МАТЕМАТИКА И МАТЕМАТИЧЕСКО ОБРАЗОВАНИЕ, 2011
MATHEMATICS AND EDUCATION IN MATHEMATICS, 2011
Proceedings of the Fortieth Jubilee Spring Conference
of the Union of Bulgarian Mathematicians
Borovetz, April 5–9, 2011

A SOFTWARE PLATFORM FOR TEACHING PROGRAMMING WITH GRADING SYSTEMS[*]

Krassimir Manev, Miloslav Sredkov, Petar Armyanov

Grading systems (GS) are inevitable component of the programming contests. Recently some projects are developed for using GS in education of programming. This paper describes a software platform aimed to integrate different GS created or used by the authors in order to provide simple and efficient environment for supporting the educational process. The main elements of the platform extracted in preliminary analysis are listed and one possible architecture of a platform is specified.

1. Introduction. The usage of computing systems (computers and corresponding software) in modern education is inevitable – for preparing and distributing teaching materials, for searching the necessary for elaborating of assignments information, for examination, for maintaining the contact among the teacher and students, and for other educational activities. These usages of computing systems are universal, i.e. intrinsic for all educational subjects.

There are more possibilities for using computing system for a teaching Informatics ad especially for teaching programming and algorithms. The extracurricular activity that is closest to teaching programming and algorithms is the *competitive programming*.

Programming contests have more than 30 years of history [1,2]. That is why a huge amount of resources – tasks, test cases, methodology, software tools, etc. – are created by organizers of programming contests. Among the software tools for organization of programming contest the most important are *grading systems* (GS) aimed to collect and evaluate contestants' solutions. Description of the functionality of GS and examples of such systems are given in [3].

Recently some attempts was made to adapt GS for teaching programming in universities as well as in secondary schools [4]. In [5] the challenges that education of programmers put in front of the teachers was outlined and for each challenge was shown how GS could help.

In this paper we present the fundamentals for development of a software platform for teaching programming and algorithms based on the existing GS. In Section 2 programming contests and GS are presented. In Section 3 the challenges of education in programming, outlined in [5], are listed. Section 4 describes the elements of a software

[*]**2000 Mathematics Subject Classification:** 68N99.
Key words: education in programming, programming contests, grading systems.
This work is partly supported by Scientific Research Fund of Sofia University under the Contract 247/2010.

platform for education in programming and algorithms. In Section 5 one possible architecture of such platform is given and in Section 6 – some conclusions are shared.

2. Programming contests and grading systems. ***2.1. Programming contests.*** In 1977 the Association of Computing Machinery starts its Collegiate Programming Contest for students in the American universities. Nowadays the ACM ICPC is a world contest [1]. The first contests for school students in Bulgaria were organized in the early 80-th of the past century. In May 1989, Bulgaria organized and hosted the First International Olympiad in Informatics (IOI) for school students [2]. Recently, different programming contests are organized by some professional organizations ([6, 7]). Many web sites propose a continuous on-line training process with competitive elements ([8–11]).

During a programming contest contestants have to solve one or more *tasks.* The *statement* of the task usually contains an object with given properties called *input.* Contestants have to write a program, called *solution* of the task, which is able for a given (but unknown by contestants) input, to find a new object– called *output*, that has to be in prescribed relations with the input. The main features of the solution that are evaluated are its *correctness* and *time complexity.* The correctness of the program is checked with prepared by the author *test cases.* The speed of the programs is controlled by a specific for each task *time limit.* If a task allows more than one correct output the correctness of the output is checked by a special program called *checker.*

2.2. Grading systems. Checking of correctness and speed of computer programs is a complex set of activities. From the end of 80's and beginning of 90's years of the past century specific software systems called *grading systems* (GS) were introduced to help evaluation of programming contests. Nowadays GS are inevitable part of the programming contests and training sites. They incorporate all elements that are necessary for organizing the contest – tasks statements, time limits, test cases, expected outputs, checkers, contestants' programs, results of evaluation, etc.

GS receives submitted by contestants programs, compile them, run them on the input test cases, check the result and assign scores in accordance with defined in the system evaluation policy. Two *styles* of evaluation are popular – positive score is assigned only when all test passed successfully (*ICPC-style*) or scores are assigned separately for each successfully passed test (*IOI-style*). Depending on the policy GS could inform the contestant for her/his results immediately after solution submit (ICPC) or after the end of the contest (IOI).

In Bulgaria three GS are in active use nowadays:

SMOC is the GS used in all national programming competitions for school students [12]. Starting as a modification of GS of IOI'2002, now SMOC is completely rewritten. It was used in organization of four international contests (including IOI'2009) and many on-lines. The main advantage of SMOC is the safety evaluation process and the precise time measurement developed in cooperation with the Technical Committee of IOI. It is closely coupled with IOI-like contests and does not support an archive of used tasks.

spoj0 is a GS designed to work continuously and to run several contests in ICPC-style at a time [13]. It was used for organizing internal contests, exams, and homework assignments in several university courses. The system contains collection of tasks, archived in groups as they were proposed for training, homework or examination contests. The system has an exporter for plagiarism checkers. spoj0 uses only tools of the operating system to provide secure execution and has no precise time measuring. System administration is performed by command line scripts on the hosting machine and is not easy for not experienced users.

