ABSTRACT

The National Science Foundation has noted declining enrollments in computing, increasing workforce needs for graduates, and changes in the computing education needed in the future. They thus sponsored four regional workshops which each suggested ways to increase enrollments: multiple entry points to the undergraduate major, better presentation of computing careers, interdisciplinary courses and projects, and innovative approaches to computer science (CS) minors. This paper describes a quarter-long full time interdisciplinary program for entry-level mathematics, science, and CS students, designed to broaden the current curriculum in response to a drop in CS enrollments and thus to better retain majors and attract students from other disciplines as minors. The program met several strategic directions identified by the NSF workshops. Organized around a forest ecology case study, the program integrated an entry level study of CS with statistics, ecology, and the philosophy and history of science. This paper describes that program, and suggests how the curricular design and materials are exportable to other institutions as linked courses or an interdisciplinary case study.

SECTION 1. WHY INTERDISCIPLINARY COMPUTER SCIENCE EDUCATION?

In response to declining computing enrollments, the Computing and Information Science and Engineering (CISE) Directorate at the National Science Foundation (NSF) funded four regional workshops to identify strategies for increasing and maintaining enrollment, and better meeting human and infrastructure needs. These efforts, the Integrative Computing Education and Research (ICER) Initiative, developed strategies to maintain the computing workforce for the nation’s economic, cultural, and democratic vitality. Each workshop included 20-35 educators from a broad range of institutions and stakeholder groups, and made recommendations to NSF. The Northwestern Regional ICER workshop held at Stanford on January 27-28, 2006, recommended five strategic initiatives:

Strategic Initiative 1: Improve the quality of computing education with repositories for curricular artifacts and encouraging curricular experimentation and innovation.

Strategic Initiative 2: Attract more people to the field through changing the image of computing, outreach to high-schools, and better understanding students via focus groups.
Strategic Initiative 3: Improve retention by recasting introductory courses as “pumps, not filters;” helping students see career opportunities; increasing student research, teaching, and collaborative work; and encouraging support groups for at-risk students.

Strategic Initiative 4: Strengthen interdisciplinary connections by offering courses for students in other disciplines, encouraging collaborative activities across disciplines for faculty, and developing and distributing interdisciplinary curricula.

Strategic Initiative 5: Meet human and infrastructure needs by eliminating structural barriers to interdisciplinary collaboration; providing institutional rewards and support for teaching, curriculum development and assessment; helping faculty think strategically inside and outside the academy, and strengthening industry involvement.

Other recent exhortations to reinvent the CS curriculum suggest similar experimentation with interdisciplinary connections, e.g., Shneiderman’s work with multimedia and CS [10].

This paper describes an interdisciplinary curricular program at The Evergreen State College that addresses Northwest ICER Initiatives 2, 3, and 4. After presenting background on Evergreen, we describe a quarter-long full time program that integrated computer science, ecology, mathematics, and the philosophy and history of science around a real world case study – an NSF funded ecology field study of forest and tree structure-function at eight Pacific Northwest sites ranging in age from 85 to 950 years. We describe initial assessment efforts, and, in conclusion, recount lessons learned and ideas for dissemination.

SECTION 2: A PROBLEM-ORIENTED COMPUTER SCIENCE CURRICULUM
The Evergreen State College is a fully accredited state-supported college founded in 1971. The primary curricular mode at Evergreen is interdisciplinary, team-taught, full time programs organized around a central theme. Faculty “translate” work within programs into quarter-credit-hours similar to those elsewhere, but are encouraged to unify disciplinary study around real world problems or program themes, integrate writing, and use active learning, e.g., small group discussions, workshops, or team projects. Also known as learning communities, this educational philosophy is promoted by The Washington Center for the Improving the Quality of Undergraduate Education, which helps non-Evergreen faculty organize interdisciplinary teaching teams and establish learning communities.

Prior to soaring CS enrollments in the 1980’s, Evergreen’s curriculum included several broad interdisciplinary computing programs; eliminating these when enrollments grew freed CS faculty to teach the “major” CS programs which were then jointly taught by faculty both in CS and related disciplines. When CS enrollments dropped in the late 1990’s, programs previously staffed by two or three faculty from mathematics, philosophy, psychology, management, as well as CS, were staffed by at most 1.5 CS faculty FTE. Thus, although the curricular model facilitated interdisciplinary computer science, the boom/bust cycle of the 80’s-90’s precluded it. The three CS programs maintained their problem-oriented focus (Data to Information, developing software in Student Originated Software, and the nature of computation in Computability), but lost their interdisciplinary emphasis. The lack of interdisciplinarity had two effects: with no place to learn about computing

---

See [http://www.evergreen.edu/washcenter](http://www.evergreen.edu/washcenter).
beyond word processing or spreadsheet training, 1) students without prior programming expertise avoided the discipline, and 2) students and faculty outside the area had little understanding of CS as a discipline. Computing became a mono-culture with its intellectual foundations not well understood by those outside.

