Improving the Viability of Mental Models Held by Novice Programmers

Linxiao Ma, John Ferguson, Marc Roper, Murray Wood

Abstract

Recent research has found that many novice programmers often hold non-viable mental models of basic programming concepts such as assignment and object reference. This paper proposes a constructivist-based teaching model, integrating a cognitive conflict strategy with program visualization, with the aim of improving novice programmers' mental models. The results of a preliminary empirical study suggest that, for the relatively straightforward concept of assignment, tight integration of program visualization with a cognitive conflict event that highlights a student's inappropriate understanding can help improve students' non-viable mental models. 14 out of 18 participants who held non-viable mental models of the assignment process successfully changed their model to be viable as a result of the proposed teaching model.

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 (McCracken et al., 2001). This poor performance is undoubtedly a major contributor to the relatively high dropout rates, of around 30-50% (Denning & McGettrick, 2005), associated with Computer Science courses. While lack of problem-solving ability is viewed as the main cause of failure in programming learning (O'Kelly et al., 2004), an earlier study (Ma et al., 2007) conducted by the authors found that students often held non-viable mental models of key programming concepts which may cause misconceptions and difficulties in solving programming problems.

Object-oriented programming is currently the dominating programming paradigm used in industry. Many introductory programming courses are teaching programming starting with object-oriented techniques. However, several programming teachers and educators (e.g. Ben-Ari, 2001a) argue that it is impossible for students to properly understand and use of object-oriented techniques without viable mental models of fundamental programming concepts such as variables and assignment.

This paper proposes a constructivist-based teaching model integrating a cognitive conflict strategy along with program visualization, aiming to improve novice programmers' mental models of basic programming concepts. A preliminary empirical study was conducted to investigate the effectiveness of this teaching model. The results show that after using the teaching model most students constructed a viable mental model of the assignment concept successfully from their previously held non-viable models. However, this occurred with both an integrated visualization, cognitive conflict strategy and a visualization strategy used alone. One reason for this could be the relatively simplicity of the assignment concept and the fact that the pre-test may have sparked conflict. Further studies are currently underway that investigate this finding based on more complex concepts such as object reference.

2. Related Work

A recent investigation that has attracted popular attention was carried by Dehnadi and Bornat (2006) who devised a questionnaire to investigate the mental models that students used when understanding assignment statements. This questionnaire included twelve questions, each comprising a small sequence of assignment statements, where students were asked to predict the values held by the variables after the execution of the program fragment. 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 results of this study showed that most of the participants in the *Consistent group*, in which the participants used the same model to answer all (or almost all) of the questions, scored a pass mark of 50 or above in the end of course exam. On the other hand, most of those in the *Inconsistent group*, in which the participants used different models for different questions and the blank group, in which the participants refused to answer all or almost all of the questions, scored below 50. Based on these results, 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*".

Based on Dehnadi and Bornat's test, a study (Ma et al., 2007) was conducted by the authors to investigate the viability[1] of mental models held by novice programmers based on the concepts of simple value assignment and the more challenging object reference assignment. It also sought to elicit the range of models held by novice programmers and to investigate the relationship between mental models and programming tasks. 90 first year programming students, who had received about 20 tutorials and 40 hours of labs, participated in this study.

A questionnaire was distributed to the participants who were asked to complete it under examination conditions. The questionnaire contained two parts: the open-ended question part and the multi-choice questions part. The open-ended question part asked participants to describe the execution of a small program which contained statements for object declaration, instance creation, and object reference assignment by using text or diagrams. The multi-choice questions part, which extended Dehnadi and Bornat's questionnaire to explore both value assignment and object reference assignment, asked participants to predict the result of executing a series of small program fragments from a collection of pre-defined answer options, each of which mapped to a possible mental model.

The results identified a variety of mental models of value and object reference assignment held by participants. Many of these models were seen as non-viable, meaning that they could result in a flawed understanding of the programs using these concepts. 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 object reference assignment. This result is of significant concern. Both assignment and object 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. The results also show that students with viable mental models performed significantly better in the course exam and programming tasks than those with non-viable mental models. This underlines how important it is to help novice programmers develop appropriate mental models of key programming concepts.

