Algorithms **2010**, *3*, 183-196; doi:10.3390/a3020183

OPEN ACCESS

algorithms

ISSN 1999-4893

www.mdpi.com/journal/algorithms

Article

Integrating New Technologies and Existing Tools to Promote Programming Learning

Álvaro Santos [1], Anabela Gomes [1] and António José Mendes [2,*]

[1] Instituto Superior de Engenharia de Coimbra, Rua Pedro Nunes, 3030 Coimbra, Portugal; E-Mails: ans@isec.pt (A.S); anabela@isec.pt (A.G.)

[2] Centro de Informática e Sistemas da Universidade de Coimbra, Pólo II da Universidade de Coimbra, 3030 Coimbra, Portugal

* Author to whom correspondence should be addressed; E-Mail: toze@dei.uc.pt; Tel.: +351-239790036; Fax: +351-239701266.

Received: 3 March 2010; in revised form: 28 March 2010 / Accepted: 1 April 2010 / Published: 20 April 2010

Abstract: In recent years, many tools have been proposed to reduce programming learning difficulties felt by many students. Our group has contributed to this effort through the development of several tools, such as VIP, SICAS, OOP-Anim, SICAS-COL and H-SICAS. Even though we had some positive results, the utilization of these tools doesn't seem to significantly reduce weaker student's difficulties. These students need stronger support to motivate them to get engaged in learning activities, inside and outside classroom. Nowadays, many technologies are available to create contexts that may help to accomplish this goal. We consider that a promising path goes through the integration of solutions. In this paper we analyze the features, strengths and weaknesses of the tools developed by our group. Based on these considerations we present a new environment, integrating different types of pedagogical approaches, resources, tools and technologies for programming learning support. With this environment, currently under development, it will be possible to review contents and lessons, based on video and screen captures. The support for collaborative tasks is another key point to improve and stimulate different models of teamwork. The platform will also allow the creation of various alternative models (learning objects) for the same subject, enabling personalized learning paths adapted to each student knowledge level, needs and preferential learning styles. The learning sequences will work as a study organizer, following a suitable taxonomy, according to student's cognitive skills. Although the main goal of this environment is to support students with more difficulties, it will provide a set of resources supporting the learning of more advanced topics. Software engineering techniques and representations, object orientation and event programming are features that will be available in order to promote the learning progress of students.

Keywords: programming learning environments; collaborative environments

1. Introduction

Most computer science higher education institutions, all over the World, are faced with student failure problems in programming courses. Several reasons have already been identified in the literature, such as the abstraction capabilities and problem solving skills required for learning programming [1,2]. Many solutions have also been proposed, but it is commonly accepted that the failure rates continue to be excessively high. It is known that some students have more aptitude than others for computer programming tasks, yet we also know that all students can achieve success in programming, if they are committed and have adequate orientation.

Teachers are aware of this situation but, often, class sizes make it almost impossible to provide acceptable orientation to all students. Each of them has his/her own background knowledge and pace, which, frequently, demands individual orientation methods.

Many tools have been developed to assist teaching, learning and training programming tasks. Every tool has its own benefits. However it is difficult to find one suitable for all students needs. Depending on the actual knowledge level and preferable study method of each student, we need to make the right tool available at the right time. Once again it is almost impossible for teachers to do this work due to class sizes.

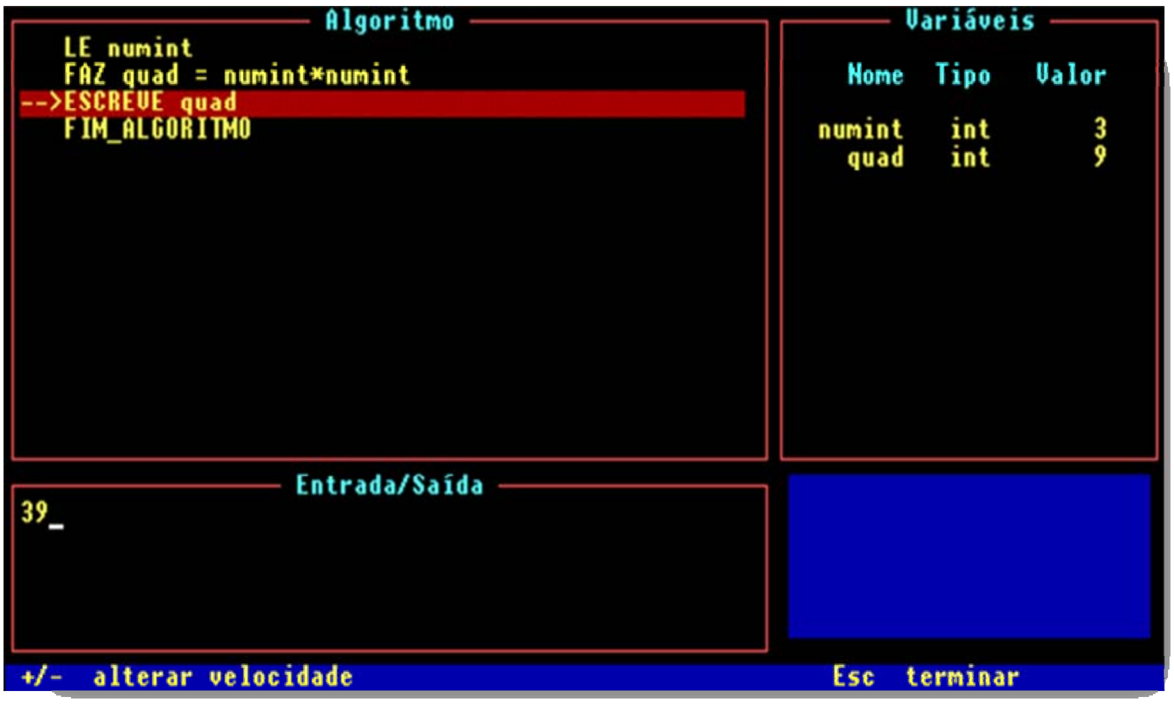
In this paper, we present some tools that our research group developed or contributed to its development during the years, focusing on some benefits that were reached with them. We also present a new environment that is under development. We think it will contribute to minimize some of the mentioned problems, namely limitations caused by class size and students heterogeneity in knowledge and pace.

