Web standards based collaboration software for the classroom

by

Raúl Marzo, B.Sc.

Dissertation

Presented to the

University of Dublin, Trinity College

in fulfillment

of the requirements

for the Degree of

Master of Science in Computer Science

University of Dublin, Trinity College

August 2011

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Raúl Marzo

August 30, 2011

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Raúl Marzo

August 30, 2011

Acknowledgments

I would like to thank my supervisor Glenn Strong for his help and assistance throughout this project.

Raúl Marzo

University of Dublin, Trinity College
August 2011

Web standards based collaboration software for the classroom

Raúl Marzo, M.Sc.
University of Dublin, Trinity College, 2011

Supervisor: Glenn Strong

This dissertation focuses on a networked software system that aims to ease the organization and delivery of group-oriented activities in a learning-by-doing approach to education. The solution built is based extensively in the use of new web standards, like HTML5, which provided us with a new set of tools that helped to create a web-based platform-independent solution that implements an enticing graphical interface, which makes it appealing to students, and real-time communication at a reasonable cost. The design of the user interface pays particular attention to staying compatible with tablet-style devices. At the same time, the system eases the workload of instructors, giving them the possibility to monitor the progress of the groups of students that are using the system, provide immediate assistance, and obtain valuable feedback that enables instructors to adjust activities to the actual needs of learners.

Contents

Acknowledgments **iv**

Abstract **v**

List of Tables **viii**

List of Figures **ix**

Chapter 1 Introduction **1**

Chapter 2 Background and motivation **3**

2.1 Pedagogical background 3
2.2 Constructivism 5
2.3 Related work 6
2.3.1 Classroom Presenter [8] 7
2.3.2 DyKnow [10] 8
2.3.3 WIL/MA [28] 8
2.3.4 eFuzion [25] 9
2.3.5 Online whiteboard 9
2.3.6 Other interesting efforts 10
2.3.7 Feature summary 12
2.4 HTML5 12
2.4.1 Canvas tag 13
2.4.2 WebSockets API 14

Chapter 3 Description of the system **15**

3.1 Activities 16
3.1.1 Annotated slideshows 18
3.1.2 Diagrams 18
3.1.3 Parson's puzzles [23] 19

3.1.4 Categorization activity . . . 21
3.2 Student side . . . 22
3.3 Instructor side . . . 23
3.3.1 Creation of activities . . . 24
3.3.2 Exercise monitoring . . . 24
3.3.3 Session management . . . 25
3.4 Session example . . . 27

Chapter 4 System architecture **32**
4.1 Use of standards . . . 40
4.1.1 Canvas element . . . 41
4.1.2 WebSockets . . . 41
4.2 User interface considerations . . . 45
4.3 Usage of tablet PCs . . . 46

Chapter 5 Evaluation **50**
5.1 Student evaluation . . . 50
5.2 Expert evaluation . . . 53

Chapter 6 Conclusion and future work **58**
6.1 Conclusion . . . 58
6.2 Future work . . . 60

Appendix A Abbreviations **64**

Appendix B Main user controls **65**
B.1 Instructor home . . . 66
B.2 Student home . . . 67
B.3 Instructor live activity review . . . 68
B.4 Instructor inactive activity review . . . 70
B.5 Activity: annotated slideshow . . . 71
B.6 Activity: categorization and Parson's puzzles . . . 73
B.7 Activity: diagram . . . 75
B.8 Activity replay . . . 77
B.9 Work session management . . . 79

List of Tables

2.1 Review of systems for collaborative learning. 13

5.1 Mean survey responses: user evaluation, 4 respondents. 51
5.2 Mean survey responses: expert evaluation, 4 respondents 54

List of Figures

3.1 Slideshow, instructor side: students will receive in real-time the annotations made by the instructor. . . . 19
3.2 Diagram activity. . . . 20
3.3 Parson's Puzzle, initial state. . . . 21
3.4 Parson's Puzzle, finished. . . . 21
3.5 Categorization activity. . . . 22
3.6 Exercise replay. . . . 23
3.7 Live exercise monitoring. . . . 25
3.8 Post exercise monitoring. . . . 26
3.9 Session management. . . . 26
3.10 Activity selection for a session. . . . 27
3.11 Distributing students in groups. . . . 28
3.12 Example of session management. . . . 28
3.13 Binary tree definitions: annotated slide. . . . 29
3.14 Diagram activity: insertion in binary search tree. . . . 30
3.15 Parson's puzzle: Preorder tree traversal. . . . 31

