International Journal of Engineering Education Vol. 32, No. 2(B), pp. 873–885, 2016
Printed in Great Britain

0949-149X/91 $3.00+0.00

OpenIRS-UCM: an Integral Solution for Interactive Response Systems*

CARLOS GARCIA, FERNANDO CASTRO, JOSE I. GOMEZ, DANIEL CHAVER and JOSE A. LOPEZ-OROZCO
Department of Computer Architecture, Computer Science Faculty, University Complutense of Madrid, 28040 Madrid, Spain.
E-mail: {garsanca, fcastror, jigomez, dani02, jalo}@ucm.es

During the last years many different tools have appeared devoted to facilitate the communication between teachers and students in a classroom. Most of these solutions, commonly referred to as Interactive Response Systems, are closed, too rigid, and in many cases too expensive for educational institutions. In this work we present an extended and improved version of the OpenIRS-UCM application, a free, easy to use and open source tool which supports diverse clickers, can be run in different platforms (Windows, Linux and MacOS) and is totally integrated with the well-known Moodle Learning Management System. We evaluate the OpenIRS-UCM application in terms of various software quality metrics (program complexity and usability), the level of acceptance currently achieved and its potential adoption in real life settings. Also, we perform an exhaustive comparison of the features that our application provides to those of various commercial tools, which reveals that OpenIRS-UCM exhibits many advantages over other major IRS players. Finally, we detail how our tool represents a remarkable advance in the research areas of engineering learning systems and engineering assessment.

Keywords: cooperative/collaborative learning; distance education and telelearning; improving classroom teaching; interactive learning environments.

1. Introduction

Student-Centered Instruction (SCI) is a novel teaching approach in which the student is placed on the center of the learning process. According to this scheme, a student can influence the content, activities and materials of a subject—and even its pace of learning—providing valuable feedback to the teacher and to the other students. Several instructors have developed and used approaches that fit the criteria for student-centered learning, such as Peer Instruction [1], Cooperative Learning [2], or Active Learning [3]. In this context, teacher-student real-time interaction in the classroom becomes essential.

In the era of the new technologies, one way of improving such interaction is through an Interactive Response System (IRS), a learning technology which allows the collection, the transfer and the display of students' responses in a classroom. Many researchers have evaluated the effectiveness of IRSs [4–6] and have described in detail their features and advantages [7, 8]. This kind of systems has been developed as a means to achieve student learning in different educational levels by meeting many educational needs. Notably, the IRSs allow:

- The evaluation of the students (and also self-assessments).
- A better interaction between students and teachers, especially in big groups.
- Debates between the students (a key element for the understanding of contents).
- To increase the motivation of the students.
- To improve the *teaching plans*.

IRS tools are usually packaged as integral hardware/software solutions. The IRS hardware includes a personal hand-held signal transmitter called *clicker* and a response signal receiver that is connected to the teacher's computer. The transmission medium can be the Internet (wired or wireless), radio or infrared. The IRS software is installed on the teacher's computer and it provides the following basic functionalities: collecting all responses within a time interval, storing and retrieving the individual responses, and generating reports.

The market has been flooded of commercial products targeting these systems; a survey can be found in [9]. Some companies like *Promethean* [10] (with ActiVote, ActivExpression2), *eInstruction* [11] (with CPS-IR or CPS Spark) or *SunVote* [12] (with devices as W40, W52 or M50) are mainly focused on *ad-hoc* proprietary hardware clickers, i.e. dedicated devices designed *ad-hoc* to communicate user answers. However, as many students currently have latest generation mobiles (like iPhone or Android based terminals) and also free WiFi connection is available in almost all schools and universities, the use of phones as clickers has become a flexible and inexpensive alternative to proprietary clickers [13]. Many companies (e.g. *SMART Technologies*, *eInstruction*, *i>clicker*, *H-ITT* or *Turning Technologies*) have recently adapted their products in order to allow smartphones to be used as clickers.

* Accepted 1 September 2015.

Also, it is easy the usage of laptops, PDAs or tablets in the same fashion.

In addition to all these commercial systems, there are also open source initiatives like the *Interactive Learning Toolkit-BQ* software package [14] which allows to combine several commercial clickers, laptops and phones with different browsers, or *Open SmartClassroom* [15] and *eRoster* [16], engaged in distance learning and which improve the interactivity of active learning. In the hardware side, *InfoCoral* [17] designs a hybrid low-cost open-source hardware to implement a clicker based on ZigBee technology. Generally, these solutions are less powerful than the commercial counterparts, but they open an interesting path for extremely low-cost solutions which could make affordable the widespread introduction of learning technology in the educational system.

In this work we focus on how the Model-View-Controller (MVC) architecture allows OpenIRS-UCM [18] to keep most of the key benefits of commercial products, removing at the same time part of their undesirable complexity. Most IRS products mentioned above provide lots of options regarding configuration, data analysis and question types. While having such a broad functionality can be viewed as an advantage, it also turns the IRS software too complex to use. In our opinion, most teachers will prefer a simpler tool like ours, which effectively performs the common day-to-day features and still includes most of the commonly required functionalities. In fact, the data (model) and the GUI (view) split favors to develop an IRS tool with all potential functionality although it presents different views. The simplest view includes most of the commonly options used by most teachers. However, it permits to change the view module and get other tools with more functionalities or new applications. Furthermore, the controller module allows the integration of almost any kind of clickers (provided that the vendor supplies the necessary dynamic libraries and documentation): both *soft-clickers* (any device, such as a PC, PDA, laptop, smartphone, etc., which, with the mediation of a software [19, 20], can send information to the application), and *hard-clickers* (a remote device conceived specifically for Interactive Response Systems, such as SunVote or H-ITT clickers).

Just as IRS tools are becoming more and more popular, *Learning Management Systems* (LMS) are also gaining much attention, and their usage is expected to steadily grow in the incoming years [21]. Blackboard, Moodle or Sakai are just a few examples of the many existing LMSs widely used across thousands of teaching institutions with millions of users through the world. Thus, beyond traditional IRS features, companies like *i>clicker*, *H-ITT*, *eInstruction* or *Turning Technologies*, and even some open source tools like the *Interactive Learning Toolkit-BQ* software package, provide support for integration with this kind of systems. Following this interesting trend, a new view is implemented in OpenIRS-UCM in order to present teaching results in the Moodle LMS at the University Complutense of Madrid (9466 and 9676 courses in the 2012-2013 and 2013-2014 academic years respectively) and the leader open source LMS regarding the number of users, even competing with Blackboard, the main commercial LMS. In this way, the integration with Moodle is carried on, which provides many features such as automatically importing students IDs to the roster file or exporting and analyzing the collected answers to Moodle.

