An Adaptive, Collaborative Environment to Develop Good Habits in Programming

Aurora Vizcaíno[1], Juan Contreras[2], Jesús Favela[3], Manuel Prieto[1]

[1]Computer Sciences Department
Universidad de Castilla-La Mancha. Escuela Superior de Informática
Ronda de Calatrava 5, 13071 Ciudad Real, Spain
{ avizcaino, mprieto}@inf-cr.uclm.es
[2]Telematics Department
Universidad de Colima, México
juancont@ucol.mx
Computer Sciences DepartmentCICESE, México
favela@cicese.mx

Abstract. In this paper we discuss how computer supported collaborative learning (CSCL) can be deployed to develop new skills and habits in students at university level. These considerations led to the development of an adaptive environment to develop good programming habits. We start by describing the difficulties in teaching and learning programming and more concretely, in making students good programmers. Afterwards, we explain why group work is an adequate approach to learn programming. Next HabiPro, an environment that trains students in Programming is described. The principal features of this system are: It is adaptive: depending on the group features the environment proposes different pedagogic methodologies and different exercises. The tool promotes collaboration and interaction among the students. The pedagogic methodologies are based on reflection, observation, and relation. Finally, we present our conclusions and discuss future work.

1. Learning and Teaching Programming

Programming is a subject which is normally taught in the first year of a Computer Science, Computer Engineering degree, or other degrees related to information technology.

Programming is characterised by being more practical than theoretical. It is a topic that must be learnt "by doing" rather than memorising.

Many researchers indicate that the information that is received but isn't used during the learning process, is difficult to remember when we need it [8].

In a procedural topic, like programming, resolving practical exercises is even more necessary than in a declarative topic like, for instance, history**.** "Procedural learning requires theoretical learning, but not in all cases. To sum up, the application of practical or operative activities and/or the use of information is implied [2].

G. Gauthier, C. Frasson, K. VanLehn (Eds.): ITS 2000, LNCS 1839, pp. 262-271, 2000.

In procedural learning students must practice, learn from their mistakes and use abilities such as observation, reflection or relation. But students are not used to this type of learning.

Working in a group or using new technology can help students to develop skills and to learn procedures. This paper explains on what occasions collaboration can improve learning, and why group work is favourable in learning programming. Afterwards, HabiPro, an adaptive and collaborative environment to train student in programming is presented.

The contents of this paper are organised as follows: Section two explains why collaboration is useful in learning programming, section three presents HabiPro a tool to develop good habits in programming. The next sections describe the structure, adaptability, and use of HabiPro. Section seven presents results obtained from experiments carried out with HabiPro and finally we present our conclusions and future work.

2. Collaboration in Learning Programming

At our university there is a large number of students that give up or fail programming. Last year, of the 339 students who were registered, only 131 students went to the exam and of them, 90 passed. Concerned about this, we asked programming students and teachers why, in their opinion, programming has an high degree of failures.

The students' answers were very diverse, but all of them agreed that they only had a few hours of laboratory practice. They also agreed that when they were at home it was more difficult to do the exercises because when they made a mistake they didn't know how to continue because the books didn't have the answers.

On the other hand, the teachers agreed that programming is a very abstract topic and on many occasions they don't know how to explain topics like recursion or data structures.

Another reason why teachers think students don't write good programs is because students do not think about the possible solutions to the problem, and which of them is the best. Pupils try to solve the problem as quickly as possible without thinking if her/his solution is good.

When we studied the answers we thought that a distributed and collaborative tool could be a solution to several of the presented problems so we decided to develop HabiPro. The current methods of teaching programming include lecturing, and laboratory sessions in which the students reinforce what they have learned in lectures by developing small programs of their own. A further way of assisting students is to provide computerised aids that are designed specifically for the novice [9].

Computer networks permit students to be connected, although they are in different places, so a distributed environment permits students to work in a group and solve exercises from their homes.

Collaborative learning has many advantages such as the interchange of ideas among the students, or an increase of the motivation to learn. There is a substantial body of empirical evidence demonstrating the positive effects of social interaction for learning [11], [10], [6]. As the results of these studies emerged, it became apparent that computer use can constitute a particularly valuable context for social interaction [3].

So currently there are many learning programs based on collaborative learning. But is collaborative learning good in all situations?

