A teaching/learning support tool for introductory programming courses

Imma Boada, Josep Soler, Ferran Prados, Jordi Poch
Dep. Computer Science and Applied Mathematics.
University of Girona
Campus de Montilivi, 17071, Girona (Spain)
Email: imma,soler,fprados,poch@ima.udg.es

Abstract— **In this paper we present a web-based tool developed with the aim of reinforcing teaching and learning of introductory programming courses. This tool provides support for teaching and learning. From the teacher's perspective the system introduces important gains with respect to the classical teaching methodology. It reinforces lecture and laboratory sessions, makes it possible to give personalized attention to the student, assesses the degree of participation of the students and most importantly, performs a continuous assessment of the student's progress. From the student's perspective it provides a learning framework, consisting in a help environment and a correction environment, which facilitates their personal work. With this tool students are more motivated to do programming.**

I. Introduction

Over the past two decades many aspects of programming have been investigated, resulting in the development of programming environments, tools, and languages to support the process of learning programming [1]–[4]. However, despite all these advances students still find programming a difficult subject to learn. In the case of beginner programmers the basis of their difficulties is a lack of cognitive skills and knowledge required to carry out the tasks necessary to develop a program [5], [6]. Among these tasks there are: the problem of understanding, analyzing and designing a solution, learning a programming language, becoming familiar with a programming environment, etc.. For most students in introductory programming courses all these tasks are completely new. Because of this they fail to engage with programming.

It has been proven that the best way of learning to program is by programming. Students are able to learn and retain knowledge better by actively programming rather than learning passively, but how can we encourage students to do programming if most of them perceive it as difficult?

In this paper we present a web-based environment developed to support teaching and learning in introductory programming courses. The proposed tool, developed in the Computer Science and Applied Mathematics Department at the University of Girona, introduces important gains with respect to the classical teaching methodology and more importantly it motivates students to program.

On the one hand the proposed tool provides a friendly learning framework that facilitates students personal work. This framework consists in a correction and a help environment. The system, driven by teacher restrictions, assigns a set of problems selected from a common repository to each student. To solve a problem the student writes the solution in a determined programming language or in pseudo-code in a file. Then they send the file to the system and the correction mechanism corrects it and shows the errors it has detected. The student has the possibility to correct the errors and send a new solution. The student can repeat these steps as many times as required until a correct solution is obtained. The help environment allows students to practice concepts by doing example exercises which have different levels of help. The students work is recorded by the system providing teachers with a tool for performing continuous assessment. Moreover, since the teacher can consult all proposed solutions they can easily detect common errors and students' weak-points. Based on this information they can adjust the contents of the lecture sessions and the problems that are assigned to the students. An important feature of the system from the teacher's perspective is that the repository problems are available for all the teachers, allowing material to be shared. To access the system the only requirement besides authorization is an Internet connection.

The rest of the paper has been structured as follows. In Section 2 we describe our current teaching methodology and its main weak-points. In Section 3 the motivation and the main objectives of the proposed project are given. The main modules that compose the system as well as the different kind of users it supports are presented in Section 4 and 5, respectively. The evaluation of the system and experimental results are reported in Section 6. Finally, conclusions are given in Section 7.

II. Our Current Teaching Methodology

In the University of Girona the introductory programming courses are taught in different technical/engineering studies in the Polytechnic Faculty. These courses consist in lectures and laboratory sessions.

- In the lecture sessions the key concepts of programming are explained. To make programming less difficult for students in these sessions we use a pseudo-code as the first language to solve programming problems. Using pseudo-code, the algorithm can be studied independently of programming languages. Only when the students are more confident with the programming concepts do we introduce the programming language. It has to be taken into account that most of the programming languages

are English-based and not all the students speak English. Therefore using a programming language is not suitable for these sessions.
The main drawback we detected with the current methodology is that we are not able to introduce the students to as many examples as we would like to. To overcome this limitation we suggest that students solve exercises similar to the examples by using the pseudo-code. Most of the students complain about the difficulty of solving these problems. Although there are different programming environments in the laboratory computers none of them are capable of interpreting pseudo-code.

- In laboratory sessions a programming language is taught by setting small scale problems, starting with easy problems and gradually becoming more complex. It is in these sessions that students have to combine their skills to solve problems and to design algorithms and programs. Moreover they have to learn the programming language and its programming environment. In addition, students need to learn testing and debugging techniques to validate programs. All the student deficiencies come up in these sessions, requiring continuous advice and feedback. Due to the large number of students giving this personalized advice is very difficult.