Also, the specific features of OpenIRS-UCM provide new opportunities to advance in the field of engineering education. Notably, this tool improves the engineering learning systems research area by achieving a high level of use and acceptance at the University Complutense of Madrid, by being employed as its official teaching test program during the next years and also by allowing new scenarios for teaching activities. Furthermore, essentially due to its ease to use, OpenIRS-UCM also advances in the research area of engineering assessment since it postulates as a major IRS contender for be employed in evaluation activities. Although it is not the main novelty, OpenIRS-UCM, as other IRS, is basically used to favor the development knowledge and competencies for most subjects.

In this paper we extend and improve the OpenIRS-UCM software presented in [22]. Notably, the current work improves the previous one, among others, in the following main aspects:

- We have extended and improved the introduction section from an educational technology point of view.
- We deeply describe how OpenIRS-UCM has been developed by means of the Software Technology.
- In order to make OpenIRS-UCM more modular, it has been developed a simple TCP communication wrapper for third-party keypads.
- A Moodle-XML format file has been incorporated to interconnect OpenIRS-UCM with Moodle LMS.
- We include a new evaluation section, where we evaluate our OpenIRS-UCM according to different software quality metrics and also perform and extensive comparison with other IRS applications.

The rest of the paper is organized as follows. Section 2 presents a general description of OpenIRS-UCM. Then Section 3 provides deep functional details

about the system. Section 4 evaluates and compares our tool to other commercial systems. Finally, Section 5 concludes.

2. Material and methods

This section is divided into four parts. Section 2.1 briefly summarizes the OpenIRS-UCM works and its different possibilities. Section 2.2 describes our tool regarding the underlying software architecture. Section 2.3 details the module that allows the connection with Moodle. Finally, Section 2.4 shows how to get our application.

2.1 OpenIRS-UCM overview

This application comes with the main goal of collecting and managing the opinions of the audience in a class. The application defines the term *session* to refer to an entity that groups all information related with a quiz. Prior to the class, the teacher may fill in this information, that entails some tasks like assigning a communication mechanism to each student, recording it properly, adding a group of questions including the right answers, etc.

Once in the classroom, the teacher starts a quiz and the system manages the collection of the students' answers. OpenIRS-UCM permits a real-time feedback by means of several chart types that reflect the audience opinions.

The application is also considerably flexible in extracting the information. As such, after the lecture, the teacher can easily export all collected data to different formats, like Microsoft® Excel, *CSV* or *XML-Moodle* (this format is supported in order to permit the interaction with the Moodle LMS) files, making it possible a proper processing of these data.

2.2 Software architecture

Figure 1 shows the software building blocks of OpenIRS-UCM. We followed the *Model-View-Controller* (MVC) pattern to design the system, which eases the task of maintaining and extending the tool functionality.

Currently, there are three *View* modules:

- An LMS module integration supposes a view which allows to widespread the answers into a Moodle LMS (see section 2.3).
- The *GUITeacher* includes the graphical support to allow that the input from the teacher could be collected through the *GUI* (more detail explanation and an example is reported in section 3). It is the only input source for the teacher, and it may trigger commands to the communication or modify the *Model*. Moreover, the *GUI* also allows representing graphically parts of the state of the *Model*: the current students' answers, charts of accumulated results, etc.
- Finally, a new view has been developed to poll the students into the class in order to collect the students' opinions about the teachers teaching task. As a result, the University Complutense of Madrid (UCM) has incorporated this tool named IRS-DOCENTIA [23] in its teaching quality program.

The *Model* corresponds to the *DB* module, which contains all the storage related functionality. OpenIRS-UCM uses *SQLite* [24] as database management system to keep data from current session, isolating the core from the input/output format desired by the user. The *DB* module provides an interface to query the session database. Currently, the session data can be imported/exported in proprietary *Microsoft® Excel* or *CSV* formats. As

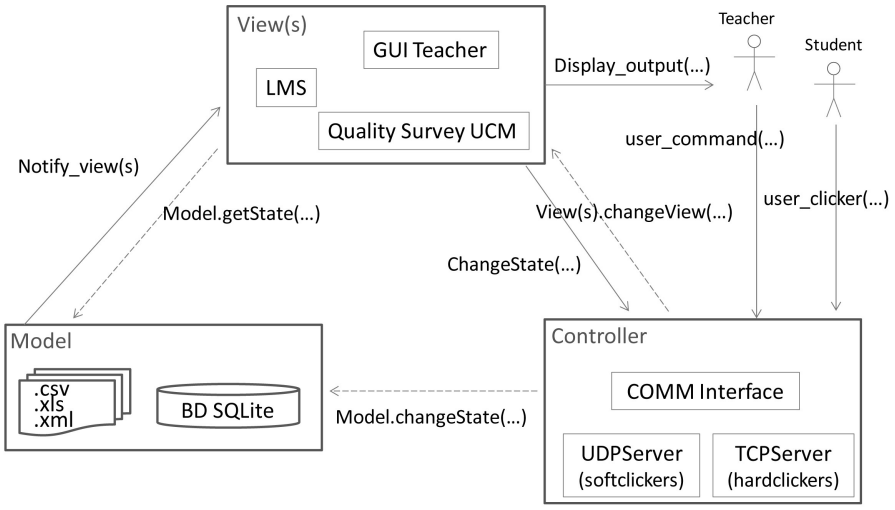

Fig. 1. OpenIRS-UCM software modules.

stated above, OpenIRS-UCM is also able to export the model in *XML-Moodle* format. This format is especially useful in teaching environments since it eases the task to import/export data from/to course management systems in other get a new view to show the results of students. So, it allows to easily extending the analysis of the collected data using the statistical tools that Moodle supports. Section 2.3 describes the Moodle module delivered with OpenIRS-UCM which further eases the data interchange between these applications.

The *Controller* is represented by the *Control* block, which works as the backbone of the system, interconnecting the other modules: it manages timing, queries the database for questions, activates the remote devices to start accepting answers and dispatches user orders to the corresponding module.

The *Communication* module encapsulates all the functions related with the remote devices. Currently OpenIRS-UCM supports third-party H-ITT and SunVote remote keypads and we have implemented UDP clients for PC and mobile phones (Android and iPhone) that work as soft-clickers. Regarding third-party hardware clickers, any could be ported to OpenIRS-UCM if the vendor provides dynamic libraries and documentation. We developed a Java interface that must be implemented by any new clickers to be introduced in the system. All the featured devices may work concurrently, and all the answers are collected in a single answer queue. Finally, the *Communication* module also declares an interface to be used by the *Controller* module to activate all connected devices, start/stop polling and fetch answers from the queue without bothering of the currently active devices.

