

# Report on the Undergraduate Curriculum

Prepared For: The Department of Computer and Information Science, University of Pennsylvania

Prepared By: The Dining Philosophers Executive Board ([dp-exec@googlegroups.com](mailto:dp-exec@googlegroups.com))

March 2, 2008

# Summary

## Introduction

On December 5, 2007, The Dining Philosophers Executive Board hosted a roundtable discussion, open—and limited—to all undergraduates in the Department of Computer and Information Science, on the state of the department's undergraduate curriculum. The 25 students in attendance included members of all years, of both genders, and representing all three primary curriculum tracks (CSCI/CSE, DMD, and ASCS). While the Board had prepared an agenda in advance, the moderator was careful to allow everyone to voice their concerns as they would and only to change topics once the Board had heard from everyone who wished to speak. Following a careful review of its notes from the event and a synthesis of their content, the Board presents a set of recommendations, both major and minor, concerning how the department might improve the quality of the undergraduate education it delivers. To the best determination of all involved in its preparation, the content of this report is an accurate representation of what was said.

## Issues

At a high level, the central issues in the discussion were as follows:

1. **The Rigor of the Curriculum:** Does the curriculum as it stands provide adequate instruction in the various areas of computer science, particularly with regards to the challenges graduates will face in their post-baccalaureate pursuits? What kinds of courses should and should not the department offer and/or require in order to ensure this? At what caliber of student are the courses aimed? Is there accommodation for ambitious students who want to study topics in more detail than is permitted by ordinary undergraduate courses?
2. **The Faculty:** To what degree are the faculty engaged with undergraduates, both in the classroom and otherwise? What is the overall quality of the teaching? How knowledgeable and accessible are professors as advisors and as facilitators of undergraduate research? Are any faculty issues limiting course offerings?
3. **The Structure of the Major:** How can the major excite students at an early stage about continuing with more advanced courses in the program? Are the transitions smooth between the courses in the introductory programming sequence (110→120→121)? Is there a place for non-majors who want to learn about computer science but not to compete directly with majors?

## Recommendations

The major conclusions of the discussion, in no particular order, are as follows:

- A. **Add more instruction in the practical applications and implementations of abstract concepts, particularly to courses in the theory sequence (260→262→320).**
- B. **Introduce the C programming language early, perhaps even in 120, so that students are prepared to use it heavily in 380 (Operating Systems).**
- C. **Offer regularly elective courses that have historically had high enrollments, such as 341 (Compilers) and, more recently, Prof. Pierce's 552 (Advanced Functional Programming).**
- D. **Create a new introductory class that exposes students to all—or at least many—of the areas of computer science in order to stimulate interest in further study.**
- E. **Adjust the difficulty level of 110 and 120 to erase the large gap in difficulty between them, and work to integrate more fully the programming and theory aspects of the curriculum.**
- F. **Increase awareness of what the department offers: remove old courses from the register, market graduate seminars and research opportunities to undergraduates, and ensure that prerequisites are explicit and reasonable. Advisors *must* be knowledgeable resources.**

# In Detail

## **A.**

Currently, many students feel poorly equipped to apply their knowledge of algorithm design and theory of computation to real-world problems, if indeed they can see any potential applications at all. 121 may aim to do this, but if so it is not succeeding. Brief programming assignments involving implementation of theory topics in 260, 262, and 320 would go a long way towards both giving students greater confidence in their understanding and showing them the value and relevance of these aspects of the discipline.

## **B.**

The C programming language receives only a cursory introduction in 240 (Introduction to Architecture), while in 380/381 students must complete a project that makes intensive and substantial use of it. Lacking a proper foundation, students at present must scramble to fill the gaps in their knowledge before they can begin to work on the project proper, adding needlessly to the difficulty level of the course. C should most likely be a topic in 120 so that 240 can complete the preparation for 380/381 with respect to the language.

## **C.**

Student interest is high in seeing more upper-level seminars on specialized topics, with 341 and 552 cited as being especially popular courses of this type; 341 is of particular note as the class has not been offered since Spring 2006 owing to the departure of Prof. Lewis. If any current courses no longer attract students, the department should investigate revising or replacing them. Above all, excellence in teaching is of paramount importance for student engagement and interest in courses; from an undergraduate perspective, this is far more critical to a faculty member's quality than is his or her research output and level of outside funding.

## **D.**

This recommendation is the amalgamation of several ideas. Some students support the division of the curriculum into "tracks," or collections of electives grouped around themes such as AI, hardware, programming languages, and others. Many also complain that the department may be losing potential majors who decide on the basis of 110 and 120 that they have no interest in computer science but have no knowledge of the other aspects of the discipline. A final concern is that minors and non-majors currently take the same classes as majors, which may be a greater challenge than they want. In all cases, the proposed course would give students a sense of the diversity in the field before requiring that they commit to more substantial coursework.

## **E.**

Simply put, 110 is currently too easy while 120 is too hard, which can make the latter overwhelming for less committed and motivated students. In the interest of promoting student success, this discrepancy must be reduced or eliminated. Even if this problem is solved, however, the two main halves of the major—programming and theory—have very little to bind them. Both kinds of courses should reinforce and supplement the material in the other (this ties into **A**).

## **F.**

The department has a great variety of offerings, many of potential interest to undergraduates, but there is no clear, central, reliable means of discovering what is available. Many courses on the department register have not been taught recently, and those that are there may have descriptions that are outdated or incorrect. Research and graduate seminars are of great interest to advanced undergraduates, particularly those who have exhausted the undergraduate courses in particular areas; professors should publicize the seminars they teach and any opportunities they have for undergraduates on their current research projects. Advisors—and, ideally, the other faculty as well—should be expert on all of the above, with rigorous standards in place to ensure that students can trust them as authorities.

# Conclusion

In closing, the Board would like to offer for consideration three matters that it believes should inform any action taken in response to the above recommendations:

1. Computer science attracts many different kinds of students with academic and professional goals that can vary wildly, as evidenced by the myriad degree plans available in the department. Recognizing this, courses, particularly the introductory required for all degree plans, must address this diversity of the population in terms of topics and coursework: one size does not fit all, yet the curriculum now operates as if this is the case.
2. Feedback, formal and otherwise, is the best means of determining what is and is not working well in any aspect of the department. Consequently, the faculty must both take current feedback seriously, particularly course evaluations—see the discussion of recommendation **C**—and work to gauge student opinion on any revisions or new initiatives it undertakes with regards to the undergraduate curriculum, preferably even before it implements them. Such practices would be an effective means of fulfilling its obligations for ABET accreditation, chiefly that the department “regularly assesses its progress against its objectives and uses the results of the assessments to identify program improvements and to modify the program’s objectives” (*2007-2008 Criteria for Accrediting Computer Science Programs*).
3. The department is exceptionally strong in many areas, especially programming languages and machine learning. It is to everyone’s advantage that the curriculum leverage these strengths as much as possible: while it should certainly offer a well-rounded selection of courses, special electives and research opportunities in these areas would both excite students about studying them and distinguish the department from its peers.

It may seem as if this report only points out deficiencies in and makes complaints about the current state of the curriculum, but the Board firmly believes that this is the case only because of the extremely high expectations students have of the department and of the university as a whole. The department is strong now, and putting these recommendations into practice can only make it stronger.

Respectfully,

---

Matthew Evans, President

---

Jeff Weinstein, Vice President

---

David Winchell, Secretary