Evaluation of Algorithmic Thinking of Students Using Control Testing Environment

Stepan Hubalovsky and Ondrej Korinek

***Abstract*—**Evaluation of students' exercises from programming is for teaching difficult. There is a number of possibility of checking programmer knowledge and skills of students. The teacher has to determine criteria of evaluation. The criteria of evaluation depends on choose approach in learning of programming. Different evaluation criteria they will be used for students of object - oriented programming, command programming or declarative programming. The paper deal with problems of checking of students' programmer exercises. Different means and tools for checking are mentioned. The universal testing environment and possible testing exercises are presented in the paper too. Results of the research are mentioned in the final part of the paper. Research deal with comparison of results of student learned by two different approaches of programming using universal testing environment with respect to algorithmic thinking of student. The presented approaches of programming are *Objects-First* and *Objects-Later*.

***Keywords*—**Algorithmic thinking, approaches in programming, *Objects-First*, *Objects-Later*, universal testing environment, method of testing of knowledge.

I. Introduction

"My program is working. How it is possible that I was worse evaluated than my colleague?" This question has to be often solved by teacher after evaluation of students' exercises from programming. Evaluation of programmer exercises is lengthy, arid and nor much favorite activities of teacher of programming. Independent evaluation is also missing if teacher has not set clear criteria of evaluation. Reason can be fatigue or stereotype repairing. Students usually get different exercises. Then argue "Colleague had simpler exercise, that I would program it." "Why ours group had more difficult exercise?" etc.

Other important factor in evaluation of programming exercises is approach of learning of programming. Different criteria of evaluation has to be set in object approach resp. structured approach of learning of programming

Stepan Hubalovsky is assoc. prof. and supervisor of Ondrej Korinek. He works at University of Hradec Kralove, Department of informatics, Faculty of Science, Hradec Kralove 500 38, Rokitanskeho 62, Czech republic, stepan.hubalovsky@uhk.cz.

Ondrej Korinek is Ph.D. student at University of Hradec Kralove, Department of informatics, Faculty of Science, Hradec Kralove 500 38, Rokitanskeho 62, Czech republic, ondrej.korinek@uhk.cz.

II. Possibilities of Evaluation of Students' Programmer Knowledge

Possibilities for evaluation of students' programmer knowledge is much. Below are mentioned most frequent possibilities with their analysis. Among most frequent possibilities belongs:

- *programming of exercises in any programming language;*
- *writing of program on paper;*
- *theoretic tests with selection of several possibilities;*
- *complementary exercises.*

The last two possibilities can be evaluated not only by teacher. They can by evaluated by systems for automatic evaluation.

A. Programming the exercises in programming language

Most frequent are exercises of type: Create program in given programming language. Teacher has to verify functionality of the program. Other important part of evaluation is check the way of programming. Program that work properly, cannot be programmed optimally based on the programming rules. Teacher has to check the program code, which spend a lot of times. Moreover this type of evaluation is not objective. On the other hand student has good feedback.

B. Programming of exercise on paper

To other type check problem are exercise programmed on paper.

This way of examination of knowledge of students was used by authors of research of two different methodics of learning of programming. Programming language was Java [1]. This way of evaluation doesn't require debug the program in IDE. The teacher spend more time by checking of the programming code. In this kind checking is error liability high. Impossibility of debugging the written code is for student demotivating.

C. Tests with selection of several possibilities

Test with selection correct answer from several possibilities doesn't have predicative appreciate. It depends on type of questions. Selection one of a number of possible answer has disadvantage. Student can accidentally guess correct answer. Taking off the points for wrong answer can be for student demotivating, but it can eliminate randomly guess of correct answer. Selection of correct answer can be also done by exclusion of others possibility of. The teacher doesn't know

how the student is thinking.

Authors of research [2] examined two various methodics of learning of programming. Learning of programming was realized by the two approaches *Objects-First* and *Objects-Later*. Students in test supplement the answer or select from several possibilities. Each methodic of learning of programming has approximately 50% questions [2].

D. Complementary exercises

Complementary exercises have higher predictive value than selective exercises. It depends on type of completion. Students can complete:

- parts of definitions;
- value of variables after program running;
- parts of code.

Concept of variable occur in different paradigms and approaches of programming. Passing and writing the values of variable in specified of part of program checkup the students understanding of program construction.

