VILNIUS UNIVERSITY

Bronius Skūpas

A METHOD FOR SEMI-AUTOMATIC EVALUATION AND TESTING OF PROGRAMMING ASSIGNMENTS

Summary of Doctoral Dissertation

Technological Sciences, Informatics Engineering (07 T)

Vilnius, 2013

Doctoral dissertation was accomplished at Institute of Mathematics and Informatics of Vilnius University in the period from 2008 to 2012.

Scientific Supervisor

Prof. Dr. Valentina Dagienė (Vilnius University, Technological Sciences, Informatics Engineering – 07 T).

The dissertation will be defended at the Council of the Scientific Field of Informatics Engineering at the Institute of Mathematics and Informatics of Vilnius University:

Chairman

Prof. Dr. Albertas Čaplinskas (Vilnius University, Technological Sciences, Informatics Engineering – 07 T).

Members:

Prof. Dr. Habil. Juozas Augutis (Vytautas Magnus University, Physical Sciences, Mathematics – 01 P),

Prof. Dr. Eduardas Bareiša (Kaunas University of Technology, Technological Sciences, Informatics Engineering - 07 T),

Prof. Dr. Habil. Genadijus Kulvietis (Vilnius Gediminas Technical University, Technological Sciences, Informatics Engineering – 07 T),

Dr. Virginijus Marcinkevičius (Vilnius University, Technological Sciences, Informatics Engineering – 07 T).

Opponents:

Prof. Dr. Dalė Dzemydienė (Mykolas Romeris University, Technological Sciences, Informatics Engineering – 07 T),

Prof. Dr. Habil. Rimantas Šeinauskas (Kaunas University of Technology, Technological Sciences, Informatics Engineering – 07 T).

The dissertation will be defended at the public session of the Scientific Council of the Scientific Field of Informatics Engineering in the auditorium number 203 at the Institute of Mathematics and Informatics of Vilnius University, at 1 p.m. on the 11^{th} of February, 2013.

Address: Akademijos st. 4, LT-08663 Vilnius, Lithuania.

The summary of the doctoral dissertation was distributed on the 11^{th} of January 2013.

A copy of the doctoral dissertation is available for review at the Library of Vilnius University.

VILNIAUS UNIVERSITETAS

Bronius Skūpas

PUSIAU AUTOMATINIO PROGRAMAVIMO UŽDUOČIŲ VERTINIMO IR TESTAVIMO METODAS

Daktaro disertacijos santrauka

Technologijos mokslai, Informatikos inžinerija (07 T)

Vilnius, 2013

Distertacija rengta 2008–2012 m. Vilniaus universiteto Matematikos ir informatikos institute.

Mokslinė vadovė

prof. dr. Valentina Dagienė (Vilniaus universitetas, technologijos mokslai, informatikos inžinerija - 07 T).

Disertacija ginama Vilniaus universiteto Matematikos ir informatikos instituto Informatikos inžinerijos mokslo krypties taryboje:

Pirmininkas

prof. dr. Albertas Čaplinskas (Vilniaus universitetas, technologijos mokslai, informatikos inžinerija – 07 T).

Nariai:

prof. habil. dr. Juozas Augutis (Vytauto Didžiojo universitetas, fiziniai mokslai, matematika — 01 P),

prof. dr. Eduardas Bareiša (Kauno technologijos universitetas, technologijos mokslai, informatikos inžinerija – 07 T),

prof. habil. dr. Genadijus Kulvietis (Vilniaus Gedimino technikos universitetas, technologijos mokslai, informatikos inžinerija — 07 T),

dr. Virginijus Marcinkevičius (Vilniaus universitetas, technologijos mokslai, informatikos inžinerija — 07 T).

Oponentai:

prof. dr. Dalė Dzemydienė (Mykolo Romerio universitetas, technologijos mokslai, informatikos inžinerija – 07 T),

prof. habil. dr. Rimantas Šeinauskas (Kauno technologijos universitetas, technologijos mokslai, informatikos inžinerija — 07 T).

Disertacija bus ginama viešame Informatikos inžinerijos mokslo krypties tarybos posėdyje 2013 m. vasario 11 d. 13 val. Vilniaus universiteto Matematikos ir informatikos institute, 203 auditorijoje.

Adresas: Akademijos g. 4, LT-08663 Vilnius, Lietuva.

Disertacijos santrauka išsiuntinėta 2013 m. sausio 11 d.

Disertaciją galima peržiūrėti Vilniaus universiteto bibliotekoje.

Relevance

Software development is increasing rapidly, and the demand for information technology (IT) professionals is growing. Many future IT professionals are introduced to the field in the form of basics of programming at school.

Teaching programming is a difficult process because it is associated with creative thinking, strictly formalized tasks, and practical programming assignments. The students have to be trained to create not only running, but also qualitatively designed, reliable, properly functioning programs. Testing and evaluation of programs developed by the students requires a lot of teacher's efforts and time.

In this work automatic program testing is defined as dynamic testing, based on black-box testing with tests prepared in advance. In order to use automatic program testing, programming tasks often are specified in detail, i.e. the required data input and output formats are introduced. This allows evaluating by applying the fact-verification method. Accurately identified verifiable fact is defined as an evaluation criterion.

Automatic programming assignments testing can be used in the teaching process, during programming exams, professional programming knowledge tests during recruitment and programming competitions.

Many authors note that fully automated testing, based on static and dynamic analysis, can not be completely fair. Therefore common practice is to use semi-automated testing of programming assignments, which is a mixture of automatic testing and manual evaluation and provides greater flexibility in use of automated testing benefits.

This thesis focuses on the exploration of possibilities of using a semi-automatic testing system for programming assignments in schools, distance education, as well as competitions and exams. Common evaluation errors are analyzed as well. The research is focused on creating a new method of evaluation for programming assignments in order to achieve high quality evaluation in acceptable time and the justification of the evaluation to the user.

Research object

The research object of this work is automatic and semi-automatic testing systems for programming assignments, their architectures, operating methods and algorithms, and interaction with the users.

Aim and Objectives

The aim of the dissertational work is to develop a method for semi-automatic evaluation and testing of programming assignments that would improve system-evaluator interaction in order to increase the efficiency and the quality of the evaluation and to implement it into a prototype of semi-automatic evaluation system.

The objectives of the dissertation are the following:

1. To perform analysis of theoretical and experimental research in the area of automatic and semi-automatic testing systems for programming assignments.
2. To investigate architectures, user interfaces and functionalities of automatic and semi-automatic testing systems for programming assignments, and to classify the encountered problems of these systems from methodological and architectural points of view.
3. To propose a method for semi-automatic evaluation and testing of programming assignments to enable more efficient and higher quality evaluation than the other known methods.
4. To perform the constructive research of the proposed method, and to investigate and evaluate the new opportunities provided by this method.

Research Methods

Constructive research was chosen as the key method for the research. Kasanen et al. (1993) states six phases of the constructive approach: (1) finding a practically relevant problem with research potential; (2) obtaining general and comprehensive understanding of the topic; (3) innovating, i.e., constructing a solution idea; (4) demonstrating that the solution works; (5) showing the theoretical connections and the research contribution of the solution concept; (6) examining the scope of applicability of the solution.

In practice I performed the steps of research in other sequence and my process was both iterative and recursive. The main reason for this was that I started research seven years ago and I innovated some of solutions for several times. Exam rules and requirements have been changed as well.

I selected the problem of creating a high quality and efficient evaluation method for programming assignments. For phase (2) I performed analyzed publications and surveys and publicly available tools. Some of the tools were used for competitions, during classes. Other systems are online systems and they were analyzed by exploring their publicly available functionalities. Some of the tools were analyzed by the documentation, as the tools themselves were not publicly available. Data about automatic evaluation systems and tasks, suitable for such systems, was collected and summarized. Information retrieval, systematic literature review, analysis, benchmarking and generalization techniques were used in the research step. Software system analysis, classification and generalization were used in research of architectures used in evaluation systems.

The innovative idea generated in phase (3) was based on the possibility to include interactivity into the semi-automatic assessment of programming assignments. I made the assumption that the iterative process of fixing errors and black box testing provided results can help the evaluator to decide the level of the achievements of the student.

For phase (4) I developed a software system for the semi-automatic evaluation of programming assignments, which was used for the experimental maturity exam evaluation in 2006. Testing this innovative idea in the real world was successful. The system for semi-automatic evaluation of programming assignments was used in the evaluation of programming exam submissions since 2006. It was improved several times through the years and it is under constant development.

For phase (5) I used quantitative research methods and expert interviewing techniques to evaluate the experimental software and effectiveness of the method . Quantitative and qualitative analysis was performed on the exam and on the experimental evaluation results which helped to draw conclusions about the influence of the method on the evaluation quality.

