leadership- and teamwork-based experience that is immensely valuable. Especially for the project leader, this experience can prove instrumental in the improvement of their team leadership and communication skills. Opportunities for students to interact with customers, develop professional skills, and handle a large degree of responsibility with long-lasting impact are almost nonexistent during the undergraduate education. These are central to their success following graduation, so this exposure is especially valuable.

Comparative Analysis

The benefits of this program were assessed relative to those of capstones and internships through a survey that was distributed to 36 students. Students were asked to identify four technical areas and four non-technical areas in which they feel their experience helped them to grow. They also rated, on a Likert scale, their confidence in their ability to apply this area to professional practice both before and after their experience. Since all three of these experience types aim to bridge the gap between curriculum and professional practice, the technical and non-technical outcomes of these experiences each fell into six common categories.

Technical outcomes included:

- **Curriculum Integration**: an appreciation for and an understanding of the technical topics and software development practices introduced in the curriculum.
- **Independent Learning and Application**: the ability to independently learn and apply material, which may include languages, frameworks, or tools.
- **Integrating Multiple Technologies**: the ability to implement systems using multiple frameworks, languages, and hardware, or as part of a larger system.
- **Working on Large-Scale Software**: experience in systems that are pre-existing, large-scale, or scalable.
- **Analyzing Cost/Benefit Tradeoffs**: competence in practically evaluating and deciding between alternative implementations or technologies based on financial cost, readability, time, and efficiency.
- **Understanding the Engineering Process**: having experience in and an understanding of all parts of the engineering process, as well as the differences between these parts.

Non-technical outcomes included:

- **Leadership**: experience leading or managing a diverse team and managing projects.
- **Professionalism**: the ability to interact with others in a professional setting, behave ethically, and interact professionally in meetings with clients or superiors.
- **Communication**: the ability to effectively use both written and oral communication, especially in technical documentation, meetings, or presentations.
- **Accountability**: the ability to deliver on verbal or written commitments, assess and report on the status of assigned tasks, and manage time effectively.
- **Teamwork**: experience working in a group and collaborating with people of a diverse background.
- **Personal Growth**: the development of maturity, independence, and other personal traits.

Table I shows the percentage of responses that indicated an improvement in each area, broken up by type of experience. For example, although capstone classes try to emphasize practices from the prior curriculum the most, as shown in Table I, internships did a better job at illustrating the relevance or applicability to professional practice. This is evident in Figure 1, on the next page, where the greatest improvement under "Curriculum Integration" was obtained through internship experiences. Furthermore, this result is reasonable, since learning about something that is later observed in industry helps reinforce the value of the curriculum.

Figure 1 compares the students' perceived average improvement in each area based on their experience type. Participants were asked to rate their confidence in applying a specific ability to professional practice on a Likert scale from one to five, with one corresponding to "not confident" and five corresponding to "very confident". Moreover, they were to evaluate this confidence both before and after their learning experience. For the purposes of this study, a student's improvement in a particular area is the difference between these two values. Therefore, an improvement of one indicates a slight improvement, an improvement of two or three indicates a significant improvement.

Internships yielded the greatest improvement in nearly all categories. The one exception is that capstone students perceived a greater improvement in their teamwork skills. This is understandable, since many of the internships were described as being very independent. Generally, the outcomes of capstones and internships are very similar to those identified in previously cited studies.

TABLE I
PERCENTAGE OF STUDENTS REPORTING IMPROVEMENT IN EACH AREA, CATEGORIZED BY EXPERIENCE TYPE

	Area	Caps.	Ints.	Exts.
Technical	Curriculum Integration	62%	21%	50%
	Independent Learning	62%	74%	100%
	Integrating Multiple Technologies	38%	47%	50%
	Working on Large-Scale Software	15%	21%	50%
	Analyzing Cost/Benefit Tradeoffs	15%	16%	25%
	Engineering Process	54%	58%	25%
Non-Technical	Leadership	62%	32%	50%
	Professionalism	38%	53%	75%
	Communication	77%	79%	100%
	Accountability	54%	58%	75%
	Personal Growth	15%	42%	25%
	Teamwork	31%	11%	100%

