



Aberystwyth University

Can Object (Instance) Diagrams Help First Year Students Understand Program Behaviour?

Ratcliffe, Mark Bartley; Thomas, Lynda

Publication date:

2004

Citation for published version (APA):

Ratcliffe, M., & Thomas, L. (2004). *Can Object (Instance) Diagrams Help First Year Students Understand Program Behaviour?*.

General rights

Copyright and moral rights for the publications made accessible in the Aberystwyth Research Portal (the Institutional Repository) are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the Aberystwyth Research Portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the Aberystwyth Research Portal

Take down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

tel: +44 1970 62 2400

email: is@aber.ac.uk

Can Object (Instance) Diagrams Help First Year Students Understand Program Behaviour?

Lynda Thomas, Mark Ratcliffe, Benjy Thomasson

Department of Computer Science, University of Wales, Aberystwyth SY23 1BJ, UK
{l.t.t., m.br., b.j.t.98}@aber.ac.uk

Abstract. This paper investigates whether first year programming students can be helped to understand program behaviour through the use of Object (Instance) diagrams. Students were introduced to this diagramming technique as a way of visualising program behaviour and then given questions that tested their understanding of object referencing. It appears that drawing their own diagrams *is* a strategy applied by more successful students. Attempts to encourage all students to use this technique through scaffolding with partially completed diagrams failed however. Weaker students did not appear to find that the partially completed diagrams helped their understanding or go on to use the technique themselves.

1 Introduction

There seems to be a general agreement amongst CS educators that many of our students have problems in mastering programming. One of the manifestations of lack of understanding of program behaviour, is that many students do not seem to be able to trace code. In particular, some students do not seem to be able to themselves produce a diagram that demonstrates an understanding of object references, which as has been pointed out, is a fundamental concept in learning Object-oriented programming [1].

In order to understand a program's behaviour it is necessary for the programmer to have a model of the computer that will execute it. This 'notional machine' provides a "foundation for understanding the behaviour of running programs" [2].

In the introductory programming sequence in Aberystwyth, we demonstrate program behaviour in lectures and tutorials mainly by drawing pictures of variables and what they reference, as in Figure 1. These diagrams represent a rough UML Object (Instance) diagram [3] - essentially providing a snapshot of the objects in a system at some point in program execution. They are a diagrammatic representation of the 'notional machine' that can then be mentally animated to observe the changes in values as the code executes. Students have had quite a bit of practice in creating these diagrams.

One problem we have observed is that when students might appropriately use this technique themselves (eg. debugging their own programs), weaker students fail to do so. We have found that these students may be impatient when the instructor resorts to drawing a diagram, then surprised that the approach works. Even in the more restricted situation of a test that asks the students to trace out what happens when just a

few lines of code are executed, as in Figure 1, scratch sheets are often returned blank. In a previous investigation we discovered that only 36% of the sheets were returned with any kind of ‘working out’.

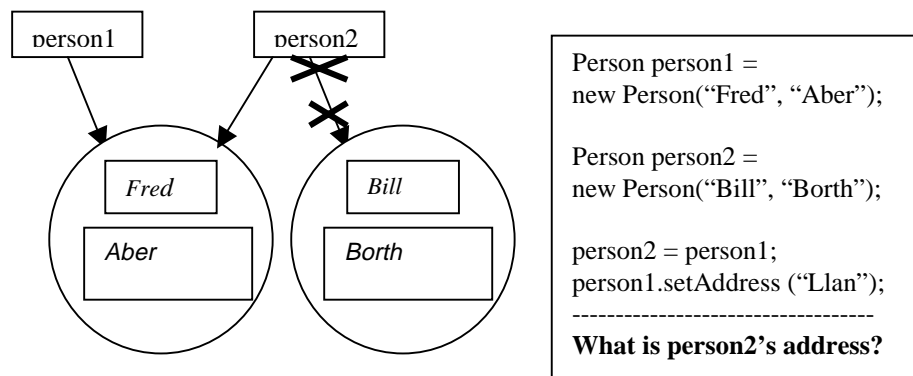


Figure 1: A partially completed Object diagram and related question

2 Background, Questions and Results

In a previous Diagrams conference, Hegarty and Narayanan [4] outlined a cognitive model of understanding dynamic systems that supposes that the viewer: decomposes the system into simpler components; constructs a static model by making representational connections to prior knowledge and other components; integrates information between different representations (e.g. text and diagrams); hypothesizes lines of action, and finally constructs a dynamic mental model by mental animation.

