Computer Supported Co-Operative Systems to Support the Problem Solving - A Case Study of Learning Computer Programming

Crediné Silva de Menezes, Orivaldo de Lira Tavares, Rosane Aragón de Nevado and Davidson Cury
credine@inf.ufes.br, tavares@inf.ufes.br, rosane.aragon@ufrgs.br, dede@inf.ufes.br

Abstract – Nowadays it is widely accepted that the teaching of computer science and engineering reaches better results when it focuses on problem solving. However, the adoption of this pedagogical approach is often hampered by the limitations of the learning environment in a typical classroom. In this work, we present a pedagogic proposal supported by a digital environment to power and to facilitate the cooperative authoring.

Index Terms – problem solving, learning of computer programming, digital environment, cooperative authoring, CSCL.

Presentation of the problem

The initial learning of programming in computer science is considered a very subjective process. Graduate students tend to see this process as a real test of their vocation. Students having little or no success in this process usually abandon programming and start looking for other possibilities, sometimes even switching to other careers. This turns the activity of introductory classes a strategic issue.

Teacher makes every effort to teach the students the syntactic and semantic aspects of the programming language to be learned. In order to facilitate the construction of correct programs to the exercises given to students, many teachers use script or program examples having similar structures to those to be constructed during these initial attempts in learning programming. This is based on the method of making the students create, in their memoirs, standard structures that can be used to solve different types of problems.

Experiments of the psychology of learning indicate that ownership of new concepts occurs in a faster and deeper way, when the subjects are concrete manipulation of objects of knowledge. This theoretical foundation can be materialized in computer laboratories, where students participate in sessions to produce solutions to problems proposed.

Therefore, the teaching of engineering and computer science reaches better results when it focuses on problem solving. However, the adoption of this pedagogical approach is often hampered by the limitations of the learning environment in a typical classroom. It is common that teacher, unwittingly, prevents the student from understanding the process of problem solving and thus, from the process of developing computer programs.

The objective of this article is to present a pedagogical proposal based on digital environment to enhance and facilitate cooperation in the development of programs (CSPL – Computer Supported Programming Learning), with the aim of graduating good computer programmers. The use of information technologies communication to support the teaching and learning of engineering and technology courses is defended by several authors [3].

Psychological Premisses

The constructivist conception of Psychology has been fertile in explanations for the genesis of the general categories of knowledge (space, time, causality, etc.), dealing in a original manner the issue of genesis and of conditions and possibilities for human knowledge. Besides creating models for explaining how the mental structures work, genetic epistemology look for explaining from what paths a subject builds the structures responsible by the act of knowing. To Piaget [7], it is possible to draw two distinct plans that cross and explain themselves: the history of scientific thought and the experimental study of cognitive development, considered since its genesis.

According to Inhelder & Cellérier [8], genetic epistemology has defined a general architecture for knowledge, tackling the knowledge’s subject according to an structural analysis (epistemic subject), whereas left open the study of a huge domain of cognitive conducts nearer to heuristics, involving rules and methods that lead to exploration and discovery (functional aspect). These conducts show procedures in which their construction area done in practical contexts and allow to see how the subjects give meaning for a specific task or activity, how they choose tools, how they take decisions and control suitability of their action approaches. These aspects add to the formation of cognitive tools that transform themselves in objects-to-think-with.

In this sense, we start from the hypothesis that Learning Projects, once allow the subject to tackle problems within their “assimilation zone” instead of be confronted with a strongly structured situation, will support learning through both invention and application of several procedures. It is a reasonable assumption that to invent a procedure, one has to assimilate new objects from the world to their schema and as a result, to assign new meanings. Considering that the schema is a cognitive tool for assimilation and that its function is to turn in cognition data from experience, that is, innovation from procedures contribute for cognition, allowing both practical
considered in the specification of CSPL:

The developing of a Computer program is considered here a process of solving problems. From an adaptation of the schemas produced are specified in terms of the context in a way to make themselves suitable, although epistemic enrichment and heuristic value are tied to ability of these schemas to be generalized to other situations.

The process of solving problems in CSPL

The developing of a Computer program is considered here a process of solving problems. From an adaptation of the ideas of [1], the following stages of problem solving were considered in the specification of CSPL:

Step 1 - Understanding the problem - defining the problem needing the construction of a computer program. At that stage, it is important to ask:
1. What is the data input?
2. What is the desired result?
3. What is the relationship between the data input and the result being produced?
4. What are the important properties that data entry has?