SECTION 3: A NEW INTERDISCIPLINARY INTRODUCTION TO CS

In spring 2005, given falling CS enrollments, faculty contrived a curricular experiment to broaden the first course. The year-long entry level curriculum was divided into two programs: a new one-quarter program to show how scientists use computing, and a new CS-focused two-quarter program: Computer Science Foundations. The one-quarter introduction was dubbed Data and Information: Quantitative Ecology, with the idea that the domain qualifier could be replaced in future years with different sciences according to other unifying themes and faculty-student interest. Objectives of the fall interdisciplinary program were to: 1) give an intellectually invigorating but broad, gentle introduction to CS, 2) level the playing field among students with varying computing experience, 3) introduce CS to students who had not considered it as a major or even a minor, and 3) bring students together early in their college careers into a “learning community” — with field trips, theme-based labs and seminars, and team-based projects — and thus to increase retention in CS.

Data and Information: Quantitative Ecology had four components, each roughly equivalent to a 4-unit course: Philosophy and History of Data-Driven Science (seminar), Introduction to Programming in Python, Introduction to Statistics using R, and an Ecology Case Study. It was directed at three constituencies: computer science, math, and natural science students. The unifying theme was data-driven science and the role information technology and computation play therein. We wanted students to understand that falling computing costs and large archives of field- and remotely-sensed- data create new problems in ecology, and that difficulties faced by scientists in using computers are often due to a lack of appropriate software or expertise. A major sub-theme was that solving these problems would require teams with ecology, computing, and statistical and mathematical expertise. Program themes were presented in the case study and seminar, the former, an introduction to data collection and management, a practicum for programming and statistics in laboratories during weeks 1-8, and a two-week team project. Seminar provided opportunities to think more broadly about the nature of science and the place of data-driven science in that spectrum than the focused case study allowed.

The programming, statistics, and seminar components were similar to traditional courses. The computer science component was taught using Python [1], which created a self-contained 1-quarter introduction to programming. Because Python offers both a gentle learning curve and powerful features and libraries, including many suitable for data manipulation such as lists, associative arrays, and dictionaries, it was ideal for this teaching. Our experience with Python corroborated Guzdial’s experiments, which used Python to teach multimedia and programming in an interdisciplinary introductory course. He subsequently observed that this approach attracted students to computer science courses [5]. The mathematical component focused on statistics [4] because data-driven science typically requires study design, data collection, and analysis. R was introduced mid-quarter. Because we emphasized translating practical problems into a statistical framework rather than
techniques and tests, we worked with student teams on projects to provide tools for data analysis. In seminar, students read about the genesis of modern science, and the role of data and computation in science [2,3,6,7,11]. While the scientific focus in the case study was ecology, seminar addressed natural and physical sciences generally. We considered the interplay among scientific and socio-political realms, and also aimed to improve students’ oral and written communication skills, and their reading and critical reasoning ability.

Students used programming and statistics for analysis of an ecological case study – a 1000 year chronosequence (1kcs) of eight Pacific Northwest forests that range in age from 50 to 950 years [13]. Topics included the human-factors of presenting information [12], descriptive data analysis (distribution, measures of central tendency, correlation of key variables by species), species diversity, and data visualization. Students worked in teams to complete weekly laboratory assignments, attended one weekly lecture, and participated in a field trip to take field measurements which they later compared to those taken by scientists. Labs included both programming and statistical analysis, providing ideas and preparing students for project work, project topics included study design, data analysis and validation, scientific visualization (e.g., the distribution of tree heights and tree density), and web-based interfaces for scientists to access or archive data. To give adequate time for project work, we started projects in the fifth week; finished the statistics, programming, and seminar in the first eight weeks; and devoted the last two weeks to project work and presentation.

There were seven case study labs (weeks 1 and 2-8) and one field trip (week 2). One of the program faculty, co-principle investigator on the NSF grant that supported the ecology and computer science research that developed the 1kcs database, wrote the labs. The 1kcs database consisted of eight MS Access databases, one for each site, each with the same schema, holding information about the site, trees (e.g., height, diameter, species, crown projection), branches (15-20 trees per site were climbed and every branch measured: height, length, aspect) understory (plants on the ground), rainfall and throughfall (how much rain reached the ground under the trees), and light penetration through the forest canopy.

