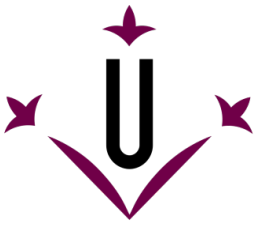

Cole-Programming: Shaping Collaborative Learning Support in Eclipse

Francisco Jurado, Ana I. Molina, Miguel A. Redondo y Manuel Ortega

Abstract— **In the teaching of those subjects related to Computer Programming competences, a teaching/learning methodology based on Problem Solving is often used, which sometimes involves the application of techniques such as Pair Programming. In addition, to perform the Programming-related tasks, it is common to use professional Integrated Development Environments. Thus, it may be interesting to integrate tools in these environments so that they can support the teaching/learning process. This article discusses how we have adapted an existing Eclipse-based system for learning Programming, so that now it has support for collaborative tasks. The new features have been specially designed to solve Programming-related assignments, and integrated into the learning environment. We also analyse the first impressions gathered from students in the initial experiences made on the system.**

Index Terms— **Learning Programming, Computer-Supported Collaborative Learning, Eclipse, Tuple Spaces, Pair Programming**

I. Introduction

DURING the process of acquiring the knowledge and the skills related to Computer Programming competences (from now on we will refer to them as Programming) [1], the students are faced with different obstacles [5] [8] [21]. In this scenario, in the subjects related to these competences, the most widely used method of teaching is often based on activities involving Problem Solving [18] [30], reinforcing the lectures in classroom with programming labs [29] [35] that provide active learning [31].

Since students must learn to design, develop, test and debug programs with certain tools that are designed to be used by professional programmers as Satratzemi points out [37], Integrated Development Environments (IDE) for professional use, such as Eclipse, JDeveloper, NetBeans, IntelliJ, etc. are often used in these programming labs. Thus, it may be an interesting approach to introduce improvements on these environments, such as tools that support the process of teaching/learning Programming-related skills.

On the other hand, the implementation of the principles proposed by the European Higher Education Area (EHEA) includes group work as one of the techniques to enhance the learning process. In this sense, Computer Supported Collaborative Learning (CSCL) [28] aims to take advantage of the synergistic effect created when several students work together to solve a particular problem, providing the computational support that allows them to communicate and coordinate in completing their learning activities.

Thus, applying the collaborative paradigm in the teaching/learning of Programming by solving problems or projects in working groups (collaborative learning) would introduce the principles of the EHEA in acquiring Programming-related competences. Not in vain, collaborative learning has been applied to the field of learning Programming. There are several proposals in this regard [38] [34] [33] [13]. Applying CSCL for learning Programming provides an environment in which students can interact and communicate with each other, typically in a distributed computing environment allowing students to get profit from the knowledge and skills of other members of the group, and so improving their own skills.

Within the scope of collaborative learning applied to Programming, a commonly applied technique is Pair Programming. In Pair Programming, two programmers work together on the same design, algorithm, code or test [39]. While one programmer performs actions on the environment, the other one looks at the steps her partner performs to try to detect errors and report them. Although this is a technique typically used in the field of software engineering, its use in educational contexts provides interesting benefits that enhance learning [36].

In this framework, our goal is to support educational methods that apply the technique of Pair Programming in a professional IDE. In particular, this paper shows how we have enhanced an existing system for learning Programming without support for collaborative tasks. This system is based on the widespread Eclipse IDE, adapted with collaborative tools that will be integrated in that environment.

Thus, the rest of the paper is structured as follows: it will begin with a background on those systems that support computational Programming labs (Section 2); then it will show how to provide collaborative learning support to the selected system for learning to program (section 3); next, it will detail the functionality specifically designed for Programming tasks implemented for the IDE (section 4); after that, it will expose a first experience and analyse the students' impressions regarding the new functionalities

Francisco Jurado is Visiting Lecturer in the Department of Computer Science and Industrial Engineering at the Polytechnic School of the University of Lleida, Jaume II, 69, Lleida (Spain) e-mail: Francisco.Jurado@diei.udl.cat

Ana I. Molina is Senior Lecturer in the Department of Technologies and Information Systems in the School of Computers Sciences at the University of Castilla-La Mancha, Paseo de la Universidad, 4. 13071 Ciudad Real (Spain), e-mail: AnaIsabel.Molina@uclm.es

Miguel A. Redondo is Professor in the Department of Technologies and Information Systems in the School of Computers Sciences at the University of Castilla-La Mancha, Paseo de la Universidad, 4. 13071 Ciudad Real (Spain), e-mail: Miguel.Redondo@uclm.es

Manuel Ortega is Full Professor in the Department of Technologies and Information Systems in the School of Computers Sciences at the University if Castilla-La Mancha, Paseo de la Universidad, 4. 13071 Ciudad Real (Spain), e-mail: Manuel.Ortega@uclm.es

DOI (Digital Object Identifier) Pendiente

(section 5); and finally, it will show some concluding remarks (section 6).

II. BACKGROUND

In order to facilitate the implementation of activities that involve problem-solving for Programming, in a scenario where classroom lectures are complemented with Programming labs to encourage active learning, an approach that can benefit students is to use real development environments. These environments are called Integrated Development Environments (IDE), that is, programs to help developers to implement other applications. Such IDEs usually include tools that support developers during the development process in its different phases.

Using a real IDE, students can manipulate a development environment as they will do in their future careers, preparing them not only to learn the concepts related to Programming, but also to learn to manipulate and to take advantage of the available tools, favouring both their learning experience and their future professional environment.

Currently there are many IDEs, either freely available or commercial. Among them, Eclipse (http://www.eclipse.org) stands as one of the most widely used [9] [16]. In fact, the possibilities for customization and expansion that Eclipse provides have not gone unnoticed by researchers in the area of learning Programming, and so we can find several solutions in this regard for Java development. These solutions consist of *plug-ins* added to Eclipse in order to make it an environment suitable for learning Programming.

Thus, Kenya Eclipse [10] [11] integrates the Kenya programming language [2] [9] in the Eclipse environment. Kenya is a sublanguage of Java. It is designed for students to focus on learning the basic programming structures, eliminating those aspects of syntax that may be irrelevant in the first sessions of learning Programming. Moving from Kenya to the equivalent Java code is straightforward, so that students can easily learn how to write the corresponding Java program. Kenya Eclipse takes all the features of Eclipse, such as the project management, the compiler output, etc. It also incorporates a module devoted to the analysis of the programming style that allows students to improve the writing and readability of the source code they write.