The OpenIRS-UCM software [25] has been released under the GNU Lesser General Public License (LGPL), which gives the freedom to use, share and modify the software. Some third party libraries were needed to support hardware clickers from different manufacturers, which usually do not provide open source drivers. Most manufacturers provide a C dynamic library as driver. To use these C drivers from Java, we created appropriate wrappers using the Java Native Interface [26].

Supporting the H-ITT clickers requires the copy righted and closed driver from the manufacturer [27]. The driver provides a virtual serial port that can be used to manage the clickers. To simplify the communication with the clickers, OpenIRS-UCM makes use of the RXTX [28] Java library, released under LGPL [29], which provides basic serial and parallel communication for the Java Development Toolkit (JDK).

SunVote devices were also supported through the RXTX library. The manufacturer provides an USB reception device that shows up in the system as a virtual serial port. We developed the communication protocol on top of the RXTX services, using closed documentation from the manufacturer. Therefore this part of the software is released as a separate executable file with copyright. Disclosing this file would need permission from the manufacturer.

The *Communication* module also contains a UDP server which virtually allows the implementation of soft-clickers in any mobile platform. Section 2.2.2 describes the soft-clickers currently implemented.

Moreover, the TCP server represents a second alternative to support third-party clickers by rightly employing the vendor software to capture the remote keypads answers and forwarding them to our TCP server, thus effectively transforming clickers into soft-clickers.

Third party libraries were also needed in the other modules:

- Apache POI Java API [30] to manage Microsoft documents. It is composed of several modules for the different Microsoft file formats. We basically use the HSSF module for the traditional XLS Microsoft® Excel file formats. POI uses Apache license [31], which grants you the possibility to distribute your software as open or closed source.
- JFreeChart [32] to display charts with the statistics of the audience responses. It is a Java library distributed under LGPL.
- JDOM [33] to read and write XML files. JDOM is complete Java-based solution for accessing, manipulating and outputting XML data from Java code. It is distributed under its own JDOM license, which is an Apache-style open source license, with the acknowledgment clause removed.
- OpenCSV [34] to read and parse *CSV* format files. It is available under Apache 2.0 license.

The multi-platform installers provided in the OpenIRS-UCM distribution [25] contain all the libraries required by the application to work except for the Java Virtual Machine (JVM). OpenIRS-UCM has been successfully tested in Microsoft® Windows, Linux and Mac Os X based systems with a 32-bit JVM. The developed code also works under a 64-bit JRE version, but some of the used libraries may not. Note that the only limitation is about the JVM version and does not affect the system itself, which may perfectly be running a 64-bit operating system.

2.2.1 Supporting more third-party clickers

As mentioned above, OpenIRS-UCM provides a common interface to port new third-party devices to the system by loading the provided dynamic libraries. H-ITT and SunVote clickers porting,

delivered with Open-IRS-UCM, are two examples on how to proceed with any other device.

In order to ease the integration of further clickers, we have also implemented a second path to port them onto OpenIRS-UCM. We include an *ad-hoc* TCP server in our *Communication* module which will listen to independent TCP clients running a wrapped modified version of the vendor binaries. OpenIRS-UCM defines a protocol and a required interface for external applications to attend this bidirectional communication channel. As an example, given the *OCX* file provided by SunVote, it is immediate to build a VisualBasic application to rightly configure and control the clickers using the interface provided by the company. Then, it is straightforward to wrap that code with the required TCP client functionality, resulting in a new independent application that, when executed, will connect to the OpenIRS-UCM TCP server. This connection allows bidirectional communication: it is possible to forward the answers from the clickers to OpenIRS-UCM and to configure the keypads from the IRS application too. Currently, to include a new clicker is as simple as adding a new entry to the *clickers.xml* file, specifying the independent application supported.

2.2.2 Provided soft-clickers

OpenIRS-UCM distribution includes the *Teletest* client, which allows laptops, PCs, PDAs and other similar devices to work as soft-clickers by connecting to the UDP server. We have developed UDP Java clients for several platforms, including Apple devices as iPhone and iPad[1]. Notably, our *Teletest* for PCs, laptops, PDAs, Apple iPhone and iPad is denoted as *iTeletest* [20] whereas *uTeletest* [19] enables that smartphones and tablets based on Android may be used as soft-clickers.

OpenIRS-UCM allows selecting the desired UDP port. Once activated, the UDP server runs as a separated thread, collecting answers from all the connected soft-clickers which are subsequently inserted in the global answer queue.

Figure 2 illustrates our *iTeletest* running on an Apple device—iPhone—as well as our *uTeletest* running on a smartphone based on Android. These ones implement a basic version like hard-clickers in order to show how the clickers (soft and hard-clickers) can be used together.

However, we must note that hard-clickers are rigid and closed since they only allow those functions provided by the manufacturer. On the other hand, as soft-clickers can be programmed, they can be extended with new functions, such as receiving

[1] This application is freely available in the AppStore, so it can be downloaded by any user of this kind of devices.

(a) (b)

Fig. 2. iTeletest (a) and uTeletest (b).

polling questions on its screens, or receiving reports about both the sent answers and the correct answers. Moreover, OpenIRS-UCM may be customized (recall that our tool is open source), allowing it to cover a wide spectrum of possibilities.

2.3 Quizopenirs Moodle module

One of the most popular LMSs nowadays is the Moodle platform [35], with 67 million users and more than 68000 sites. In particular, Spain is the second country in terms of the number of registrations. In our institution, the University Complutense of Madrid, Moodle constitutes the most demanded LMS. Thus, quizzes are used in 434 courses during 2012–2013 academic year and 417 courses in 2013–2014 academic year, with multichoice questions representing respectively the 72.6% and 72.8% of all generated questions which demonstrates the potential use of OpenIRS-UCM. The rest of the questions corresponds to other Moodle's question types as short answers, description, numerical, and so on.

Moodle is a complete and flexible platform which, among other features, allows the incorporation of self-made modules. Specifically, we have developed the *quiz-openirs* module, following strictly the specifications described in the Moodle guidelines for contributed code [36], to interconnect the Moodle 1.9.9 version with our OpenIRS-UCM tool because it was the platform operative in UCM (the Moodle 1.9.x version is the most used by far [37]) when this module was developed.

The new module allows the teacher to create quizzes of different question types such as multiple-choice, true-false or short answers. Notably, the behavior of the multiple choice quiz type resembles the polling method employed in the OpenIRS-UCM tool.

The interconnection between Moodle and Open-

IRS-UCM is performed by means of two XML files generated from the OpenIRS-UCM application: one including information about questions, grades and penalization rates, and another one which stores the students' answers. These two files are imported into the *quizopenirs* module, which performs the following tasks: (1) creates a new entry in the *question bank*, (2) creates a new quiz to which it attaches the questions, (3) loads the students' answers, and (4) grades the quiz, and closes it.

