by Tony Clear,
Auckland University of Technology

CS1: The Most Dangerous Course for CS Educators to Teach?

This column reflects on some of my own experiences, observations, and research insights into CS1 teaching over more than 25 years in my own institution and others. The challenges facing first year programming educators and the inability of universities and their managers to learn from the copious literature relating to the teaching of introductory programming seem to be perennial. This places first year programming educators in some peril!

To place the issues in context it has been observed that "academia … follow[s] the technology hype cycle … attention to technologies, management models and research approaches have characteristics of fashion cycles." [16] This phenomenon seems to particularly bedevil approaches to teaching introductory programming. Academic leaders seem to be prone to be captured by fads, even though "A management fashion is a relatively transitory belief that a certain management technique leads to rational management progress." [1] While we have noted the importance of adaptability to paradigm shifts as a core competency for computing students [6], we could extend that to include their educators and academic managers.

Motivating our own research findings into teaching novice programmers, was a sequence of prior studies that included those of McCracken [12] and Leeds [10], with general findings that novice programmers fared far more poorly on their assigned tasks than expected. In the resulting BRACElet project [5] (a several-year multi-institutional, multinational study into the teaching of novice programmers), we concluded that

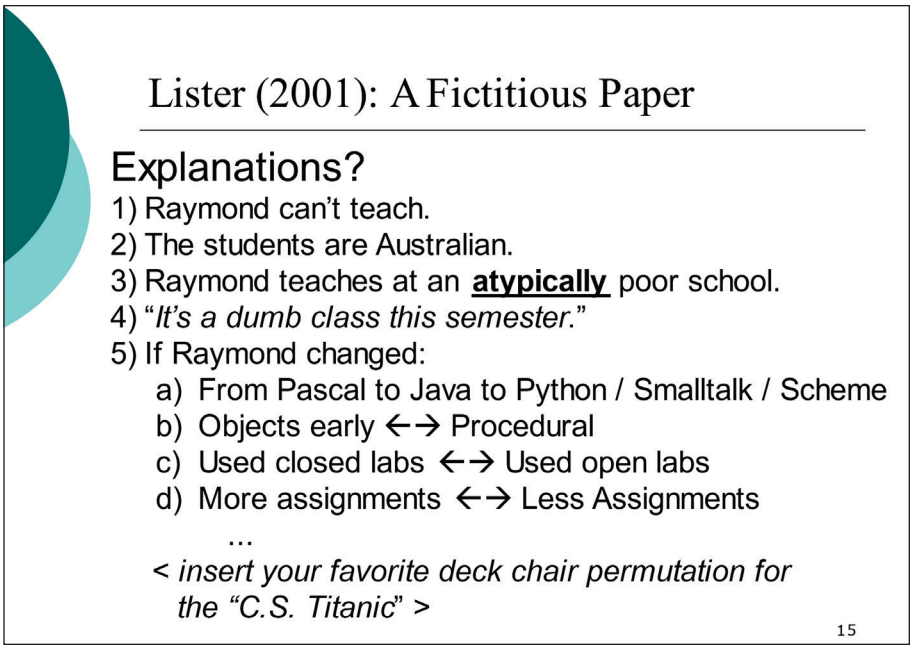

Figure 1: Putative empirical data from a study of novice programmer performance and likely explanations [9]

It has been observed that "academia … follow[s] the technology hype cycle … attention to technologies, management models and research approaches have characteristics of fashion cycles."

> The stages through which novice programmers develop to a strongly relational performance level, and the time that this development process may take, needs further investigation and may have been significantly underestimated in many modem computing curricula. Students who cannot read a short piece of code and describe it in relational terms are not well equipped intellectually to write code of their own. [19]

The responses of many CS educators and their leaders to these research-evidenced realities have been disappointing. Ray Lister's slide in Figure 1 presented at a 2007 BRACElet workshop [9] , sums it all up.

So, let's move back from the research to the evolution of practice in one setting.

When we explore a few cycles and paradigms of programming teaching at my own institution we can see a set of mini revolutions. Beginning in the Computing Department within the Business school before the turn of the century we taught Pascal to students studying introductory programming in a certificate program, and Quick Basic for introductory programming in the National Diploma in Business Computing program. Students had to navigate the aptitude test before acceptance onto the program, but acceptance onto the Business program was a further backstop option. But lest we think all was 'well in the state of Denmark' in the golden era of bright and motivated students, one early review of the courses in 1999 concluded:

> Programming assignment set at too high a level, High drop-out rate, Low average pass rate, Marking of assignments and written part of Exam leaves a lot of leeway—more precision is needed, style and level of examinations was considered ok, Main point of recent failure seems to be assignment and exam, Environment QBASIC is easy but not in tune with modern development environments. [13]