4.1 Client-server architecture. . . . 34
4.2 Web server modules. . . . 35
4.3 WebSocket server architecture. . . . 36
4.4 Apple's iPad. . . . 48

B.1 Instructor home. . . . 66
B.2 Student home. . . . 67
B.3 Instructor live activity review. . . . 68
B.4 Instructor inactive activity review. . . . 70
B.5 Controls for annotated slideshow. . . . 71
B.6 Controls for categorization activity. . . . 73
B.7 Controls for diagram activity. . . . 75

B.8 Controls for activity replay. 77
B.9 Controls for session management. 79

Chapter 1

Introduction

During the last few years, laptops and tablet PCs have been introduced in classrooms extensively, both in elementary schools and universities. This phenomenon has happened both in first world countries with the introduction of one-computer-per-child pilot programs in schools, and also in developing countries thanks to initiatives like the One Laptop Per Child (OLPC) program [17]. Unfortunately, in most cases teachers don't take advantage of the potential for change in the learning experience that these devices could provide: blackboards have been replaced in some way by electronic whiteboards, paper and notebooks have been replaced by laptops, etc. As a result, the process of teaching is still predominantly based on the classic instruction, where knowledge is transmitted in a unidirectional flow in a teacher-centered environment.

In contrast to this, we introduce in this paper a computer-aided approach to education where students are encouraged to "learn by doing" in a collaborative environment, where they can be guided and receive feedback from the teacher, and also from peer students. To do this, we propose the use of a web-based system where the students are, in the first place, organized in groups, and secondly, work on a shared artifact in a set of activities specifically designed to promote collaboration among the members of the groups.

To implement this kind of system, we have tried to come up with a web-based platform-independent solution that provides an enticing graphical interface and, at the same time, allows real-time communication at a reasonable cost. These are in fact two important features of the still evolving HTML5 standard, which provided us with the tools to achieve both goals in an elegant way thanks to the canvas element and WebSockets, respectively.

Another factor taken into consideration was the rapid popularization of devices like Apple's iPad during the last few months, coupled with the appearance of a large number of new players in the tablet PC market. The range of features these devices provide and their ease of use have increased the penetration rate of these devices and improved

the notion users have of tablet PCs. As a result, some authors consider them as perfect substitutes of notebooks in a classroom environment [20], enhancing the possibilities for collaboration in the classroom given their wireless connection capabilities.

We also had in mind that most tablet PCs input revolves around a touch or multitouch screen, which are very different to the keyboard/mouse combination traditionally used in desktop and laptop computers. Given the cross-platform capabilities of the implemented collaboration system, the whole user interface had to be optimized for its use in different devices, allowing a similar user experience both in laptops and touchscreen-enabled devices as Apple's iPad.

The rest of this document is organized as follows. Chapter 2 presents an overview of the pedagogical background for this project, and describes previous efforts done in this field. In chapter 3, we detail the features included in the system built as a result of this project. Chapter 4 presents the system's architecture. Chapter 5 shows the results obtained from the evaluation of the software with actual users and experts in the field of IT and education. Finally, chapter 6 concludes and presents future work.

Chapter 2

Background and motivation

2.1 Pedagogical background

As we mentioned in the introduction, different programs have made widespread the use of computers in schools and universities. However, the use of computers by itself is not enough to enhance the process of instruction and learning that takes place in schools, which in most cases is still focused on the process of teaching, where knowledge is transmitted in a unidirectional flow in a teacher-centered environment. Additionally, in many cases technology only replaces its "analogical equivalent": blackboards have been replaced in some way by electronic whiteboards, paper and notebooks have been replaced by laptops or tablet PCs, etc. Therefore, if we want to get the most of computers in a classroom environment, the introduction of electronic devices in schools has to be coupled with an effort to change the way contents are presented using technology.

