Using Student Understanding to inform the development of Learning Content[*]

Frances Sheridan[†]

Orla Lahart[†]

[†]National College of Ireland

Abstract

The challenges associated with teaching computer programming are recognised (Biddle & Tempero, 1998; Jenkins, 2002). However, categorising the different ways in which students understand or think about programming is central to improving teaching and learning in the area. This paper presents an empirical study, which investigates students' understanding of the iteration concept in computer programming. The study involved two phases carried out over two academic years. Phase one involved 22 student interviews using a phenomenographic research approach (Stamouli, 2007). A phenomenographic research study identifies a finite set of ways in which students understand particular phenomena. The data arising from student interviews formed categories of understanding, which indicate that although students may be proficient in programming iteration, a deep understanding of the concept may be lacking. While this type of data may also be captured in exams, an empirical research study such as this is necessary to explain the origins of the understanding or misunderstanding.

As a result of phase one, a number of modifications were made to the teaching resources associated with iteration and their delivery. Phase two involved 18 student interviews using a phenomenographic research approach. Once again, data from these interviews formed categories of understanding. A comparison of those categories arising from phase one and phase two of the empirical study indicate a deeper understanding among students from phase two of the research study. This suggests that analyzing student understanding and using this to inform the development of learning content may have a positive effect on teaching and learning.

This paper has the following format, section one provides an overview of the key challenges with teaching computer programming. Section two provides a detailed explanation of the phenomenographic research approach. Section three presents phase one of the empirical study and its results, with section four outlining the modifications made to teaching resources based on those results. Section five describes phase two of the study with its results and section six draws conclusions and outlines the next steps in the research journey.

[*]URL: http://ocs.aishe.org/index.php/international/2009/paper/view/134

1. Introduction

We have recently experienced a decline in interest among students looking to study Computer Science. This has led to a shortage of software engineers, computer programmers and computer systems managers (Expert Group on Future Skills Needs, 2006). In particular, a potential shortfall of up to 2,300 computing graduates in Ireland is predicted by 2010 (Expert Group on Future Skills Needs, 2003). A number of factors affecting computer science as a discipline have been identified (Denning, 2004). And computer programming has been proposed as a primary factor (Cantwell Wilson & Shrock, 2001; Jenkins 2002). To tackle this, an appreciation of student understanding of computer programming is desirable. One way to gain this understanding is through phenomenographic research.

2. Phenomenographic Research

The phenomenographic research approach involves "the empirical study of the differing ways in which people experience, perceive, apprehend, understand and conceptualise various aspects in and of the world around us." (Marton 1994). The approach emerged from the question of why some people are better learners than others and involves an understanding of the distinction between "what" people understand and "how" they understand it (Marton & Dalghren, 1976). A phenomenographic research study identifies a finite set of ways in which people understand particular phenomena. These terms of understanding are known as categories of description which then form the outcome space for the study.

This paper presents an empirical study, which investigates students' understanding of the computer programming concept of iteration. The study involved two phases and was conducted over two academic years. As a result of phase one a number of modifications were made to the teaching resources associated with iteration and its delivery. The research study will now be described in detail.

3. Study – Phase One

Participants comprised students across three different computer science programmes at the National College of Ireland. All students were currently studying a computer programming module where iteration was a core topic. An opportunistic sample of 22 students participated in phase one of the study. Students first completed a questionnaire to gather information on age, gender, prior programming experience and educational background. In addition to this basic statistical information, this questionnaire also contained three questions pertaining to the students' knowledge on programming. Following this, students were asked to participate in interview sessions. Twelve students were interviewed individually and 10 students were interviewed in a group session. Each interview was transcribed verbatim in its entirety. This was an important phase of the study as with phenomenographic studies interviews play a critical role in forming the outcome space.

Data for this study was analysed in a phenomenographic manner. As outlined earlier, all interviews were transcribed verbatim. This allowed the researcher to capture not only what the subject said, but also the context and atmosphere in which it was delivered. Throughout the first reading of the transcripts, keeping all research questions in mind, utterances of relevance or interest were marked. At this early stage the utterances were interpreted within the context of the overall interview. This collection of marked statements was then compiled and grouped, taking into account both relationships and differences. These groups formed the data pools which would be the basis for the next step of analysis. The process was interative, during which some utterances were removed and some regrouped. This resulted in the emergence of 3 categories under the theme of repetition.