When people begin to learn programming, apart from buying a book about the language that they are using, on many occasions they also attend courses and if they have access to Internet, they join message lists, or news and work groups where they can ask questions or advice from other programmers.

So the students look for the experience and collaboration of other people. This shows that in programming learning collaborative techniques are often used because students join together to write programs and to take advice about their doubts in a spontaneous and natural way.

3. What is HabiPro?

HabiPro (Habits of Programming) is a pedagogical and collaborative software designed to develop good programming habits. It doesn't try to teach programming but to develop in the novice students skills such as observation, reflection or structure, which are necessary to become good programmers.

The interface of the application has two windows (Fig. 1). One of them is a chat window that permits communication among students. And the second one is a shared window (work window) where students must collaboratively solve a problem.

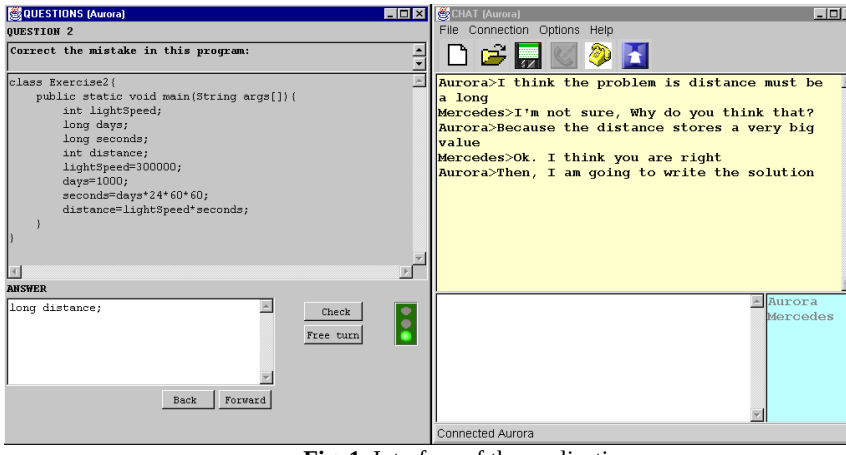

Fig. 1. Interface of the application

The work window can present four types of exercises:

- Finding the mistake: Students must find a mistake in a program. All programmers at some time must find the mistake because the program doesn't work. It is convenient for novice students to be in the habit of thinking and predicting which is the mistake or mistakes that don't permit the program to work correctly. In these types of exercises the most frequent mistakes made in Java programming are shown.

- Put a program in the correct order. With this exercise we try to help students to learn to structure a program and reflect on the order in which the sentences should be.
  To solve the exercises the screen is divided into two parts. In the first one the disarranged program appears. When the students choose a sentence it is automatically put into the second window. At the end, students check if they have ordered the program correctly. In this case, the program is presented with comments and indenteds. The visualisation of a perfectly ordered program helps the students to realise that the presentation of a program aids a better understanding of what the program does.
- Predicting the result: This exercise also attempts to show the importance of creating programs which are easy to understand (with comments and with significant names of variables) even for other people. At the beginning, programs without comments and variables with random names will be shown to the students and they must guess what the program does. Next, a similar program will be shown but in this case the program will have adequate comments and significant names of variables so that students can see that in the second case it is easier to trace a program.
- Completing a program: Students must write one sentence that is omitted. In this exercise we try to make sure that there are different solutions. The system only accepts the best solution. So students learn that normally a problem can be solved using different techniques but in many cases one solution is better than the rest. They must try to find this solution. For instance, the system shows a program where students must declare a variable. It is possible to use an integer but perhaps only a byte is really necessary, so students can learn it is important to save memory when possible.

When a group proposes a solution, if it is incorrect the system shows four types of 'help'. This assistance has two goals: first, to help the group to find the solution, and second, to obtain information about the group. The following explains how HabiPro achieves both goals.

The first type of assistance gives clues to the student about how they can solve the problem. So they must think about and reflect on the solution. This type of help is situated at the beginning to try to influence students to choose it (first button).

The second help button shows the solution and an explanation of it because the problem is solved with that technique.

The third one shows a similar example of the problem that students have to solve, and its solution. In this case students must use the techniques of comparing and observation to detect which part of the example looks like their problem and relate the solution in the example to their own problem.

The last one shows only the solution to the problem.