Complete of parts of code has the highest predictive value. Students have to understand the principle of the problem. Only after understanding can complete missing parts of code. This type evaluation has several disadvantages:

Impossibility of debugging of the program may cause wrong syntax of the commands;

Not only one solution is correct;

Setting of given part of the code lead student to one given solution;

Presented solving can limit student during testing.

E. Evaluation by method Design Patterns First

There is presented in paper [3] possibility of evaluation of students' programming exercises. Further there is presented learning of programming by method Design Patterns First in the paper. The student solution of the task can be evaluated automatically, „*…if the students will have to implement some interface.*" [3].

III. Automatic evaluation

There are several systems for automatic evaluation of testing exercises. Universal testing environment used for testing of students at Faculty of Science, University of Hradec Kralove is described in the following.

„*Universal Testing Environment (UTE) is an electronic online testing system designed for the creation, operation and administration of the tests*"[4].

Creation of Universal selection is described in detail in [5]. Web sites of the universal testing environment can be found testing environment, used programming language, introduction of original testing progress – e.g. new questions in [6].

The universal testing environment is large number of testing questions. Following type of the questions can be created by the template:

- Completion of text;
- Category – sorting items to given category;
- Multiple- choice – withdrawal one or more answers from wider possible selections;
- Assignment of couples – assignment of correct answer to single label;
- Assembling words – make - up words from single parts;
- Sentence – make - up words into sentences – correct sequence of commands.

It is possible to creation and another own template.

Advantage of the testing environment (compared with paper testing) is instantaneous evaluation. Students can immediately after finishing the testing look through analysis of the test. Individual questions have scales. Comparison of similarities with answer can be also set in the evaluation. The testing environment doesn't require for correct answer accurate values – approximate values are valid [6].

The advantage of this type of testing is objectivity of the testing. Testing is not time consuming.

IV. Programming Faculty of Science, University of Hradec Kralove

Subjects concerning programing are learned in Faculty of Science, University of Hradec Kralove from first semester. The subject *Algorithms and data structures* (ALGDS) is the start subject at the first semester. The learning of ALGDS proceeds in pseudo-code of language Pascal. The goal of the subject is developing of algorithmic thinking of the students. They have to handle the create algorithm based on simple and structured variables and matrixes [7].

The subject is finished by passing credit and examination test [8].

A. Algorithmic thinking

Algorithmic thinking is important for correct algorithm design that fulfil all features of algorithm and is based on fundamental algorithm terms. Algorithms occur not only in programming, but also in daily common activities. Algorithmic thinking is developed by experiences from common life.

B. Programming 1

Subject of Programming 1 follows the subject ALGDS. Programming language is C# or Visual Basic. The student were divided into two homogenous group n subject programming 1 in academic year 2013/2014. The first group was learned according to *Objects-First* approach to programming. The second group was learned according to *Objects-Later* approach in programming. The first group emphasized to object construction (algorithmic construction was learned less). The second group emphasized algorithmic construction (object construction was learned less).

V. Research of approaches in Programming

Twenty eight students attend subject Programming 1 in the academic year 2013/2014 1. Students were divided into two homogenous group according to results of examination test from subject ALGDS. There were to 14 students in both

groups. The approaches in programming differ by scope of the topics. To success in the subject the have to gain 60% from two parts of testing project. Test consists of practical part and (programming) and theoretical part (10 questions).

Theoretical part of test includes 10 questions of universal testing environment. Test is divided into three parts:

- completion code exercises;
- questions of object approach and structured approach of programming (classification of items to categories, selection of one's or more answers from, assignment one or more correct answers to labels and making up the words into sentences.
- completion of correct values of variables by debugging the program code.

A. *Completion code*

Two questions are selected by universal testing environment. Students have to complete programming code of given method. Two questions are selected from list of exercises. All exercise have same weight. The exercises consist of sequence tasks and matrix tasks. Sample of exercise is shown in Figure 1. This set of questions is focused to basic terms of structured programming. Students have to think algorithmically during the test.

