Providing End-to-End Perspectives in Software Engineering

Michael Herold, Joe Bolinger, Rajiv Ramnath, Thomas Bihari, Jay Ramanathan
Department of Computer Science and Engineering, The Ohio State University
herold@cse.ohio-state.edu, bolinger@cse.ohio-state.edu, ramnath@cse.ohio-state.edu, bihari@cse.ohio-state.edu, jayram@cse.ohio-state.edu

***Abstract* – In order to better prepare students for professional practice, we have created a software engineering curriculum that provides an end-to-end perspective that begins with the business context of software, and goes all the way to the ongoing management of software services after deployment. This paper examines how the theoretical aspects of this broad-based curriculum may be effectively delivered through a single course within a traditional computer science program. This curriculum is under a diverse set of constraints and requirements, such as the need for pedagogical consistency, faculty development, consideration of the learning style of computer science students, and a need for an effective continuous improvement process. Our approach uses "engineering-oriented" analysis frameworks such as Porter's Five Forces model for the business aspects, and attribute-driven design for software architectures, an "inverted" classroom mode of teaching where lectures are delivered on line with interactions and exercises that promote active learning reserved for the classroom, case studies developed from real projects to serve as concrete examples, open discussion boards and weekly short quizzes for concept refinement and retention, and a paper-based project where students apply the concepts learned. Faculty development and replication outside the current site are also discussed.**

Index Terms – Active learning, Business context, Inverted classroom, Software engineering education

1. Background and Problem Statement

During our combined 50 years in industry our take on entry-level hires has been that traditional software engineering programs were too narrowly focused - mostly teaching the constructional aspects of small-scale software development using structured development processes. Students emerging from these programs suffer from lack of awareness and misconceptions about software engineering. For example, they are naïve when it came to software processes, either wanting to follow a prescribed process such as [1] or agile [2] by the book or follow no process at all (i.e. rather than tailoring the process to the situation). As a side observation, students could not easily internalize and put into practice the principles of software engineering covered in the standard textbooks [1]-[2], because students did not have the experiences to relate the concepts to. Our perception is supported by other work [3]-[7].

We felt, therefore, that there was a clear need for an *end-to-end* curriculum that educated students in the business-strategic context of computing, all the way to the management of software services after deployment, while covering the people and process aspects, as well as the technical aspects in a nuanced and systematic manner. Hence, we set about designing such a curriculum.

There were many risks in this effort. There were no existing models of such a broad curriculum to borrow from. There were limits to the number of credit hours we could allocate to such a curriculum within our CS program; thus, we could not simply put together material from the subject areas, because it would not fit even in multiple courses. It was hard to find faculty with the needed range of industry experience and pedagogical expertise; hence we needed to develop them (this is an old problem - see [4]). We needed to ensure student acceptance of the large proportion of softer topics. Finally, we needed to ensure ABET acceptance, to not risk affecting the credibility of our program.

Through the aegis of an NSF-CCLI grant-funded project we developed a practice-based curriculum to address the above needs, risks and constraints. This redesigned curriculum consisted of 4 courses (see Section 0). This paper specifically covers *one* of the 4 courses in the curriculum, namely the Software Engineering (SE) course, which is the foundation course in the group.

The rest of the paper is structured as follows: Section 2 presents the context of this course. Section 3 describes the topic areas, and Section 4 describes the delivery and elements and how they complement each other. Section 5 presents evaluation and results. Section 6 covers related research in order to position this work in the context of the other excellent work done in this area, Section 7 concludes with lessons learned and future work.

2. Program Environment

The computer science program at Ohio State may be characterized as a traditional computer science program. About 20 years ago, the program was strengthened primarily by moving traditionally graduate-level courses – like algorithms, networking, databases, computer architecture, and computing theory into the undergraduate program. Software engineering was also one of the courses moved into the undergraduate program at that time. Introductory courses are taught in a component-based language named Resolve.

