Research of Methods of a Multidisciplinary Approach in the Teaching of Algorithm Development and Programming

Štěpán Hubálovský
Department of Informatics, Faculty of Science, University of Hradec Králové, Hradec Králové, Czech Republic
stepan.hubalovsky@uhk.cz

Abstract

One of the most important tasks in teaching of algorithm development and programming is to use a suitable method of developing theoretical knowledge of algorithm development and programming. Drawing on his experience, the author has found that the best method to be used is the system, multidisciplinary approach. The paper focuses to introduction the system and multidisciplinary approach to education of algorithm development and programming and development of algorithmic thinking as well as to methods of elimination of problems associated with the practical skill and techniques of structured and object oriented programming. The methodology of research investigation by evaluation of the errors in principles of object oriented programming is presented too. The results of the research investigation confirmed positive effect of system and multidisciplinary approach to the correct use of algorithmic design, development of algorithmic thinking and development of thinking in principles of object oriented programming.

Keywords


Introduction

Teaching of the basics of algorithm development and programming is realized not only at universities but also in secondary schools. To a limited extent these issues can be also met in some primary school, especially in specialized classes or extracurricular activities.

The output of the process algorithm development and then programming is computer program - computer simulation. Computer program is an application that runs on a computer and that has the standard user interface. Standard user interface is based on the windows (forms) in which other active elements (objects) are located - the application must be in accordance with the principles and paradigms of object-oriented programming.

The results of the research conducted among students of specialized secondary school and university students are discussed in the paper. In the research we have studied influence of two seemingly unrelated aspects that affect the teaching of algorithms and programming.
The first aspect is related to discussions on whether it is better to teach the algorithm development and programming either step by step from algorithmic constructions through structured programming to object oriented programming or to start with object-oriented programming and structured programming understand as only like subset. Both alternatives of teaching have their opponents and proponents. Proponents of structured approach proceed mainly from their practice, which in teaching algorithms and programming has been working for decades. The proponent of OOP based teaching is e.g. (Pecinovský, 1999).

The second aspect influencing the teaching of the algorithm development and programming is the choice of methods. In principle, it can be chosen between two fundamental, opposite methods. The first is primarily based on mono-disciplinary approach, where teaching is clearly structured and the emphasis is on interpretation and understanding of algorithmic structures. Understanding the algorithmic structure is the primary objective of the teaching. In contrast, teaching can be based on a multi-disciplinary approach. The understanding of the algorithmic structures is secondary, but significant issues. In this context, we can talk about the systems approach. Algorithm development and programming becomes more of a means to reach the goal end, not its own goal.

System approach in education

System approach is closely related to the concept of system. The system in general terms can be based on e.g. understood as a defined set of elements and relationships that creates an integrated whole. The system is also part of external reality and communicates with its environment through its inputs and outputs. System with no links to the surroundings is closed system. In teaching practice, we find open systems i.e. systems with inputs and outputs.

In view of this perspective we can consider a system of class as a group of students who interact (communicate), and interact with its environment, i.e. the teacher, other classes using the inputs and outputs. Under the system, rather a systemic approach, however, can be understood also abstract issues like learning style. For a system approach is generally considered such way of solving the problems, where phenomena and processes are studied comprehensively in their internal and external contexts (Wilson, 2001). System approach in pedagogy means formulation, understanding and solutions of the studied problem under the consideration that the corresponding processes, events and phenomena that objectively exist in the world and which are transformed into the model learning situations.

In connection with the concepts of system and system approach is necessary to mention the other term that is commonly used in pedagogy – interdisciplinarity. This concept can be understood e.g. based on (Checklan and Poulter, 2006) as a method of linking and active cooperation between different sciences in order to achieve an integrated and synergistic results in theoretical and practical professional activities, science and research.

The systems approach rather than interdisciplinarity talk about multidisciplinarity. A multidisciplinary approach is used mainly in the method of solving problems and creating technically functional units, such as the creation of the computer program – see e.g. (Hubálovský, 2011). The main reason for the introduction of multidisciplinarity in connection with the system and a systems approach is that the research system as a whole requires a high degree of highly specialized knowledge in different disciplines and these specializations has to be closely linked by mutual connection. Only a multidisciplinary approach will ensure that the whole system will be studied systematically and comprehensively.
Although the interdisciplinary approach in the context of learning process is frequently discussed, the concept of system approach in educational practice is not sufficiently specific and widely implemented (Checklan and Poulter, 2006). Unfortunately, this fact concerns also learning of algorithm development and programming, which from a practical point of view without the system and multidisciplinary approach cannot do.

