Informatics: A Focus On Computer Science In Context

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

ABSTRACT
Because the field of computer science has broadened so much in recent years, traditional degree programs are becoming crowded with new courses, each introducing its own "essential" topic. However, with more and more such courses, it is no longer possible to cover every topic in a single, coherent, four-year program. Many alternative approaches are available to address this situation. At UC Irvine, we have chosen a solution in which we offer four coordinated degree programs: a B.S. in Computer Science & Engineering, a conventional B.S. in Computer Science, a new B.S. in Informatics, and a broad overview B.S. in Information and Computer Science. Of these, the B.S. in Informatics is the most innovative, focusing on software and information design. Context plays a particularly strong role in our B.S. in Informatics: Placing software development in context is critical to the delivery of successful solutions, and we educate our students accordingly. We present our definition of informatics, detail our curriculum, describe its pedagogical characteristics and objectives, and conclude with some critical observations regarding informatics and its place in computer science education.

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

SIGCSE'05, February 23–27, 2005, St. Louis, Missouri, USA.

1. INTRODUCTION
In recent years, the field of computer science has grown tremendously in both breadth and depth. On the one hand, new subfields have emerged: bioinformatics, security, gaming, and others. On the other hand, existing subfields such as software engineering, computer networking, programming languages, and theory have grown significantly in knowledge and pedagogical approaches.

It is now recognized that undergraduate computer science degree programs can no longer cover all aspects of the field comprehensively [1]. Institutions across the country are employing a variety of approaches to designing their undergraduate computing curricula to counter this problem. Four canonical strategies may be applied: (1) survey the field at a high level, (2) provide a more configurable program, (3) lengthen the degree program, and (4) offer separate, diversified degree programs.

Survey the field at a high level. Under this strategy, students learn about a broad range of topics. A typical program may include, for example, courses in programming, computer architecture, operating systems, networking, programming languages, compilers, databases, graphics, artificial intelligence, software engineering, human-computer interaction, and social and ethical issues. Other courses may be included, usually depending on the interests of the faculty. This kind of degree program is often the result of incremental modification; as new topics emerge, the program changes to incorporate new courses on these topics. A problem with this strategy, though, is that as the number of topics increases that "any undergraduate really should know," the program strains at the seams, reducing opportunities for elective courses and sacrificing depth for breadth.

Provide a more configurable program. Under this strategy, a curriculum is partitioned into core and optional courses. In some degree programs, students may arbitrarily choose optional courses, but a more typical approach is to group optional courses into concentrations and require students to take one or more concentrations to bring some depth to their studies. While increasing flexibility, this kind of strategy also has drawbacks. As the number of optional courses rises, chains of prerequisites restrict which courses can be taken when; since a given course is seldom offered every term, care must be taken in scheduling. Students may become stuck, may simply choose concentrations based on which courses best fit in their schedules, and may not have a good grasp of how the different classes in their program complement each other to form a coherent course of study.

Lengthen the degree program. Although not an option often considered, the five-year bachelors/masters combination common in Europe is one form of lengthening the degree program. Adopting such an approach is possible, for example with a six-year bachelors/masters program. This, however, requires a commitment of time and funding that many students may be reluctant to make as they graduate high school. Moreover, economic downturns and outsourcing notwithstanding, current projections indicate a high demand for technology-skilled workers over the next decade [2]; constricting the pipeline at the front end may not be wise policy.

Offer separate, diversified degree programs. This solution strikes a balance between depth and breadth: By offering multiple degree programs, each with its own particular focus, it is possible to offer a range of options to students and still provide an in-depth education within each option. Especially when the degree programs share some courses in the first year, this option allows sufficient flexibility for students who may wish to change while at the same time providing in-depth treatment of specialized topics in each degree in the later years. (The greater the overlap of early courses across programs, the easier it is for the student to switch programs.) The drawback is, of course, that students must choose a focus early, although this is slowly but surely mitigated by the introduction of computer science in high-schools; prospective students can familiarize themselves with a range of computing topics before they enter college, enabling them to make finer subdisciplinary distinctions early in their college career.