In addition, they have empirically validated the design guideline that people learn more from being induced to mentally animate a system before viewing an animation than by passively viewing the animation. If we examine Object diagrams we can see that the first two steps outlined in the Hegarty/Narayanan model are essentially the creation of the basic diagram, and the last three steps involve the actual tracing of the program code with reference to that diagram. This provides some justification that our approach of encouraging students to produce Object diagrams for their notional machine is a reasonable one. We were, however, frustrated that many students did not produce such diagrams themselves and we wanted to encourage them to do so.

We performed an experiment that sought to answer three research questions about the use of Object diagrams (see below) The students were divided into a control group and a group who were given partially completed object diagrams (the Experimental group). Although these students were not explicitly told to use the diagrams, diagrams were identified as belonging with particular questions. A follow-up test was also given. See [5] for a complete report on the experiment.

Is drawing some kind of Object-like diagram correlated with success in solving multiple-choice tracing questions? There is an indication that the technique *is* used by higher achieving students. The follow up test did not provide students with dia-

grams but a higher score was obtained by those beginners who drew their own. This score came close to statistical significance ($p=.07$ with one tailed t-test).

Table 1: Correlation between Diagram Use and Performance

Group	Follow-up Test Average - on tracing questions
Beginners who do not use diagrams (23)	47%
Beginners who do use diagrams (45)	68%

Does providing students with scaffolding in the form of partially completed Object diagrams help them correctly answer multiple-choice tracing questions?

It seemed very unlikely that students would NOT do better if they were given partial diagrams than if we simply give them the code with no help whatsoever. We wanted to confirm this conjecture but were very surprised by the results. When we initially tested the experimental group against the control group we discovered that Object diagrams hardly helped students trace code at all (not significant). When we looked at the results in a follow-up test, we saw that students who had been given the diagrams on the first test did slightly (but not significantly) *worse* than the control students.

Table 2: Intervention Test Results

Group	Test Average
Beginners Control (28)	28%
Beginners Experimental (40)	36%

In light of this result it was not surprising that the answer to our third research question: **Do students who have been provided with this scaffolding continue to use it in such multiple-choice questions?** was ‘no’. Since the technique was not ‘useful’ why would students continue to use it?

We are still pondering the results of this experiment. We assumed that we were providing the students with what seems like a reasonable notional machine, but we have considered that by producing the diagrams ourselves we have removed the first two steps of the Hegarty/Narayan cognitive model and thus short-circuited the students’ creation of their *own* mental model. Research is still on going.

References

1. Holland, S., R. Griffiths and M. Woodman. Avoiding Object Misconceptions. In Proceedings of SIGCSE 1997. ACM Press, 1997.
2. Robins, A., J. Rountree and N. Rountree, Learning and Teaching Programming: A Review and Discussion, Computer Science Education, Vol. 13, Number 2, and June 2003.
3. Fowler, Martin with Kendall Scott. UML Distilled. Addison Wesley, 2000.
4. Narayanan, N. Hari and Mary Hegarty. Communicating Dynamic Behaviors: Are Interactive Multimedia Presentations Better than Static Mixed-Mode Presentations? in Theory and Application of Diagrams 2000, Springer Lecture Notes in Artificial Intelligence, 1889.
5. Thomas, L.A., M. Ratcliffe, B. Thomasson, Scaffolding with Object Diagrams in First Year Programming Classes: Some Unexpected Results, SIGCSE 2004 (to appear).