

Supporting teaching/learning activities using an environment based on virtual tools^{*}

Lucia GIRAFFA, Sabrina MARCZAKⁱ, Gláucio ALMEIDA
Pontifícia Universidade Católica do Rio Grande do Sul – PUCRS
Computer Science School – Computer Science Post Graduate Course
Ipiranga Avenue, 6681 – Block 16 – Porto Alegre – RS – Brazil 90610-900
{giraffa, smarczak, ac107026}@inf.pucrs.br

Abstract. The aim of this paper is to describe the designed process of an educational environment. This environment intends to support teaching/learning activities related to Algorithms and Programming domain. The software was modelled based on a framework supported by virtual tools. It also considers a methodology based on our previous experiences with Virtual Classes (VC). The association of such experiences with VC mixed with traditional classes' methods allows us to enrich and to organise the learning activities. The set of virtual tools must be considered as resources that teachers and students can use to exchange information and to build knowledge about the domain. We also discuss some aspects related to Algorithms and Programming teaching process. We do not intend to show experimental results. We claim that we can build educational environments taking advantages from agents' technology, and a mixed methodology based on presential and virtual classes. We believe it is not possible to create good and helpful educational software without those premises.

Introduction

In real life, in order to solve problems in any activities we need to have a plan. Building a plan implies the selection of cognitive skills associated with the reasoning process [1]. If we consider a computer as a tool to solve problems we must have in mind the classes of problems that we can handle. The set of problems that we can solve using a computer as a tool depends on the mathematical model that allows us to implement the solution in a language, and a structure in accordance to the computer architecture. Anyone who had been faced with the hard task of programming a computer for the first time realised that it is not an easy task. It is difficult to understand immediately the features and the particular set of mental abilities needed to build a specific computerised solution. The reasoning process associated to the solution process starts with the identification of the problem nature, its features, the available data, the output expected, the complementary data and information needed to be obtained, and the pre-requisites associated with the knowledge of the domain. It is natural that if we do not know the concepts related to the topics, and the related formula, it becomes impossible to model a solution. At first, it seems silly and obvious for the expert (senior programmer). However, it is not so clear for the beginners. The ability to separate the parts of the problem, the domain characteristics, the available data, the expected results, and the process to obtain them, is not so simple for those who never had such experience before. It increases much more when you have to think using a reduction model like Input → Process → Output (IPO) metaphor. We based our programming

^{*} Research supported by Sesu/MEC and Microsoft XML Center.

ⁱ Research also partially sponsored by the Dell/PUCRS Agreement, through the Brazilian Federal Law for Information Technology (Law n. 8.248/91).

activities using this IPO scheme. It is simple if you compare with our natural human information process.

Algorithms and Programming teaching process need to create opportunities for students to understand the way they solve problems using a computer. The IPO scheme is a metaphor to understand the way the computer processes the information. However, it is not natural way for human beings. We do not stop a reasoning process and think: "Ok, this is the data, and I want to achieve that. So, I need to do this, that, and so on". It happens in a natural way, and generally people do not have in mind these steps, even it happening all the time. Teachers use this metaphor (IPO) to show the students that it is necessary to develop some skills and abilities that can help them to organise the solution based on their own perception. We must have in mind the way computers can process information versus the way people process information. We must learn how to explicit our reasoning process in low level (machine level). People have multiple intelligences, and different background and experiences [2]. It interferes in the way they perceive and organise a problem solution. There is not a pattern to solve a problem. Even in Sequential Algorithms, you can have alternative ways to organise the solution.

The process used to build the solution is the main gain obtained by the students. It is necessary to work on the meta-level. The students do not need to know anything about it. But, the teacher must understand it clearly. That is why it is so important to create learning situations based on incremental degree of complexity associated to the exercises and tasks. So, it is necessary to include a set of tools and functionalities to guarantee interaction among students and teachers.

Teachers do not have enough time to check all students' requests during practical classes. Classes have a weekly schedule, and it is not enough to handle with the student's doubts. The number of exercises and tasks sent for the students must be organised, classified, and evaluated in a short period of time. It demands from teachers a lot of time.

