Towards Gender-Inclusive Programming Pedagogy

Authors
Bedour Alshaigy, Oxford-Brookes University
Michael James Scott, Falmouth University
Mark Zarb, Robert Gordon University

Abstract
It is well documented that there is a gender gap in STEM subjects such as computer science (Camp, 1997; Gürer et al., 2002). Declining numbers of female graduates (Universities UK, 2015) has serious implications for representation in the workforce and consequently the design of technology products (Rosser, 2006). A vast amount of research explores the reasons behind the gap (Beyer et al., 2003; Margolis et al., 2003). Some of the reported difficulties include low confidence levels (Beckwith, 2004) and distorted perception of performances (Chen, 1986). It is important to recognise that female students can be disproportionately affected by such issues and, while teaching practice should be inclusive (Pulimood et al., 2008), pedagogies should be designed accordingly (Alvarado et al., 2014).

The particular challenge associated with low confidence and distorted perception can be explored through the lens of control and value appraisals. Students make self-efficacy attributions in may ways, and sometimes do not make positive appraisals after otherwise positive programming experiences (Kinnunen & Simon, 2012). This has implications on key variables that influence programming anxiety, practice behaviour, and subsequent achievement (Scott & Ghinea, 2014). However, there are strategies can make students consciously aware of their cognitive behaviour and help them to make constructive appraisals.

To this end, the authors explore practices that influence appraisals, leading to changes in self-perceptions and confidence levels in ways that typically lead to improvements in programming achievement, such as: personal robots (Scott et al., 2015; Zarb et al., 2016); media computation (Rich, Perry & Guzdial, 2004; Guzdial, 2013); psychological scaffolding (Cutts et al., 2010; Scott & Ghinea, 2013); and the “trio of best practice” consisting of contextualisation, peer instruction, and pair programming (Porter & Simon, 2013).

References