The main drawbacks of this methodology can be summarized as follows. From the teacher's point of view, we don't have no enough time to give more examples, students are not motivated to work, they feel alone in front of the computer and we have no time to give them personalized attention. From the students' perspective they have no problem in solving exercises with pen-and-paper, but when they suppose that they have a solution they would like to have a tool to correct it.

III. The ACMEp Project

Conscious of the weak points and limitations of our current teaching methodology a group of teachers of our Department decided to develop a tool to overcome them and also to provide students with learning support. Since 1998 our Department has been developing the ACME project [7]. The acronym ACME stands for "Continuous Assessment and Improvement of Skills" in Catalan. The leading ideas of this project can be summarized in two main points:

- Communication through the Internet, which allows the student to communicate with the system from any computer with a standard Internet connection.
- Use of symbolic manipulation software to create and correct online the exercises of an intermediate level mathematics course in technical/engineering studies.

The whole system has been designed with a modular structure which makes it adaptable to subjects other than mathematics. Moreover, the system is showing very promising results. Motivated by the good results obtained with ACME we decided to extend the system in order to support teaching of introductory programming courses. This new system is denoted ACMEp ("Continuous Assessment and Improvement of Skills in programming")

A. Objectives of ACMEp

To develop the ACMEp system we have to take into account that the system has to be able to:

- Validate code written in pseudo-code or in any of the other programming languages(C, C++, Java, Pascal,...) taught in introductory programming courses in our University.
- Support a system for continuous assessment of the students progress.
- Make personalized attention to the students easier.
- Support an auto-learning environment

IV. ACMEp Description

In this section we give a high level overview of the main modules that compose ACMEp (see Figure 1).

A. The Repository of Problems

ACMEp maintains a repository of problems common to all the introductory programming courses. This repository is the core of the system. It allows teachers to share a set of material which is not possible for individual teachers working in parallel.

The problems of the repository bring together a set of parameters (degree, subject, level of difficulty, keywords, ...) and a set of testing values with their solutions. All these parameters are fixed by the teacher who enters the problems. The system does not establish any dependence between the problem and the programming language.

The repository also has help-exercises. These exercises, that will be described in the next section, have some help levels associated to them that guide the student to the solution.

B. Work-book Generation Module

This module generates a personalized work book for each student. A work book is composed of exercises grouped into topics. The exercises are selected from the repository following the teacher restrictions. Such restrictions correspond to the following parameters: number of problems to be assigned to each student for each topic, the course topic and the problem's level of difficulty. For example, we can select three problems from the Introduction to Data Structures topic, one with a difficulty of level 1 and the rest of level 3. Problems can be added to the work-book at any time.

C. Correction Module

This module provides an environment to correct on-line a solution designed for a given problem.

A solution to a given problem has to be written in a text file in the language fixed by the teacher which can be a programming language or pseudo-code. The text file with the solution has to be sent to the system. The system calls the appropriate compiler and afterwards it shows the compiling errors, if there are any. In the case that there are errors they can be corrected and a new solution can be sent. This process can be repeated until a solution with no compiling errors is obtained. In this moment the solution is validated using the set of testing values for the problem, showing the correct solutions and the solutions obtained with the proposed solution. In the case that there are errors a new solution can be sent. The system records all text files sent by a student.

To support pseudo-code solutions we have developed a pseudo-code compiler which generates JVM (Java Virtual Machine) code [8]. This compiler provides all the facilities to identify syntax errors easily.

D. Help Module

This module makes a model exercise similar to the exercises introduced in the lecture sessions available to the student. This exercise has three different levels of help associated to it that can be consulted by the student if they ask for help. These levels are:

- *Level 1.* An explanation of the main decisions taken to design the algorithm is shown.
- *Level 2.* The algorithm is seen as a set of independent parts. The student has the possibility to solve these parts in an independent manner instead of considering the whole algorithm. Each part can be corrected independently. There is also a set of questions which lead the student to the solution of this particular part of the solution. These questions must be answered sequentially. In each of these questions, after a number of unsuccessful trials, the correct answer is shown together with an explanation. If this is not enough the whole solution of this part is shown.
- *Level 3.* The solution of the problem is given.

The information required in each one of the help levels is introduced by the teacher who enters the problem.