The Donald Bren School of Information and Computer Sciences at UC Irvine has chosen this latter option: Instead of offering one configurable degree program (as it has done for many years), it now offers four closely coordinated programs. These programs allow students the choice of focusing on the lower layers of computer science (e.g., hardware design, embedded systems, computer networks, sensor networks) with a B.S. in Computer Science & Engineering (offered together with the Henry Samueli School of Engineering), on the middle layers (e.g., databases, computer systems design, theory, programming languages, artificial intelligence) with a B.S. in Computer Science, and on the upper layers (e.g., software engineering, human-computer interaction, computer-supported collaborative work, organizational information systems) with a B.S. in Informatics. In addition, they may still choose the generic, configurable B.S. in Information and Computer Science, should they desire an overview of the field rather than the in-depth exploration provided by the other three degree programs.

Here we describe the Informatics program, which we developed and now offer with support from the U.S. Department of Education's Fund for the Improvement of Post-Secondary Education (FIPSE). We designed it from the ground up to focus on the upper layers of computer science, complementing the school's other degree programs, to serve as an example of innovative curricula and effective pedagogy, and to promote inclusive participation by a broadly representative student body. This program admitted its first students in September 2004.

2. INFORMATICS

The term "informatics" has long been used in Europe to describe the entire field of computer science, from computer engineering to information systems and related fields, but in the U.S., that usage has not caught on. A small but growing number of U.S. universities are now developing new programs in informatics [3, 4, 5, 6], programs not equivalent to general computer science but concentrating instead on the upper layers of the field, moving away from a focus on computers alone to a focus on *computing in context*. Defined as the study of the design, application, use, and impact of information technology, informatics applies information technology to real world problems, designs and develops new uses for information technology, and aims to understand the impact information technology has on people [6].

To position Informatics clearly in the broader area of computing, we augment two diagrams of the Computing Curricula 2004 draft (CC 2004) [1]. Figure 1 presents the first diagram, with Informatics filling the hole that exists between "software" and "organizational needs" with "context". This signifies that Informatics builds a bridge from computer science and software engineering to information technology, a bridge that is formed by making context central to the education. The motto is that software is not developed as an isolated artifact, but is always a solution to a problem, addressing software *and* information, development *and* design, technical *and* social factors, as well as creation *and* study of implemented solutions [7].

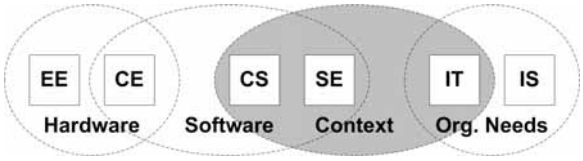

Figure 1. Informatics centers on context.

The second augmented diagram identifies the coverage of different areas of concern for Informatics. In Figure 2, we have drawn a complement to the diagrams in the CC 2004 draft that show the areas of concern for CE, CS, SE, IS, and IT. Informatics sets itself apart by focusing squarely on application domains and software development, from both a theoretical and practical perspective. The superimposed grey oval represents the additional context that Informatics addresses, both in terms of the organization and systems issues to be supported and the system infrastructures available. Of note is that Informatics encompasses most of the areas covered by the SE diagram in the CC 2004 draft. This is by design, since we believe software engineering is at the heart of Informatics. We believe that SE must be augmented to give students an adequate education for addressing real-world problems effectively. An Informatics education includes significant aspects of other disciplines, among them social science, cognitive science,

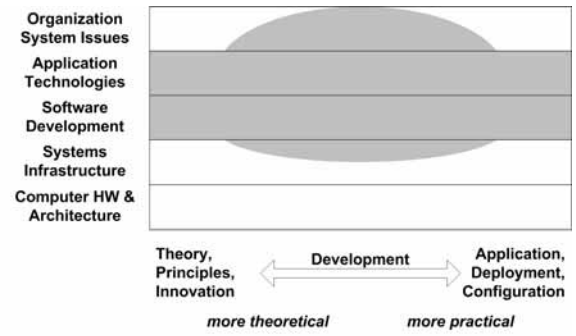

Figure 2. Areas of concern for Informatics.

computer-supported collaborative work, human-computer interaction, organizational studies, and particular application areas as well as a considerable portion of the core of computer science.

At present, the field of Informatics is still searching for the best way to educate its students. Different institutions take different approaches. At Indiana University, students take a set of core courses in computer science, followed by one or more application area specializations [6]. At the University of Washington, students focus more on the information aspects of Informatics, with courses in databases, information management, information system design, determining information needs, searching for and presenting information, etc. [3]. More experience is needed, and we anticipate that many more programs will emerge in the near term. While these may differ in their precise implementations, we predict they will fall in the outlined domain and revolve around issues of design, context, and understanding impact.