In the following text the both aspects influencing the teaching of algorithms and programming, namely:

- structured form versus object oriented form of teaching and
- mono-disciplinary versus multi-disciplinary method of teaching

will be focused from a system approach point of view.

**System approach and structured and object programming**

As mentioned in the introduction, the first aspect that influences learning algorithms and programming is influenced by the form of performed teaching:

- structured form, that teaching is divided into learning algorithms, structured programming and object-oriented programming in the end;
- object oriented form, i.e. from the beginning of the instruction focuses on object-oriented programming, with the principles of the algorithms are part of this instruction.

*Structured programming* is programming based on the structure of the program, which comes strictly from the algorithm flowchart. From the system approach point of view the algorithm as well as structured program (written in any structured language - Pascal, C++, VB Script) can be understood as system, because they have properties of the system – algorithm interacts with its environment through inputs and outputs, consists from elements that are affected by interactions. Another division algorithm to subsystems is possible, from a practical perspective, however unreasonable.

In this context it is necessary to mention what types of exercises are used for training the algorithm development and structured programming. The exercises reflect two facts. Firstly, in the past, early in the courses of programming (mainly structured programming - Pascal, Basic, etc.) has been teaching of programming realized by teachers who also taught mathematics, or had to mathematics very close. Second, math problems are basically the simplest tasks, can be clearly described, defined and then developed by algorithm and rewritten to the program structure. That, however, seems at first glance a logical and simple, brings disadvantages. Algorithm development and structured programming explained by the mathematical tasks usually focus on rewriting the mathematical equations and formulas to the algorithms regardless of their complex systems integration with the exercises from real life. Used tasks are often artificial and divorced from reality. System and multidisciplinary approach is missing. Students, who do not have sufficient mathematical skills, do not understand the task and it can result in resistance to the algorithm development and subsequently to programming.

In contrast, the basic paradigm of *Object oriented programming (OOP)* is to model on the computer the real-world situation. The OOP applications are developed based on already created components. The basic terms of OOP are object, event abstraction, encapsulation, inheritance and polymorphism. From a system approach point of view the objects can be understood as open subsystem of whole application. Every object - the subsystem is a complete system - consisting of elements (a list of properties, event handlers), communicates with its environment through inputs (events, parameters) and outputs (methods and parameters).
From the above mentioned it is clear that both the structured and object-oriented programming can be viewed using the systems approach. The difference is in complexity of "result" of activity. In the structured approach programmer creates the system - i.e., links connecting the elements and creates a system - program (application). In object oriented programming, programmer works with already made objects (subsystems) and defines their properties, inputs and outputs, i.e. links to the external environment. Here he creates a system - a program (application) but by means of the "connection" of the subsystems (objects).

Multidisciplinary approach in education of programming

The second aspect influencing the development of algorithmic thinking is the choice of methods. As mentioned in the introduction, we can choose between mono-disciplinary and multidisciplinary approach.

Multi-disciplinary approach according to (Milková, 2011) is closely related to a systems approach, modeling and computer simulation.

Modeling

Modeling is a method that is often used in professional and scientific practice in many fields of human activity.

The main goal of modeling is not only describing the content, structure and behaviour of the real system representing a part of the reality but also describing the processes.

The process can be understood as series of transformations that changes the input values to output values. From the system point of view the process is dynamic system in which the values of the characteristic of the system elements are changed under the influence of the external elements.

The models are always only approaching of the reality, because the real systems are usually more complex than the models are. The model is always to be understood as simplification of the original – see e.g. (Kučera, Houška and Beránková, 2008; Houška and Beránková, 2007).

The first step in the process of computer simulation is creation of conceptual model of the studied real system / real process. Conceptual model can be represented in different way. The most used representations are:

- mathematical equitation;
- process charts.

Simulation

The process of modeling is closely related to the simulation. Simulation can be understood as process of executing the model. Simulation enables representation of the modelled real system and its behaviour in real time by means of computer. The simulation enables also visualization and editing of the model – see e.g. (Sokolowski and Banks, 2009).

A typical simulation model can be written both through specialized programming languages that were designed specifically for the requirements of simulations, or the simulation model can be created in standard programming languages and spreadsheets (MS Excel) - see e.g. (Lovászová and Hvorecký, 2005).

The whole process of transformation from a real system, the simulation model and its visualization is shown in Figure 1.
Another important trend that follows from the paradigm of object oriented programming is that the learning is fundamentally accompanied by multidisciplinary approach. Exercises are based on the modeling of real phenomena, events and processes. Using object oriented programming, it is possible to create complex tasks in which students have to first define the problem, i.e. create simplified model and would be able to describe this model using the principles of the algorithm, i.e. determine the input and output variables and the process by which input data are transformed into output data. Finally, the student should be able to create object oriented application in the appropriate programming language as a computer simulation of the model.