An interesting point of view of how instruction should be modified in order to accommodate the advantages of using computers in a learning environment is the one proposed by Alessi and Trollip [7]. They understand instruction as "the creation and use of environments in which learning is facilitated". Consequently, the goal of the whole instruction process is facilitate learning, and to do so in an efficient way it must include several essential activities:

- Presenting information
- Guiding the learner
- Practicing
- Assessing learning

The first three phases are based on research on successful classroom instruction [27], while the last one is proposed by Alessi and Trollip in order to help instructors to check the level of understanding the students have achieved after the first three stages have been completed. We explain what each stage consists of in the following paragraphs.

Presenting information To teach new content to students, firstly an instructor has to present the related information to them. This may be done in a number of ways: for example, it can be done using verbal or pictorial information, put together in a set of slides. Another way to perform this part of instruction is through example, presenting students with a series of samples that explain the basics of a particular matter.

The presentation of information can also happen without the mediation of an instructor: when a student reads a text book, watches a multimedia tutorial, or engages in an activity that involves seeking and understanding existing knowledge, information is also being presented.

Guiding the learner Once the contents to be learned have been presented, it's time for the students to start practicing them with the close guidance of the instructor. This process might involve the learners asking questions or trying to solve problems using the rules and principles they were previously presented. During this stage of instruction, the instructor is closely involved in the students' work, observing what they are doing, correcting the errors they might make, giving suggestions and hints in order to orient the learner, or repeating the exposition of the basic concepts presented if they weren't fully understood. These actions can also be performed by a fellow student who did understand the information provided, not only by the instructor.

This is for Alessi and Trollip probably the most important stage of instruction, which involves repeating a process in which the learners try to apply the new knowledge they were presented with, in combination with the guidance and corrections made by the instructor if they fail to use that knowledge correctly.

Practicing The third phase of instruction involves students practicing the contents that were presented in the previous two stages of instruction, in order to fully understand those contents, using them fluently. This phase is also considered learner-centered. Therefore, an instructor will only supervise students while they practice, pointing out the mistakes they might make. The goal of this stage of instruction is that students are able to master the contents presented beforehand, using them quickly or fluently, with few or no errors.

Assessing learning The previous three phases are usually considered the three main parts of instruction. Alessi and Trollip add a fourth step that consists in assessing the students' knowledge as it is built during previous stages of instruction. Its purpose is promoting students' learning instead of grading it, giving at the same time instructors the opportunity to adapt teaching to the actual learning needs of the learners. It usually involves a process in which teachers and students obtain feedback from each other over the course of the instruction. This feedback is usually gathered in a question-driven way, where students are presented with different problems that they have to solve individually or in groups; after some discussion, their answers are collected by the instructor and later discussed by the whole class. The main advantage of this phase, as pointed in [12], is that provides teachers with a notion of the actual understanding that the students have achieved, allowing them to adapt instruction to the actual state of the students' knowledge. It also helps students to be more engaged and motivated to learn, as they are also aware of the limits of their own understanding.

As it can be seen, stages two and three, as proposed by Alessi and Trollip, are learner-center, based in an approach that emphasizes "learning by doing". If we consider the possibility of performing these stages of instruction in a collaborative environment, where the student can be guided and receive feedback from the teacher and also from peer students, we will find that this approach resembles a constructivist view of education.

2.2 Constructivism

Constructivism argues that humans generate knowledge and meaning from an individual interpretation of what they perceive. Therefore, the constructivist learning theory maintains that knowledge is not received from outside, but it is constructed in our heads, learning being a process in which people actively build their knowledge. Accordingly, instruction should facilitate the construction of knowledge, providing the students with the tools necessary to perform the process of learning accordingly.

This means emphasizing the active process of learning —in contrast with teaching and instruction activities—, and promoting the discovery of information, done in a direct way by the students with the guidance of instructors. Moreover, as its name suggests learner construction is also an important part of constructivist learning environments, which often include activities like the construction of simulations or expert systems, construction of essays or newspapers, etc. Conducting this type of activities usually means that students have to set a goal, discuss and negotiate a plan to achieve that goal, and carry out that plan, evaluating the results obtained.