[1] In this study, we define a viable model as meeting two conditions: 1) It has to match with the model of how a programming concept actually works (appropriate); 2) it has to always match with the actual model (consistent).

3. A Teaching Model Based On the Integration of Cognitive Conflict and Visualization

To facilitate novice programmers constructing viable mental models it is proposed that an approach to teaching programming that emphasizes constructivism (Ben-Ari, 2001a) rather than objectivism (Vrasidas, 2000) might be helpful. Objectivism claims that there is one true and correct reality. The learning process is to transfer the objective knowledge into a learner's mind (Vrasidas, 2000). Constructivism argues that traditional approaches to teaching based on objectivism 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 existing cognitive structures (Ben-Ari, 2001a).

One of the key teaching strategies based on a constructive perspective is that of *cognitive conflict* strategy, which emphasizes explicitly challenging students' pre-existing ideas and motivating them to correct more appropriate understandings. 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 correct construction of inter-dependent models. This is not an easy task, especially for programming students. As Lui et al. (2004) 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 visualization has the potential to create a suitable learning environment (Ben-Ari, 2001b). Visualization 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 student's pre-existing models. It is therefore proposed that a potential way forward is to adopt an approach based on cognitive conflict to help students realize 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.

A teaching model, which integrates a cognitive conflict strategy and program visualization is suggested as a means of improving a student's mental models of programming concepts. There are four stages in this teaching model:

- **Preliminary Stage:** Instructors investigate the pre-existing mental models held by programming students and identify typical inappropriate models;
- **Cognitive Conflict Stage:** Trigger a discrepant event to explicitly challenge students' pre-existing mental models and push students into cognitive conflict status;
- **Model Construction Stage:** Help students construct viable mental models by using visualization
- **Application Stage:** Students go on to solve a programming problem by using the constructed mental model.

To facilitate this teaching model a web-based learning environment was developed integrating a cognitive conflict strategy and visualization technique. The conflict event is triggered by asking students to predict the result of a program, which is designed to cover a collection of inappropriate mental models, i.e. the students with the inappropriate mental models will fail the test and be warned that their answer is wrong. Then a visualization tool (Figure 1) simulates and visualizes the dynamic execution processes of a program, similar to the one used to trigger conflict. The visualization tool allows students to execute the program step by step. When each statement is executed, the dynamic execution process is visualized by using

graphical representations and animation. Meanwhile, students are also provided with the textural explanations of the execution process.

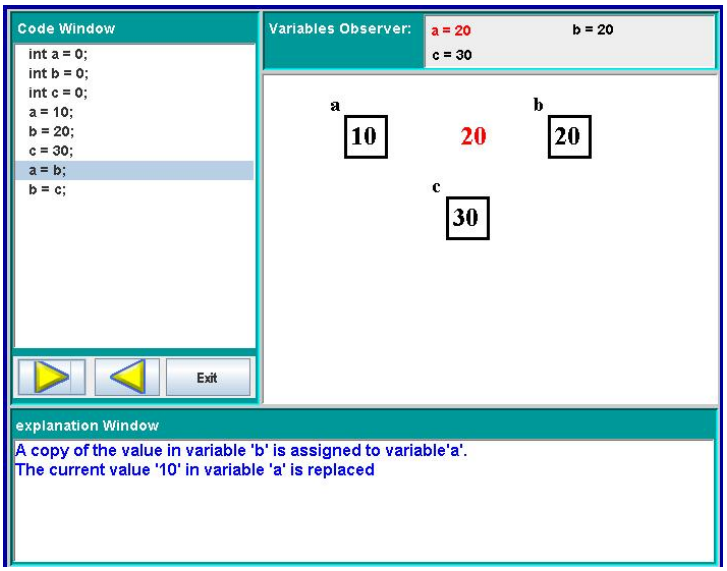

Figure1: the visualization tool

4. Investigating the Effectiveness of the Teaching Model

A study was conducted to investigate the effectiveness of the teaching model, focusing on the roles of cognitive conflict and visualization. 60 volunteers were recruited from students who were in the introductory programming course. This group was recruited from the year following that used in the earlier study (Ma et al., 2006). The experiment was arranged at the fifth week of the course, after the participants had been introduced to and practiced the assignment concept along with other basic concepts.