In phase (6) I assumed, that the improved method for semi-automatic evaluation can be applied for other evaluation systems for programming assignments. I analyzed the required changes to include the new approach into the Learning Management System Moodle Edujudge plugin.

Scientific Novelty

1. I presented a new programming assignment evaluation method. Repetition of the evaluation process which is foreseen in the method allows precise diagnosis of errors in the program being evaluated.
2. The proposed new user interface solutions allow the evaluator to obtain more meaningful feedback messages from the automatic testing system.
3. The proposed method is more efficient than other known semi-automatic evaluation methods.

Defended Statements

1. The proposed method for semi-automated testing and evaluation of programming assignments allows more efficient evaluation than the manual evaluation method.
2. The programming assignment evaluation results generated by the evaluation systems based on the proposed method for semi-automated testing and evaluation for programming assignments demonstrate the same quality level as the one achieved by manual evaluation and outperforms that of automatic evaluation.

Structure of the Dissertation

The dissertation consists of the terms and abbreviations section, six chapters, list of references and two appendices. It is written in Lithuanian.

1. Introduction

The first chapter is introductory. It contains the research problem and its relevance, research objectives and tasks, research methods, scientific novelty and defended statements.

2. Overview of surveys, software and assignments in Automatic Evaluation Systems for Programming Assignments

In this chapter, the main directions of development of methods used in automatic and semi-automatic assessment for programming assignments are presented and described.

Teaching has long been studied in the field of Computer Aided Assessment (CAA) access. However, open questions, algorithms, and program evaluation often requires significantly more time than the test questions with implicit assessment.

In analysis of development of automatic assessment systems for programming assignments recent reviews and surveys made by Ala-Mutka (2005); Douce et al. (2005); Ihantola et al. (2010); Liang et al. (2009); Queirós and Leal (2012); Romli et al. (2010) were used.

Beginning of automatic evaluation for programming assignments was found in Hollingsworth (1960) paper. In summary, automated assessment of programming assignments has been practiced since programming has been taught.

There are two main areas of use for automatic evaluation systems for programming assignments: curricular and competitive. However my analysis of requirements for systems have found that exam requirements are somehow different from typical curricular system.

Douce et al. (2005) identified three generations of automatic evaluation systems for programming assignments. Research of automated assessment systems is constantly growing and it is possible to find relation between assessment systems generations and objects of research.

First generation of systems activated researches in following area: possibilities and concepts of automatic evaluation; a high-level programming language support in systems; possible options for system expansion.

When second generation of systems were presented researches in following areas were initiated: multi-language support in automatic evaluation systems; semi-automatic testing and grading support systems; feedback improvement for users; content (tasks, tests, student) management; static analysis, programming style analysis; use of systems in programming competitions; an independent use of the system by student.

Third-generation systems opened up new areas of research: web interface possibilities for automatic evaluation systems; cloud computing and remote servers use for testing; the ability to use multiple programming languages for the same task; diversity of feedback for the user; plagiarism detection; various

ideas for assessment improvement; management of taught courses and integration into learning management environments; automatic test generation; interoperability for automatic testing systems.

Most of these areas of research are still actively discussed in articles.

Last researches mainly explore the systems integration to learning management systems, automatic test generation and system interoperability problems.

Approaches used in Automatic Evaluation

Ala-Mutka (2005) analyses set of automated evaluation systems for programming assignments and proposes classification model. Distinction is made between two methods of program analysis - static and dynamic.

Dynamic analysis allows analysing execution results, discovering errorous constructs. Evaluation can use following:

- functionality analysis;
- performance analysis;
- student ability to test;
- other features (use of programming language constructs, data structures, etc.).

Static analysis is used for:

- coding style assessment;
- syntax and some semantic errors;
- calculation of some metrics for source code (source code comment rate, etc.)
- program design and interface assessment;
- use of datastructures;
- keyword search;
- program structure analysis;
- plagiarism detection.

Systems are divided into fully automatic and semi-automatic. Fundamental difference between them is in evaluation algorithm as both of them use automated testing. In semi-automated system final score depends on human evaluator who uses automatic testing results and makes final decisions on evaluation of individual components in the final assessment.

There are also solutions where the applications are collected in the assessment of other students. Thus attract the number of reviews quickly handle large volumes students work in universities.

The automated testing of programming assignments usually is based on black box approach. Evaluation is based on analysis of results provided by tested program. Correct results are aggregated to some score by aggregation function. Most popular aggregation functions include *partial scoring*, *all-or-nothing* and *partial all-or-nothing*.

Black box testing is one of the most popular, but problems are known for long time. For example E. Dijkstra (1972) criticized it claiming that "program

testing can be a very effective way to show the presence of bugs, but it is hopelessly inadequate for showing their absence".

The main criticism for black-box testing can be visually shown in *Ishikawa* type diagram of problems and their causes (Fig. 1).

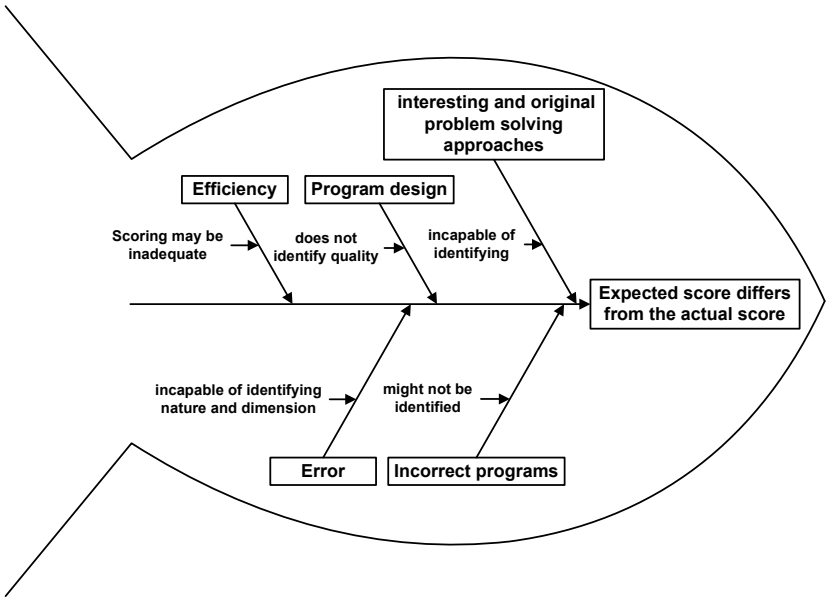

Figure 1: *Ishikawa* type diagram of problems and their causes visualizes black-box testing problems

Analysis of popular automatic evaluation systems

Author carried out analysis of several popular automatic evaluation systems. Information was collected about the systems in the literature, system websites, and publicly available documentation. Some of systems were analysed in action. Others were accessible only to commercial users and data gathered from articles.

Studied systems University of Valladolid Programming Training System (UVA Online Judge), EduJudge system, U.S.A. Computing Olympiad Training Site (USACO training program), Warwick University student evaluation system (BOSS Online Submission System), University of Nottingham student evaluation systems (Ceilidth, CourseMarker), International Informatics Olympiad'2002 competition system (IOI'2002 Contest and Grading System), Lithuania Olympiad in Informatics competition system, prototype of Lithuania maturity exam system.

Analysis showed that most systems have their own specific task description format. This causes problems to migrate tasks from system to another. Although the IMS Global Learning Consortium has proposed questions and tests XML storage scheme for IMS QTI, but at moment it is not acceptable for programming tasks. Learning object model of the IEEE LOM as the basis for programming tasks should be appropriate, but programming task requires

extra data (test data, results, test program, time constraints, etc.), which is not supported at moment.

Another common feature of systems - the majority of systems use black-box testing dynamic analysis methodology. They program performance measured by whether presented data sets the program gives the correct answers. So, when designing tasks, substantial time should be devoted to evaluation of the scheme selection, selection of good test data.

A large part of the systems does not provide localization capabilities.

The system value is raised by semi-automatic evaluation availability. Both dynamic and static analysis currently are incapable to evaluate programs in all aspects. This can be needed to evaluate not functioning programs. Therefore, opportunity for teachers to intervene in the assessment of student work is very important. Interaction of automated testing with manual evaluation was not under research in studied systems and articles.

Assignments suitable for automatic evaluation

Programming assignments were classified in several ways: according to data input and output methods that are suitable for automatic evaluation and according to the algorithms required to solve them.

Assignment classification based on input and output method:

- data read and the output written from/to standard I/O file;
- data read and the output written from/to specified files;
- I/O carried out through specified library;
- only output of the program is submitted.