2. Developed Tools

2.1. VIP

VIP (the Portuguese acronym for Interactive Program Visualization, Figure 1) was the first tool developed by our group, in the late 1980s [3]. At that time programming and the world of computers was essentially based on a black screen with a blinking cursor. The programming tasks and, mainly, debugging tasks were a hard job. Furthermore, trying to explain this new computing world to students entering the university with no knowledge about computers and programming was very difficult. VIP was a system that contributed in minimizing these difficulties, simplifying the understanding of basic programs. VIP allowed the students to write pseudo-code and observe the consequent simulation. With this environment it was possible to run each pseudo-code line step-by-step and observe the evolution of variable values and, also, the output of the program.

Figure 1. The VIP interface.



```
   LE numint
   FAZ quad = numint*numint
-->ESCREVE quad
   FIM_ALGORITMO
```

```
  Nome   Tipo   Valor

numint   int        3
  quad   int        9
```

```
39_
```

2.2. SICAS (2000)

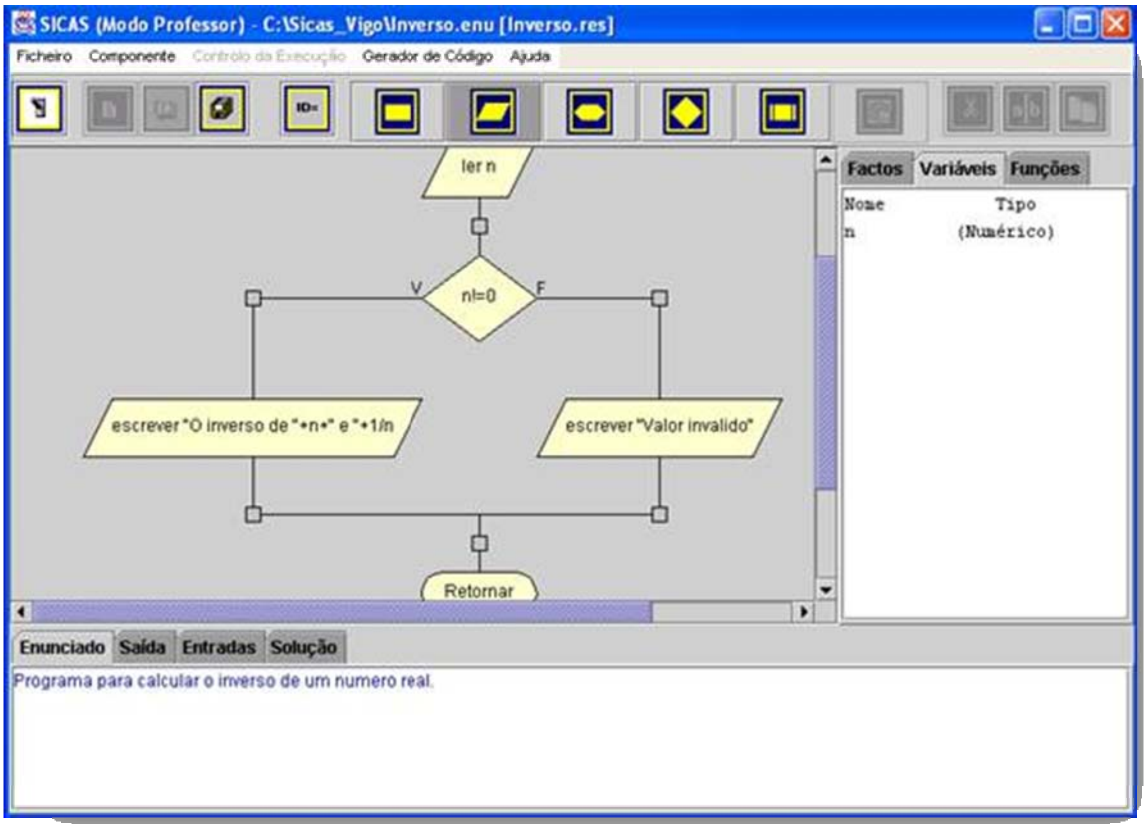
In the late 1990s our group developed SICAS (a Portuguese acronym for Interactive System for Algorithm Development and Simulation, Figure 2). It was based on constructivist theories and aimed to create a pleasant programming tool that students may enjoy to use. SICAS doesn’t include any explanatory material, but presents an environment that allows students to develop their capacities on the basis of practice, allowing them to design, observe, analyse and visually simulate algorithms. The idea is that students create solutions, detect eventual errors made, correct them and learn from those activities. This system allows students to implement algorithms to solve problems, using a flowchart representation. After that it is possible to execute/simulate the solution step-by-step, or continuously at a low or fast speed. The entire interface was constructed to be simple and easy to use. In this tool the complexities were minimized to focus the students’ attention in the real task—problem solving.

SICAS provides support for the most common initial programming examples, since it allows the utilization of assignments, basic input/output, conditional structures and loops. It also permits the definition of functions to support more complex problems, including recursive functions. Although we tried to simplify the syntactical details students have to tackle, the environment has all the basic structures needed to start programming. It also supports common data types, namely numbers, strings and one-dimensional arrays.

Students can export their solutions to some other representations, namely Java, C and pseudo-code. Our representation options, allowing the students to write their algorithms without using a common programming language, stress our view that the important part is to develop algorithm creation skills, leaving the programming language syntactic details to a less important role.

SICAS has two working modes: a student mode and a teacher mode. The second has some extra features, allowing teachers to create new problems. This involves the problem description, a solution example and a set of input/output to allow the environment to test student solutions. A more complete description of SICAS can be found in [4,5].

Figure 2. SICAS.



Nome	Tipo
n	(Numérico)

