A PHILOSOPY OF COMPUTER SCIENCE COURSE FOR COMPUTING PRACTITIONERS

Extended Abstract

Matti Tedre
University of Joensuu, Department of Computer Science and Statistics
P.O.Box 111, FIN-80110 Joensuu, Finland
firstname.surname@cs.joensuu.fi

1. BACKGROUND

Computer science is a relatively young discipline, and the short history of electronic digital computing is loaded with a great variety of different approaches, definitions, and outlooks on computer science. Computer science has been woven together from a number of different disciplinary strands, yet the resulting science offers a variety of unique ways of explaining phenomena, such as computational models and algorithms. Interdisciplinarity is an integral part of computer science, but it also makes normative and descriptive statements about computer science difficult. An overarching account or a set of rules for computer science research should cover fields such as software engineering, complexity theory, usability, the psychology of programming, management information systems, virtual reality, and architectural design.

It is important for computer scientists to understand the challenges and possibilities that the vast diversity of computer science research can cause. Many disputes and misunderstandings between computer scientists from different branches could be avoided by appreciating the different views of what computer science is and how computer scientists should work. Even more importantly, computer scientists should know that there are no generic approach suitable for all subjects in computer science. Mathematical and computational models are precise and unambiguous, yet they fail to capture the richness of physical and social reality. Narratives and ethnographies are rich in dimensions and sensitive to detail, yet equivocal and context-dependent.

To cope with the variety of topics in computer science, computer scientists employ a vast diversity of research methods (e.g., Tichy et al., 1995; Glass et al., 2004). However, it seems that computer science education regularly fails to provide computer scientists the methodological and disciplinary understanding that interdisciplinary work requires. It has been argued that the typical computing researcher learns his or her research skills from a sort of master-apprentice relationships with his or her professors and from examining successful prior research (Glass, 1995). The official ACM/IEEE curriculum recommendations (Denning et al., 2001) do not include a course on methodology, research design, or research paradigms. In addition, it has been argued that computer scientists publish relatively few papers with experimentally validated results, and that a description of methodology is often missing from research reports in computer science (Tichy et al., 1995; Vessey et al., 2002).

Many computer scientists can justly be offended by such unfair accusations as the ones above. Certainly there must be many academic institutions in which computer scientists are given proper, formal training on research design, research paradigms, work in interdisciplinary fields, epistemological and methodological considerations, and so forth. Certainly many computer scientists meticulously report their research methodology in their publications. And certainly, many computer scientists are knowledgeable with the ontological and epistemological underpinnings of the particular research methodologies they utilize. At the department of computer science and statistics, University of Joensuu, we have come to a conclusion that it is a proper part of computer scientist's education to be aware of epistemological and methodological issues in computer science. Hence, we have included, in our curriculum, a course that deals with those issues—issues that fall in the domain of the philosophy of computer science. In this extended abstract we sketch the contents of our course and its aims (a detailed description of our course can be found at Tedre, *s.a.*).

2. THE PHILOSOPHY OF COMPUTER SCIENCE ELSEWHERE

The term *philosophy of computer science* can be found in a number of places, in different meanings. In addition to original literary contributions (e.g., Colburn, 2000:III), special issue of *The Monist* (vol.82, no.1), E-CAP conferences, and dedicated web sites (e.g., http://pcs.essex.ac.uk/), the term has been used in a number of university courses worldwide. Take, for instance, William J. Rapaport's course at the State University of New York at Buffalo, USA (Rapaport, 2005), Gordana Dodig-Crnkovic's course for students at Swedish universities (Dodig-Crnkovic, 2006), and Konstantine Arkoudas' course at Rensselaer Polytechnic Institute, USA. If one expands the search terms a bit, one can easily find courses on the philosophy of computing, the philosophy of artificial intelligence, the philosophy of the mind, the philosophy of information, and so forth. However, those courses are a bit far from our concerns. In this course, the philosophy of computer science is considered as a philosophy *of* a specific discipline similar to the philosophy of physics, the philosophy of biology, and the philosophy of mathematics (cf. Colburn, 2000:129-131; Shapiro, 2000:*vii*). This course is specifically aimed at practically oriented computer scientists, not at people specializing in philosophical issues that surround computing.