Assignment classification by algorithms

UVA OnlineJudge automatic evaluation system and the database of problems were used for possible analysis of algorithms expected to be implemented in the solutions. The main source for problem solving algorithms was `http://www.algorithmist.com/`. Research was made on 656 tasks. The analysis result is presented in fig. 2.

3. Development of semi-automatic evaluation method

When designing Maturity exam of Information Technology in Lithuania (IT VBE – *lith. Informacinių technologijų valstybinis brandos egzaminas*) it was decided that the submitted programs designed by graduating high-school students should be evaluated if they do not compile. It was also agreed that the evaluation should be positive, i.e. the points should only be given for the skills demonstrated by the student. From the exam requirements it was clear that evaluation should be either manual or semi-automated. Manual evaluation was excluded due to limited number of evaluators and the limited time resources.

When considering the possibilities of applying semi-automated evaluation for evaluation of IT VBE submissions, it was raised hypothesis that deeper

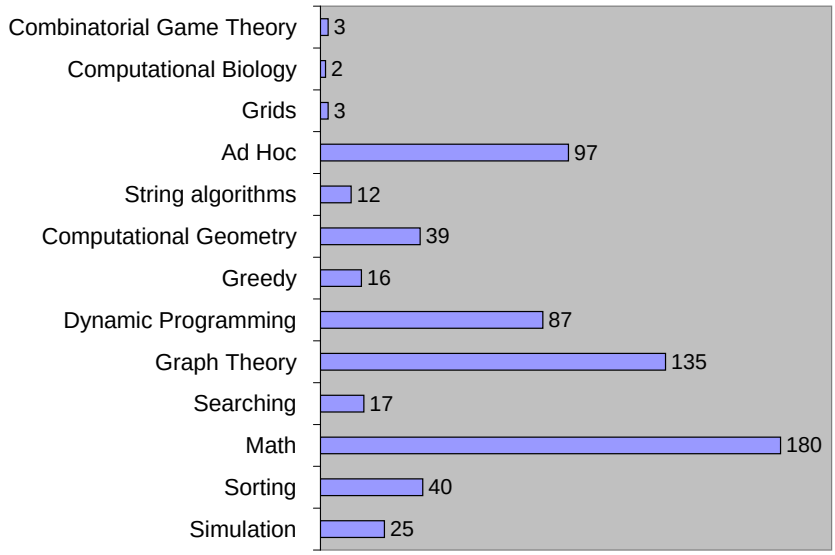

Figure 2: Task classification by algorithms

analysis of the process of manual evaluation of programs designed by the students would enable to create a new semi-automated testing and evaluation method which would have the same quality and precision as manual evaluation but higher efficiency.

Long observation of IT teachers evaluating programs manually as well as observation of the students searching for mistakes in their programs served as a premise for stating the hypothesis above. The process of designing a short program (i.e. not longer than 100 lines) is similar to that of designing complicated software systems – program modification alternates with program testing with some specific data.

I attempted to benefit from the experience of evaluation in Lithuanian Olympiads in Informatics. However it turned out that the demands of the exam are different from that of the Olympiads:

1. The evaluation system should operate in the environment as similar as possible to that used during the exams by the students (Windows OS, localized FreePascal compiler).
2. The task cannot be considered as partially solved if the program passes just some of the grading tests. The tasks are rather simple and the reasons for each failure should be identified.
3. The program with minor mistakes (missing punctuation mark, undeclared variable, etc.) cannot be assigned few or no points because of those mistakes only.
4. There is no need to support the role of the student in the exam evaluation system, because the student cannot observe his/her score directly during the exam.

There were defined the following most important *functional requirements* for the IT VBE evaluation system:

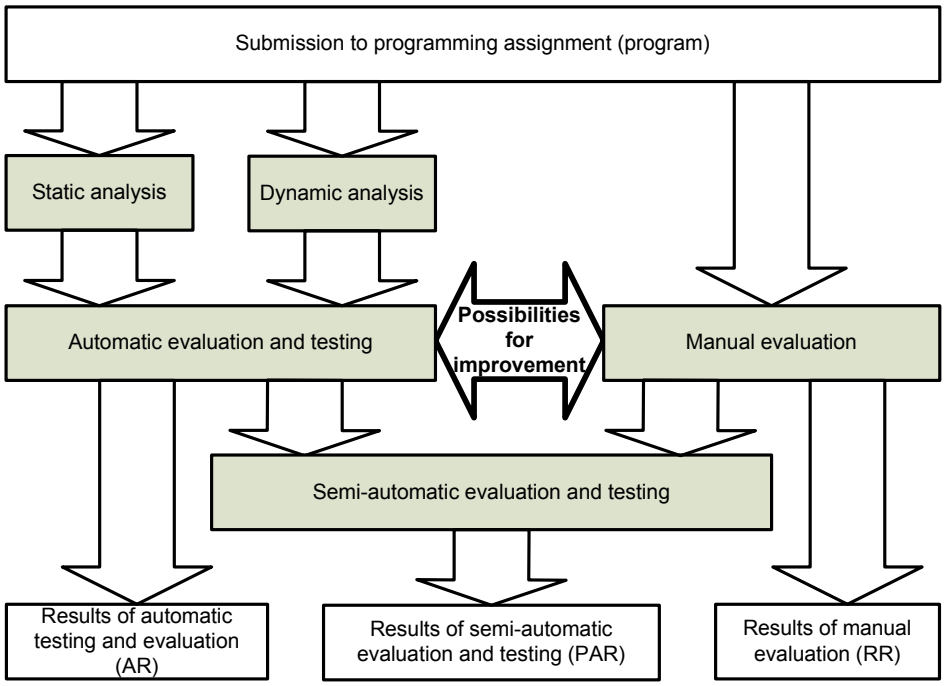

Figure 3: Scheme illustrating suggestions how to improve the evaluation process

During the evaluation semi-automated evaluation system should provide the administrator functionality to:

- load task specification together with tests into the system;
- modify task evaluation criteria;
- to compose, assign and present to the evaluator an evaluation package containing a set of submitted programs to be evaluated;
- to assign an evaluation package to another evaluator;
- to observe and record the outcome of evaluation process.

Semi-automated evaluation system should provide the following functionalities for evaluating a submitted program:

- to be able to execute statistical analysis plug-ins and provide the results to the evaluator;
- to automatically test the submitted program using black-box testing;
- to present the results of automated testing and the program being evaluated to the evaluator;
- allow to modify the program being evaluated and to retest the modified program without changing scores assigned by automated testing; to display the successfully passed tests, to visualize the changes made by the evaluator and to allow to restore the initial version of the submitted program;

- to allow to fill/modify the scores obtained from semi-automated testing if automated testing did not assign any points to the program;
- to allow to fill/modify the manual evaluation scores.

There were also added *non-functional requirements*. Those will not be emphasized in the formalized problem statement in this thesis, because their implementation is not so abstract. Here are the most important ones:

Semi-automated evaluation system should allow:

- to evaluate incomplete programs in a positive manner;
- to help finding the exact location of an error;
- to provide clear feedback messages to the user-evaluator;
- to operate fast;
- to allow the evaluator to comfortably experiment with the program being evaluated.

The requirements of the system to exam specification

Each **assignment** given in the exam should satisfy the following requirements:

- problem formulation should contain set of values (domains) of input and output data.
- problem formulation should contain clear definition of data input/output flows as well as data input/output format requirements.
- sample solutions should be known beforehand and the computer working time of sample program solving the task is acceptable with all possible input data sets.
- there should be prepared tests (input/output data sets) which can illustrate the correctness of the solution in various cases.
- it should be known the way how to verify the correctness of output of the solution.
- manual grading instruction should be prepared.

When preparing for automated evaluation **test data sets** have to be prepared as well as the checker (i.e. the program verifying the correctness of output provided by the program being evaluated to a test data set) has to designed and implemented.

Test data sets should satisfy the following:

- short and understandable to the evaluators;
- to identify possible incorrect solutions;
- to demonstrate the functionality of the program in various situations.

The criteria for **alternative and semi-automated evaluation** should be prepared. It is highly important to prepare and specify the grading scheme. However preparing the criteria for alternative grading is not an easy task, because:

The Criteria have to be clear and unambiguous to the evaluators, precisely corresponding to possible task solutions, assigning points for the skills demonstrated by the student;

The Points for the criteria should correspond to the exam matrix.

Due to the reasons above the pilot evaluation is often arranged in order to specify the evaluation schemes before starting the real evaluation session.

The semi-automatic evaluation method improvements implemented in the IT VBE Evaluation System

After formalizing the problem and emphasizing the role of the evaluator (i.e. dissociating from many roles common in various grading systems) there was created the prototype of semi-automated evaluation system which implemented the grading of one program. In the system there was implemented safe black-box testing, reviewing the program under evaluation and collecting scores from manual grading.