Theme:	**Category Name**	**Category Description**

Repetition	Repetition as a pre-determined counter controlled attribute of iteration	repetition of a block of code is controlled and pre-determined by a counter mechanism
	Repetition as a counter controlled attribute of iteration	A)repetition of a block of code is controlled by a counter mechanism but not necessarily pre-determined
		B)As above but there is an assumption that the counter must also have a role in the block of code being repeated
	Repetition as an event controlled attribute of iteration	repetition of a block of code can be controlled by an event such as user input or a counter

These categories represent the different levels of understanding of iteration among participants. In the first category, students understand loop repetition to be counter controlled and assume that a counter is always necessary for a loop to repeat, and that the loop will repeat a predetermined number of times. Here the repetition is experienced as being controlled by a counter and pre-determined by the programmer before runtime. This understanding is evident in the following excerpt taken from an interview with Andrew. For the purpose of anonymity all names are replaced by pseudo names.

> Interviewer: Ok, very good. How do you control how many times a loop runs?
>
> Andrew: Well you need to input that into your loop like if you wanted it to be 9 times you might put in 9 and if you wanted it to be 4 you might put in 4 or something.

In the second category, students understand loop repetition to be counter controlled but not necessarily pre-determined. Here students understand that the number of iterations can be determined by the user at runtime. When asked to explain what a loop is, Glen explains that “A loop is a part of a program that allows certain parts of it to be repeated over and over again for a certain amount of times specified by the user.” Glen is immediately identifying that repetition can be controlled by the user. Within this category, one variation occurred where the understanding is as outlined above, except that there is also an assumption that the counter must also play some role within the loop process. For example, the value of the counter must be printed or added to a sum variable or used to access the contents of an array. During the study students were often observed printing the value of i through each iteration of the loop simply because they thought that they had to. This variation was evident more through researcher observation than interview data, therefore it is not a category in its own right but merits mentioning.

In the third category students understand that the number of times a loop repeats can be determined not only using a counter, but it can also be event controlled. This is experienced as the user being asked if they would like to run the loop again and then depending on the user input, the loop may or may not be repeated. This loop does not use a counter as in the previous two categories. Fred describes one example of this below.

> Interviewer: And have you ever done any loops where the user would control when the loop would end?
>
> Fred: Yea em, they were one of the first loops we did, they were actually quite simple eh, even because you just let the user press yes to redo the loop.

Through further analysis of the interview data, questionnaire answers and following discussion with the module lecturer, one key notion was identified as a possible foundation for these categories. Although there are three types of loop in the java language, students seem to favour the for loop most of the time. This was initially observed by the researcher during the study when students were attempting to complete tasks. Most, if not all of the students, first tried the for loop when writing a program that required a loop. This observation was further confirmed with comments and answers students gave during interviews. When asked how do you decide which loop to use Brian replied, “You just generally use a for loop.” Other students explained how they related different examples with particular loop types. The three categories of description that emerged from this to form this outcome space represent three varying levels of understanding present among the participants. While none of these understandings are incorrect, some simply represent deeper and perhaps more thorough understanding than others.

4. Modifications to Teaching Resources

In an endeavour to progress students’ level of understanding from that outlined in the first category to a deeper level of understanding as outlined in the second and third categories, two primary modifications were made to the teaching resources. Firstly, the order in which the loop structures were taught was adjusted in an endeavour to decrease students pre-occupation with the for loop. Students were introduced to the while loop first followed by the for loop. Secondly, students were presented with a number of problems which endeavoured to separate the counter and process aspects of repetition. In particular these problems involved printing the words of well known songs where repeating words were handled using repetitions. An example of this type of teaching resource is the Five Little Ducks nursery rhyme.

Five little ducks

Went out one day

Over the hill and far away

Mother duck said

“Quack, quack, quack, quack”

But only four little ducks came back

Four little ducks

Went out one day

Over the hill and far away

Mother duck said

Quack, quack, quack, quack"

But only three little ducks came back. Etc.

There is clear repetition between the verses and within each verse ("Quack, quack, quack, quack."). Also, such a problem allows for a deeper understanding of how a repetition might be controlled by an event such as user input. For example, the code might include questions such as, "Would you like to see the words again?" It was expected that the provision of this type of real world problem might encourage a deep understanding of the concept of iteration.

