Connecting Reality with Theory - An Approach for Creating Integrative Industry Case Studies in the Software Engineering Curriculum

Joe Bolinger, Michael Herold, Rajiv Ramnath, Jayashree Ramanathan
The Ohio State University, bolinger@cse.ohio-state.edu, herold@cse.ohio-state.edu, ramnath@cse.ohio-state.edu, jayram@cse.ohio-state.edu

Abstract - **Case studies have been successfully integrated into a wide variety of educational contexts and disciplines. Today, case studies are increasingly accepted as valuable teaching tools in science and engineering curriculums to complement the underlying theory of the field. Well-articulated cases can reinforce abstract concepts, demonstrate the nature of real client interactions, and showcase the relevance of soft skills to students that lack significant practical experience. However, assembling and delivering quality case studies to students requires a great deal of practical disciplinary knowledge, and a careful alignment of the case content and delivery style with curricular objectives, course learning outcomes, and the overarching institutional format. In this paper, we summarize our experience with an approach for constructing case study teaching materials that are integrative and deep in content, but also carefully aligned to the core principles and format of a senior-level software engineering course. Our approach ensures that the cases are complex enough to retain their realism and intrinsic appeal, while mirroring the format and objectives of the course such that the cases reinforce key points in a familiar and consistent fashion to the students.**

Index Terms – Case study, case study teaching, creating case studies, software engineering education.

INTRODUCTION

Case studies have a rich history in many educational settings and their relevance to science and engineering disciplines has been steadily growing [1]-[3]. The depth and practical relevance of a well-presented case serves as a good complement to the relatively dense theory that is a necessary component of an engineering curriculum, and it can help facilitate more interactive teaching methods and active styles of learning [4]. Nonetheless, taking a history of real world events and repackaging them into a set of useful classroom materials is a difficult art form to master, which requires both disciplinary and educational knowledge and skills.

Despite the prevalence of example case studies and high-level guidelines for authors of new cases [2, 3, 5]-[10], there are few prescriptive methods for ensuring that cases are created in a manner that is tightly integrated with the core concepts and principles of a course. The most skilled storytellers have a keen ability to tailor their tales to fit their audience, and good cases should leverage this quality to the extent possible.

The most compelling cases are not pulled off a library shelf or selected from a general repository and told with the generality of a scientific theory. Instead, they are localized and idiosyncratic to some degree. They are told from a perspective that the students of a particular institution can relate to, and they resonate with these students in a manner that is consistent with the theory and abstractions that they have been taught. A properly crafted case is tailored to the audience, and instructors are in need of tools that can help them rework case materials in a fashion that is consistent with the students' total educational experience. Any lack of consistency will be perceived as a distraction from the heavy course load that engineering students already bear, however interesting that temporary distraction may be.

In this paper, we begin by introducing two initial sets of case study materials that were presented and used separately in multiple offerings of the same course over a two-year period. Next, we present a method for systematizing the process of creating case study materials based on a careful analysis of the course's format, content, and learning outcomes. Then we describe one of these cases in more detail to show how it was used to develop a standardized structure for cases, and to illustrate some of the valuable software engineering relevant lessons that it brought into the classroom. Finally, we discuss how this method has allowed us to quickly transfer the experiences that our graduate-level students gain through local industry-led projects into useful case study materials that retain their unique and localized flair, but also remain deeply integrated with the principles and nomenclature of the course by conforming to a standard model.

RELATED WORK

A key challenge to instructors in many fields, like software engineering, is teaching students how to cope with the human element of their future professional careers. Software engineering is a people-oriented profession, in which success relies as much on correctly identifying problems to be solved as it does actually solving them [11].

978-1-61284-469-5/11/$26.00 ©2011 IEEE

For instance, Carroll [7] notes how case studies of software projects have an authentic quality to them, and that this authenticity provides a very effective context in which to teach students about software usability issues and a related set of engineering methods. The authentic nature of cases, including real people and places, complex problems, and the reality of unexpected outcomes and odd circumstances, all work together to make what is being taught more convincing than it would be if simple and contrived illustrations of the methods that were taught had been used.

Despite the popularity of the case method, methods for developing cases are typically at a very high level of abstraction [5, 6, 8, 9, 12]. They are often more like writing guidelines than they are like meticulously constructed recipes for successfully crafting new cases. This is to be expected to some degree because the case approach to teaching is a generally applicable one [1], and all of the instructional value of a case stems from the content of the narrative that is told [13, 14]. Nonetheless, the same story can be retold in any number of formats or styles, and efforts like [3, 7, 14] show that within various sub-disciplines of a field there are most likely ways of formatting or structuring case studies that are generally better or worse. Although some educators are natural storytellers, there are many excellent teachers that do struggle with this art form and systematic and reusable methods can help.