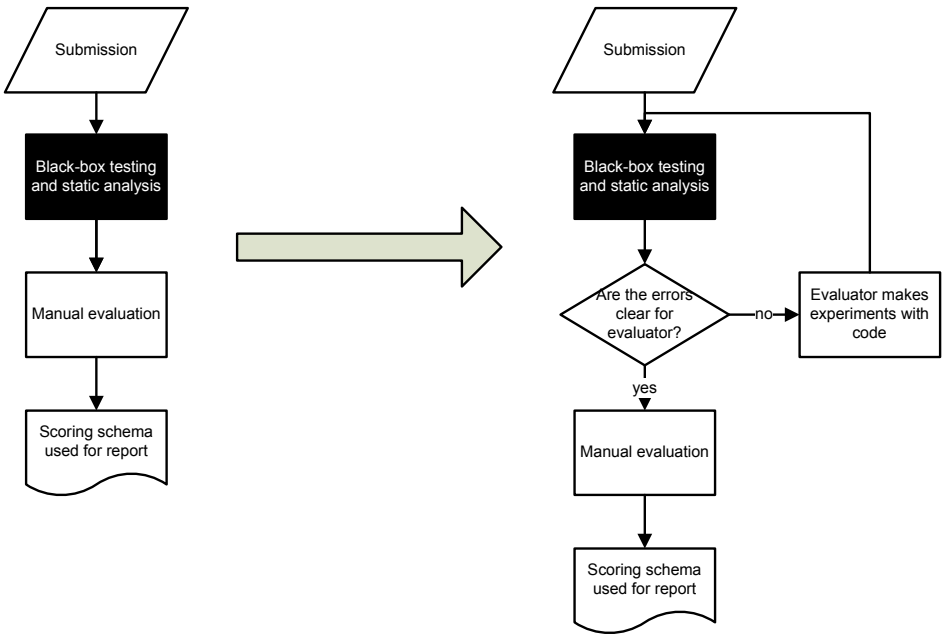

Figure 4: Semi-automated program testing process improvement

After some experiments with the evaluation system being developed it was observed that for the evaluators it is not easy to find a mistake in the program. In such case the evaluators used to open the program in some widely accepted

IDE and started experimenting there. However most IDE's do not provide functionalities for testing the program with group of tests. This observation suggested the idea to transfer some IDE functionalities to the evaluation system. The first and the essential improvement was the following: program text review component was replaced by program text editing component and there was added a button allowing batch retesting of the modified program. This improvement is illustrated in fig. 4.

The evaluators appreciated the improvement however new problems occurred: the evaluators couldn't recall all the modifications they made to the program. Testing results had been changed as well. Therefore it was decided to implement into the system the functionality which would allow to store the original submission and to log program text modifications made by the evaluator. Two tile windows containing the original submitted program and the modified program (fig. 5) were added as functionality to the evaluation system.

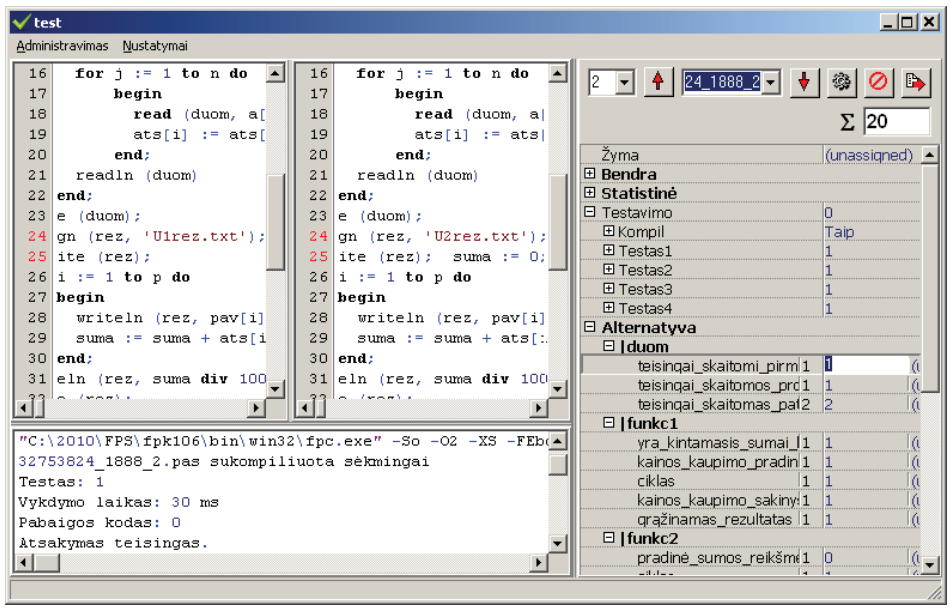

Figure 5: The screen-shot of IT VBE evaluation system displaying original and modified program texts

It was decided to store separately the results of automated testing of the original program. Thus the scores of the experimental (i.e. modified programs) testing is separated from the grading process. Those scores provide information about the impact of modifications to program functionality; however the weight of the modifications as well as impact to semi-automated evaluation is decided by human evaluator.

Fig. 6 contains conceptual class diagram which clearly separates testing scores of the originally submitted program and the modified program.

When improving the prototype iteratively it turned out that the evaluators need more clear feedback about why the program output is not correct. Typical checkers (output correctness verification programs) did not provide

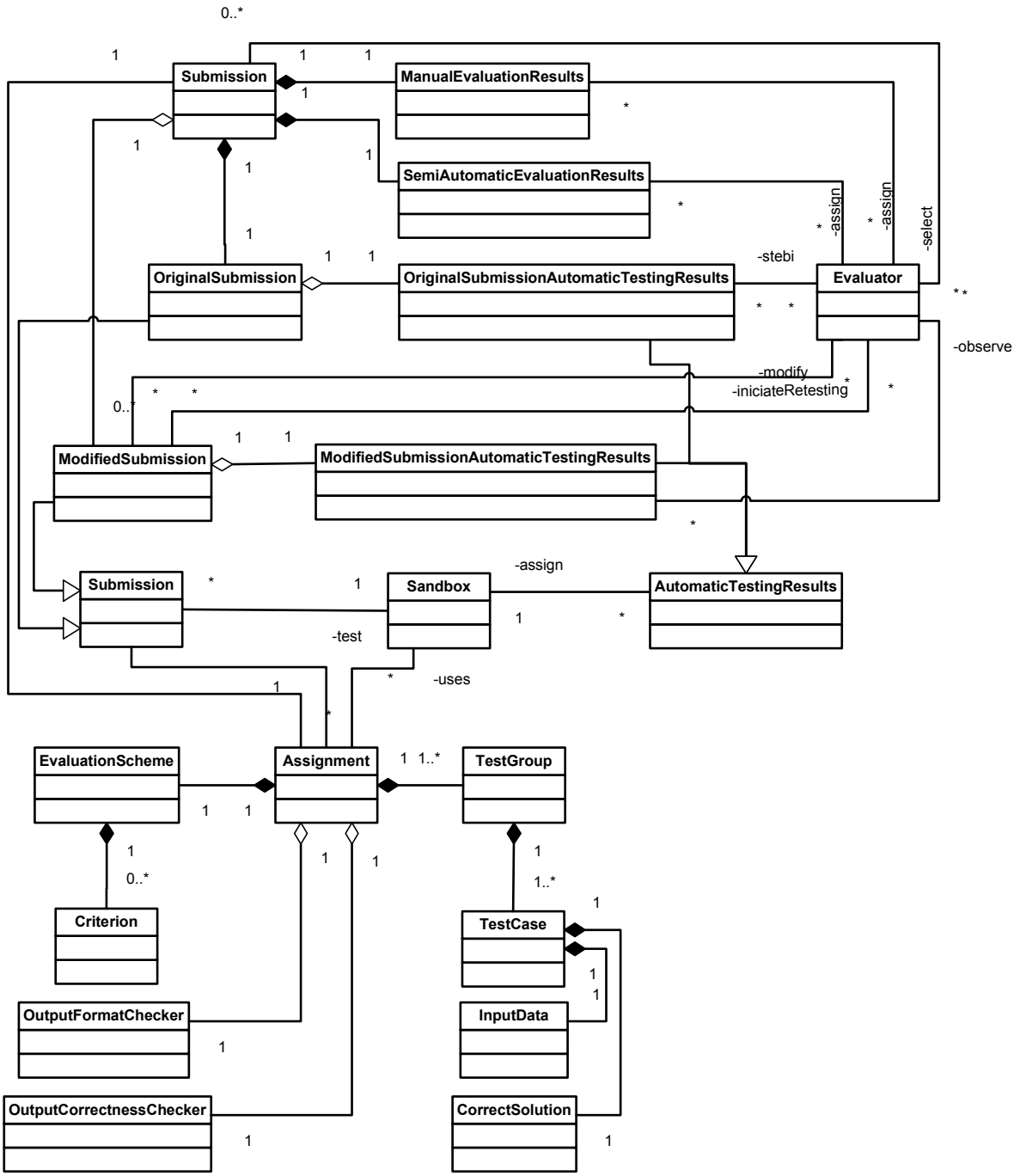

Figure 6: Class diagram of the evaluation system prototype

the efficient feedback: they only identified the output file line which differs from unique correct answer line. It was concluded that in order to provide better diagnose the correspondence of the output format to the required output format described in the problem formulation. Lexical analyzer is a good solution to that problem. However development of lexical analyzer for each task is time consuming. Therefore it was decided to decompose checker by forwarding the evaluation to two other programs: the program verifying output format correctness and the simplified program verifying output correctness (fig. 7)

