Available online at www.sciencedirect.com

ScienceDirect

Procedia - Social and Behavioral Sciences 143 (2014) 352 – 357

Procedia
Social and Behavioral Sciences

CY-ICER 2014

Design patterns in the teaching of programming

Rostislav Fojtik[a]*

[a]*University of Ostrava, 30.dubna 22, Ostrava, 70100, Czech Republic*

Abstract

Teaching algorithmization and programming has been recently going through big changes trying to react to the dynamic development of software industry. Previously used methodical process, development models, or programming languages do not conform to current requirements. The results of the surveys in primary and secondary schools, we can say that the teaching of programming and algorithms are not sufficiently exploited. The aim of this paper is to present practical experience of the author teaching programming and the possibilities of using design patterns in the teaching of programming. According to the performed analyzes the procedures and methodologies of teaching programming shows that Design Patterns are used only marginally. For these reasons, students learn to improper practices that subsequently applied in practical solutions programs. According to the experiments show that the correct use of the teaching of design patterns can improve student performance in programming

Keywords: Design Patterns, object-first, object-oriented programming, teaching programming.

1. Introduction

Teachers often ask which programming language to start with. This approach is, however, unsuitable. The first question when creating a programming course should not deal with a programming language, but with selection of a suitable paradigm and the objective of the course. If the objective is to teach students to create algorithms and think logically, the most suitable, particularly for students of primary schools, seems to be microworlds, such as Karel or Logo. These tools are well-arranged, simple, and demonstrative. The students usually see their results of the advance in a graphical visualization. The tools do not take long to learn the language rules and users can really focus

* Corresponding author: Rostislav Fojtik Tel.: +420 603 167 678.
E-mail address: rostislav.fojtik@osu.cz

doi:10.1016/j.sbspro.2014.07.493

on creating algorithms and simple programs and they do not wander in language complexities and concrete details. However, if the objective is to learn students to program so that their knowledge and skills could be used in practice, first of all we have to choose a suitable paradigm. Currently, the most spread software development approach is object-oriented.

2. Research in primary and secondary schools

The main purpose of the survey was to determine how access primary and secondary schools to teach programming. [2] [3] The first study group included 63 secondary schools. 84 % of respondents are involved in teaching programming. The most frequently used language was Pascal. Unfortunately, some teachers confuse development environment with the programming language. An example might be a development tool Delphi that over 10 % of respondents mistakenly considered as a programming language.

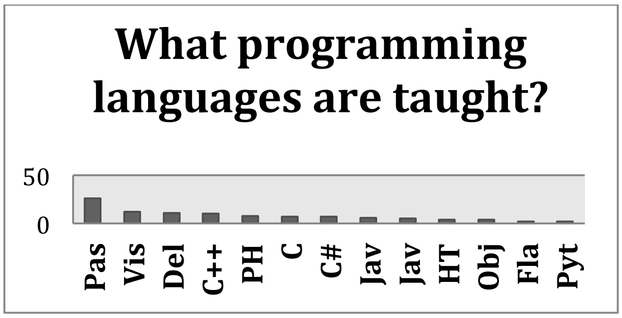

Fig. 1. *The use of programming languages in secondary schools*.

The second group consisted of 71 primary schools and 15 grammar schools (lower level). 32 % of respondents taught programming within the scope of computer equipment or pet ring teaching programming. 22 % of respondents taught programming in earlier times, and the remaining 46 % of programming not taught. One of the reasons for the low representation of teaching programming is still a lack of approbation teachers.

33 % of respondents use for learning the basics of algorithms and programming uses the microcosm. Microworlds are development tools. Their main advantage is simple to use and clarity. These tools are often tailored for smaller children, but they can be successfully applied to high school. They are most used Logo, Karel and Baltik.

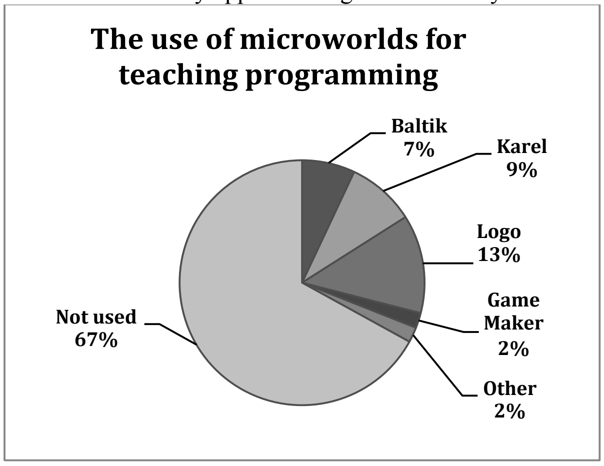

Fig. 2. *The use of microworlds in education programming in elementary schools*.

