A Long-Term Evaluation and Reformation of an Object Oriented Design and Programming Course

Stelios Xinogalos(*), Maya Satratzemi(**)
(*)Department of Technology Management, (**) Department of Applied Informatics
University of Macedonia, Greece
{stelios, maya}@uom.gr

Abstract

In this paper we present important results from a long-term evaluation of an "Object-Oriented Design and Programming" course. In its last form the course is based on the combined use of the microworld objectKarel and the environment BlueJ, while some important modifications on the original teaching approach based on BlueJ have been made.

1. Introduction

In this paper we describe the evolution of an "Object Oriented Design and Programming" compulsory course taught at a Technology Management department. The course was offered for the first time four academic years ago and uses Java, the BlueJ environment [4] and the book "Objects First with Java: A practical introduction using BlueJ" [1]. In its last form the course uses, in addition to BlueJ, the microworld objectKarel.

In the next sections we present important data regarding the course reformation during the last four academic years and the results of the assessment that guided this reformation.

2. Course evolution

The initial design of the course

The course was taught for the first time the academic year 2005-06. Heavily based on BlueJ, the accompanying text book [1] and the established guidelines for teaching with BlueJ [3] we organized eleven two-hour lectures and eleven two-hour labs.

A vast amount of data was collected during the lessons and their analysis identified several difficulties, which were categorized as follows [5]:

Category 1 – "typical" difficulties encountered independently of the programming paradigm.

Category 2 – difficulties attributed to the special characteristics of OOP:

Subcategory 2.1 – constructors
Subcategory 2.2 – object instantiation
Subcategory 2.3 – "set" methods (mutators)
Subcategory 2.4 – "get" methods (accessors)
Subcategory 2.5 – method calling
Subcategory 2.6 – access modifiers
Subcategory 2.7 – object collections
Subcategory 2.8 – inheritance
Subcategory 2.9 – abstract classes & interfaces

Some characteristic difficulties of these two categories are presented in Table 2. Despite these difficulties, students managed to carry out their assignments and comprehend basic OOP concepts. However, it became clear that some adjustments should be made, since some difficulties were attributed to specific features of the course design and BlueJ.

Re-designing the course

Based on the results of evaluating the course the following adjustments were made [6]:

- Students used the interactive GUI of BlueJ for creating objects and calling their methods in the first three lessons and in the 4th lesson (instead of the 11th lesson in the 1st course) they started using the main method.
- Students used the direct manipulation features of BlueJ with more caution and after the 4th lesson they were always asked to achieve the desired result by providing source code too.
- Students started to develop simple projects from scratch much earlier in the semester.
- The lesson about debugging (6th lesson) was omitted and two lessons were devoted on object collections (ArrayLists), which turned out to be one of the most difficult concepts for students.
- Special didactical situations and assignments were designed based on the results of the assessment.

978-0-7695-3711-5/09 $25.00 © 2009 IEEE
DOI 10.1109/ICALT.2009.131

The evaluation of the re-designed course gave better results than the 1st version of the course, as can be seen in Tables 1 and 2. The evaluation showed that [6]:

- *Some difficulties that were attributed to the emphasis given on the features of BlueJ and the late use of main were eliminated* (Table 2): (1) "forgetting to declare the type of the variable that keeps a reference to an object being instantiated" [5], was not recorded in the 2nd course; (2) the percentage of students calling a non-void method as a void method was reduced from 33% to 6%.
- *Some difficulties attributed to the emphasis given on existing projects were addressed satisfactorily:* developing projects from scratch early helped students face more effectively difficulties, such as leaving the body of a constructor empty (subcategory 2.1), omitting the type of an object variable (subcategory 2.2), and directly accessing private fields outside their class instead of using "get" methods (subcategory 2.4, 2.6) (Table 2).

Although the re-designed course gave better results, several difficulties continued to exist. The 3rd year of teaching the course the only change was a refinement of the assignments and the activities carried out at labs.

