Investigating the Viability of Mental Models Held by Novice Programmers

Linxiao Ma, John Ferguson, Marc Roper, Murray Wood
Department of Computer and Information Sciences
The University of Strathclyde
Glasgow, G1 1XH, UK
{linxiao.ma, jf, marc, murray}@cis.strath.ac.uk

ABSTRACT

This paper describes an investigation into the viability of mental models used by novice programmers at the end of a first year Java programming course. The qualitative findings identify the range of mental models of value and reference assignment held by the participants. The quantitative analysis reveals that approximately one third of students held *non-viable* mental models of value assignment and only 17% of students held a *viable* mental model of reference assignment. Further, in terms of a comparison between the participants' mental models and their performance in in-course assessments and final examination, it was found that students with viable mental models performed significantly better than those with non-viable models. These findings are used to propose a more "constructivist" approach to teaching programming based on the integration of "cognitive conflict" and program visualisation.

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computer and Information Science Education—*Computer Science Education*

General Terms

Human Factors

Keywords

Mental Models, CS1, Programming, Novice

1. INTRODUCTION

Current programming education seems far from successful. A 2001 ITiCSE working group (the "McCracken group") conducted a multi-national, multi-institutional study to assess the programming ability of first-year programming students and found that most students performed much more poorly than expected - the average score was only 22.89 out of 110 points on the general evaluation criteria [11]. This poor performance is undoubtedly a major contributor to the relatively high dropout rates, of around 30-50% [5], associated with Computer Science courses. While lack of problem-solving ability is viewed as the main cause of failure in programming learning [12], previous studies [1, 7, 8] found students often hold non-viable mental models of key programming concepts which may cause misconceptions and difficulties in solving programming problems. As Gentner [6] pointed out: 1) "*the errors that a learner makes can help reveal what the learning processes must be*"; 2) "*if typical incorrect models are understood, then instructors and designers can create materials that minimize the changes of triggering errors*". This paper aims to investigate the mental models of basic programming concepts (focusing on assignment and reference) held by novice programmers, and to study the relations between novice programmers' mental models and their performance in exam and programming tasks.

This study was prompted by the recent, somewhat controversial, study conducted by Dehnadi and Bornat [4], who explored the relationship between consistency of response and programming ability. Their original test was extended to cover the concept of reference assignment as well as value assignment, and further extended to capture qualitative data on participants' mental models.

Although not directly comparable with the Dehnadi and Bornat study, our investigation revealed some parallels regarding consistency. However, of more interest was the variety of mental models of value and reference assignment held by participants. Many of these models were seen as non-viable, meaning that they could result in a flawed understanding of programming concepts. Perhaps unsurprisingly, it was found that students with viable mental models performed significantly better in the course examination and programing tasks than those with non-viable mental models. Of more interest is the fact that many students holding non-viable models of the more advanced concept of reference assignment still managed to perform well in the course assessments. Rather more disturbing is the discovery that some students still held non-viable models of the relatively simple concept of value assignment at the end of the course. The problem of helping students to convert this diverse and persistent range of non-viable mental models into viable ones presents a real challenge to computer science educators.

2. RELATED WORK

Although a large number of studies have been conducted in cognitive science research to elicit people's mental models, not much work has been done to investigate novice programmers' mental models of programming concepts. A recent investigation that has attracted popular attention was carried by Dehnadi and Bornat [4] who devised a questionnaire to investigate the mental models that students used when understanding assignment statements. This questionnaire included twelve questions. Each question presented a small sequence of assignment statements. Students were asked to predict the values held by the variables after the execution of the program. A collection of mental models that a student might use to answer the questions were mapped by Dehnadi and Bornat, using their teaching experience of introductory programming courses, to a set of multiple choice answers.

