PORTAL FOR INTRODUCTORY INSTRUCTION IN COMPUTER PROGRAMMING

Volodymyr Mykhaylovych Bondarev
PhD, full professor of the software engineering department
Kharkov National University of Radio Electronics, Ukraine

Oleksandr Fedorovych Osyka
PhD, associate professor of the software engineering department
Kharkov National University of Radio Electronics, Ukraine

Summary. The task of the project was to create an Internet-based universal set of services for a study course “Computer programming” and alike. The services support various academic activities: lectures, tests, tutorials, labs, and unsupervised students work in the course. Many services are united around a database of computer programming problems. Instructors and students are provided with different tools. Instructors use services that help in preparation for classes, automate knowledge monitoring, check the authenticity of problem solutions, work for study motivation of students, etc. Students get access to course lecture notes, problems for solution with automatic solution verification, means of online course discussion with peers and instructors, etc. The portal has been successfully used for four years at the Kharkov University of Radio Electronics, Ukraine.

Keywords: Internet portal, University education, computer programming, problem database, solution verification, knowledge monitoring, lectures, tests.

1. INTRODUCTION

Computer programming is a prestigious specialty that attracts a permanently growing number of students. To meet the demand and to keep educational standards high, many researchers and application programmers concentrate on the development of computer-based methods and tools for teaching computer programming [1,2,3,4]. Such efforts are quite diverse in their aims and approaches. Here we consider just some types of software systems that are promising for teaching and learning computer programming at universities.

Many computer experts work on creating multi-language programming environments with integrated databases of problems that provide a number of services for students undergoing various programming tests [5,6]. Many software packages assist lecturers in preparing lecture notes and slides, delivering lectures and publishing notes/slides, etc [1,7]. Some packages have been developed to facilitate arranging and conducting tutorials by blending software with traditional classroom teaching and learning approach [8,9]. Knowledge monitoring is of a paramount importance for motivating students in their studies. The more students...
are convinced that an objective knowledge monitoring is inevitable, the better they study. Knowledge monitoring can be just one of the functions of a large comprehensive educational system or the primary task of a specialized system for monitoring students’ knowledge, attendance, ratings, etc [4,10]. Certainly, there are other academic activities that are also aided by computer educational systems.

So there is a large variety of computer educational systems. Some of them are comprehensive that provide many services to instructors and students. Usually, they take all-round care not only of the study process but also of other adjacent activities, such as registration of class attendance, registration of payments for education, development of teaching materials, etc. Some functions may be unclaimed at a particular university, others may require inconvenient data formats, etc. Such a system can be unpractical for teaching and learning a computer programming course. For example, it may not support running a code of a program or its segment.

If possible, academic staff prefer easy-to-use specialized computer systems that serve the needs of a particular course, particular academic activity, particular university, etc. This tendency leads to using several computer packages in one programming course for supporting all kinds of its activities. It may be an inconvenience by itself for instructors and students.

The aim of the present project was to create a specialized programming instruction portal that gives a substantial assistance to instructors and students in teaching and learning introductory-level university courses like “Introduction to computer programming”, “Computer programming”, “Data structures”, etc.

Main features of the portal are as follows:
- It can be used to serve the needs of a University, a faculty or a department;
- It contains several hundred of programming problems for solutions using C#, JavaScript, Python, Haskel, and C++;
- Instructors are offered services in creating and publishing lecture presentations, conducting lectures and tests, arranging computerized problem solutions during tutorials and labs, obtaining data about students activity in the course;
- Students get access to published lecture notes, can solve problems, get automatic solution verifications, discuss solutions, can see their grades and rating in the course.
- The portal background contains a database of problems offered for solution that performs such functions: presenting problem statements, verifying solutions, adding new problems to the database, and editing problems.

Below we describe the components of the portal.

2. PRESENTATION PREPARATION

An instructor uses a presentation while delivering a lecture. It is also utilized by students as lecture notes when they study the subject out of a classroom. A lecture presentation can include a content which is visible to students when they study unsupervised and is hidden at a lecture time. An instructor can create and save on the portal a series of presentations for the entire course.

To a certain degree, the portal automates the construction of a lecture presentation. An instructor prepares only a “rough copy” of a presentation. It obtains
its final layout after it is compiled into a special HTML document. Best examples of Internet courses show that a “discussed” part of a presentation is not removed from the screen immediately. It is replaced by a new material gradually as it comes into play. This method of displaying a presentation text is used in the portal under discussion. Figure 1 shows some initial text and its corresponding compiled HTML document.