A majority of undergraduate students in the computer science program, a significant number of graduate students in our Masters' program, as well as a few students in the PhD program take this Software Engineering (SE) course. All students have taken a very intensive group-project-based course prior to SE, and are quite good at writing code and managing the software process at a small team level. In general, all students are highly technically oriented. Except for a handful, most have not had much opportunity to develop soft skills or work in situations where they have to make decisions using incomplete information.

This (SE) course serves as a de facto[1] foundation for a collection of other SE courses – on requirements analysis, software design, and enterprise technologies. The graduate students in this course also typically take an enterprise architecture course. The SE course is also a pre-cursor to a set of Capstone courses (that we collectively term CAP), where student teams work on real-world projects with actual sponsors, in a senior-project like experience. There is no explicit instruction component in CAP; students simply do projects, and instructors and the sponsor provide feedback as the project proceeds. CAP has been *designed* as the project follow-on to SE. CAP is additionally important, because it enables a longitudinal evaluation of SE.

Before the redesign, the instructors of this course were mostly (traditional) computer science faculty (with a research focus on software engineering). After the redesign the instructors have been a Professor of Practice (a clinical-track faculty member), and Senior Lecturers (adjunct faculty) all with considerable industry experience. The course was developed and initially taught by the professor of practice. Subsequent offerings of the course were taught by the Senior Lecturers.

3. COURSE TOPIC AREAS

We begin with examining competitive positioning and strategy and business structure (value chain) using the techniques in [8], followed by a balanced scorecard [9] exercise, which, in turn, leads into domain and problem analyses to develop a portfolio of applications traceable to the competitive strategy and aimed at improving the enterprise's competitive position. One application from this portfolio is selected and taken forward through the rest of the software development lifecycle (SDLC).

This portfolio development exercise leads to requirements identification of functional requirements (using use cases) and solution analysis (using scenarios or user stories) of the selected application. Emphasis is given to deriving non-functional requirements from the business and problem analysis, making them testable and incorporating these tests in an acceptance plan.

[1] When the University shifts to semesters, this course will be the formal pre-requisite.

There is a short introduction to business-IT alignment, along the three dimensions of planning, cultural and structural dimensions, which leads into software processes. Here we discuss incremental and iterative software development, traceability, verification and validation and work-product orientation [10]. Finally, we compare and contrast agile and structured processes. The intent of this progression is to give the students techniques to design a hybrid business-aligned process consisting of a mix of structured and agile elements [11]. Project management techniques for other aspects of project management are also covered. Thus, we discuss parametric and linear techniques for effort estimation, as well as risk management [12].

Next, we introduce architecture, cover quality attribute driven design [13] (noting that the quality attributes are the same as the non-function requirement identified earlier), and emphasize the use of multiple views in documenting the architecture.

We discuss responsibility-driven design [14], and *in the context of frameworks*. We point out that these frameworks are either explicit ones, such as .NET, or implicit ones determined by the legacy software in the enterprise. We show how architecture and design patterns serve as the bridge that help translate the pristine designs created by responsibility-driven design to designs that fit within the constrains of the framework.

Finally, we discuss the deployment and management of applications within a data center, and present the Information Technology Infrastructure Library (ITIL).

We make this obvious breadth manageable by focusing on principles and techniques. We refer to these principles and techniques as *frameworks*. We identified (and customized) frameworks for each topic mentioned above. TABLE shows the frameworks used in each of the course topic areas.

Also necessary in managing the breadth is to let the in-depth learning happen through discussion and application of the frameworks in the project rather than through lecture. The course delivery elements used to achieve this are discussed in the next section.

4. COURSE DELIVERY AND MANAGEMENT ELEMENTS

Our strategy is to provide coverage of the SE topics through the on-line lectures, and use challenge-based instructional

TABLE I
TOPIC AREAS AND CORRESPONDING FRAMEWORKS

COURSE TOPIC	FRAMEWORKS
Business Strategy and Alignment	Porter's Five Forces Model, Value Chain, Balanced Scorecard
Requirements and Analysis	Business Process Analysis, CRC Cards
Software Process	IBM Object-Oriented Technology Center Process, Agile
Project Management	Use-case points, Riskology
Architecture	Attribute-driven design
Design	Responsibility-driven design
IT Service management	IT Infrastructure Library

elements (projects and games) in class for the actual learning. We've used a range of inter-linked elements to develop, teach, evaluate and continuously improve the extensive range of material in a single one-term class. These elements also play a role in faculty development.