The result of this test divided participants into three distinct groups: 1) *Consistent group* - in which the participants used the same model for all, or almost all, of the questions; 2) *Inconsistent group* - in which the participants used different models for different questions; and 3) *Blank group* - in which the participants refused to answer all or almost all of the questions. Most of the participants (21 out 27) in the consistent group scored a pass mark of 50 or above in the end-of-course exam, while most of those (26 out 34) in the inconsistent group and blank group scored below 50.

Dehnadi and Bornat argue that they had found a test to "*predict success or failure even before students have had any contact with any programming language with very high accuracy*". They even suggest that "*programming teaching is useless for those who are bound to fail and pointless for those who are certain to succeed*".

Apart from Dehnadi and Bornat's work, there are several other studies that have been conducted to investigate the mental models held by novice programmers. Bayman and Mayer [1] asked participants to explain the steps that the computer would carry out for a range of statements written in BASIC. The test revealed a wide range of misconceptions, e.g., students treated the equals sign in an assignment statement as an equality and thought the equation was stored in memory. Kahney [8] provided participants with three programs and let the participants assess whether each program produced the required recursion. The participants' reaction to the three programs was used to identify whether the participant held a *loop* model or a *copies* model. Gotschi et al. [7] studied first year programming students' mental model of recursion by analysing their "trace" records obtained from examinations and class tests. All of this research on programmers' mental models of recursion found that novice programmers often held a non-viable *looping* model of recursion, rather than the viable *copies* model.

3. RESEARCH AIMS

Intrigued by the study of Dehnadi and Bornat, the aim of this work was to study the range and consistency of mental models held by students of both simple (value assignment) and more challenging (reference assignment) programming concepts towards the end of a first-year course. It also sought to elicit the range of models held by novice programmers and to investigate the relationship between mental models and programming tasks.

4. METHOD

4.1 Participants

Volunteers were recruited from students who were in the introductory programming course, Programming Foundations, at the University of Strathclyde. The test was conducted at the end of the course when the participants had received about 20 lectures and 50 hours of tutorials and practical work. 124 students were in the course and 90 of them volunteered to take part in the test. An analysis of students performance with in-course assessment revealed that more "weak" students did not participate in this test.

In addition, the previous programming experiences of the participants were also investigated. The results showed that most participants had some procedural programming in high school although they had not taken a formal programming course before.

```
Person a, b;
a = new Person ("Jack");
b = new Person (“Tom”);
b = a;
```

Figure 1: the program used in the open-ended question

4.2 Test Questionnaire

In this test, a questionnaire was distributed to the participants who were asked to finish it under exam conditions. This questionnaire was designed to investigate students' mental models of value assignment and reference assignment. For *value assignment*, a Java primitive type value is copied from the result of the evaluated expression on the right of the assignment operator to a variable on the left. For *reference assignment*, the address of an object is copied from the right and the copy is stored in a variable of reference type on the left.

The questionnaire contained two parts: the *open-ended question* part and the *multi-choice questions* part.

The open-ended question asked participants to describe the execution of a small program (figure 1) by using text or diagrams. The use of the open-ended, unstructured question was expected to reveal more unanticipated information on the participants' mental models.

The multi-choice questionnaire contained questions asking participants to predict the result of executing a small program from a collection of pre-defined answer options, each of which mapped to a possible mental model. The first part of the questionnaire used Dehnadi and Bornat's questionnaire to study the participants' mental models of *value assignment*. The second part, devised by the authors, covered four questions to study the participants' mental models of *reference assignment*. The programs used were similar to the one used in the open-ended question (figure 1). Again, the multi-choice answers were based on a full range of possible viable and non-viable mental models, such as "*A copy of the object referenced by the variable on the right-hand-side is given to the variable on the left-hand-side*" [This is one of the inappropriate models]. (Please refer to earlier work by the authors [10] for the full questionnaire and complete list of models).

5. RESULTS

5.1 The results to the open-ended question part

25 out of the 90 participants provided too brief, or unclear, answers to the question with the result that no valid

information could be deduced from their description. Of the remaining 65 participants, 11 were identified as using appropriate mental models of assignment and reference concepts when answering the open-ended question. Most of them were able to present a clear and proper explanation of type declarations, constructors, and other concepts.