We have been working with Algorithms and Programming classes during the last 17 years. Our methodology is based on a set of activities where students must perform in order to acquire knowledge and skills in an incremental way. It means, they have to perform many and different tasks in order to create a particular style to solve the problems. We organise the algorithms contents in the traditional way: sequential, repetition, interactive, and so on. The basic point of our work is related with the use of different virtual resources to help the students learning process, and a student assistant. The set resources are:

- virtual FAQ (Frequently Asked Questions);
- on-line glossary of terms;
- on-line help system;
- communication tools (e-mail, chat, and forums);
- exercise databases (with comments about the solutions);
- students production database (working as a student modelling module);
- an artificial agent to help the teacher to organise the set of material produce by students;
- blackboard linked with a student agenda;
- some tools to test the algorithms or the programs developed by students;
- an assistant student to attend the students when the teacher is not available. This student is called *Monitor*. S/he has a schedule distributed in hours during the week. S/he works closely with the teacher. The interaction between the teacher and the monitor expand the students advising process.

Many of these tools were developed by partners from other universities (see Section 2 for details).

Some projects were studied in order to help us to identify and to model the set of features to be included into our system [3]. The studied environments are:

- *Laura* [4], tool to support the teaching of Fortran language;
- *BRIDGE* [5] and *Proust* [6], systems to teach Pascal language;
- *Greaterp* [7], tutorial system to teach LISP language;
- *Individualised Course for the C Programming Language* [8], hypertext-based system to teach C programming language;
- *ELM-ART*, Web-based ITS (Intelligent Tutoring System) to teach LISP language. [9, 10];
- *ADAPT* (ADA Packages Tool), system to teach Ada language [11];
- *C-Tutor*, ITS to teach C programming language, analyse students' programs and give intention-based diagnosis [12];
- *HabiPro* (Habits of Programming) pedagogical and collaborative software designed to develop good programming habits. It doesn't try to teach programming but to develop in the novice student's skills to become good programmers [13, 14];
- *Virtual Campus PROLOG Tutor*, Web-based educational system to teach Prolog language [15, 16].

Each one of those environments has a different teaching methodology proposal, examples of exercises grouped by levels of complexity, how to conduct a collaborative work, among others. Our project has in common with those systems the following aspects: specific tools to test algorithms, on-line learning materials, communication tools, and an agenda to schedule the student's assignments. Our work extended such environments architecture by modelling other communication resources (like the blackboard), the inclusion of glossary of terms and a help system (both on-line).

The aim of our project is to enrich the results achieved with our methodology developed to support Algorithms and Programming teaching-learning process. The methodology was tested during many semesters using virtual resources without a formalised framework. The tests performed with those tools helped us to understand the amount of information generated from students and teachers interactions. After several experiences the resources were jointed into a virtual environment, having the methodology principles as uphold. So, we formalised a framework to embed those features.

This paper is organised as follow. The Section 1 describes the Algorithms and Programming teaching-learning methodology. Section 2 presents the environment architecture and its features. This section also briefly describes the agent that monitor and organise the student's information set. Section 3 presents final considerations and restrictions faced during the project design process. Moreover, it presents the future works that we intend to do in order to overcome the system limitations. At final section, the references mentioned in this paper are presented.

1. Algorithms and Programming teaching-learning methodology

Computer Science novices have real difficulties with the subjects related to Algorithms and Programming. The main possible causes are study reasoning process based on memorisation, and lack of Mathematics pre-requisites, necessary to formalise solution models.

Our educational approach is based on the constructivism paradigm where learner has a central position in the teaching-learning process. It means the student performance needs to be monitored in a personal way. In order to create a virtual environment to support this approach we designed a society composed by people and virtual resources. The virtual resources are software with different functionalities. We introduced some resources modelled using agent technology. So, this society can be viewed as a hybrid multi-agent system. Considering learning as a central agent into this society implies to have in mind that all decisions, information, and interactions are related to the student actions. The teacher works as a coach, or a guide, and sometimes, as an assistant. In order to perform these multiple strategies, the teacher needs different tools. So, it is necessary to provide a variety of resources to make it possible.

In order to organise the domain, i.e., organise the contents of each subject Conceptual Maps (CMs) were used [17]. CMs are hierarchical graphical representations composed by nodules linked by arches. The nodules contain information and the arches link the information according to the relation function. The CMs allow to represent different levels of concepts, and to organise hierarchically the contents to be taught. These maps can be done through a specific tool or directly in the paper, by hand. The CM helps the teacher to obtain the relationship and the interdependence among disciplines contents.

