ThinkingISsues

Tony Clear
School of Computer and Information Sciences
Auckland University of Technology,
Private Bag 92006, Auckland 1020, New Zealand
Tony.Clear@aut.ac.nz

On the Necessity of Removing "Cruelty" from the Teaching of Computing?

In his famous article [1] Edsger Dijkstra reflected upon how cruel it would be to truly teach computer science. For some reason the CS community have over the years taken the sadistic element of his entreaties to heart. Why is this so? Does it have intuitive appeal to those CS educators who believe in the "hard man" school of computer science and the "real programmers don't eat quiche" model of education.

For those of us who worked in industry I think we can all identify the type - they used to be called systems programmers in the old days. They bewailed the dumbing down of programming, with COBOL's grandiose claims to remove the need for good old assembly language programming. It was going to make programming easy, as later would fourth generation languages and all the subsequent over-hyped marketing exercises proposing the end of programming. I remember in the early 80's working on small Olivetti data capture machines with one and a half Kilobytes of programmable memory. These octal machines had cassette drives, and two programming languages (MPS and TPS) one of which claimed to be the high level language which end users would use to program with. They both looked like quaint, rather obscure assembly languages to me, and the claims seemed nonsensical.

However, since those days the scope of computing and computing careers has ballooned, and while easy programming may still remain something of a chimera, there are certainly other aspects of computing which are more accessible and more clearly offer a combination of technical and social careers.

The reducing centrality of programming in the computing curriculum is something that we now somehow need to adjust to, a concern also voiced recently by Denning & McGettrick [2], whose response is a much broader hybrid curriculum incorporating applications and centred around a theme of innovation.

On a more specific front, the growth of the so-called "Information Technology" discipline and accompanying curriculum [3] evidences an increasing need to focus on the design, management and support of IT infrastructure, and like it or not this will provide the future careers for many of our graduates.

Is it fair to provide a curriculum geared primarily to producing researchers and scientists, ("the small elite cadre of theoreticians" in Terry Winograd's critique of [1]), when graduates will mostly assume rather more prosaic roles in support of a burgeoning range of discipline sub-specialties, application domains and related career options.

In the present climate where the gloss has gone off computer science as an area for study and there is diminishing enthusiasm for computing careers in general, we need more creative strategies to make our discipline family attractive and relevant to students. These trends when related to the high school sector are seeing us attract a much more mixed student body, in some case weaker students who have been counseled into Information Technology electives as an easy study option [4], or in other cases maybe just a group of students who are a bit ‘geeky’ and have liked whatever it is that they have perceived as computing from an early age. We can probably succeed with the latter well motivated group, by continuing traditional strategies, but even then, the abrupt adjustment to a more rigorous discipline framework may have the effect of dampening their former enthusiasm for the subject. Regardless, this dual combination is a good recipe for turning others off the discipline – the absence of women and the issues with minority under representation will be further exacerbated. Who would students look for as a role model, the archetypal computing student, and how many would aspire to be that? Dilbert might even look relatively “on to it” by comparison.

Considering the ACM curriculum on networked computing [5], geared towards two year colleges raises for me some interesting questions. The graduates of this curriculum, if they go directly into employment in the industry, are likely to find roles in IT support. Over time many of them will no doubt grow in their careers and seek more challenging options. Will a computing degree be their logical next choice? Or have we set the barriers too high? Will they gain any credit for their studies to date? I ask this question because ACM’s IT curriculum [3] makes very little reference to this prior ACM sponsored effort, and it is unclear to me how two year college students would progress through a CS or IT curriculum based upon a prior networked environment course of study.

The issue is relevant far beyond the US of course. Many countries have a two tier system in which vocational institutions may prepare students for either direct entry to work or for higher university studies. If, as environmental conditions indicate, we are, at least for a while, to have a weaker cohort studying computing, then how long can we hold to a model of computing as an “elite” subject, with a Darwinian “survival of the fittest” selection process through the CS1 and CS2 courses.