Each lab consisted of 1) a statement of learning objectives, 2) how to find files needed to complete the lab such as data, sample Python programs, and R scripts, 3) detailed directions for submitting lab results, 4) directions for carrying out the lab, and 5) questions for students to answer while doing the lab. Most labs had three parts: 1) a warm-up (questions and activities every student should have been able to complete), 2) a main part (questions and activities that constituted the desired learning for the lab, and 3) activities that faculty did not expect students to complete. Many of the third kind of lab activity grew into student term projects. Most data files were given to the students as excel spreadsheets or flat files, to facilitate their completing the labs. Lab topics are listed below:

1. Evaluate illustrations from a pre-publication description of the 1kcs study design.
2. Working as the field trip team, enter data collected on the field trip, and use histograms to compare those to data collected by professional scientists.
3. Perform descriptive statistics to determine if certain stand and tree characteristics differ as forest stands age. This involved both running simple statistics in excel (or other

---

4 See http://www2.evergreen.edu/quantecology for detailed information about the program, incl. teaching materials.
application) and writing a Python program to calculate the Shannon Diversity Index.

4. Study, implement in Python, and run some measures of stand structure, including basal area, branch count and branch density, canopy cover, and gap.

5. Program in Python a simple stem map visualization, and use R to plot and correlate tree diameter and height, by species and by stand age.

6. Write a specification for refining the stem map Python program (Lab 5), and extend the Lab 5 work in R to determine when two populations are statistically different.

7. Use R to run a chi square test and design a test for a new ecology question.

SECTION 4: RETENTION AND ASSESSMENT.

Most students (24 of 36) came to the program for an introduction to CS; five natural science students came to improve quantitative skills. We had a range of students: 6 freshmen, 9 sophomores, 13 juniors, 7 seniors and 1 post baccalaureate; 7 with prior preparation in CS. 23 continued in Computer Science Foundations. One ecology student took Foundations in spring only. Thus, though the program attracted only one student to CS who would not otherwise have gone, it possibly improved retention by almost 20% (Table 1). Preliminary analysis of the college’s assessment of student quantitative skills indicates that those who took this program were not different from those, in the sciences, who did not.

<table>
<thead>
<tr>
<th>Enrollment Statistics</th>
<th>fall</th>
<th>winter</th>
<th>spring</th>
</tr>
</thead>
<tbody>
<tr>
<td>2006-7 CSF students</td>
<td>36: 7</td>
<td>23</td>
<td>1</td>
</tr>
<tr>
<td>…from intro CS in</td>
<td>prior (adv) CSF</td>
<td>96%</td>
<td>75%</td>
</tr>
<tr>
<td>Quantitative Ecology</td>
<td>5 adv. Non-CS</td>
<td></td>
<td>18, 75% (79% from F)</td>
</tr>
<tr>
<td>2005-6 CSF students</td>
<td>27</td>
<td>21</td>
<td>16</td>
</tr>
<tr>
<td>…from other programs</td>
<td>23</td>
<td>78%</td>
<td>66%</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>52%</td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Student retention in first year computer science (CSF).

SECTION 5: LESSONS LEARNED, DISSEMINATION, AND CONCLUSIONS

Aside from minor changes in program design such as starting R earlier, we found the program essentially sound. Student projects were among the best faculty had seen for a one quarter course with a major course load, and most students noted that the program and project were successful academic experiences. We make three observations: 1) the case study could be used as an independent interdisciplinary course with pre- or co-requisites, 2) the case study with weekly paired labs, a field trip, and a team project created both respect for expertise of students from other disciplines and camaraderie among students who later took the more intense CS Foundations, and 3) effective assessment requires pre-planning and is much easier with institution-wide programs that provide timely feedback to faculty.

Our program differs from other similar programs, e.g., Univ. Montana’s Computational Biology or UC Berkeley’s Mechanisms of an Organism in that our faculty worked collaboratively on the entire program – explicitly integrating the three disciplines around program themes based on faculty interests. A second aspect of collaboration is student to student; working together encouraged creativity and the tackling of more difficult problems. Pair design fits well into this framework [9], as two people working on a problem often complement each other, especially if at a similar skill level. In addition, the social
interaction counteracts some of the isolating aspects of CS and math studies.

We believe that institutions that do not offer linked courses or full time programs could gain some of the benefit of interdisciplinary experience such as this program offered by offering an applied case-study-oriented course, perhaps with the two course components as prerequisites. Such broadening of introductory CS programs might increase retention, attract additional students to CS as minors or majors, and provide ways for computer science faculty to educate students and faculty outside the area about the discipline. A pilot course of the case study component will be offered in the summer 2007.

REFERENCES