The class materials are available for the students through the discipline's homepage. The teacher interacts with the students using synchronous tools (chat) or asynchronous tools (e-mail and forum).

The methodology used in the conduction of disciplines is based on the resolution of problems orderly by levels of complexity. The student's duty is to study previously the related theory available on the course Web pages. The agenda allows the student to know in advance the set of activities to be performed during all week. The teacher organise the schedule and respective materials related to each class. Each new lesson begins with a summary of the class, a set of text to present the content, some examples and few questions to check the student comprehension. These questions will be used to start the interactive classes. The teacher tries not to present the exercises solutions. S/he stimulates students to present their solutions for the entire group. Her/his intention is to try to show that there are multiple ways to solve the same problem. The mistakes made by students are also similar. The solutions possibilities differ by optimisation degree.

In order to reinforce this some examples of solutions developed for other students and monitors are available on class homepage. The Monitor plays an important role. The students have a chance to talk about their doubts with another colleague. The teacher supervises the Monitor all the time. This approach has being used for six semesters. In each semester we were refining the operational aspects related to the use of virtual resources and the distribution of the contents. The most important points are:

- it is necessary to organise a general activities schedule for all semester classes. It helps to show for the student the volume of work they will faced (readings, exercises, practical research, tasks, and so on);
- all related material has to be available at least two class before it will be used;
- the support services should be aligned with the available technology. Some of them had been showing inadequate or obsolete;
- Monitor as a team member. We must provide her/him facilities to access the discussion list, and receive/send messages for the students.

We can stand out some positive aspects related to students' behaviour/activities: engagement during the classes, information management in order to prepare themselves previously for the classes, and exchanging solutions with the colleagues.

However, this methodology generates an enormous volume of information to be handling by the teacher. For example: the teacher and the student have to manage the enormous amount of e-mails/forums, and check the updated material. Management information process was done manually by the teacher. In this context, the environment was replaced by the teacher. So, we are changing this integrating all the resources into an environment. The system will now control the information management.

2. PROOGRAMA: a teaching/learning environment based on virtual tools

As mentioned before, the PROOGRAMA environment aims to support teaching/learning activities. It is based on a set of resources and functionalities that allow students and teacher to exchange information about results from evaluating process. The Figure 1 presents the PROOGRAMA architecture.