The exercises realized by the multidisciplinary principles can be applied:

- In the field of sciences - in case of models of physical and chemical processes and phenomena;
- In the technical field - in case of the process control of machines and simple robots;
- In the humanities and social studies - in case of the processes and phenomena associated with this issue;
- In management - in case of the management processes and quality control processes.

Mathematical calculus is essential in a “model” method, but only as a secondary issue. The standard method of a simple rewriting of mathematical expressions and equations into algorithms and then into the programs is maximally eliminated while using this approach.

**Research investigation**

Let us return now to the questions of the influence of both the above discussed aspects of the development of algorithmic thinking and learning algorithms and programming. The answer to this question we find in research investigation conducted among secondary school students and high school bachelor students. Students were divided into three groups.

The first group (hereafter designated as Group 1) were taught in three stages from the theory of algorithms (1st semester), through structured programming (2nd semester) and object oriented programming (3rd semester).

The teaching the students in the second group (Group 2) were based on object oriented programming approach. Algorithms development and structured programming principles have been an ongoing part of teaching object oriented programming.

In both groups, lessons proceeded by mono-disciplinary approach without application of methods of systematic and multi-disciplinary approach. The workshops of these two groups were primarily based on the standard basic algorithmic structures, which are mentioned, for example (Milková, Hubálovský and Pražák, 2010).
In the third group (Group 3) the teaching were based on the object oriented approach and by the method of system, multi-disciplinary approach. The seminar of algorithm development is practiced within the implementation of computer simulations of conceptual model of the real system.

At the end of the course program of study all students were asked (in the final essay) to create object oriented application. The essence of this final seminar work is create a mathematical model of real system and corresponding simulation program in form of object oriented application, in which there are assessed not only application functionality but also

- errors in the principles of object oriented programming and
- errors in the principles of algorithm construction.

Errors in object oriented programming

These are the errors by which we assess how students manage the principles of the object oriented programming.

Errors, which students engage in their application, can be divided into a number of criteria – see e.g. (Hubálovský and Šedivý, 2010).

First, they are the errors essential for program running. Most of these errors are detected by the compiler during the translation process; others are detected during the testing process.

Second, they are the errors which do not shape the program functionality. These errors arise during the program’s structural development. These types of errors, in many cases, are not recognized by programmers, because the program appears to work properly. These errors, however, are fundamental, because of cooperation in program-creating team. These errors were focused in our research investigation. These types of errors are detailed specified in Table 1 in the section Results of the research investigation.

Errors of algorithm development

Within these errors, we perform the evaluation of errors committed by students in algorithmic structures, i.e., how the basic properties of the algorithm are complied. The main emphasis in the evaluation was placed on the correct identification of inputs and outputs of the algorithm and generality.

Within the errors of algorithm development the errors in declaration of variables are evaluated too. Each variable should represent some aspect of the real system as well as conceptual (mathematical) model. We distinguish three types of variables:

- **Input variables** represent the inputs of the real system. There is a need to have really chosen those variables which are focused during the description of the real system. It makes no sense to choose such an input data variable that the simulation does not change, or which does not affect the simulation process. On the other hand, the number of input variables must be sufficient so that the simulation model is as general as possible, i.e., solve a whole class of similar models.

- **Output variables** represent those features of the studied system that led to the creation of simulation model, i.e. the output variables provide information that is either explicitly or implicitly necessary to achieve the objectives of process modeling and simulation. Here again, similar considerations for choice and number of output variables are valid as input variables.
- **Auxiliary variables** are for user of simulation program unavailable, only serves to temporarily preserve the values of quantities needed to run the simulation program.

These types of errors are detailed specified in Table 1 in the next section.

**Example of final seminar work**

*Problem definition*

A homogeneous cylinder of radius $r$ rolls the plane from the left to the right with velocity $v$. With respect to $r$, $v$ and $\alpha$ it has to be decided whether the cylinder rolls over the edge or rebounds from the edge. The situation is shown on Figure 2. If the cylinder, under specified conditions, rebounds, it is necessary to find a place on the inclined plane where the cylinder snaps. The situation has to be analyzed on the basis of available physical laws.

Full solution of this task is shown e.g. in (Hubálovský, 2010). In this paper the solution will be only hinted.