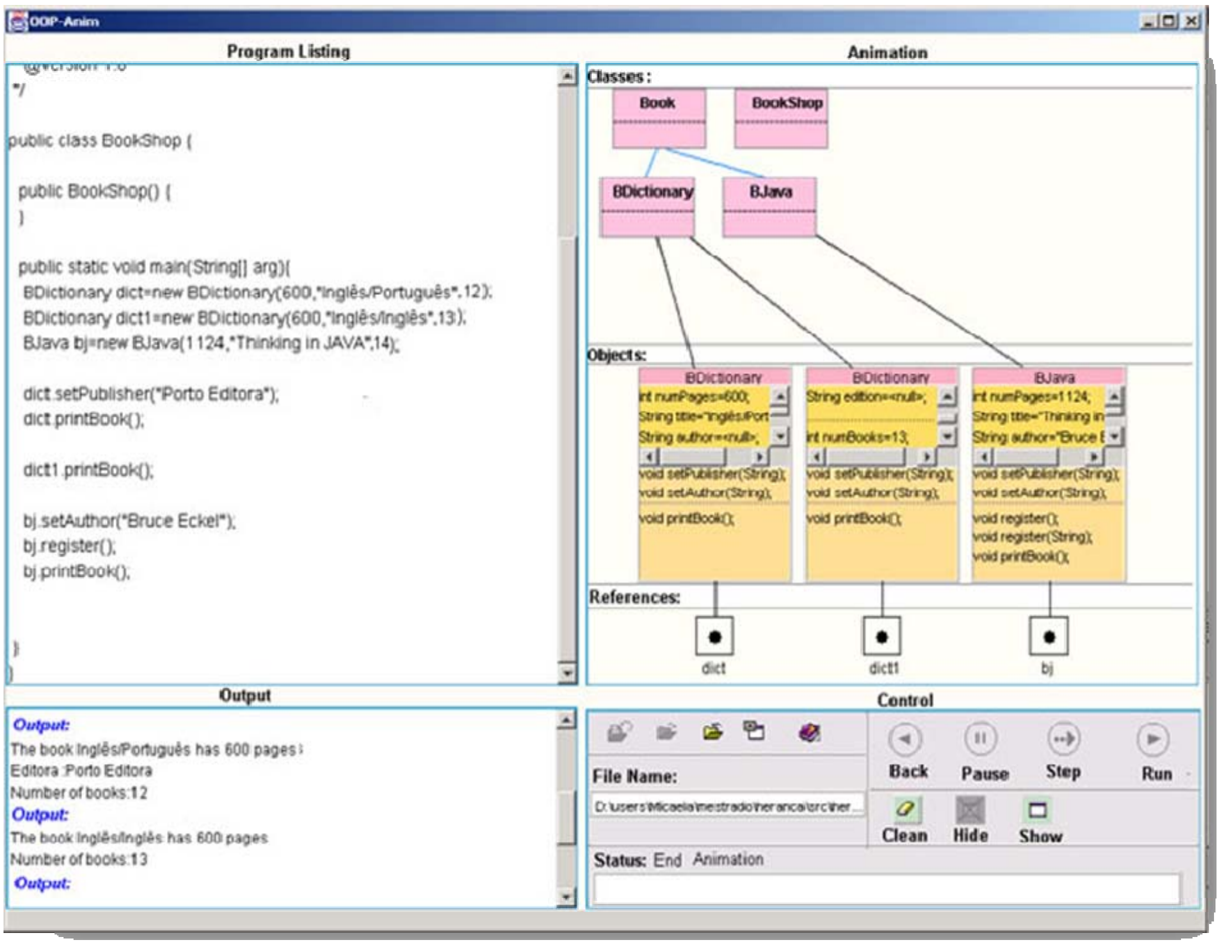
Programa para calcular o inverso de um numero real.

2.3. OOP-Anim (2004)

Based on SICAS' underlying ideas we also developed OOP-Anim (Figure 3). This system intends to help students understand basic object oriented concepts through the creation and visualization of programs. Like SICAS, OOP-Anim allows program simulation, but in this case they have to be written in Java. With OOP-Anim students can understand the effect of every instruction in terms of existing class definitions, object instances and related references.

Operations in OOP-Anim start with a Java program, normally written by the students. After that it is possible to control code execution and visualize the effect of each instruction in terms of program output and the object instances that will be created, accessed and modified. A more detailed description of OOP-Anim can be found in [6].

Figure 3. OOP-Anim.



```
*/

public class BookShop {

  public BookShop() {
  }

  public static void main(String[] arg){
   BDictionary dict=new BDictionary(600,"Inglês/Português",12);
   BDictionary dict1=new BDictionary(600,"Inglês/Inglês",13);
   BJava bj=new BJava(1124,"Thinking in JAVA",14);

   dict.setPublisher("Porto Editora");
   dict.printBook();

   dict1.printBook();

   bj.setAuthor("Bruce Eckel");
   bj.register();
   bj.printBook();


 }
}
```

```
Output:
The book Inglês/Português has 600 pages
Editora :Porto Editora
Number of books:12
Output:
The book Inglês/Inglês has 600 pages
Number of books:13
Output:
```

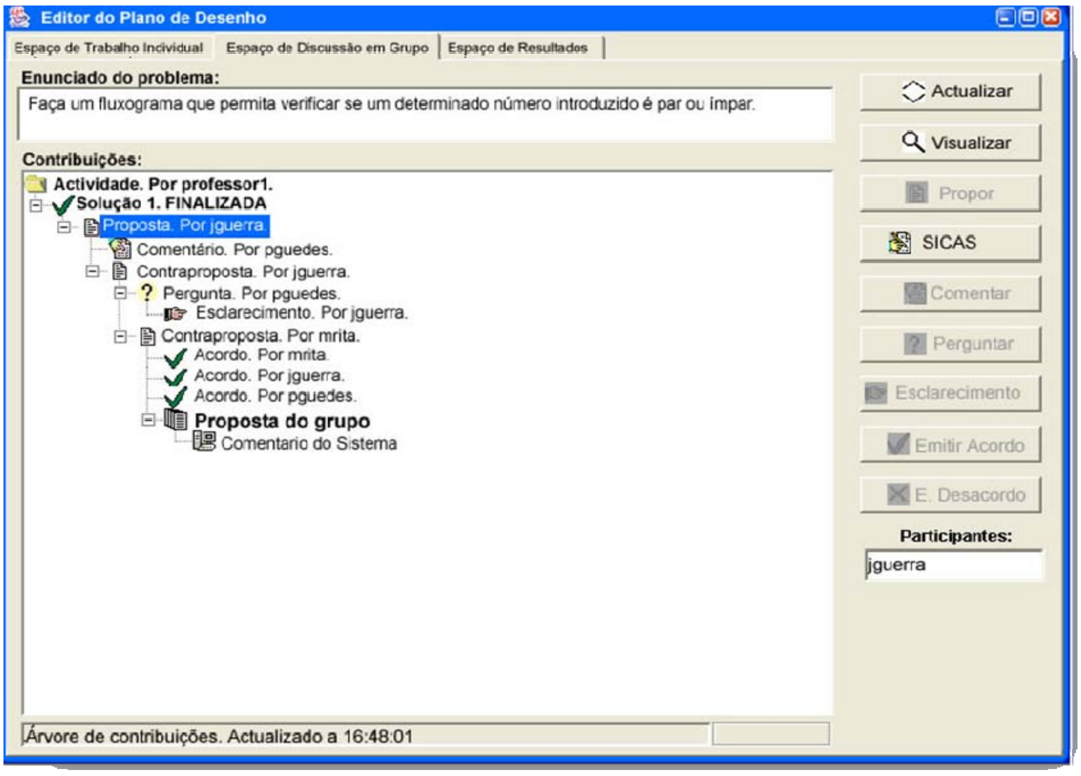
2.4. SICAS-COL (2005)

Considering the potential of collaborative learning and educational simulation we decided to create a new version of SICAS that could support remote collaborative learning activities. The new tool was called SICAS-COL (Figure 4) [7] and resulted from the integration of SICAS with PlanEdit [8], a collaborative tool created within the scope of project Domosim-TPC [9], developed by our colleagues of the CHICO group at the University of Castilla—La Mancha (Spain).