Another Eclipse-based learning environment is ProPAT [3]. This tool allows the teacher to propose a new Programming assignment by specifying the assignment name and a template with the syntactical structure the students must follow. This template is then checked against the abstract syntax tree of the code written by the student, so that the system can provide students with information about what was right and wrong in the solution they have developed.

In the same way, when analysing the solutions developed by students, we can find COALA (Computer Assisted Environment for Learning Algorithms) [24] [25]. This environment allows the teacher to specify Programming assignments (along with the ideal representation according to the evaluation criteria and the test cases to be met), the distribution of these assignments to the students, the delivery of the solutions developed by the students, and the automatic evaluation of such solutions by using Fuzzy Logic. COALA uses this evaluation with two objectives [25]: on the one hand, to provide students with a brief explanation about what is right and wrong in their solution so that they can understand and improve it; on the other hand, to provide feedback to the system, allowing it to determine which the next learning activity is, among those specified by the teacher, to be done by each specific student.

Among the projects that apply the CSCL principles and the extensibility of Eclipse we can find Jazz Sangam [12] and RIPPLE [6]. More specifically, these tools seek to build environments that enable Pair Programming. So, Jazz Sangram [12] is a *plug-in* designed to support collaborative work in the Eclipse environment. Jazz is an IBM technology for collaborative development, which provides project management tools, version control, instant messaging, etc.; meanwhile Sangam [22] is a *plug-in* for Eclipse that allows collaborative work using the technique of Pair Programming. Integrating Jazz and Sangam derives in this application. Similarly, RIPPLE [6] builds a Pair Programming environment in Eclipse by means of a shared editor and a simple chat, employing the Sangam architecture to do so. However, these environments do not have collaborative tools for communication and coordination so widely used as forums (which also enable asynchronous communication), voting pool tools (which structure decision-making, etc.). Furthermore, they do not incorporate awareness mechanisms [15] that yield information on the status of other members of the working group, their progress and their future intentions. Furthermore, they do not have tools to monitor and track the student activities, which could be interesting for teachers (both for analysis of the learning process, and for intervening in the group's work sessions as needed).

Thus, we can see how the Eclipse environment is suitable as a basis to develop applications for learning Programming. On the one hand, the use of Eclipse allows eLearning developers to focus on implementing purely educative functionality, freeing them from the implementation of common tools such as project management, code editors, integration with the compiler and debugger, etc. On the other hand, the developed educational tools are integrated into similar environments students will find in their future careers.

This paper takes as its starting point the above-mentioned distributed environment COALA, which was developed as a result of our previous research on the field of systems for learning to program, but it proved lacking in support for collaborative tasks. The next section will show how to provide collaborative support to COALA.

III. PROVIDING ECLIPSE SUPPORT FOR COLLABORATION

As mentioned in the previous section, the starting point for our work is COALA [24] [25]. It was developed as *plug-ins* for Eclipse in order to allow the use of a real Programming environment in an academic scenario, which makes use of educational standards to facilitate reusability and integration in other systems, highlighting the use of IMS -LD [23]. The Eclipse *plug-in* for COALA allows communication features via a tuple spaces server [19] [20] [27]. This server acts as a shared central memory in a

network environment where several clients and services can store and read information in the form of tuples [26] [27].

The choice of tuple spaces looks to be appropriate to implement collaborative eLearning systems [7] [17]. Thus, it seemed feasible to use the architecture of COALA to implement new services, drawing on both the extension and customization features of Eclipse, and the scalability of a system based on tuple spaces.

Thus, we implemented three of the most common collaborative services as a *plug-in* for Eclipse. We called it COLE-Programming. These services are: a chat to allow communication in synchronous way, a forum to allow asynchronous communication, and a voting pool that allows the students to pool, to sound out, to come to agreements, etc.

These services are provided with specific features and functionality, especially designed to perform Programming tasks. As an example, Figure 1 shows the functionality of the chat and the forum, highlighting not only the exchange of messages inherent in this type of service, but features such as code sharing or exchanging error and warning messages from the compiler or runtime environment.

As mentioned above COALA makes use of IMS-LD. There are several proposals to allow integration of these collaborative services in the system [4] [14] [32]. However, given the architectural properties of COALA for integrating agents and services through the use of tuple spaces [27], the integration task is reduced to the implementation of proxies, which abstract the data model of each service to a tuple model.

Figure 2 shows the diagram of the architecture of COALA once the COLE-Programming *plug-in* is added. As it is shown in this figure, all clients and services are connected via a TCP connection. The tuple space server we have selected for the implementation is SQLSpaces (http://sqlspaces.collide.info/) (Figure 2 at right), which is distributed under a free license AGPL. This server runs on a Java Virtual Machine (JVM) and implements the corresponding logical translation between tuples and a relational database. In the case of COALA, we have chosen a MySQL database on which to perform persistence.

Due to the fact that different clients and services can interact, they require the corresponding libraries and stubs that support the middleware interaction (SQLSpaces client stubs in the figure), whether running on a JVM or used from any Programming language other than those supported by SQLSpace. This is the case of the features available for the COALA *plug-in* inside Eclipse (shaded area in the centre of the figure) and the COALA services available (Figure 2 on left). These allow the teacher to specify the Programming assignments and the test cases to meet, to distribute assignments to students, to deliver the students' solutions and the automatic evaluation of such solutions, etc.

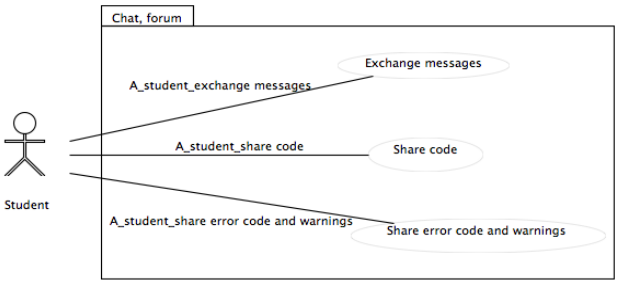

Fig. 1. UML Use-Case Diagram for the chat and forum tools

On the client side (central part of the figure), we can see how Eclipse is stated as the platform users interact with it. Both the Eclipse platform and the SQLSpaces client stubs run on a JVM. In order to facilitate the implementation of the upper layers several proxies have been implemented, which provide transparency enough with respect to the middleware.

