article:1146

Learning and Teaching Programming: A Review and Discussion

The full article is a review and discussion of research relating to teaching and learning programming. This is a broad field, with a substantial literature spanning over 30 years. Our focus is on research conducted from a psychological / educational perspective, with a particular emphasis on the topic of novices learning their first programming language. The goal of the paper is to summarize and organize the main results in the literature, and to draw out certain themes for discussion. As such our methodology is that of a review, and our "data" consists of the many papers that we considered. Standard definitions of what constitutes "research", "programming", "programming language" and so on, form a broad and general conceptual framework. Within this framework have found it productive to view computer science education (CSEd) research as an interdisciplinary "trading zone" (Fincher & Petre, 2004), involving concepts drawn from psychology, education, software engineering and computer science.

A number of trends can be identified in the literature. The first is a distinction between novices and experts, with an emphasis on the many deficits of novices. The second trend is the distinction between knowledge and strategies. An important though ill-defined concept is the schema / plan as the most important building block of programming knowledge. An important but open question is why and how different strategies emerge, and how these are related to underlying knowledge. The third trend is the distinction between program comprehension and generation, with models of the former being more numerous. The distinction has implications for course design and assessment - comprehension based assessment tasks may not be a good measure of the ability to write programs. The fourth, recent trend, is a comparison of object oriented and procedural programming styles.

It is clear that novice programmers face a very difficult task. Learning to program involves acquiring complex new knowledge and related strategies and practical skills. Novices typically have many deficits in both knowledge and strategies. In terms of language features, loops, conditionals, arrays and recursion have all been identified as especially problematic. It may be helpful to make aspects of control flow and data flow explicit, and avoid "hidden" actions or states. Several authors have suggested, however, that the most important deficits relate to the underlying issues of problem solving, design, and expressing a solution / design as an actual program. As well as acquiring knowledge and strategies, novice programmers must learn to develop models of the problem domain, the notional machine, and the desired program, and also develop tracking and debugging skills so as to model and correct their programs. Explicitly identifying and addressing each of these topics may be beneficial. Finally, novice's problems exacerbated by the fact that where knowledge and strategies are learned, they are often fragile (not applied, or misapplied). Further research may be useful to determine whether this is a deficit in accessing learned material, recognizing the situations in which it is appropriate, or

having the confidence to use it / experiment. The overall picture of the issues that emerged from our review can be summarized in a 3x3 matrix, where one dimension covers attributes that the novice programmer must develop (knowledge, skills, models), and the second dimension covers the phases of creating a program (design, generation, evaluation).

From our point of view as teachers there is a distinction which is much more important than the novice vs. expert distinction which has received so much attention in the literature, namely the distinction between effective vs. ineffective novices. What underlying properties make a novice effective? How can we best turn ineffective novices into effective ones? We suggest that the most significant differences relate to strategies rather than knowledge. Most current course designs and textbooks are organized around presenting knowledge. The strategies for accessing and applying this knowledge typically receive much less attention, but they are crucial to the learning outcome. What strategies do effective novices employ, how do they relate to their knowledge and their relevant mental models, and can these strategies be taught to ineffective novices?

We hope that the review and discussion of the issues covered in this paper will be of interest to those in other disciplines, such as engineering, who are involved in teaching or learning complex bodies of structured knowledge and the strategies and models that are required to apply such knowledge in practical situations.

This work was supported by internal University of Otago Research into Teaching grants.

Author 1: Anthony Robins email: anthony@atlas.otago.ac.nz

Author 2: Janet Rountree email: noemail@nae.edu

Author 3: Nathan Rountree email: noemail2@nae.edu

Reference: Fincher S. & Petre M. (2004) Computer Science Education Research. RoutledgeFalmer,Taylor & Francis Group: London and New York.

: Back to Summer 2005 Issue Vol. 1, No. 1

: Back to List of Issues

: Back to Table of Contents