Novices and Program Comprehension: Does Language Make a Difference?

Judith Good
Department of Artificial Intelligence
University of Edinburgh
80 South Bridge, Edinburgh EH1 1HN
Scotland UK
judithg@dai.ed.ac.uk

Paul Brna
Computer Based Learning Unit
University of Leeds
Leeds LS2 9JT
England UK
paul@cbl.leeds.ac.uk

Richard Cox
Human Communication Research Centre
University of Edinburgh
2 Buccleuch Place Edinburgh EH8 9LW
Scotland UK
rcox@cogsci.ed.ac.uk

Background

This study examined the effect of programming paradigm on novice program comprehension. It follows work which sought to characterise program comprehension by experts (Pennington, 1987) and novices (Corritore & Wiedenbeck, 1991) in terms of **information categories**. These categories, described by Pennington, are: **operations** (information about specific actions which take place in the code), **control flow** (information about sequence of events occurring in the program), **data flow** (the transformations which data objects undergo during execution, including data dependencies and data structure information), **state** (time-slice descriptions of the state of objects and events in the program), and **function** (information about the overall goal of the program, essentially, "What does the program *do*?").

Pennington (1987) studied expert COBOL and FORTRAN programmers and proposed a two-stage process of program comprehension in which the preliminary dominant mental representation tends to be organised in terms of low-level control flow information. A more in-depth understanding of the program leads to the building of a *domain model* based on a more functional perspective, and to an increase in the use of functional and data flow statements to describe programs.

A similar study of Pascal novices found that they also tended to use low-level procedural statements when summarising a program, but that subjects in the upper ability quartile tended to behave more like expert programmers in terms of an increased ability to handle dataflow and functional questions (Corritore & Wiedenbeck, 1991).

These findings raise the question of whether "procedural predominance" results from the use of procedural programming languages. Pennington herself questioned the generality of results for languages other than COBOL and FORTRAN: there was evidence that COBOL programmers were better at questions about dataflow than FORTRAN programmers, and that this might be due to features of the languages themselves.

To address this question, a study was carried out on Prolog novices. Since Prolog emphasises functional and data flow requirements, it was expected that novices would provide a more abstract, functional account of program behaviour, rather than a procedurally oriented one.

The Study

The subjects were 74 first year university students enrolled in a Prolog course. Each received a packet containing a short description of the experiment, instructions, a practice problem, six, randomly-ordered, short recursive programs and a questionnaire on their programming experience. Subjects had five minutes to study each program, answer the accompanying questions and write a summary statement before turning to the next program.

Results and Discussion

Results on the comprehension questions showed much the same trend as that of previous studies: questions focussing on low-level aspects of control flow were answered significantly more easily than questions requiring other types of information. However, data flow questions were significantly harder to answer than any other type of question.

Following Pennington, the program summaries were classified into three categories: procedural, data flow and function. Procedural program summaries predominated (45%), although not to the same extent as in previous studies (Pennington found 57% while Corritore and Wiedenbeck found 50%). Furthermore, functional program summaries dominated over dataflow program summaries (38% vs 17%), a trend not found in either of the two previous studies.

In further analyses, program summaries were independently rated on a four point scale according to correctness and completeness of explanation. It was found that functional program summaries were much more likely to be rated 'Good' or 'Excellent' (66.7%) than data flow (51.1%) or procedural (50%) summaries. The implication that a functional description is more likely to be correct is consistent with the notion that the functional description correlates well with the termination of the program comprehension process.

It was concluded that the choice of programming language significantly affects novice program understanding, although procedurally oriented viewpoints tend to predominate even when, as in the case of Prolog, functional, data flow approaches are emphasised. The implications for the teaching of programming are clear – more effective methods of changing novices' "procedural bias" are required beyond simply teaching new programming languages.

REFERENCES

Corritore, C., & Wiedenbeck, S. (1991). What do novices learn during program comprehension?. *International Journal of Human-Computer Interaction*, *3*(2), 199–222.

Pennington, N. (1987). Stimulus structures and mental representations in expert comprehension of computer programs. *Cognitive Psychology*, *19*, 295–341.