This functionality is very useful from both students' and teacher's point of view: the students may check their answers, which lets them know their own progress and knowledge acquisition. Moreover, the teacher can exploit the broad diversity of tools provided by the Moodle platform.

Going inside the internal architecture, the installation of this new module just involves the creation of a few new entries into the internal SQL Moodle database, which implies no interference with its architecture. This aspect is pivotal in order to keep the system integrity. For example, if the administrator wanted to remove the module, the uninstallation process would be as simple as with other module of Moodle, not affecting to any retrieval process or the system backup.

2.4 Where to get it?

We have uploaded two videos to YouTube, one with a short presentation of our tool [18], and another with a detailed tutorial of how to use OpenIRS-UCM [38]. We have also created a repository [25] in order to facilitate the distribution of our application, which includes:

- A collection of installable files (with the necessary libraries) for different operating systems.
- A collection of soft-clickers which emulate hardware clickers through a software application and Internet connection.
- The OpenIRS-UCM source code, which allows the users to further develop the tool or just to adapt it to their own environments.
- The source code of the *quizopenirs* module, to allow its full installation in a Moodle platform.
- A video-tutorial with a complete guided example.
- Example files that simplify the configuration process.

3. OpenIRS-UCM operation

Figure 3 outlines the process of collecting the students' opinions with OpenIRS-UCM, which is divided into three stages. In the next subsections, we briefly describe each of these stages. More extensive explanations can be found in the video tutorial [38] provided along with the OpenIRS-UCM package [25].

3.1 First stage: configuration task

Prior to the class, the teacher must configure the Open-IRS-UCM tool. As we said before, the application organizes tests into *sessions*. Therefore, the teacher should first create a new session, and then fill in the associated information, like inserting the questions, determining the connection type and the clickers that will take part in the test, or defining the operating mode. Concerning this last feature, our tool provides two different modes: *Anonymous*, where no student identity is stored, and *Conventional*, where each clicker is linked to a specific student (the association between students and clickers can be defined with a file, prior to the class, or

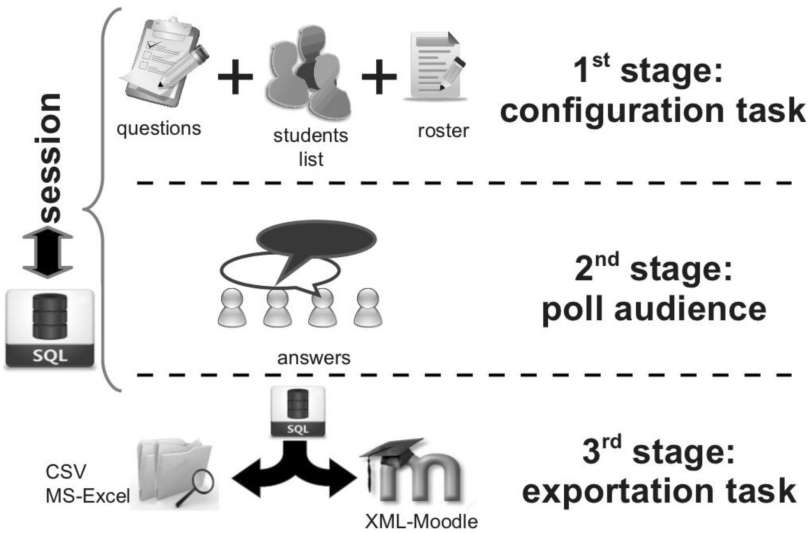

Fig. 3. Stages involved in the process of collecting students' opinion.

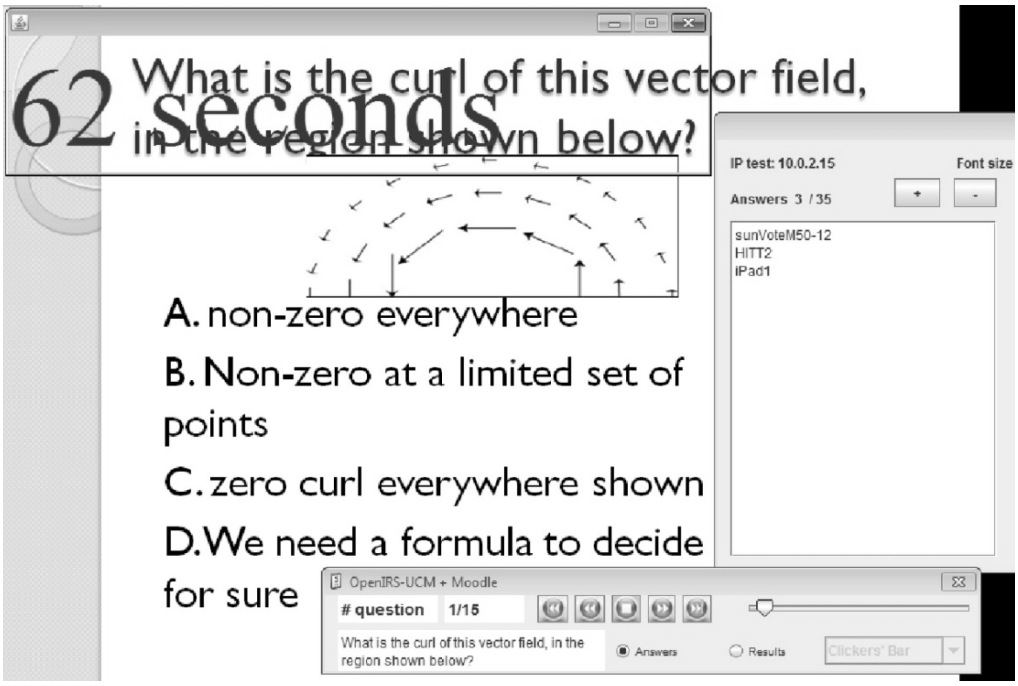

Fig. 4. Example of a test execution.

with the *Real Time Registration* tool, once in the class).

3.2 Second stage: poll audience

Once the configuration task is completed, the teacher is ready to poll the audience opinion. Figure 4 illustrates an example where the first question of a test is being performed, the students with the clicker identifier *sunVoteM50-12, HITT2* and *iPad1* have already sent an answer, and there are still 62 seconds left. In this case, a *Microsoft® PowerPoint* presentation illustrates on the background the statement for the aforementioned question and the 4 suggested answers. However, we should emphasize that OpenIRS-UCM is totally independent of the teaching method employed and it can use any slide-oriented application (*Microsoft® PowerPoint*, *LibreOffice*, LaTeX, etc).