In our own institution we have operated a dual tier model internally for some time. Our Diploma in IT students have traditionally studied towards careers in IT support, a two year study programme from which at one stage we wholly removed programming. But the quite distinctly separate courses mean that they gained little credit towards our heavily software development focused Computer and Information Sciences degree. In many cases they were better to progress through our Business Degree which had Information Technology (IS) and E-Business options as majors. This was also a curriculum with minimal programming at which they could succeed, and many good graduates have resulted from that hybrid study progression.

More recently we have modified the structure of our Computer & Information Sciences degree so that while omitting Computer Engineering, it now encompasses the majority of the remaining four computing disciplines [cf. 6] from Computer Science, Software Engineering, Information Technology and Information Systems. Majors include Computer Science, Software Development, Net-centric Computing, IT Security, Information System Sciences and IT Services.

The benefits of this we hope are that the tracks will: accommodate a wider variety of student interests; produce useful and employable graduates; meet the changing needs of the IT industry and offer viable “staircasing” or study progressions for students of varying abilities.

The programming thread has been adapted with a relatively standard programming intensive CS1 and CS2 option (employing JAVA as the programming language), alongside a less intensive programming sequence (programming fundamentals at a lower (perhaps CS0) level (employing Javascript as the programming language) followed by a high-level scripting languages course (still being finalised but including either one or a mix of PERL, Python and UNIX shell scripting). We hope this will prepare students for the forms of programming required in the systems infrastructure space, where small glue-like cooperating program routines are common. We also hope this will offer a suitable grounding for studying the higher level courses in the new degree majors, or signifying to students who simply cannot navigate the lesser level of programming within this course of study, that another degree option may be more sensible.

While the jury is out on this initiative, we have moved in this direction to provide meaningful and achievable courses of study in computing for the students who elect to study with us. For our institution which has a proud history of quality education producing work ready graduates, this seems like a reasonable strategy. For other institutions the paths may be quite different, with the Denning & McGettrick model [2] being one possible solution.

Many of the problems today's Computer Science educators are facing may essentially be due to the inherent tensions and contradictions of teaching a 'modern' discipline in a 'post-modern' world [6]. Whatever the response to the present challenges, persisting with a curriculum and performance expectations set by elite institutions, is something akin to cruel and unusual punishment for the student cohorts we are now seeing. Justifiably they are voting with their feet.

1. Dijkstra, E. On the Cruelty of Really Teaching Computer Science. *Communications of the ACM*, *32* (12). 1398-1404.
2. Denning, P. and McGettrick, A. Recentering Computer Science. *Communications of the ACM*, *48* (11). 15-19.
3. Ekstrom, J., Gorka, S., Kamali, R., Lawson, E., Lunt, B., Miller, J. and Reichgelt, H. Computing Curricula Information Technology Volume [retrieved from: http://www.acm.org/education/IT_2005.pdf, 21/07/2005], ACM SIGITE Curriculum Committee, New York, 2005, 115.
4. Clear, T. and Bidois, G. Fluency in Information Technology – FITNZ: An ICT Curriculum Meta-Framework for New Zealand High Schools *Bulletin of Applied Computing and IT* (retrieved from: http://www.naccq.ac.nz/bacit/0303/2005Clear_FITNZ.htm, 14/12/2005), 2005.
5. Klee, K., Austing, R., Campbell, R., Cover, C. and Currie-Little, J. Guidelines For Associate-Degree Programs To Support Computing In A Networked Environment. (retrieved from: http://www.acm.org/education/ACMGuide.pdf, 12./03/2004], ACM Two-Year College Education Committee, New York, 2000, 1-39.
6. Shackelford, R., Cassel, L., Cross, J., Davies, G., Impagliazzo, J., Kamali, R., Lawson, E., LeBlanc, R., McGettrick, A., Slona, R., Topi, H. and vanVeen, M. Computing Curricula 2005 The Overview Report including The Guide to Undergraduate Degree Programs in Computing [retrieved from: http://www.acm.org/education/Draft_5-23-051.pdf, 21/07/2005], Joint Task Force ACM, AIS, IEEE-CS, New York, 2005, 46.
7. White, L. and Taket, A. The Death of the Expert. *Journal of the Operational Research Society*, *45* (7). 733-748.