On top of this layer of proxies, we can find the model-view-controller layers proposed by Eclipse. This consists of SWT Content Providers and the related SWT Views and Editors for each tool to implement, either of those implemented in COALA (central shaded right of the figure) or the new tools (chat, forum and voting pool).

As it can be appreciated, the implementation of the collaborative tools in the COALA architecture is straightforward. We only need to implement the corresponding proxies, content providers and views/editors to make them available to the different clients. The messages and information the users will interchange will be represented by means of tuples accessed by the working group on the tuple space.

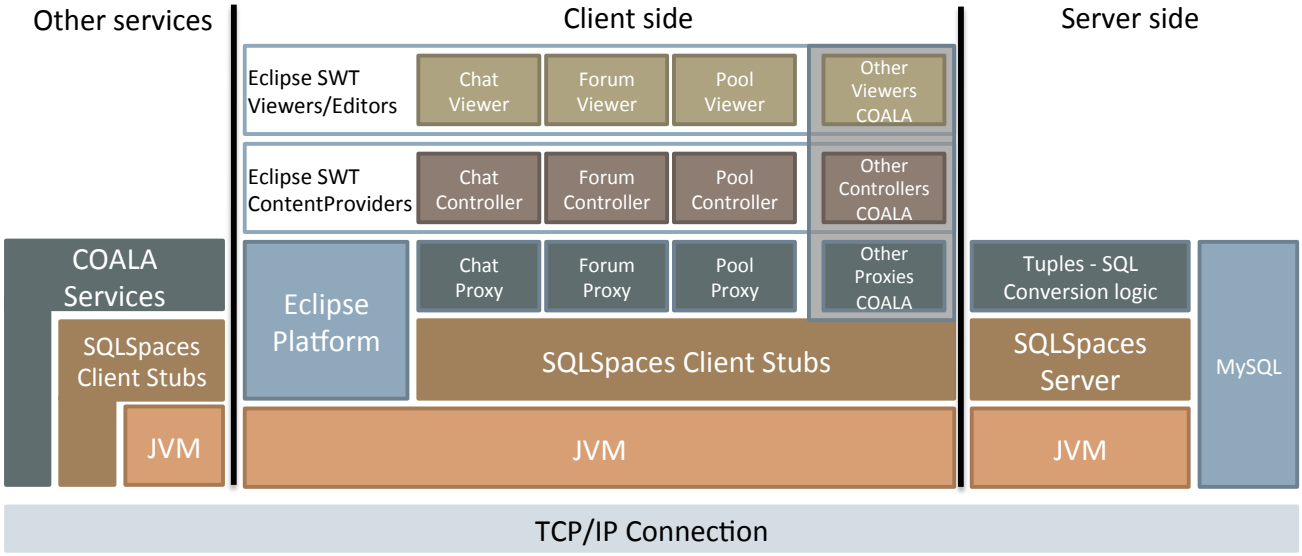

Fig. 2. Diagram of the system architecture

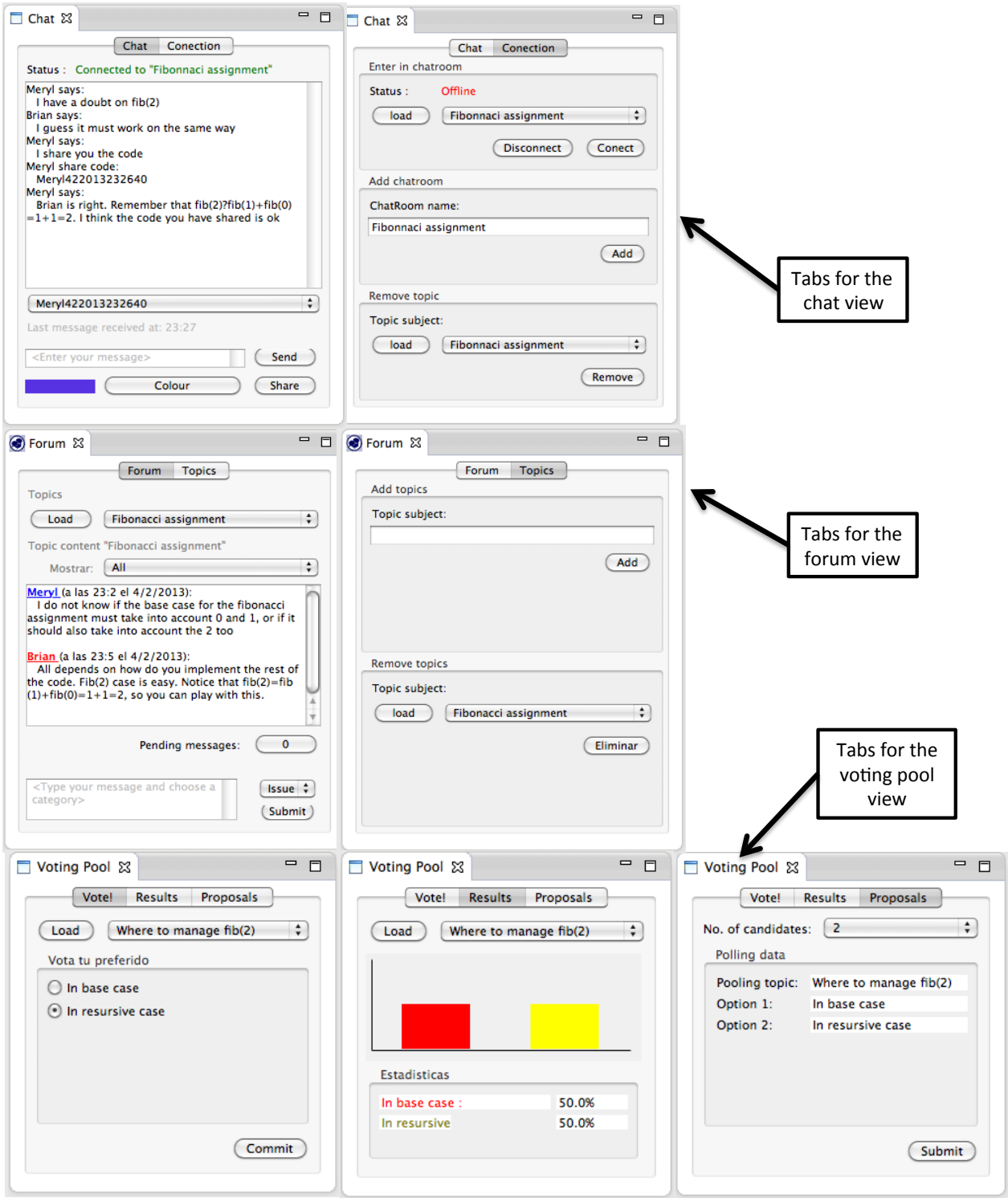

Fig. 3. Collaborative tools added by COLE-Programming to the COALA environment

In the following section we will describe, by way of examples, some collaborative services implemented in order to test the validity of our approach and the specific functionality of the Programming domain that can be included to facilitate collaborative tasks.

IV. COLE-PROGRAMMING: PLUG-IN TO SUPPORT COLLABORATION

COLE-Programming (Collaborative Learning Programming) is the name for the developed Eclipse *plug-in* that implements the collaborative tools for the COALA environment.

Figure 3 shows how COLE-Programming implements several views for a voting pool (on bottom), a chat (on top-left) and a forum (on top-right) with some Programming specific functionalities we will describe in detail next.

Thus, in order to facilitate the information exchange, we have added the necessary functionalities to allow sharing code, error messages, warning from the compiler, etc. Therefore, when a student wants to share one of these kinds of usual messages in a chat or forum, a dialog window as the one shown in figure 4 is opened. This window allows introducing a description message, the kind of information to share (part of a source code, an error or warning message,

etc.) and the information to share. In the chat and forum, a message with the form *"<user> has published the <kind of message>: <id>"* is posted, which means that the information has been published for those people who are connected to the chat room or reading the thread of a forum. The students will be able to select, by using a combo box of the chat view, the "id" related to the shared information, and then an editor will appear showing the message and the description.