We concur with William J. Rapaport (2005) in that although a number of outstanding monographs and anthologies in the philosophy of computing has been published, good textbook candidates are rare (especially for the purposes and aims of this course). On top of that, since giving non-native English speakers a large number of original texts in English seemed unreasonable, we decided to prepare course readings about a number of central themes in the course. This way we were able to pull together the essence of a good number of original texts without giving an undue burden of translation on the students (the intellectual content of the course causes enough burden!). The course readings describe, in English, the cruces of the topics included in this course, yet the readings present those crucial points in a more compact and hopefully easier form than the original texts. Our course readings are publicly available under the creative commons license (Tedre, 2007).

3. THE COURSE IN THE UNIVERSITY OF JOENSUU

In this course the focus is practical and straightforward in the sense that we deal with issues that can make a difference in our students' work, research, and writing, and in the sense that we try not to delve too deep into sophisticated speculations. We deal with philosophical issues that directly concern the everyday work of computer scientists. Our course, titled "The Philosophy of Computer Science", is an online course with no contact teaching. Students in two Finnish universities—the University of Joensuu and the University of Kuopio—are entitled to participate in the course. Online course is the only viable alternative for a course that is held simultaneously for students in spatially distant universities in a relatively large and sparsely inhabited country. The course involves weekly tasks, which include reading, writing, reflecting, and commenting on other students' writing.

The course is targeted for graduate (M.Sc) and postgraduate (PhD) students who have studied computer science long enough to have a good idea of research in computer science. Graduate students are required to have a M.Sc thesis topic chosen, and their coursework includes an in-depth analysis of some philosophical aspects of their work. Postgraduate students are also expected to relate their own research with the paradigms of computer science; describe the intellectual foundations of their research; and explain the applicability, limitations, and boundaries of their research results. The course is divided into four themes, explained below.

Theme 1: What is Computer Science?

The question "What is computer science?" is fundamental to this course. The question of the identity of computer science has puzzled even—and perhaps especially—the most authoritative figures of computer science. Because of the question's relationship with other topics in the course, the question is explicitly visited three times during the course. Firstly, the question is posed as a pre-course question; secondly, the question is revisited after the students have become familiar with the disciplinary history of computer science; and thirdly, the question is revisited after the students have become familiar with the ontological, epistemological, and methodological issues in computer science. Each of these times the students are asked to write forum posts and to reflect on their own learning process.

Theme 2: What is Science and How Is It Done?

In order to be able to discuss the concept of *science* in computer science, the students need to understand the difficulties in defining science. Hence, we discuss, for instance, the concepts of pure and applied science, the problems of dichotomous positions towards science (e.g., "hard" and "soft"), the aims of science, and the Science Wars. Because the course is specifically focused on the philosophy of *computer science*, the topics above are tightly linked with computer science. The students continue to ponder whether their own PhD or M.Sc theses deal with a priori or a posteriori knowledge; if they employ inductive, deductive, or abductive reasoning; and if they require causal explanations, functional explanations, intentional explanations, or something else. The students are also asked to analyze the aims, boundaries, and intellectual traditions of their own research—which is something that is not always explicit in computer scientists' education.

Theme 3: Mathematics, Engineering, and Science

Understanding the intellectual connections of computer science and other disciplines, as well as understanding the cross-connections in computer science, is of major importance in understanding the nature of computer science. Based on a well-known tripartite of computer science into its theoretical/mathematical tradition, modeling/abstraction tradition, and design/engineering tradition (Denning et al., 1989), we discuss the roles that mathematics, engineering, and science play in computer science. The students are introduced to arguments that emphasize mathematics over the other traditions, as well as to arguments that emphasize engineering or empirical traditions over the other traditions. The aim of this part is to portray the roles that each of those traditions play in computer science, how computer scientists should work if computer science were taken strictly in terms of one tradition, and where the limits of each of the traditions lie. The students are asked to present and defend their own views and to critically evaluate other students' writings.

Theme 4: What Is the Philosophy of Science?