If the teacher wants to display the responses of a question during the class, the *Results* option can be used, which provides different graphical formats: Clickers Bar, Answers Bar, Answers Pie, and Yes/No Pie.

3.3 Third stage: exportation task

Once the test has been completed, the teacher can export the information associated with a session to *Microsoft® Excel*, *CSV* or *XML-Moodle* files. This allows the teacher to perform a deeper data analysis. For example, interacting with Moodle (through the *XML-Moodle* files), allows the teacher, among other things, to easily extract some interesting aspects of the test, like the average mark, whether there exist questions with special impact, or if the repetition of the test is advisable in order to obtain a better evaluation of the students progress.

4. Results and discussion

In this section we first evaluate our OpenIRS-UCM tool according to different selected metrics and then we compare our software to various commercial IRS tools.

4.1 Software quality metrics

Notably, we first evaluate our application regarding a couple of software quality metrics: the *program complexity* and the *usability*. The software quality metrics are employed in the software engineering field in order to report a quantitative measure of degree to which a software possesses a given attribute. The motivation of this kind of metrics is varied: estimate the cost of future products, evaluate the productivity impact of new tools, improve software quality, etc. Among all existing software quality metrics we focus on program complexity and usability for different reasons: by one hand, being OpenIRS-UCM a open source tool, it is essential to guarantee an easy extension of the software by adding new functionalities, which requires that our code was as simple as possible, e.g. the software must exhibit a reduced program complexity; by other hand, being the ease of use one of the most appealing features of our tool it seems to

be obvious to evaluate the experience of new users when they deal with our software.

4.1.1 Program complexity

As for evaluating the program complexity of the OpenIRS-UCM code, we choose the widespread used *McCabe's Cyclomatic Complexity* (MVG) metric, which is a measure of the decision complexity of the functions which make up the program. Furthermore, MVG is a quantitative measure of testing difficulty and an indication of ultimate reliability.

The methodology we use to determine the MVG of OpenIRS-UCM is the following: we leverage the CCCC tool [39], a metric analyzer for Java, C and C++ codes, which reports an extensive set of metrics related with the code of the application under evaluation. Thus, we run CCCC upon all the source files integrating the OpenIRS-UCM code to extract the derived MVG value.

Experimental data from multiple studies have revealed that the MVG value should be no more than 10 to keep the program complexity in acceptable values and, according to the results reported by the CCCC tool, the OpenIRS-UCM's average MVG per module is just 3.85, which confirms that the source code of the current version of OpenIRS-UCM is still far enough from the conflicting value, so that testing difficult can be considered as moderately low.

4.1.2 Usability

As for evaluating OpenIRS-UCM's usability we employ the well-known and widespread used System Usability Scale (SUS) [40]. The SUS is a 10-item questionnaire used after the respondent has had an opportunity to use the system evaluated, but before any debriefing or discussion takes place. According to SUS guidelines, respondents should be asked to record their immediate response to each item, rather than thinking about items for a long time.

The methodology we follow to evaluate the OpenIRS-UCM usability employing SUS is as follows: first we randomly contact with several teachers (29) from different schools (different research areas) of the University Complutense of Madrid and we invite them to use OpenIRS-UCM to perform a simple test over the corresponding subject's contents during one of their classes. Second, once the test has been performed, the teachers respond to the SUS questionnaire we provide them using a Google form. Third, we score each of the questionnaires according to SUS guidelines [40] in order to obtain the final results.

Note that we considered as enough the amount of samples collected (29) since, according to [41], SUS is the questionnaire that converges most quickly to the *correct* conclusion when the number of samples increases, making the correct decision over 90% of the time when the number of participants is 12 at least.

Figure 5 shows a histogram with the distribution of the 29 SUS scores obtained when using OpenIRS-UCM. We must note that the feasible SUS scores range from 0 (minimum usability) to 100 (maximum usability). We observe how most of the teachers who tested our tool deliver a high value of usability for our software. Notably, 16 out of 29 users report an usability value of 80 or higher and 22 users obtain SUS scores of 70 or higher. The mean value derived from the current data set is 77.7 which confirms that the OpenIRS-UCM tool can be considered as easy to use.

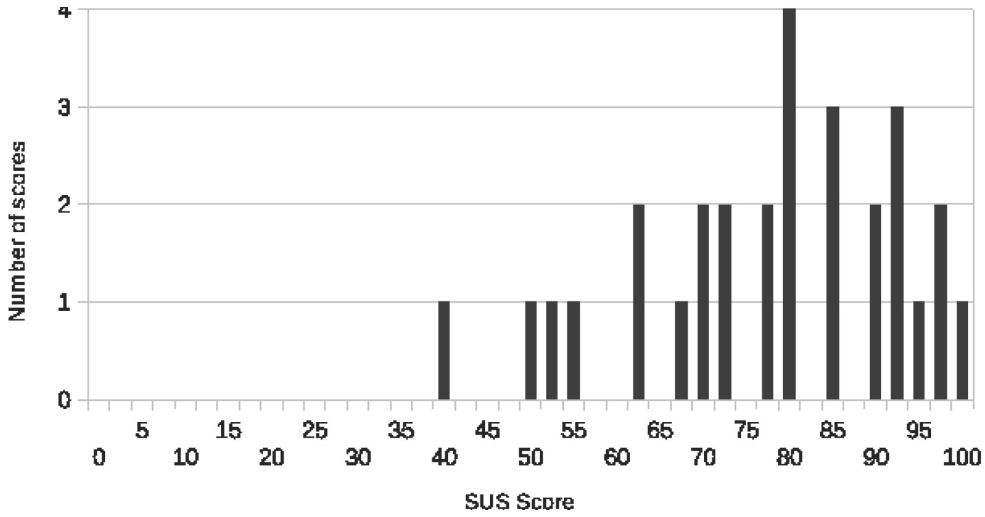

Fig. 5. Distribution of SUS scores.