TOWARDS A STANDARD METHOD

I. Initial Case Studies

When we began using case studies as part of one of our software engineering courses, we took a simple and straightforward approach. As part of our involvement in an NSF-IUCRC program, we have access to a rotating collection of graduate students that have been involved in various industry-led projects. These projects can range in complexity and duration from a few months to a few years, and many have the potential to be transformed into very compelling cases. To create our first set of cases, a couple of graduate students that were involved in different projects met with an instructor and discussed the goals of each case and the basic format of the materials to be produced. These students later played various roles in delivering parts of their cases in subsequent offerings of the course, either through in-person lectures and class discussions or though online videos assignments that preceded follow-up debates in class.

To keep the course fresh, we used this effort as an opportunity to develop a standard method to create case materials that could be reused as more and more projects matured. This standard process would allow us to keep the material interesting for the class, and also give the students that worked on the projects another avenue to present, share, and benefit from their work. After the first set of case materials were fully assembled, they were presented in multiple offerings of the same course, and a qualitative evaluation was carried out in one these offerings that showed a generally positive student response [15].

The initial version of our method focused heavily on what *content* to include in the cases. Since experienced students were the primary authors of the case materials, not the instructors, it was important that they be given clear and reliable guidelines for identifying the appropriate content that they could pull from their experiences and share. However, at that time we neglected the importance of *format*. It was not until after we had completed both of the initial cases that we realized the importance of the presentation format, and how difficult it could be to make the same logical point through the lens of different case narratives and distinct authors.

The method we developed relies on a structured analysis of the course that the case will be presented in, which breaks down the course into themes, topics, and methods (more detail can be found in [15]). Essentially, the themes and topics correspond to the core conceptual issues that the course addresses while the methods refer to concrete items that students are expected to learn in order to increase their level of understanding about the overarching topic. For instance, one documented learning outcome of our course is that students will master the concept of system quality attributes (or non-functional requirements), how to extract them through the process of requirements analysis, and how to apply architectural patterns and tactics in order

Category	**Business Context**				
Theme	*Environment*	*Enterprise Architecture*		*Innovation*	
Key Topics or Theories	Sense/Respond	Business Strategy	Business Tactics	Institutionalization	Market Maturity
Method(s) & Technique(s)	Porter's 5 Forces	Value Chain, Component Business Model	Balanced Scorecard	Resource/Values/Processes	Innovation Zones
Example(s)	ECO, SID	ECO			

Category	**Software Engineering Process**						
Theme	*Structured Process*		*Agile Processes*		*Software Process Tailoring*		
Key Topics or Theories	Project Workbook	Scenario Driven	Agile Mindset	Customer Driven	Project Shape	Phases & Timeline	Lean
Method(s) & Technique(s)	Risks, Constraints, Roles	Requirement Specifications	Agile Manifesto	Stories, Burn-down	Spider Diagram	Inception - Transition	YAGNI, JIT
Example(s)	ECO, SID	SID			ECO	ECO	

Category	**Software Engineering Practices**					
Theme	*Requirements*		*Analysis*			*Deployment & Maintenance*
Key Topics or Theories	Elicitation	Functional & Non-functional	Conceptual Modeling	Project Estimation		Service Delivery
Method(s) & Technique(s)	Use Cases, Scenarios, Ethnography	Stakeholder Identification	UML, CRC Cards, XP Metaphore	Funnel Curve, G/Q/M, Use Case Points		ITIL
Example(s)	ECO	ECO, SID	SID			
Theme	*Design*				*Implementation & Verification*	
Key Topics or Theories	Architecture	Risk Management	System Qualities	Patterns	Frameworks	Patterns
Method(s) & Technique(s)	Multi Viewpoints, 4+1	Transitions & Indicators, CBAM	Sensitivity/Tradeoff Points, ATAM, Utility Tree	EAI Patterns	Standards based, J2EE	GOF Patterns
Example(s)	ECO, SID	ECO	ECO, SID			ECO