5. Research Study – Phase two

A total of 18 students participated in phase two of the study. Once again, participants comprised students across three different computer science programmes at the National College of Ireland. Similarly to phase one of the study, participants were asked to complete a questionnaire to gather basic statistical information as well as assess students' knowledge on programming iterations. All 18 students agreed to be interviewed and two group interviews took place where students were asked questions pertaining to iteration, and then engaged in discussion of the topics amongst themselves. Once the interviews were complete all data was transcribed verbatim and the data was analysed in the same manner as in phase one of the study. The final outcome space for phase two of the study consists of only two categories of description under the theme of repetition as described in Table 2.

Theme:	**Category Name**	**Category Description**
Repetition	Repetition as a counter controlled attribute of iteration	repetition of a block of code is controlled by a counter mechanism but not necessarily pre-determined
	Repetition as an event controlled attribute of iteration	repetition of a block of code can be controlled by an event such as user input or a counter

The first of these categories is similar to the second category identified in phase one. In this category students understand loop repetition to be counter controlled, but also acknowledge that the number of repetitions is not necessarily predetermined. This is apparent in the following interview excerpt.

> Interviewer: Can we write a loop without knowing beforehand how many times it is going to run?
>
> Bart: Could it be like when, it's eh, say you are using the for loop, you say int i is eh, equal to zero and then i less than the amount of times the user enters.

Here Bart acknowledges that although the loop is counter controlled, the number of times it repeats can be determined by the user at runtime. This conception was further supported by answers given on the questionnaire before the interviews. For example, when asked, "What is a loop?" Ian answered, "A loop is a specific code to repeat lines of code a set number of times."

The second category is similar to the final category identified in phase one. Here students understand that loop repetition can be either counter controlled or event controlled. The following interview excerpt highlights one example provided by students of how loops can be event controlled.

> Interviewer:Can you give me an example of where you might use a loop?
>
> Tom: Running a program again. Like with iBox, you ask the user do you want to run the program again and if they say yes then it does, or no, then it doesn't.

Here Tom demonstrates a clear understanding that the user can control, through his/her actions, how many times a loop will repeat. This again is further supported by a number of written responses to the question, "What is a loop?" John answered, "A set of instructions which is run until a certain condition is true or false," and Peter wrote, "By a statement which will return either true or false." The use of the words true and false in these answers and the acknowledgement that some condition or statement will be tested shows a clear understanding of event controlled iteration as described with the nursery rhyme example in section 4.

While there are still two different categories of understanding present in phase two of the study, there is a clear distinction between outcome spaces of the two phases. There was no evidence in phase two to support the presence of the first category from phase one. This indicates that the changes made to the course material following phase one had a positive impact on students' understandings in phase two, and suggests that using students understandings to inform the development of learning content may have a positive effect on teaching and learning.

6. Conclusion & Further Research

The results of this study provided a reminder of the importance of real-world examples for student learning. It was evident that the greater the number of examples experienced by students the deeper the level of understanding. This is significant for educators as it reinforces the benefits of learning-by-doing. Furthermore, this research provides a reminder for educators that what appears to be student understanding may actually be a mechanical response to a well practiced problem or similar set of problems. Therefore, there is a continuous responsibility for the refinement of teaching resources and indeed assessment to encourage deep learning and understanding among our students. In order to do this there is a need to continuously explore students understanding and indeed misunderstanding.

7. References

Biddle, R., & Tempero, E. (1998). Java Pitfalls for Beginners. SIGSCE Bulletin, 30(2), 48-52.

Cantwell Wilson, B., & Shrock, S. (2001). Contributing to Success in an Introductory Computer Science course: A Study of Twelve Factors. SIGSCE Bulletin, 33(1), 184-188.

Denning, P. (2004). Great Principles of Computing. Paper presented at the grand Challenges in Computing. Teeside University, UK.

Expert Group on Future Skills Needs (2003). The Fourth Report. Ireland: Forfas.

Expert Group on Future Skills Needs (2006). National Skills Bulletin. Ireland: Forfas.

Jenkins, T. (2002). On the Difficulty of Learning to Programme. Paper presented at the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences. Loughborough: England.

Marton, F. (1994). Phenomenography as a Research Approach. International Encyclopedia of Education, 8(2), 4424 – 4429.

Marton, F & Dahlgren, L.O. (1976). On non-verbatim learning III: The outcome space of some basic concepts in economics. Scandinavian Journal of Psychology, 17, 49 – 55.

Stamouli, I. (2007). Learning Object Oriented Programming from the students perspective. Unpublished doctoral thesis. University of Dublin Trinity College, Dublin, Ireland.