Step 2 - Planning –
- Identification of methods and / or functions that could be made known to get the desired solution;
- Development of one or more options for the solution of the problem;
- Preparation of a plan to test the program, with a list of organized test cases, including data entry and expected output.

Step 3 - Development – coding and testing the alternative solutions in a programming language, testing of that solution. This stage can be sub-divided in the following steps:
- Implementation of the solution: after the planning of the solution;
- Testing of the solution: the analysis of the behavior of the program, according to the test plan, helps to find the points of imperfection of the solution and facilitates its correction.

Step 4 - Evaluation of the process and its results - reflection on each step of the process of building programs, in order to consolidate the strengths and correct the weaknesses observed in each of these steps. Compare the solutions achieved with respect to various aspects including performance. To each one solution the following questions can be asked at this stage:
1. Is this the best way?
2. Does this solution obtain components reusable in the future?

Step 5 - Socialization of the results - this step is important in any learning environment. It exposes the student ideas to others in the class. Each student can have an important experience disclosing his/her ideas to the entire class. At the same time, to see the suggestions made by other colleagues, and make suggestions to the work of those others, is an important strategy to enhance learning (peer review).

Assumptions used in the specification of CSPL

The pedagogical approach adopted and the computer environment are based on the following assumptions:

- Students will be solving programming problems - he/she should attempt to solve the presented problems alone, before asking for help from the teacher or from a fellow student.
- Pro-activity in the elicitation of the problem - in assisting a student the teacher should never offer a pure and complete presentation of the solution. Assistance must be given through clues that help the student to find the solution by him/herself.
- To plan the solution is to pave the way for success. In the search for the solution to a problem, the main tool being used is the question. The student must seek to formulate questions that will help him/her to understand the problem and develop a solution, such as: Have we already solved this problem before, with data slightly modified? Have we already solved a problem like this? What is the relationship between this problem and another known problem? Can the problem be divided into smaller problems? Can this problem be generalized in a manner to reach another known problem? Do we know a similar problem we can generalize to approach this problem?
- Become expert in solving problems. Each student must be encouraged to improve his/her method of solving problems through the definition of systematization suitable for the process. It is important for the student to learn to associate the current problem with other problems already solved. This way, an individual solution can be built and improved every time he/she solves a new problem.
- One is not enough, two is good and three is better. The student should be encouraged not to be satisfied with a single solution. The more solutions found, the more skillful he/she will be [2]. Furthermore, he/she can compare the alternatives and be able to choose the most appropriate. The choice should always be based on objective criteria such as efficiency of the code, reuse, ease of maintaining the code, and so on.
- Practice leads to perfection. In order to learn a language, even artificial, it is necessary to practice it intensely.
- The computer helps to test hypotheses and implement ideas. The feedback on the correction of each program, in the case of programming languages, is facilitated by the use of a computer and appropriate
programming environment, in a cyclical process of reflection-action-action, or planning-preparation-test.

It is important to improve the learning process. A crucial part of this is to record the whole process used to arrive at solutions to each problem, that is, the path to build the programs to be tested. This record shall include: 1) the test plan, the interactions with the teacher, the monitor, and with fellow students, both in the same group and in others, 2) all tests performed in the programming environment, both the successful and unsuccessful ones, and 3) the conclusions of the group on the solutions developed and tested.

The development of projects proposed by the subject itself will strengthen the motivation and encourages the formation of autonomous individuals [9].

**Methodology used in the elicitation of the requirements of CSPL**

In the following, we describe some features and adopt the methodology for a class of Introduction to Programming that served as a stage for the elicitation of the requirements of CSPL. The time of 4 hours per week allotted to the class was divided into 2 hours of classroom instruction and 2 hours of instruction in the laboratory. Classroom instruction was conducted in person by a teacher presenting each topic of the program of the course.

The laboratory was used to solving problems. A list of problems to be solved in groups of up to three students was presented at the beginning of each session. These groups were defined in classroom and remained the same over the duration of class.

In these activities we used the learning based on the resolution of problems [5] to support students to transform the experience of the classroom into personal knowledge.

Each group could interact with the teacher, through a chat. The interactions sought by teachers could be grouped into: clarification on the statement of the problem and attempts to validate the solution. With respect to the second group, our intention is to carry out a search where they exploit the prior knowledge of the student and help them in making a bridge for drafting the new solution. In these dialogues is possible to talk about similar easier problems, to suggest exploitation of other situations, and also to suggest alternative ways so.