As observed in [18], despite the perennially waged "language wars," it has been observed that though there may be several valid motivators for a change of programming language, "student performance is not hindered by language." Subsequently, when our institution migrated to university status and the teaching of degree programs, we taught an Information Technology Major for the Business Degree. This program had removed programming from the curriculum (due to high failure rates for business students) and aimed to equip Business Analysts for practice as the desired graduate profile, which it achieved very successfully. Our first technical major (software development) for the Bachelor of Applied Science opted for Borland Delphi as the programming language for the first three programming courses on the degree—providing some continuity of language for students on the program. Moreover, it provided continuity for the academics teaching the new courses [18] (as there was continuity from Pascal which could be taught procedurally with console mode, or event driven with Borland's GUI, or as Object Oriented Pascal if desired. In the subsequent Bachelor of Information Technology which morphed into today's Bachelor of Computer and Information Sciences we migrated from Delphi to Java as our introductory programming language. The initial pedagogy then chosen was Objects First with BlueJ [7], and despite the commitment of the teaching team (and while competency and 70-75% pass rates were achieved), against the desired 80%+ targets, this was deemed insufficient. Issues with the progression to CS2 and algorithmic skills also had to be addressed.

After one of our periodic paradigm shifts (cf. Fig 1) when we lost a key member of the teaching team and added a second campus as a teaching location, that occasioned a change of teaching team configuration and an enthusiastic migration to an in-house version of a Robot Microworld with BlueJ and Java. This brought its own issues compared to using the vanilla version of Karel the Robot [2]. After a short period and further dissatisfaction with pass rates, we discarded the Objects First approach in favour of adopting a course based on a procedural approach, somewhat reminiscent of Stuart Reges at Washington State [11]. More radically we decided to migrate to the use of Raspberry Pi's and C for the introductory programming course, as a strategy to motivate Engineering and Computing students. This experiment led to the burnout of the course teacher, in the face of the blame culture noted in Ray's figure 1, and his resignation from the University. His replacement then adopted a more cautious approach staying with C as the programming language but discarding the complications introduced by the Raspberry Pi technology. So now after a period of stability and the stress of emergency remote teaching while trying to survive the pandemic [8], we are moving into a further cycle of dissatisfaction and the teaching team is to change again. Ah 'plus ça change, plus c'est la même chose!'

After a period of stability and the stress of emergency remote teaching while trying to survive the pandemic, we are moving into a further cycle of dissatisfaction and the teaching team is to change again. Ah 'plus ça change, plus c'est la même chose!'

Ironically the area in which we have had the most success in applying the insights into teaching novice programming gained from the BRACElet project [5], was on the pre-degree certificate [15] and CS0 foundation programming course. Perhaps this was too low stakes for it to receive the degree of scrutiny accorded to a CS1 course as a key building block for a large degree program. But my colleague Phil Robbins has done a great job in teaching a research-informed basic course covering iteration, sequence, and choice, using C#—thereby differentiating the language—in a mostly procedural fashion, and applying a competency-based pedagogy.

So much for the history. So, let's look at the drivers of these "revolutions" in cyclical waves. In the first-year course educators must serve the conflicting needs of several stakeholders. In the New Zealand context, we have the imposed "Educational Performance Indicators" where students are meant to achieve a standard level of success—a rough Pareto law of 80–85% pass rates on each course (with institutional penalties for failing to do so). Alison and I have critiqued the

To today's pioneers, this is just the dangerous territory which an introductory programming educator must (often all too transitorily) inhabit! So don't take things too personally.

application of this policy framework in [3], while the international benchmarks for pass rates in CS1 have been calculated as 67.7% [17], so in the face of that evidence an arbitrary 80–85% target has little point. The educator would then have to distort the results to achieve the targets, raising the question of whether we are measuring educational performance or conformance [4]? But the multi-stakeholder question arises again when we consider what do we really mean by educational quality? Arnold Pears has observed that we can think of three approaches to quality: *education as production*—delivering the curriculum, *education as service*—satisfying the students, and *education as development*—developing the student [14]. In a neo-liberal model of education, the focus is on the student 'as customer' so the 'service' model has dominance. But the key stakeholders of education are more than that—students as lifelong citizens of whom their parents may be proud, employers receiving well-educated professional graduates, society as beneficiaries of an educated populace to which they have contributed with financial support, respected educators with integrity, educational institutions with a multifaceted awareness of what it takes students to achieve by surmounting challenges in learning.

So, to my colleagues over the years who have been overridden by their superiors when they passed too few students or diverted to teach another course when the pressures of the waves of fashion became too high, and to today's pioneers, this is just the dangerous territory which an introductory programming educator must (often all too transitorily) inhabit! So don't take things too personally... ❖