Reaching a final design of the course?

In our opinion, a major source for students' difficulties is a flawed comprehension of OOP concepts. This is more intense when we have to deal with more advanced concepts, such as inheritance, polymorphism and overriding. This belief combined with the advantages of microworlds for introducing students to programming led us in studying students' conceptual grasp of OOP concepts in a course with BlueJ (the 1st course described in this paper) and a teaching with objectKarel. The most important finding was that the students taught with objectKarel were found to have a significantly better conceptual grasp of OOP concepts than the students taught with BlueJ [7].

The results of this study strengthened our intention to devote the first two lessons of the course for introducing students to OOP concepts with the programming microworld objectKarel, which is based on Karel++ [2]. objectKarel uses a metaphor of robots carrying out various tasks in a restricted world. The use of objectKarel aimed at presenting in a clear and concise way the most fundamental OOP concepts: object, class, message/method, object instantiation, inheritance, polymorphism and overriding.

Another change in the course was the sequence of activities/assignments used for comprehending ArrayLists: object diagrams were used for comprehending the structure and operations of an ArrayList; students filled in blanks that represent error prone elements in an excerpt of code, in order to think more consciously about ArrayLists; students developed projects with ArrayLists from scratch.

The first results of the course based on objectKarel and BlueJ seem to be positive (Tables 1, 2). However, we believe that the impact will be much deeper regarding the concepts of inheritance, polymorphism and overriding, which have not been examined yet.

3. Evaluation results

Our main concern regarding the use of objectKarel was whether the acquired knowledge would be transferred afterwards to Java [4]. The experience of using objectKarel was, however, more positive than expected. More students participated in the lessons, completed the assignments and gained confidence in their ability to program, in comparison with previous years. What is more important is that students did transfer the knowledge acquired in the context of objectKarel to Java. In the 3rd lesson, when Java and BlueJ were used for presenting class definitions students made the connection with the concepts taught in objectKarel and asked the teacher for confirmation.

In Tables 1 and 2 we present comparative results for the three versions of the course, based on written exams. Table 1 presents the percentage of students that answered questions concerning main OOP constructs and the percentage of completely correct answers. Table 2 presents comparative results regarding the frequency of specific difficulties from the categories recorded in the 1st course. The percentages are calculated based on the number of students that actually answered the corresponding questions.

Table 1. Correct answers.

	1st: 05-06		**2nd: 06-07**		**3rd: 08-09**	
	Answered (%)	**Correct (%)**	***Answered (%)***	**Correct (%)**	***Answered (%)***	**Correct (%)**
(2.1) constructor	*54*	**54**	*89*	**58**	*94*	**72**
(2.3) "set" method	*79*	**50**	*81*	**56**	*95*	**73**
(2.4) "get" method	*88*	**79**	*84*	**72**	*97*	**87**
(2.7) ArrayList	*76*	**4**	*59*	**8**	*47*	**14**
(2.2, 2.5) main	*45*	**0**	*52*	**3**	*76*	**9**

Table 1 shows that students' active participation and achievements in the exams improve, as the course evolves. The only exception is ArrayLists that are still a great source of difficulty. The number of students that comprehends constructors (2.1), "set" methods (2.3), "get" methods (2.4), and "main" that involves object instantiations (2.2) and method calling (2.5) increases importantly in the last version of the course.

The results regarding specific difficulties (Table 2) show a gradual improvement for most of the subcategories, and in few circumstances slight variations. An exception is the 1st category of "typical" difficulties that are independent of the programming paradigm and refer mainly to parameters, return values and calling void and non-void methods. These are concepts taught in a prior "Computer Programming" course based on C, and it is clear that special attention must be paid on teaching them.

Table 2. Students' difficulties (%).