4.1 Method

The study aimed to investigate whether or not the proposed teaching model was able to help participants construct viable mental models of value assignment, and especially, whether or not the participants experiencing cognitive conflict performed better than others. In the pre-test, participants' pre-existing mental models of the assignment concept were elicited by using a simplified version of Dehnadi and Bornat's questionnaire. In addition, there was an additional answer option in each question for participants to state whether or not they thought the program fragment in the question could execute correctly. If they thought there was any problem in the program, they were asked to explain it. Participants who were found to be holding non-viable mental models were separated equally into two groups: the CC+Viz, which used all the functions of the suggested teaching environment, i.e. first experienced the conflict event and then used the visualization tool; and the Viz group which only used the visualization tool, but without experiencing an explicit conflict event. In the post-test, the participants' mental models of the assignment concept were investigated again using a simplified version of Dehnadi and Bornat's questionnaire, but with different questions. In addition, an open-ended question asked the participants to describe the execution of a program fragment which included assignment statements. Furthermore, a questionnaire was employed to collect the participants' qualitative feedback of the teaching environment.

4.2 Results

The pre-test revealed that 22 (37%) participants consistently used the appropriate mental model, i.e. **M2** on Dehnadi and Bornat's mental model list, while 12 (20%) participants used inconsistent models (**MIncon**), and 26 (43%) consistently used inappropriate models, covering **M9**, **ME**, **MUR**, **M11Ss** and **M2Ss** (refer to Table 1 for the explanation of each model). The inappropriate models can be separated into two categories: the inappropriate models of the assignment process, covering M9, ME, MUR, and M11Ss; and the inappropriate models of execution flow, covering M11Ss and M2Ss. (Note that M11Ss actually covers two inappropriate models: "*M11*" is an inappropriate model of assignment; and "***Ss***" is the inappropriate model of execution flow.) According to Dehnadi and Bornat, the *Ss* model "*derived from the misconception that assignments execute simultaneously; each line of code is an individual statement and should be treated separately*"

Model	Description of the Model
M2	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.
MIncon	Different models are used to answer the collection of questions.
M9	Nothing happens when an assignment statement is executed.
ME	Viewing "=" as a compare operator.
MUR	A variable can not be "rewritten", i.e., the variable can be only written once.
M11Ss	Variables swap values when an assignment statement is executed + Ss Model
M2Ss	M2 + Ss

Table 1: Mental Model List

10 out of the 38 participants who held non-viable mental models (inconsistent models or consistently inappropriate model) did not finish the post-test. The data from the remaining 28 participants were available for analysis. Table 2 shows the distribution of these participants.

Group	MIncon	M9	ME	MUR	M11Ss	M2Ss	Total
CC+Viz	6	0	2	1	1	4	**14**
Viz	2	1	4	0	1	6	**14**
Total	**8**	**1**	**6**	**1**	**2**	**10**	**28**

Table 2: The distribution of the participants whose data is available to analyse

As Table 2 shows, 18 participants (10 were in the CC+Viz group and 8 were in the Viz group) held inappropriate models of the assignment process (M9, ME, MUR, and M11Ss) while 12 participants (5 were in the CC+Viz group and 7 were in the Viz group) held inappropriate models of execution flow (M11Ss and M2Ss).

The result show that all 18 participants who held inappropriate mental models of the assignment concept, no matter which group (CC+Viz or Viz) they were in, made changes to their mental model of the assignment process (Table 3). 14 of them (78%) successfully changed their models into an appropriate one (only for the assignment process), while the remaining 4 participants changed their model from ME and M11Ss to inconsistent models. The CC+Viz group appears to perform slightly better than the Viz group: 9 out of 10 (90%) participants in the CC+Viz group changed their model of the assignment process to an appropriate one while 5 out of 8 (62.5%) participants made the successful change in the Viz group.