Maycamp Arena is a site dedicated especially to training and organizing on-line programming contests [14]. The GS of the site is a cloning of spoj0, maintaining an evaluation in IOI-style. Archived tasks are classified by the domain of the used algorithm. Very important feature of Maycamp Arena is the collection of statistical data about performance of contestants and hardness of the tasks.

3. Challenges of teaching programming. Analyzing in [5] the education in programming and algorithms, in order to identify how GS could help the process, we outlined the following main challenges:

3.1. Preparing samples and etudes is really hard even for experienced teachers. A huge amount of programming etudes could be taken from archives of programming contests. For this purpose the archived tasks have to be **appropriately classified** by domain and hardness, and the solutions – by their effectiveness. Above mentioned GS have some elements of the necessary functionality but they are not developed enough.

3.2. Checking large amount of assignments is the greatest challenge for the teacher. To read the program code and to understand the idea of the program is quite difficult. And here is the greatest advantage of using GS. They are able to **check automatically** assignments, to evaluate them and to produce the corresponding reports within a minute.

3.3. Depending on the stage of the educational process, ***different kind of evaluations should be applied***. For exercises in class it will be helpful for the student to receive from the system the inputs and the correct output too, in order to debug the program. When the student is not able to solve the task then GS could provide a trivial solution and recommend more easy tasks to be solved as a preparation, etc.

3.4. Evaluation of students manually is very rough – usually the teacher assigns and checks 2-3 home works and 1-2 quizzes that cover part of the material. Using GS, teachers could check practically each of the studied topics, providing ***more flexible and adequate grading***.

3.5. Despite the efforts of the teachers, ***the plagiarism*** still exists in education. Embedding in GS a tool for discovering of plagiarism is a possibility to eliminate this negative phenomenon.

4. Elements of the platform. The main element of the platform is the `Task`. One attempt to formalize the presentation of competitive tasks was described in [15]. We are giving an alternative formalization below:

```
Task { ID; Domain; Kind; Evaluation;
   Formulation { Statement;Input format; Output format;
      Constraints; Example; Illustration; Explanation; }
   }
   Solutions {
      Solution { Source; Explanation; Complexity; }
      ...
   }
   Test set { Time limit; Checker;
      Test cases {
        Test case { Input; Output; Auxiliary; Max score; }
        Test case { Input; Output; Auxiliary; Max score; }
        ...
   }
   Statistics { ... };
}
```

Most of the subelements of the Task are self-explanatory. Possible values of the Domain are the subjects of the tasks – sorting, searching, graphs, etc. The element Kind describes the content of the student submit – output only, usual program, `main()` function to communicate with module(s) of the author, module(s) to communicate with `main()` function of the author or program fragment. In Evaluation the style of evaluation will be marked – ICPC-style, IOI-style or not so strong styles (open input, open input and output, etc.)

The element:

```
User { ID { Login; Password; } Role;
       Personals { Name; Nick; e-mail; ... } Statistics { ... }
}
```

is for identification of users. Possible roles are Administrator, Teacher and Student. A person will be able to register in two roles. For example, a Student will be able to enter as a Teacher, to compose a training assignment and then to enter as a Student and to try to solve it.

The GS of the platform will be described with the element `Grader { ID; Address; ... }`. Beside the ID and the address of the server some additional subelements will be provided depending of GS.

The main activity of the platform will be implemented through the elements:

```
Contest { ID; User; Date; Start; End; Tasks { Task; ...  } }
Submit { ID; Contest; User; Task; Language; Source; Date; Time;}
Grading request { ID; Grader, Submit; Result; }
```

where Contest.User identifies the Teacher that defined the contest and Submit.User identifies the Student that sent the Submit

Some other elements will be necessary that could not be specified so strictly, for example different kind of Reports, Rankings and Statistics.

5. Architecture of the platform. One possible architecture of the platform that incorporates the existing GS is shown in Figure 1.

The Repository will contain all necessary for the functionality of the platform data – classified Tasks, registered Users, planned Contests, Submits of the Students, Grading requests, Statistics, etc. The data will be stored in the Repository through the Web-based Teacher's/Student's interface, presented with some language for data structuring (XML, for example). The necessary Reports and Statistics will be extracted from the Repository and visualized through the Teacher's/Student's interface, too.