The remaining 54 participants were identified as holding at least one inappropriate mental model. These are categorised as follows (further details can be found in earlier work by the authors [10]).

5.1.1 Inappropriate mental models of assignment

- 6 participants used the inappropriate model of *assigning from the left to right*.
- 4 participants viewed "=" as a *compare operator*, rather than an assignment operator.
- A participant explained the assignment "b=a" as "*b is a subclass of a*" while another participant explained it as "*a instance of b*".
- A number of participants appeared to hold other inappropriate models but these were difficult to categorise from the descriptions e.g., "*b=a means both people are made equal to each other*".

5.1.2 Inappropriate mental model of reference and reference assignment

- 7 participants were found holding a *Stored at* model, i.e., an object is "stored at" a reference variable.
- 21 participants were found holding an *Is* model, i.e., the execution of an assignment statement is just a process to name the object.
- 4 participants were found holding an *Assign* model. For this model, an object is "assigned" to a variable. In other words, the value or object is copied and the copy is "sent" to the variable.
- 19 participants were found holding a *set equal to* model. The participants with this model described the statement $b = a$ as "*b is set to become equal to a*".
- 4 participants were found holding a *Component assignment* model, i.e., *b=a* means that some attributes of the object "in" variable a replace the corresponding attributes of the object "in" variable b.

5.1.3 Inappropriate mental models of other concepts

- 4 participants believed that an object would be created automatically when a reference variable was declared. This might explain why novices often miss the constructor in their program and struggle with the *null pointer* error.
- 3 participants held an inappropriate model that an object is a subset of a class.

5.2 The results to the multi-choice questions

Following Dehnadi and Bornat, participants were categorised as consistent or inconsistent (no one refused to answers). The consistent group can be further divided into a *consistently appropriate* group and a *consistently inappropriate* group. Furthermore, it is useful to compare the participants who held viable mental models (those in the consistently appropriate group) with those who held non-viable mental models (those in the consistently inappropriate group and the inconsistent group). In this paper, a **viable** model is defined as meeting two conditions: 1) It has to match with the model of how a programming concept actually works (appropriate); 2) it "*always*" has to match with the actual model (consistent).

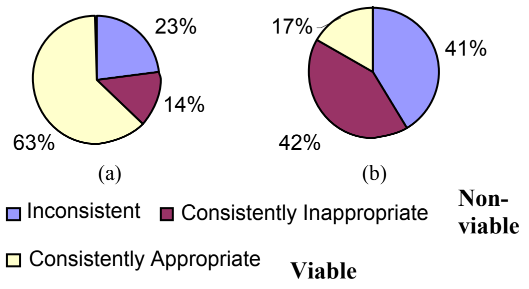

Figure 2: The percentage of each group separated based on mental models of value assignment (a) and reference assignment (b)

As Figure 2a shows, the test revealed that 69 (77%) of participants used consistent (or almost consistent)[1] mental models when answering the **value assignment** questions. The remaining 21 (23%) participants used inconsistent mental models. In the consistent group, 56 (63% of all participants) participants used consistently appropriate mental models, while the other 13 (14% of all participants) used consistently inappropriate mental models, covering: 1) *Value copied from left to right ($a \Rightarrow b$; a unchanged)*; 2) *Nothing happens (a, b unchanged)*; 3) *"=" as a test of equality;* 4) *statements are executed simultaneously.*

The test found that 53 (59% of all participants) used consistent mental models and 37 (41% of all participants) used inconsistent mental models of **reference assignment**. In the consistent group, 15 (17%) participants had a consistently appropriate mental model, while 38 (42%) participants had consistently inappropriate mental models, covering: 1) *Object copied from left to right ($a \Rightarrow b$; a unchanged)*; 2) *Object copied from left to right ($a \Rightarrow b$; the content of object in a is erased)*; 3) *object copied from right to left ($a \Leftarrow b$; b unchanged)*; 4) *Nothing happens (a, b unchanged).* Figure 2b shows the percentage of participants in each group.