PlanEdit was designed to support student discussions during problem solving group activities. It was first used in problem solving in the domotics field, but it can be used in other areas. By associating PlanEdit collaborative support with SICAS flowcharts, we were able to develop a powerful instrument to promote group discussion in the context of programming tasks. Each group member can propose solutions, expressed as SICAS flowcharts. With PlanEdit, students can contribute to a group solution. They can visualize proposals, discuss them, suggest modifications to proposed solutions, present alternative solutions, make comments, vote to express agreement or disagreement and choose the direction of teamwork development.

SICAS-COL is organized in three workspaces: a space for individual work, a space for group discussion and a space to share results. When the student is in the individual workspace, he has the opportunity to create, modify or visualize solutions. In the discussion workspace, students can discuss solutions and propose alternatives. The result workspace is used to store proposed solutions and other documents related with the problem at hand.

Figure 4. SICAS-COL.



Enunciado do problema:
Faça um fluxograma que permita verificar se um determinado número introduzido é par ou impar.

Contribuições:

```
Actividade. Por professor1.
⊟ Solução 1. FINALIZADA
    ⊟ Proposta. Por jguerra.
        Comentário. Por pguedes.
        ⊟ Contraproposta. Por jguerra.
            ⊟ ? Pergunta. Por pguedes.
                Esclarecimento. Por jguerra.
            ⊟ Contraproposta. Por mrita.
                Acordo. Por mrita.
                Acordo. Por jguerra.
                Acordo. Por pguedes.
                ⊟ Proposta do grupo
                    Comentario do Sistema
```

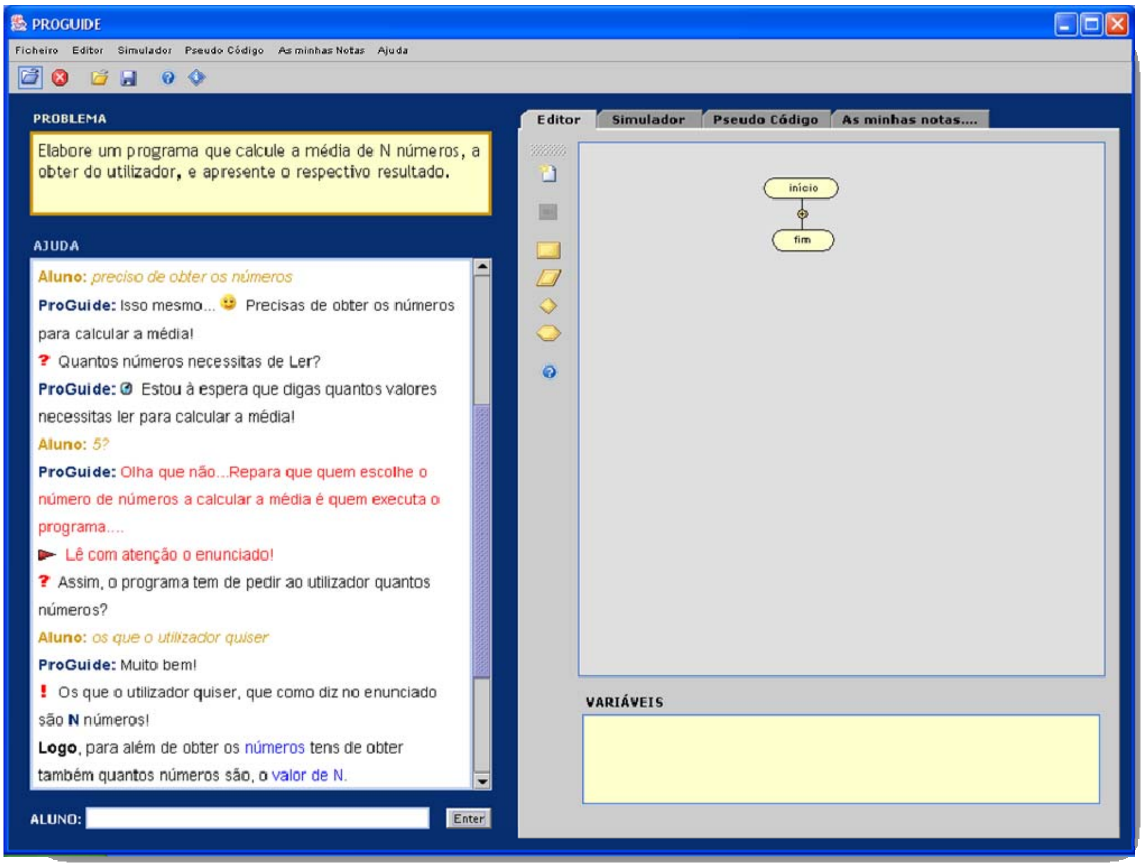
Participantes:
jguerra

Árvore de contribuições. Actualizado a 16:48:01

2.5. ProGuide (2005)

Although the use of SICAS has shown good results with some students [10], we verified that some other students needed a more guided approach, so in 2004, a new tool was developed, keeping in mind the difficulties of students that don't know how to begin to solve a problem. This system is ProGuide (Figure 5) and its main goal is to help reduce the difficulties that weaker programming students show in the initial learning stages [11]. It is based on a tutoring system model. ProGuide interacts with students when they are trying to solve a problem, using an algorithm representation similar to SICAS flowcharts. The system presents warnings about problems that can appear and questions students, trying to guide them in the different steps that lead to the problem solution. The interaction is based on internal structures that store information on problems and their solutions.

Figure 5. ProGuide.



PROBLEMA

Elabore um programa que calcule a média de N números, a obter do utilizador, e apresente o respectivo resultado.