Another important aspect of this learning theory is related to the context in which the

activities are carried out, and how the knowledge gained while performing those activities can be transferred to other settings. An approach often used to achieve this goal is denominated "anchored instruction", and involves setting up learning activities embedded in a real-world environment. This way, students can relate the activities they are performing with real problems they might face in their life, thus improving the engagement of the students in the activity as they see it as meaningful and worth doing.

The constructivist approach also emphasizes collaboration, where learners are encouraged to work together with a common goal in mind. This enhances dialog, which becomes a fundamental part of instruction, either in small groups or as a whole class. This way students can share, challenge and compare their knowledge. Discussion is also helpful in order to consolidate their knowledge, as they have to articulate and express it, and also confront it with other different choices [9]. On the downside, there are also some potential disadvantages ascribed to collaborative environments: for example, some students may benefit more than others from this kind of setting [21].

Constructivism also puts emphasis on learning rather than on instruction, suggesting that learners should be more autonomous in their actions, making them feel that they choose what to do, and thus playing down the figure of the instructor. It's also important that students reflect about the act of learning, discussing in the classroom about the purpose, design and unfolding of the course; doing this, they reflect on how to learn, facilitating the development of lifelong learners.

Taking all this into account, we consider that an approach to the second and third stages of instruction (as described by Alessi and Trollip, guiding the learner and practicing) in which most of the principles of the constructivist view of education are taken into consideration, will provide a good basis to construct the collaboration software described in this document.

Now it is time to revise different software solutions that have been developed to enhance instruction in a classroom environment with the use of computers, and which we took as a reference during the initial stages of the development of this project.

2.3 Related work

With the introduction of computer in classrooms, and specially with the emergence of tablet PCs, several efforts have tried to make the most of these devices in this environment. Many of the resulting systems use "digital ink", provided by early implementations of Tablet PCs. These devices developed the pen and paper metaphor, allowing users to input data into a computer system in a familiar way: handwriting. In most cases, these systems also recognize different types of gestures, which are associated with common

commands. Additionally, their size is similar to the size of a notebook, so they can be easily manipulated. These devices usually include wireless communication capabilities to establish connection with a main access point, as it is common that these systems use a client/server approach. This communication is sometimes also established directly between peers.

As we mentioned in the introduction, during the months previous to the writing of this document devices like Apple's iPad have increased the penetration rate and improved the notion users have of tablet PCs, as they are smaller and easier to use. The most notable difference between iPad and previous tablet PCs is that Apple's product is controlled by a multitouch display that enables users to interact with the device directly with their hands, in an even more natural way.

2.3.1 Classroom Presenter [8]

This tablet PC-based software, developed at the University of Washington, tries to enhance the way contents are introduced by an instructor to the students in the form of interactive annotation-enabled slides. Contents are presented by an instructor as a set of slides that are broadcast to the students' devices over a network; as these slides are displayed, the instructor is able to stress any important point annotating the slide with digital ink. In the same way, the students can take private notes on their device for later revision. Additionally, the instructor can hand out exercises in the form of slides where an activity or question is presented: in this case, student annotations can be received by the instructor as anonymous submissions, which he or she can make public later for further discussion.

The two main advantages this software provides are, in the first place, that it gives instructors the possibility to obtain feedback from the students in real-time using its anonymous submission capabilities; the second one comes from the fact that students are encouraged to make use of the knowledge gained in the lectures, working individually or in groups on the contents presented in the classroom, and discussing their answers when they are made public by the instructor.

Ubiquitous Presenter [29]

Trying to complement the functionality provided by Classroom Presenter, Ubiquitous Presenter was developed at the University of San Diego aiming to address two main concerns about that software:

- Every student needs to have access to a Tablet PC in order to receive and annotate the slides.

- A late-joining student accessing a lecture enabled by Classroom Presenter won't be able to see the instructor's annotations made prior to the moment he or she joined the lesson.