FIGURE 1
COURSE TOPICS AND CASE COVERAGE FRAMEWORK

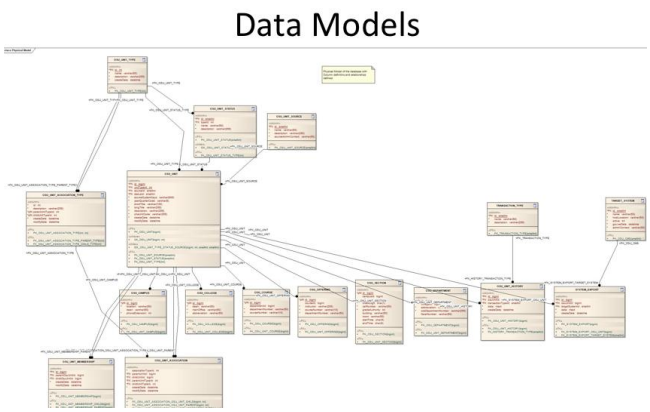

FIGURE 2
SID CASE EXAMPLE

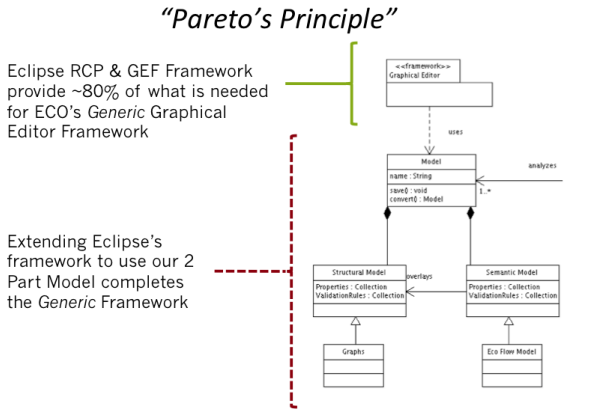

FIGURE 3
ECO CASE EXAMPLE

to achieve these quality attributes. This topic is discussed in class through a variety of logical techniques and frameworks that help the students better grasp and apply the concept, such as the utility tree technique that is a part of the Architecture Tradeoff Analysis Method (ATAM) [16].

Breaking down the course in this fashion allows the author of a new case to do two things. First, he or she can use it to filter their experiences so that the final case materials are relevant to the course and extraneous details can be omitted. In other words, elements of any case study materials can be clearly mapped directly to the course topics. Second, it allows the author to phrase the case narrative in a way that will be familiar with the students. For example, if a potential case provides a good example of how a team of software developers have tailored their engineering practices to deal with the forces in their environment (another key topic in our course), the course break-down shows that it should be illustrated to the students through the use of a certain kind of spider diagram [17] because the students will have already learned about that particular type of diagram.

Figure 1 provides an example of how our course was broken down at one point in time and how two cases, ECO and SID, were used to illustrate various topics in the course by reinforcing the techniques and methods that were discussed as a part of the coursework. For example, it shows that the ECO case was used to discuss business strategy through the application of the Value Chain model [18], while the SID case was not used to discuss strategic concerns. It also shows that each case covers and supports the course in different ways.

II. Why Format Matters

Our method allowed us to create these two cases, determine which cases were better examples of various topics, throw out excess detail, and rephrase the narratives in terms that would be consistent to the students. However, the format of both cases was substantially different as a result of the distinct authors, despite having developed a common frame of reference. Figures 2 and 3 are excerpts from each of theses cases that were both intended to make roughly the same point. Specifically, their purpose was to show how a particular non-functional requirement had an effect on their respective system's design, and they both do so through a UML diagram that acts as a catalyst for discussion.

The SID case was developed shortly before ECO and it was presented in a relatively story-like and narrative format. It was presented much like an average project report would be, without any superimposed structure. In contrast, the ECO case was presented though a structured pattern-based format. In this fashion, each chunk of the ECO project that was presented to students was accompanied by a relevant pattern and a discussion of how the generic strategies embodied by each pattern helped solve a real problem. As a result each aspect of the case was presented in a very consistent and predictable format consisting of the introduction of an abstract pattern, a discussion of a real problem, and finally a resolution of the problem using relevant techniques from the course.

This gave the entire ECO case a more modularized or short-storybook feel, while the SID case appeared more like a novel. It is unlikely that either approach is universally better than the other. However, the chunked nature of the pattern-based approach appears to have been well received by the students during our initial evaluation, and we have also found that following it makes it easier for our graduate students to document their project experience as cases without as much confusion and cogitation that we experienced when we created our first pair of cases.