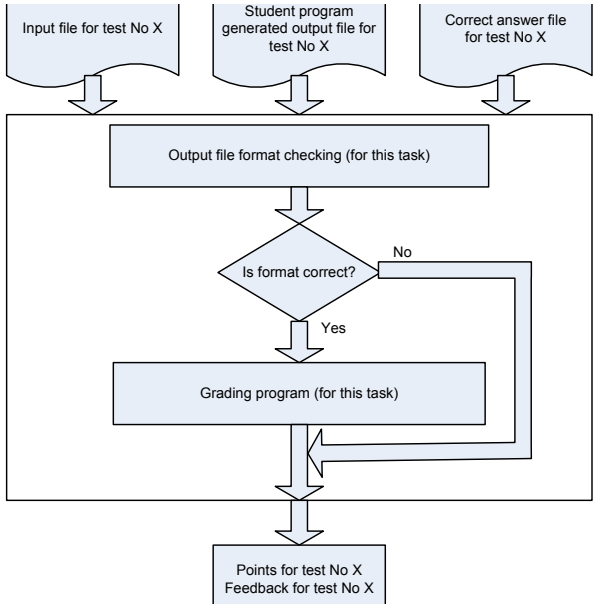

Figure 7: The improved black-box testing method component: the program evaluating the correctness of program forwards the evaluation to two other programs

Thus instead of creating a lexical analyzer for each task there was created one lexical analyzer and output formatting requirements were given to the analyzer in a form of configuration data. The output correctness verification program was simplified because in case of unique solutions for most assignments it was enough to compare files ignoring repeated whitespaces.

The evaluation system was complemented with static analysis plug-ins and some of those performed automated evaluation of programming style. After several evaluation system improvement iterations its component scheme is as illustrated in fig. 8.

The Development of IT VBE Practical Task Evaluation System

IT VBE syllabus allowed to use Pascal as well as C++ programming languages during the maturity exam. The flexibility allowing the students to choose between two programming languages stimulated adaptation of the evaluation system to the new situation.

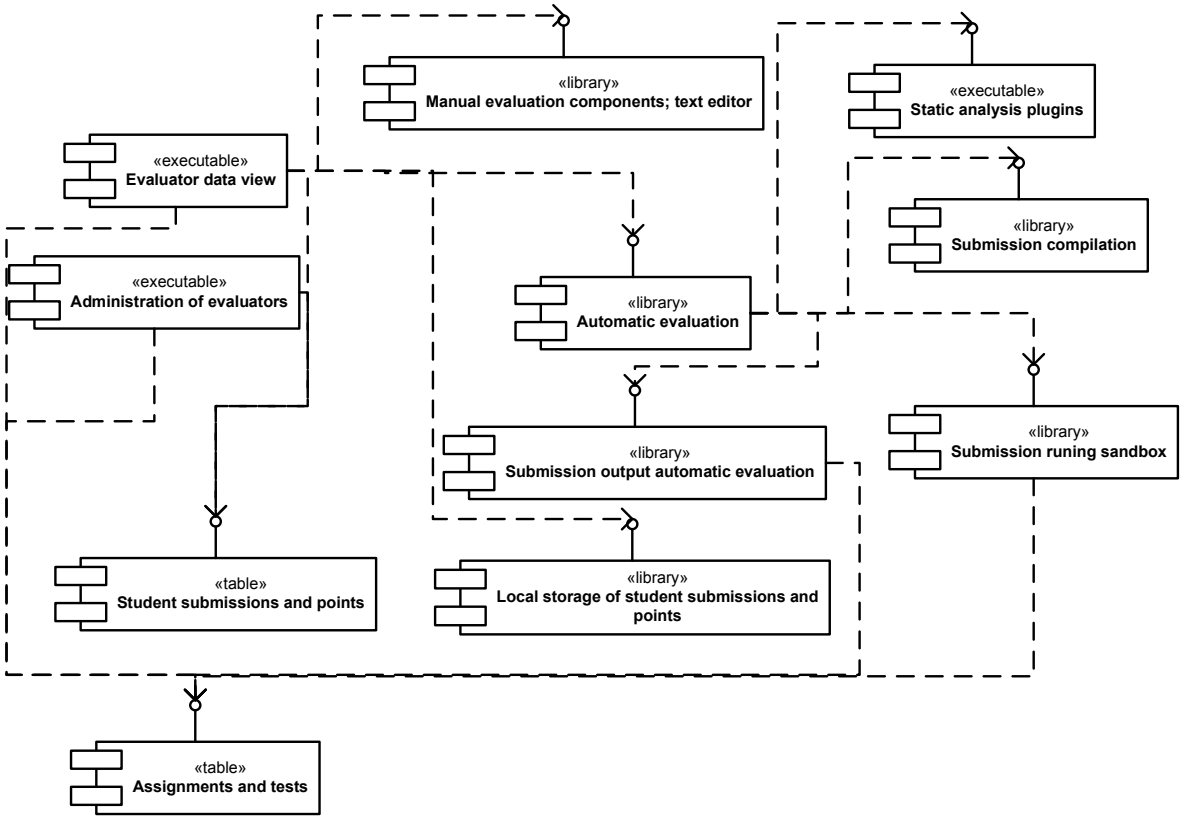

Figure 8: Component scheme of semi automated testing system

There were developed many systems for automated testing of students' programs. Often they are multilingual, i.e. they can be used for testing programs written in different languages. However this is common for the systems that use dynamic analysis and in particular black-box testing. Adaptation of other systems is a complicated process as it requires extensive analysis of each newly added language, it also requires complicated programming, and therefore such adaptation is rarely performed. The adaptability of semi-automated evaluation systems to new programming languages has not been investigated. Multilingualism is common in programming contests, however rear in programming exams.

I conducted an empirical research of theoretical possibilities to adapt current IT VBE semi-automated practical task evaluation system to new programming languages. The analysis and modification of IT VBE was performed. The programming language choices of some winners of Lithuanian informatics Olympiads were investigated; the survey of some selected students was performed and analyzed. There were assessed the threats of multilingualism in IT VBE to the fairness of evaluation. Methodology for diagnosing and decreasing such threats was prepared as well.

After analyzing IT VBE program evaluation system it was concluded that the system is closely related to Pascal programming language: it does not allocate task based on programming language, program text display and mod-

ification component uses Pascal syntax highlighting, it is closely related to FreePascal compiler, programming style analysis is performed using special static analysis plug-in which is not suitable for other programming languages.

After preliminary analysis of IT VBE evaluation system, there was performed a comprehensive analysis of source code of all the system as well of its all modules: The following modules were identified as strongly connected with Pascal programming language:

- storage of student programs module;
- user interface functionality that enables display and review of the original and the modified programs;
- student program compilation module;
- student program execution module;
- student program static analysis plug-ins.

When modifying the evaluation system, I made efforts not change the user interface significantly and to maintain the system functionality. The majority of changes were made not in the commonly used user interface parts.

Static analysis plug-ins caused major difficulties when adapting the system to the new programming language. Some plug-ins were disabled due to limited possibilities to use them with several languages. An example of such disabled plug-in is programming style evaluation plug-ins. One of the reasons for this decision was that there are many commonly accepted C++ coding style and program text layout conventions and it is hard to find common rules except for being consistent. On the other hand Codeblocks IDE which is used during the exam provides automatic source code layout.

4. Experimental part

For the experimental study of the method, IT VBE practical task evaluation software was used. The software has been used for the evaluation of maturity exam for six years. Each year around 2,400 programs developed during graduation exam by high school graduates are evaluated using the software with the implemented method. All the programs submitted for evaluation are graded twice. In case of significant (3 or more points) difference in marking between different evaluators, the submission is re-evaluated by the third evaluator. The latest mark is accepted as final.

The team of evaluators consist of about 30 people. The evaluation session lasts one week (five working days). Evaluators are selected from IT teachers working at middle schools and high schools as well as university teachers. The team of evaluators is rather stable: annually change up to three evaluators.