	Model Change Successfully					Model Change Failed		
Group	**MIncon => M2/Ss**[2]	**M9 => M2/Ss**	**M11Ss => M2/Ss**	**ME=> M2/Ss**	**Total**	**M11s => MIncon**	**ME => MIncon**	**Total**
CC+Viz	6	0	1	2	**9**	1	0	**1**

[2] M2/SS means the model is M2 or M2Ss

Viz	2	1	1	1	**5**	0	3	**3**
Total	**8**	**1**	**2**	**3**	**14**	**1**	**3**	**4**

Table 3: the distribution of participants who changed their mental model of assignment process

With regard to execution flow, the results showed that 6 out of 12 (50%) participants who held the *Ss* model changed their model to the appropriate one, while the remaining 6 participants did not make changes to their models (Table 4). Similar to the situation of model changing for the assignment concept, the CC+Viz group appears to perform slightly better than the Viz group: 3 out of 5 (60%) participants changed their models in the CC+Viz group while 3 out of 7 (43%) participants changed their models in the Viz group.

	Model Change Successfully			**Model Change Failed**		
Group	**M2Ss => M2**	**M11Ss => M2**	**Total**	**M2Ss => M2Ss**	**M11Ss => M2Ss**	**Total**
CC+Viz	2	1	**3**	2	0	**2**
Viz	3	0	**3**	3	1	**4**
Total	**5**	**1**	**6**	**5**	**1**	**6**

Table 4: the distribution of participants who changed their mental model of execution flow

Along with quantitative data, qualitative data was also collected using a questionnaire. This feedback revealed three important characteristics of the teaching environment. Firstly, the animation to simulate the dynamic process of assignment could challenge a participant's pre-existing understanding of the assignment concept. Secondly, the animation was viewed as being very helpful in promoting understanding of the concept. Finally, the step by step execution was viewed as another helpful feature, which in the words of one student "*broke down the changes taking place*".

5.3 Discussion

The pre-test, which investigated the pre-existing mental models held by participants, revealed that students often held similar inappropriate mental models. There were 10 participants who held the ME model and 12 participants who held the M2Ss model, while only 4 participants held other inappropriate models. In this case, it would seem a relatively simple task for instructors to design learning materials that could change the inappropriate mental models held by most students. In addition, well-designed learning material can cover many different kinds of mental models, i.e. the same learning material is capable of changing different kinds of inappropriate mental models. For example, the visualization tool used in this study is capable of changing all the inappropriate mental models of the assignment concept identified in the pre-test. In addition, the cognitive conflict question is also capable of triggering cognitive conflict for all the inappropriate mental models of the assignment concept identified in the pre-test. This implies that the "conceptual change" based teaching strategy is a practical proposal.

In this study, all the participants, no matter which group they were in, made changes to their mental model of the assignment process. This implies that the visualization tool, even though not using an explicit cognitive conflict strategy, was able to challenge a student's pre-existing ideas of the assignment concept.

One possible explanation for this is that the assignment concept is relatively straightforward. When the animation simulates the process of assignment, it is not difficult for participants (e.g. those who viewed "=" as an equal sign) to realised they were holding an inappropriate mental model. In addition, it also implies that the animation may be a better tool to promote conceptual change than the traditional textual and static learning materials. This study was conducted after the participants had covered the assignment concept using traditional learning materials delivered in a traditional lecture based course. Those traditional learning materials did not help many of the participants to realise that their understanding of the assignment

concept was inappropriate. It is even possible that the participants did not engage, or only engaged superficially, with the traditional learning materials.

Furthermore, the visualization/animation has been found as an effective way to help students construct viable mental models of the assignment concept. Nearly 80% of participants constructed a viable mental model from their non-viable one by using the visualization tool. On the other hand, a few students still did not manage to construct a viable mental model of the assignment concept, even though they had realized their pre-existing mental model was inappropriate. For those students, the construction of viable mental models might require more time and support.

