INSIGHT: Thinking Issues

Tony Clear

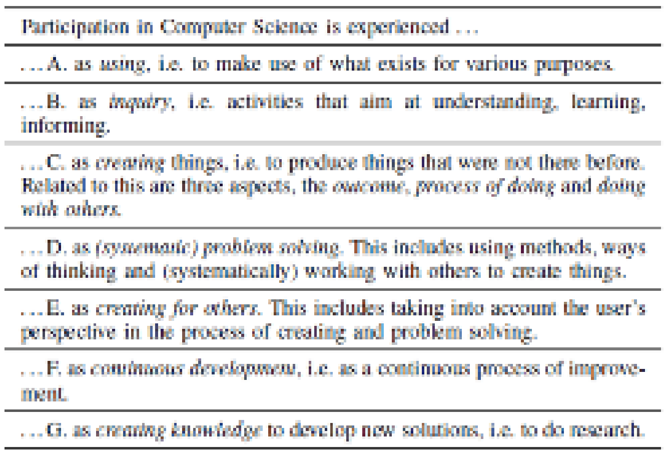
Narrowing Computer Science – the Unconscious Messages that Educators Give?

Are we conscious, as CS Educators, of the all too narrow perceptions of the discipline and the profession that we frequently transmit? Yet defining CS is not easy. It is something of a hydra - a many headed beast - which can be perceived from multiple perspectives [5], ranging from the sciences to the arts. In his excellent book on the shaping of the computing discipline [12], Matti Tedre observes that CS can be viewed from scientific, mathematical or engineering perspectives. If engaging in studying a CS degree can be viewed as a process of enculturation, what perceptions of the discipline and the profession should a CS graduate be imbued with? Should they be narrow and strongly technical to suit the needs of entry level positions, or more broad for a balanced and holistic perspective on the discipline and to prepare graduates for a developing professional career and more general citizenship? If we choose the latter then what forms of pedagogy might be effective in achieving that goal?

Towards the end of last year I visited with colleagues at the UPCERG research group at Uppsala University in Sweden to continue our work on international collaboration and Open Ended Group Projects (OEGP) [4], a pedagogical strategy of long standing at Uppsala. This occasioned considerable discussion and reflection on both the '*IT In Society*' course, then underway with US partners at Rose Hulman Institute of Technology and Gannon University, and the teaching of OEGP courses in general. We had presented a paper earlier in the year based upon work by Anne-Kathrin Peters on the identity of Computer Science students and how they perceived the discipline [11], and I had participated in an ITiCSE working group which had resulted in a major report on teaching global software engineering courses [3]. A common thread across these experiences was the challenge facing educators in motivating students to engage in work outside their comfort zone. In a context where quality assurance regimes are demanding measurements of service delivery and student satisfaction, so that increasingly 'keeping students happy' is an important part of an educator's role, this is clearly a problem. My Swedish colleagues at UPCERG reported the findings of a recent national Swedish study on student expectations from a University course. The overwhelming theme in student responses was that the course should be 'very tightly structured' so they would have a very clear idea of what they needed to do to succeed in the course. Yet surely the quality of the student experience is more than simply feeling comfortable in a course which does not challenge the student expectation of a nicely structured sequence of activities. The fallacy in the overreliance on student satisfaction measures is that only one facet of educational quality is being measured here, namely the education as "delivery of service" dimension [9]. The aspects of education as "production of the curriculum", or even more vitally "education as development of the student", are omitted.

If we consider the goals of an OEGP course, the focus transcends solely discipline based skills, to incorporate broader professional capabilities or competencies, which students would be expected to manifest in a professional setting. For instance selected learner skills for global software engineering highlighted in [7], include: “Knowledge of negotiation skills and contract writing in a common language; Managing ambiguity and uncertainty; Skills to gain the team's confidence and trust; Ability to think from the perspective of the other side, teamwork skills; Informal communication and improvisation skills”. The previously expressed student desire for a tightly structured course experience runs directly counter to the needs in an OEGP course for students to take initiative, show leadership and cope with the uncertainty and confusion implicit in the list of competencies above. So inherent in the design of the course, ‘producing the curriculum’ is intended to develop these student capabilities through exposure to authentic learning experiences in a discipline context [6]. Innate in that course design also is a desire to “develop the student” in these professional dimensions in order to produce adaptive graduates capable of working effectively in a world where professional practice increasingly demands the ability to work with remote teams across borders of time, space and culture [3,8]. I recently met in New Zealand with a graduate of the 2009 *IT in Society* course, which I had participated in, and he observed that his international experience gained through the course had helped him secure his first job in the software profession and he was now working actively on a day to day basis with teams across borders.