AJUDA

Aluno: *preciso de obter os números*

ProGuide: Isso mesmo... Precisas de obter os números para calcular a média!

? Quantos números necessitas de Ler?

ProGuide: Estou à espera que digas quantos valores necessitas ler para calcular a média!

Aluno: *5?*

ProGuide: Olha que não...Repara que quem escolhe o número de números a calcular a média é quem executa o programa....

Lê com atenção o enunciado!

? Assim, o programa tem de pedir ao utilizador quantos números?

Aluno: *os que o utilizador quiser*

ProGuide: Muito bem!

! Os que o utilizador quiser, que como diz no enunciado são **N** números!

Logo, para além de obter os números tens de obter também quantos números são, o valor de N.

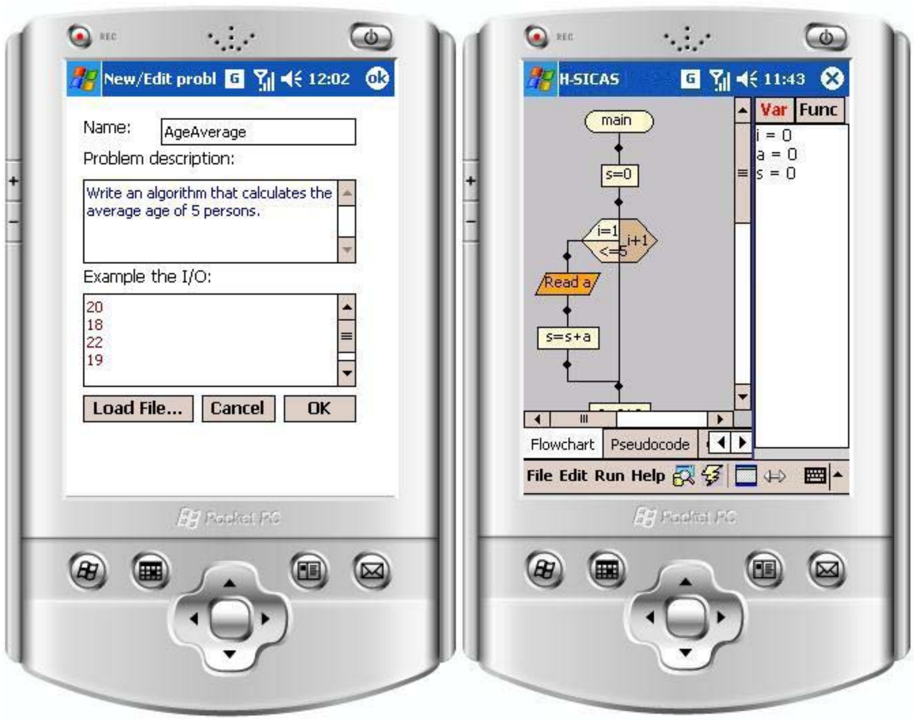
VARIÁVEIS

2.6. H-SICAS (2008)

One of the most important aspects that may influence learning success is the students' motivation. Any methodology, context or technology that can be a step forward in this process must be used. Technologies that student's are aware of and like can be an important mechanism to promote motivation. Therefore, they may help reach the desired success level in the learning process. Nowadays, students are familiar with most mobile devices, especially, computer science students. So, we decided to take advantage of the attractive features that these devices present, to have benefits in terms of an educational programming point of view.

H-SICAS (Figure 6) is an adaptation of SICAS to be used on mobile devices [12]. Perhaps the actual state of these devices is still not the best for programming learning, especially due to the limitations imposed by screen sizes and input interaction restrictions. However, as stated, the main goal is to exploit every kind of system or technology that can motivate students in the programming learning process.

Figure 6. H-SICAS.


Name: AgeAverage

Problem description:

Write an algorithm that calculates the average age of 5 persons.

Example the I/O:

20
18
22
19

i = 0
a = 0
s = 0

3. Analysis of Tools

We think that each presented tool maintains its own identity and usefulness. However, none of the available tools completely fulfils the needs of all programming students. All these developments are justified because we have students with different levels. A very weak student could start with ProGuide. After the development of some abilities that already allow him/her to develop simple solutions, he/she could work individually using SICAS. It is also possible to initiate group work using SICAS-COL to support interactions between students, even at a distance. When students start to program using object oriented concepts, they can use OOP-Anim to help them to understand underlying concepts, object relations and dependencies.

If we consider a learning style model, such as the one proposed by Felder and Silverman [13], we can see that our tools tend to favour students with certain characteristics. One of the model dimensions characterizes students as verbal or visual. It is clear that our tools, making extensive use of visualization, are more oriented to visual students than to their verbal colleagues. We can find in the literature many references to studies that conclude that most engineering students are visual [14–24]. However, we believe that a pedagogical environment should support different types of students. Ideally, the environment should adapt itself to each student learning style, presenting adequate learning activities. Thus, our proposal is centred firstly in a personalized education, which adapts the activities to each student in accordance with his/her cognitive state, rhythm and learning style.

We also noticed that flowcharts are relatively easy to understand by students. In general, students understand examples made with SICAS and are able to predict their output. Moreover, after analyzing some examples, many of them are able to solve simple problems. Using the SICAS options for code generation, students understand that the conversion from flowcharts to a real program is almost direct

However, when we ask students to close SICAS and create programs from scratch, in a classical environment, the difficulties start again. Initially, SICAS was planned only for initiating students in very simple programming tasks. Nevertheless, the experience has shown that it is desirable to extend the period of its use. The tool should help in the next learning phase when students start to code directly in a programming language. For that, we must extend SICAS features not only in terms of the language itself, but also to include other concepts, making SICAS's lifecycle and utility longer. We must extend data types, support object oriented programming and other common programming activities. Support to remote teamwork and synchronized mechanisms that allow program development using flowcharts or real code should also be included. However, even extending the environment characteristics, we don't want to lose one of the best features of SICAS, the possibility that students have to create and simulate their own algorithms/programs.

4. Proposal

Each of the tools presented in the previous section was planned for a specific context, subject, a limited set of lessons or to help in workgroup projects. Every tool has a major or minor contribution but also an additional workload in the learning process due to the time needed to understand each of them. Every time we want to present a new tool, even simple and user friendly, we need to explain some fundamental aspects of its interface and functionalities. Sometimes, the time spent exceeds the time that the activity needs. Associated with this, teachers are frequently confronted with heterogeneous classes with different problems, occurring during tool presentation, which compromises the normal class evolution.