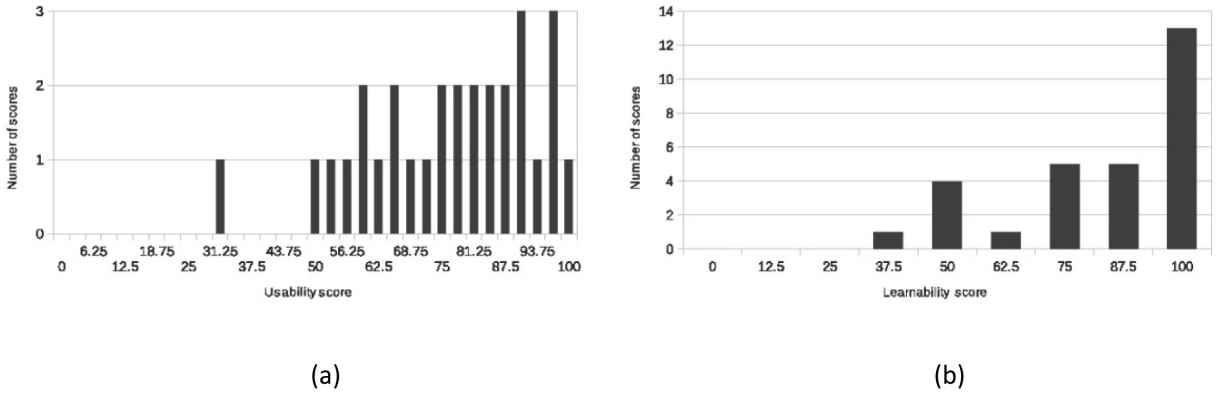

Fig. 6. Distribution of Usability scores (a) and Learnability scores (b).

In addition, in [41] the authors suggest that the SUS actually can be decomposed into two factors: *Usability* (8 items) and *Learnability* (2 items). The Learnability factor is related with the effort required to *learn* the usage of the tool and includes items 4 and 10 of the SUS questionnaire: "I think I would need the support of a technical person to be able to use this system" and "I needed to learn a lot of things before I could get going with this system" respectively. Figure 6 (a) and (b) show the distributions of the Usability and Learnability factors, which again report high average numbers (76.8 and 83.2 respectively). Note that, for instance, in item 4 of the SUS, 16 out of 29 teachers chose the minimum score[2]—1 point—and other 8 selected the next score—2 points—which clearly suggests that teachers, independently of their particular research field, can easily use our OpenIRS-UCM on their own.

Finally, in order to complete the usability analysis, Table 1 shows basic statistical information about the distribution of SUS scores, usability and learnability.

4.2 Acceptance

Although the acceptance and the current inclusion of an IRS software in the teacher workflow is not easy to quantify, we must highlight that, although the scope of our tool is still somehow limited, several pilot tests using OpenIRS-UCM have been performed successfully in 24 out of 26 schools in the University Complutense, which has lead the institution to purchase a high amount of clickers in order to satisfy the growing demand for using our application. In fact, our University has incorporated OpenIRS-UCM the next years for its official teaching test program.

[2] The possible scores for each question are integer numbers ranging from 1 (i.e. strongly disagree) to 5 (i.e. strongly agree).

Table 1. Basic statistical information about the SUS, usability and learnability distributions

	SUS	Usability	Learnability
N (number of samples)	29	29	29
Minimum	40	31.25	37.5
Maximum	100	100	100
Mean	77.7	76.8	83.2
Variance	242.83	282.87	393.70
Standard Deviation	15.58	16.82	19.84
Standard error	0.82	0.56	2.53
Median	80	78.13	87.5

4.3 OpenIRS-UCM vs. other solutions

In this section we qualitatively compare our tool to other open-source or commercial systems, exploring different features.

4.3.1 Overview

Here we provide a first insight of different IRS tools regarding the following four key functional features: the capability to integrate soft and hard-clickers using the same software, the dependence of the system on the connection with a server, how they operate to make the roster file to link clickers with students, and finally the availability of the IRS tools for different platforms.

1. First, we must note that most companies providing hard-clickers also offer soft-clickers that may be used in conjunction with the former (e.g. SMART Technologies). Nevertheless, most of these companies are not currently able to integrate them in just one tool, i.e., the software does not consider the simultaneous usage of both hard and soft-clickers and therefore it turns necessary to include an additional tool or even to use different software. For instance, *H-ITT* and *Turning Technologies* provide a specific software for soft-clickers usage denoted as *MultiPoint* and *ResponseWare* (or *RemotePoll*) respectively. Conversely, in OpenIRS-UCM all clickers are integrated and used in

the same fashion, so that no differences exist among hard and soft-clickers.

2. Second, as we pointed out in the introduction, most commercial solutions need to have a connection with a server in order to get responses from soft-clickers. Instead, OpenIRS-UCM turns the teacher's PC itself into a server that receives all answers (from hard-clickers and soft-clickers), so it can be used when WiFi connection with an external server is not available. Moreover, a stand-alone solution could be implemented in no-WiFi zones using an USB router connected like an RF receptor for creating a local network between teacher's PC and soft-clickers.
3. Third, regarding the difficulty of our solution in making the roster, we can conclude that it is similar to that of other solutions. Whereas Educlick always performs an on-line record of the clickers before starting the test in order to create the roster, our solution—as well as *Turning Technologies* and *i>clicker*—import the students data from Moodle for obtaining the IDs, then the roster is created and, when the poll is carried out, the resulting solution file is exported to Moodle.
4. Fourth, with respect to the availability for different platforms, we must highlight that most of the clickers companies only develop their software for Windows and Mac platforms. However, in our case, and also *eInstruction* or *H-ITT*, the application can be used in Windows, Mac and Linux.

4.3.2 Integration with LMS

Finally, we next analyze more extensively how different tools, namely *H-ITT*, *Turning Technologies*, *i>clicker* and OpenIRS-UCM, integrate with LMSs, a key aspect from the education engineering perspective:

H-ITT just integrates its software with BlackBoard and WebCT LMSs, although *Educlick*—the *H-ITT* partner in Spain—does offer integration with Moodle (which is used in the *Campus Virtual* of Polytechnical University of Madrid). This one is the most sophisticated solution of all analyzed ones, and performs as follows. Educlick allows the teacher's PC to connect online with the LMS and executes an interface on the browser like a typical IRS tool, where the questions are chosen from a Moodle questionnaire and are answered by the students using the H-ITT clickers. Even more, *Educlick* has also developed the integration of LMSs with the CPS clickers of eInstruction. The only drawback of this solution is that it requires an Internet connection in the classroom.

Turning Technologies and i>clicker provide integration capability with the most popular platforms (BlackBoard, Moodle and Sakai) in a simpler way than *H-ITT* and *Educlick*: the software connects with the LMS and imports the data about the students of a particular class into a file, in order to generate the roster. Then, using these data, the software exports to the Gradebook just the global results per student obtained in the chosen poll.

OpenIRS-UCM goes a step further with respect to the solutions updating the Gradebook, as not only the final mark is added but also all answers are uploaded to Moodle. Besides, our solution, as also occurs with Educlick, allows leveraging all analysis tools that Moodle offers for the management of the test answers. The main different feature of our tool, that we consider an important advantage, is that OpenIRS-UCM performs the surveys off-line, i.e., the teachers perform the questions in the classroom and then they may import the session desired—or even the questions—to Moodle. Therefore, neither internet connection in the classroom nor using the H-ITT clickers is required; even more, many kind of clickers can be used simultaneously in a poll.