As the course title implies, the course is closely connected with the philosophy of science. Most concepts and terms used in the course indeed belong to the vocabulary of the philosophy of science. Because most of the students have no previous familiarity with the philosophy of science, they are introduced to a number of central issues in the philosophy of science, such as the foundations of science, its central assumptions and limitations, its implications, and what constitutes scientific progress. They are also introduced to some of the most notable schools in the philosophy of science, as well as to some critical views of science. When speaking about the philosophy of science, we focus on questions that can be easily connected with computer science rather than on generic questions about all sciences. The students are required to relate questions central to the philosophy of science with their own PhD or M.Sc thesis topics.

4. CONCLUSIONS

Present-day computer science combines so many different research traditions that conflicts between paradigms are inevitable. A course that covers epistemological and methodological concerns can help students to avoid some of the most obvious pitfalls in research. From our point of view, a practitioner-oriented course on the philosophy of computer science should be aimed at providing a broad understanding of different views of what computer science is, how research in different branches of computer science is done, why research is done the way it is, what are the strengths and limitations of different traditions in computer science, and how the philosophy of science can be of help for computer scientists.

Our course is aimed at encouraging critical reading and well-argumented writing. The students learn that there are many problems that do not have clear-cut answers; they learn that there are many open problems where multiple incompatible, yet credible viewpoints can be defended. The students learn to articulate their own positions, to defend those positions, to comment and criticize other positions, and to reflect and rethink their positions according to criticism. The students also get the chance to really think about the intellectual foundations of their own work and their own theses.

REFERENCES

Colburn, Timothy R. (2000) *Philosophy and Computer Science*. M.E. Sharpe: Armonk, NY, USA.

Denning, Peter J. (Chairman); Comer, Douglas E.; Gries, David; Mulder, Michael C.; Tucker, Allen; Turner, A. Joe; Young, Paul R. (1989) Computing as a Discipline. *Communications of the ACM* 32(1), 9-23.

Denning, Peter J.; Chang, Carl (chairmen) and IEEE/ACM Joint Task Force for Computing Curricula (2001) *Computing Curricula 2001*. Available on-line at http://www.sigcse.org/cc2001/ (retrieved February 14th 2007)

Dodig-Crnkovic, Gordana (2006) *What is Philosophy of Computer Science? Experience from the Swedish National Course*. European conference on Computing And Philosophy – E-CAP'06, June 2006, NTNU, Trondheim, Norway. Extended abstract available at http://www.anvendtetikk.ntnu.no/ecap06/program/Dodig-Crnkovic.pdf (retrieved February 14th 2007)

Glass, Robert L. (1995) A Structure-Based Critique of Contemporary Computing Research. *Journal of Systems and Software* 28(1995), 3-7.

Glass, Robert L.; Ramesh, V.; Vessey, Iris (2004) An Analysis of Research in Computing Disciplines. *Communications of the ACM* 47(6), 89-94.

Rapaport, William J. (2005) Philosophy of Computer Science: An Introductory Course. *Teaching Philosophy* 28(4), 319-341.

Shapiro, Stewart (2000) *Thinking About Mathematics: The Philosophy of Mathematics*. Oxford University Press: Oxford, UK.

Tedre, Matti (2007) *Lecture Notes in the Philosophy of Computer Science*. Lecture notes for the Department of Computer Science and Statistics, University of Joensuu, Finland. Available at http://cs.joensuu.fi/~mmeri/teaching/2007/philcs/ (retrieved February 14th 2007)

Tedre, Matti (*s.a.*) Know Your Discipline: Teaching the Philosophy of Computer Science. To appear in the *Journal of Information Technology Education*. Accepted February 11th, 2007. Accessible on-line at http://jite.org/

Tichy, Walter F.; Lukowicz, Paul; Prechelt, Lutz; Heinz, Ernst A. (1995) Experimental Evaluation in Computer Science: A Quantitative Study. *Journal of Systems and Software* 28(1995), 9-18.

Vessey, Iris; Ramesh, V; Glass, Robert L. (2002) Research in Information Systems: An Empirical Study of Diversity in the Discipline and Its Journals. *Journal of Management Information Systems* 19(2), 129-174.