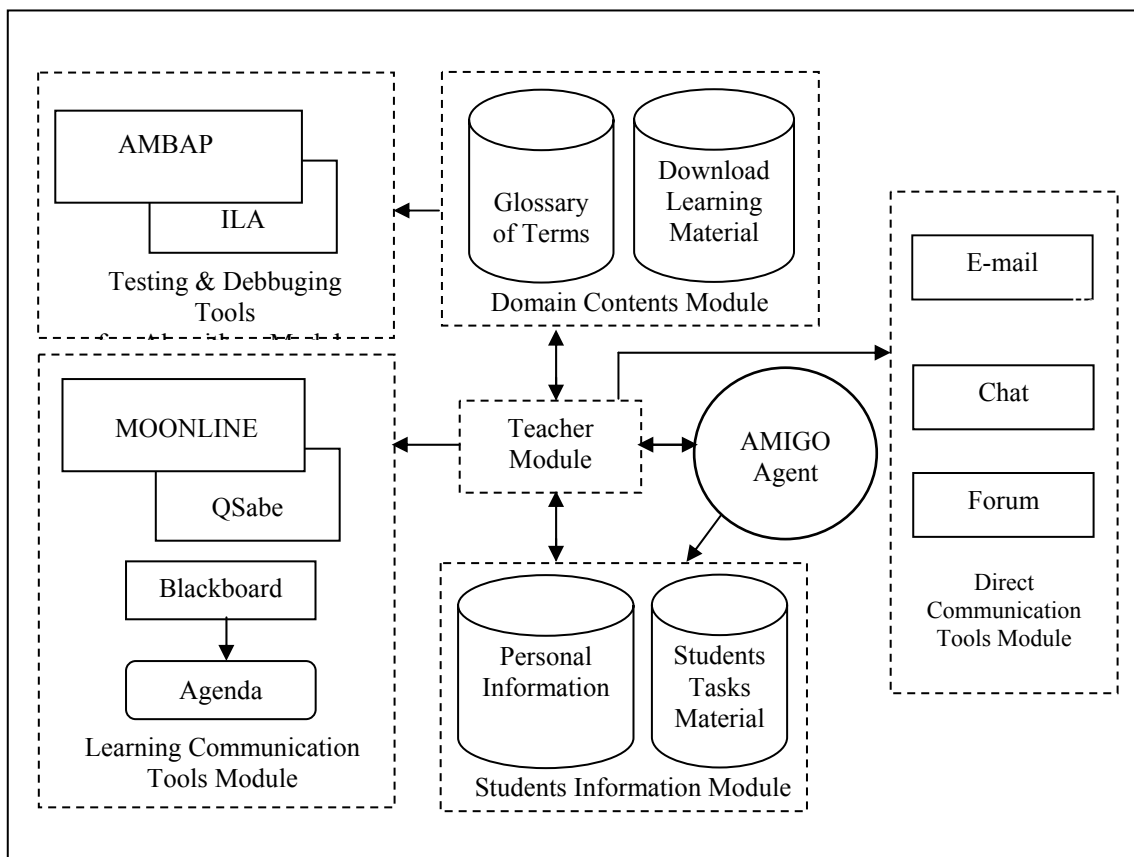


Figure 1. Resources into the PROOGRAMA architecture

The PROOGRAMA environment will offer different working areas, according to the user profile. The teacher is responsible to organise all the information materials related to: repositories, agenda, blackboard, help systems contents, and forum mediation. To perform these tasks the teacher uses the *Domain Contents Module*, selecting exercises to be testing into the *Testing and Debugging Tools for Algorithms Module*, and the *Learning Communication Tools Module*. The Monitor (student assistant) will help the students through the MOONLINE tool (describe forward in this section).

The AMIGO agent will inform the teacher about the student performance related to the evaluation process. The AMIGO agent will also provide a set of reports about each student.

Moreover, a class report can be generated. The student can access information, exchange experiences, and solve doubts using the *Direct Communication Tools Module*. Likewise, all participants can use this module.

In order to help the student to identify and to understand the steps related to the algorithm solution they could use AMBAP - *AMBiente de Aprendizado de Programação* [18] and ILA - *Interpretador de Linguagem Algorítmica* [19]. These tools were developed by Brazilian researches. The AMBAP system allows the student to build programs using an algorithmic language based on Structured Portuguese. Withal, the student can analyze their mistakes and misconceptions. The ILA tool is a compiler that allows testing the algorithms built using AMBAP. ILA is an AMBAP system component.

The communication among the students and the teacher occurs through chat and e-mail. Almeida [20] developed these tools based on Hansen [21] work. This communication is based on Web Services technology.

The student's schedule and tasks will be showed on *Blackboard* and in the *Agenda*. MOONLINE (*Monitoria On-line*) developed by Gava [22] allows people to exchange information to solve common doubts. They can ask question, and obtain answers from a database organised as a FAQ (*Frequented Asked Questions*). This FAQ will be maintained by the Monitor according to the teacher supervision. To better understand the FAQ database we use the QSabe tool. The QSabe name came from a short of a Portuguese expression that means, "Who knows the answer?". The QSabe is capable to identify the content of a question. It analyses the word string. It uses a *thesaurus* where the most common words used in the domain can be found [23]. The QSabe behaviour rules are configured by the teacher. QSabe selects who (in the group) has the knowledge to answer a question if the system could not find the answer.

Any activity related to the evaluation process could be seen at the *Student Information Module*. Each student has access to her/his profile organised by the AMIGO, and supervised by the teacher. AMIGO will help the teacher and students to identify where are the information, and the better choice to be used. AMIGO architecture and its functionalities are presented on section 2.1.

2.1 The AMIGO¹ agent

AMIGO is an agent designed to help the teacher to organise the students' information set. Figure 2 shows AMIGO architecture, which is composed by four modules. The *Task Manager Configuration Module* allows the teacher to configure agent's behaviour. For example, the teacher can select if the agent will manage the uploaded material or not, the reports that will be sent for students, and so on.

The *Updating Student Information Module* is composed by two modules: *Checking Task Base* and *Students Files Organizing*. The first module is responsible to control student's received material. It means the agent will control the blackboard calendar and deadlines defined for each task. The second module checks the files sent by the students. It will check if the file is not empty and contains the specified content. This verification is made by comparing the file extension. The *Students Files Organizing Module* is also responsible to organize the files sent by the students in the *Students Task Material* database. It will happen according to teacher configuration done by the *Task Manager Configuration Module*. AMIGO will create and send reports to the teacher. These reports are created, sent and managed by the *Teacher Report Module*. These modules aim to create reports to follow student's performance.