While the visualization tool helped improve the participants' mental models of the assignment process successfully, it seemed less effective for improving the mental models of execution flow. Half of the 12 participants who held inappropriate models of execution flow did not realize they were holding inappropriate models after using the visualization tool, even when some of them were challenged with a cognitive conflict question. This result led the authors to review the example used in the visualization tool and the cognitive question. As mentioned earlier, the inappropriate mental model derived from two misconceptions, namely: 1) assignments execute simultaneously; and 2) each line of code is an individual statement and should be treated separately. While the step by step execution mode of the visualization tool is capable of correcting the first misconception, unfortunately, the example (figure 2a) used in the cognitive conflict question and the visualization tool failed to trigger cognitive conflict when a participant held the second misconception. As figure 2a shows, the execution of Line1 does not affect the result of the execution of Line2, i.e. no matter whether or not the Line1 is executed, the result of Line2 is always "b = 30; c=30". In this case, even though the participants held the second misconception, they can still pass the example successfully. Actually, it is easy to solve this problem by using another example, e.g. the example in figure 2b.

```
    int a =10, b=20, c=30;
Line1: a = b;
Line2: b = c;
```

(a) the current example

```
    int a =10, b=20, c=30;
Line1: a = b;
Line2: c = a;
```

(b) the suggested example

Figure 2: the current example and modified example used in this study

In this case, it is not difficult to understand why so many participants in the experiment did not change their mental models of execution flow. This finding explains why some teaching materials (inc. visualization-based materials) are not always helpful for improving students' understanding, even though those materials have been viewed as well-designed by instructors. When instructors design materials based on their views, but without considering students' pre-existing mental models, those materials might miss models held by some students. In this case, those students cannot change their mental models, even though they are engaging with the materials. This reveals a weakness of the objectivism-based teaching approach and highlights the value of using a constructivism-based teaching approach.

One unavoidable limitation of this study was its lack of a long-term investigation into the effects of the teaching model on the durability of the mental models constructed. According to Norman (1983), mental models are unstable. Students may lose some functional "details" of their mental models over time. On the other hand, students might need more time to construct viable mental models.

6. Future work

This study reveals the positive role that visualization plays towards improving students' mental models of a relatively simple programming concept, namely assignment. In this study, the importance of the cognitive conflict strategy was not obvious; there was no major difference between the CC+Viz group and the Viz group. However, the assignment concept is relatively straightforward and as a result it was easy for students to become engaged with the learning materials, whether or not they were challenged explicitly by a cognitive conflict event. With a more complex concept it is expected that the cognitive conflict event will play a more significant role in encouraging students to become engaged with the learning materials. An experiment is currently underway to investigate the effectiveness of the learning model with the more complex concept of reference concept. The results from this study will be available at the time of the workshop.

In addition, further studies are ongoing to investigate the robustness of the mental models adopted by students over a period of time.

7. Conclusion

This paper has proposed a constructivist-based teaching model which integrates a cognitive conflict strategy along with program visualization. The results of this initial study suggest that, for the relatively straightforward concept of assignment, tight integration of program visualization with a cognitive conflict event that highlights a student's inappropriate understanding can help improve students' non-viable mental models. While most students successfully constructed a viable mental model of the assignment concept using the teaching model, the importance of the cognitive conflict component within the model remains less obvious, perhaps due to the simplicity of the assignment concept. A further study is suggested to investigate the effectiveness of the teaching model for a more complex concept, e.g., object reference assignment. In addition, further studies will be also conducted to investigate the long-term effects of the teaching model on the construction of mental models.

8. Reference

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

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

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

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

[5] 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.

[6] L. Ma, J. Ferguson, M. Roper, and M. Wood. Investigating the viability of mental models held by novice programmers. In 38th ACM Technical Symposium on Computer Science Education, Covington, Kentucky, March 2007.

[7] 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.

[8] D. A. Norman. Some observations of mental models. In D. Gentner and A. L.Stevens, editors, *Mental Models*, pages 7 –14. Lawrence Erlbaum Associates, Hillsdale, NJ, 1983.

[9] 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.

[10] 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.

[11] 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.