The key element that enables this course is the inverted classroom [15]. Students are required to view online lectures (now on YouTube.com) prior to coming to the classroom. In-class time is reserved for quizzes, discussion and project work. The inverted classroom also plays to the capabilities of the faculty who are currently teaching the course. It takes the lecture portion (where they are the weakest, not having had the experience) out of their hands and replaces it with in-class discussion, where they are responding to the students and are able to bring in their experience and specific examples – which are their strengths.

After the lecture has been assigned, an on-line discussion board is opened up to discuss the lecture. Any and all relevant questions and comments are permitted. Faculty, the teaching assistant as well as other students provide responses.

After each lecture, a short (10 minute) in-class open-book, open-notes quiz attempts to verify that students have indeed viewed and digested the lecture. All quizzes have just one question of the form: "Explain <concept> using a concrete example (one that has not already been presented in the lecture or the discussion board)". The thinking is that students have to have internalized the lecture in order to come up with an example. The relevance and degree of detail of the example determines the grade on the quiz. The grading is out of 10, and is rubric-based, so points should be considered an indication of the students' level of learning rather than a percentage.

There is no textbook for this class. The course material consists of video lectures created from audio-overlaid PowerPoint slides, along with online reference material consisting of specific sections in books, articles and papers.

In order to provide grounded examples of the concepts and the frameworks in action, case studies have been created from industry-sponsored projects done by CETI[2] using a prescriptive process described in [16]. So far, three case studies have been completed, two have been used in the classroom, and one (which we call ECO [16]) has been used multiple times. A guest lecture on Agile at Microsoft created by agile trainers at Microsoft is also a standard offering in the class. This provides concrete examples of agile concepts, and how they might be customized in specific environments. We have had several other guest lectures; most recently from a medical supplier company that is going through an agile transformation. We record these lectures, and have permission to replay them in subsequent classes.

GAME-BASED INSTRUCTION IS ANOTHER INTEGRAL PART OF THE COURSE PLAN. ONE COMPONENT OF THE COURSE – AGILE DEVELOPMENT – IS

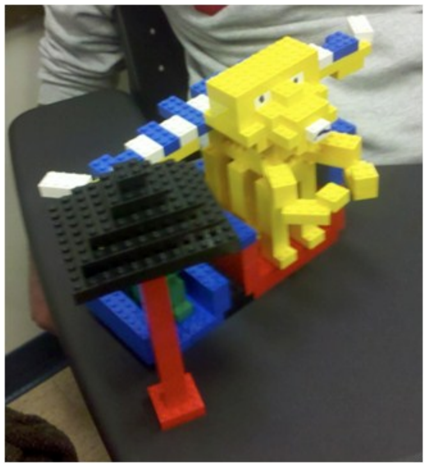
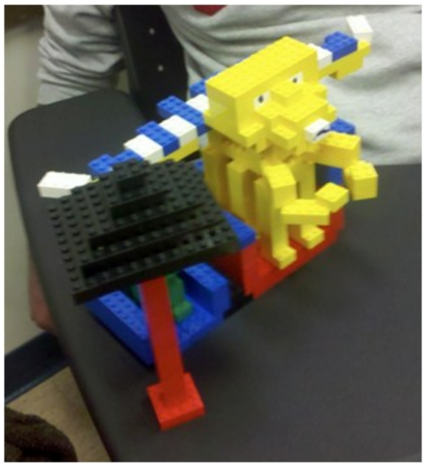


FIGURE 1
A LEGO® ANIMAL CREATED IN THE AGILE GAME

TAUGHT USING A LEGO®-BASED GAME WHERE STUDENTS BUILD AN



"ANIMAL" (SEE

FIGURE 1) based on simple story cards that request features of the animal, such as "Give the animal legs", or "Make the animal one color". The game illustrates the agile development concepts of iterations, backlogs, relative estimation, retrospectives and project velocity, and principles such as close customer interaction.