Figure 3 shows a concrete example of the pattern-based approach used in the ECO case. At this point, the Pareto Principle is introduced as a generic pattern for organizing teams of software developers and resources [19]. The figure is annotated to show that the design of ECO was achieved largely through the use of an off-the-self framework with minor extensions (solid green section), and with a relatively small, but critical, set of custom built components that were most critical to the project's success (dashed red section). This quickly leads the students into discussions of how project risks are often reflected in software architecture, how teams decide where to put the most skilled developers

or do cross-training, and the additional trade-offs associated with build versus buy decisions.

After the SID and ECO cases had been successfully used a number of times we continued our effort to develop additional cases. As noted, we found that the pattern-based approach was not only useful as a means to communicate with the students, but that it was also a very useful technique for creating the case materials. It would be relatively unusual for full-time instructors, like ours, to play significant roles in the projects that are ultimately used as cases. Consequently, the quality of a case is heavily reliant on the ability of those that were involved to recall their experiences and to generate the bulk of the case study material on behalf of the instructor. Although some amount of expert re-work is possible, instructors cannot simply make up potentially interesting aspects of the case if the primary sources neglected to mention them or articulate them clearly.

During the creation of the ECO and SID cases, there was a fairly high amount of discussion between the instructor and those contributing a case about what to include, how to properly phrase things, etc. However, by following the pattern-based format after the ECO case we found that we could give contributors a script that they could follow in order to more easily generate the majority of a complete case study. It did not eliminate the back-and-forth discussion entirely, but it did make the process smoother and more efficient to a noticeable degree. A basic outline of this script is shown in Table I.

Following the pattern-based approach, contributors of a case are provided with a discrete set of patterns at the start of the process that affords them a very explicit means to navigate through their experiences, select appropriate examples, and stimulate their creativity as an author. Rather than being directly probed for relevant information by an instructor, the patterns give the original sources of a case a very direct mechanism to identify potentially relevant experiences and to organize them into a coherent case. Similarly, because patterns are conceptually at a high-level, it is quite easy to restate them in terms that relate to concrete topics in the course. Hence, it becomes easier for someone that is familiar with the course but not necessarily the particular case, like an instructor, to rework some details without diluting from the authenticity of the case.

Due to the popularity of the patterns in various software engineering sub-fields it is quite easy to find repositories of patterns for this purpose. To date, we have primarily relied on an *Organizational Pattern Reference* from [19], and, to a lesser extent, a classic *Software Design Pattern Reference* from [20]. The course our script is based on is heavily focused on software architecture topics, which is why we have chosen the pattern repositories that we have. However, it can be adapted to other courses by locating alternative repositories of patterns or heuristics, which are available for many topics areas such as usability, physical architecture, graphical user interface or interaction design, etc.

TABLE I
CASE STUDY SCRIPT

Section 1 - Context
1. Introduce the project and the sponsor / business
2. State the problem
3. Characterize the context / environment
a. By choosing 1 or 2 techniques from the *Business Context Category*
Section 2 – Software Engineering Process
4. Introduce the people
5. Characterize the project
a. By choosing 2 or 3 patterns from the *Organizational Pattern Reference*
b. For each, show how the pattern influenced the process by choosing 1 technique from the *Software Engineering Process Category*
Section 3 – Analysis & Design
6. Restate the problem from step 2
7. Characterize the analysis process
a. By choosing 2 or 3 techniques from the *Software Engineering Practices Category*
b. For each, show why and how the pattern from step 5a was relevant to the analysis activity
c. For each, show the result of the analysis activity and its relation to the system's high-level design
8. Characterize some of the software system's low-level design
a. By choosing 1 or 2 techniques from the *Software Engineering Practices Category*
b. For each, show how an implementation strategy can be developed by using the *Software Design Pattern Reference*
c. For each, explain how the implementation satisfied the high-level design

Although the pattern approach does prescribe a basic structure for case materials, it does not constrain the format entirely. For instance, it can be used to develop lectures, debate topics, group activities, or whatever else fits with an instructor's teaching style. The key point is that it gives the primary authors of cases enough structure and guidance to document their stories precisely and consistently and then hand them off. Re-packaging the completed case into various artifacts, such as lectures or assignments, is something that can safely be done by an instructor without a significant loss of authenticity or quality.

III. A Dynamic Case Library

The pattern-based case method has given us a way to elicit good examples of software engineering practices from mature industry-driven projects in a simple and consistent fashion. Beyond our two initial cases, we have created two more cases using this method and another is in progress. Figure 4 shows excerpts from the ECO case and two of the new cases that were developed using the method. Of course, the figure does not truly capture the fidelity of the actual case materials. It shows only some visual cues that are presented as part of interactive class discussions, assignments, and other supplementary materials.