Fig. 1. Initial presentation text and its corresponding HTML document

A rough copy (that is an initial text) of a presentation is a plain text with a minimal number of tags. The text consists of small fragments which we will call “sections”. The start of a section is labeled by a tag which also denotes the section type: @1, @2 indicate headers, @3 – text, @4 – definition, @5, @6 – problems. A section ends where the next section begins. Blank lines between sections have no significance. Here is a brief description of slide sections, which are sufficient for the vast majority of presentations in the course.

@1 type of a section is a header of the entire presentation or its part. The initial text header is a single line. In its formatted presentation, it is the same line in the font which is appropriate for the header. The first header is the title of an entire presentation.

@2 type of a section is a sub-header. It has the same structure as a header and is used to single out a smaller fragment of a presentation.

@3 type of a section is a main text. It is the section type where three styles can be applied: {{program code block}}, {highlighted text fragment}, [[ reference]]. The content within double square brackets informs about the type of a reference. In the final presentation, it can be a picture, a reference to an internet resource or a question of the author. Such questions are discussed later. Besides, in the final formatted presentation, all words in Latin font are highlighted.
@4 type of a section is a definition. It is a simple framed text. It attracts attention to important text fragments, e.g. to description of the syntax of some operator.

@5 type of a section contains a problem with a step by step solution. Sometimes it is desirable to invite listeners to compose a simple program or, at least, to think how it can be done. Some solution variant should be prepared in advance, to be presented later in one piece or by parts. Such a section begins with the header and the statement of a problem which is followed by blocks of the program code. At first, students see only the header and the statement of a problem with its solution (code blocks) hidden. After some time for students to think about possible solutions, an instructor opens solution blocks, one at a time, by clicking on the problem header.

@6 type of a section contains a programming problem with automated solution verification. Students have to solve such problems after a lecture, using lecture notes. In the initial text, such a section contains only the problem number and header. Its format is as follows: @6 problem_number/problem_header.

3. DELIVERING LECTURES

To deliver a lecture, an instructor chooses a corresponding presentation and starts its demonstration. The HTML document of the presentation is opened, one slide at a time, sequentially.

In the upper part of the browser window, there is a semitransparent panel that shows the remaining lecture time, the total number of lecture slides and the number of slides opened. The panel also contains indicators of questions asked by listeners, a pencil for drawing on a current slide. The panel is visible when a lecture is being delivered. It is hidden when a student looks through a published presentation.

During a lecture, students can put their questions to the instructor not only orally but also in a written form, by sending them from the student version of the portal. The instructor sees a question indicator. He/she can open the question and answer it orally, at the lecture, or later, in a written form. Written questions and answers are saved in the portal chat as messages. Note: With such an interaction format, an instructor can, as well, put his/her own question by “hiding” it in an appropriate place of the presentation. It may help to make the lecture more animated and encourage listeners to ask their own questions. After a lecture or immediately before it, the instructor may publish the presentation to make it accessible to all registered portal users.

4. TESTING

The best way to assess the ability to program computers is ask an examinee to write a program or its fragment. Writing an error-free hand-written or typed code poses an additional challenge for a student and produces disputable results. On the other hand, analyzing somebody’s writing is painful for an instructor, taking into account a possible number of code pages. That is why classroom quizzes and larger tests are organized in the same way as unsupervised problem solving activity of students, using the service of the database. In order to prepare a test, it is sufficient to choose an appropriate problem and set the time period for its solution. Figure 2, its left part, shows an example of a data sheet for preparing a test.
Certainly, one test can include several problems of different complexity levels. Our teaching experience showed that it is better to conduct several tests, each including one problem, which is worth 60, 75 or 90 points. For example, at first, one problem is offered with 60 points reward and 15 minutes for its solution. Those, who successfully completed the first test, are offered the second test, 75 points worth and 25 minutes for its solution. Students, successful in the second test, are invited to do the third one for 90 points and 30 minutes of solution time. Note: If the time assigned for some problem is over and nobody solved the problem, the instructor can give some extra time for the problem solution.

