When Theory Meets Practice: Enriching the CS Curriculum Through Industrial Case Studies

Joan Krone
Denison University
Granville Ohio 43023
740-587-6484
krone@denison.edu

David Juedes
Ohio University
Athens, Ohio 45701
740-593-1596
juedes@ohiou.edu

Meera Sitharam
University of Florida
sitharam@cise.ufl.edu

Abstract

Most computer science departments provide their students with a mathematical foundation which enables them to master theoretic concepts necessary for algorithm analysis, formal specifications of software, reasoning about correctness of software, and classifying problems as solvable or unsolvable. Most departments also require students to design and implement a variety of programs ensuring that students become familiar with issues of language syntax and semantics.

However, students often express the concern that they are applying their theoretic background to "toy" problems only. They wonder what "real" problems look like and whether or not their background will be sufficient for them when they are faced with an industrial situation. At the same time companies who are hoping to hire well prepared computer programmers, systems analysts, and other technical staff express their concern that recent graduates entering the work force are not adequately prepared for dealing with large problems in a real world setting.

The approach presented here addresses the problem of bridging the gap between theory and practice by actively seeking out industrial partners who provide academics with real problems that can be addressed by teams of students in the academic setting.

1. Introduction

Most undergraduate computer science degree programs, in keeping with the curricular recommendations of the ACM and IEEE [6, 7] provide students with a mathematical foundation which enables them to master theoretic concepts necessary for algorithm analysis, formal specifications of software, reasoning about correctness of software, classifying problems as solvable or unsolvable, and understanding computational complexity. Most degree programs also require students to design and implement a variety of programs ensuring that students become familiar with issues of language syntax and semantics, along with software engineering principles. This range of experiences usually provides students with a balanced technical understanding of the three essential processes in computer science: theory, abstraction, and design [6].

While the three essential processes in computer science education are spread throughout the computer science curriculum, often one or two of these processes dominates the others in any given course or context. For this reason, students sometimes express the concern that they are applying their theoretic background to "toy" problems only or that they will never use their theoretic background. While it is easy to excuse students for failing to appreciate the relationship between the abstract and practical aspects of computer science, there is a measure of truth in their concerns.

Concerns similar to those expressed by students have been articulated in a number of recent forums and related position papers, such as Aho *et al.* [1], Block [5], Santosa [14], which have brought to the forefront several crucial deficiencies in computer science education. A careful reading of these papers reveals a common theme - - a lack of connection between academic work and work in the "real" world (industry).

The work described here seeks to tighten the loose connection between theory and practice in computer science education by utilizing a partnership between academia and industry to document industrial problems with non-trivial technical issues and to transfer that knowledge into the classroom. This partnership is based on the following premises:

1. A wealth of techniques and results from the world of academic computer science and mathematics can be applied to practical problems.
2. A positive benefit can result from increased interaction between theoreticians and practitioners.

A crucial component of our method is in the delivery of the knowledge gained from industrial partners to the classroom. Here we employ the case study approach. This approach has been used successfully for many years in a variety of business schools, and we borrow heavily from their pedagogical approach [1].

2. Addressing the problem

To address the problems expressed in the introduction, we formed a collaboration among three computer science faculty in three separate departments, and we developed a course in which students work with both an industrial partner (or an industrial case study) and a faculty member to apply theory and current research to real problems. This approach goes far beyond the usual internship by setting up a partnership in which students, faculty, and industrial partners work together, each bringing a special perspective to the particular problem.

It is of interest to note that the three faculty members involved are not only from different institutions, but from different kinds of institutions in far different settings. Both Juedes, at Ohio University in Athens, Ohio, and Sitharam, formerly at Kent State University and now at the University of Florida, teach at public institutions with both undergraduate and graduate students. Krone is at a private college with only undergraduate students. We have found that our approach works well in both kinds of settings.

Leading up to the implementation of this new course, we engaged in a variety of preparation activities. These activities included six crucial steps. Some of these had to be completed before the teaching of pilot sections began and others are a work-in-progress. The following is a description of the tasks that we believe are crucial to develop such a course.

- **Identification of industrial partners**: The faculty must find companies or organizations who are enthusiastic about this plan. So far the three faculty who developed this course have found several companies who agreed to participate. They include consulting firms, service companies, and at least one government organization.