The heterogeneity issue represents a serious problem because it is something that the teachers are confronted with right from the beginning of the semester. Each student has his/her own knowledge level, learning style, goals and social context. Each student has his/her own pace!

In the first few lessons, when the teacher starts to introduce the concepts, the feedback that she/he usually receives comes from the students that already have some background in the field. Other students that have some difficulties postpone their doubts, hoping that their doubts will be clarified with the rest of the class. As a result, teachers follow the more advanced students' pace. When the teacher realizes the real situation she/he is forced to reorganize activities, concentrating her/his efforts on students that show more problems. Sometimes, it's like starting from scratch. This duality creates many problems to the instructor, as it is very difficult to find a rhythm and level of detail that is adequate to most students.

Another point that we must be aware of is the fact that programming learning is, essentially and understandably, based on practical activities. Nevertheless, we can't forget the importance of lectures about concepts and other activities usual in programming courses. It would be beneficial to join diverse tools and teaching approaches that create a consistent environment. This environment should permit a balanced skill development, both theoretical and practical and adapted to each student pace.

Nowadays, one of the more used methods to grant access to contents is the e-Learning platform. It has the advantage of allowing students to access contents asynchronously, at their own rhythm. However, the usual e-Learning platforms are not prepared for programming particularities.

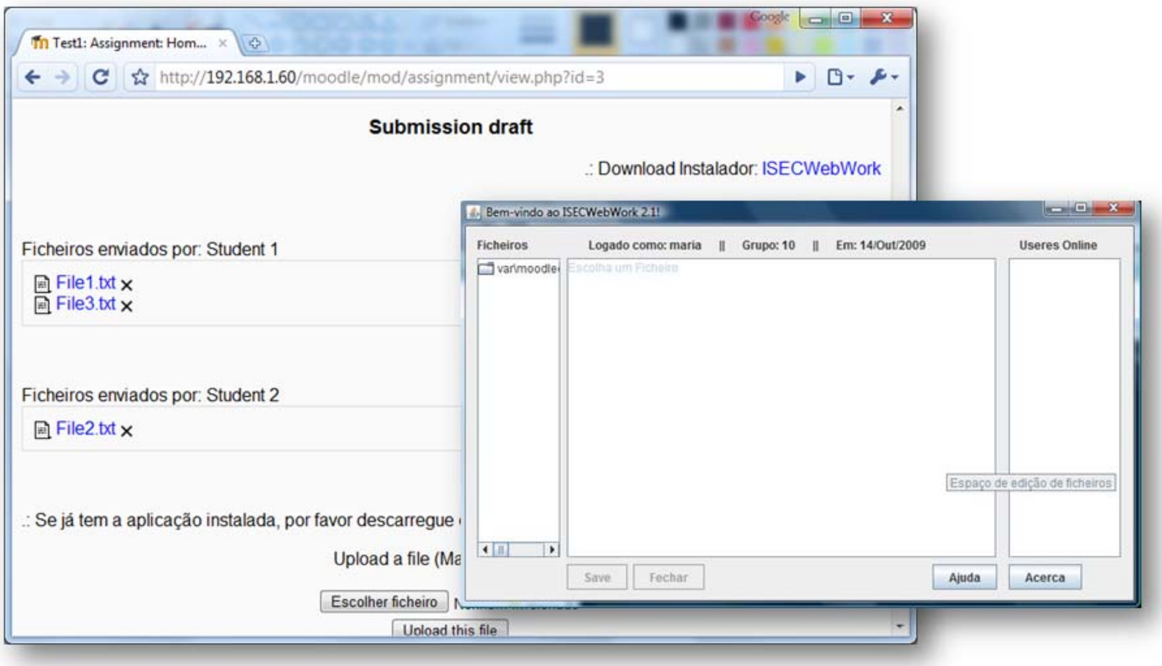
What we are now proposing is to take advantage of an e-Learning platform, already known by students, enriched with a set of tools adapted to programming learning needs. One of the most important features is the ability to define learning paths tailored to each student's actual level, eventually based on student learning styles and the student evolution. These learning paths are formed by lessons on fundamental concepts, but also include exercises that permit knowledge consolidation. Students must be able to access the right tools that are adapted to each exercise, which should be chosen automatically by the e-Learning platform. It is also important that the platform and tools allow collaboration among students and between teacher and students.

The prototype that we are now developing is integrated in the Moodle e-Learning platform. We are developing some new types of activities to include in Moodle that can answer the needs already identified. It is essential for the definition of learning paths to introduce mechanisms that can restrict access to some activities. In the next version of Moodle, version 2.00, it is already expected that some of these mechanisms will be implemented, such as the "Conditional activities" [25]. Nevertheless, we think that this new improvement is not enough for our purposes. Different students should have different conditions to trigger each activity. It's also expected that the next version will implement "Progress tracking" which will permit the definition of personal learning plans based on courses completed or outcomes reached. Even if these new features are available in the next version, we need to enrich them with some automation to help teachers define different and appropriated plans for each student. To automate the process we will take into account the results of some inquiries (to test learning styles and programming background), the competences associated with each activity, the skills that students should develop and, when possible, successful and unsuccessful history cases.

Another essential tool that should be worked on is related with assignments development. In the present version of Moodle, there are some types of assignments, but they only permit the organization and control of the delivery process. Moreover, the current available assignment types were not thought for programming contexts. Even with the more advanced type of assignment, that permits multiple file submission, we can't create a repository that allows effective teamwork. When one uploads a new file, other members can't access it. We have already developed a new assignment prototype (Figure 7) that allows the constitution of a group portfolio by all workgroup elements. This new type of assignment has associated a system that permits the edition of each file, simultaneously and synchronically, by all group elements. This system provides a secure access to Moodle platform files, available for that group in one precise subject/course. For now documents are always assumed as text files but we want to extend the features of this system. For that, we are already developing a new SICAS (SICAS NG), based on the original one, but enriched with features that we identified as essential during these past years.

SICAS NG will follow the original goal of SICAS: being an easy to use tool that allows the creation and simulation of algorithms using a flowchart representation. Moreover, we also want the new version to be an effective tool for more advanced stages in the programming learning process. In this new version, the object concept will be supported and related concepts like encapsulation, inheritance and polymorphism, will be available. Other types of representations will be used alongside with flowcharts, namely UML and text based representations. Another feature of this new version is the possibility to collaborate in a programming project. This feature should allow simultaneous editing by the group elements. It is also planned that in SICAS NG students can ask for their teachers help. If the teacher is online, it will be possible for her/him to integrate a collaborative session, make changes to the code or comment on some parts. SICAS NG can also be used in a classroom environment to let the teacher build examples, step-by-step, collaboratively with students. Students can follow the construction of the example on their own computer, in the classroom or some other place. During the explanation, the teacher can ask a student or the class to make a specific improvement of an example. After that, the teacher can present a personal solution or choose one of the students' solutions as a good (or bad) example to the class. All the SICAS NG projects can be available through Moodle platform or stored locally.