Besides, we have added to the forum a message filtering mechanism for the conversation thread, taking into account the kind of message (identified issue, solution proposal, critical comment, doubt, etc.) as we can see on top of figure 5. A colour code helps to identify the kind of comment so that the students can quickly identify the kind of message. In addition, a combo box allows them to filter the kind of message to be shown in the view, so that, for instance, they can focus their attention only on issue identification, proposal to solve the issues, doubts exposed, etc.

In order to support the teacher during the analysis and monitoring processes for collaborative assignments, COLE-Programming implements a monitor view (figure 5 on bottom). This allows checking some collaboration features, such as the number of contributions for a specific student for each collaborative tool, the rate of participation, the users a specific student has collaborated or shared information with, etc.

Throughout this section, it has been shown how we have implemented specific functionalities and features in order to perform better collaborative Programming activities. Thus, we have shown how COALA has been enhanced by adding a chat and a forum with these new features, as well as by incorporating a voting pool and a monitor view for collaborative activities, which can be used by teachers. Hence, COLE-Programming allows the use of COALA in collaborative learning environments. The next section will detail a first experiment conducted in order to analyse the impressions of COLE-Programming by students.

V. A First Experience with Students

Currently, the computational support in most of educative institutions is limited to deploying a Learning Management System (LMS). This kind of environments provides services and tools, such as didactical resource repositories, auto-evaluation quizzes for students, tools used by the teacher for monitoring the students' progress, and collaborative tools (chats, forums or voting pool) which add a collaborative perspective to this kind of computational support. These last kinds of collaborative services and tools can be used to cover some coordination, cooperation and communicative necessities between teacher and students.

Thus, in a first experience, we have tried to check if students prefer to use collaborative tools specifically designed for Programming, which are embedded within the integrated development environment they use to perform their tasks, or otherwise those provided by other generic LMS-based alternatives.

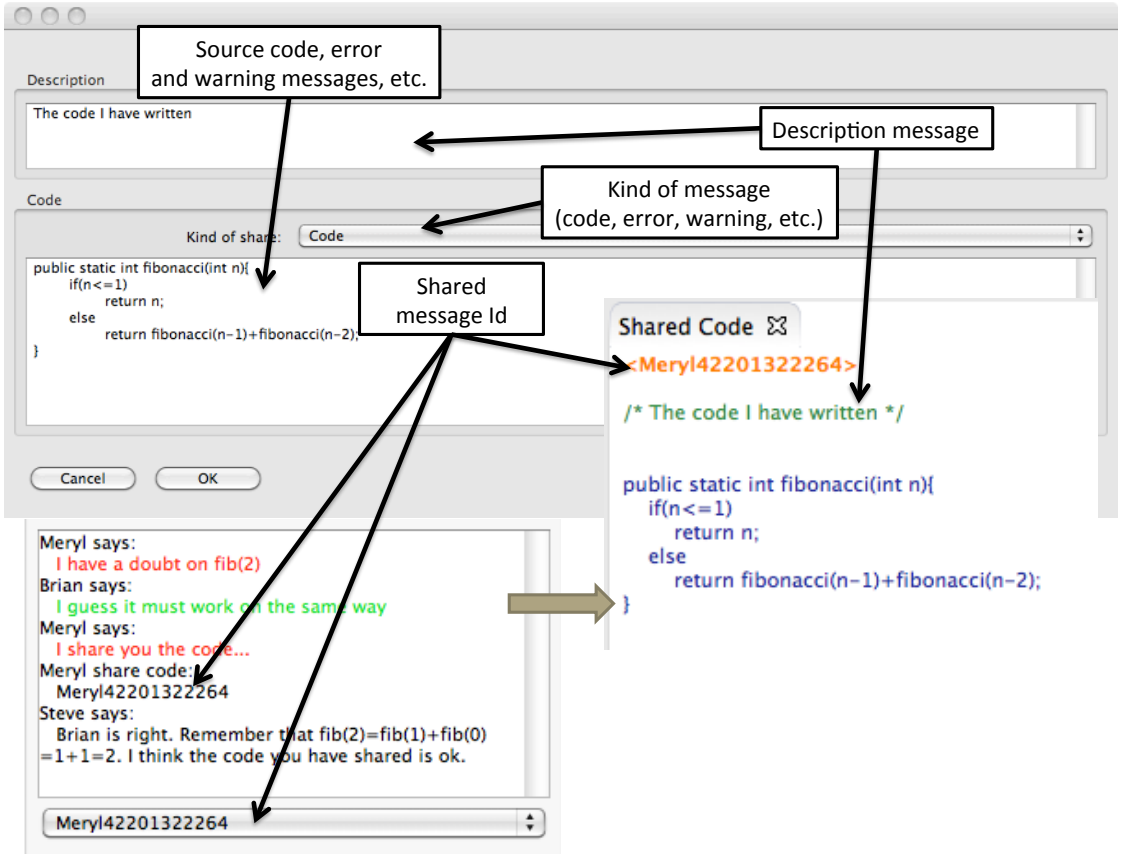

Fig. 4. Sharing code, error and warning messages, etc.