We performed an analysis of educational programs, curricula and thematic programs selected 27 secondary schools. There were 14 secondary schools (college) and 13 secondary technical or vocational schools. The graph shows the number of schools teaching algorithmic programming or as a separate subject or as part of other courses.

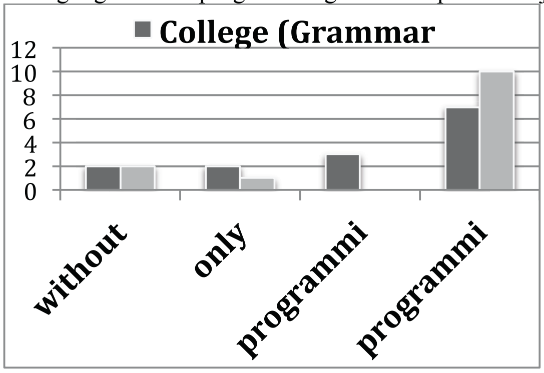

Fig. 3. *Teaching programming in selected schools*.

3. Selection of programming language

Selection of a suitable programming language is an important step in preparation of educational course. It is necessary to select a language conforming to the object-oriented paradigm. Most languages do not have object attributes implemented very suitably (e.g. Pascal). On the other hand, there are languages that are of high-quality from the OOP point of view, but they are less used in practice (e.g. Smalltalk).

Programming languages can be divided into two groups:

- pure OOP languages
- hybrid OOP languages

Programming languages from the first group use only the object-oriented model. We cannot create independent functions that are not a part of the class. We also do not use global variables declaration. We can classify here languages Smalltalk or Eiffel. A disadvantage of programming language Smalltalk is less frequent use in practical projects. This language gives a basis for Objective-C, which serves as a basic tool of programming in operating system Mac OS X. Unfortunately, it is unusable in other systems. In addition, syntax of these languages differs from a large number of popular programming languages. Objective-C code example:

```
MyClass *myObject = [[MyClass alloc] init];
[myObject setAttribute: 1];
```

C# or Java code:

```
MyClass myObject = new MyClass();
myObject.setAttribute(1);
```

On the other hand, hybrid programming languages use as a model object-oriented, procedural, and imperative models. Thus they merge various paradigms and it is up to the programmer which approach will be preferred. A typical representative of this group is programming language C++, which builds on a purely non-object approach of language C and its object extension.

Another division of programming languages depends on the extent of type check. So-called static languages are based on knowledge of data types and they perform more frequent type checks in the compilation time. C++ can be again taken as an example. Dynamic languages (such as Smalltalk or Ruby) have weaker knowledge of types in a part of variables declaration and they usually carry out checks during the programme runtime. Dynamic languages are usually interpreted.

Programming language C++ is not suitable for an introductory course. It concerns a relatively complicated language, which does not have strong control mechanisms. Work with pointers requires its profound knowledge. If not, the program can contain less visible errors. It concerns a language that enables the programmer almost everything, but it presumes that the programmer knows exactly what he is doing. Moreover it belongs to so-called hybrid languages, i.e. languages connecting procedural and object-oriented approach. Programs can be created according to one or the other paradigm, or we can even use both approaches in one code.

Suitable languages for an initial course of object-oriented programming can be languages Java or C#. Syntactic rules are very similar and currently used in other programming and scripting languages.

Before teacher draw thematic plan should make clear goal of teaching programming:

- Learn the basics of algorithms - here is advisable to use a simple and straightforward means such as microworlds.
- Master the specific language - pupils already know the basics of programming and given the profile of the graduate study program to handle a specific programming language.
- Learn how to program by modern standards - it means focusing on object-oriented approach, which is applied in current practice most often.

4. Design patterns in teaching

Design patterns describe high quality practical solutions to recurring programming problems. There are 23 basic design patterns that can be divided into three groups [6]:

- Creational Patterns
- Structural Patterns
- Behavioral Patterns

Design patterns are a professional programmer practice relatively well known and used. If we look into the current programming textbooks, we find that the theme of design patterns is often omitted or is included at all until the end of the course.

Currently begins in programming using a system that from the beginning emphasizes the architecture of the program and hence the appropriate use of design patterns. [4] Students are being informed individual elements of a programming language and algorithmic procedures. Students from these courses usually better understanding of the principles of good program design.

What is important to include the teaching of programming topics of proper analysis and design programs, such as is seen in the results of the competition programs for high school students (college). [7] The contestants include in their schools among the best students in the subject of computer science (programming). Students usually their contest forms separately outside the classroom. Unfortunately, many programs do not have the correct structure. Students have little information about the architecture and analysis programs. They usually solve their programs of trial and error, and a large part of the programs entered in the contest is not suitable structure.

