AGISA: an integrated system for classroom administration

Sónia Sousa Rogério Reis Luís Damas

Technical Report Series: DCC-2005-07

Departamento de Ciência de Computadores – Faculdade de Ciências

&

Laboratório de Inteligência Artificial e Ciência de Computadores

Universidade do Porto

Rua do Campo Alegre, 823 4150 Porto, Portugal

Tel: +351+2+6078830 – Fax: +351+2+6003654

http://www.ncc.up.pt/fcup/DCC/Pubs/treports.html

Abstract

Computers play an important role at all levels of education. Nowadays, its application scope is not limited to computer related classes as they can be used with great benefit in an wide range of subjects. It is not uncommon to see a teacher with poor computer knowledge, trying to use computers to improve students' interest in the subjects being taught, and thus improve their performance. The actual software panorama lacks tools to help teachers to use computers in an easy and straightforward way, that permit to control and monitor classroom work, to help building exercises, and the automatic verification and grading of the respective answers. The main goal of **AGISA** is to solve some of those problems.

AGISA pretends to be an innovative concept, bringing a computer "user friendly" framework for classrooms in an Free Software environment. The system was written in Python [3] for GNU/Linux systems and thus can be adapted and modified by everyone accordingly to his specific needs. Our system, GNU/Linux [1] and all the tools used are available without any cost or restriction under GPL [2], and this can be decisive for schools with a low budget.

AGISA provides the following main features:

- Creation and storage of exercises and checking and grading of their answers.
- Student and class data management (gradings, personal data, etc...).
- Control computers access to the Internet (filtering contents and sites or restrict all kind of access).
- Control student's computers and applications running on them.
- Allow a student to present its work to a group of other students using his own computer.
- Retrieve files and folders from student's computers.

The graphical interface tries to minimize the need of knowledge in the computers, "glueing" all the pieces of software that are available for GNU/Linux, hidding their cumbersome configurations and options, and giving an impression of a single "suite" of software, simple and easy to use. Our goal is to give to the teacher the opportunity to use features that were only available to experts in computer programming and system administration.

1 A general description

AGISA is mainly a patchwork of many other pieces of software, glued using Python in such a way teacher is not required to have a profound knowledge of the Linux system to be able to experiment all its potential.

All the information is stored in a relational database managed by a MySQL engine. The information kept includes classes composition as well as the grades of each student. Moreover, the system stores quizzes and the detailed description of each student performance and results. These data is written in XML files for which we definded XML dialects in order to save them in a human readable format. To parse and process this information we use XSLT stylesheets and its Python module support.

To improve the ease-of-use, the system provides a way to draw the layout of the classroom (*"Design classroom layout"* Module), by specifying the place of each computer. In this layout the system automatically identifies when a student logs in and associates his login to the corresponding computer representation. This allows to have a graphical interface where a set of students is easily selected, whenever it is required as a target for an action.

In the next sections we present the main features offered by **AGISA**.

Figure 1: Initial screen

1.1 Remote Control

The teacher can "spy" and control the computer (screen, keyboard and mouse) of each student.

1.2 Monitor Screens Module

Visualization of a thumbnail of screens is available, so the teacher can follow the work of several students at the same time.

1.3 Process Control

Any program used by the student can be monitorised and controlled. With a simple click, the teacher can terminate applications in all select computers.

1.4 Internet Access Control

Internet access can be constrained to a set of URL's either by giving a list of permitted sites or by forbidding a set of other pages. The teacher has the possibility to construct these lists interactively (navigating through the WWW) as well as by examining the logs of the Web visits of each student, selected by ranges of date. These actions can be applied to the whole class or to a set of students. Because all these controls use a HTTP proxy with cache, **AGISA** offers a kind of optimization of internet bandwidth thus avoiding its performance degradation.

1.5 Demonstration Module

The teacher can select a student to be the group leader. This student will take command of his own computer in order to complete a given task acting as the group tutor. All other members of the group can follow these actions in their own computer's screen.