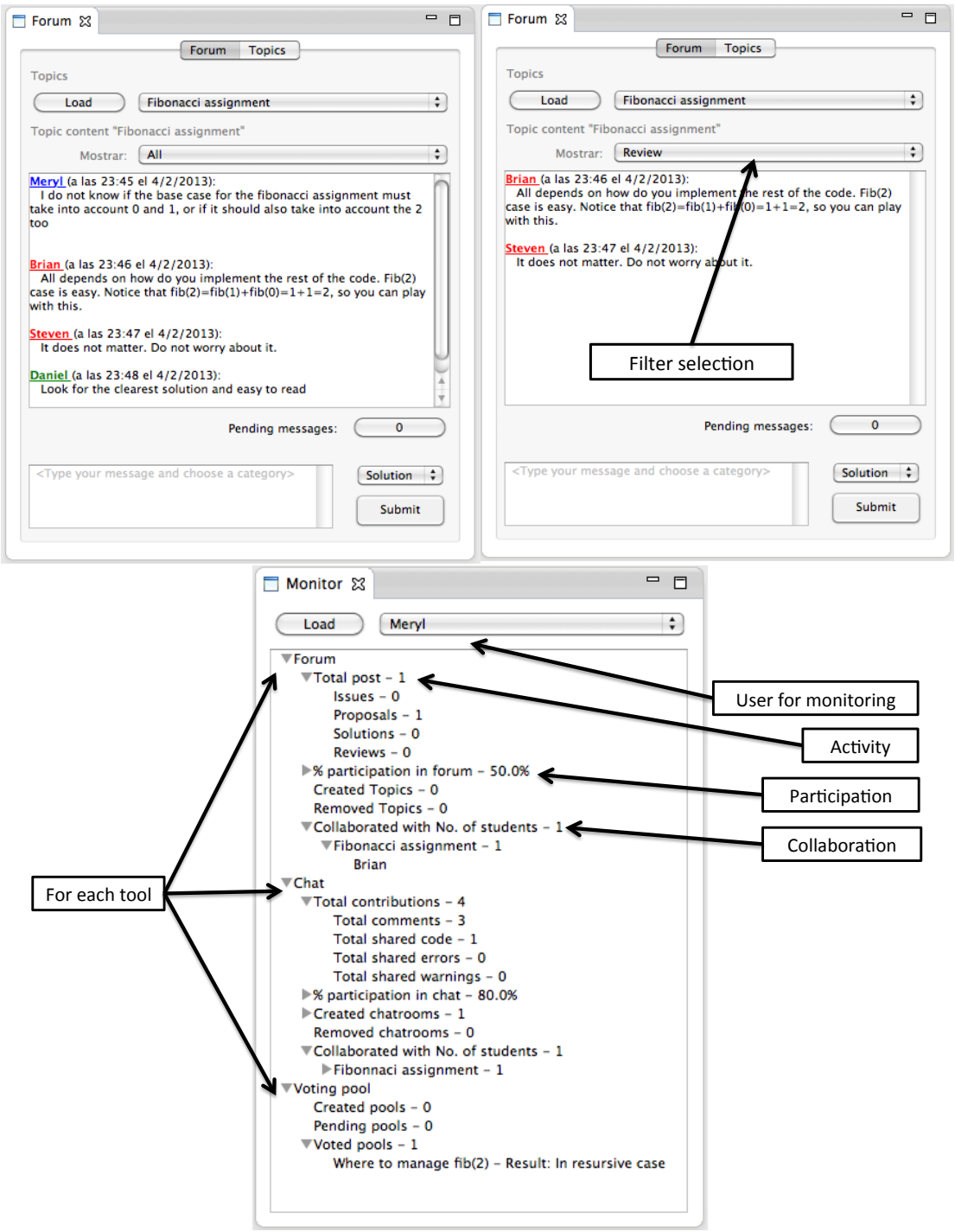

Fig. 5. Message filtering in forum (at the top). Collaboration statistics in the monitor view (at the bottom)

To do so, we took the evolution in COALA carried by the COLE-Programming *plug-in* with tools specifically designed for learning Programming. Then, students used them to perform Programming tasks, so that they could contrast the new tools against those provided by a widely used LMS, in our case Moodle (http://moodle.org/).

A. *Participants*

In the conducted experience, a group of sixteen students of the subject Collaborative Systems for the fifth year in Computer Engineering from the University of Castilla-La Mancha (UCLM) was involved.

This student profile was selected because they had the necessary knowledge in the field of Programming in order to assess the support tools to Programming, as well as in the field of collaborative systems, which allow them to provide a more objective and critical issues in the collaborative tools implemented.

B. *Preparing the Experience*

Before the experience started, in order to know whether the students' answer was biased due to their predilections, habits, etc., a questionnaire was designed to identify their degree of knowledge and preferences about the Programming IDEs (and Eclipse in particular). This served to know what kind of communication/coordination tools they often use when they make collaborative Programming

tasks, what kind of information they interchange through, and so on.

Students ordered their preferences about collaborative tools and the kind of information they usually exchange on a Likert scale of 1-5 (1 being the least preferred option and 5 the most desirable one).

As results of the questionnaire, it is noteworthy that students prefer the email (35% and 41% identified it as the first and second choice respectively), followed by the chat (44% identified it as first choice) and forum (22% identified it as a second choice) as the collaborative tools to use when solving Programming activities in group, regardless of whether they are integrated in the environment or not.