Task assignment: Complete code of method so that, the maximum value is find in each column of the matrix. The maximums are saved to new sequence.

```
public int[] Extremy(int[,] matice)
{
     [      ] p = new int[matice.GetLength([ ])];
 for(int j = 0; j < matice.GetLength([ ]); j++)
 {
 int ex = [          ];
 for(int i = 1; i < matice.GetLength([ ]); i++)
 if (matice[[      ]] [ ] ex)
                [ ] = [          ];
          [      ] = ex;
 }
 return [ ];
}
```

Fig 1 Sample of exercise for code completion

B. *Research of results of students' test – theoretical part*

Five students from experimental group and control group succeeded in theoretical part of the test.

To determine whether the median of the results of student in theoretical part of test is the same for the both groups of students was used the nonparametric Mann-Whitney test.

Calculated P-value is P = 0.57

with significance level $\alpha = 0.05$,

so we cannot reject the null hypothesis that the median of results of the students from theoretical part of the test is the same between control group of student and experimental group of student. Among groups there is not statistically significant difference.

Box plots chart of result of both groups of students is shown on Figure 2.

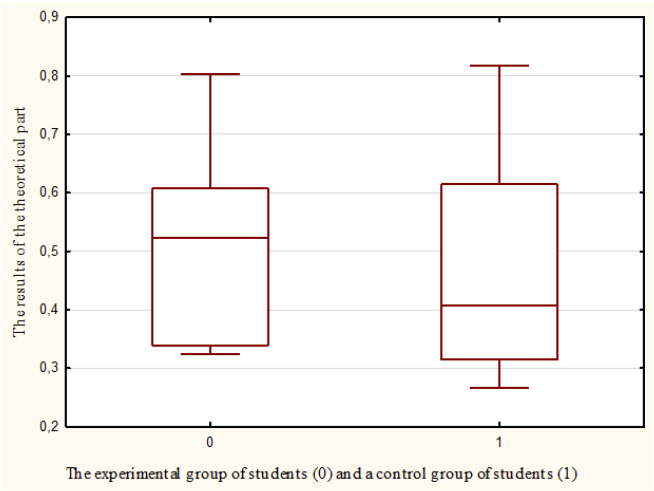

Fig 2 Box plot diagram of result of theoretical part of test project of control group of students and experimental group of students

Average result of theoretical part of test of experimental group of students is 52.6%. Average result of theoretical part of test of control group of students is 47.8%. Experimental group of students has a bit better results than control group of students.

C. *Research of results of students' test – code completion*

To determine whether the median of the results of student in code completion part of test is the same for the both groups of students was used the nonparametric Mann-Whitney test.

Calculated P-value is P = 0.301

with significance level $\alpha = 0.05$,

so we cannot reject the null hypothesis that the median of results of the students from code completion part of the test is the same between control group of student and experimental group of student. Among groups there is not statistically significant difference.

Box plots chart of result of both groups of students is shown on Figure 3.

Average result of code completion part of test of experimental group of students is 34.4%. Average result of theoretical part of test of control group of students is 26.3%. Experimental group of students has a bit better results than control group of students.

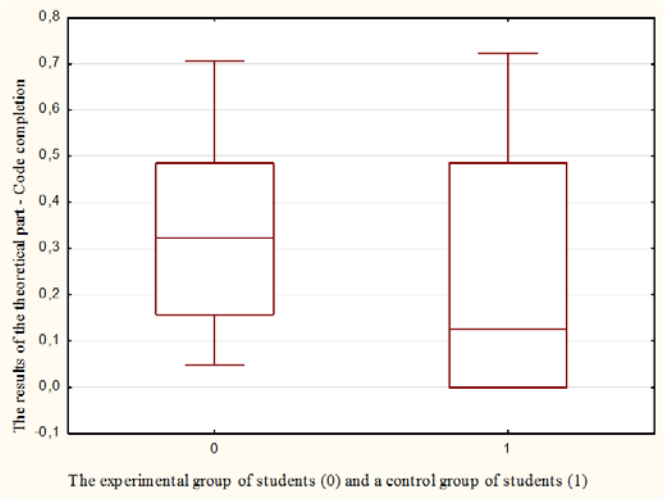

Fig 3 Box plot diagram of result of code completion part of test project of control group of students and experimental group of students

VI. Conclusion

The paper deals with possibilities of testing of student from programming. Each of possibility has benefits and disadvantages. It depends on teacher which method for evaluation of student knowledge is used. The paper presents computer universal testing environment and type of tasks that are used for testing of students of Faculty of Science, University of Hradec Kralove in subject Programming. The paper presents the research of results of bachelor degree students of Didactic of informatics in the academic year 2013/2014. The results of theoretical part of the test of students was investigated in two groups of student. The first group of student was taught by *Objects First* approach of learning of programming, the second group of student was taught by *Objects Later* approach of learning of programming. *Object-First* group of student had a bit better results of the theoretical part of the test.