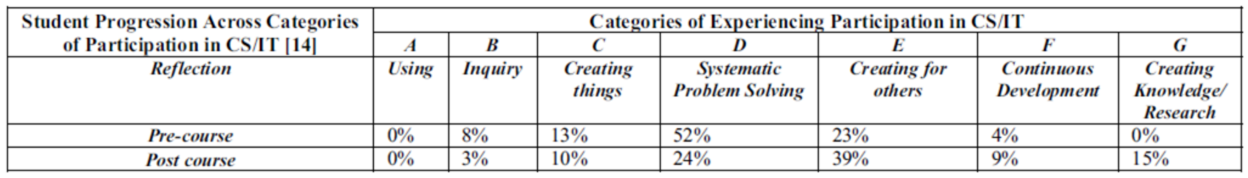
So in addition to a natural enough student desire for structure, (maybe carried over from an earlier schooling culture of ‘sound bite’ learning), what is it about Computer Science that would make such a course inherently challenging? In the phenomenographic study reported in [10] Anne-Kathrin Peters reported on the variety of ways in which CS students at the end of their first year of study, had experienced the discipline. These are presented in Figure 1.



Participation in Computer Science is experienced …
…A. as *using*, i.e. to make use of what exists for various purposes.
…B. as *inquiry*, i.e. activities that aim at understanding, learning, informing.
…C. as *creating* things, i.e. to produce things that were not there before. Related to this are three aspects, the *outcome*, *process of doing* and *doing with others*.
…D. as *(systematic) problem solving*. This includes using methods, ways of thinking and (systematically) working with others to create things.
…E. as *creating for others*. This includes taking into account the user's perspective in the process of creating and problem solving.
…F. as *continuous development*, i.e. as a continuous process of improvement.
…G. as *creating knowledge* to develop new solutions, i.e. to do research.

Figure 1. Categories Describing Qualitatively Different Ways of Experiencing The Phenomenon Participation in CS/IT [10]

In our analysis for [11] presented in table 1, we noticed that over half of the students at the outset of the OEGP course (even though this was an advanced undergraduate level course) had expressed their understanding of the CS discipline at the level of “systematic problem solving”. It was not until completion of the course that a progression to considering the discipline in a broader fashion had occurred. We see a considerably greater awareness and consideration for those for whom the technology was being created; some acknowledgment of systems as ongoing and developing entities and a number seeing scope for CS as a vehicle for research and knowledge creation.



Student Progression Across Categories of Participation in CS/IT [14]	**Categories of Experiencing Participation in CS/IT**						
	A	***B***	***C***	***D***	***E***	***F***	***G***
Reflection	***Using***	***Inquiry***	***Creating things***	***Systematic Problem Solving***	***Creating for others***	***Continuous Development***	***Creating Knowledge/ Research***
Pre-course	0%	8%	13%	52%	23%	4%	0%
Post course	0%	3%	10%	24%	39%	9%	15%

Table 1: Student Progression across Categories of Participation in CS/IT [11]

In many respects these were heartening findings, but issues nonetheless persisted with nearly 40% of the student body seeing CS in a narrower light, with a focus on problem solving, programming and algorithmic solutions, or largely "programming in the small" for tightly specified problems. Some students were dissatisfied that they had not written a software program by the end of the course, and saw the course as not relevant to their learning, since they were not interested in Health IT, in the user experience aspects of systems or determining requirements for an ill-defined problem area in a "programming in the large" setting. In discussions relating to this year's iteration of the course, many students saw their value as a CS student lying in their technical skills in "back-end programming", which imbued them with a powerful sense of self-worth, and they subsequently shied away from the "fuzzy" area of human computer interaction, for fear of being considered technical refugees and programming incompetents.
Yet what is a "technical" skill area in global software development? For instance the ACM taskforce report has noted:

"Even in the technical areas, there needs to be significant attention to front-end systems development activity, e.g., requirements elicitation, customer understanding, design, and systems and enterprise integration" [2].