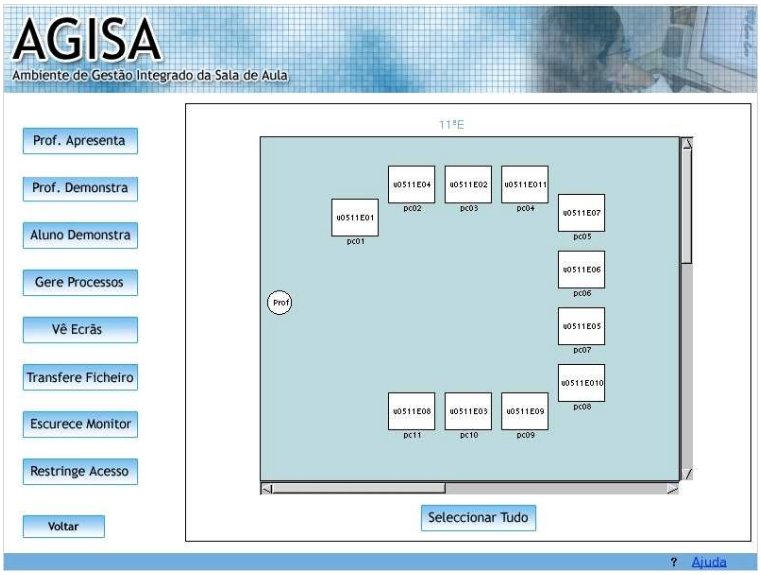

Figure 2: Classroom layout design

1.6 File Transfer

A selected set of files can be transferred to/from a set of students home directories. This feature can be used for download exemplification files necessary for training, or to upload the results. Fetched files are stored using filenames and paths and the data is organized by student, class, date and time.

1.7 "Turn-Off Screens" Module

To avoid any distraction caused by computer's screens, all screens in the class can be turned-off with a single action. All students' actions on the systems will then be blocked until a resume action is executed again. Screens are not really turned-off but the only blocked (by a `xlock` utility).

1.8 Presentation by the Teacher Module

As well as a student is assigned as group leader, the teacher can demonstrate some actions for a set of students. Moreover textual and sketches annotations can be added to the screen to make a presentation clearer. These annotations can be stored in a XML file, that can be latter used in similar sessions. Pauses can be specified between the drawing of each annotation to automate a presentation.

1.9 Quizzes

AGISA provides a graphical interface to assist the construction of quizzes. Quizzes' storage permits the creation of a repository library that can be shared between all system users. Quizzes can be of the following types: multiple choice, matching/ordering, gap-fill and crossword. After created and stored, quizzes can be made available for on-line submission. Because quizzes are translated to HTML, any Web browser can be used by the student

to answer these tests. After submission, automatic grading is performed and the grade is immediately presented to the student (if so desired).

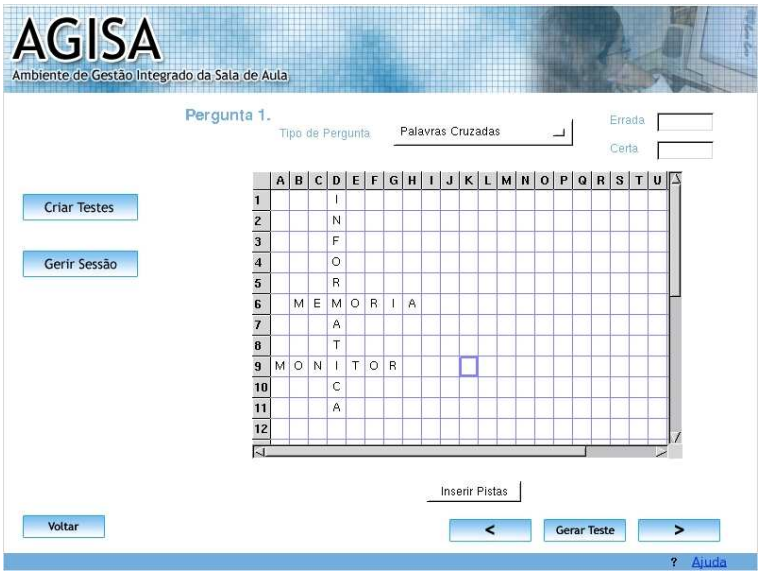

Figure 3: Constructing a crossword quiz.

To solve the issues related to student's identification, for each quiz and for each student a new password is generated. This avoids any possible exchange of passwords prior to the quiz session.

Student submissions are only allowed once. Exceptional situations can be contemplated and allowed expressly by the teacher.