To overcome these problems, Ubiquitous Presenter limits the use of Classroom Presenter on a Tablet PC to the instructor side: the slides presented to the classroom are transmitted to a web server, which translates them into pictures that can be accessed by the students using a web browser. These pictures are updated with every stroke of digital ink the instructor performs using Classroom Presenter, and with a short delay are downloaded to the students' laptops, who can annotate them using a keyboard. These annotations can be submitted to the web server, which does a reverse translation process that allows the instructor to receive the submissions as Classroom Presenter native slides.

2.3.2 DyKnow [10]

Developed by Dave Berque at the DePauw University in Indianapolis, DyKnow is, along with Classroom Presenter, one of the most widely deployed systems for collaborative note taking. Like Classroom Presenter, it makes the most of pen-based laptops or Tablet PCs, but in this case it also allows input via keyboard. In a best case scenario, in a classroom enhanced with DyKnow every student will have access to a pen-based Tablet PC. Meanwhile, the teacher will be able to use an electronic whiteboard where he or she can present the contents of a lecture, which can be annotated as they are covered. These annotations are broadcast in real-time to the students' devices, who can complement them with their own private notes, also using digital ink. This combination of notes can be saved at the end of a session for later review.

Similarly to Classroom Presenter, DyKnow enables anonymous submissions from the students that can be reviewed by the instructor, who can grade them or make them public for further discussion. This software also allows instructors to give temporary control of the main display (the one used by the teacher, which is showed in the whiteboard) to one of the students, who can present his annotations to the class.

Additionally, DyKnow provides instructors with the ability to monitor students' devices, blocking their screen if the Tablet PC is being used as a distraction (browsing the Internet, for example).

2.3.3 WIL/MA [28]

WIL/MA (Wireless Interactive Lectures in Mannheim) is a system developed at the University of Mannheim that enables instructors to receive feedback from students in large

classrooms. Implemented in Java, WIL/MA can be installed in different handheld devices. It has three main features:

- Present students with quick quizzes and questions: the students can submit their answers using WIL/MA, which will aggregate them and show the results in a graphical way. These activities are formulated in the form of multiple choice questions, fill-in text fields (for unconstrained answers) or clickable images.
- Gather feedback from the students about a particular aspect of the lecture.
- Ask the teacher direct questions that can be answered using WIL/MA, shared with the class, or replied in person by the instructor.

2.3.4 eFuzion [25]

eFuzion is a collaborative note taking system similar to the ones previously presented, which was developed at the University of Illinois. Similarly to Classroom Presenter or DyKnow, it allows instructors to present students a series of slides that he or she can annotate using a pen-based Tablet PC. The slides, along with the annotations made by the instructor, are transmitted over a wireless network to the students' laptops or mobile devices, who can also annotate them privately. It also facilitates the use of quizzes and activities that the students can answer using their mobile devices. These answers are gathered by the system, showing aggregated results in a graphical way.

SLICE [16]

Also developed at the University of Illinois, SLICE (or "Student Learn in Collaborative Environment") is an interesting software system implemented using an extreme approach to the Model-View-Controller design pattern. SLICE is not a ready built system for student collaboration, but a platform to build applications that empower this kind of learning environment.

It is composed by a nucleus, built in C#, which allows the execution of Python scripts that define the actual features of the applications and their view layer behavior. Examples of these applications comprise annotated presentations, classroom interaction (polls), shared code review or exam grading.

2.3.5 Online whiteboard

Specifically designed to implement distance learning courses, there are several online collaborative drawing tools available on the Internet. We now review two of the most com-

plete solutions based on the use of an online whiteboard.

Dabbleboard

Dabbleboard (http://www.dabbleboard.com) is an online application designed exclusively for collaborative white boarding. Its virtual whiteboard can be manipulated simultaneously by several users, allowing the use of freehand drawing, shapes, import images and even PDF files or PowerPoint presentations, which can be annotated. One notable feature it provides is its drawing interface, which is able to translate several types of strokes into the corresponding predefined shapes, like squares or diamonds. It also includes several communication channels (text chat, voice and video chat), and adds the possibility to save the contents of the whiteboard online, or export them as an image archive to a computer.