Students do a project that involves application of all the frameworks. The project is an "on-paper" project that involves no software implementation. On several occasions in the project, decisions needed to be made with incomplete information.

Students use the Unified Modeling Language in all diagrams where possible. Students are not expected to learn all the facets of UML; rather students are asked to use UML to effectively document decisions, in order to show *how* each project decision meets its goal. For example, students have to show *how* each non-function requirement is met by one or more architectural decisions.

5. EVALUATION AND ASSESSMENT

This course and the follow-on CAP courses are key to ABET accreditation of our program. Assessment and

[2] CETI (http://www.ceti.cse.ohio-state.edu) is a NSF Industry-University Collaborative Research Center (IUCRC), an industry-driven consortium that researches the application of computing technology to address enterprise-scale problems.

evaluation of the course is now therefore very important, and is being done extensively, and both formatively and summatively. Two types of formative assessments are being used with respect to evaluating and improving student learning. (1) Quizzes are marked up and returned in as timely a manner as possible (usually in the class following the class in which the quiz was given). Students who are assessed to have not provided good answers are advised on their answer as well as on the studying behavior expected of them so they might learn better. (2) The project is divided up into parts corresponding to the various SDLC phases, and made due in intervals. These parts are also marked up and returned for students to improve upon and re-submit (without prejudice, unless there is clear evidence of lack of effort), through discussion if necessary. The final grade for the project is, in fact, only assigned at the end. The summative assessment of the student consists of the quizzes, the final project grade and a final examination.

The formative and summative assessments of the students serve as assessment components of the course as well. Three additional assessments have been put in place for the course. First, at the beginning and end of the course, the students were asked a simple open-ended question, "Describe how you would identify, design and develop a large-scale software application for an enterprise." This question is intended to baseline the student at the beginning of the course with respect to knowledge of the end-to-end concepts that the course will cover, and to test their concept recall at the end of the course. Second, a small number of students were interviewed at the end of the course about their reaction to the course methods. Thirdly, we used the official university Student Evaluation of Instruction to assess student perceptions of the various dimensions of the course, as well as to elicit open-ended comments.

A final component of the SE assessment is from the new instructors of SE. After they have taught the course (having been given a walkthrough of the syllabus and the material) we assess their buy-in, comfort level and their rating of SE through one-on-one interviews. Assessment results are presented in Section 0.

Assessment Results and Evaluation

This course was run three times (in 2009) to work out any kinks (and there were several) before we felt the assessment data was representative of the course content and design rather than of execution. Also, with the assessments that depend on student feedback (such as the interviews and SEIs), student response rate is not particularly high (about 15%), with the responding demographic potentially biased from actual student population.

Quiz scores demonstrate good initial retention by students. Average scores in the quizzes in the last term (40 students) were between 7.14/10 (quiz 3 on IT Alignment) and 9.7/10 (Quiz 10 on the Case Study). We typically discard the aberrantly low results of the first quiz (we believe, and this is confirmed by anecdotal student feedback, that this is due to the students not yet being used to the inverted classroom method).

The "before-and-after" questionnaire showed progression in the students' conception of software engineering. About 40% of the students had elevated their conception of business context and requirements identification from tactical ("automate a process") to strategic ("reduce buyer power"). About 75% of students referred to specific techniques ("use attribute-driven design to derive the architecture") to perform tasks that they had previously referred to in generic terms ("design the system"). A high 95% of "after" responses showed awareness of non-functional requirements. About 25% of students included risk management in their concept of project management.

Two areas of "systemic" lack of student understanding were identified from the project, specifically, (a) acceptance tests for non-functional requirements and (b) risk management. With respect to acceptance tests, a large part of the problem, we feel, is the paper nature of the projects, when the student, also playing the customer, is tempted to just say "I accept it when it's good". We will address this through clear examples of acceptance tests in the case studies. With respect to teaching risk management, we are developing a risk-management add-on to the LEGO® game.