1.10 Help System

A comprehensive contextual help system is available in all situations.

2 About the Software Used

AGISA is entirely written using Free Software. Apart from the philosophy of Free Software that we subscribe, its great advantage is to have permanent bug corrections, updates and features added by its vast community of users that will be automatically available for the **AGISA** users. The fact that Free Software is cost free and naturally available with the source code permits the software experimentation and/or its adaptation/customization. This may be a decisive difference from the proprietary systems. In this spirit we did not try to write a new complete application, but instead to "glue" various programs, libraries and utilities to create an intuitive and innovative environment for the classroom.

Visualization of the student's screens, demonstration of actions, and visual aid presentations rely on an implementation of VNC, `x11vnc`. VNC [6] is a protocol for remote access that uses TCP/IP and client-server architecture. Both the client and server may reside in different computers, architectures and operating systems.

When the teacher needs to use some resources of the graphical environment of a student, some kind of authentication is needed by the X11 server. **AGISA** uses the MIT-magic-cookies mechanism to accomplish this. The passwords are transmitted via a cryptographic

channel to difficult any eavesdropping or illegal intrusion. These secure channels are created with the use of `ssh` [5]. `ssh` is client-server TCP/IP service that creates a channel using Public Key Cryptography, through which client applications can communicate with X11 servers.

For file transfer rsync is the only program used. This ssh utility provides file copying using the same authentication mechanisms of ssh conferring to these actions the same cryptographic security. Moreover there is no waste of unnecessary bandwidth, all data is compressed/decompressed "on the fly" and existing files are not retransmitted.

Internet access control is implemented using a the `Squid` [4] service proxy. In all student's computers, IP routing is configured such that way no direct connection is possible. All external connections must be accomplished using a proxy — the teacher's system. In the teacher's system, control can be configured by the `Squid` own access lists.

Quizzes are presented to students, and answered by them, through a dynamic Web page using CGI's.

3 A Sample Session with AGISA

Suppose we want to use **AGISA** in a class about the importance of spices trade in the XVIth century in Europe's History.

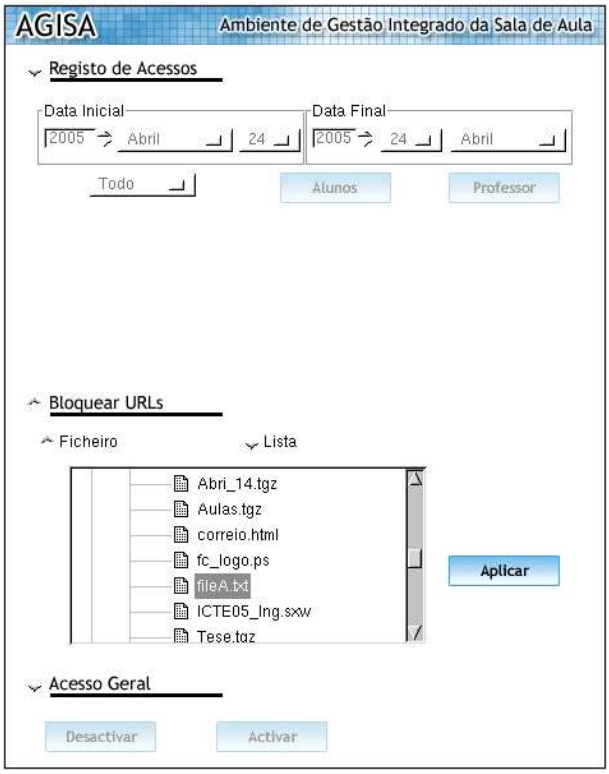

Figure 4: Internet access control

Resources needed are: 20 computers (one for each student), access to a printer, and a set of electronic slides to make a presentation on the topic.

Beforehand the teacher preparing the class took the following steps:

- Using the *"Design classroom layout"* module he/she made an approximate layout of the computer's distribution in the room and assigned to each system its IP name.
- Created with a text editor (`emacs`, for example) a file (file A) describing the unwanted URLs that students normally visit, using the complete URLs or regular expressions.
- Wrote a quiz with questions about the topic. The quiz has mix of multiple choice and gap-fill questions.
- Created another file (file B) describing the unique Web pages to be allowed during the quiz session.