The choice of the type of help gives information about the group motivation and their willingness to learn. If a group frequently chooses the help button that only shows the solution, this indicates that their motivation is very poor because they aren't interested in knowing why that is the correct solution. On the other hand if other groups prefer to use clues or compare the example with the problem this indicates that the group likes thinking about the solution and that the group is active. If a group prefers to see the solution with an explanation it is because the students are

interested in knowing "the reason why things are as they are". In this way, the solution is not only a tool of assistance for the group, but also for the system itself. The system is able to detect each group's characteristics depending on the type of assistance chosen.

4. Structure of the System

HabiPro has a client-server architecture (Fig. 2). The client and the server are divided into two parts. Those of the client are formed by:

- Interface: A chat window where students write possible solutions to the problem. A shared work window of the type WYSIWIS (What You See Is What I See). And a small window where the information about the other users appears are the windows that the system presents to the users.
- A collaborative component, which gives collaboration awareness to the students. This part is very important in non-presence collaboration where students lack some advantages like eye contact or knowing the companion's emotional state by her/his tone of voice. For this reason a window is added indicating how many students are connected, their names and if it is possible, their photo. It also shows the frequency with which each person takes part in the conversation in order for the students to know who they are working with and which people are participating most.
- Collaboration awareness and emotional state during the collaboration process are important topics that are currently being researched. Systems to detect the emotional state of companions are currently being designed [5].
- Student memory: In this part all actions performed by the student are stored. Part of this information will be sent to the server and the rest will be used to produce statistics about the progress and the participation of each student.
- Communication component: This allows the sending of messages to the server and the reception of messages from the server.

The server architecture is composed of three parts:

- The communication part: This permits a group of students to be connected, and to be able to send and receive messages. The main functions of these components are:
  - Connection: Each time the server detects that a client wants to connect, the server has to create a new communications channel and to tell the connected clients that a new client is connected.
  - Disconnection: When a client abandons the application the server informs the rest of the clients who has disconnected.Send control events and messages to the clients.
- Specific database: The database stores the exercises, solutions and aids (clues, explanations, and examples) that HabiPro proposes to the group. If we want to

use the application to teach other topics, it is necessary to change the information in the specific database for the new information related to the new topic.

- Group model: This is an essential part of the system. "A model of cooperative problem-solving should predict what forms of cooperation can exist and ideally what interactive learning mechanisms they trigger" [1].
  The group model is based on the model proposed by Ana Paiva [7]. The more we know about someone, the more we know his/her necessities and preferences. In the same way, the more information the group model has about the people with whom it interacts, the more precise it is in determining what exercises are needed.
  The group model represents and characterises the group. The group model may include two types of information: One of them relates to pedagogical aspects, and the second relates to social aspects. This is because when a group is learning, apart from knowing what knowledge the group has, it is important to also know the characteristics and preferences of the group.
  The information stored in the group model will be used to adapt the system to the group.
  In relation to pedagogical aspects, the group model stores theses points:

  - What abilities the group has. This information is obtained from the exercises done by the group.
  - What type of exercises the group prefers. After each exercise each student must indicate, his or her opinion about the degree of difficulty of the exercises and if s/he thinks the exercise is interesting, boring, etc, by filling in a brief questionnaire.
  - What mistakes the group has made. These are obtained from the mistakes made in solving the exercises.

In relation to social aspects, it is important to know the degree of motivation of the students and with what frequency each person collaborates:

- Motivation: The system has four different types of help (explained in the second point). Depending on the type of help chosen, HabiPro can suppose the degree of group motivation.
- Participation: Using the conversation in the chat window HabiPro checks how many times each student takes part in the conversation. If the same student always takes part perhaps there was a leader of the group, or maybe the rest of the group are passive people. This is important information if the system is to succeed in allowing everybody to take part in solving a problem.