During the evaluation, the evaluators get the packages with student submissions each package containing 10 submissions. The packages are distributed randomly however the same evaluator cannot get the same package for a second time.

Experimental Research of the First Defended Statement

Defended Statement No1. The evaluation of submissions (programs) to IT VBE practical assignments is more effective (i.e. faster) by applying the improved semi-automated evaluation method if compared to that using manual evaluation.

Quantitative statistical research was chosen to test the hypothesis. It was tested the speed of evaluation by evaluating 50 programs designed by high school students both in the proposed semi-automated evaluation and manually. The evaluators answered the questions about the time spent for evaluation.

It has been experimentally investigated the average time it takes for the evaluator to evaluate a package containing submissions of ten students using IT VBE practical task evaluation software and the average time it takes if the evaluator is provided only a standard IDE.

The choice of the evaluators and measurement of their working time was seriously considered, because:

- the efficiency of different evaluators is different. The amount of evaluated programs during one week evaluation session differs twice between the fastest and the slowest evaluators.
- the efficiency of the evaluators rises during the evaluation week. This is due to the improving evaluation software usage skills and due to the adaptation to evaluate the programs solving particular task.
- there is possibility to conduct the experiment of manual evaluation, because the work is very intense, the time for the evaluation of maturity exam submissions is strictly limited and all the submissions have to be evaluated during the evaluation session.

To the experiment there were invited five average speed evaluators. The difference in the amount of evaluated programs during the evaluation session did not exceed 70. The working speed of the evaluators that participated in the research is presented in fig. 9: the average amount of the evaluated programs equals 190.

The evaluators participating in the experiment were asked to conduct experimental manual evaluation one month after the evaluation session was over. This ensured that the skills to evaluate concrete task gained during the IT VBE exam evaluation would be lost to some extent. The survey of the evaluators conducted right before the experiment revealed that they little remembered evaluation criteria and instructions as well as typical mistakes encountered in the submitted programs.

The five packages to be evaluated during the experiment were chosen from the IT VBE practical task submission packages assigned for evaluation on the morning of the second day of the evaluation week. The motivation for this was that during the first day much time is spend to improving evaluation instructions, trial evaluation. Regular evaluation starts on the second day.

The second criterion for choosing the packages was assigned grades. The

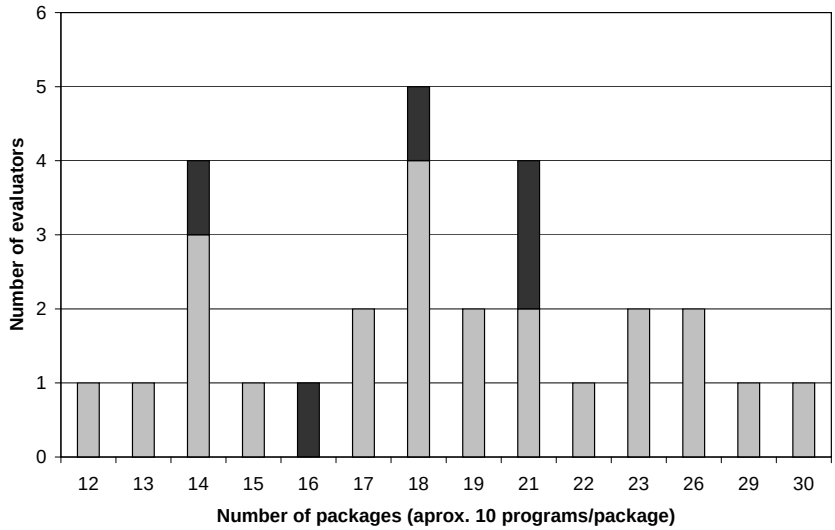

Figure 9: Histogram of the working speed of the evaluators in 2012 during IT VBE evaluation session. The evaluators invited to the experiment are shaded black.

submissions that are evaluated fastest are those which pass all the tests successfully or those alike to "Hello world" program (i.e. containing no solution).

The third criterion for the choice of the package was the requirement that at least one of the evaluators participating in the research had evaluated it during the exam session. This (i.e. the same packages were given to the same evaluators for re-evaluation) was done deliberately to ensure that the packages contain submissions requiring the same amount of efforts to evaluate them. The evaluators stated that they did not recall that they had already graded those packages.

After comparing the required evaluation time of each of the five evaluators (table: 1) if can be stated that manual evaluations of an evaluation package requires 43 to 117% more working time. The decrease of the evaluation time using the proposed semi-automated evaluation method and the corresponding software is significant and confirms the first hypothesis.

Experimental Research of the Second Defended Statement

The quality of evaluation of IT VBE practical tasks using the proposed semi-automated testing method is not worse than or the same as that using manual or fully automated evaluation.

The defended statement was investigated using qualitative research, comparative analysis and surveys. Several different research methods were chosen because it was required to compare the quality of evaluation using very different evaluation methods.

The author of the thesis claims that the evaluation performed by a high quality expert can be considered as benchmark. Many evaluation criteria applied when evaluating submitted programs cannot be implemented automati-

Evaluator	No. of programs evaluated in eval. session	Semi-automated evaluation, hours	Manual evaluation, hours.	Time ratio, manual/semi-aut.
A	140	01:38	02:20	1.43
B	210	01:15	01:50	1.47
C	180	01:20	02:35	1.94
D	210	00:50	02:00	2.40
E	160	00:53	01:55	2.17
Total		05:56	10:40	1.80

Table 1: Comparative data of time required for evaluation when evaluating a package of ten programs in the semi-automated and the manual ways

cally. Moreover modern programming languages are improved to be convenient for the user, therefore only the user can fully understand the peculiarities of the program designed by another user.

A comparative analysis of final grades of 50 evaluated programs was performed. One part of grades was obtained by manual evaluation during the above described experimental research. The other part of grades was obtained by applying semi-automated evaluation during the evaluation session of IT VBE. Responses of the evaluators to the survey were also included into the research. IT VBE evaluators indicated in the open questions of the survey that using IT VBE semi-automated evaluation system allows them "to perform evaluation faster". The respondents considered the evaluation system as an aid to manual evaluation, i.e. the user interface and the functionalities do not influence the quality of evaluation.

During the evaluation of IT VBE submissions, the submissions which were graded with 2 or more points difference by two different evaluators are assigned for grading for the third time. When comparing manual and semi-automated evaluation results, 12% of submissions (6 cases) had 2 or more point difference and were re-evaluated. This is a lower number than in IT VBE evaluation session in 2012, where 22.8% submissions were re-evaluated. It can be concluded that the grades assigned by manual and semi-automated evaluation are similar and this proves the second defended statement that *the quality of semi-automated evaluation using the proposed method is not worse than that of manual evaluation.*

Comparison of quality of evaluation IT VBE practical task submissions when applying IT VBE semi-automated evaluation software to that of automated evaluation

I analyzed IT VBE'2010 practical task submissions with maximum points from at least one evaluator for all alternative criteria. Table 2 contains the data about the scope of the exam and the number of analyzed programs.

	Submitted total	Submissions with points for altern. criteria
First task	1224	34
Second Task	1019	62
Both tasks	2243	96

Table 2: Statistical data about the submissions of IT VBE'2010

Submissions described above were chosen for the research because the evaluators considered them to be close to the working ones (otherwise those submissions would not have received many alternative points). Such programs comprise about 5% from all submissions and confirm the assumption that the quality of automated evaluation is not satisfactory for the evaluation of IT VBE.

This research revealed different attitudes of the evaluators, because not in all cases both evaluators assigned full points for alternative criteria. The criteria for which such grades differed were investigated more thoroughly. The reasons for different grading were ambiguous criteria and subjectivity of the evaluators. An example of ambiguous criterion could be "Global variables are not used". If a program included a global array different evaluators interpreted that differently. Another example could be the criterion "Initializing sum variable". Some evaluators assumed that global variables do not have to be initialized, because they are initialized automatically by FreePascal compiler.

I also analyzed typical errors and the reasons why the programs did not get the points for testing. The following reasons were discovered: using different dialects of Pascal, inattentive commenting, errors done in a hurry (e.g. wrong file names).

Another interesting founding was that the difference between grades assigned during the third and the second evaluation was lower than between the grades assigned during the second and the first evaluation. This can be explained by the increasing qualification of the evaluators for the given task.

I also analysed histogram of semi-automatic evaluation results for the year 2010 exam. Use of improved method enabled to spread results wider in the histogram. This helps to rank students in better way, so the quality of evaluation is better.