As CS educators we are implicated in the view of the discipline that we transmit to our students. So it seems to me that if we are only imparting a very narrow technical perspective on the discipline, we are not only doing our students a disservice in preparing them for the globally linked careers that lie ahead, but failing in our duty to produce ethically sensitised professionals. And this in an era in which the ethical challenges facing CS professionals have never been greater. How can such narrowly [or even 'irresponsibly'] trained technicians hope to deal with such challenges as: securing a fair balance in the privacy of our citizens against ever more invasive corporate data mining of the digital traces of their lives; not becoming complicit in the surveillance mechanisms of oppressive regimes; addressing the ethical challenges posed by AI and robots in the homes; not to mention acting to prevent the widespread unleashing of lethal autonomous weapon systems in yet another ruinous arms race [1]. It is critical that we seek for more open learning strategies, such as OEGP course models, to broaden both our own and our students' outlooks.

1. Arkin, R. "The case for banning killer robots: counterpoint." *Communications of the ACM,* 58 (2015): 46-47.

2. Aspray, W., Mayadas, F., and Vardi, M. "Globalization and Offshoring of Software - A Report of the ACM Job Migration task Force." New York, USA, ACM, 2006: 1-286.

3. Clear, T., Beecham, S., Barr, J., Daniels, M., McDermott, R., Oudshoorn, M., *et al.*, "Challenges and Recommendations for the Design and Conduct of Global Software Engineering Courses: A Systematic Review." in *Proceedings of the Working Group Reports of the 2015 Innovation & Technology in Computer Science Education Conference*. Vilnius, Lithuania, ACM, 2015: 1-39.

4. Daniels, M., Cajander, Å., Pears, A. and Clear, T. "Engineering Education Research in Practice: Evolving Use of Open Ended Group Projects as a Pedagogical Strategy for Developing Skills in Global Collaboration." *International Journal of Engineering Education,* 26, (2010): 795-806.

5. Goldweber, M., et al. "Historical Perspectives on the Computing Curriculum - Report of the ITiCSE'97 Working group on Historical perspectives in Computing Education." *SIGCUE Outlook*, (1997): 94- 111.

6. Herrington, J., Oliver, R. and Reeves, T. "Patterns of Engagement in Authentic Online Learning Environments," in Proceedings of the *19th Annual Conference of the Australasian Society for Computers in Learning in Tertiary Education (ASCILITE 2002).* Auckland, New Zealand, ASCILITE, 2002: 279-286.

7. Monasor, M. J., Vizcaino, A., Piattini, M., and Caballero, I. "Preparing Students and Engineers for Global Software Development: A Systematic Review." in *Proceedings of the IEEE 5th International Conference on Global Software Engineering.* Princeton, USA, IEEE, 2010: 177-186.

8. Noll, J., Beecham, S., and Richardson, I. "Global Software Development and Collaboration: Barriers and Solutions." *ACM Inroads,* 1 (2010): 66-78.

9. Pears, A. N. "Does quality assurance enhance the quality of computing education?" in *Proceedings of the Twelfth Australasian Conference on Computing Education-Volume 103.* Brisbane, Australia, Australian Computer Society, 2010: 9-14.

10. Peters, A-K., Berglund, A., Eckerdal, A. and Pears, A. "First year computer science and IT students' experience of participation in the discipline." *in Proceedings of the 2014 International Conference on Teaching and Learning in Computing and Engineering (LaTiCE 2014).* Kuching, Malaysia, IEEE, 2014: 1-8.

11. Peters, A-K., Hussain, W., Cajander, A., Clear, T. and Daniels, M. "Preparing the Global Software Engineer." in *Proceedings of the IEEE 10th International Conference on Global Software Engineering*. Castilla La Mancha, Spain, IEEE, 2015: 61-70.

12. Tedre, M. *The Science of Computing: Shaping a Discipline.* (Boca Raton, FL, CRC Press: 2014).

TONY CLEAR
Auckland University of Technology
Engineering, Computer & Mathematical Sciences
Private Bag 92006, Auckland
1010 New Zealand
Tony.Clear@aut.ac.nz