Theoretic part of the test was divided into three groups of questions – e.g. completion code tasks. These questions are focused on algorithmic construction used in structured programming. Statistically significant difference between *Object-First* group of student and *Object-Later* group of student was not found. Results of research are a bit surprising. *Object-First* group of students had better results in algorithmic constructions test than *Object-Later* group of students. The results of the research are beneficial and enable improve universal testing environment.

Acknowledgment

This research has been supported by: Specific research project of University of Hradec Kralove, Faculty of Education in 2015 and Specific research project of University of Hradec Kralove, Faculty of Science in 2015.

References

[1] J. Sajaniemi and H. Chenglie, "Teaching Programming: Going beyond 'Objects First'" PPIG 2006 University of Sussex, Available: http://www.ppig.org/papers/18th-sajaniemi.pdf

[2] A. Ehlert and C. Schulte, "Empirical Comparison of Objects-First and Objects-Later" In Proceedings of the fifth international workshop on Computing education research workshop, ICER '09, New York, NY, USA: ACM, ISBN 978-1-60558-615-1, s. 15–26. Available: https://home.cc.gatech.edu/csed/uploads/2/ehlert2009.pdf

[3] R. Pecinovsky, "Metodika design patterns first a vyhodnocování studentských úloh" Tvorba Softwaru 2007 [online]. 2007 [cit. 2015-10-01], Available: http://vyuka.pecinovsky.cz/prispevky/2007-SW_Metodika_DPF_a_vyhodnocovani_uloh.pdf

[4] P. Voborník," Universal Testing Environment as an External Tool of Moodle" In: 10th International Scientific Conference on Distance Learning in Applied Informatics (DiVAI 2014), 5.-7. 5. 2014, Štúrovo, Slovakia, Prague: Wolters Kluwer, 2014, pp. 215-225. ISBN 978-80-7478-497-2.

[5] P. Voborník, "Univerzální testovací prostředí", Ph.D., University of Hradec Králové, [online] Available: http://download.petrvobornik.cz/docs/disertace.pdf [Accessed 1 February 2014].

[6] P. Voborník, "Univerzální testovací prostředí" Web, [online] Available: http://www.alltest.eu [Accessed 1 February 2014].

[7] E. Milková, "Algoritmy: základní konstrukce v příkladech a jejich vizualizace", Hradec Králové: Gaudeamus. ISBN 978-80-7435-064-1.

[8] E. Milková and O. Kořínek, "Students' Programming capabilities evaluation" Efficiency and Responsibility in Education (ERIE 2013), Efficiency and Responsibility in Education (ERIE 2013), Prague, pp. 434 - 440

Stepan Hubalovsky was born in Trutnov, Czech Republic in 1970, he obtained master degree in education of mathematics, physics and computer science in 1995, Ph.D. degree in theory of education in physics in 1998 both in Faculty of Mathematics and Physics, Charles University in Prague, Czech Republic and assoc. prof. degree in system engineering and informatics in 2012 in University oh Hradec Kralove, Czech Republic. He worked 5 years as master of mathematics, physics and computer science on several secondary schools. He works as associate professor on University of Hradec Kralove from 2006. He interested in algorithm development, programming, system approach, computer simulation and modelling. Assoc. prof. RNDr. Stepan Hubalovsky, Ph.D. is member of Union of Czech Mathematicians and Physicist.

Ondrej Korinek was born in 1985 in Horice, Czech Republic. He graduated from the University of Hradec Kralove, Czech Republic in 2010, where he studied Education Mathematics and Computer Science for Secondary Schools. Since 2012, he has continued his studies of Information and Communication Technology in Education at postgraduate level. Since 2010 he has been working as an ICT teacher at VOS and SPS in Jicin, Czech Republic. He is interested in algorithms, programming methodology, database systems and modern technology.