References

1. Baskerville, R. and Myers, M. Fashion waves in information systems research and practice. *MIS Quarterly*, (2009) 647-662.
2. Becker, B. Teaching CS1 with Karel the Robot in Java. In *Proceedings of the thirty-second SIGCSE Technical Symposium on Computer Science Education*, (Charlotte North Carolina USA: ACM, 2001), 50-54.
3. Clear, A. and Clear, T., (2014). Introductory Programming and Educational Performance Indicators - a Mismatch. in *Proceedings of ITx New Zealand's Conference of IT*. (Auckland, New Zealand: CITRENZ, 2014).123-128: http://www.citrenz.ac.nz/?page_id=1296. Accessed 2022 Aug 26.
4. Clear, T. Assessment in Computing Education: Measuring Performance or Conformance? *SIGCSE Bulletin*, 40, 4 (2008),13–14: https://doi.org/https://doi.org/10.1145/1473195.1473201
5. Clear, T. Edwards, J. Lister, R. Simon, B. Thompson, E. and Whalley, J. The teaching of novice computer programmers: bringing the scholarly-research approach to Australia. in *Conferences in Research and Practice in Information Technology Vol. 78*. (Wollongong, Australia: ACS, 2008), 63-68), https://doi.org/https://dl.acm.org/doi/10.5555/1379249.1379254.
6. Daniels, M., Cajander, Å., Eckerdal, A., Lind, M., Nylén, A., Clear, T., and McDermott, R. Competencies for Paradigm Shift 'Survival'. in *45th ASEE/IEEE Frontiers in Education Conference*. (El Paso, TX, USA: IEEE, 2015), 1424-1429: https://doi.org/10.1109/FIE.2015.7344255
7. Goldweber, M., Barr, J., Clear, T., Davoli, R., Mann, S., Patitsas, E., and Portnoff, S. A framework for enhancing the social good in computing education: a values approach. *ACM Inroads*, 4 1 (2013), 58-79, https://doi.org/10.1145/2432596.2432616
8. Hodges, C., Moore, S., Lockee, B., Trust, T., and Bond, M. The difference between emergency remote teaching and online learning. *EDUCAUSE Review,* 3 (2020), http://hdl.handle.net/10919/104648, https://er.educause.edu/articles/2020/3/the-difference-between-emergency-remote-teaching-and-online-learning. Accessed 2021 Nov 21.
9. Lister, R. (2007, May 30, 2007). Observations about the Emerging Method of Multi-Institutional Education Research, presented at *Bracelet Workshop Auckland University of Technology*, (Auckland, New Zealand, May 30, 2007).
10. Lister, R., Adams, E., Fitzgerald, S., Fome, W., Hamer, J., Lindholm, M., McCartney, R, Mostrom, J., Sanders, K., Seppala, O., Simon, B., and Thomas, L. A Multi-National Study of Reading and Tracing Skills in Novice Programmers. SIGCSE Bulletin, 36 4 (2004), 119-150.
11. Lister, R., Berglund, A., Clear, T., Bergin, J., Garvin-Doxas, K., Hanks, B., Hitchner, L., Reilly, A., Sanders, K., Schulte, C., and Whalley, J. Research Perspectives on the Objects-Early Debate. *SIGCSE Bulletin*, 38 4 (2006), 146-165.
12. McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagen, D. Kolikant, Y., Laxer, C., Thomas, L., Utting, I., and Wilusz, T. A Multi-National, Multi-Institutional Study of Assessment of Programming Skills of First Year CS Students. *SIGCSE Bulletin*, 33 4 (2001).
13. NDBC Programme Team. Review of PD/PP100 course. (Unpublished, Auckland New Zealand: Auckland Institute of Technology, 1999), 1-4.
14. Pears, A. Does quality assurance enhance the quality of computing education? in *Proceedings of the Twelfth Australasian Conference on Computing Education-Volume 103* (Brisbane Australia: ACS, 2010), 9-14.
15. Spooner, K. and Clear, T. A Foundation Programme Preparing Students for Future Study in Computing and Engineering Degrees in *Proceedings 2016 Learning and Teaching in Computing and Engineering (LaTICE 2016)* (Mumbai, India: IEEE, 2016), 66-70: https://doi.org/10.1109/LaTiCE.2016.41.
16. Stahl, B. From computer ethics and the ethics of AI towards an ethics of digital ecosystems. *AI and Ethics*. 2 (2021), 65–77.
17. Watson, C. and Li, F. Failure rates in introductory programming revisited. in *Proceedings of the 2014 conference on Innovation & technology in computer science education* (Uppsala Sweden: ACM, 2014), 39-44: https://doi.org/10.1145/2591708.2591749.
18. Whalley, J. CSEd Research Instrument Design: The Localisation Problem. in *Proceedings of The Nineteenth Annual NACCQ Conference* (pp. 307-312). (Wellington New Zealand: NACCQ, 2006), 307-312: https://www.citrenz.ac.nz/conferences/2006/papers/307.pdf. Accessed 2022 Aug 12.
19. Whalley, J., Lister, R., Thompson, E., Clear, T., Robbins, P., Kumar, P., and Prasad, C. An Australasian Study of Reading and Comprehension Skills in Novice Programmers, using the Bloom and SOLO Taxonomies. in *Conferences in Research and Practice in Information Technology* Vol. 52. (Hobart, Tasmania, Australia: ACS, 2006), 243-252).

Tony Clear
School of Computing and Mathematical Sciences
Auckland University of Technology
Private Bag 92006
Auckland, 1142 New Zealand
Tony.Clear@aut.ac.nz

DOI: 10.1145/3571089