Consistent requiring MVC architecture leads students to practice good habits. Their applications often correspond to the correct structure. A survey of the winning students to demonstrate knowledge of design patterns contributes to quality proposals applications. Students were informed architecting applications and gave her a large part of energy solutions. It is therefore appropriate to include design patterns at the beginning of training courses. One of the oldest and most frequently used patterns is the Model-View-Controller.

The following example, we show the benefits of using design patterns in teaching. It is necessary to create a program that will manage shopping tropical fruit in a particular country, depending on the season. If students are familiar with object-oriented programming, is usually invades ProductAfrica design classes (e.g. fruit from South Africa) and ProductSpain (fruit from Spain). If their knowledge of object-oriented design will be more than just knowledge of the concepts object and class design for these common ancestor class or interface. The main program will then appear in subsequent solutions, which, depending on the particular month of the year will create a variety of objects (buy fruits in different countries):

```
for(int month=1; month<12; month++)
{
   if (month >=5 && month <=9)
   {
ProductSpain p = new ProductSpain();
      p.fromCountry();
   }
   else
   {
ProductAfrica p=new ProductAfrica();
      p.fromCountry ();
   }
}
```

The solution looks good only until such time as we need to modify the program to other conditions of purchase tropical fruits. For example, we need to shop in other countries, depending on other external factors. In the preceding code will need to be attributed to other conditions and significantly interfere with the main algorithm. Would appear more appropriate to use Factory Method design pattern [1].

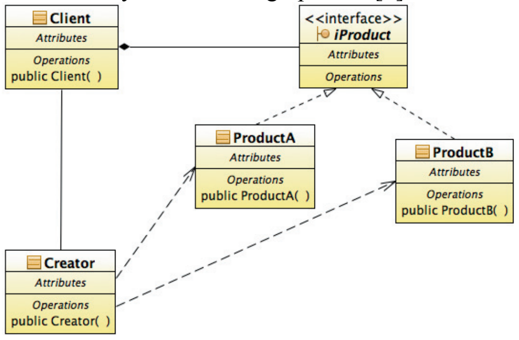

Fig. 4. *Class Diagram - Factory Method pattern*.

Part of the design pattern is a Class Creator. Instance Creator class may serve as a buyer of tropical fruits. Responsibility for the correct purchase in a particular month move from the main program on this subject. You can only edit the class Creator and the main program may remain unchanged.

```
Creator buyer = new Creator();
IProduct prod;
for(int month=1; month<=12; month++)
{
   prod = buyer.FactoryMethod(i);
}
```

We tested the practical pedagogical experiments in teaching programming in the Department of Computer Science and University of Ostrava in the courses Programming in C++, Development of mobile applications for iOS and programming courses for high school students. The quality of the created programs was dependent on whether students learn design patterns and architecture programs. Students do not understand the design patterns only as an add programming, but as its an important and integral part of the development programs.

5. Conclusion

Experience and results of experiments that the use of design patterns and the object-oriented approach in teaching programming can be successfully used not only at universities, but at lower level of educational institutions as well. And not only as a supplement, but as the principal programming paradigm. However, it is very important to set a suitable methodology of education. An important role in this process is played by clearness, adequacy, and quality of examples.

Design patterns currently represent one of the most powerful tools in the design and subsequent implementation of programs. Design patterns should not be included as the last chapter books and training courses, but it is better to mingle commentary from the outset. Applications of design patterns in teaching will not only students but also teachers better understand and correctly define the objectives of teaching programming.

References

BISHOP, J. *C# 3.0 Design Patterns*, O'Reilly Media 2008, ISBN 0-596-527773-X

FOJTÍK, R., JIRÁSKOVÁ, D. Programování na základních a středních školách. *Objekty 2009*. Gaudeamus, Univerzita Hradec Králové, 2009. s. 75-83. ISBN 978-80-7435-009-2.

FOJTÍK, R., DROZDOVÁ, M. Teaching of programming at secondary schools. *ICTE 2009.* Ostrava: University of Ostrava, 2009. s. 77-81. ISBN 978-80-7368-459-4.

GOVENDER I., From Procedural to Object-Oriented Programming (OOP) - An exploratory study of teachers' performance, *SACJ*, vol. 46. 2010. ISSN 1015-7999.

SAELI, M., PERRENET, J., JOCHEMS, J., ZWANEVELD, B., Teaching Programming in Secondary School: A Pedagogical Content Knowledge Perspective, *Informatics in Education - An International Journal*, vol. 10_1/2011, ISSN 1648-5831

STERKIN, A., Teaching Design Patterns, (online), URL: <http://ww1.ucmss.com/books/LFS/CSREA2006/FEC4388.pdf>

Soutěžní přehlídka studentských programů, (online), URL: http://www.gvoz.cz/akce/2013_spsp.php