5.3 Comparison with the assessment results

The participants took part in four in-course assessments throughout the introductory programming course and one final examination at the end of the course. In the in-course assessments, participants were asked to complete a small program that was evaluated on a 5-point scale. The mean of the marks for the four assessments for each participant was calculated and used in the data analysis. The final exam was paper-based, and evaluated on a 100-point scale (one participant was absent from the final exam).

To form a comparison with Dehnadi and Bornat's result, the inconsistent group and the consistent group were compared based on the participants' mental models of value as-

[1]The participants who used a consistent model to answer 10 or more questions (total 12 questions) were put into the consistent group. This is to minimize the interference of carelessness.

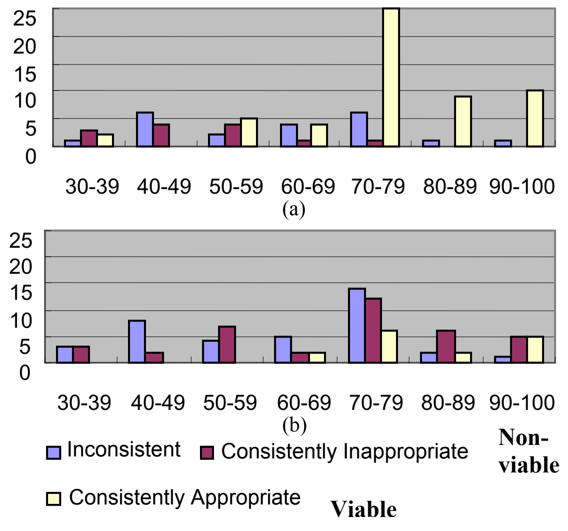

Figure 3: the results of the final exam for the participants in the inconsistent group, consistently inappropriate group, and consistently appropriate group based on (a) value assignment and (b) reference assignment

signment[2]. The results obtained match Dehnadi and Bornat's with the consistent group indeed performing significantly better than the inconsistent group in both the final exam ($p^3 = 0.027$) and the in-course assessments ($p = 0.024$). However, the separation is not clean. 14 out of 21 (67%) participants in the inconsistent group scored 50% or above in the final exam, while 9 out of 68 (13%) participants in the consistent group scored below 50%.

In this test the consistent group can be further divided into a consistently appropriate group and a consistently inappropriate group. The participants were first separated based on their mental models of **value assignment** (figure 3a). The results reveal that the consistently appropriate group performed significantly better in the final exam than the consistently inappropriate group ($p<0.001$) or the inconsistent group ($p<0.001$). In addition, the consistently inappropriate group was found to be significantly worse than the inconsistent group ($p=0.034$). This is at odds with Dehnadi and Bornat's result. For the in-course assessments, the consistently appropriate group was also found to perform better than the inconsistent group ($p = 0.049$) and the consistently inappropriate group ($p=0.070$)[4]. No statistical difference was found between the consistently inappropriate group and the inconsistent group ($p = 0.883$).

The consistent group can also be divided into the consistently appropriate group and the consistently inappropriate group based on their mental models of **reference assignment** (figure 3b). Similar to the situation for value assignment, the results show that the consistently appropriate group performed significantly better in the final exam than the consistently inappropriate group ($p=0.011$) and the inconsistent group ($p<0.001$). For the in-course assessments, the consistently appropriate group also performed significantly better than the inconsistent group ($p<0.001$) and the consistently inappropriate group ($p=0.005$). In addition, and in contrast to the results for value assignment, the consistently inappropriate group would appear to perform better than the inconsistent group ($p = 0.079$ for in-course assessments; $p=0.071$ for the final exam).

