Debugging: a review of the literature from an educational perspective

Extracts from McCauley et al *(2008)*

Why do bugs occur?

Summarizing our cited works, the literature seems to agree that the programmer experiences a breakdown. Some, but not most, of those breakdowns are caused by misconceptions about language constructs. Most errors seem to result from a chain of cognitive breakdowns. These breakdowns occur in skill, rules, or knowledge. Rule breakdowns cause errors in carrying out programming plans. Within rule breakdowns one might either apply the wrong rule or a bad rule; bad rules particularly may result from fragile knowledge. Knowledge breakdowns may result from both fragile knowledge and the superbug[1] causing incorrect programming plans. Thus, while there is no simple answer to the question, the findings cited offer useful insights.

What types of bugs occur?

…We see that most bugs are the result of something missing or malformed… Most of the novice bugs reported here are either language liabilities or algorithm awry… Some bugs appear to be endemic to novice programming, regardless of language or paradigm [e.g. off-by-one errors, operator precedence errors, misplacement of code in (or out of) a loop].

Implications for learning and teaching

The implications of the literature for teaching and learning debugging begin the day students arrive in our classes. The root causes of bugs stem from preconceptions, misconceptions, and the fragility of knowledge and understanding of programming and the language in which they work. The bugs themselves are varied and finding them quickly depends on an ability to exhibit program comprehension on a large scale rather than small chunks. There is evidence that experts do bring particular skills to debugging and that good debugging practices can be taught.

[1] According to McCauley et al (2008), the 'superbug' is a 'belief that there is a hidden mind somewhere in the programming language that has intelligent interpretive powers.' (p71)