- **Selection of appropriate problems**: Each of our group of faculty worked together with their particular industrial partners to choose problems or problem parts that are reasonable for the given students and the time constraints.

- **Construction of a reading list**: Appropriate materials such as software engineering papers, specialized background for the specific problem, and other relevant books or articles must be made available for students.

- **Preparation of case studies**: Our group of faculty, together with student assistants, selected some problems to solve in advance of the course. Using a standardized format, they wrote up solutions as case studies. These case studies may be used in their entirety as examples, or the initial part of them may be used as an assigned project to students enrolled in the class. The format used for writing up the studies was chosen so that the problem statement and background material could be separated from the solution easily.

- **Teaching the course**: Each of the three faculty members have taught a pilot section of the course and one has taught additional sections since the pilot. The problems addressed have included a medical imaging problem, a Y2K problem for an electric company, a database conversion problem, and a problem to compute heights on the surface of an object with the goal of constructing an image of that surface.

- **Maintenance of a website**: A website has been set up to share the ideas and materials that comprise the outcome of this project. As new case studies are completed, they will be added to the site, http://www.denison.edu/mathsci/tracs.

3. Description of the course

The initial goal of our group was to design a new capstone course for computer science majors. The purpose of the course is to provide students with an opportunity to apply the formal methods and approaches that they have learned in their undergraduate studies to practical case studies in industry and/or government.

The typical prerequisites for the pilot sections included data structures and algorithm analysis, computer organization, theory of computation, operating systems and/or architecture, some mathematics (calculus and/or discrete math), and software engineering. Most students had at least one, usually more, additional electives in computer science.

Preparation for the course includes the setting up of industrial partners who agree to work together with academic faculty to select particular problems that students can work on with reasonable expectation of success.

4. Choice of problems

Problems were chosen to satisfy the following goals:

- To expose students to a broad variety of large, prototypical, real-world projects.

- To apply a combination of formal and theoretical concepts, techniques and results in real situations.

- To illustrate recent theoretical research results in the context of a chain of applications leading to the solution of a real-world problem

- To involve the students in significant software implementation while being conscious of its role in the larger context of an industrial case study project.

5. Preparation and use of case studies

Of course, no matter how appropriate a problem might be for illustrating specific concepts and ideas, the practical issue of whether or not it is reasonable to expect that the problem can be solved within the time frame agreeable to both to the academic setting and the industrial setting must be a major point of concern.

To help in identifying what problems are workable, the faculty hired students to help with the preparation for the course. Both faculty members and students met with industrial partners to select problems. These students worked on the problems under the direction of the faculty members and the industrial partners. In some cases, the problem needed to be cut down, but usually it was possible to work out a solution.

The students and faculty documented the industrial problems as case studies, using a specific format. The problem statement and background material were given first, followed by a proposed solution. These case studies served several purposes. First, they served as a proof of concept, in the sense that some students could successfully apply their theoretic background to these "real" problems. Second, each case study was used as examples for students taking the course. Each study illustrates to the student how a problem should be described and how a solution can be documented. Third, it is possible to use just the problem statement and background portion of the case study as an assignment for students in the course to carry out.

6. Methods and examples

Any one of three approaches can be used in teaching the course, and all three have been used by the faculty who designed the course. In all three methods, the students work in teams of three or four on a team.

In the first approach, students work on a problem independently from the industrial setting. In this case, the students take the problem, design a solution, and possibly implement the solution, but the industrial partner does not rely on the student team to complete the job. This method would be appropriate when a previously written case study is used. With this approach, the interaction between academia and industry takes place primarily before the problem is assigned. The faculty member, together with a student worker, solves the problem before the course begins, writes the solution into a case study format, and completes the connections with the industrial partner. When the course begins, the students who work on this problem do not see the previously constructed solution until they have completed their work.

The advantage of this method is that the faculty member already knows a lot about various aspects of the problem and can give hints and guidance based on certain knowledge of where pursuit of a solution takes one. The disadvantage is that students do not get the exhilaration of meeting the industrial deadline.

In the second approach, student teams work on a given problem in parallel with a team from the industrial partner. Here the student team may meet with the industrial team on occasion to compare their proposed solutions.

Both the students and the industrial workers benefit. Students get the experience of working on an industrial problem, noting the pressure of meeting milestones and fitting their work into a larger context. Industrial workers get some fresh insights from students.