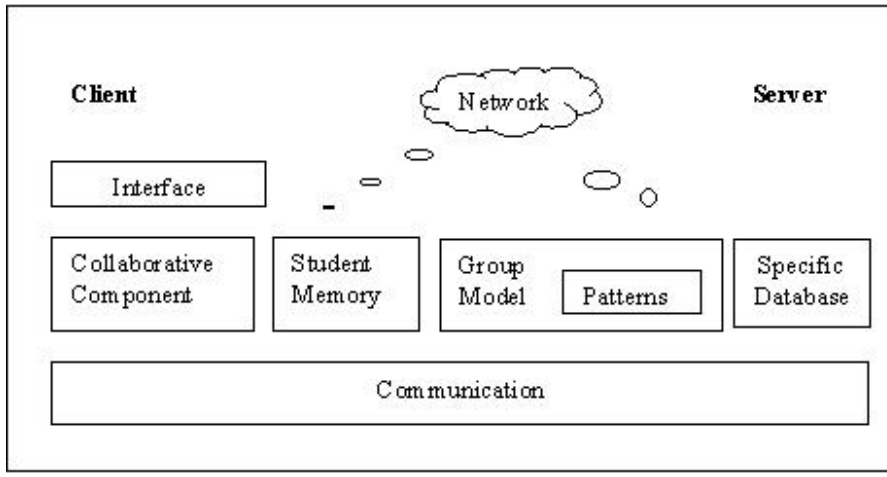

Fig. 2. Architecture of the HabiPro

5. Why HabiPro is Adaptive?

The group model has stored different social and pedagogic patterns. Each pattern has characteristics that describe behaviours. The patterns also contain a list of exercises, types of clues and pedagogic techniques.

While the group is working with HabiPro the group model compares the new information with the characteristics in the patterns and tries to classify the group in a pattern. Once the group is classified, the pattern indicates which exercises and work methodology is the most adequate for a group with a type of specific behaviour. For instance, a group could have a pedagogic pattern with the follow characteristics:

- Mistakes: The group can't find the mistakes in the problems.
- Preferences: Exercises that involve filling in the program.
- Frequent Solution chosen: Solution without explanation.

For this type of pattern the group model proposes:

⇒ To show low level exercises.

⇒ Not allowing the group to choose help without explanation with out letting a certain amount of time elapse.

⇒ To show some "finding mistakes exercises" adding some clues in order to aid students to find the mistakes.

And the same group could have a social pattern with these characteristics:

- Participation: Only one or two people take part.
- Motivation: Low

This pattern proposes five activities that HabiPro can use to increase motivation and participation:

⇒ HabiPro activates the rotator turn system, so that all students must take part in the work.
⇒ Presenting more attractive exercises.
⇒ The system produces personality questions, indicating the name of the person who must answer.
⇒ The system gives points to the person who writes the correct solution (like a game).
⇒ By showing statistics, which demonstrate the performance of other groups that worked with is system, thus the group can compare this with their own performance.

The group model in this case is not only the representation of the characteristics of the group, it is also the component that permits the system to be adaptive.

6. Evaluating HabiPro

Several versions of HabiPro have been implemented and tested. Experimenting with these versions has allowed us to establish the more adequate number of people that can work using HabiPro, and observe what social protocols are used by the students in order to come to an agreement, or give solutions.

Firstly, a version with free floor protocol was tested. Twenty-three students divided into different sizes of groups (in order to research in which groups collaboration was better) had to solve twenty exercises. Each student used a computer, and they interchanged ideas via the chat.

When students agreed about a solution one of them had to write the solution in the answer window and if the solution was correct, the system automatically showed all group students the next exercise.

This version presented a problem: Some students wrote the solution without consulting with their colleagues, so if the answer was correct all the students advanced to the next exercise, although some of the group members did not understand why the solution was as it was**.** This fact produced a feeling of anger and frustration in the other members of the group. The same students proposed a solution to the problem: The system should only accept a solution when it was proposed by all the students (each student has to write the solution in the answer window). Currently, a version with this feature is being implemented and we hope to test it soon.

The second version has a turn protocol. Only the student whose turn it is can write the solution in the answer window. A student can take a turn until another student asks for it. The turn protocol facilitated communication because students did not have to spend time deciding who would write the answer.

In this version, the interface shows a traffic light that tells the student if she/he has a turn (green light) or if she/he hasn't it (read light). So, the person who has the green light knows that he has to write the answer in the adequate window.

We could prove that the turn protocol is also convenient for collaboration, because if one student did not collaborate his/her companion could give a turn to this person and in this way motivate his/her collaboration.

The experiments were done with groups of different sizes: five groups formed of two students, three groups of three and one group with four people.

When students finished solving the exercises they filled in a form where they answered some questions and they could also express opinions about the experience.