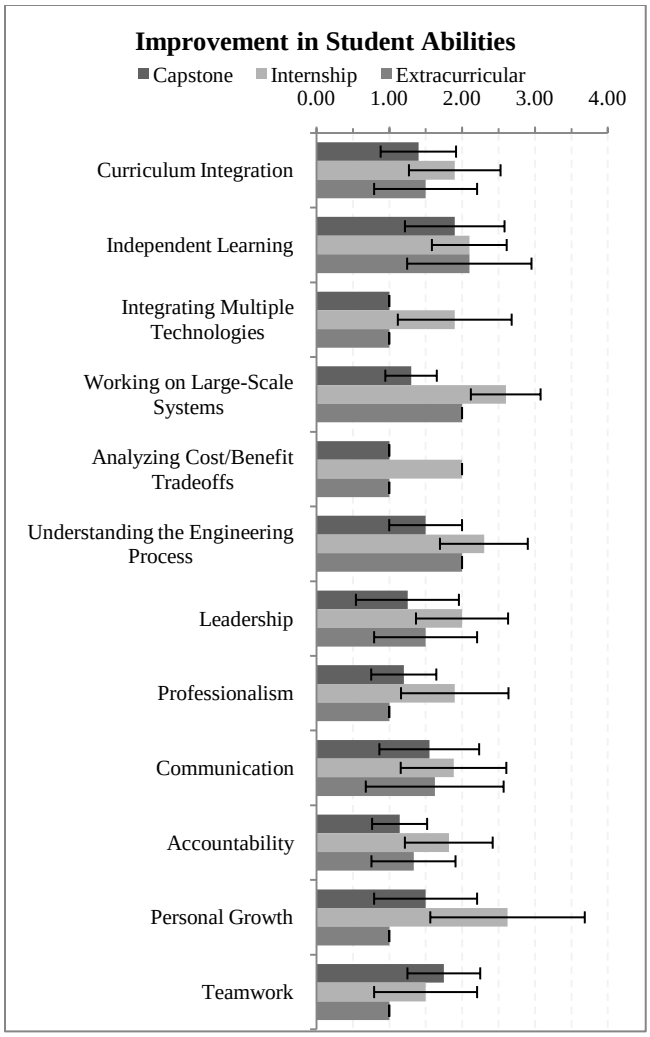

FIGURE 1
AVERAGE CHANGE IN STUDENT CONFIDENCE IN TECHNICAL AND NON-TECHNICAL ABILITIES, RANKED ON A LIKERT-SCALE

DISCUSSION

In all three categories of experience, the values obtained from the survey seem reasonable. The results on capstone and internship experiences are consistent with the results of the literature survey. In both the literature survey and the student survey, capstone classes typically yielded a greater understanding of the curriculum and improved leadership skills. Internships, on the other hand, bolstered independent learning skills, led to a better understanding of the engineering process, and improved students' accountability and professionalism.

One potential concern is that within these extracurricular projects, the benefits obtained by a student depend on the role that the student had within the project. For example, it is much more likely that the project leader who had been in his or her position for six months would have gained much more leadership experience, become better at performing cost/benefit analysis, and have a more thorough understanding of the engineering process than a student who worked on smaller parts of the software for only two months. Academic progress also impacts the benefits of these experiences on students. Additionally, the small sample size used in this study limits the implications of the results. In the future, these results should be corroborated with a larger group of students.

Based on the results, we can quantifiably deduce several benefits of these real-world, long-term, student-led projects over traditional approaches. Many of these are specifically targeting the greatest weaknesses of capstones and internships, at least as far as learning experiences: the short duration of the experience, a focus on project completion instead of learning, the occurrence late in the curriculum, and the potential delay in graduation.

The greatest benefits stem from the fact that the project is long-term. Unlike capstones and internships, where the focus of the project is simply completing the work assigned, the long-term nature of these projects allow learning to be a primary focus as well. The undergraduate college experience does not typically foster the ability to learn independently and find one's own resources for problem solving, so this experience is both necessary and valuable.

Furthermore, since these projects are both real-world and long-term, they allow participants to gain practical experience in all stages of the software development lifecycle. One of the projects is already released and still requires on-going maintenance. Through exposure to this aspect of software development, students learn the importance of documentation and how to maintain and debug complex, production systems.

The nature of these projects also provides students with a rare opportunity to experience real-world development challenges, like clarifying ambiguous problem statements, designing large-scale software, and integrating several different technologies. Students also can join early on and gain insights through pair programming that place them ahead of their peers. Although similar experiences may be obtained through internships or capstones, due to the project duration, improvements may be more likely to occur in these extracurricular projects.

The traditional methods do hold several benefits over this approach. Internships provide students with more opportunity to integrate multiple technologies and become very familiar with specific elements of the engineering process. They also lead to a greater development of professionalism and personal growth, since they remove students from the campus environment and place them in the actual workplace. For many students, an internship provides the first glimpse at "real world" living, and signals an important step in their personal and professional development.

Capstone classes tend to provide stronger teamwork-building experiences, since students are all at a similar point of their academic careers and there is no "appointed" team leader. However, students in capstone project teams are usually in the same discipline because it is a requirement for the major or specialization. Extracurricular projects do not

have that requirement, as students from computer science, business, art and design can all be valuable (and obtain valuable experiences) throughout the project.