¹ AMIGO means an abbreviation for *Agente para MonItorar e Gerenciar infOrmações* (Agent for information management). The word in Portuguese means *friend*.

Moreover, the *Student Messages Module* is responsible to organize and to communicate students exchanging information done by e-mail. It intends to help the students to receive information even if they are not currently working on PROGRAMA environment.

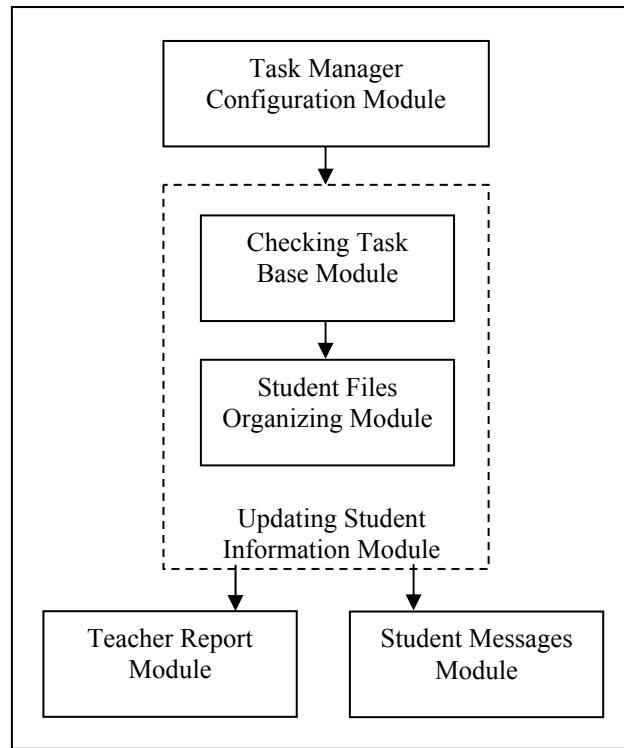


Figure 2. The AMIGO agent architecture

3. Final considerations and future works

This is an ongoing work. However, we faced with some limitations due our previously design. We do not design a student's model. We will just store information generated by students' activities. We intend to add into the students' information module, the choices made by students when they selected something in the environment (interface options). Likewise, we do not have any record about time they spent on tasks. For example, how many times the students activated the help system, and what kind of information they were looking for. We do not know either if they use or not the tools to test the algorithms. We just have records from e-mail and forum discuss.

If we add more information about the student, it allows teacher to better organise the student interface and personal students plans. Some of these considerations can be solved developing student personal assistants (agent approach). We expect to work on it using our partnership with Universidade Nova de Lisboa (UNL). UNL is a Portuguese university at Porto City in Portugal. An international co-operative project supported by the Scientific Technologic Office from both countries is on the way. The aim of this new project is to explore and to apply results involving Natural Language Processing (Portuguese). We intend to analyse the files contents instead of only the extension.

At this moment, the AMIGO agent can check the students' files extensions. For example, we can test if the file contains a source code or an executable program. AMIGO can test those files in order to check if the program can be runned, or if the source code can

be compiled. If the student sent a text file, AMIGO can test if the file name attends the specification asked by the teacher. However, we can not know if the content is really related to the task. Summarising, we can organise the set of information generated by our methodology into the framework. Although, we need to improve the way we handle with the students' preferences and particularities, and to improve given support for teacher evaluation process.

In fact, we intend to include more agents in order to cope with the new functionalities related to the student model and files analysing. It also explains the existence of the AMIGO as an agent into the PROGRAMA environment. Our intention is to have a real multi-agent system.

