Expertise and the comprehension of object-oriented programs

Simon P. Davies

Department of Psychology
University of Hull
Cottingham Road
Hull, HU6 7RX, UK
S.P.Davies@psy.hull.ac.uk

Keywords: Object-Oriented Programming, program comprehension

Abstract

This paper reports an experiment concerned with the comprehension of briefly viewed object-oriented programs. The results of this study suggest that experts are able to quickly and accurately extract object-oriented information from briefly presented programs. However, as presentation length increases, there is no difference between novices and experts in terms of different comprehension categories. The results presented here may support the claims made about the naturalness of object-oriented programming with state and operation informatiom predominating in expert's representations of such programs. This in turn is seen as reflecting salient object-oriented features of the code. However, the data also suggests that some caution is required in assessing claims about the psychological 'naturalnes' of object-oriented programs.

Introduction

One issue which has received significant attention in the literature surrounding contemporary programming languages concerns the claimed psychological advantages of object oriented (OO) languages over their procedural counterparts. For example, some advocates of object oriented programming have made strong claims about the cognitive benefits of this paradigm.

Typically, the purported advantages of such languages include claims to the effect that they constitute a more natural mapping with the external world of modelled objects and events that they are intended to represent (Booch, 1986; Rossen and Alpert, 1990). However, empirical research which has addressed these claims in more detail presents a rather different picture. For example, Davies et al (1995) showed that novice OO programmers tended to focus upon objects and inheritance when asked to classify fragments of code, whereas experts focused upon the functional properties of the program fragments. This was a counter intuitive finding, since one might predict that experts would focus upon objects and their relationships. However, this study may be methodologically problematic in that the demand characteristics associated with it may mean that for some reason novices reacted to the task they are faced with in a different way to the experts, perhaps producing a classification which they believed would be expected by the experimenter (Davies and Castell, 1992).

In this paper we have attempted to try and tap more directly the cognitive representations used by novice and expert programmers by presenting programs for a very short period of time. This may seem non-naturalistic, but some have argued that the process may be likened to reading a large program where decisions have to be made about where to expend effort on the basis of minimal information (Green et al, 1991). In this respect we are interested in evaluating the form of knowledge representation used by programmers in an experimental situation which is less prone to the demand characteristics which may exist in other studies where responses may be based upon the participants expectations about what might seem appropriate in certain circumstances.

There are several questions that arise in the context of this study. One question concerns the issue of whether information can be extracted in very short presentations of programs and if so, what information? We feel that this may have implications for our understanding of comprehension during debugging activities and may indicate that different forms of information are likely to be extracted with different frequencies. Another question we are able to address is whether novices differ from experts in their ability to extract salient information from programs and if so, in what way do they differ. For example, are any differences simply quantitative in the sense that experts are simply better at answering all categories of comprehension question? Conversely, it may be the case that some categories of comprehension question are answered correctly more frequently than others and this may be moderated by both the duration of presentation and by the expertise of the programmer. Consideration of brief presentations of programs may provide some evidence of the salience of certain information structures.

Methods

There were 32 participants in this study. One group of 16 participants were novice programmers. All participants in this group were first year undergraduate computer science students studying a course on object-oriented methods which involved the use of C++. A second group of 16 participants were all experienced C++ programmers who were either teaching this language or used it extensively in their research. The mean length of exposure of the group to C++ was 3.7 years.

This study employed a mixed design. The between subject variable was expertise with two levels, Novice and Expert. The within subject variable was question category with five level (See below). A second between subject variable was viewing time, which had two levels; Short/Long viewing time (2 seconds/10 seconds). All participants were briefly shown 4 small programs written in C++. The programs were presented for either 2 seconds or 10 seconds. The participants were then asked to answer a series of comprehension questions. There were 5 comprehension questions associated with each program. These were based upon Pennington's classification scheme which considered comprehension categories relating to function, data-flow, control-flow, operations and state (Pennington, 1987). The order of presentation of the different program viewing times and the comprehension questions were fully randomised.

Results

	Function	D-Flow	C-Flow	Operations	State
Experts	54	23	26	65	76
Novices	32	35	56	43	22

Table 1. Percentage correct responses to different categories of comprehension question (D-Flow=Data-Flow; C-Flow=Control-Flow) in the short presentation condition

These data were analysed using a mixed model ANOVA. This revealed a main effect of Expertise ($F_{1,30}=21.92$, $p<0.001$) and Question Category ($F_{4,30}=12.23$, $p<0.001$) and an interaction between these factors ($F_{4,30}=7.23$, $p<0.01$).

	Function	D-Flow	C-Flow	Operations	State
Experts	71	62	70	69	74
Novices	43	39	72	45	31

Table 2. Percentage correct responses to comprehension questions (D-Flow=Data-Flow; C-Flow=Control-Flow) in the long presentation condition

These data were analysed using a mixed model ANOVA. This revealed a main effect of Expertise ($F_{1,30}=18.67$, $p<0.001$). In the case of this study, the effects of Question Category was not significant nor was the interaction between these factors.

Discussion

We have suggested that the responses to questions concerning very short presentations of a program may tell us what programmers look for first in a piece of code - in other words, what is the salient information contained in particular programs which programmers are able to access quickly and accurately. The findings reported in this paper suggest that, in general, expert programmers are able to extract more information from very short displays than are novices. While this is not a surprising finding, the performance of experts across the different comprehension categories is variable. For short presentations of programs, questions about function, operation and state are answered correctly significantly more frequently than questions about data and control flow. Novices, in contrast seem to have an undifferentiated representation with the exception of questions regarding control flow, which, in this case, are answered correctly significantly more frequently than any other comprehension category. In the longer presentation condition, experts answer all categories of comprehension questions correctly more frequently than novices, and there is no significant difference in answers across category. In the case of novices, responses to questions regarding control flow are answered correctly significantly more frequently than any other category.

The results of this study suggest that experts are able to quickly and accurately extract object-oriented information from briefly presented object-oriented programs. However, as presentation length increases, there is no difference between different comprehension categories. In contrast, novices appear to focus upon control-flow and this is not affected significantly by length of presentation. Our results may support the claims made about the naturalness of object-oriented programming and that state and operation may predominate in expert’s representations of such programs, reflecting, in turn, salient object-oriented features of the code. However, once the programmer is able to view a program for longer periods, then no particular source of knowledge predominates. This may suggest that brief presentations of programs may provide one way to access the knowledge structures used by programmers which are free from the demands of other studies where participants may be responding in a way in which they consider is appropriate to the experimental situation.

References

Booch, G. (1986). Object-Oriented development. *IEEE transactions on software engineering,* SE-12, 211-221.

Davies, S. P. and Castell, A. M., (1992). Contextualising design: Narrative and rationalisation in empirical studies of software design. *Design Studies*, 13, 4, 379-393.

Davies, S. P., Gilmore, D. J. and Green, T. R. G. (1995). Are objects that important? Effects of Expertise and Familiarity on the classification of object oriented code. *Human-Computer Interaction*, 10, 2 and 3, 227-248.

Green, T. R. G., Petre, M and Bellamy, R. K. E., (1991). Comprehensibility of Visual and textual programs: A test of superlativism against the ‘match-mismatch’ conjecture. *Empirical Studies of Programmers, 4th workshop,* Ablex, Norwood, NJ.

Pennington, N. (1987). Stimulus structures and mental representation in expert computer programmers. *Cognitive Psychology*, 19, 295-341

Rosson, M. B. and Alpert, S. R., (1990). The cognitive consequences of object oriented design. *Human-Computer Interaction*, 5, 345-379.