Figure 7. New assignment prototype.

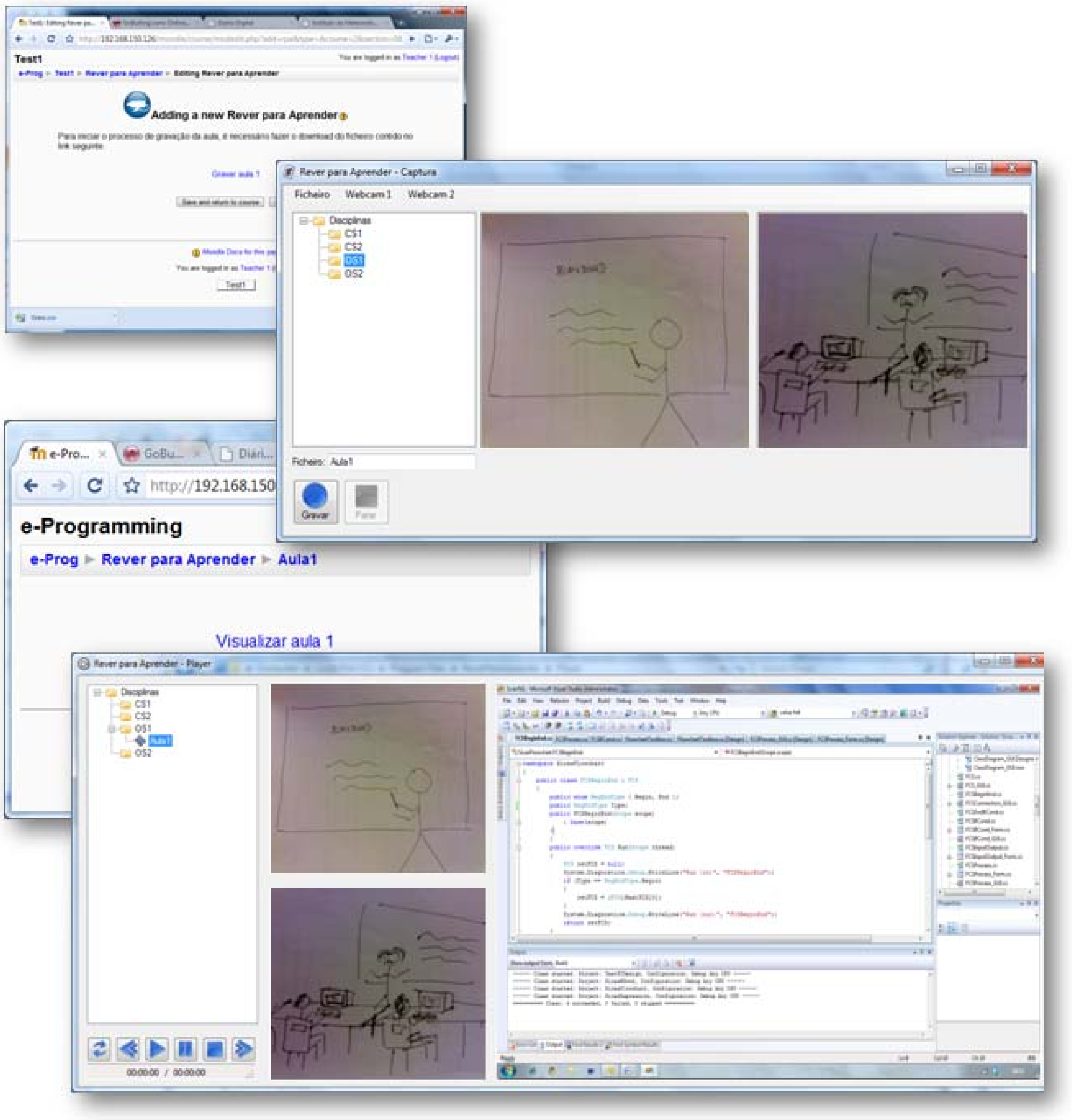

Test1: Assignment: Hom...
http://192.168.1.60/moodle/mod/assignment/view.php?id=3
Submission draft
.: Download Instalador: ISECWebWork
Ficheiros enviados por: Student 1
File1.txt
File3.txt
Ficheiros enviados por: Student 2
File2.txt
.: Se já tem a aplicação instalada, por favor descarregue
Upload a file (Ma
Escolher ficheiro
Upload this file
Bem-vindo ao ISECWebWork 2.1!
Ficheiros
Logado como: maria || Grupo: 10 || Em: 14/Out/2009
Useres Online
var\moodle
Escolha um Ficheiro
Espaço de edição de ficheiros
Save
Fechar
Ajuda
Acerca

Another point that we will focus on will be the Moodle lesson activity. We want to make available complete lessons with concepts, examples and programming exercises. Examples that were built during the related classroom lesson can possibly be integrated in this lesson activity. In order for this to happen, we need to provide SICAS NG with a "step record" and a feature for read-only projects with automatic play, enabling the revision of steps. Associated with this improvement we will integrate a new sub-system, for which we already have a functional prototype (Figure 8).

It supports lesson recording and lesson reviewing ("replay lessons"). For a simpler use of this new system, we integrated it through a new activity developed for Moodle. When this activity is created by the teacher, it will allow the recording of core points in the lessons. The core points identified were the teacher's voice, the computer in focus in the class (usually, the teacher's computer), the "blackboard" and, optionally, the classroom environment. External applications that are needed for the lesson record phase and, later, for lesson reviewing by students are executed transparently from the e-Learning platform. The information recorded is stored in a second server, to face problems related with the bandwidth and service quality. Even though a second server is used, we attempt to minimize security problems by only allowing access to previously authenticated users within the e-Learning platform.

One of the features that we intend to create is the lesson record editing to allow teachers to remove parts that are not relevant to the learning process.

Figure 8. Lesson recording and reviewing.