The third approach has the student team working in real time on their problem, taking full responsibility for the solution. In this method, frequent meetings with the industrial partner are necessary. This approach often work well when it can be arranged. With this approach students gain a variety of experiences, such as the practical concerns of meeting deadlines, compromising on assignment of tasks, articulating challenges and accomplishments to an industrial partner, and noting how compromises are made when disagreements arise.

7. Some specifics

In one pilot section, there were four teams of three or four students each. Two teams worked in parallel, one worked independently, and one worked in real time. The team who worked in real time were offered jobs by their industrial partner and one of the members is currently working at the site, continuing responsibility for the software his team designed and implemented while enrolled in the course.

The real time team worked on one aspect of a Y2K problem for a local electric company. Their mission was to take the existing software system for receiving requests and to replace it with a new system. The old system had been written for DOS and was not Y2K compatible. The students designed and implemented a new comparable system for a windows environment. Their work included the preparation of appropriate documentation both for future maintenance programmers and for users.

The work involved a lot of issues: the identification of abstract data types for the solution (modularization), efficiency concerns (both time and space), some re-engineering (students had to figure out what the old system did and then build a new system to accomplish the necessary tasks). Students made use of their software engineering background, as well as use of their data structures and algorithm analysis courses. There were also human issues. The current users had been used to a command driven environment and the new system has a graphical user interface and is menu driven.

The team who worked on a previously solved problem applied what they had learned in their mathematics courses, data structures and algorithm analysis, and software engineering to take data from a medical CT-scanner to produce a computer image. The work involved applying computed tomography concepts to the collected so as to convert the data to a from that could be used for graphical displaying of the related image. The team had the experience of needing to learn some new mathematics and even a bit of physics in order to understand the problem. Of course, they liked doing so, recognizing that the field of computing often requires its practitioners to learn material in other fields in order to serve those in that field.

As an example of a group who worked in parallel with an industrial partner, one team took a problem provided by a consultant for NASA. In this case the consultant was working on the same problem of improving their methods for producing surface images from data collected with a laser beam. The students met with the consultant once a week, both contributing ideas and getting clarification from the industrial partner. The students used one method and the consultant another, the intent being to compare methods for “adjusting” the data.

Since those pilot sections were taught, Krone has continued to use this approach in her software engineering course, having acquired some additional industrial partners who have provided new problems. For example, GTE has worked remotely with teams of our students who contributed toward the creation of a networked customer database.

A local company in Granville worked with teams of students to create special software to be used in medical diagnosis of stroke victims. Students worked closely with their industrial partner, meeting at least once a week to discuss all aspects of the project. This was a particularly good example of working with an industrial partner all the way from the initial pinning down of requirements to the design of a solution and ultimately the beginning of development of the agreed upon solution. In this particular case, we had student teams follow up from one semester to the next so that they could continue work on the project they had started. The team from one semester turned over what they had done to a new team for second semester. However, all the students remained in contact with the industrial partner and benefited from seeing the project progress.

In another case Dean Witter in Chicago presented us with a problem requiring a student team to update (actually completely replace) software that their company used to allow their employees to provide instant information to customers about their accounts. The company flew our students to Chicago periodically to meet with them and they also kept in touch via email. The students were able to complete the project with a product that is much faster than the one it replaced, as well as more useful. The students and the company were quite pleased about the use of theory in analyzing algorithms for efficiency on the one hand and the successful communication of requirements on the other.

Every team presented weekly progress reports for the class as well as for their industrial partner. This activity gave students an opportunity to hear about what other teams were doing and to offer suggestions and to get ideas for their own work. In fact, many students indicated that the progress reports were an especially important part of the course.

On the website http://www.denison.edu/mathsci/tracs there are links to some case studies and more information about this approach to the teaching of software engineering. On the website http://www.denison.edu/~krone there is a link to some recent projects done by students at Denison.

8. Assessment

Without exception, students enrolled in the course reacted in a positive manner. Most students considered this course to be a true capstone experience, one in which they were able to synthesize the various components of their curricular background so as to apply it in a real setting.

The faculty who have been teaching this course have given out student evaluation forms, asking students to give comments about what they did and did not like about the course. The comments made indicated that students found the course to be the most challenging one in their experience. They also indicated that while there was a heavy demand on them, they liked having the experience of seeing a real problem and doing whatever background reading it took to understand and attack that problem.