However, it is very common the arising of other types of interaction. In the haste to complete the list of problems, students who participate in this type of session for the first time, try to ask questions about how to solve, hoping for more direct answers from the teacher, because usually they are much more concerned about ratings then exactly about learning.

Another very often type of interaction is related to the syntactic knowledge of the programming language, and to the types of errors reported by the environment. With respect to such issues, we have chosen to encourage the student to seek support in online manuals and other suitable materials. That way it is possible to solve those questions without, however, to answer the main question directly. This attitude contributes to the development of autonomy. With respect to direct search for answers, we show the students, not always with great acceptance of them, that to show the direct path will not be convenient for their intellectual growth.

A website was created (Figure 1) with information on the program of the course, methodology used, criteria for assessment of students etc. The lists of the exercises to be solved by the groups of students were published on this site. Each group set up a website in pbwiki [9] to publish the reports of all activities in the class developed by the group.

The report of the activities developed in the laboratory was published in a page of the site of the group at the end of the session. Importantly, this report included the plan to test the interactions with the teacher responsible for the laboratory, and all sessions of testing on the programming environment HUGS. The report also included unsuccessful sessions, so that the complete journey of learning of the group was recorded. During the interval (a week) until the next laboratory session, each group could improve or correct the solutions arrived at in the laboratory by posting a new version of the report.

During the laboratory classes, the groups interacted with the teacher via the chat gmail. This feature provided an interface allowing the teacher to see immediately, when a group required attention. This proved out to be very appropriate for the process.

Virtual forums for the purposes of peer review were envisioned. Here members of one group would examine the laboratory sessions of another group.

Each group developed a programming project on a theme set by the group. The pedagogical model of Learning Project [6] was adopted. This consists of the students forming groups to develop a program of medium size in the course of the semester. This approach differs radically from traditional educational projects because in it the student defines the problem and solves it, not the teacher. In this approach the teacher does not “give” the knowledge but he works like a facilitator in the learning process. Thus, the student is also an author of his productions, building, at the same time, his/her autonomy.

At the end of the academic period, interviews were conducted between each student and one of the teachers of the class for a joint evaluation of the activities developed during the project. Table 1 presents the basic questions answered by students individually during the interview in the Programming Project.
III-METHODOLOGY
1. Classroom lessons.
2. Problem solving in lab. In each session will be presented a battery of problems to be solved in groups. All the works will then be published for members of the class at the end of the lesson. These exercises can be improved and re-presented until the next lesson.
3. Virtual forums for debate. The group works developed in the laboratory sessions will be examined by members of other groups, in a process of scientific peer review.
4. Lecture of articles and debates. Two articles will be read and debated during the development of the discipline. Debates will be virtual. At the end of each debate students will be invited to produce synthesis and to write articles of basic scientific research level.
5. Development of a programming project. Students will form groups to develop a small size program during the semester.
6. Workshop for problem resolution. The programming projects developed by the groups will be presented and debated in a special session.

IV-ASSESSMENT
1. Problem solving in groups.
2. Written individual evaluations.
3. Individual problem solving (in lab).
4. Programming project in groups.
5. Individual assessment.

Figure 2 presents the structure of site produced at the end of the semester.

The students' performance
The students participated in an intense experience for resolving problems. At the same time they had many opportunities to learn and to consolidate their knowledge of the specifications of programs in functional programming. They complained about the very short time given to process very large lists of problems. In addition, they complained about the two individual tests of programming they underwent in the laboratory.

Furthermore, the students complained about the poor interaction with the teacher responsible for the groups in the laboratory. This was due to very poor network infrastructure of the laboratory and to the extreme difficulties resulting from having only one teacher instructing 14 groups simultaneously. Moreover, being accustomed to receive direct replies to their questions from teachers at high school, some of the students complained about the "evasive answers" of the laboratory teacher. Those "evasive answers" were intended to make the students think about the problems, to open them to a thinking process that eventually would enable them, by themselves, to resolve the problems.

Preliminary Specification of CSPL

During participation in didactic activities of the teaching-learning of computer programming found it necessity to use interaction tools and areas for publication for the development of these activities. After thoughtful consideration of this necessity, we succeeded in identifying the key elements for an environment dedicated to the creation of practical programming class utilizing the process of solving problems presented in this paper.

In the following subsections, we present the specification for an environment to support the cooperative learning of programming (CSPL) with computer resources appropriate to the methodology described in the previous section.