The summarized results of the experimental research are as follows:

- The developed IT VBE practical task evaluation software has implemented the proposed semi-automated testing and evaluation method.
- Approximately 10 000 programs have been evaluated using this system since 2006.
- The evaluation efficiency increase obtained by applying the proposed semi-automated testing and evaluation method has been investigated. It has been concluded that the evaluation is performed 1.4–2.2 times

faster if the proposed semi-automated evaluation is applied instead of manual evaluation.

- I analyzed evaluation quality when applying the new semi-automated testing and evaluation method. The analysis confirmed that the evaluation quality when using the proposed evaluation method is not lower than that when using manual evaluation, and the evaluation quality is lower if compared to typical automated and semi-automated evaluation (especially if the programs being evaluated are not functional).

5. Possible Applications of Proposed Semi-Automated Evaluation Method

The proposed improved semi-automated evaluation method does not create separate class of evaluation methods and software, but rather should be considered as collection of improvements to typical semi-automated evaluation systems. Therefore it is easily implementable into other semi-automated evaluation systems with clearly defined teacher-evaluator role.

In order to confirm the assumption above there were made the experiments with e-learning environment *Moodle* plug-in. That was made in accordance with *Edujudge* project. The required software enabling the suggested method was implemented in the plug-in and it turned out that this was rather easy from the technical point of view. Most problems arose from the fact that the use of data from the teachers' side had to be changed. The teacher role did not have a foreseen functionality neither to edit the submissions of the students, nor to evoke the automated program testing module. Therefore in the experimental *Edujudge* module the teacher and the student roles intertwined.

Implementation of distributed response evaluation program did not cause problems. It was partly because the module is very isolated and has no direct relationship with the whole system, makes no direct changes to the database and works only on demand of the central testing system part. The main problem remained to ensure delivery of detailed feedback to the students and the teachers.

Other direction of application of new semi-automatic method may have good educational implication. The quality of feedback is highly important factor of learning efficiency. Often the feedback consists of summary of testing (passed/failed tests), the number of assigned points and a phrase from predefined dictionary. Meanwhile, the application of this method can allow to return the corrected functioning program in a form of feedback. The sequence in which the evaluator made the corrections until the program started functioning is also important. If that sequence is returned as feedback it can show the student the way the evaluator was thinking and searching for bugs and mistakes and teach the student to search for mistakes as well.

Results and Conclusions

- I investigated and classified the functionalities of modern automatic and semi-automatic programming assignments testing systems and discovered that this class of software still encounters a lot of problems, the systems are being improved and new testing systems are being developed.
- There is little research on semi-automated evaluation systems in the scientific papers. I discovered no research on the interaction between the evaluation results provided by the automatic evaluation system and the manual evaluation results.
- Improvements to semi-automatic evaluation method presented in this work allows the evaluator to use automatic testing system interactively, to carry out the experiments with the program being evaluated, to analyze interactively the consequences of the errors discovered in the programs of the students, and to consider the weights of the mistakes made by the students.
- Lithuanian Information Technology State Matura exam practical task assessment software presented in this work implements the proposed semi-automatic evaluation method. The system has been used successfully by the National Examination Centre; it is constantly being improved, adapted to the changing needs.
- I compared the effectiveness of manual evaluation method to proposed semi-automated evaluation method. The efficiency increased from 1.4 to 2.2 times.
- Qualitative analysis of evaluation results shows that results obtained by using the proposed semi-automatic evaluation method and the automatic testing differ significantly. It was confirmed that the application of the proposed method resulted in higher quality of evaluation.

The Practical Value of the Results

The created method has been implemented into the Evaluation system for Lithuanian Information Technology Maturity Exam. This raised the efficiency of the evaluation several times.

The method has also been also been implemented and tested in the Edu-judge plug-in to the learning management system *Moodle*. The experiment confirmed that the improved semi-automatic evaluation method can be easily implemented in other systems.

List of Literature, referenced in this Summary

Ala-Mutka, K. (2005). A survey of automated assessment approaches for programming assignments. *Computer Science Education*, 15(2):83–102. 8, 9

Douce, C., Livingstone, D., and Orwell, J. (2005). Automatic test-based assessment of programming: A review. *ACM Journal of Educational Resources in Computing*, 5(3):1–13. 8

Hollingsworth, J. (1960). Automatic graders for programming classes. *Communications of the ACM*, 3(10):528–529. 8

Ihantola, P., Ahoniemi, T., Karavirta, V., and Seppälä, O. (2010). Review of recent systems for automatic assessment of programming assignments. In *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*, Koli Calling '10, pages 86–93, New York, NY, USA. ACM. 8

Kasanen, E., Lukka, K., and Siitonen, A. (1993). The constructive approach in management accounting research. *Journal of Management Accounting Research*, 5:243 – 264. 6

Liang, Y., Liu, Q., Xu, J., and Wang, D. (2009). The recent development of automated programming assessment. In *Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on*, pages 1 –5. 8

Queirós, R. and Leal, J. P. (2012). Programming exercises evaluation systems - an interoperability survey. In *CSEDU (1)*, pages 83–90. 8

Romli, R., Sulaiman, S., and Zamli, K. (2010). Automatic programming assessment and test data generation a review on its approaches. In *Information Technology (ITSim), 2010 International Symposium in*, volume 3, pages 1186 –1192. 8

Presentations in Scientific Conferences

- 12th International Conference on Computer Systems and Technologies, CompSysTech'2011, Vienna, Austria, Vienna University of Technology.
- 2nd International Conference Social Technologies'2011: ICT for Social Transformations, Vilnius, Lithuania, Mykolas Romeris University, Faculty of Social Informatics.
- 15th International Scientific Conference "Computer Days – 2011", Klaipėda, Lithuania, Klaipėda University.
- 10th International Conference on Computing Education Research, "Koli Calling 2010", Koli National Park, Finland.
- 51st Conference of the Lithuanian Mathematical Society, 2010, Šiauliai, Lietuva, Šiauliai University.
- 14th Annual Conference on Innovation and Technology in Computer Science Education ITiCSE'2009, Paris, France, Université Pierre et Marie Curie.
- 14th International Scientific Conference "Computer Days – 2009", Kaunas, Lithuania, Kaunas University of Technology.
- 8th International Conference on Computing Education Research, "Koli Calling 2008", Koli National Park, Finland.

Publications printed in the peer-reviewed periodicals:

- **SKŪPAS Bronius** (2011). Pasikeitimų IT valstybiniame brandos egzamine analizė. *Informacijos mokslai*, ISSN 1392-0561, 57, 115–123.

- **SKŪPAS Bronius** (2010). Feedback improvement in automatic program evaluation systems. *Informatics in Education*, ISSN 1648-5831, Vol. 9, N. 2, 229–237.

- **SKŪPAS Bronius** (2010). Automatinio ir pusiau automatinio vertinimo ypatumai IT valstybiniame brandos egzamine. *Lietuvos matematikos rinkinys*, ISSN 0132-2818, 51, 154–159.

- POHL Wolfgang, BURTON Benjamin A., DAGIENE Valentina, **SKŪPAS Bronius**, FAKCHAROENPHOL Jittat, FORIŠEK Michal, HIRON Mathias, OPMANIS Martinš, van der VEGT Willem (2010). Get Involved! The IOI Workshop 2010, Its Goals and Results. *Olympiads in Informatics*, ISSN 1822-7732, Vol. 4, 158-169.

- **SKŪPAS Bronius**, DAGIENĖ Valentina, REVILLA Miguel (2009). Developing Classification Criteria for Programming Tasks. *ACM SIGCSE Bulletin*, ISSN 0097-8418, Vol. 41, N. 3, 373–373

- **SKŪPAS Bronius** (2009). Automatinio mokinių programų vertinimo sistemų lyginamoji analizė. *Informacijos mokslai*, ISSN 1392-0561, 50, 147–152.

Other publications:

- DAGIENĖ Valentina, **SKŪPAS Bronius** (2011). Semi-Automatic Testing of Program Codes in the High School Student Maturity Exam. *Proceedings of the 12th International Conference on Computer Systems and Technologies*, (ACM International Conference Proceedings Series Vol. 578), ISBN 978-1-4503-0917-2, 564–569.
- VERDÚ Maria P., VERDÚ Elena, de CASTRO Juan P., PÉREZ María Á., REGUERAS Luisa M., DAGIENĖ Valentina, **SKŪPAS Bronius** (2010). Making a Contest with Edujudge and Questournament. *Edujudge System Handbook: How to Organize Programming Competitions in Moodle Courses*, ISBN 978-84-937580-4-2, 99–128.
- **SKŪPAS Bronius**, DAGIENĖ Valentina (2010). Observations from Semi-Automatic Testing of Program Codes in the High School Student Maturity Exam. *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*, ISBN 978-1-4503-0520-4, 31–36.
- DAGIENĖ Valentina, DAGYS Viktoras, **SKŪPAS Bronius** (2009). „EduJudge“ projektas programavimo kompetencijoms gerinti. *Kompiuterininkų dienos–2009, Informacinės ir komunikacinės technologijos mokykloje*, 162–164.
- **SKŪPAS Bronius**, DAGIENĖ Valentina (2008). Is Automatic Evaluation Useful for the Maturity Programming Exam? *Proceedings of 8th International Conference on Computing Education Research*, ISBN 978-1-60558-385-3, 117–118.