To use the help environment the student accesses the system in the usual way and asks for an exercise of a particular type. Then he can try either to solve it or ask for a hint. In the second case the first level of help is activated, the system gives an explanation of the algorithm. When the student thinks he has the right answer, he enters it and the correction module corrects it. After a number of unsuccessful trials the second level of help is activated. After a number of unsuccessful trials, the correct code is shown together with an explanation and if this is not enough, the whole solution of the exercise is shown.

E. Assessment Module

All the student's work is recorded in the data base of the system. From this data base quantitative data, such as number of errors, types of errors, time taken to complete the problem, etc. can be collected by the teacher and used for different purposes. On one hand, this information can be used as a continuous assessment tool. Periodically the teacher selects a set of problems related to each topic from the repository, using the parameters associated to the problems. From these exercises the system automatically generates an exercise book that is different for each student. Students have to solve all the problems, using the correction environment, and send the solutions to the system before a fixed deadline. The students' progress through their personal exercise book provides the basis for the continuous assessment of their skills and gives valuable information about their difficulties.

On the other hand, this information can be used to guide the student through those topics which present a greater difficulty for them.

Teachers can enter the system in a matrix-like way: either they can choose a student and look at the state of their work or they can select a topic and look at its state with respect to the whole group of students.

F. Student-Teacher Communication Channel

The system establishes a virtual communication channel between the teacher of a group and all the students that compose this group. This channel can be used by the teacher as a virtual tutoring system. When the teacher consults the student's work they can detect syntax errors or design errors. In these situations the teacher has the possibility to help the student in order to correct these errors. The different ways to communicate with the student are:

- *Sending an email to the student.*
  In this case once the email is written it is sent to the student's mailbox.
- *Attaching a note to the problem.*
  In this case the teacher writes a note that can be attached to the problem. When the student enters the system and accesses this problem he will see a flag indicating that there is a teacher's note.
- *Attaching the note to all the students with the same exercise.*
  In general there is more than one student with the same exercise and with the same error. To avoid attaching one note to each one in an independent way there is the possibility to automatically attach the note to all of the student's work.
- *Writing a message in the forum.*
  The forum is a virtual space that holds all the messages written by students and teachers of the same group with comments related to the problems, or comments on different aspects of the course in general.

With this communication channel the student does not feel alone in front of the computer.

V. ACMEp Users

The ACMEp system supports three different kind of users: the system manager, the teachers and the students.

A. System Manager

The system manager performs all the typical tasks related to this role: they control access to all ACMEp applications, they are in charge of maintenance of all the system information, they authorize new users,....

B. The Teacher

The system distinguishes two categories of teachers: the teacher who is in charge of the course and the others. Each teacher is assigned to at least one course and is automatically assigned a set of students.

Once the teacher has authorization to access the system if he is in charge of the course he is authorized to:

- Introduce new problems into the repository following the introduction guidelines.
- Define the parameters to generate work-books for the students.
- Define the criteria to perform the continuous assessment.

Obviously, the teachers can consult the work of the students assigned to their groups, and have access to all the facilities in order to communicate with their students and to the correct and help modules.

C. Students

Once a student has authorization to access the system he is automatically assigned to a group. The system also assigns the student a work-book composed of a set of the exercises which are grouped according to the topics fixed by the teacher. The student has access to the help module and also to the correction module. He can access and send messages to the group's forum.

VI. Experimental Results

Currently, the system is being used in two experimental groups of students. In this section we describe the applied methodology and also the impressions and results obtained from teachers and students.

A. Applied Methodology

We have conceived the system as a a web-based tool for the reinforcement in teaching introductory programming courses and not as a tool to substitute the classical methodology described in Section 2.

The experimental introductory programming course is composed of four topics. Theoretical concepts of each topic are presented in one or two lecture sessions according to their complexity. Each lecture session is two hours long and the number of students in the class is between 60 and 70 for each group. Laboratory sessions are also grouped according to topics and they are devoted to practicing theoretical concepts by using a programming language. There are twenty students in each laboratory group, and each student has a computer. Lecture and laboratory sessions are synchronized in such a way so that first a topic is introduced and explained in the lecture sessions and then this topic is practiced in the laboratory.