References

- [1] Laird, J. (1995) *Mental models: towards a cognitive science of language, inference, and consciousness*. USA, Harvard University Press.
- [2] Gardner, H. (1993) *Frames of mind: the theory of multiple intelligences*. USA, Basic Books.
- [3] Marczak, S.; GIRAFFA, L. (2002) *Ambientes Inteligentes para Suporte ao Ensino de Programação*. Trabalho Individual I (Mestrado em Ciência da Computação) - PPGCC, FACIN, PUCRS, Porto Alegre. (In Portuguese)
- [4] Adam, A.; Lawrent, J. (1980) A system to debug students programs. In: *Journal of Artificial Intelligence*, v. 15.
- [5] Bonar, J.; Cunningham, R. (1985) BRIDGE: tutoring the programming process. In: *Journal of Artificial Intelligence*.
- [6] Johnson, W; Soloway, E. (1987) PROUST: an automatic debugger for Pascal programs. In: *Artificial Intelligence in Education: applications and methods*. Addison Wesley.
- [7] du Boulay, B.; Sothcott, C. (1987) Computers teaching programming: an introductory survey of the field. In: *Artificial Intelligence in Education: learning environment and tutoring systems*, v. 6.
- [8] Kay, J.; Kummerfeld, R. (1994) "An Individualised Course for the C Programming Language", Available on-line in <http://archive.ncsa.uiuc.edu/SDG/IT94/Proceedings/Educ/kummerfeld.html>. (Consulted in March 2002)
- [9] Brusilovsky, P.; Schwarz, E.; Weber, G. (1996) ELM-ART: An Intelligent Tutoring System on World Wide Web. In: *Intelligent Tutoring Systems Conference, 3, 1996, Montréal, Canada*. Montréal.
- [10] Weber, G.; Brusilovsky, P. (2001) ELM-ART: An Adaptive Versatile System for Web-based Instruction. In: *International Journal of Artificial Intelligence in Education*, 12, 351-384. Available on-line in http://cbl.leeds.ac.uk/ijaied/abstracts/Vol_12/weber.html. (Consulted in April. 2003)
- [11] Vikki, F.; Wiedenbeck, S. (1996) An Intelligent Tool to Aid Students in Learning Second and Subsequent Programming Languages, *Computers & Education Magazine*, vol. 27, no. 2.
- [12] Song, J. *et al.* (1997) An Intelligent Tutoring System for Introductory C Language Course, *Computers & Education Magazine*, vol. 28, no. 2.
- [13] Vizcaíno, A. *et al.* (2000) An Adaptive, Collaborative Environment to Develop Good Habits in Programming. In: *Intelligent Tutoring Systems Conference, 5, Montréal, Canada, June 2000*. Montréal.
- [14] Vizcaíno, A.; Prieto, M. (2000) Examining the Effectiveness of New Technology in High School (number: 6) Available on-line in <http://citeseer.nj.nec.com/464747.html>. (Consulted in apr. 2003)
- [15] Peylo, C. *et al.* (2000) A web-based intelligent educational system for PROLOG. Available o-line in <http://citeseer.nj.nec.com/552823.html>. (Consulted in apr. 2003)
- [16] Gust, H. *et al.* (1999) The Virtual Campus Prolog Learning Environment". In: *Artificial Intelligence in Education Conference*, France, April 1999. France.
- [17] Ausubel, D. *et al.* (1980) *Psicologia Educacional*. Editora Interamericana, New York. (In Portuguese)
- [18] Almeida, E. S. *et al.* (2002) *AMBAP: Um Ambiente de Apoio ao aprendizado de Programação*. In: CONGRESSO NACIONAL DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 22, 2002, Florianópolis. *Anais...* Florianópolis: SBC. (In Portuguese)
- [19] Evaristo, J.; Crespo, S. (2000) *Aprendendo a Programar: Programando numa Linguagem Algorítmica Executável (ILA)*. Porto Alegre, Book Express. (In Portuguese)
- [20] Almeida, G. (2003) Developing Web services for e-mails and chats. Porto Alegre: PPGCC, FACIN/PUCRS. Technical Report.

- [21] Hansen, R. P. (2003) *GlueScript*: Uma linguagem específica de domínio para composição de *Web-Services*. Dissertação (Mestrado em Computação Aplicada) - CCET, UNISINOS, Porto Alegre. (In Portuguese)
- [22] Gava, T. B.; Menezes, C. S. (2000) *Moonline*: Um Sistema Multiagentes Baseado na Web para Apoio a Aprendizagem. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 11, 2000, Maceió, AL. *Anais...* Maceió: UFAL. (In Portuguese)
- [23] Menezes, C. *et al.* (2002) “QSabe - Trocando Experiências sobre Informática Educativa em uma Rede de Educadores”. Available on-line in <http://www.inf.ufsc.br/sbc-ie/revista/nr2/Menezes02.htm>, (Consulted in feb. 2003) (In Portuguese)