		1st	2nd	3rd
1	Missing arguments in method calls	53	33	45
1	Missing ( ) in methods without arguments	27	6	0
1	Calling a non-void method as void	33	6	30
2.1	The parameters in a constructor have the same name as fields but "this" is not used	8	0	2
2.1	Fields are assigned values of undeclared identifiers and not of the parameters	0	7	7
2.1	Wrong initialization of fields	29	33	23
2.1	The body of the constructor is empty	8	0	0
2.2	The type of object variables is missing	13	0	0
2.3	The name of the field being updated is used as argument in "set" methods	0	6	0
2.3	Declaring as return type of a "set" method the type of the parameter (instead of void)	15	12	5
2.4	Method - field conflation in "get" methods	7	0	0
2.4, 2.6	Direct access of private fields outside their class instead of using a "get" method	32	3	0
2.5	Calling a method without an instance	7	12	0
2.7	Manipulating an ArrayList			

Finally, Table 2 shows that difficulties concerning ArrayLists remain. However, this is not completely correct. Although the majority of students cannot write code for manipulating an ArrayList, improvement has been made. We cannot compare students' achievements in the three versions of the course, since the first year of teaching the course our analysis was mainly qualitative. Nevertheless, this is possible for the next two versions of the course. The comparative results (Table 3) were recorded in a similar method developed by students in both courses. The results in Table 3 show an improvement regarding basic operations on ArrayList collections. Nearly all the students (97%) that implemented the related method iterated the ArrayList collection, while most of them (72%) retrieved the objects stored in it correctly in the last teaching of the course. Furthermore, all the students accessed the fields of the retrieved objects using "get" methods.

Table 3. Students' difficulties with ArrayList (%).

	2nd	3rd
The ArrayList is not iterated	8	3
Accessing private fields outside their class		
- Direct access without an instance	47	0
- Direct access: <object>.field	3	0
Retrieving objects from an ArrayList	45	28
- while loop is used but objects are not retrieved	13	17
- the name of the ArrayList field is used as type of the variable for the object retrieved	13	0
- the object is not assigned to a variable	5	0

4. Conclusions

In this paper we presented the long-term evaluation and reformation of an OOP course based on BlueJ. The last change in the course was the use of objectKarel, which is anticipated to help in comprehending concepts, such as inheritance. It seems that, sometimes, more than one tool is needed in order to support demanding cognitive areas, such as programming. These tools can be complementary educational programming environments, tutorials, videos and so on. In any case, continuous evaluation of teachings is necessary in order to reach valid conclusions, develop appropriate educational material and move to reformations. When the results of such studies are made available, great support is provided for improving the teaching of OOP.

5. References

[1] Barnes, D. & Kölling, M., *Objects First with Java: A practical introduction using BlueJ*, Prentice Hall, 2004.

[2] Bergin, J., Stehlik, M., Roberts, J., and Pattis, R. Karel++ - A Gentle Introduction to the Art of Object-Oriented Programming. 1997, John Wiley & Sons.

[3] Kölling, M. & Rosenberg, J., (2001), Guidelines for Teaching Object Orientation with Java, *ACM SIGCSE Bulletin*, Vol. 33 Issue 3, 33-36.

[4] Kölling, M., Quig, B., Patterson, A., & Rosenberg, J. (2003), The BlueJ system and its pedagogy, *Journal of Computer Science Education*, 13(4), 249-268.

[5] Xinogalos, S., Sartatzemi, M. & Dagdilelis, V. (2006), Studying Students' Difficulties in an OOP Course based on BlueJ, *IASTED International Conference on Computers and Advanced technology in Education*, Lima, Peru, 82-87.

[6] Xinogalos, S., Satratzemi, M., Dagdilelis, V. & Evangelidis, G. (2007), Re-designing an OOP based on BlueJ, *Proceedings of the 7th IEEE International Conference on Advanced Learning Technologies,* Niigata, Japan, 660-664.

[7] Xinogalos, S. (2008), Studying Students' Conceptual Grasp of OOP Concepts in Two Interactive Programming Environments, *Springer Communications in Computer and Information Science*, Vol. 19, 578-585.