Student Reactions to Course Elements

The students interviewed uniformly liked the "Socratic style" (as one student put it) of the inverted classroom. Of the 8 students who provided SEI comments, results were mixed, ranging from "The lecture videos online are a smart use of time…" to "…it takes some getting used to being responsible for lectures that don't occur in the classroom at all" to "… difficult to watch over an hour of lecture videos multiple times per week on top of all of the document reading/writing required…" All students assessed the workload of this class as higher than expected.

Interviewed students were asked how they studied from the online lecture. Most gave an explanation and evaluation similar to this: "Printed slides and took notes on the slides… highlighted questions … Felt the organization was good and didn't cause extra work to integrate the material."

All interviewed students commented positively on the game. The students clearly had fun! One student wanted "more difficult tasks". The assessment identified the need for having a co-located, collaborative customer as the primary learning of the game.

The most-used ECO case study [16] was rated between 3 (neither effective nor ineffective) and 5 (highly effective) in reinforcing concepts taught in the class. One of the students commented that ECO's one-developer nature hindered ability to see how teams would work. However, one of the points being made by ECO was that canonical software processes break down in real-world application precisely because of structures like in ECO! In other words, the student missed a point of the lecture. This is an issue to be remedied when we teach SE again.

The Microsoft lecture had a range of ratings from 3 to 5. The other guest lecture was given a high rating of 5.

The project is arguably the weakest element of the course in terms of student acceptance. Most students were resigned to it. A few students liked it, but a small, highly vocal set of students consistently equated it with "busy work". From the instructors' perspective though, the project has been invaluable in evaluating student understanding. Also, students in the CAP course often use the project workbook in SE as a template.

Instructor Acceptance

These were assessed by one-on-one interviews of two the three instructors who taught the course apart from the course developer – and all first-time instructors of the course. One instructor rated the inverted classroom method as requiring more work, partially because the instructor himself felt the need to internalize the broad range of concepts. However, both rated the (inverted classroom) method as effective (4/5) and improved their ability to teach because they could work much more directly with the students, and had more opportunities to bring in their own experiences[3]. The game was rated as highly effective (5/5).

Instructors felt that additional incentives and penalties to increase student participation towards the dog days of the quarter should be considered. However, this was not rated as a serious issue, but rather just a side comment.

An unexpected outcome reported by the instructors was that having the course developer's voice on the lectures *initially* diminished their authority. However, this authority was soon re-established, and even provided an avenue to the instructor to provide counterpoint views where appropriate.

In general, instructors felt that their experience was good and would be even better the second time around.

6. Related Research

In this section we describe key related work and position our work against what has been done before. To begin with, [17] characterizes the existing state of SE education, and separates programs into "SE-Heavy" courses that might fit in a specialized software engineering program and "SE-lite" courses that might fit into a broad computer science program. Taken by itself, the SE course is SE-Lite, however, the SE curriculum taken as a whole is certainly SE-Heavy.

Several papers focused on the gaps in software engineering education. Good examples are [17] and [18] that make two points relevant to this paper – (a) the need for focusing education appropriately, and (b) the need for communicating industrial reality more effectively.

Recent models of SE education have looked at learner-centered pedagogy, problem-based learning, active learning, and taking a constructivist approach to learning [19]-[21] and [15] are experience reports on the inverted classroom. References [23]-[25] discuss the case-study approach in its various forms. Our own work [16] describes *how* a case study may be efficiently developed. Reference [26] describes the need for teaching "contextualized" approaches – such as teaching students of the need for customization of software engineering processes. A similar point is made in [22].

Related to our use of games as simulation exercises, [27] presents a simulation-type game in which players can simulate and see the effects of various software processes. A useful paper that describes how to improve the communicability of (i.e. increase the learning from) simulation-type games is [28].

Finally, [29] supports our conclusion of the need for an ecosystem (like our IUCRC) for faculty and curriculum development.

