24

Learning from other disciplines: pedagogic models within computer science and from elsewhere

Sally Fincher
Computing Laboratory, University of Kent at Canterbury, Kent CT2 7NF, England, e-mail: S.A.Fincher@ukc.ac.uk

Abstract

This short paper deals with the similarities (for teaching) between computer science as a discipline and other disciplines: for example it is like maths in one way and like medicine in another, and like English in yet another. The paper addresses the question as to how - as teachers - we can (and should) use examples from the teaching of those disciplines to enhance our own.

Keywords

Informatics, other disciplines, curriculum (general)

1 BACKGROUND

Teaching within computer science (CS) has traditionally been accomplished by the delivery of a large quantity of knowledge-based lectures supplemented with practical laboratory sessions and, increasingly, small-group and project work. This pedagogic pattern is not confined to CS, but is common across cognate disciplines (Hativa and Marincovich, 1995): mastery of facts, presented in incremental stages, from simple to complex, each stage building on the last, is the common paradigm for teaching.

Whilst this lecture-based pattern guarantees the efficient *presentation* of material, it frequently does not address other learning issues; for example, learner autonomy and synthesis of knowledge (Van Heuvelen, 1991). This is a particularly potent question for CS where we aim not only to equip the students with a coherent

Informatics in Higher Education F. Mulder & T. van Weerts (Eds.)

corpus of material, but also to have them acquire particular skills throughout their undergraduate career. For CS is not just a knowledge-based subject. As well as producing graduates with subject knowledge, we also expect them to have the ability to practice, especially with regard to the skill of programming.

2 WHAT IS DIFFERENT ABOUT CS AS A DISCIPLINE?

Programming requires a combination of applied knowledge and practical skill. It is: conceptually challenging, requiring a high level of analytical skill to break down a given problem into patterns which can be effectively mapped into an implementation language. And it is complex, involving both technical and design knowledge to create a program which not only works, but makes efficient (and, in the best examples, elegant) use of the given resources. It is also continually demanding, set in the context of a young and rapidly changing technology which affects technical objectives, requiring that a skilled programmer achieves an ever-evolving attitude and skill set. This combination of requirements is not so frequently seen in the 'hard' sciences from which CS grew, where the application of learned knowledge often *is* the practice of a discipline.

3 EXISTING ADOPTION AND INTEGRATION

CS already borrows teaching models from some other disciplines. For example, the teaching of CS is like management science in its real-world subject matter. Almost all CS work (excepting the most formal aspects) is required to exist in the world, interacting with other systems, both human and electronic. The use of case studies by management scientists (whether 'real' or constructed as a teaching tool) which model the complexity of interactions that can occur in real-life systems are increasingly being adopted within the teaching of CS.

Architecture and engineering, like CS, also build things and these things must work both within themselves and within the world. Those CS educators who adopt the label 'software engineers' recognize an affinity here and borrow pedagogic practices - for example test suites and benchmarking - as well as those teaching objectives which relate to design concerns (and constraints) working to combine functionality with aesthetics. All these are concepts which are familiar to our practice.

These 'borrowed' models undoubtedly already enhance the teaching and learning which occurs within CS. However, these models are in fact the easy ones: they cross over where the intent of the teaching is the same. In some instances, they have crossed from the practice of their originating discipline with the practitioners, who take up the teaching of CS as a supplementary or alternative subject to that in which they were trained. Indeed, this is the same way that the 'traditional' models were transferred, from the home disciplines of the first generation of CS educators

which, in general, were maths, physics and engineering (although often these educators are not aware of subsequent changes in the educative models of those original fields).

4 OTHER DISCIPLINARY MODELS

However, as the discipline of CS develops and matures into its own shape, distinct from the disciplines in which it originated, so other educational models become appropriate. It is unlikely that these models will transfer with practitioners in the same way; this time we shall have to seek them out. For example, those skills which programming requires, whilst unusual in scientific subjects, are not infrequently found in, and are sometimes traditional to, the teaching and learning in other subject areas. Law, for example, has long recognized the difference between learning law and being a lawyer and pedagogy in that discipline reflects this objective. A simple example of this is the practice of moots, where cases must be demonstrated and argued in a public arena. Medicine, too, is a content-dense subject which additionally attempts to deliver high-level skills of learner autonomy and synthesis (in the case of medicine these skills are manifested in areas such as diagnosis) and have adapted their teaching practices to reflect this (McMaster, 1997).

Although less frequently identified, there are also, perhaps, common approaches between the teaching of CS and some of the disciplines within the humanities. A poem uses a language. What makes language poetic is the approach which informs the use; the conventions of form, the density of meaning and the deliberation of the construction. Once a student is in command of language (the knowledge base) then methodologies from the teaching of English - critical approach and deconstruction of texts - might be seen to have an applicability. Perhaps especially so with regard to the 'reading' of programs written by others, a difficult process requiring both interpretative skills and commentary.

5 A FINAL OBJECTION

It might be argued that the intent, the educational objectives, which pertain in these other disciplines are too far removed from ours. If teaching methods were adopted from maths and physics and engineering, it was because they were relevant to the intent of our teaching in a way which these other areas are not.

This seems to me to be a poor argument, because intent is never identical, but it can be analogous. To use examples I have cited here, we are already teaching that learning CS is not the same as being a computer scientist and that the construction of (and appreciation of) a good program is formed by rules and informed by commentary. However, our existing educational models and methods are tools not well adapted to these tasks and our educational objectives would be better served,

our teaching enhanced and the learning of our students facilitated, by models which have been developed to address them, informed by the successes of other disciplines.

6 REFERENCES

Hativa, N. and Marincovich, M. [eds.] (1995) *Disciplinary Differences in Teaching and Learning: Implications for Practice.* New Directions for Teaching and Learning no. 64. Jossey-Bass, San Francisco.

McMaster (1997) *The "McMaster" Model of Medical Education.* The Education Section of the McMaster Faculty of Health Sciences: gopher://fhs.csu.McMaster.CA:70/11/edu.

Van Heuvelen, A. (1991) Learning to think like a physicist: a review of research-based instructional strategies. *American Journal of Physics,* **59** (10), 891-897.

7 BIOGRAPHY

Sally Fincher holds degrees from the University of Kent (BA) and Georgetown University, Washington DC (MA). She currently is working as teaching development officer within the Computing Laboratory at the University of Kent at Canterbury from where she runs the UK Computer Science Discipline Network (CSDN) and manages project EPCOS (Effective Projectwork in Computer Science). She is a member of the conference of Heads and Professors of Computing Learning Development Group.