However, it does illustrate how each of the authors was able to select relevant patterns (shown in quotes) from a pattern repository and apply them in the context of their respective project in order to describe how various factors shaped their internal engineering process. Ultimately, this is one of the lessons that we never want left out of a case or underemphasized in its presentation: that processes must be

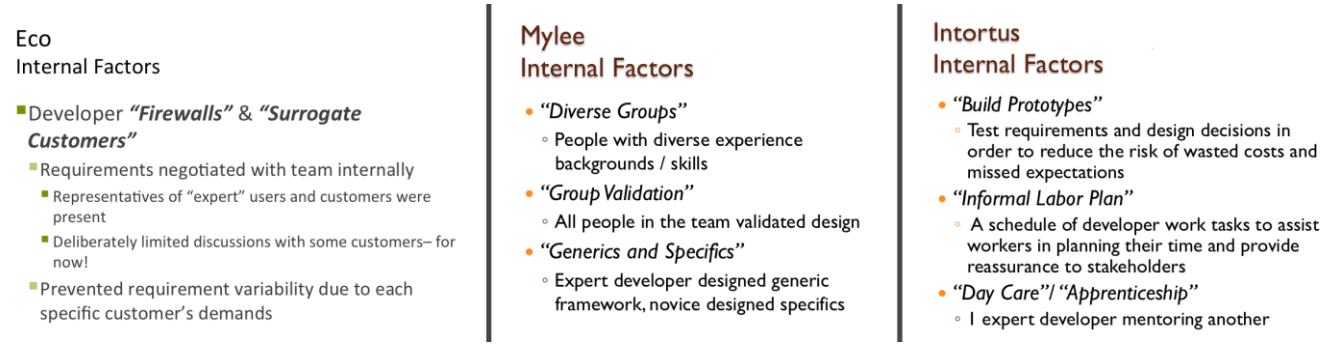

FIGURE 4
EXAMPLES FROM THREE CASES USING THE PATTERN-BASED METHOD

tailored to fit their projects. While this message can be embedded in a case study regardless of its format, our pattern-based script helps ensure that the primary sources of a case's content focus on this message (step 5 in Table I). This is particularly important in situations where the primary authors are neither instructors nor experts in the case method, like ours.

In addition, our structured course framework allows the available cases to be directly related to topics in the course so that appropriate examples can be selected based on the needs and preferences of the students at any given time. Unlike related efforts to amass a library of historical cases, our approach is dynamic and biased toward the generation of new cases rather than archival and excessive reuse. By following a repeatable method for creating new cases and having a systematic means for relating them to course topics, we hope to be able to leverage case materials on an as-needed basis and cater to the varying needs and interests of our students.

Cases that are timely, relevant to current social issues, or embody hot topics in the field are likely to have a greater appeal to students. A dynamic library of cases can leverage this and shape the classroom experience in ways that dated materials cannot. However, this flavor of relevancy comes at a price. Exhaustive evaluation and continual improvement of individual cases is no longer possible and more attention must be paid to the process of creation. Teaching through a dynamic case library requires generational methods, like ours, which can be incrementally improved.

CONCLUSIONS

Constructing case studies for an engineering classroom is not as simple as telling a good story or even presenting it well. Cases must be deeply integrated with the content of the curriculum, reinforce core concepts in the vernacular of the classroom, and capture the interests of the students. While cases that reflect significant or well-known events can be quite compelling, so are those that are recent, laden with familiar characters and an accessible cast, and set in places not far from home. Through our involvement in a variety of industry-led students projects we have been working to develop a generational method for creating integrated case studies that leverage all of these qualities.

To date, we have completed three cases that conform to our standard pattern-based case model and have more in development. An evaluation of our first case based on the standard model has given us some level of confidence that the approach is effective and that the pattern-based case template is well received by our students. However, a complete evaluation of our process is challenging and is part of our on-going work. For example, does the process make creating case materials easier for those involved (graduate students and instructors in our case), or reduce the opportunities for error? We believe that it does given what we have learned from our experiences transitioning from developing case materials without a framework to following a process that follows a pattern. However, each case is unique in terms of complexity, the topics it can be used to illustrate, and other factors that make evaluation a complex and long-term endeavor.