Comparing figure 3a and figure 3b it can be seen that the majority of participants (42 out 56) who were in the consistently appropriate group for value assignment changed to the consistently inappropriate group (30 participants) or inconsistent group (12 participants) for reference assignment.

6. DISCUSSION

This study has identified a collection of non-viable mental models of assignment and reference concepts held by novice programmers. The quantitative analysis revealed that, at the completion of the first year course, one third of students still held non-viable mental models of value assignment, with only 17% of students holding viable mental models of reference assignment. This result is of significant concern. Both assignment and reference are key concepts in object-oriented programming. The high failure rates in programming courses are not surprising if students still do not understand these basic programming concepts at the end of courses.

This study also found that many participants held viable mental models of value assignment but non-viable models of reference assignment. This result is not surprising given that the concept of reference assignment is much more complex than the concept of value assignment.

In addition, the results also show that the students with viable mental models performed significantly better in the course exam and programming tasks than those with non-viable mental models. This reveals how important it is that novice programmers develop appropriate mental models of the key programming concepts.

However, the sheer diversity of mental models held for the reference assignment concept, and the different results for value assignment and reference assignment suggest that viable mental models may take some time to form, or that the current mental models held by students are partially functional which may work in certain circumstances (and which they consequently may be reluctant to abandon). Handling this diversity within a cohort and helping students to develop viable models from non-viable ones is a real challenge for computer science educators.

To address these findings it is suggested that an approach to teaching programming that emphasises constructivism [2] rather than objectivism [14] might be helpful. It is suggested that many students have deeply rooted, pre-existing ideas of how computing concepts such as assignment might operate. Constructivism theory argues that traditional approaches to teaching based on lectures and textbooks are too passive and do not do enough to challenge pre-existing ideas and to help students create viable mental models. Instead constructivism argues that students actively construct knowledge by combining the experiential world with exist-

[2]We must be cautious that Dehnadi and Bornat's test was conducted before their programming course and at week 3 when students had just learned the assignment concept, but this test was conducted at the end of a 20 week programming course.

[3]The Wilcoxon Mann-Whitney Test was used to compare the consistent group and the inconsistent group. The Wilcoxon Mann-Whitney Test was used as the data was not normally distributed.

[4]Although it is not statistically significant, but still strong.

ing cognitive structures [2]. One of the key teaching strategies based on a constructive perspective is that of *cognitive conflict* which explicitly challenges existing ideas in order to encourage the learner to recognise errors in their understanding and to try to force the improvement of non-viable mental models [13].

However, it should be noted that cognitive conflict alone is unlikely to be sufficient to achieve a change in non-viable models. Students must be supported to create new viable models, and concepts must be presented in an order and fashion that allows the proper construction of inter-dependent models. This is not an easy task, especially for programming students. As Lui et al. [9] have highlighted, "*Computer programming is all fabricated that finds few parallels in the physical world*". The novice programmer lacks the necessary base knowledge for constructing viable models of programming concepts. Hence they often misuse their previous knowledge or adopt intuitive models. To address this, Ben-Ari has suggested that program visualisation has the potential to create a suitable learning environment [3]. Visualisation techniques have been used for over 20 years and have, arguably, not been as successful as hoped for. A possible reason for this is that they have been used from a traditional, objectivist perspective, ignoring a students pre-existing models. It is therefore proposed that a potential way forward is to adopt an approach based on cognitive conflict to help students realise that there is a problem with their current understanding and to use a visualization-oriented learning environment to support them in correcting their non-viable models.

Finally, it is worth observing that the results appear to support those obtained by Dehnadi and Bornat in that the consistent group performed significantly better than the inconsistent group. However, the results also show that the separation is not clean, particularly so in the case of the more advanced concept of reference assignment. Many participants in the inconsistent group still passed the examination. This would indicate that the current evidence is insufficient to suggest that it is "useless" to teach the inconsistent group and "pointless" to teach the consistent group as they claim.