Such an approach spares time and nervous energy of an instructor. Now the main task of the instructor is to ensure the trustworthy results of a test. We have to admit that, with the development of communication tools, methods to get somebody else’s solutions also become more sophisticated. A student can use a mobile phone or computer Internet connection for passing problem identification data to another person, to get the problem solved. Students can invent something unknown at present. Fighting against all such methods is wearing and, in a certain sense, humiliating. For this reason, the test conducting system has a built-in tool for automatic “plagiarism” detection. It assesses the similarity degree of a given solution compared to some previously received solution of the same problem.

Figure 3 shows that a student with the nick-name 8Voloboy has three possible prototypes, the student 8Feschenko - two prototypes, and the student 8Sorokin – one prototype (8Shporta). It leads to a possible assumption, that the author of the original solution was the student 8Shporta, and the students, with names colored in red, just borrowed that solution. But it is only an assumption which needs a further verification. A click on any of the prototypes opens a corresponding pair of solutions for their visual comparison. It is apparent from Figure 3 that there is no complete coincidence of solutions, but the similarity is substantial.
The system's database stores all attempted solutions of a problem, successful and failed. For example, if two solutions differ in 100 symbols, and were submitted within 10 seconds then it is evident they were obtained by different people. Certainly, the “anti-plagiarism” system can sometimes produce wrong signals, in particular, if a problem is simple and its solutions are short. Note: The task of the system is not to turn an instructor into a detective, but rather to discourage students from cheating. This goal is achieved to a sufficient extent.

During a test, the instructor sees on the screen of his/her computer the status of each student (a problem has not yet been opened, is being solved, has been solved successfully, solution time expired), the solutions submitted for verification by the system and also the system response on each solution. After a test, successful and failed solutions can be discussed within a student group or on individual bases.

All the data accumulated during a test (problems, list of test participants, and all solution attempts) are saved and can be used for a further analysis.

5. DATABASE OF PROGRAMMING PROBLEMS WITH AUTOMATIC SOLUTION VERIFICATION

Problems from the database do not require students to write complete programs with data input and output of the result. A student is supposed to produce a code segment that does exactly what is stated in the problem specification. Such an approach makes it possible:
- To precisely state very different problems;
- To relieve a student of a routine work.

Here are examples of database statements of simple problems to be solved in C#.

**Problem 1. Assignment.** Declare an integer variable `a` and write value 5 in it. Declare an integer variable `b` and store value 1 in it. Increment both variables by 1, add values of `a` and `b`, store the sum in a variable `c`.

**Problem 2. Factorial.** Declare a static method `Fact()` which receives an integer non-negative number and returns the factorial of this number. (Note: 0! = 1.)

**Problem 3. Circle with constructors.** Declare a class `Circle` with public properties `X` and `Y` (coordinates of the centre) and `R` (radius) of double float type. The
class should contain a public constructor with three parameters (ordinate, abscissa, and radius) and a default constructor which initializes all properties by zero values.

Problem 4. Generic delegate. Declare a generic delegate with the name Del. The delegate should reference functions that have two parameters of the same type and return a logical value.

There can be other problem statement formats, for example, “correct an error in a given code” or “modify a given code to achieve a certain goal”, etc.

Figure 4 shows, at the background, an index page of the problem database and, at the foreground, the data sheet for composing a new problem. Besides a problem statement, the database record of a problem contains the author's solution, and a verification context. A verification context is a complete program with an input point and a specified place where to insert a verified code. When some solution is submitted for verification, it is inserted into its context. The newly composed program is compiled and executed. The context performs a test of the solution module. As a rule, it informs about the test result by throwing an exception that displays an appropriate message. The message is directed to the student in response to the solution. The insertion place of a possible solution is marked by means of two special comments.

When preparing a problem, the instructor places the author's solution into the code as a special comment. Such a solution is a necessary part of the problem. On one hand, the problem is included into the database only if the author's solution is approved by the context. Certainly, the fact that the author's solution is accepted by the context does not guarantee the absence of bugs in the context. But it shows that the context is in a workable state. On the other hand, an instructor can show the author's solution as a possible example when discussing test results with students.
At the present time, the problem database does not have any private sections. All problems may be used by any instructor. Each instructor may enter problems into the database or edit available entries. As a rule, preparation of a new problem takes from 15 to 90 minutes.

6. FEEDBACK

The portal provides instructors with a feedback on the study process. Firstly, it supplies an up-to-date information about the unsupervised work of students (use of lecture notes, solution of problems). Secondly, it makes available current and final grades for quizzes and labs in a course. The information is presented in graphical and table formats.

