EDITORIAL

Debugging by novice programmers

Brian Kernighan asserted that debugging a computer program is twice as hard as writing the code in the first place. While computing education research has probably focused more on how novices write their code in the first place, there is also an extensive literature on how novices debug programs. In the first paper of this special issue on debugging, McCauley et al. provide a much needed review of that literature. An interesting feature of this review is the age distribution of the papers cited. A near equal number of the papers are from this current decade and from the 1980s, with those two decades accounting for almost all of the citations. This age distribution is probably a reflection of the rise, fall and rise again of computing education research in general, with the 1990s having been something of a “CSEd Winter”.

In the second paper of this special issue, Fitzgerald et al. repeat an experiment first carried out by Katz and Anderson in the 1980s. What we need in the computing education research literature are more reports like this paper, by researchers who have tried to repeat other researchers’ original work. While Fitzgerald et al. reasonably justify their repetition because “the languages, environments, and even the students, have certainly changed in over 20 years”, such repetitions are just as important for work reported in more recent years. A research community grows as much through reports of repetitions, either confirming or refuting the earlier work, as it grows through reports of original work.

Work in the 1980s on novice programmers was dominated by the cognitive perspective, with researchers focussing on the reasoning of novices. The third and fourth papers of this special issue reflect the growth of the constructivist perspective in computing education after the 1980s.

In the third paper of this special issue, Simon et al. investigate the real world debugging experiences of students before they commenced learning to program. They conclude that students have few prior experiences that prepare them for debugging programs. Furthermore, some of their prior experiences run counter to expert programmer debugging strategies. For today’s novice programmers, the expression “knowing what’s under the hood” may have no literal meaning – have you looked under the hood of a car recently? A generation who have grown up using computers, without any realistic chance of actually understanding what is “under the hood” of those computers, may have a general mental orientation very different from those of us who teach them and who are a generation older. The work of Simon et al. suggests there are aspects of the existing culture of computing that are understood implicitly by the older generation, but which need to be taught explicitly to today’s generation of novice programmers.

In the fourth paper of this special issue, Ben-David Kolikant and Mussai report upon their investigation of student conceptions of program correctness. Many computing teachers will be surprised to find that their students may have a very different
understanding of this fundamental concept. Whereas teachers see correctness as black-or-white, students appear to see shades of grey. The work of Kolikant and Mussai suggests that, in addition to explicitly teaching debugging strategies, we may also need to explicitly teach the very notion of correctness.

The methods used in the third and fourth papers will probably generate some comment and even criticism from readers. The primary value of these papers is in the fresh, provocative insights offered about novice debugging. Just as Fitzgerald et al. repeated an experiment from the 1980s, some members of the CSEd community now need to repeat the work presented in the third and fourth papers, either confirming or refuting this work.

All these papers were refereed by at least three reviewers. I thank those reviewers for their time and insightful comments — Marzieh Ahmadzadeh, Michael Clancy, Martin Dick, Stephen Edwards, Brian Hanks, Christopher Hundhausen, Yifat Kolikant, Gordon Lingard, Myles McNally, Michael de Raadt, Rob Rist, Nathan Rountree, Carsten Schulte, Allison Elliott Tew, Lynda Thomas, Errol Thompson and Susan Wiedenbeck. I also thank the authors, for their cooperation and patience through several iterations of their papers.

Raymond Lister

*University of Technology, Sydney*