7. CONCLUSION

This paper describes an investigation into mental models of basic programming concepts held by novice programmers. In this study, mental models were captured using a multiple choice questionnaire and by getting students to describe their understanding using text and diagrams. These results were then used to compare groups based on viable and non-viable models against performance in exams and programming tasks.

As a result, this study identified a collection of non-viable mental models of basic programming concepts (focusing on assignment and reference) held by first year programming students. The quantitative analysis revealed that at the completion of the first year course one third of students still held non-viable mental models of value assignment, with only 17% of students holding viable mental models of reference assignment. In addition, it was found that the students with viable mental models performed significantly better than those with non-viable mental models.

The study reveals the importance of helping students develop viable mental models. However, the traditional teaching approach, based on the theory of objectivism, does not do enough to challenge pre-existing ideas and to help students create viable mental models. Instead, it is proposed that teaching strategies based on the theory of constructivism, integrating cognitive conflict with visualisation, be used to help students create viable mental models and to challenge and help repair non-viable models.

8. REFERENCES

[1] P. Bayman and R. E. Mayer. A diagnosis of beginning programmers' misconceptions of basic programming statements. *Commun. ACM*, 26(9):677–679, 1983.

[2] M. Ben-Ari. Constructivism in computer science education. *Journal of Computers in Mathematics and Science Teaching*, 20 (1):45–73, 2001.

[3] M. Ben-Ari. Program visualization in theory and practice. *Informatik/ Informatique*, 2:8–11, 2001.

[4] S. Dehnadi and R. Bornat. The camel has two humps. Middlesex University Working Paper, 2006, http://www.cs.mdx.ac.uk/research/PhDArea/saeed/.

[5] P. J. Denning and A. McGettrick. Recentering computer science. *Commun. ACM*, 48(11):15–19, 2005.

[6] D. Gentner. Mental models, psychology of. In N. Smelser and P. B. Bates, editors, *International Encyclopedia of the Social and Behavioral Sciences*, pages 9683–9687. Amsterdam Elsevier Science, 2002.

[7] T. Gotschi, I. Sanders, and V. Galpin. Mental models of recursion. In *SIGCSE*, pages 346–350, 2003.

[8] H. Kahney. What do novice programmers know about recursion? In *The CHI 83 Conf. on Human Factors in Computer Systems*, pages 235–239, Boston, MA, 1983.

[9] A. K. Lui, R. Kwan, M. Poon, and Y. H. Cheung. Saving weak programming students: applying constructivism in a first programming course. *SIGCSE Bull.*, 36(2):72–76, 2004.

[10] L. Ma, J. Ferguson, M. Roper, and M. Wood. Investigating novice programmers' mental models. Technical report, the EFoCS Group, University of Strathclyde, 2006, http://www.cis.strath.ac.uk/ linxiao/TechReport2006.doc.

[11] M. McCracken, V. Almstrum, D. Diaz, M. Guzdial, D. Hagan, Y. B.-D. Kolikant, C. Laxer, L. Thomas, I. Utting, and T. Wilusz. A multi-national, multi-institutional study of assessment of programming skills of first-year cs students. *SIGCSE Bull.*, 33(4):125–180, 2001.

[12] J. O'Kelly, S. Bergin, P. Gaughran, S. Dunne, J. Ghent, and A. Mooney. Initial finding on the impact of an alternative approach to problem based learning in computer science. In *Pleasure By Learning (PBL) conference*, Cancun, Mexico, 2004.

[13] P. Scott, H. Asoko, and R. Driver. Teaching for conceptual change: A review of strategies. In R. Duit, F. Goldberg, and H. Niedderer, editors, *Research in Physics Learning: Theoretical Issues and Empirical Studies*, pages 310–329, 1992.

[14] C. Vrasidas. Constructivism versus objectivism: Implication for interaction, course design, and evaluation in distance education. *International Journal of Educational Telecommunications*, 6(4):339–362, 2000.