Figure 5 shows the graph of “attendance” of the published internet lectures. The horizontal axis presents the course study time and the vertical axis – the number of unique accesses to lecture files. Lectures are published by the instructor as they are delivered. In that order they are presented in the graph. Using filters, the instructor can choose only the student contingent which is required. Note: The published lectures are accessible to all registered users, not only to “official” students. The example from Figure 5 shows a gradual decline in students' interest with the progress of the course, but not very fast. It corresponds approximately to the weekly number of problems solved by students.

Current and final course grades are another type of feedback information. At present, the database stores only the grades of quizzes. If an instructor issues other grades also (e.g., for a course assignment or labs) it is possible to upload such grades in the portal as an Excell table. These grades will be used in performance analysis together with data from the database. This table can be completed by other data, like External Independent Testing grades or grades in mathematics.
The statistical distribution of grades and the correlation between numeric indicators of the study process are the subject of careful analysis. The quality of students’ knowledge is the result of interaction of many factors. It is assumed that, with a sufficiently large group of motivated students and a good quality of study process, the course grades should have a near to normal distribution. The analysis begins with the visualization of data in the form of a histogram. By its form and moments of distribution, it is possible to make some conclusions about the grading system. The correlation analysis provides a greater amount of information about it. (See Figure 6)

Absence of correlation where it is expected, for example, between the grades for quizzes and labs, may prompt an instructor to reconsider its grading policies. It may be especially important if many student groups study the same subject under the guidance of different lecturers and assistant instructors.

7. FUNCTIONS FOR STUDENTS

Students have access to all published lecture notes for all courses present at the portal. Any time students can solve problems found in lecture notes and get automatic solution verifications. Students can look through their own solutions of problems and somebody else’s solutions after submitting their own. It is possible to discuss solutions in the chat where students can also ask questions about lectures and problems. Questions are answered by other students or by the instructor who is in charge for the subject.

Problem solution achievements are reflected in individual and group ratings. The ratings are unofficial. They do not have a direct influence on the semester final grades in the course. But the instructor can use them for charging extra points for the earliest solutions of problems or for solutions of hardest problems.

Fig. 6. Grades correlation analysis
The student version of the portal includes a page for solving test problems. When an instructor activates some test, the page shows a problem statement, an editor window for entering a solution code, and the button “Verify”. A student enters the problem solution, presses the button, and sees if it passed the verification successfully. In the case of success, the page with the problem is closed in three seconds, otherwise, the error message is displayed and the student has to continue attempting to solve the problem. The page shows the time remaining for the current problem solution. The test is done under the supervision of a monitor who is responsible for authenticity of test results.

8. PORTAL IMPLEMENTATION

The last portal version was implemented on the base of the platform ASP.NET Core 2.0. It is a collection of several web-applications united by a common database. The portal functions related to problem solving are supported by the web-service of the programming problems database [3].

9. CONCLUSION

The above described educational portal has been used at the Kharkov National University of Radio Electronics, Ukraine, for 4 years. It has proven to be not only beneficial but also indispensable. During that period of time, the number of student groups in the department of Software Engineering has increased from 5 to 11. Beside teaching the University students, the portal is used in various training sessions and courses delivered for external clients. An important part of the portal services is performed by the database of programming problems which supports unsupervised work of students and their knowledge monitoring. The automatic verification of test solutions still needs some improvement since it does not assess the quality of a solution code. The search for such an improvement is in progress.

ACKNOWLEDGMENTS

The authors wish to thank Professor Zoya Vladimirovna Dudar, Head of the department of Software Engineering, Kharkov University of Radio Electronics, for her invariable support for the development and application of new web-based tools in teaching and learning at the department. This work was also supported by the management of the Global Logic company.

References:
[2]. Stepik - an educational platform and a constructor of on-line courses. [An Internet resource], access address: https://stepik.org/.
[3]. CS Academy - educational project in the field of computer programming. [An Internet resource], access address: https://csacademy.com/
[4]. PROMETEUS - free on-line courses developed by instructors of the Kyiv National University, the Kyiv Polytechnic University and Kyiv-Mohgiliansky Academy, Ukraine. [An Internet resource], access address: https://prometheus.org.ua/.
[5]. TopCoder - a corporation that organizes contests in computer programming. [An Internet resource], access address: http://www.topcoder.com/
[6]. ejudge – a system intended for different arrangements where automatic program code verification is required. [An Internet resource], access address: http://ejudge.ru/