Test1

e-Prog ▸ Test1 ▸ Rever para Aprender ▸ Editing Rever para Aprender

Adding a new Rever para Aprender

Para iniciar o processo de gravação da aula, é necessário fazer o download do ficheiro contido no link seguinte:

With this new environment, we plan to create a context that gives a wider and more structured support to students. Adaptability is a key issue, as we feel that many classes and activities fail to reach their objectives with many students. So, it is important that alternative strategies and activities are available, to support student's different needs.

5. Conclusions

Many efforts have been made to support programming learning activities. Nevertheless, the results are far from having a clear success level. Problems with the programming learning process, and consequent failure rates, create classes with an enormous number of students. The problem increases even more with the usual class heterogeneity, with students of different levels, with different knowledge and pace. A classical approach to these classes is bound to failure because teachers can't meet each student's needs. In this paper the efforts that our group developed to face programming problems were presented. This effort resulted in a set of tools to support programming teaching and learning. We also presented a new environment currently under development. It integrates diverse technologies to support different activities suitable for a wide range of students' cognitive needs.

References and Notes

1. Bennedsen, J.; Caspersen, M. Abstraction ability as an indicator of success for learning object-oriented programming? *SIGCSE Bullet.* **2005**, *38*, 39–43.
2. Jenkins, T. On the difficulty of learning to program. In Proceedings of the 3rd Annual LTSN_ICS Conference, Loughborough, UK, August 2002; pp. 53–58.
3. Mendes, A.; Mendes, T. VIP—A tool to visualize programming examples. In Proceedings of the EACT 88—Education and Application of Computer Technology, Malta, October 1988.
4. Gomes, A.; Mendes, A.J. SICAS: Interactive system for algorithm development and simulation. In *Computers and Education: Towards an Interconnected Society*; Ortega, M., Bravo, J., Eds.; Kluwer Academic Publishers: Dordrecht, The Netherlands, 2001; pp. 159–166.
5. Gomes, A. *Ambiente de suporte à aprendizagem de conceitos básicos de programação*. MSc Thesis, Faculdade de Ciências e Tecnologia da Universidade de Coimbra, Coimbra, Portugal, November 2000.
6. Esteves, M.; Mendes, A.J. OOP-Anim, a system to support learning of basic object oriented programming concepts. In Proceedings of the CompSysTech'2003—International Conference on Computer Systems and Technologies, Sofia, Bulgaria, 2003.
7. Rebelo, B.; Mendes, A.; Marcelino, M.; Redondo, M. Sistema Colaborativo de Suporte à Aprendizagem em Grupo da Programação—SICAS-COL. In Proceedings of the VII Simpósio Internacional de Informática Educativa, Leiria, Portugal, November 2005.
8. Redondo, M.A.; Bravo, C.; Ortega, M.; Verdejo, M.F. PlanEdit: An adaptive tool for design learning by problem solving. In Proceedings of the 2nd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH2002), Malaga, Spain, May 2002; pp. 560–563.
9. Redondo, M.A.; Bravo, C.; Bravo, J.; Ortega, M.; Organizing activities of problem based collaborative learning with the DomoSim-TPC system. In *Computers and Education: Towards a Lifelong Learning Society*; Llamas, M., Fernández, M.J., Anido, L.E., Eds.; Kluwer Academic Publishers: Dordrecht, The Netherlands, 2003; pp. 37–49.
10. Gomes, A.; Mendes, A.; Marcelino, M. Avaliação e evolução de um sistema de apoio à aprendizagem da programação. In Proceedings of the VII Congresso Iberoamericano de Informática Educativa, Monterrey, México, October 2004.
11. Areias, C.M.; Mendes, A. ProGuide: A dialogue-based tool to support initial programming learning. In Proceedings of the 3rd E-Learning Conference—Computer Science Education, Coimbra, Portugal, September 2006.

12. Marcelino M.; Mihaylov T.; Mendes A. H-SICAS, Handheld algorithm animation and simulation tool to support initial programming learning. In Proceedings of the 38th ASEE/IEEE Frontiers in Education Conference, New York, NY, USA, October 2008.
13. Felder, R.M. Learning and teaching styles in engineering education. *J. Eng. Educ.* **1988**, *78*, 674–681.
14. Rosati, P.A. Comparisons of learning preferences in an engineering program. In Proceedings of the 26th Frontiers in Education Conference, Salt Lake City, UT, USA, November 1996.
15. Constant, K.P. Using multimedia techniques to address diverse learning styles in materials education. *J. Mater. Educ.* **1997**, *19*, 1–8.
16. Paterson, K.G. Student perceptions of internet-based learning tools in environmental engineering education. *J. Eng. Educ.* **1999**, *88*, 295–304.
17. Rosati, P.A. Specific differences and similarities in the learning preferences of engineering students. In Proceedings of the 29th Frontiers in Education Conference, San Juan, Puerto Rico, November 1999.
18. Buxeda, R.; Jimenez, L.; Morell, L. Transforming an engineering course to enhance student learning. In Proceedings of the ICEE 2001 International Conference on Engineering Education, Oslo/Bergen, Norway, August 2001.
19. De Vita, G. Learning styles, culture and inclusive instruction in the multicultural classroom: A business and management perspective. *Innov. Educ. Teach. Int.* **2001**, *38*, 165–174.
20. Livesay, G.A.; Dee, K.C.; Nauman, E.A.; Hites, L.S.J. Engineering student learning styles: A statistical analysis using felder's index of learning styles. In Proceedings of the 2002 ASEE Conference and Exposition, Montreal, QB, Canada, June 2002.
21. Lopes, W.M. *ILS-Inventário de Estilos de Aprendizagem de Felder-Soloman: Investigação de sua Validade em Estudantes Universitários de Belo Horizonte*. MSc Thesis, Universidade Federal de Santa Catarina, Florianópolis, Santa Catarina, Brazil, 2002.
22. Kuri, N.P.; Truzzi, O.M. Learning styles of freshmen engineering students. In Proceedings of the 2002 International Conference on Engineering Education, Manchester, UK, August 2002.
23. Seery, N.; Gaughran, W.F.; Waldmann, T. Multi-modal learning in engineering education. In Proceedings of the 2003 ASEE Conference and Exposition, Nashville, TN, USA, June 2003.
24. Zywno, M.S. A contribution of validation of score meaning for Felder-Soloman's index of learning styles. In Proceedings of the 2003 ASEE Conference and Exposition, Nashville, TN, USA, June 2003.
25. *Moodle 2.0 Roadmap*. Available online: http://docs.moodle.org/en/Roadmap (accessed on 25 February 2010).