Case studies are useful supplements to the large amount of theoretical material that is necessary in engineering disciplines. The reality of the situation that a case embodies, through all of its complexities and subtle quirks, can reinforce abstract concepts and practical scenarios in ways that simplified examples cannot. Rather than reusing static materials from a library of cases, we have presented an approach for generating case studies quickly and systematically. Most importantly, it provides a venue for our senior-level students with recently acquired practical experience to share the lessons that they have learned with other students in a way that is deeply integrated, consistent, and true to the principles and objectives of our program.

ACKNOWLEDGMENT

This research is supported by the National Science Foundation under Grant No. 0753710 and the CERCS IUCRC Center for Enterprise Transformation and Innovation (CETI).

REFERENCES

[1] Herreid, C. F., "What is a case? Bringing to science education the established teaching tool of law and medicine," *Journal of College Science Teaching*, Vol 27, No 2, November 1997, pp. 92-94.

[2] Raju, P. K. and C. S. Sankar, "Teaching real-world issues through case studies," *Journal of Engineering Education*, Vol 88, No 4, October 1999, pp. 501-508.

[3] Hilburn, T. B., Towhidnejad, M., Nangia, S. and Li Shen, "A case study project for software engineering education," *Proceedings of the 36th Annual Frontiers in Education Conference*, October 2006, pp. 1-5.

[4] Bonwell, C. C. and J. A. Eison, "Active learning: Creating excitement in the classroom," *ASHE-ERIC Higher Education Report*, ED336049, 1991.

[5] Jiang, H., Ganoe, C. and J. M. Carroll, "Four requirements for digital case study libraries," *Education and Information Technologies*, Vol 15, No 3, September 2010, pp. 219-236.

[6] Dolmans, D., Snellen-Balendong, H., Wolfhagen, I. and C. Vleuten, "Seven principles of effective case design for a problem-based curriculum," *Medical Teacher*, Vol 19, No 3, September 1997, pp. 185-189.

[7] Carroll, J. and M. Rosson, "A case library for teaching usability engineering: Design rationale, development, and classroom experience," *Journal on Educational Resources in Computing*, Vol 5, No 1, March 2005.

[8] Herreid, C. F., "What makes a good case?" *Journal of College Science Teaching*, Vol 27, No 3, December/January 1998, pp. 163-165.

[9] Herreid, C. F., "Cooking with Betty Crocker: A recipe for case writing," *Journal of College Science Teaching*, Vol 29, No 3, December/January 2000, pp. 156-158.

[10] "Case collection." 2010. National center for case study teaching in science. http://sciencecases.lib.buffalo.edu/cs/collection. Accessed: 30 March 2011.

[11] Shaw, M., Herbsleb, J. and I. Ozkaya, "Deciding what to design: Closing a gap in software engineering education," *Proceedings of the 27th International Conference on Software Engineering*, May 2005, pp. 607-608.

[12] Herreid, C. F., "Twixt fact and fiction: A case writer's dilemma," *Journal of College Science Teaching*, Vol 31, No 7, May 2002, pp. 428-430.

[13] Herreid, C. F., "Case studies in science: A novel method of science education," *Journal of College Science Teaching*, Vol 23, No 4, February 1994, pp. 221-229.

[14] Clancy, M. and M. Linn, "Case studies in the classroom," *Proceedings of the 23rd SIGCSE Technical Symposium on Computer Science Education*, March 1992, pp. 220-224.

[15] Bolinger, J., Yackovich, K., Ramnath, R., Ramanathan, J. and N. Soundarajan, "From student to teacher: Transforming industry sponsored student projects into relevant, engaging, and practical curricular materials," *Proceedings of the 2010 IEEE Conference on Transforming Engineering Education: Creating Interdisciplinary Skills for Complex Global Environments*, April 2010.

[16] Bass, L., Clements, P. and R. Kazman, *Software Architecture in Practice*. Boston: Addison-Wesley, 2003.

[17] Boehm, B. and R. Turner, *Balancing Agility and Discipline*. Boston: Addison-Wesley, 2003.

[18] Porter, M., *Competitive Advantage*. New York: Free Press, 1998.

[19] Coplien, J. and N. Harrison, *Organizational Patterns of Agile Software Development*. Upper Saddle River: Pearson Prentice Hall, 2005.

[20] Gamma, E., Helm, R., Johnson, R. and J. M. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Boston: Addison-Wesley, 1995.

AUTHOR INFORMATION

Joe Bolinger, Ph.D. Candidate, Department of Computer Science and Engineering, The Ohio State University, bolinger@cse.ohio-state.edu

Michael Herold, Ph.D. Student, Department of Computer Science and Engineering, The Ohio State University, herold@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

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