Furthermore, the analysis of the obtained data allowed us to determine the kind of information (in addition to text messages) that students usually exchange in these collaborative Programming sessions, highlighting the error and warning messages (38% of the students marked it as first choice and 25% as second) and verbatim copies of code fragments (31% identified it as first choice and 19% as second).

C. *Describing the Experience*

To perform the experience, we paired students randomly. Also, the students were given the information available on the COLE-Programming website (http://chico.esi.uclm.es/coala), including the description of the collaborative tools and a user manual.

The experience lasted two hours, in which students had to solve two Programming assignments and complete the appropriate questionnaires after each one. Each questionnaire included a student ID, which served to identify the participants in the other activities of the experience, thus preserving anonymity. Experience consisted of the following stages:

1. *First assignment session (Eclipse + Moodle tools):* For this first session, we gave the students the sentence of a Programming assignment. They had to do it by using the Eclipse environment and the external tools for communication and coordination included in the Moodle LMS deployed in our university, which the students know well enough. The approximate time of completion of this first Programming assignment was 40 minutes.
2. *Assessment questionnaire (Eclipse + Tools Moodle):* Students completed a questionnaire in order to get feedback about their impression of the collaboration tools for the Programming session. The time for completion was 10 minutes.
3. *Second assignment session (Eclipse + COLE-Programming):* The same working pairs who participated in the first session held on a second Programming assignment (with a difficulty degree quite similar to that from the first stage) by using the tools embedded in Eclipse, that is, the Cole-Programming *plug-in*. The approximate time of completion of this second Programming exercise was 40 minutes.
4. *Assessment questionnaire (Eclipse + COLE-Programming)*: Students completed a questionnaire in order to get feedback about their impression of the collaboration tools for the Programming session. The time for completion was 10 minutes.

The questionnaires consisted of both a series of questions that students had to answer following a Likert scale with scores from 1 to 5 (1: none, 2: low, 3: occasionally, 4: often, 5: very much), as well as of open-ended questions where they could express their opinion.

After gathering all the information generated during the experience, it was the moment to analyse and to interpret it, in order to assess and contrast the collaborative tools that implement COLE-Programming and those from Moodle, which are not specific for Programming tasks, so that we can compare both approaches. In the next section we will show and analyse the results.

D. *Results Obtained from the Experience*

After the two sessions in which students used Eclipse with collaborative tools (on the one hand, those from the Moodle platform; on the other, those integrated in COALA by means of the COLE-Programming *plug-in*), they seem to find the chat tool integrated into COLE-Programming easy to use, since it incorporates specific functionality for Programming tasks.

Figure 7 shows the students' evaluation of the generic collaborative tools and of those integrated into COLE-Programming. Analysing this figure we can see how the use of forums and voting pool seems less widespread (most students answer *"Not applicable (N/A)"* when scoring these tools). In the open-ended questions, students explained the reason for their penchant for using the chat. They consider the chat as that collaborative tool that allows them to communicate with their partners in a more immediate way, allowing a faster and less structured dialogue.

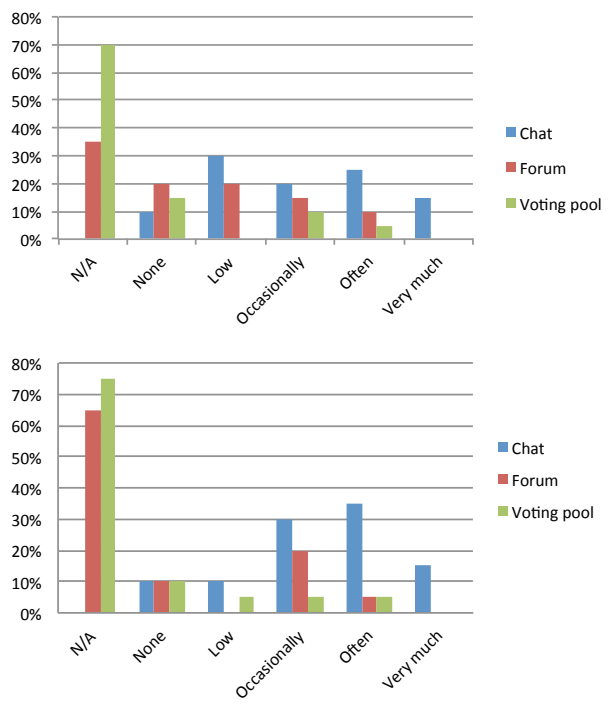

Fig. 7. Rating collaborative tools. At the top, score given to the generic tools. At the bottom, score to the integrated tools implemented in COLE-Programming

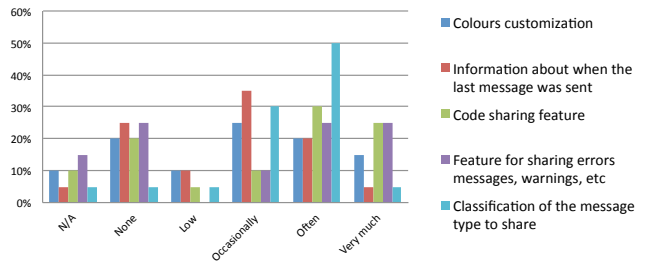

Fig. 8. Score for the features introduced in the collaborative tools of COLE-Programming

Furthermore, in Figure 7 we can see how the students give a higher score to COLE-Programming collaborative tools. Thus, in the case of generic tools, students consider the use of the chat and the forum *nothing, little or occasionally interesting*, and they judge *occasionally, quite or very valuable* their equivalents in COLE-Programming, especially the chat. The high values in the option *"not applicable (N/A)"* for the forum and the voting pool tool were due to those students who chose not to use them. This confirms the predilection students for the chat tool.

The main highlighted issue in the generic collaborative tools was that the students were not able to share code with them. They lacked features such as a code indentation or syntax colouring. They also point out that using external windows apart from the working environment (Eclipse) forces them to constantly change their focus of attention, and it diverts them from the undertaken task.

Similarly, students analysed the features introduced in the COLE-Programming tools specific for Programming tasks, such as those that allow sharing code, error and warning messages, etc. The results can be seen in Figure 8. It shows how the features for sharing code with proper indentation and formatting, as well as the exchange of error and warning messages from the output of the compiler, etc., are top rated by the students. That is, those features added specifically for Programming tasks were the most helpful for students. In addition, they are also interested in some improvements in the context-awareness, such as customizing colours to identify users involved in collaboration and information instantly sending the last message.

