Observing Mental Models in Novice Programmers

Richard Bornat[1], Saeed Dehnadi[1] and David Barton[2]

R.Bornat@mdx.ac.uk, S.Dehnadi@mdx.ac.uk, bartond@trinityhigh.net
[1]School of Science and Technology, Middlesex University, London, UK;
[2]Redditch Trinity High School, Redditch, UK

Abstract. A test which partitions subjects into those who appear to use an algorithmic model of program execution and those who do not has been automated. Experiments have been conducted in a UK school with a year cohort of students aged 13-14 and in a Mexican university with a cohort of novice computer scientists. In the school about a third of subjects appeared to use an algorithmic model, which we find surprisingly many; in the university there were around half in the same category, which is in line with university results in the UK. Operation of the online test and the analysis tool is described. Interviews with subjects in the school revealed some ways in which the algorithmic classification may be expanded. End-of-course results are not yet available for the test subjects, so statistical associations have not yet been explored.

1 Background

Dehnadi (2006, 2009) observed that some novices confronted by simple programming exercises give rational but incorrect answers. Their answers are algorithmically plausible though not always orthodox: for example, they might assign the value of a variable from left to right rather than right to left, or move a value in an assignment rather than copy it. He devised a test made up of questions about assignment and sequence programs, delivered to novice programmers without giving an explanation of the questions; those who consistently gave algorithmically-plausible results were significantly more likely to pass the end-of-course examinations than those who did not. Dehnadi et al. (2009) summarise the results of his experiments, applied to a large number of students in a wide range of universities in the UK, showing that consistent use of an algorithmic model is not simply the result of background programming experience, and that by contrast such experience on its own has little or no effect on success in the end-of-course examination; it also gives references to and discussion of previous related work.

Robins (2010) took note of Dehnadi's results, and hypothesised that learning to program is difficult because courses present a sequence of topics, each strongly supporting the next. If a student fails to understand one topic, then the next becomes far more difficult. In statistical simulations he showed that quite weak topic-on-topic dependencies produce strongly bimodal course results, a ubiquitous effect in first courses in programming which is otherwise hard to explain. It seems likely that Dehnadi's test quantifies a cognitive obstacle which trips up many students early on in their first programming course. We hypothesise (at this stage without experimental confirmation) that many novices find it difficult at first to understand that machines act utterly formally, without considering the consequences and without intention.

Ford and Venema (2010) administered Dehnadi's test to a course cohort *after* the course examination. Only 50% of those who *passed* the course could answer the multiple-assignment questions using the correct programming-language model. This opened up a new way of using the test, as a measure of successful learning in supposed practioners.

We want to administer Dehnadi's test more widely, in schools as well as in universities, and to administer it in non-English-speaking countries. To do so we decided to construct an automated version.

2 An online test

Dehnadi's test was paper-based, which made it difficult and expensive to administer – all the test cohort had to be gathered, examination-style, in the same room at the same time – and

expensive to assess – he had to read every test script and apply a fairly intricate algorithm to come to a judgement of 'consistency' or 'inconsistency'.

We had hoped to be able to integrate the test into one of the widely-used ILEs (Interactive Learning Environments) such as Blackboard or Moodle, so that teachers could easily administer it as part of the normal course activity, could receive data on performances of the individuals and groups, and could easily correlate the test results with course results. We would also avoid difficult questions about data protection by operating entirely inside the ILE firewall. But the question format of the ILEs we looked at were unhelpful, and none of them were able to implement the subtleties of the assessment algorithm (see section 2.1).

```
int a = 10; int b = 20; a = b;
int a = 10; int b = 20; b = a;
int big = 10; int small = 20; big = small;

int a = 10; int b = 20; a = b; b = a;
int a = 10; int b = 20; b = a; a = b;

int a = 10; int b = 20; int c = 30; a = b; b = c;

int a = 5; int b = 3; int c = 7; a = c; b = a; c = b;
int a = 5; int b = 3; int c = 7; c = b; b = a; a = c;
int a = 5; int b = 3; int c = 7; c = b; a = c; b = a;
int a = 5; int b = 3; int c = 7; b = a; c = b; a = c;
int a = 5; int b = 3; int c = 7; b = a; a = c; c = b;
int a = 5; int b = 3; int c = 7; a = c; c = b; b = a;
```