The single largest challenge within these extracurricular projects is maintaining a solid commitment from students. This is hardly an issue with capstones and internships, since students are typically either receiving a grade or receiving a salary. Thus far in these extracurricular projects, most students have participated as volunteers. The general outcome was that students were initially enthusiastic, especially towards the beginning of the academic term, but as classes began to require more of their time, their participation dropped off almost completely. In addition to the increasing demands of their academics, it is possible that students lost interest in the project, or may not have been receiving tasks of an appropriate difficulty level. There seem to be two possible ways of maintaining student commitment throughout the project: either pay them, or offer course credit or research credit for their participation. Many students work concurrently with taking classes, and the opportunity to work in their field of interest provides valuable experience. Offering course credit for such projects would also help ensure that these projects remain a priority during the middle of the term, when classes become much more demanding of students' time.

CONCLUSION

Bridging the gap between the skills developed in the curriculum and the skills required for professional practice has traditionally been accomplished through the use of capstone courses and internships or co-ops. An additional alternative is through offering real-world, long-term, student-led software development projects. Participation in any of these experiences allows students to build upon what they have learned in the classroom and be more prepared for professional practice following graduation.

Specifically, real-world, long-term, student-led projects provide several benefits that capstone classes and internships do not. Through a survey of students, these projects were determined to be at least as effective as capstone projects, and nearly as effective as internships, at developing the abilities required of a professional software engineer.

These projects are not without their drawbacks. Students in these "extracurricular" projects do not all receive similar leadership experiences, and often stop participating for a variety of reasons. In the future, several ways of preventing this can be explored for sustaining the effectiveness of the approach. It is possible that by distributing leadership responsibilities and increasing the emphasis on teamwork, as well as solidifying student commitment through course credit or funding, these drawbacks may be overcome. Even with these drawbacks, however, real-world, long-term, student-led software development projects appear to play a valuable, if not essential, role in an undergraduate education in computer science.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under NSF CCLI Grant No. 0837555 and the CERCS IUCRC Center for Enterprise Transformation and Innovation (CETI), supported by the NSF-IUCRC Program, Grant No. 0630188.

We would like to thank Michael Herold for his valuable feedback and support throughout the writing of this paper.

REFERENCES

[1] Schlimmer, J. C., Fletcher, J. B. and Hermens, L. A., "Team-Oriented Software Practicum", *IEEE Transactions on Education*, Vol. 37(2), 1994, pp. 212-220.

[2] Denning, P. J., "Educating a New Engineer", *Communications of the ACM*, 1992, pp. 83-97.

[3] "Salaries & Statistics", *Engineering Career Services at The Ohio State University*, https://career.eng.ohio-state.edu/salaries-statistics.php, Accessed: 29 March 2011.

[4] "Criteria for Accrediting Engineering Programs", 2010, *ABET, Inc.*, http://www.abet.org/Linked Documents-UPDATE/Criteria and PP/E001 10-11 EAC Criteria 1-27-10.pdf, Accessed: 22 March 2011.

[5] James, R. H., "External Sponsored Projects: Lessons Learned", *SIGCSE Bulletin*, Vol. 37(2), 2005, pp. 94-98.

[6] Linhoff, J., Settle, A., "Motivating and Evaluating Game Development Capstone Projects", *Proceedings of the 4th International Conference on Foundations of Digital Games*, 2009, pp. 121-128.

[7] Lekhakul, S., Higgins, R. A., "Senior Design Project: Undergraduate Thesis", *IEEE Transactions on Education*, Vol. 37(2), 1994, pp. 203-206.

[8] Jin, M., "Redesign of the Computer Science Capstone Course by Integrating the Major Field Test (MFT)", *Journal of Computing Sciences in Colleges*, Vol. 24(1), 2008, pp. 239-246.

[9] Dansberry, B., Krech, J., Becker, B., "A Web-Based System to Document Learning Outcomes in a Mandatory Cooperative Education Program", *Proceedings of the 34th ASEE/IEEE Frontiers in Education Conference*, 2004, pp. F2F-1-5.

[10] Jimenez, M., Palomera, R., Toledo, M., "Undergraduate Research and Co-op Education: A Winning Combination", *Proceedings of the 32nd ASEE/IEEE Frontiers in Education Conference*, 2002, pp. S4C-13-17.

[11] Sabag, N., Trotskovsky, E., Schechner, P., "Internship as an Obligatory Requirement for the Degree of B.Sc. in Electronic and Electrical Engineering", *International Conference on Information Technology: Research and Education*, 2006, pp. 94-98.

[12] Payne, D., "Engineering Interns Get Real World 101", *Electronic Engineering Times*, 13 November 2000.

[13] McDermott, K. J., Nafalski, A., Göl, Ö., "Project-based Teaching in Engineering Programs", *37th ASEE/IEEE Frontiers in Education Conference*, Vol. 37, 2007, pp. S1D-11-17.

AUTHOR INFORMATION

Christopher Dean, Undergraduate Student, The Ohio State University, dean.836@osu.edu

Rajiv Ramnath, Director, C.E.T.I., Associate Professor of Practice, Department of Computer Science and Engineering, The Ohio State University, ramnath@cse.ohio-state.edu

Thomas D. Lynch, Ph.D. Student, The Ohio State University, lynch.268@osu.edu