7. Lessons Learned and Future Work

Our most significant lesson learned is that effectively teaching a broad, but interlinked, set of SE concepts within the constraints of a typical CS program is indeed feasible. Students are mostly accepting of the course, and adequately demonstrating recall and application. The inverted class approach is promising; however, care must be taken in its planning and delivery so that lectures are sequenced properly. Most importantly, the lectures and study material must be stripped to their essentials and designed extremely well so that students are not doing an amount of work significantly in excess of the credit hours granted.

There are also certain key elements for replication of such a model at locations outside of CSE at OSU (one of our highly desired goals). The first is that faculty who teach SE must have a strong industry background and significant breadth of enterprise-scale software engineering experience. Any CS department that seeks to use this course as a model needs to have access to such faculty. The second necessary element for replication (and, specifically, the ongoing renewal of this course) is the ability of course-developers to participate in real enterprise-scale projects from which they can develop case studies. We are fortunate that the existence of our NSF-IUCRC facilitates our engagement in these kinds of projects with local industry.

Positive student attitudes are key for any course innovations to succeed. The more we can incorporate active learning that is not seen as "artificial" the better students are engaged.

Future Work

The move to a semester system will add 4 weeks, which will help slow down the pace of the course, as well as allow us to add testing and quality assurance, and refine framework-based design and IT service management. More games are also being designed.

We hope to begin longitudinal assessments as well, at least via the Capstones. Also, students in SE are all in their senior year, and hence enter the workforce a short time after taking SE. Thus, there is the opportunity to assess the

[3] We were very pleased with this observation, because that was precisely what we hoped would happen!

impact of SE at least in the early part of the students' profession. To this end, we are building a tracking database.

Finally, we are working with 3 partner universities on dissemination efforts, and an NSF TUES expansion grant. We welcome additional collaborators.

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 also acknowledge CETI industry partners, notably, Nationwide Insurance, the City of Columbus, the OSU Office of the CIO, Mylee Inc., Ohio Department of Job and Family Services, and TechColumbus. We thank ThoughtWorks, Inc. for the use of their LEGO® game.

REFERENCES

[1] Sommerville, I., Software Engineering (9th Edition), Addison Wesley, 2010.

[2] Pressman, R., "Software Engineering: A Practitioner's Approach", McGraw-Hill Publishers, 2009.

[3] Hawthorne, M., Perry, D., E., "Software Engineering Education in the Era of Outsourcing, Distributed Development, and Open Source Software: Challenges and Opportunities", Proceedings ICSE'05, May15-21, 2005, St. Louis, Missouri, USA, 2005.

[4] Werth, L., "Software Engineering Education: A Survey of Current Courses", ACM SIGSOFT SOFTWARE ENGINEERING NOTES vol 12 no 4, Oct 1987.

[5] Ghezzi, C., Mandrioli, D., "The Challenges of Software Engineering Education", ICSE 05, May 15–21, 2005, St. Louis, Missouri, USA.

[6] van Vliet, H., "Reflections on Software Engineering Education", IEEE SOFTWARE, May/June 2006.

[7] Ramnath, R., "Global Software Development for the Enterprise", Proceedings of the COMPSAC 2006 Workshop on Global Software Development, Chicago, USA, 2005.

[8] Porter, M., "Competitive Strategy: Techniques for Analyzing Industries and Competitors", Free Press; 1998.

[9] Kaplan, R., Norton, D., "The Balanced Scorecard, Measures that Drive Performance", Harvard Business Review, 1992.

[10] IBM, "Developing Object-Oriented Software: An Experience-Based Approach", Prentice Hall; 1st edition, December 1996.

[11] Boehm, B., Turner, R., "Balancing Agility and Discipline: A Guide for the Perplexed", Addison-Wesley Professional, 2003.

[12] DeMarco, T., Lister, T., "Waltzing With Bears: Managing Risk on Software Projects", Dorset House, 2003.

[13] Bass, L., Clements, P., Kazman, R., "Software Architecture in Practice", Addison-Wesley Professional; 1st edition (December 30, 1997.