For the sake of clarity and better comparison, Table 2 summarizes the main features that some of the most relevant IRS tools and our proposal are able to support.

Table 2. Comparison of the features offered by other solutions and ours

Tools	**Platforms** (Windows/ Mac/ GNU-Linux)	**LMS** (Blackboard/ Moodle/ Sakai)	**Integration with LMS** (Questionnaries/ Gradebook)	**Hard-clicker third-party companies**	**Soft-clicker/** One solution	**Open source**
OpenIRS-UCM	Yes/Yes/Yes	No/Yes/No	Yes/Yes	Yes	Yes/Yes	Yes
H-ITT	Yes/Yes/Yes	Yes/No/No	No/Yes	No	Yes/No	No
H-ITT (Educlick)	Yes/No/No	Yes/Yes/Yes	Yes/Yes	No	Yes/No	No
Turning Techn.	Yes/Yes/No	Yes/Yes/Yes	No/Yes	No	Yes/No	No
i>clicker	Yes/Yes/No	Yes/Yes/Yes	No/Yes	No	Yes/No	No
SMART Techn.	Yes/Yes/No	No/No/No	No/No	No	Yes/No	No
eInstruction	Yes/Yes/Yes	Yes/Yes[a]/Yes[a]	Yes[a]/Yes	No	Yes/Yes	No

[a] Educlick is an eInstruction Alliance Partner and offers its EPS/Grade solution in order to integrate CPS/vPad clickers with LMS platforms.

4.4 Adoption in real life settings and contribution to engineering education

Although our tool provides somehow similar functionalities to other IRS tools, the key distinctive OpenIRS-UCM features (ease to use, free and open source, common interface to port new third-party devices to the system, allows the usage of soft-clickers without requiring any different software, can be used when WiFi connection with an external server is not available) dramatically simplify teachers' work and also open a broad avenue of new scenarios to perform teaching activities:

- Practical classes given in a laboratory, where the computers may operate as clickers.
- Remote teaching via video conference, where the speaker may interrogate the audience or just check the student's attendance.
- New scenarios outside the classroom: With the widespread use of smartphones, it is affordable to quiz anywhere if 3G/4G connection is available or if a local WiFi network (using a USB router) is implemented. Hence, new scenarios outside the classroom (workshops, labs, hospitals, museums, fieldworks, etc), where teaching becomes an even more dynamic and unforeseeable task, are possible now.

According to [42] there is a wide variety of research areas in the field of engineering education. Notably, this report identifies five broad research areas: (1) engineering epistemologies (what constitutes engineering thinking and knowledge), (2) engineering learning mechanisms (how learners develop knowledge and competencies), (3) engineering learning systems (instructional cultures and institutional practices), (4) engineering diversity and inclusiveness (how human diversity contributes to engineering processes and products), and (5) engineering assessment (development and use of assessment methods, instruments and metrics).

We consider that all the aspects mentioned before supposes a considerable improvements in the area of engineering learning systems, which is shown by the facts that our tool is currently used in practically all of the schools within the University Complutense of Madrid and it has become the official teaching test program over the coming years. Besides, mostly due to OpenIRS-UCM ease of use and possible usage in new teaching scenarios, it advances in the research area of engineering assessment as it postulates as an optimal instrument to perform evaluation activities. We would like to emphasize that OpenIRS-UCM as most IRS can also be seen a learning system which promotes the learning process and skills development.

5. Conclusions

In this work we have described in detail the OpenIRS-UCM IRS system, available in [25], where successive versions will also be allocated in the future. OpenIRS-UCM development would not be possible without the use of a Software Engineering methodology.

The MVC architectural pattern for the software development design has allowed us to implement an Interactive Response System and others teaching tools. We have developed new views and a few new methods to achieve the development of successful teaching tools. Indeed, we have built a view completely integrated with the Moodle LMS (to the best of our knowledge, it is the only free and open source IRS tool that enables the integration with an LMS); and a new view to implement another response system in order to carry out successful quality survey in the UCM. It is worth mention that nowadays Moodle 2.6 is the LMS used in UCM, so we plan to implement a new view to this new Moodle version.

Its key advantageous features open a broad spectrum of situations in which the application can be adopted, significantly improving the engineering learning systems field. Our tool has been tested in depth in order to evaluate its usability, acceptance, and complexity, delivering satisfactory results. Moreover, it is being integrated into the teachers' workflow in our University, and other Spanish Universities have already expressed their interest in using our tool. Thus, given the proven success of OpenIRS-UCM, we think that we should keep working on its development, incorporating support for new clickers, platforms and LMSs, or implementing new views which include new useful features that do not compromise its current ease of use.

Acknowledgements—This work has been supported by the University Complutense of Madrid through the Educational Innovation Projects PIMCD 614-07/08, 204-08/09, 293-09/10, 211-10/11, 189-11/12, 1-12/13 and 187/13, and by the Spanish government through the research project TIN 2012-32180.

References

1. E. Mazur, *Peer Instruction: A User's Manual*, Prentice Hall, 2007.
2. D. W. Johnson, R. T. Johnson and K. A. Smith, *Active learning: cooperation in the college classroom*, Interaction Book Company, 1991.
3. C. C. Bonwell and J. A. Eison, *Active learning: Creating excitement in the classroom*, School of Education and Human Development, George Washington University, 1991.
4. K. M. Moss and M. Crowley, Effective learning in science: The use of personal response systems with a wide range of audiences, *Computers and Education*, **56**(1), 2011, pp. 36–43.
5. M. A. Purvis, B. T. R. Savarimuthu and M. K. Purvis, Architecture for active and collaborative learning in a distributed classroom environment, *ACM Advanced Technology for Learning*, **3**(4), 2006, pp. 225–232.