An Environment for cataloging of users and disciplines

An environment to support that pedagogic proposal should provide educational facilities for the following activities for resolution of problems and the development of learning projects: a) problem elicitation, b) planning of the solution, c) implementation of the solution, d) preparation of a test plan of the proposed problem solution, with a list of test cases, including data input and the corresponding results, e) the test of the solution, f) evaluation of alternatives, g) dissemination of results. In addition, the CSPL provides the following functions: a) interaction between the teacher and the students as well as between students, which is essential to the understanding of problem, b) discussion forums c) calendar of activities d) editing of documents, e) editing and publication of web pages, f) web search g) editing and formalization of the solutions h) simulation of the solutions (i.e. interpretation of programs); i) shared editing, publication and exchange of files.

In the following, we describe an environment that has the following facilities:
Cataloging of users (teachers, students, coaches, employees etc.);
Cataloging of disciplines;
Association between users and disciplines;
Publication of libraries in support of a particular discipline with possible association with a given session of classroom practice.

**Editor for Practical Sessions**

A recourse for the creation of practical classes that entails:
- The insertion and publication of problems in the list of problems in a classroom laboratory;
- Linking of bibliographic references to the problem.

**Reporting of Activities and socialization of the results**

This is a recourse to record and publish all activities performed by the user. This feature creates the report of each session held in CSPL.

The CSPL needs resources to record and publish all activities undertaken by the student individually or by his/her group, while building a program that solves every problem proposed in the laboratory by the group.

With this feature, the CSPL creates a site for each user and registered group, where all records of the sessions conducted by the user or the group are published. Thus, at the end of each session using the CSPL, a report of all activities is published in the corresponding site without requiring extra effort of the student or the group.

In this environment there should be support for the debate between the groups associated with the solutions presented for the proposed problems. Moreover, it is in this environment that is materialized the workshop for the learning socialization.

**Learning Project Editor**

For the exposure of the development of the learning projects, each group must build a site where the results of their research will be published as the group progresses. This site should provide facilities for the post of feedback from peers and from the team teaching.

**Support Tools for Understanding the Problem**

(Elicitation)

The following tools are needed to support the understanding of the problem:

1. Chat

Every problem must be associated with chats, which may begin synchronous and later become asynchronous. These chats can be:
- A chat for individual conversations with the teacher, monitor or with the members of the class.
- A chat for the group including the teacher, the monitor or other groups.

These chats should be conducted in such a manner that the teacher immediately knows when a student or a group needs his/her attention.

2. Mural of chat

Associated with every problem there is a Mural where messages are published, grouped by similarity, and posted in the chat rooms of the associated problem.

3. Forum

Every problem should be linked to a forum for individual or group conversations including the teacher, the monitor or other members of the class. This forum must easily allow users to identify the new messages posted on the forum.

4. Mural theme of the forum

Each forum has an associated mural, where messages are posted on the forum grouped by affiliation of the issues.

5. Agenda

The agenda comprises the registration and publication of the commitments of individuals or of groups.

**Final considerations**

This article has presented important issues to be addressed in the disciplines of introductory programming. With an approach based on the problem solving and in learning projects, it presented a proposal for a methodology and the specification for a virtual environment to support the programming of the process of teaching-learning. Practical aspects of teaching experience requirements were submitted for the elicitation of the virtual environment (2) presented to support the process of teaching-learning.

In general, the tools of computer-supported environment are useful to support the learning of initial programming. The choice of these tools is based on a methodology that emphasized relevant aspects in the training of capable programmers, such as understanding of the problem, planning of the tests, preparation of various alternatives for the solution of the problem, working in teams, and systematic reporting of the activities developed.

We do believe that the ideas and tools presented here are a major contribution in the pursuit of improving the learning of computer programming, mainly because it brought out the initial form of CSPL.

An experience with the pedagogical approach of learning project was conducted with the classes of introduction to the computer programming in computer science and engineering courses, using a preliminary version of the CSPL.

**Acknowledgement**

We would like to thank Markku Kari (markku@karinmake.com) for spending his precious time helping us with the correct use of the English language.

**References**


Author Information
Crediné Silva de Menezes, Associate Professor, Federal University of Espirito Santo, credine@inf.ufes.br, Member of Brazilian Computer Society.

Orivaldo de Lira Tavares, Associate Professor, Federal University of Espirito Santo, tavares@inf.ufes.br, Member of Brazilian Computer Society.

Rosane Aragón de Nevado, Associate Professor, Federal University of Rio Grande do Sul, rosane.aragon@ufrgs.br. Member of Brazilian Computer Society.

Davidson Cury, Associate Professor, Federal University of Espirito Santo, dede@inf.ufes.br Member of Brazilian Computer Society.