[14] Wirfs-Brock, R., Wilkerson, B., "Object-Oriented Design: A Responsibility-Driven Approach", pp.71 –75, Proceedings of OOPSLA, 1989.

[15] Manning, K. S., "Work in Progress - Outsourcing the Lecturing in an Engineering Physics Class", 40th ASEE/IEEE Frontiers in Education Conference, Washington, D.C. 2010.

[16] Bolinger, J., Yackovich, K., Ramnath, R., Ramanathan, J., Soundarajan, N., "*From Student to Teacher: Transforming Industry Sponsored Student Projects into Relevant, Engaging, and Practical Curricular Materials*", Proceedings of the Transforming Engineering Education: Creating Interdisciplinary Skills for Complex Global Environments, Dublin, Ireland, April, 2010.

[17] Mitchell, W., "Is Software Engineering for Everyone?", Proceedings of the Mid-South College Computing Conference, 2004.

[18] Lethbridge, T., LeBlanc Jr., R., Sobel, A. E. K, Hilburn, T., Díaz-Herrera, J.L., "SE2004: Recommendations for Undergraduate Software Engineering Curricula", IEEE SOFTWARE, November/December 2006.

[19] Ludi, S., Natarajan, S., Reichlmayr, T., "An introductory software engineering course that facilitates active learning", Proceedings of the 36th SIGCSE technical symposium on Computer science education, February 2005.

[20] Hadjerrouit, S., "Constructivism as Guiding Philosophy for Software Engineering Education", SIGCSE Bulletin, December 2005.

[21] Quade, A., "Developing a Hybrid Software Engineering Course that Promotes Project-Based Active Learning", TiCSE'06, June 26–28, Bologna, Italy, 2006.

[22] Bareiša, E., Karčiauskas, E., Mačikėnas, E., Motiejūnas, K., "Research and Development of Teaching Software Engineering Processes", International Conference on Computer Systems and Technologies, CompSysTech, 2007.

[23] Burge, J., Troy, D., "Rising to the Challenge: Using Business-Oriented Case Studies in Software Engineering Education", Proceedings of the 19th Conference on Software Engineering Education & Training (CSEET'06), 2006.

[24] Varma, V., Garg, K., "Case Studies: The Potential Teaching Instruments for Software Engineering Education", Proceedings of the Fifth International Conference on Quality Software (QSIC'05), 2005.

[25] Hilburn, T., Towhidnejad, M., Nangia, S., Shen, Li. "A Case Study Project for Software Engineering Education", 36th ASEE/IEEE Frontiers in Education Conference, 2006.

[26] Fendler, J., Winschiers-Theophilus, H. "Towards contextualised software engineering education: an African perspective", Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering, ICSE, 2010.

[27] Navarro, E., van der Hoek, A., "Design and Evaluation of an Educational Software Process Simulation Environment and Associated Model", Proceedings of the Eighteenth Conference on Software Engineering Education and Training, April, 2005.

[28] Peixoto, D., Prates, R., Resende, R., "Semiotic Inspection Method in the Context of Educational Simulation Games", Symposium for Applied Computing, SAC, 2010.

[29] Nikolov, R., Ilieva, S., "Building a Research University Ecosystem: the Case of Software Engineering Education at Sofia University", ESEC/FSE, 2007.

AUTHOR INFORMATION

Michael Herold, Ph.D. Student, Department of Computer Science and Engineering, The Ohio State University, herold@cse.ohio-state.edu.

Joe Bolinger, Ph.D. Candidate, Department of Computer Science and Engineering, The Ohio State University, bolinger@cse.ohio-state.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 Bihari, Senior Lecturer, Department of Computer Science and Engineering, The Ohio State University, Senior Consultant, Nationwide Insurance, bihari@cse.ohio-state.edu.

Jayashree Ramanathan, Director, C.E.T.I, Associate Research Professor, Department of Computer Science and Engineering, The Ohio State University, jayram@cse.ohio-state.edu.