About the Author

Bronius Skūpas was born in Lithuania in Vilnius on 20 August 1974.

In 1992, he graduated from Vilnius 41 secondary school. In 1996 he acquired Bachelor's Degree in Informatics from Vilnius University Faculty of Mathematics. He gained Master's Degree in Informatics at Vilnius University Faculty of Mathematics in 1998. In 2011 he gained qualification of a senior teacher of Information Technology.

From 2008 to 2012 he has been at PhD studies in Vilnius University Institute of Mathematics and Informatics.

Reziume

Darbo aktualumas

Programinės įrangos kūrimo tempai spartėja ir tam reikalingi vis nauji informacinių technologijų (IT) profesionalai: projektuotojai, programuotojai, testuotojai. Šie būsimi darbuotojai dar mokyklose supažindinami su IT pradmenimis, programavimo pagrindais. Jie taip pat mokosi savarankiškai, tęsia mokslus aukštosiose mokyklose.

Programavimo mokymas yra sudėtingas procesas, nes jis sietinas ir su kūrybišku mąstymu, ir su griežtai formalizuojamomis užduotimis, praktiniais programavimo darbais. Mokiniai ir studentai (toliau mokiniai) turi būti mokomi kurti ne tik veikiančias, bet ir kokybiškai suprojektuotas, patikimas, teisingai veikiančias programas. Mokinių sukurtų programų patikrinimas bei įvertinimas reikalauja daug mokytojo ar dėstytojo pastangų ir laiko.

Siekiant galimybės daliai vertinimo kriterijų taikyti automatinį programų testavimą programavimo užduotys yra griežčiau specifikuojamos, t.y. tiksliai nurodoma, iš kokių duomenų ir kokius rezultatus reikia gauti, dažnai nurodomi darbo metodai ar dalis naudotinų algoritmų. Tai leidžia vertinimo metu taikyti faktų tikrinimo metodą. Tiksliai identifikuotas galimas patikrinti faktas vadinamas vertinimo kriterijumi.

Vertinimo kriterijų, tinkamų automatiniam testavimui, atpažinimas yra svarbus, nes jis leidžia vertinti darbus efektyviau. Taip pat tai leidžia sukurti mokymo modelius, kuriuose didžioji techninio darbo dalis atliekama automatizuotai ir mokytojas gali koncentruotis į svarbiausias mokymo problemas.

Automatinis programavimo užduočių sprendimų testavimas aktualus ne vien tik mokymo proceso metu, bet ir per programavimo egzaminus, profesinius programavimo žinių patikrinimus, priimant į darbą, organizuojant programavimo varžybas.

Daug autorių pastebi, kad visiškai automatinis testavimas, besiremiantis statine bei dinamine analize, negali būti pilnai objektyvus. Todėl pasitelkiamas pusiau automatinis testavimas, kuris leidžia lanksčiau naudotis automatinio testavimo privalumais.

Darbe tiriamos galimybės naudoti pusiau automatinio mokinių parašytų programų testavimo sistemas programavimo mokymo mokyklose, nuotolinio mokymo, taip pat varžybų ir egzaminų kontekstuose. Taip pat analizuotos pasitaikančios vertinimo klaidos. Pasinaudojant žinomais vertinimo procesais sukurtas naujas metodas tikslesniam ir greitesniam testavimui bei vertinimui.

Darbo tikslai ir uždaviniai

Darbo tyrimų objektas yra automatinės ir pusiau automatinės programavimo užduočių testavimo sistemos, jų architektūra, veikimo metodai ir algoritmai, sąveika su vartotojais.

Darbo tikslas: išanalizavus esamas automatines ir pusiau automatines programavimo užduočių vertinimo sistemas susisteminti architektūrinius sprendimus ir sukurti vertinimo metodą, kuris patobulintų sistemos-vertintojo sąveiką ir padidintų vertinimo efektyvumą ir kokybę.

Darbo uždaviniai:

1. Atlikti automatinių ir pusiau automatinių programavimo užduočių testavimo sistemų teorinių ir eksperimentinių pasiekimų analizę.
2. Ištirti automatinių ir pusiau automatinių programavimo užduočių testavimo sistemų savybes, jų sąsajų su vartotojais ypatumus bei teikiamas paslaugas, išskirti ir susisteminti šių programų sistemų metodines ir architektūrines problemas.
3. Pasiūlyti patobulintą metodą pusiau automatiniam programavimo užduočių vertinimui ir testavimui, kuriuo remiantis būtų galima pagerinti darbų vertinimo kokybę bei efektyvumą.
4. Atlikti pasiūlyto metodo tyrimą konstravimu ir įvertinti metodo teikiamas naujas galimybes.

Darbo rezultatų aprobavimas

Sukurtas metodas pritaikytas Lietuvos valstybinio informacinių technologijų brandos egzamino praktinės dalies vertinimo sistemoje. Jo naudojimas kelia egzamino vertinimo efektyvumą keletą kartų.

Metodas taip pat bandytas su virtualios mokymosi sistemos Moodle sistemos Edujudge įskiepiu, sukurtu pagal ES finansuotą projektą. Įskiepis tobulintas siekiant mokytojo komfortiškesnio vertinimo ir mokiniui aiškesnių vertinimo rezultatų pateikimo. Įsitikinta, kad metodas gali būti nesunkiai diegiamas ir kitose sistemose.

Skaityti 3 pranešimai Lietuvos konferencijose, 6 pranešimai tarptautinėse konferencijose. Publikuoti 6 straipsniai recenzuojamuose periodiniuose leidiniuose ir 4 straipsniai atspausdinti tarptautinių konferencijų leidiniuose ir patalpinti ACM duomenų bazėje. Be to, yra 2 publikacijos knygose.

Disertacijos struktūra

Darbą sudaro: terminų ir santrumpų žodynėlis, penkios pagrindinės dalys – skyriai, išvados ir rezultatai, naudotos literatūros sąrašas ir priedai.

Pirmajame skyriuje pateikiamas darbo įvadas. Pristatomas darbo aktualumas, tyrimų objektas, darbo tikslai ir uždaviniai, tyrimų metodai, mokslinis naujumas, praktinė darbo reikšmė, ginami teiginiai ir darbo aprobavimas.

Antrajame skyriuje pateikiama informacija apie automatines ir pusiau automatines programų testavimo sistemas, pateikiama medžiaga apie atliktą šias sistemas aprašančių straipsnių analizę. Taip pat pateikiama informacija apie

dabartinių sistemų klasifikavimą, jų veikimo principus ir pasirinktų sistemų lyginamoji analizė.

Trečiajame skyriuje detalizuojamas tyrimo konstravimu darbo planas, iškeliamos darbinės hipotezės, abstrahuojami reikalavimai kuriamam metodui, paaiškinami konkrečios kurtos sistemos techniniai reikalavimai ir metodo kūrimo eiliškumo tvarka pateikiami projektiniai metodo sprendimai. Taip pat pristatoma, kaip kintantys reikalavimai egzamino vertinimui ir vykdymui įtakoja metodo sprendimus, bei grąžina į sistemos perprojektavimo fazę.

Ketvirtajame skyriuje pateikiama informacija apie atliktus eksperimentinius sukurto metodo efektyvumo ir jo taikymo kokybės tyrimus.

Darbo pabaigoje pateikiamas rezultatų apibendrinimas ir išvados.

Prieduose pateikiama: egzamino sąlygų ir vertinimo schemų pavyzdžiai, klausimynas vertintojams.

Trumpos žinios apie autorių

Bronius Skūpas gimė 1974 m. rugpjūčio 20 d. Vilniuje. 1992 m. baigė Vilniaus 41-ąją vid. mokyklą (dabar Karoliniškių gimnazija). 1996 m. Vilniaus universiteto matematikos fakultete įgijo informatikos bakalauro kvalifikacinį laipsnį. 1998 m. Vilniaus universiteto matematikos fakultete įgijo informatikos magistro kvalifikacinį laipsnį. 2001 m. Vilniaus licėjuje įgijo informatikos vyresniojo mokytojo kvalifikacinę kategoriją. 2011 m. Vilniaus licėjuje įgijo informacinių technologijų mokytojo metodininko kvalifikacinę kategoriją. 2008–2012 m. doktorantas Vilniaus universiteto Matematikos ir informatikos institute.