The active component of the platform is the Scheduler. It continuously scans the list of the stored Contests and Submits. For each actual Submit the Scheduler forms the corresponding Grading request, sends it to the appropriate Grader and stores the obtained Result back in the Repository. For the purpose, a specific communication protocol for each of the existing Graders has to be included in the Scheduler in order

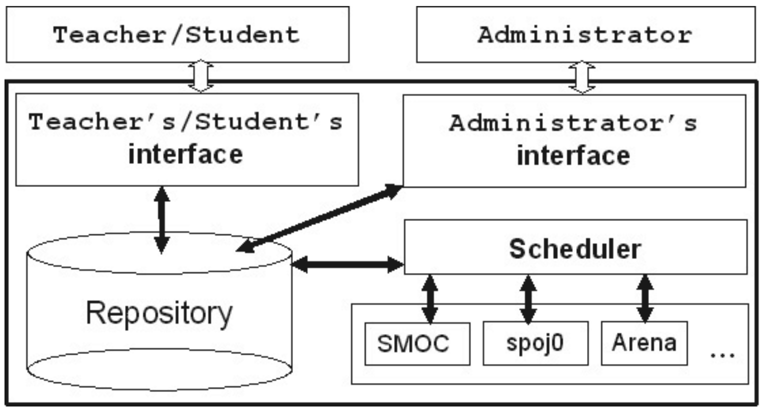

Fig. 1. Architecture of the platform

to be able to transform `Grading requests` from the specified above format to formats different `Graders`.

Because of the specific character of administration of the Repository, Scheduler and Graders, the corresponding functionality is concentrated in the Administrator's interface.

6. Conclusion. The implementation of the specified platform is in progress and should be relatively easy. The three mentioned above GS are used in many national and international contest and proved their quality. Something more, each of these GS has own scheduler so the **Scheduler** of the platform could be implemented as an extension of some of the existing schedulers.

Some parts of the necessary **Teacher's/Student's interface** are also implemented in the existing GS, as well as the reporting of the `Results`. But **Teacher's/Student's interface** of the platform has more functionality which have to be built from scratch. Unfortunately, existing GS have no **Administrator's interface** at all and this interface has to be built from scratch too.

The most difficult for implementation part of the platform is the implementation of the Repository – to collect tasks, to classify them and to store them in the specified format. Hundreds of tasks created for national contests are spread in different places and are not completely documented. Preparing these tasks for usage within the platform will need a huge amount of manual work.

The main advantage of proposed platform is the integration of all positive elements of existing GS. We expect that it will lead to intensification of the teaching process and preparation of more qualified programmers.

REFERENCES

[1] International Collegiate Programming Contest. `http://cm.baylor.edu/welcome.icpc`

[2] International Olympiad in Informatics. `http://ioinformatics.org`

[3] Kr. Manev, M. Sredkov, Ts. Bogdanov. Grading Systems for Competitions in

Programming, *Mathematics and Education in Mathematics*, Proc. of 38-th Spring Conference of UBM, Borovetz, 2009.
[4] P. Ribeiro, P. Guerreiro. Early Introduction of Competitive Programming. *Olympiads in Informatics*, vol. 2, 149-162, 2008.
[5] Kr. Manev, M. Sredkov, Ts. Bogdanov, V. Mihov. Grading Systems in Teaching of Programming, to appear.
[6] Top Coder Competitions. `http://www.topcoder.com/tc`
[7] Google Code Jam. `http://code.google.com/codejam`
[8] USACO Training Gateway. `http://train.usaco.org/usacogate`
[9] UVa Online Judge. `http://uva.onlinejudge.org`
[10] Timus Online Judge. `http://acm.timus.ru/`
[11] Откритый кубок по програмированию. `http://opencup.ru/`
[12] SMOC grading system. `http://openfmi.net/projects/pcms`
[13] SPOJ0 grading system. `http://judge.openfmi.net`
[14] Maycamp Arena. `http://arena.maycamp.com`
[15] T. Verhoeff. Programming Task Packages: Peach Exchange Format. *Olympiads in Informatics*, **2** (2008), 192–207.

Krassimir Manev
Miloslav Sredkov
Petar Armyanov
Faculty of Mathematics and Informatics
St. Kliment Ohridski University of Sofia
5, J. Bourchier Blvd
1164 Sofia, Bulgaria
e-mail: manev@fmi.uni-sofia.bg
miloslav@gmail.com
parmyanov@fmi.uni-sofia.bg

СОФТУЕРНА ПЛАТФОРМА ЗА ПРЕПОДАВАНЕ НА ПРОГРАМИРАНЕ СЪС СЪСТЕЗАТЕЛНИ СИСТЕМИ

Красимир Манев, Милослав Средков, Петър Армянов

Състезателните системи (СС) са незаменимо средство за организация на състезания по програмиране. Напоследък СС се използват и в обучението по програмиране. В статията е предложена платформа, която да интегрира възможностите на СС, създадени или използвани от авторите. Целта е изграждането на проста и ефективна среда за обучение по програмиране, подпомагаща учебния процес. Специфицирани са основните елементи на платформата, като резултат от предходно изследване, и една нейна възможна архитектура.