Fig. 1. Dehnadi's test questions

We had already developed a program which could generate the paper version of Dehnadi's test from a textual description such figure 1, using a formal description of each of his models to generate the answers and producing LaTeX code for the question and answer sheets. We took the output of the program and transcribed it into SurveyMonkey (SurveyMonkey, 2012). Each question was coded as a multiple choice; subjects could tick as many responses as they wished; there was a text box to enter alternative answers. A sample question is shown in figure 2. There were some difficulties in the transcription, so to improve accuracy we modified the generator program to produce also a text file that could be pasted, piece by piece into SurveyMonkey to produce an exact version of the test. Question answers are presented in a randomised order by the SurveyMonkey mechanism, in order to avoid the questionnaire bias which might arise if the same model appeared in the same answer-position in each question.

2.1 Analysing the output

SurveyMonkey can generate a CSV (comma-separated values) output of survey responses, each line of the output corresponding to one session with a particular subject. Our generator program was modified to analyse this output, applying a version of Dehnadi's original assessment algorithm (Dehnadi, 2006, 2009). Mental models are made up of an assignment model (one of M1-M11, each describing the action of a single assignment statement) and a sequence model (one of S1-S3, each describing the action of a number of assignments written one after the other). Each mental model determines an answer to each question, which can be a tick in a single box or in several boxes. In all but the first three single-assignment questions there are many responses – sets of ticks – which are ambiguous in their interpretation.

To resolve this ambiguity Dehnadi had used a marksheet (figure 3) with a column for each assignment model. An ambiguous answer marked all the columns whose assignment model generated that answer. Marks were notionally in pencil. Then marks in the column with the most ticks were notionally inked, and 'consistency' in answering the test was judged as follows:

5. Read the following statements and tick the box next to the correct answer below.

```
int a=10;
int b=20;

b=a;
```

The new values of a and b:

- ☐ a=0 b=10
- ☐ a=10 b=10
- ☐ a=20 b=0
- ☐ a=20 b=20
- ☐ a=10 b=30
- ☐ a=0 b=30
- ☐ a=30 b=20
- ☐ a=30 b=0
- ☐ a=10 b=20
- ☐ a=20 b=10

any other values for a and b:

Fig. 2. One of Dehnadi's test questions in SurveyMonkey

Participant code	Age	Sex	Time to do test	Prior programming	A-Level/s	Prior programming courses	Course result

Questions	Assignment								No effect	Equal sign	Swap values	Remarks (including participants' working notes)
	Assign-to-left		Assign-to-right		Add-Assign-to-left		Add-Assign-to-right		Values don't change (M9) / S	Assign means equal (M10) / S	Swap values (M11) /Ss / I	
	Lose-value (M1) /Ss / I	Keep-value (M2) /Ss / I	Lose-value (M3) /Ss / I	Keep-value (M4) /Ss / I	Keep-value (M5) /Ss / I	Lose-value (M6) /Ss / I	Keep-value (M7) /Ss / I	Lose-value (M8) /Ss / I				
1												
2												
3												
4												
5												
6												
7												
8												
9												
10												
11												
12												
C0												
C1												
C2												
C3												

Additional notes:

Fig. 3. Dehnadi's marksheet

1. A response with six inked marks in the same column for Q1-Q6 (single and double assignment) was judged consistent.
2. Otherwise, a response with 8 or more inked marks in the same column was judged consistent.
3. Otherwise, a response with fewer than 8 marked rows (two-thirds of questions) was judged blank.
4. Otherwise, the response was judged inconsistent.