A class interaction with **AGISA** can be described in the following way:

1. **AGISA** system is launched.
2. Using *"Access restriction"* module the teacher applies file A rules to all the students.
3. The whole class is selected for target and the *"Demonstration by the teacher"* module starts to show the pre-prepared presentation to all the classroom.
4. The teacher opens mozilla to use google to search for "Vasco da Gama", showing these actions and its results to all students; and suggests some Web links that should be used to proceed a research on the subject.

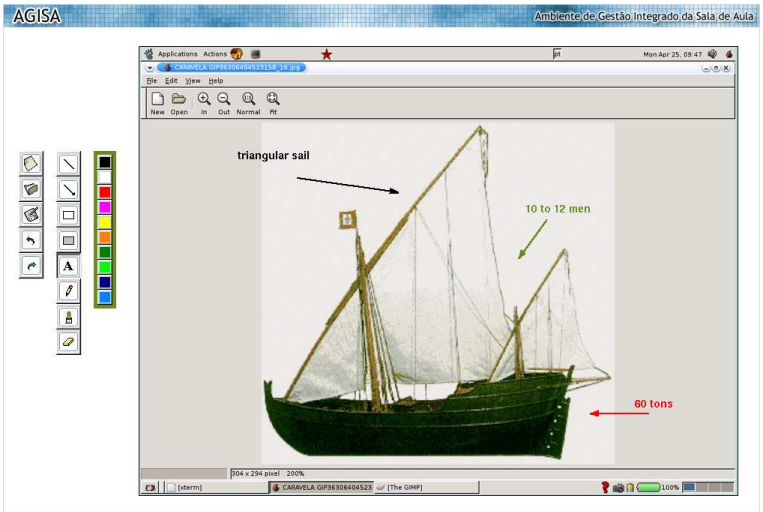

Figure 5: Presentation by the teacher

5. End of demonstration;
6. Initiates a quiz session, generates and prints a set of passwords for this quiz.
7. Gives to each student his/her correspondent password;
8. Applies file B as restrictions rules to Web use.
9. During the quiz session, the teacher can follow students' work (using the *"Monitor Screens"* option) and can examine their logs in order to have an idea of the extent of their work.

10. Quiz session ends.

11. Realizing that some of the students had special difficulties in some aspects, the teacher chooses one of the students (without those difficulties) to lead a clarifying Web search to be followed by all the others (*"Demonstration"* module).

12. Turned off all screens (*"Turn-Off Screens"* module) and started a reading period on a text book on the subject;

13. Using an image of a caravel, the teacher begins to annotate the image explaining its various parts and its functions (*"Presentation by the Teacher"* module). These annotations can be saved, to use in future sessions.

14. Ends this presentation.

15. All the students are asked to make a small composition about XVI sea vessels and to store it in a file called "`composition1.txt`".

16. Using *"File Transfer"* module, the teacher fetches all the compositions requested to be graded later.

4 Conclusions

The system is being tested in two portuguese high schools: Escola EB3/Sec Professor Reynaldo dos Santos — Vila Franca de Xira and Escola EB3/Sec José Régio — Vila do Conde. From this set of tests we aim to estimate the advantages and disadvantages of the use of this kind of software in the classroom, what are the weak-points and what features we still need to implement, and to understand what is the acceptance of such system both among teachers and students.

We are specially interested to understand in what extend the use of a system like this can constitute a constraint to these student's freedom, not allowing the discovery of the computer system by himself, or if by other hand, the use of a program like **AGISA** although enforcing the focus of the student's activity in the class and avoiding some of the more disturbing activities gives to the student a computer system that he "owns".

After this early test stages we plan to make the system publicly available in a web site. With the growing of the users' community we expect to have "localized versions" (versions translated for the various languages) that permit a wider scope of its use.

Free Software movement and its influence in high schools is at its very early stages and many still look to this kind of software with disbelief. We think that this kind of initiative can have a positive impact on the Free Software image among teachers showing its possibilities, usability and competence, and thus constituting a contribute to its expansion.

References

[1] Gnu. http://www.gnu.org.

[2] Gnu public licence. http://www.gnu.org/copyleft/gpl.html.

[3] Python "official" web page. http://www.python.org.

[4] squid. http://www.squid-cache.org.

[5] ssh. http://www.openssh.com.

[6] vnc. http://www.realvnc.com.