Hence, after enhancing COALA by implementing the COLE-Programming *plug-in* with collaborative tools specifically designed to accomplish Programming tasks, and after contrasting their use against generic collaborative tools provided by an LMS to perform collaborative tasks, it has been shown how students seem to find especially more attractive those integrated into the Programming environment.

In light of the experience described, it seems that students prefer to use specific collaborative tools for Programming embedded within the integrated development environment they use to perform the tasks, against those provided by generic alternatives not integrated, such as those included in a LMS.

VI. Concluding Remarks

Pair Programming is one of the techniques used to solve Programming assignments in the context of teaching/learning Programming.

Throughout this article, we have shown how to provide support for collaborative tasks to a system for learning Programming which was initially devoid of it. To do so, it has been shown how the requirements to provide support for collaboration have been introduced in the architecture. Then, it has been shown how the functionality has been implemented as a *plug-in* for the Eclipse environment which we have called COLE-Programming.

Subsequently, we have analysed the students' first impressions regarding the tools integrated in the environment. These impressions suggest that students find interesting those features introduced in the tools by COLE-Programming specifically designed for Programming tasks and embedded within the integrated development environment commonly used for their Programming tasks.

Acknowledgements

This research has been partly supported by the Ministry of Science and Innovation through the project *"Software environment for learning Programming in group and its integration by mean of standards in learning management systems"* (in Spanish) (REF: TIN2011-29542-C02-02) with ANEP evaluation, by the Regional Government of Castilla-La Mancha (Spain) through the projects iColab, INTEGROUP (REF.: PPII11-0013-1219) and GITE-LEARN (PEII11-0133-6335), and by the University of Castilla-La Mancha through the teaching innovation project Prog-Colab.

References

[1] ACM/IEEE (2008), 'Computer Science Curriculum 2008: An Interim Revision of CS 2001. Report from the Interim Review Task Force', Technical report, ACM/IEEE Computer Society.

[2] Allwood, T.; Chatley, R. & Sackman, M. (2004), 'Kenya. Technical Report', Technical report, Imperial College London.

[3] de Barros, L. N.; dos Santos Mota, A. P.; Delgado, K. V. & Matsumoto, P. M. (2005), A Tool for Programming Learning with Pedagogical Patterns, in 'eclipse '05: Proceedings of the 2005 OOPSLA workshop on Eclipse technology eXchange', ACM, New York, NY, USA, pp. 125-129.

[4] de-la Fuente-Valentn, L.; Pardo, A. & Kloos, C. D. (2011), Generic service integration in adaptive learning experiences using IMS learning design, in Computers & Educcation 57(1), pp. 1160-1170.

[5] Bonar, J. & Soloway, E. (1983), 'Uncovering Principles of Novice Programming', POPL '83: Proceedings of the 10th ACM SIGACT-SIGPLAN symposium on Principles of Programming languages, ACM, pp. 10-13.

[6] Boyer, K. E.; Dwight, A. A.; Fondren, R. T.; Vouk, M. A. & Lester, J. C. (2008), A Development Environment for Distributed Synchronous Collaborative Programming, in 'ITiCSE '08: Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education', ACM, New York, NY, USA, pp. 158-162.

[7] Brecht, J.; DiGiano, C.; Patton, C.; Tatar, D.; Chaudhury, S. R. and Roschelle, J. & Davis, K. (2006), Coordinating Networked Learning Activities with a General-purpose Interface, in 'Proceedings of the 5th World Conference on Mobile Learning'.

[8] Brusilovsky, P.; Calabrese, E.; Hvorecky, J.; Kouchnirenko, A. & Miller, P. (1998), 'Mini-languages: a Way to Learn Programming Principles', Education and Information Technologies 2, pp. 65 -83.

[9] BZ Research (2008), '5th Annual Eclipse Adoption Study', Technical report.

[10] Chatley, R. (2001), 'Java for Beginners', PhD thesis, Imperial College London.

[11] Chatley, R. & Timbul, T. (2005), 'KenyaEclipse: Learning to Program in Eclipse', SIGSOFT Softw. Eng. Notes 30(5), 245—248.

[12] Devide, J. V. S.; Meneely, A.; Ho, C.-W.; Williams, L. & Devetsikiotis, M. (2008), Jazz Sangam: A Real-Time Tool for Distributed Pair Programming on a Team Development Platform, in 'Workshop on Infrastructure for Research in Collaborative Software Engineering, Atlanta, GA,'.