Our B.S. in Informatics distinguishes itself from existing degree programs in Informatics in two ways: (1) its solid technical foundation, building upon a very comprehensive software engineering background, and (2) an integrated social and technical approach from the beginning. Starting with the introductory course in the first year and continuing all the way through the capstone senior project course, we train students continually to examine the broader picture, develop an understanding of the problem context, and apply their technical skills to design and develop an appropriate solution. We achieve this with a curriculum that balances course sequences in software engineering, programming languages, human-computer interaction, organizational computing, and databases with a variety of individual courses in project management, computer-supported collaborative work, information retrieval, information visualization, and so on.

3. INFORMATICS CURRICULUM

Our new Informatics curriculum, shown in Table 1, is designed in accordance with the definition and observations above. Our courses all run on the quarter system. We marked each course as follows: unmarked courses are taught and specified by Department of Informatics faculty; courses marked with (cs) originate in the other degree programs in the school and are typically offered by faculty in its other departments; courses marked with (o) discuss non–computer-science topics; and courses marked with (b) signify electives or courses that satisfy the university's breadth (general education) requirements.

We designed our program from scratch, without being limited by existing courses. We did take advantage of existing courses when they fit our vision, but we were not constrained in any way. The result is a curriculum consisting of fourteen new and thirteen existing courses in computing, as well as three mathematics courses and the required breadth (general education) courses.

The first year provides students with a hands-on introduction to the broad field of Informatics, anchored by the new core course. This three-quarter sequence introduces students to the Informatics philosophy of considering context from their first quarter in the program. We want the students to develop a mindset in which context and design come first, rather than starting off with a focus on programming alone. The course does have a strong emphasis on writing programs in the functional and object-oriented styles (as does the rest of the curriculum; students will practice more programming than is common in most CS programs), but the assignments and class meetings put contextualized problem solving and design first and programming second.

The first year also teaches students formal reasoning (logic) and problem solving skills through a sequence of three courses: abstract reasoning, discrete math, and data structures analysis and implementation. Together with the statistics course in the second year, the materials introduced in these courses lay the mathematical foundation for the rest of the program.

The second year builds up a portfolio of foundational techniques and skills that further establish the discipline of Informatics and provide a "toolbox" that students will use in future years to solve large-scale information and software design problems. Courses discuss the role of various programming languages and conceptual approaches (including how special-purpose languages can help solve certain problems elegantly); introduce user interface design (from the multiple perspectives of theory, established practices, and hands-on development); present software engineering methods, notations, and tools; and establish the roles of requirements elicitation and quality assurance in successfully carrying out a system design and development project.

The third year builds upon the foundational techniques and skills introduced in the second year; it covers information and software design from two different but related perspectives. First, there is a three-course series that describes how information and software design affect the real world, i.e., the social and organizational context in which a solution is ultimately placed. A second three-course sequence (actually starting in the last quarter of the second year) introduces technical approaches to design and large-scale problem solving with software. Combined, the two sequences present a comprehensive overview of design from both a technical and social perspective.

The fourth year is built around a year-long capstone project in which groups of students address a significant project, typically from an outside client. In addressing this project, students must bring together materials from previous years (tools, skills, processes, ethnographic methods, design approaches, and many others) to complete their project successfully. The fourth year also includes more advanced courses on databases, information retrieval, information visualization, project management, and computer-supported cooperative work.

This program, unlike many current computer science and engineering curricula, affords students the flexibility to take elective courses or to undertake undergraduate research projects. Providing these opportunities allows students to pursue interdisciplinary interests and maintains their enthusiasm and motivation.

The program has several distinguishing characteristics:

- *A smaller number of math courses.* Traditional computer science programs often include math courses intended mainly to enhance students' "mathematical maturity" and formal reasoning skills. While we value mathematics as a discipline and accept that math helps students develop reasoning skills, we designed our program to include only those math or other foundational courses that contribute directly to an understanding of software and information design. We believe students can develop critical thinking skills in many ways and we particularly advocate building these skills in the domain

in which they will be applied. Hence, many of our courses are structured to require formal or quantitative reasoning, weighing of alternatives, and creative thinking. A good example is the software design series, which emphasizes creative problem solving and designing appropriate solutions to challenging problems.