In the first step the student should provide kinematics analysis find mathematical model of the trajectory:

\[
\begin{align*}
  x \in (-\infty; 0): & \quad y_p(x) = r \\
  x \in (0; x_{impact}): & \quad y_p(x) = r - 5 \left( \frac{x}{v} \right)^2 \\
  x \in (x_{impact}; \infty): & \quad y_p(x) = (-\alpha)x + \frac{r}{\cos \alpha},
\end{align*}
\]

(1)

**Figure 2: Example of entering seminar work**

In the second step students should define the appropriate algorithm, input, output and auxiliary variables of the problem solution and the whole situation should be animated with appropriate tools.

**Source code of the solution in Pascal programming language is as follows:**

```pascal
default
begin
  read (r, v, \alpha)
  Rebound := False;
  I := 0;
  while I <= 200 do
    begin
      X := I * 0.075;
```
\[
Y_c := f_c(X,r,v,a) \\
Y_p := f_p(X,r,v,a) \\
\text{if } Y_c < Y_p \text{ then} \\
\begin{align*}
&\text{Rebound} := \text{True}; \\
&\text{Ximpact} := X; \\
&\text{end}; \\
&I := I + 1; \\
&\text{end}; \\
&\text{write}(\text{Rebound, Ximpact}); \\
&\text{end}.
\]

Results of research investigation

Evaluation of errors was done so that if the same errors repeat several times in the computer application, repeating the same errors several times, it was counted only once.

Types of errors and their percentage in the three groups of students are summarized in the Table 1.

<table>
<thead>
<tr>
<th>Type of Errors</th>
<th>Percentages of errors of object oriented programming</th>
<th>Percentages of errors of algorithm development</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Group 1</td>
<td>Group 2</td>
</tr>
<tr>
<td>Complex classes</td>
<td>23 %</td>
<td>12 %</td>
</tr>
<tr>
<td>Inheritance</td>
<td>45 %</td>
<td>27 %</td>
</tr>
<tr>
<td>Labelling</td>
<td>12 %</td>
<td>13 %</td>
</tr>
<tr>
<td>Encapsulation</td>
<td>21 %</td>
<td>08 %</td>
</tr>
<tr>
<td>Complex method</td>
<td>19 %</td>
<td>13 %</td>
</tr>
<tr>
<td>Repeating code</td>
<td>22 %</td>
<td>10 %</td>
</tr>
<tr>
<td>Working with exceptions</td>
<td>15 %</td>
<td>14 %</td>
</tr>
<tr>
<td>Logic in the presentation layer</td>
<td>21 %</td>
<td>06 %</td>
</tr>
<tr>
<td>Difficulty managed code</td>
<td>23 %</td>
<td>08 %</td>
</tr>
<tr>
<td>Encapsulation</td>
<td>25 %</td>
<td>12 %</td>
</tr>
<tr>
<td>Complex method</td>
<td>12 %</td>
<td>13 %</td>
</tr>
<tr>
<td><strong>Average percentages of errors of OOP:</strong></td>
<td><strong>22 %</strong></td>
<td><strong>12 %</strong></td>
</tr>
</tbody>
</table>

Discussion of the results

The above results indicate that students taught by the principles of object oriented principles (Groups 2 and 3) make mistakes in object-oriented applications to a lesser extent. This is clearly
demonstrated by roughly half and a lower incidence of errors associated with pure object oriented approach - errors in the inheritance, encapsulation, logic in the presentation layer, etc. The problems in object oriented approach in Group 1 can be explained and related to the bad habits fixed in the course of teaching structured programming and their transfer into the object oriented programming.

Furthermore, the results of the survey confirmed the positive effect of system and multidisciplinary approach to the correct use of algorithmic design and development of algorithmic thinking. By analysis of these types of errors were found that students whose learning of algorithm development and programming is done by mono-disciplinary form (Groups 1 and 2) are not able to identify inputs and outputs, and often make mistakes in generality.

We can say that the optimal solution for education of algorithm development and programming is object oriented approach together with a multidisciplinary approach.

Conclusion

Traditional education of algorithm development and programming in the form of algorithm construction and structured programming fixes in students habits that are hard to overcome the object oriented programming. On the contrary, teaching programming from the beginning of the object oriented approach fixes the students' systems thinking and systems approach, which just creates in students attention shift from parts to whole, from elements to objects and relationships between them and from structures to processes. The optimal solution of teaching of algorithms and programming is system and multidisciplinary approach that is based on identification real systems, processes, events and interaction. Without this broad-based systemic approach is education of programming and algorithm development inefficient.

Acknowledgement

This paper is published thanks to the financial support of specific project of Faculty of Science, University of Hradec Králove.

References