Note that the assessment ignored the subject's use of sequence models, because of the difficulty of analysing the test results on paper. Note also that 'consistency' is an abbreviation for 'consistent use of a recognised rational mental model': Dehnadi was not judging a psychometric attribute of the subject, but rather a particular characteristic of their test performance.

The analysis tool uses a similar algorithm, but takes account of the use of sequence models to refine its judgements. Each subject's answer to each question is a set of ticks (write-in answers are converted to ticks of imaginary boxes with that answer). The first three questions don't require a sequence model, so answers to those questions are interpreted ambiguously as using any one of the sequence models. Rather than using 'consistent' and 'inconsistent', which can be misinterpreted, it makes judgements 'Algorithmic' and 'Unrecognised'.

Each question-response – a set of ticks – corresponds to a set of mental models, use of any of which will generate that response. The tool looks for the mental model which appears most often in the responses over the whole test (by analogy with Dehnadi's 'inking' step). If a single model is used in each of the first six questions, the response is judged 'Algorithmic (first 6)'; otherwise a single model used eight times gives 'Algorithmic overall'. This is a harsher assessment scheme than Dehnadi's because, for example, a subject who uses M2+S1 in four questions and M2+S2 in four others would not be judged Algorithmic, though Dehnadi would have labelled them 'consistent': so there's a lesser judgement 'Possibly algorithmic', assessed by considering only the assignment model and ignoring the sequence models used. There is also a judgement 'No change' applied to those who simply ticked the values from the original state (Dehnadi would have judged them 'consistent' using model M9).

3 A first experiment

We applied the test to 126 school students in an 'academy'[1] at the end of year 9 (ages 13-14). These students had been exposed to programming with MIT Scratch (MIT, 2007) and to ICT tools such as Microsoft Office (Microsoft, 2012). The analysis of their responses is shown in table 1.

Table 1. School experiment

Algorithmic		Possibly algorithmic		Unrecognised	No change	Blank
overall	first 6	overall	first 6			
42	2	4	0	70	5	3

Although these students were not complete novices, they had not been exposed to any notion of assignment, hardly at all to sequence, and not at all to formal programming notation. In fact most judged Algorithmic used the sequence model S3, in which assignments are executed in parallel. They were a complete year cohort, not a group self-selected for their interest in programming. In undergraduate novice computer scientists we have typically seen 50% or more judged algorithmic; we were surprised to find that as many as a third of these school students appeared to use an algorithmic model in 8 out of 12 questions, and that almost none had answered fewer than 8 questions.

[1] A state-funded school with a comprehensive – non-selected – intake and a state-approved aspirational agenda for its pupils.

Because the test was marked by a program, we could analyse the responses in minutes and review the judgements immediately. The school allowed us to interview some subjects on the afternoon of the day they took the test, some selected by them and some selected by us. The school's selection included both Algorithmic and Unrecognised subjects, in each group some who their teachers had expected to be placed there and some surprises. We picked out some more Algorithmic and Unrecognised individuals, and in the time we had left we tried to select at random from the students in the classroom. Overall we interviewed about fifteen subjects, out of 40 who had taken the test that morning.

We found that every interviewed student judged Algorithmic, prompted only by the question "what did you think was happening?", reported use of the mechanism identified by the analysis tool. Three of those judged Unrecognised reported something new to us: two seemed to have switched models mid-test, and one seemed to be using two models at the same time, reporting both answers. Some other Unrecognised students answered that question with a shrug, and we hadn't enough time to probe their thinking.

The analysis tool was refined to try to pick up the sequential and concurrent model users. Observation of the data showed that some subjects had ticked all or almost all the answer boxes in each question, and a judgement 'Ticked everything' was applied to them: we feel that those responses are a kind of protest, more Blank than Unrecognised. It was also possible to see that some students appeared to be algorithmic in 6 out of the last 9 questions (double and triple assignments). The refined analysis is shown in table 2. The drop in 'Algorithmic overall' is due to a decision to demote consistent use of the Equality model to 'Possibly algorithmic'; other changes are due to introduction of new judgements.