6. D. M. Shaffer and M. J. Collura, Evaluating the effectiveness of a personal response system in the classroom, *ACM Teaching of Psychology*, **36**(4), 2009, pp. 273–277.
7. C. Doe, A Look at Student response systems, *Multimedia and Internet@Schools*, **17**(4), 2010, pp. 32–35.
8. T. C. Liu, J. K. Liang, H. Y. Wang and T. W. Chan, The features and potential of interactive response systems, *Proceedings of International Conference on Computers in Education (ICCE)*, Hong Kong, 2003, pp. 315–322.
9. R. C. Lowery, Teaching and learning with interactive student response systems: A comparison of commercial products in the higher-education market, *In Annual meeting of the Southwestern Social Science Association and its affiliates*, 2005.
10. Promethean, http://www.prometheanworld.com/us/english/education/products/, Accessed July 20, 2013.
11. eInstruction tool, http://www.einstruction.eu/education/, Accessed July 18, 2013.
12. SunVote, http://www.sunvote.com.cn/, Accessed January 8, 2014.
13. J. Stav, K. Nielsen, G. Hansen-Nygard and T. Thorseth, Experiences obtained with integration of student response systems for ipod touch and iphone into e-learning environments, *Electronic Journal of e-Learning*, **8**(2), 2010, pp. 179–190.
14. Interactive learning toolkit-bq, http://galileo.seas.harvard.edu/login, Accessed January 8, 2014.
15. Y. Suo, N. Miyata, H. Morikawa, T. Ishida and Y. Shi, Open smart classroom: Extensible and scalable learning system in smart space using web service technology, *IEEE Transactions on Knowledge and data engineering*, **21**(6), 2009, pp. 814–828.
16. Y. Tokiwa, K. Nonobe and M. Iwatsuki, Web-based tools to sustain the motivation of students in distance education, *Proceedings of IEEE International Conference on Frontiers in Education (FIE)*, San Antonio, TX, USA, 2009, pp. 1438–1442.
17. I. A. Zualkernan, InfoCoral: Open-source hardware for low-cost, high-density concurrent simple response ubiquitous systems, *Proceedings of IEEE International Conference on Advanced Learning Technologies (ICALT)*, Athens, Greece, 2011, pp. 638–639.
18. OpenIRS-UCM Short Presentation, http://www.youtube.com/watch?v=bcbhmQ272jM, Accessed October 10, 2013.
19. uTeleTest, https://play.google.com/store/apps/details?id=com.ucm.dacya.uteletest&hl=it, Accessed October 4, 2013.
20. iTeleTest, http://www.dacya.ucm.es/iteletest/Site/iTeleTest.html, Accessed October 4, 2013.
21. S. Smith, G. Salaway and J. B. Caruso, *The ECAR Study of Undergraduate Students and Information Technology*, Educause Center for applied research, 2009.
22. C. Garcia, F. Castro, J. I. Gomez, C. Tenllado, D. Chaver and J. A. Lopez-Orozco, OpenIRS-UCM: an open-source multi-platform for interactive response systems, *Proceedings of the ACM Conference on Innovation and Technology in Computer Science Education (ITICSE),* Haifa, Israel, 2012, pp. 232–237.
23. DOCENTIA: an IRS System to poll in UCM, http://www.ucm.es/irs-encuestas, Accessed November 26, 2013.
24. SQLite, http://www.sqlite.org, Accessed February 18, 2014.
25. OpenIRS-UCM Repository, http://sourceforge.net/projects/openirs-ucm, Accessed May 21, 2014.
26. Java Native Interface, http://docs.oracle.com/javase/7/docs/technotes/guides/jni/index.html, Accessed November 5, 2013.
27. H-ITT software development kit, http://www.h-itt.com/downloads.htm, Accessed October 5, 2013.
28. RXTX, http://fizzed.com/oss/rxtx-for-java, Accessed November 3, 2013.
29. GNU-LGPL license, http://www.gnu.org/licenses/lgpl.html, Accessed May 6, 2014.
30. Apache POI—the Java API for Microsoft Documents, http://poi.apache.org, Accessed March 7, 2014.
31. Apache License, http://www.apache.org/licenses/LICENSE-2.0.txt, Accessed April 12, 2014.
32. JFreeChart, http://www.jfree.org/jfreechart, Accessed October 12, 2013.
33. JDOM, http://www.jdom.org, Accessed October 12, 2014.
34. OpenCSV, http://opencsv.sourceforge.net, Accessed January 22, 2014.
35. W. Rice, *Moodle E-Learning Course Development*, Packt Publishing, 2006.
36. Moodle guidelines for contribution code, https://docs.moodle.org/dev/Coding, Accessed February 6, 2014.
37. Moodle Stats, https://moodle.org, Accessed January 14, 2014.
38. OpenIRS-UCM tutorial, https://www.youtube.com/watch?-v=4guODSY_C04, Accessed March 3, 2014.
39. CCCC tool, http://cccc.sourceforge.net, Accessed April 6, 2014.
40. J. Brooke, SUS: A quick and dirty usability scale. In: Jordan, P., Thomas, B., Weerdmeester, B. (Eds*.), Usability Evaluation in Industry*. Taylor & Francis, 1996, pp. 189–194.
41. J. R. Lewis and J. Sauro, The factor structure of the system usability scale, *Proceedings of the International Conference on Human Centered Design (HCD),* San Diego, CA, USA, 2009*,* pp. 94–103.
42. Special Report, The Research Agenda for the New Discipline of Engineering Education, *Journal of Engineering Education*, **95**(4), 2006, pp. 259–261.

Carlos García received his MS in Physics and PhD in Computer Science from the Universidad Complutense de Madrid (UCM) in 1999 and 2007, respectively. His research interests include parallel computing, computer architecture and code optimization for high performance computing. His current research also addresses heterogeneous systems and energy-aware trade-off.

Fernando Castro obtained the MS degree in physics from University of Santiago de Compostela in 2000, the MS degree in electrical engineering and the Ph.D. degree in computer science from the University Complutense of Madrid (UCM) in 2004 and 2008, respectively. He is now a teacher assistant in the Department of Computer Architecture, UCM. His research interests include energy-aware processor design, efficient memory management and OS scheduling on asymmetric multiprocessors. His recent activities also focused on the software engineering, exploring new tools aiming to improve the classroom teaching.

José Ignacio Gómez received a Ph.D. degree in computer science from the Universidad Complutense de Madrid (UCM) in 2007. He is an assistant professor in the Department of Computer Architecture, UCM. His research interests include high-performance computing, massively parallel architectures and design and management of memory systems for embedded systems.

Daniel Chaver received the degree in physics from the University of Santiago de Compostela, Spain, in 1998, and the Electrical Engineering and Ph.D. degrees from the Complutense University of Madrid, Spain, in 2000 and 2006, where he is

currently an Associate Professor. He has been teaching different subjects related to Computer Science and Electrical Engineering since 2000. His current research interests include: (1) architectural techniques for managing efficiently the memory hierarchy and (2) OS scheduling techniques for asymmetric multiprocessors.

Jose A. Lopez-Orozco received the M.Sc. and Ph.D. degrees in physics from the Physics Science Faculty, University Complutense of Madrid (UCM), Madrid, Spain, in 1991 and 1999, respectively. In 1992, he joined the Department of Computer Architecture and Automatic Control, UCM, as a Teaching Assistant and, later as an Assistant Professor. Since 2002, he has been an Associated Professor. His current research interests include applications of Automatic Control, artificial neural networks, robotics, and multisensor fusion systems.