The ACMEp system has been used in lecture sessions and also in laboratories in the following way:

- A personalized exercise book was assigned to each student at the end of the first lecture session. To support the lecture sessions, in each session we suggest that students solve a set of help exercises. These problems have to

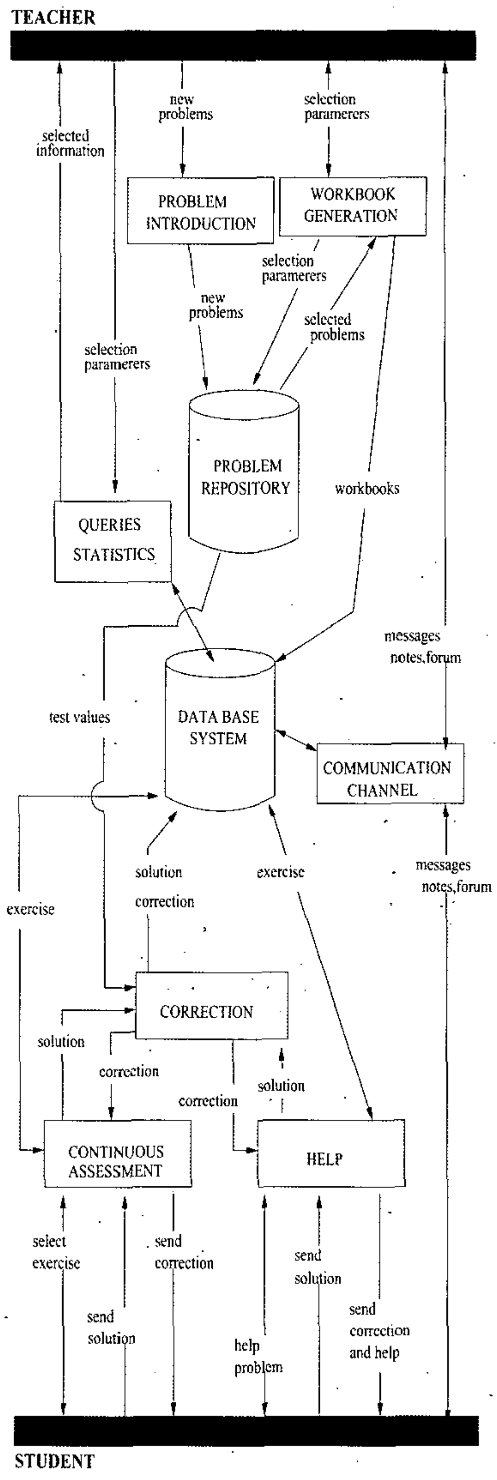

Fig. 1. Main modules of the ACMEp system. with the teacher and the students interaction.

Topic	Assigned Problems	Sent Solutions	Problem Lectures
1	3	4.3	6.5
2	3	6.2	12.3
3	2	8.1	14.6
4	2	7.2	13.8

TABLE I

STATISTICS RELATED TO COMPULSORY EXERCISES ASSIGNED AT THE END OF EACH TOPIC OF THE COURSE.

Session	Assigned Problems	Sent Solutions	Problem Lectures
1	12	2.24	3.3
2	12	2.36	3.4
3	12	2.21	3.3
4	12	2.17	3.4
5	12	2.23	3.4
6	12	2.14	3.5

TABLE II

STATISTICS RELATED TO EXERCISES COMPULSORY IN LABORATORY SESSIONS.

be solved using pseudo-code. Although the exercises are not compulsory we suggest the students solve them to reinforce theoretical concepts and also to detect possible misunderstandings.
At the end of a topic a set of exercises related to it is added to the student's personal book. These exercises are not the same for all the students. They are similar to the help exercises that have been proposed during the lecture sessions. In this case, there is no possibility of help. These exercises are compulsory and must be solved before a fixed deadline.

- ACMEp is used in all the laboratory sessions. Each student has a set of exercises assigned to them that has to be solved during the session. For each session the teacher selects a set of help exercises similar to the assigned ones. These help exercises are solved in the programming language that is being taught and are used to introduce the syntax and semantic of the programming language and also to review theoretic concepts. We spend half an hour of the two hours session on this part. Once this introductory explanation has been done students have to solve their exercises using the programming language.
  The exercises in each session are attached to their personal work- book some days before the laboratory session to give them the chance to solve them using pseudo-code. Although the possibility to solve in pseudo-code is optional we suggest that they do it.
  Since not all the students finish all the problems in the session accessing them is possible until midnight of the day of this session then access is denied. In this way we give students the possibility to correctly complete their work book. All the laboratory sessions are compulsory.

