A B.S. Degree in Informatics: Contextualizing Software Engineering Education

André van der Hoek, David Kay, and Debra J. Richardson
Department of Informatics
Donald Bren School of Information and Computer Sciences
University of California, Irvine
Irvine, CA 92697-3425 U.S.A.
+1 949 824 6326
{andre,kay,djr}@ics.uci.edu

ABSTRACT
Software engineering (SE) is very different in focus from traditional computer science: it is not just about computers and software, but as much about the context in which they are used. This means we must teach about software and information, development and design, technical and social issues, while creating solutions as well as understanding and analyzing them. In effect, we must teach a discipline broader than SE or CS alone for SE education to be effective. At UC Irvine, we designed and now offer a program doing just this – a four-year B.S. degree in Informatics. The major brings topics in SE together with human-computer interaction, computer-supported collaborative work, social analysis, and management, along with other application disciplines. Here, we discuss the philosophy behind the major, its structure, and the questions concerning SE education that the new major raises.

Categories and Subject Descriptors
K.3.2 [**Computers and Education**]: Computer and Information Science Education – *computer science education, curriculum*

General Terms
Design

Keywords
Informatics, education, computer science education, software engineering education, contextual learning

1. INTRODUCTION
Educating students in software engineering (SE) is difficult. This is clear from dedicated conferences, myriad proposed approaches to teaching the material, and other widespread attention being paid to the subject. It is interesting, within this context, to examine different philosophies of the nature of SE education. Currently, these philosophies fall into two main streams of thought: SE education based on mathematical and engineering principles (as in the McMaster undergraduate degree in SE [1]) and SE education based on the topics that are traditionally considered part of SE (as in the ACM/IEEE curricular guidelines for SE [2] or ABET SE criteria [3]). Both these philosophies have their proponents and opponents, and have been implemented at various institutions.

It is worth comparing the curricula resulting from these schools of thought with the fundamental issues that make day-to-day software engineering difficult.

1. *Software / information.* Software is never the goal in and of itself. The real goal is information (which software can help process or produce, typically as part of a larger system) that has an effect on the real world.
2. *Development / design.* Design, broadly construed, is the most difficult phase in SE. It represents the transition from the problem to the solution and, as such, requires a combination of creative and technical abilities. Yet, the focus of most SE educational materials is development, not design.
3. *Technical / social.* While software by its nature is technical, in the end it is always placed in social settings. People, groups, organizations, and cultures, after all, are its main users.
4. *Creation / study.* SE curricula focus on software creation, but we must frequently study software *in situ* to understand its position, use, impact, and so on.

In essence, these four issues bear upon the context of software engineering. While most existing SE programs focus on the first dimensions (software, development, technical, creation), few (if any) incorporate serious treatment of the second (information, design, social, study). Even for design, while we know a lot about design notations, we do not know how to design solutions effectively and with confidence.

2. INFORMATICS
We argue that for SE education to be effective, it must be taught with the context in which software engineering takes place and must therefore present a much broader perspective than SE or CS alone. It must address all eight dimensions introduced above, comprehensively throughout the entire program. Furthermore, the topics must be introduced and taught in an integrated manner to avoid students treating each topic as its own disconnected, insular domain, and to let students appreciate the breadth of the subject

ICSE'05, May 15–21, 2005, St. Louis, Missouri, USA.
ACM 1-58113-963-2/05/0005.

from the start. At UC Irvine, we have defined and now offer such a program, a Bachelor of Science degree in Informatics [4].

Informatics is the study of the design, application, use, and impact of information technology. UCI's Informatics curriculum provides some innovative approaches to the issues raised above (note that our curriculum is quarter-based):

- We focus on the information that drives the need for software with a two-course sequence on databases followed by a course on information retrieval and a course on information visualization, as well as a new introductory course sequence that shuns the traditional programming-only focus by situating its projects in information needs.
- We address issues of design, tools, and infrastructure with a two-language first year that already introduces a mini focus on designing solutions, a two-course sequence in programming languages with an emphasis on creating and evaluating useful "little" languages, a year-long course sequence on software design, and a focus throughout on the identification and use of appropriate tools.
- We address societal, ethical, and cultural issues with a full-year course sequence on the social and organizational impact of computing, part of which includes an ethnographic focus on observing the effects of systems on their users.
- We address the intrinsically distributed and team-oriented nature of SE with group work starting in the first year and with courses in project management and computer-supported collaborative work.
- We address complexity and impart experience with realistic case studies starting in the first year, with project-based courses every year, and with a concluding year-long senior design project for a real-world customer.

Additionally, the Informatics program has courses in common with UCI's long-standing major in Information and Computer Science. It includes courses in logic, discrete mathematics, and statistics. The technical and mathematical underpinnings of the field form an essential component of the program, but its fundamental design principle is the pervasive attention it pays to the broad context in which SE takes place – everything else is secondary to this principle.

The result, then, is a degree program that at its core introduces the topics of software engineering, information retrieval and management, programming languages, human-computer interaction, computer-supported collaborative work, privacy and security, and the effects of technology on society. At its periphery, the major touches upon many additional disciplines and application domains, including management, digital arts, visualization, economics, social science, cognitive science, organizational computing, game technology, and many others.

3. RESULTING CHALLENGES

While we believe the Informatics major is a step forward in SE education, a number of critical challenges remain for the community to address:

1. *How can we incorporate context in the educational experience?* The Informatics major does so with particular courses and a concerted effort (for instance, in a new integrated first-year course) to provide a broadly balanced view throughout. This represents one approach to incorporating context; additional experiences are needed.
2. *How should we teach design?* While the Informatics major contains many design courses and raises design issues throughout, there is scarce literature and little consensus on how to teach design effectively. We intend to combine the teaching of "raw" techniques with extensive practice and study of existing designs. The questions are whether this will be sufficient and whether there are better options.
3. *How should we develop students' appreciation for issues of complexity and scale?* The Informatics major includes a year-long design project in which students conduct a project for a real customer. It also uses SE project simulation software for "hands-on" practice with larger, hypothetical situations in a safe setting [5]. Again, the questions are whether these are sufficient and whether there may be better options.

Answers to these questions are at the heart of a high-quality SE education. The lessons we learn as we answer them will benefit the community at large and the sooner the community sorts them out, the better.

4. ACKNOWLEDGMENTS

The Informatics major at UCI is sponsored in part by the Fund for the Improvement of Postsecondary Education (FIPSE), U.S. Department of Education.

5. REFERENCES

[1] http://www.cas.mcmaster.ca/cas/undergraduate/

[2] http://sites.computer.org/ccse/

[3] http://www.abet.org/

[4] http://www.ics.uci.edu/informatics/ugrad/

[5] http://www.ics.uci.edu/~emilyo/SimSE/