Students also learned a lot about group responsibilities and group interaction. Students often make the assumption that working in a group will make less work for each member. Of course, they found out many realities about group dynamics and their comments made it clear that they understand the challenges of how to apportion work within a group and how to come to agreements about what should be done when and who is responsible for what.

The grading of the students was a challenge for us. Of course we wanted to give recognition for individual performance, but since the projects were group projects, there is an argument that there should be a group grade. To address this situation we prepared a list of items that each group would be graded on. The list included such items as meeting specifications, design of solution, the implementation chosen, testing plans, setting and meeting milestones, and others. For that list each group received a group grade. To adjust for individual performance, each student turned in a peer evaluation of every member on that student's team. Surprisingly enough, students were quite honest in giving credit where credit was due. In most cases students on a given team gave their highest scores to the same team member.

From the point of view of the industrial partners, it is also significant to note that industrial partners have been positive about this experience as well. Many have asked to be included when the course is next taught. Although most company employees are quite busy and do not have a lot of time to wax eloquent about such a course, many have noted that they have benefited from the liaison with academia and intend to continue with it.

9. Conclusions and recommendations

Work on this project began in June, 1998. Industrial partners have been highly enthusiastic about interacting with academia.

This partnership provides a two-way benefit between academia and industry. Students and faculty have the opportunity to apply theoretic results and current research ideas to industrial problems, thereby becoming aware of the challenges, both technical and non-technical, facing industry.

At the same time, industrial partners get a chance to see what is currently in the computer science curriculum and to both give and receive suggestions about the content. They also have the opportunity to meet and evaluate newly educated students who may make significant contributions in the future. Equally valuable is the chance to become aware of recent discoveries that may enhance their capabilities.

The faculty members involved are planning to write up more of the case studies they have been using. Hopefully, time will permit that activity to continue so that their efforts can be shared with others who would like to use them.

10. Acknowledgments

This work was supported by NSF-CCLI award 9752671.

References

[1] Alfred Aho et al. "Emerging Opportunities for Theoretical Computer Science," manuscript, October 15, 1996.
[2] *Algorithms in the Real World*, Carnegie Mellon University, See http://www.cs.cmu.edu/afs/cs/academic/class/15850c-s96/www/home.html
[3] Louis B. Barnes, C. Roland Christensen, and Abby Hansen**,** *Teaching and the Case Method, 3rd Edition***,** Harvard Business School Press, 1994.
[4] John Bennett, "Building Relationships for Technology Transfer," Communications of the ACM, Vol. 39, No. 9, September, 1996, pp. 35-37.
[5] I.E. Block, "NSF Initiative Promotes Sea of Change in Math Education," SIAM News, September, 1996, pp1, 12-13.
[6] *Computing Curricula '91*, Association for Computing Machinery and the Computer Society of the IEEE, 1991
[7] *Computing Curricula 2001, Steelman Draft (August 1, 2001),* Association for Computing Machinery and the Computer Society of the IEEE, 2001. See http://www.acm.org
[8] "Emerging Issues in Interdisciplinary Education," SIAM News, October 1996, pp.6.
[9] Jim Foley, "Technology Transfer from University to Industry," Communications of the ACM, Vol. 39, No. 9, September, 1996, pp. 30-31.
[10] Mark Guzdial and Rick Weingarten, "Research in the Union of Computer Science and Education," manuscript. See http://www.cc.gatech.edu/gvu/edtech/nsfws

[11 Jeff Johnson, "R < ----- > D, not R&D," Communications of the ACM, Vol. 39, No. 9, September, 1996, pp. 32-34.
[12] Jane Linder, "Writing Cases: Tips and Pointers," Product Number 391026, Harvard Business School Publishing, 1994.
[13] "Mathematics Speaks (and Listens) to Industry," SIAM News, January, 1995, pp.10.
[14] Fadil Santosa, "What industry needs from academia: CSE Education in the 21st Century" IEEE Computational Science and Engineering, Summer, 1996, pp. 4 - 9.
[15] Benson P. Shapiro, "Hints for Case Teaching," Product Number 585012, Harvard Business School Publishing, 1985.
[16] Benson P. Shapiro, "Hints for Casewriting," Product Number 587052, Harvard Business School Publishing, 1986.
[17] Benson P. Shapiro, "Introduction to Cases," Product Number 584097, Harvard Business School Publishing, 1988