In Table I some statistics obtained from the two groups of students are reported. The total number of students is 120. Data is related to lecture sessions and it has been grouped into topics. For each topic of the course we give the number of problems that have been assigned to each student, the average number of solutions that the student proposed to the exercises and the number of lectures related to the problems completed. From the table it can be observed that the third topic is the most difficult. The same information has been collected for the laboratory sessions (see Table II). If we compare this table with Table I we can see that the number of sent solutions decreases, this is due to the fact that since students are more confident with pseudo-code the application of the programming language becomes easier.

B. Evaluation of the Tool

We have evaluated the system from the teacher and the students perspective. The results of this evaluation are reported in this section.

- Teacher Evaluation
  From the teacher's first impressions, we can remark that the environment is easy to use. It does not require any installation, only a web connection. More importantly, it provides gains with respect to the classical teaching methodology in the sense that it offers a system for the continuous assessment of the student's progress, makes personalized attention to the student easier and assesses the degree of participation of the students.
  The concentration of problems that were traditionally dissipated in parallel development of course material in the common repository is considered as one of the most attractive features of the system.

- Students Evaluation
  The students' impressions have also been positive especially the fact that to access the system they only need an Internet connection, no kind of software installation is required.
  During the different sessions students were asked to comment on the problems they faced while using the system. The responses were very positive. The students do not only learn theoretical concepts more rapidly compared with other courses, but also feel more motivated to solve the proposed problems. The possibility to correct a problem in real time encourages them to work until the correct solution is found. They also judged the communication channel with the teacher to be very positive. The fact of seeing comments attached to their incorrect solution makes them feel like they have support from the teacher all the time. They feel like they have continuous supervision and constant feedback.
  Laboratory sessions are more profitable, and most of the students usually solve the problems in pseudo-code before the laboratory session, this facilitates our work since most of their doubts are reduced to questions related to the programming language more than design problems.

An ACMEp system demonstration is available at *http://acme.udg.es/demoITHET04*

VII. Conclusions

We have presented ACMEp a web-based tool developed in the Computer Science and Applied Mathematics department at the University of Girona, for the reinforcement in teaching and learning of introductory programming courses. The key features that make the proposed system particularly attractive are:

- Communication through the Internet, which allows the teachers and the students to communicate with the system from any computer with a standard Internet connection.
- Validate code written in pseudo-code and in other programming languages used in introductory programming courses, which provides students a valuable tool for their personal work.
- Support a system for continuous assessment of the students progress.
- Make personalized attention to the students easier. The system establishes a virtual student-teaching communication channel that can be used as a virtual tutoring system.
- Support an auto-learning environment.
- Concentration of problems in a common repository, which allows the teacher share material.

This tool overcomes the main limitations of the classical teaching methodology. Currently, the system is being used in different experimental groups of students. The impressions and results obtained from teachers and students are very positive.

Acknowledgment

ACMEp is supported in part by the Ministry of Universities, Research and Information Society from the Government of Catalonia EXP.295-MQD 2002.

References

[1] F.P. Deek, "A framework for an automated problem solving and program development environment," Vol.3.N.3, Transactions of the SDPS, September 1999, pp.1-13.
[2] S. Fincher, "What are we doing when we teach programming," 29th ASEE/IEEE Frontiers in Education Conference, San Juan, Puerto Rico, November 1999, pp.12a41-5.
[3] T. Muldner and E. Shakshuki, "A New Approach to Learning Algorithms," TR-2003-02, Jodrey School of Computer Science, Acadia University, Wolfville,NS,Canda B4P 2R6, November 2003.
[4] L. Mc Iver, "Evaluating Languages and Environments for Novice Programmers," Proc. 14th Workshop of the Psycology of Programming Interest Group, J.Kuljis,L.Baldwin and R.Scoble (Eds.), June 2002, pp. 100-110.
[5] T. Jenkins, "On the difficulty of learning to program," 3rd Annual LTSN-ICS Conference, Loughborough University, 2002, pp. 53-58.
[6] P. Byrne and G. Lyons,"The Effect of Students Attributes on Success in Programming," Proc. of ITiCSE 2001, pp.49-52
[7] J. Soler, J. Poch, E. Barrabes, D. Juher and J. Ripoll. *A tool for the continuous assessment and improvement of the student's skills in a mathematics course.*Proceedings of the Symposium TICE (Technology of Information and Communication in Education for Engineering and Industry) pp. 105-110, Lyon, 2002
[8] T. Lindholm and F. Yellin *The Java Virtual Machine Specification Paperback.* Addison-Wesley Pub Co, 2nd edition (April 14, 1999) ISBN: 0201432943