Table 2. School experiment (refined analysis)

Algorithmic					Possibly algorithmic			Unrecognised	No	Ticked	Blank
overall	first 6	last 9	sequential	concurrent	overall	first 6	last 9		change	everything	
39	2	7	7	1	5	0	2	54	5	3	3

Dehnadi's marksheet assessment would have reported 48 'consistent'; the tool reports 46 in the corresponding columns 'Algorithmic overall', 'Algorithmic (first 6)', 'Possibly algorithmic (overall)' and 'Possibly algorithmic (first 6)'. It has spotted 6 rejections where the marksheet algorithm would have had 3, though a human marker would surely have noted the 'Ticked everything' responses at least informally as rejections.[2] It has demoted 3 'consistent' responses to 'No change' and recognised two 'inconsistent' responses as the same thing.

Some of the new judgements may or may not correlate with success in the course examination. The 'Algorithmic (sequential)' category is small but may prove interesting; the 'Algorithmic (concurrent)' category may be an artefact of looking too hard. So may the 'first 6' and 'last 9' judgements. 'last 9' is especially problematic in this experiment: inspection of the tool's output showed that one way to produce it was to tick the same answer in each of the last six questions, an aspect of questionnaire bias that we hadn't previously noted. The generation/analysis tool has already been modified to check tests for that particular bias so that in future experiments we don't provoke such opportunistic behaviour.

4 A second experiment

LimeSurvey (Limesurvey, 2012) is an open-source tool which provides a very similar mechanism to SurveyMonkey but also allows direct input of survey descriptions. Although the copy-and-

[2] Dehnadi didn't see of that kind of response in his experiments. It may be that school students are more rebellious than undergraduates and college students, or it may be a consequence of online administration of the test. At the time of administration the analysis tool didn't generate this judgement, so we weren't able to interview any of the corresponding subjects.

Table 3. Mexican experiment (refined analysis)

Algorithmic					Possibly algorithmic			Unrecognised	No	Ticked	Blank
overall	first 6	last 9	sequential	concurrent	overall	first 6	last 9		change	everything	
44	2	7	3	1	3	0	3	20	1	0	8

paste mechanism of SurveyMonkey allows accurate transcription of test questions and answers, it is somewhat tedious, and SurveyMonkey requires payment in order to support a test with as many questions as Dehnadi's together with administrative questions such as 'what is your name', auxiliary questions such as 'have you programmed before', and so on. We modified the generator tool to generate an XML file describing the test to LimeSurvey (still using the original test as in figure 1, because we hadn't at that time recognised the 'last 9' questionnaire bias). Edgar Cambranes-Martinez of the University of Sussex translated its English texts into Spanish. This Spanish version was tried out on 92 students from a university novice computer science cohort in Mexico.

Analysis of the results is in table 3. This result is like Dehnadi's results (Dehnadi, 2009) in UK universities: 49, or just over half, would have been judged 'consistent' by him (Algorithmic or Possibly algorithmic, overall or first 6). The 'Algorithmic (sequential)' category is present, as in the first experiment, but proportionally a little smaller in this case (3/93 rather than 7/126). Most of the other categories are populated too, but we note that there were no 'Ticked everything' protests.

5 The benefits of interviewing

The statistical association of 'consistent'/'inconsistent' judgements in Dehnadi's test with success/failure in the course examination is highly significant. But viewed as a predictor of success or failure in the course examination the test doesn't do so well (Bornat et al., 2008). Less than 20% of his 'consistent' subjects fail, which makes 'consistency' look like quite a good predictor of success, but over 40% of 'inconsistent' subjects pass, which makes 'inconsistency' not such a good predictor of failure (see (Dehnadi et al., 2009), table 9). Those 'false negatives' are clearly very interesting.