The group with four companions agreed that it was very difficult to work with so many people, and in many cases no one took part in the activities. On the other hand, the groups with three members solved less exercise that the groups with two members, this could be because the groups with three people spent more time in communication, and the negotiation was more difficult than in groups with two members.

We can conclude that HabiPro woks efficiently with groups with two members, or a maximum of three. With more people collaboration decreased and communication was more difficult.

We can also observe that adding a turn protocol increases the performance and facilitates collaboration.

7. Conclusions

Collaborative learning has many advantages but before applying collaboration in a topic we must study whether the topic is really adequate to be handled in a collaborative environment.

In this paper we have explained why programming is a field which can be learned by using collaborative techniques.

HabiPro, a tool for developing good programming habits in the novice students has been presented and its main characteristics such as adaptive techniques and methods to teach the student to think have been explained.

Creating adaptive collaborative systems is more difficult than creating individual adaptive systems because apart from the pedagogical aspects, we must also take into account aspects related to social relations and group dynamics.

8. Future Work

When a set of people work or study in a group a figure usually appears who influences in the group, sometimes voluntarily and other times involuntarily. This figure is the leader, who can influence the group positively or negatively. If we can insure that the leader's influence is good we can help learning to be more efficient.

We want to add a virtual student in HabiPro. This virtual agent must collaborate with the group, and perhaps on some occasions it could carry out the roll of leader.

The virtual student is a software student who can control communication. For instance when a person doesn't take part in the conversation, the virtual student can say to him/her: "What do you think about this?". Also the software student can

propose correct answers when the rest of the members in the group have proposed wrong ideas.

The student's virtual knowledge could be similar to the knowledge of a teacher or an expert, but it is better for learning if students work with a student although it is "virtual" one than with a teacher. This is because when somebody tries to collaborate with somebody who has a higher status (a boss, teacher...), it is not collaboration but obligation, so the motivation is less than in the case where the members of a group belong to the same status. Collaboration is more likely to occur between people of a similar status than between a boss and his/her employee, or between a teacher and a pupil [4].

Currently we are carrying out an experiment that will allow us to know and limit the language that students use in the communication process in order to know the discourse universe that the virtual student must have.

References

1. Baker, M. "The roles of models in Artificial Intelligence and Education Research: a prospective view". International journal of Artificial Intelligence in Education, 1999.
2. Castañeda Yánez, M. 1995. "Análisis del aprendizaje de conceptos y procedimientos". Editorial Trillas.
3. Crook, C. "Computers and the Collaborative Experience of Learning". London: Routledge.
4. Dillenbourg, P.; "Introduction: What Do You Mean By Collaborative Learning?". In Collaborative Learning. Cognitive and Computational Approaches. Edited by Pierre Dillenbourg. Elservier Science, 1999.
5. García, O.; Favela, J.; Machorro, R. "Emotional Awareness in Collaborative Systems". In Proceedings 5th International Workshop on GroupWare. Cancún, Mexico, September, 1999.
6. Light, P.; Littleton, K.; Messer, D.; Joiner, R. "Social and communicative processes in computer-based problem solving". European Journal of Psychology of Education, 9 (1), 93-109. 1994.
7. Paiva, A. "Learner Modelling for Collaborative learning Environments". In Boulay, B., Miyoguchi, R.(Eds.). Artificial Intelligence in Education. pp 215-222. IOS Press.Pg 215-222, 1997.
8. Schank, R.; Kass, A. "A Goal-Based Scenario for High School Students". Communications of the ACM. Vol 39, Nº 4. 1996.
9. Smith, P.A.;Webb G. F. " Evaluation of Low-Level Program Visialization for Teaching Novice C Programmers". In proceedings of ICCE'99, 7th International Conference on Computers in Education. Chiba, Japan, November, 1999.
10. Teasley, S.; Roschelle, J. "Constructing a joint problem space: The computer as a tool for sharing knowledge". In Computers as Cognitive Tools (pp 229-257). S. P. Lajoie & S.J. Derry (Eds.). Hillsdale, NJ: Lawrence Erlbaum Associates. 1993.
11. Tudge, J.; Rogoff, B. "Peer influences on cognitive development: Piagetian and Vigotskian perspectives". In Interaction in Human Development (pp. 17-40). M.H. Bornstein & J. S. Bruner (Eds.). Hillsdale, MJ: Lawrence Erbaum Associates. 1989.