This software can be used in any computer with Internet connection, as it's built with Adobe Flash technology.

Blackboard Collaborate™

Blackboard Collaborate™ is another solution that, as Dabbleboard, comprises an online whiteboard that can be manipulated simultaneously by several users. It enables the use of freehand drawing, shapes, and import images or PowerPoint presentations that can also be annotated. Additionally, this software provides powerful tools to organize "virtual rooms", so users can collaborate in small groups. To promote collaboration in distance learning environments, it also includes Voice over IP and video chat, public and private text chat, and quick delivery of quizzes and polls. Blackboard Collaborate™ also provides the possibility to save the manipulations made on the virtual whiteboard, combined with the voice comments recorded during a session, for later review.

Blackboard Collaborate™ is designed to be used through a web browser, providing that the Java plugin is installed (it needs Java Web Start or Java SE to work).

2.3.6 Other interesting efforts

Apart from the previously presented systems, it's worth to briefly review some other efforts made in this field which provide fewer functionalities, are in a prototypical stage, or that were not specifically designed for a learning environment, but could be used in it with minimal modifications.

Computer aided Team-Based Learning [18]

This software is focused on a teaching technique called Team-Based learning, which emphasizes the application of the concepts presented as a mean to learn, applying new knowledge to more and more complex problems. To achieve this target, a common practice is conducting group activities. The authors of this project built a prototype that tried to overcome some of the problems that arise when setting up group based work in a classroom, using mobile devices to enable communication between teammates and also between the instructor and the students.

The prototype developed showed how, once the students had logged on, the instructor could tie them to a group and give them a problem to work with. Then, the prototype would provide a group of students with a shared working area, where they can modify the puzzle they are presented in real-time, and later send their answer to the instructor once it's been completed. The instructor would then receive the submitted work, being able to make the "puzzles" unavailable to the groups or share their answers with the whole class for further discussion.

Real-time document collaboration

[24] presents an application specifically developed for Apple's iPad that enables a group of people —each one using his/her own device— to mark up and share annotations on the same text. These annotations, which are tied to specific passages of text, are broadcast by the system to some or all of the users that have joined a working session. The annotations are permanently stored in a server, and kept visible for the users in a notification area of its user interface, which enables quick access to them.

Although it doesn't seem that this system was developed specifically for a learning environment, this approach overcomes the limitations to work in groups in a single text, even allowing remote interaction. These factors make this kind of system suitable for its use in a classroom environment.

ClassInHand [11]

Developed at Wake Forest University, ClassInHand is one of the first classroom collaboration tools built. Its functionality is similar to systems like Classroom Presenter or DyKnow, including remote control of regular PowerPoint slideshows. It also enables instructors to pose quizzes and polls among the students, whose numeric or textual answers are aggregated and displayed by the system in real-time at the instructor's device.

In contrast with the aforementioned systems, ClassInHand was developed to be used in small mobile devices like Pocket PCs. Another particularity of ClassInHand is that

it doesn't need to run in an external server in order to provide access to the contents shared by the instructor. Instead, it runs on a small footprint web server executed in the instructor's mobile device, which serves the contents of the lecture to the students.

Google Docs

Although it wasn't originally created to enable collaboration in a classroom environment, Google Docs and similar other cloud-based office suites (like Windows Live Office) provide a set of tools that could be used to enhance the online creation and edition of documents, while collaborating in real-time with other users. Among all the features this software includes, the most relevant in this environment are:

- Creation, edition, display and sharing of text documents, spreadsheets, presentations and diagram by multiple users at the same time.
- User notification of changes performed to any specified regions, via e-mail.
- Computer-mediated communication, which includes time-stamps, profile pictures, ownership rights, comment resolution, and integration with email and notification settings.
- Use of visual cues (e.g. cursors with different colours are displayed for every user concurrently editing a document) that improves user awareness.
- Revision history, including the author of the changes that correspond to each version of a document.

2.3.7 Feature summary

In table 2.1 we summarize the main features that each of the previously described systems enable in a collaborative learning environment. There are two main reasons behind the selection of these particular features: on the one hand, we want to show which of these features are supported by each of the systems described earlier; on the other hand, we are also interested on the hardware and software platforms used to implement them, in order to find the most suitable combination for this project.