It was always clear that Dehnadi's experiments were incomplete without interviews with both kinds of subjects. We have so far carried out only a very few very unstructured interviews under difficult conditions, but we immediately saw that there were some previously unrecognised ways of answering the test with an algorithmic mindset. So far those categories which we have recognised and can pick out from the data with an analysis tool cover few subjects, but we expect that more careful interviewing of more subjects judged Unrecognised will refine our analysis still further.

In the school experiment, almost all of the subjects placed in one of the algorithmic categories were those expected to do well by their teachers, but our tool also picked out some others. From the 40 that were tested during our visit the tool spotted four of these surprises: each one on interview reported use of the model the tool had identified. The teachers were delighted because they had an opportunity to congratulate students who didn't otherwise get much encouragement, and the students were gratified.

In at least one case an interview revealed a novel description of an algorithmic model. It was unfortunately impossible to explore that description further at the time. Future experiments will certainly try to explore subjects' descriptions in depth.

One surprise in our interviews was that at least one student, although judged Unrecognised, is keen to learn programming, but only if it has nothing to do with ICT. We were able to assure him that it doesn't, but we don't yet know if he will be successful.

6 Conclusions and further work

Now that we have an online test and an automatic analysis tool, it is much easier to conduct an experiment and to analyse its result. The analysis tool has already shown us some ways in which the test can be improved (removal of one kind of questionnaire bias) and can recognise some new groups who seem to be responding algorithmically. It remains to be seen how closely its judgements align with course success.

The most striking result from the two experiments we have conducted with the new tools is that interviewing subjects soon after test administration is very illuminating. We expected to find something, but not so much, so soon and so easily. We intend to do very much more interviewing in future experiments.

Interviews can do much more than tell us if our tools are getting the 'right' result. We want to know why subjects who aren't recognised as algorithmic by the tools answer as they do, because we are interested in understanding the obstacle to learning which the test has begun to quantify. We haven't so far been able to conduct interviews which probe in that way: certainly, short interviews with apparently un-algorithmic school students were unproductive.

7 Acknowledgements

We are grateful to the staff of Trinity High School, Redditch, for allowing us to test their students, and to the students for their good-humoured cooperation. We are grateful to Edgar Cambranes-Martinez for assistance in translating the test into Spanish, and for administering the test in Mexico.

References

Richard Bornat, Saeed Dehnadi, and Simon. Mental models, consistency and programming aptitude. In Simon and Margaret Hamilton, editors, *Tenth Australasian Computing Education Conference (ACE 2008)*, volume 78 of *CRPIT*, pages 53–62, Wollongong, NSW, Australia, 2008. ACS. URL http://crpit.com/confpapers/CRPITV78Bornat.pdf.

Saeed Dehnadi. Testing Programming Aptitude. In P. Romero, J. Good, E. Acosta Chaparro, and S. Bryant, editors, *Proceedings of the PPIG 18th Annual Workshop*, pages 22–37, 2006. URL http://www.ppig.org/papers/18th-dehnadi.pdf.

Saeed Dehnadi. *A Cognitive Study of Learning to Program in Introductory Programming Courses*. PhD thesis, Middlesex University, 2009.

Saeed Dehnadi, Richard Bornat, and Ray Adams. Meta-analysis of the effect of consistency on success in early learning of programming. In *PPIG '09: Proceedings of the 21st Annual Workshop of the Psychology of Programming Interest Group*, 2009. URL http://www.ppig.org/papers/21st-dehnadi.pdf.

Marilyn Ford and Sven Venema. Assessing the Success of an Introductory Programming Course. *Journal of Information Technology Education*, 9:133–145, 2010.

Limesurvey. Website, 2012. URL http://www.limesurvey.org.

Microsoft. Office website, 2012. URL http://office.microsoft.com.

MIT. Scratch, 2007. URL http://scratch.mit.edu/.

Anthony Robins. Learning edge momentum: A new account of outcomes. *Computer Science Education*, 20(1): 37–71, 2010. doi: 10.1080/08993401003612167.

SurveyMonkey. Website, 2012. URL http://www.surveymonkey.com.