- *An interdisciplinary approach.* Our Informatics major addresses the broad set of issues surrounding design, including initial requirements gathering, estimating and measuring the impact of alternative solutions, and implementing those solutions—all from a multi-disciplinary perspective that includes computer science, information science, organizational science, social science, cognitive science, and others. It is not sufficient to teach mechanical design notations and principles; these must be placed in context, examined from multiple perspectives, and honed and practiced to develop the designer's ability to propose solutions that effectively solve the problem at hand. We address these issues throughout, particularly in the third-year sequence on social and organizational impact and the fourth-year senior design project.

- *A focus on design.* As the linchpin of our curriculum, we take a distinctly design-oriented approach to the materials in the Informatics program. Traditionally, design is underrepresented in current computer science curricula; at best, a typical software engineering class introduces notations and lets students practice, at best, a few designs. With the exception of some specific software engineering programs [8, 9], there simply is no room in the curriculum to teach additional material and practice more. Our curriculum turns this notion on its head. We introduce design from the beginning, have multiple course sequences on the topic, examine it from a multidisciplinary perspective, and promote extensive practice in actually creating high-quality designs. Even compared to existing software engineering degree programs, this is a much broader and more in-depth treatment.

We anticipate that our students will be able to function in a variety of different jobs. They will be familiar with all aspects of the software development process, from initial requirements gathering to the delivery of a solution. They will know that software is merely a part of an overall solution that addresses the information that an organization manages, manipulates, and visualizes. They will design and develop integrated software and information systems. They will be technically solid. They will know that their activities have both a technical and social aspect, and know how to create and also analyze solutions. In sum, they will be prepared to deal with real-world problems in context and approach them from an informed computing perspective.

4. PEDAGOGICAL CHARACTERISTICS

We designed the Informatics major to incorporate a set of best pedagogical practices. To meet the program's goals, we felt it was critical not just to provide the right set of courses but also to address the pedagogy for structuring, sequencing, and delivering those courses. Not surprisingly, then, the curriculum makes wide use of multi-course sequences. These course sequences provide continuity, help place the topics in their broader context, and provide added depth as one topic builds upon another. In our software design sequence, for example, students in the first course ("Design I") are introduced to software design notations and principles, refinement into code, architectural styles, and design patterns. The second course ("Design II") broadens the study to large-scale systems, reuse, product families, real-time systems, and application frameworks. Finally, the third course ("Software Architecture, Distributed Systems, and Interoperability") expands the repertoire to distributed, decentralized design.

Group work is often required in our courses. In the first year, we start with pair programming and build that up in later courses to larger group projects. In the final year, groups of students participate in a year-long senior design project. We support this teamwork by addressing the tools and approaches necessary to manage the work effectively. The second-year "Methods and Tools" course, for instance, introduces tools for such tasks as configuration management, bug tracking, and process management. In the senior year, students take a full course in project management as they start on their senior design project.

Table 1. Required Curriculum for the UC Irvine B.S. in Informatics.

	Fall	***Winter***	***Spring***
First	Informatics Core	Informatics Core	Informatics Core
			Informatics Research Topics Seminar
	(o) Critical Reasoning	(o) Discrete Mathematics	(cs) Fundamental Data Structures
	(b) Writing	(b) Writing	(b) Writing
Second	(o) Statistics	Human-Computer Interaction	Project in HCI & User Interfaces
	(cs) Concepts Programming Languages I	Concepts Programming Languages II	Software Design I
	Software Methods and Tools	Requirements Analysis & Engineering	SW Specification & Quality Engineering
	(b) Breadth	(b) Breadth	(b) Breadth
Third	Social Analysis of Computerization	Organizational Information Systems	Project Social/Org. Impacts of Computing
	Software Design II	SW Arch, Dist. Sys., & Interoperability	(cs) File and Database Management
	(b) Breadth	(b) Breadth	(b) Breadth
	(b) Breadth / Elective	(b) Breadth / Elective	(b) Breadth / Elective
Fourth	Senior Design Project	Senior Design Project	Senior Design Project
	(cs) Project in Database Management	(cs) Information Retrieval	Information Visualization
	Project Management	CSCW	(b) Breadth/ Elective
	(b) Breadth / Elective	(b) Breadth / Elective	(b) Breadth / Elective

We use case studies to provide realistic, practical experiences to our students. This starts with a case study of a web store that we introduce in the first quarter. Students are not expected to build the entire store, of course, but will build individual components that must be integrated with the existing implementation. Having access to the full working example, however, is critical, as it provides relevancy and forms a context for the concepts that are taught. Moreover, students can freely explore other aspects of the application and interesting problems arise that frame the theory of design and allow practical examples of the theory. These case studies occur throughout the program. The design courses, for instance, study and dissect designs of actual systems. The senior design course is, in some ways, a large, experiential case study with an actual customer, in which the students must manage, design, and implement an entire project from start to finish.