[13] Duque, R. & Bravo, C. (2008), Analyzing Work Productivity and Program Quality in Collaborative Programming, in 'ICSEA '08: Proceedings of the 2008 The Third International Conference on Software Engineering Advances', IEEE Computer Society, Washington, DC, USA, pp. 270-276.
[14] Dodero, J. M. & Ghiglione, E. (2008), ReST-Based Web Access to Learning Design Services, in IEEE Trans. Learn. Technol.1 (3), pp. 190-195.
[15] Dourish, P. and V. Bellotti (1992). Awareness and Coordination in Shared Workspaces. Proceedings of the Conference on Computer Supported Cooperative Work CSCW ´92, Toronto, Canada, ACM Press, New York.
[16] Eclipse Foundation (2009), The Open Source Developer Report, May 2009. Eclipse Community Survey.
[17] Eden, H.; Einsenberg, M.; Fischer, G. & Repening, A. (1996), 'Making learning a part of life', Commun. ACM 39(4), pp. 40-42.
[18] Garrido, J.; Noguera, M.; Gonzalez, M.; Gea, M. &Hurtado, M. (2006), Leveraging the Linda Coordination Model for a Groupware Architecture Implementation, in 'Proceedings of 12th International Workshop on Groupware', Springer LNCS, pp. 286-301.
[19] Gelernter, D. (1985), 'Generative Communication in Linda', ACM Transactions on Programming Languages and Systems 7 (1), 80-112.
[20] Giemza, A.; Weinbrenner, S. & Engler, J.and Hoppe, H. (2007), Tuple Spaces as Flexible Integration Platform for Distributed Learning Environments, in 'Proceedings of ICCE 2007', pp. 313-320.
[21] Gomes, A. & Mendes, A. J. (2007), Learning to Program - Difficulties and Solutions, in 'International Conference on Engineering Education – ICEE 2007', pp. 283-287.
[22] Ho, C.; Raha, S.; Gehringer, E. & Williams, L. (2004), Sangam: A Distributed Pair Programming Plug-in for Eclipse. in 'Systems, Languages, and Applications (OOPSLA), volume Eclipse Technology Exchange at Object-Oriented Programming'.
[23] IMS-LD, (2003), 'IMS Learning Design. Information Model, Best Practice and Implementation Guide, XML Binding, Schemas. Version 1.0 Final Specification', Technical report, IMS Global Learning Consortium Inc, Online athttp://www.imsglobal.org/learningdesign (Last visited 08/06/2012)
[24] Jurado, F.; Molina, A. I.; Redondo, M. A.; Ortega, M.; Giemza, A.; Bollen, L. & Hoppe, H. U. (2009), 'Learning to Program with COALA, a Distributed Computer Assisted Environment', Journal of Universal Computer Science 15(7), 1472-1485.
[25] Jurado, F.; Redondo, M. A. & Ortega, M. (2009), Providing Instructional Guidance with IMS-LD in COALA, an ITS for Computer Programming Learning, in 'Proceedings of the Eclipse/Jazz Technologies for E-Learning, Special Session in Distance Education Technology (DET 2009) International Workshop, The 15th International Conference on Distributed Multimedia Systems (DMS'09)', pp. 211-215.
[26] Jurado, F.; Molina, A. I.; Redondo, M. A. & Ortega (2011), Un Enfoque Distribuido Basado en Estándares para la Integración de Servicios y Agentes en Sistemas de eLearning, in Sierra, J.L. & Sarasa, A. eds., Avances en Ingeniería del Software Aplicada al E-learning, Universidad Complutense de Madrid - Área de Ciencias Exactas y de la Naturaleza, pp. 87-102.
[27] Jurado, F.; Redondo, M. A. & Ortega, M. (2012), Blackboard Architecture to Integrate Components and Agents in Heterogeneous Distributed eLearning Systems: An Application to Learning to Program, Journal of Systems and Software. vol. 85, Issue 7, pp. 1621–1636
[28] Koschmann, T. & Erlbaum, L., ed. (1996), CSCL: Theory and Practice of an Emerging Paradigm, Laurence Erlbaum Associates, Publishers.
[29] Kumar, A. N. (2002), Learning Programming by Solving Problems, in 'Informatics Curricula and Teaching Methods', pp. 29-39.
[30] Makkonnen, P. (2000), Do WWW-based Presentations Support Better (Constructiristics) Learning in the Basics of Informatics?, in '33rd Hawaii Int. Conf. System, Sciences'.
[31] McConnell, J. J. (1996), 'Active Learning and its Use in Computer Science', SIGCUE Outlook 24(1-3), pp. 52-54.
[32] Muñoz-Merino, P. J.; Kloos, C. D. & Naranjo, J. F., (2009), Enabling interoperability for LMS educational services, in Computer Standards and Interfaces 31 (2), pp. 484–498.
[33] Pérez, J. R. P.; del Puerto Paule Ruiz, M. & Lovelle, J. M. C. (2006), SICODE: A Collaborative Tool for Learning of Software Development, in 'IV International Conference on Multimedia and Information & Communication Technologies in Education (m-ICTE2006)'.
[34] Redondo, M.; Bravo, C.; Marcelino, M. & Mendes, A. (2004), Tools for Programming Learning: an Approach to Provide a Social Perspective Using Collaborative Planning of Design, in 'IADIS International e-Society 2004 Conference', pp. 315-322.
[35] Schollmeyer, M. (1996), Computer Programming in High School vs. College, in 'SIGCSE '96: Proceedings of the twenty-seventh SIGCSE technical symposium on Computer science education', ACM, New York, NY, USA, pp. 378-382.
[36] Toll III, V.T., Lee, R. & Ahlswede, T. (2007), 'Evaluating the Usefulness of Pair Programming in a Classroom Setting', Computer and Information Science, ACIS International Conference on 0, 302-308.
[37] Satratzemi, M.; Dagdilelis, V. & Evagelidis, G. (2001), A system for program visualization and problem-solving path assessment of novice programmers, in 'ITiCSE '01: Proceedings of the 6th annual conference on Innovation and technology in computer science education'.
[38] Vizcaíno, A. (2001), 'Enhancing Collaborative Learning Using a Simulated Student Agent', PhD thesis, Universidad de Castilla-La Mancha.
[39] Williams, L. & Kessler, R. R. (2002), Pair Programming Illuminated. Addison-Wesley.

Francisco Jurado is a Visiting Lecturer in the Departament d'Informàtica i Enginyeria Industrial (Department of Computer Science and Industrial Engineering) at the Escola Politècnica Superior (Polytechnic School) of the University of Lleida, His research areas include Intelligent Tutoring Systems for Learning to Program, heterogeneous distributed eLearning systems, eLearning standards and Computer Supported Collaborative environments. He got his degree in Computer Science Engineering in 2005 and his Ph.D. degree in 2010, both by the Escuela Superior de Informática (College of Computer Science and Engineering) University of Castilla-La Mancha.

Ana I. Molina, received her Computer Science (2002) degree and her PhD (2007) from University of Castilla – La Mancha (Spain). She has joined the Escuela Superior de Informática (College of Computer Science and Engineering) of University of Castilla – La Mancha. In addition to teaching, her main interests are in the field of New Information Technologies applied to collaborative learning and computer-human interaction.

Miguel A. Redondo has a PhD in Computer Science (2002) and is an associate professor at the Escuela Superior de Informática (College of Computer Science and Engineering) of University of Castilla – La Mancha. His research interests focus on the fields of new Information Technologies applied to Computers in Education and Computer-Human Interaction.

Manuel Ortega is a Full Professor at the Escuela Superior de Informática (College of Computer Science and Engineering) at the University of Castilla - La Mancha (Spain). He received his BSc (1982) and PhD (1990) degrees from Universidad Autónoma de Barcelona (Spain). He is the director of the "Computer Human Interaction and Collaboration" Research Group. His main interests are in the field of New Information Technologies applied to collaborative learning and computer-human interaction.