Each year culminates with a project course in which a particular case study wraps up the year. We will have showcase days in which students at all levels in the program present their projects to a public audience that includes representatives from local industry. Such industrial contact further motivates the students.

We use varied teaching approaches throughout the curriculum. Principled use of case studies, as described above, puts us in the domain of problem-based learning. A speaker series in the first year introduces students to the broad topics and research problems in Informatics, further enhancing the context. We dissect, study, and analyze from a design perspective actual software and information systems. Real customers provide the projects for the senior design course. Advanced software engineering simulations let students work through more aspects of the software development process than they could experience in the field. Together, these and other approaches provide the students with an engaging experience that is clearly related to their future endeavors.

Putting all these pieces into place requires a significant commitment from the faculty with a huge potential payoff. Specifically, the above factors—increased program coherence, ties to realistic problems, and practical, creative exercises accompanying theoretical course materials—are among the strongest factors contributing to increased participation of underrepresented populations in CS programs [10].

5. CONCLUSIONS

We created this B.S. in Informatics degree with three goals: (1) build a complementary degree program focused on both software and information design, (2) develop an exemplary program with effective pedagogy and an engaging curriculum emphasizing real-world problems and creative solutions, and (3) promote access to the program, retention, and degree completion for a broadly representative group of students.

This major (and the other new majors in the school) were proposed, approved by the campus, and implemented successfully in a very cooperative climate. Incoming students have chosen among the programs, with an initial first-year class of 31 in Informatics (out of approximately 180 first-year students schoolwide).

The program has the strongest technical component of any current U.S. Informatics program, befitting its home in an information and computer science school with a long tradition of technical excellence. The program meets a particular real-world need for broadly trained system designers with strong implementation skills [2]. It also resists outsourcing, since high-level design requires intensive interaction with clients and an understanding of their organizational, social, and cultural context, all of which is hard to achieve from halfway around the world. It should particularly attract students with more interest in designing solutions to real problems than in designing smaller, faster, and cheaper computers.

With enrollments in computer science programs currently trending downwards, schools must explore ways of attracting more students to the field. The approach taken at UC Irvine is to diversify degree offerings, providing programs tailored to diverse students' goals and interests. Our B.S. degree in Informatics is one part of this strategy, offering a novel combination of concepts and skills that expands the range of computing curricula.

6. MORE INFORMATION

More information about the UC Irvine B.S. in Informatics can be found at: http://www.ics.uci.edu/informatics.

7. ACKNOWLEDGMENTS

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

8. REFERENCES

[1] ACM, AIS, and IEEE-CS Joint Task force for Computing Curricula 2004, *Computing Curricula 2004,* http://www.acm.org/education/curricula.html.

[2] U.S. Bureau of Labor Statistics, *2002-2012 Employment Projection*, http://www.bls.gov/news.release/ecopro.nr0.htm.

[3] University of Washington Information School, B.S. of Science in Informatics, http://www.ischool.washington.edu.

[4] York College of Pennsylvania, B.S. of Informatics, http://www.ycp.edu/academics/.

[5] Montclair State University Department of Computer Science, B.S. in Science Informatics, http://cs.montclair.edu/undergraduate.html.

[6] Indiana University School of Informatics, B.S. of Informatics, http://www.informatics.indiana.edu/.

[7] A. van der Hoek, D.G. Kay, and D.J. Richardson, *A B.S. in Informatics: Contextualizing Software Engineering Education* (in submission).

[8] Rochester Institute of Technology Department of Software Engineering, B.S in Software Engineering, http://www.se.rit.edu/degrees.html.

[9] Milwaukee School of Engineering, B.S. in Software Engineering, http://www.msoe.edu/eecs/se/.

[10] Margolis and Fischer, *Unlocking the Clubhouse: Women in Computing.* Cambridge: MIT Press, 2001