2.4 HTML5

As we mentioned in the previous section, web browsers displaying HTML files can be used to provide access to the contents used by an instructor during a lecture. This field of

	Annotated presentations	Student annotations	Student submission	Instruction replay	Group oriented shared artifact	Real time broadcast	Surveillance	Hardware platform	Software platform
Classroom Presenter	✓	✓	✓	?	✗	✓	✗	Pen-based Tablet PC	Based on MS PowerPoint
Ubiquitous Presenter	✓	✓	✓	✓	✗	✓	✗	Tablet PC & Web browser	MS PowerPoint & HTML
DyKnow	✓	✓	✓	✓	✗	✓	✓	Pen-based Tablet PC White-board	KPL Corba
WIL/MA	✗	✗	✓	✗	✗	✗	✗	Handheld devices	Java
eFuzion	✓	✓	✓	?	✗	✓	✗	Pen-based Tablet PC	.NET (C#) Python
SLICE	✓	✓	✓	?	✗	✓	✗	Pen-based Tablet PC	C# Python XML
Dabbleboard	✓	✓	✗	✗	✓	✓	✗	Web browser	Adobe Flash
Blackboard Collaborate™	✓	✓	✓	✓	✓	✓	✗	Web browser	Java
Computer aided Team-Based Learning	✗	✗	✓	✗	✓	✓	✗	Mobile device	Java
Real-time document collaboration	✗	✗	✗	✗	✓	✓	✗	iPad	Objective-C
ClassInHand	✗	✗	✓	✗	✗	✗	✗	Pocket PC Web Browser	HTML
Google Docs	✗	✗	✗	✗	✓	✓	✗	Web browser	Web based

Table 2.1: Review of systems for collaborative learning.

computer science is constantly evolving, and in the last few years has been revolutionized by the implementation of new standards that enhance the experience of using a web browser. One of the cornerstones of this revolution is HTML5.

In an effort that started in 2004, HTML5 is being developed as a new version of HTML4, XHTML1, and DOM Level 2 HTML, that, in combination with CSS3 and JavaScript, provide a whole range of new possibilities for the development of web applications. In this section, we review the most relevant features HTML5 provides for the implementation of this project.

2.4.1 Canvas tag

The HTML5 standard defines the <canvas> element as "a resolution-dependent bitmap canvas which can be used to rendering graphs, game graphics, or other visual images on the fly" [13]. Drawing is performed using specific JavaScript functions designed for this purpose, which manipulate graphics at a pixel level. They allow drawing lines, simple

shapes (as rectangles) and more complex shapes, text, the incorporation of images that can be modified in different ways (scale, translate, rotate), smooth transition between colours (gradients), background patterns, and shadows. Additionally, the contents drawn in a canvas element can be exported as an image file (in PNG or JPEG format). All these capabilities make this HTML5 component a suitable candidate for the implementation of complex graphics that enhances the presentation of contents in a platform independent way, with the only requirement of using a compatible web browser.

2.4.2 WebSockets API

This communication feature added in HTML5 is specially important, as it allows full-duplex communication between a server and a web browser, overcoming many of the current problems that exist in order to implement real-time web applications. Additionally, the traffic generated and the latency in communications, in comparison to current implementations (polling, long-polling, streaming), is greatly reduced with the use of WebSockets.

The use and current state of both HTML5 components is discussed in depth in section 4.1. of this document.

Chapter 3

Description of the system

Revisiting table 2.1, which summarizes the main features provided by systems created for collaborative learning, we can observe that none of the tools revised complies with all the features we have considered necessary to facilitate group collaboration in a classroom environment.

Systems like Classroom Presenter, Dyknow or eFuzion are focused on a traditional classroom environment, where the instructor presents new contents to the students using a series of slides, who can also annotate them privately using digital ink. Systems specifically designed to enable distance learning, like Dabbleboard and Blackboard Collaborate^{TM}, also include this functionality. This type of interaction fits with the first phase of the model for successful instruction proposed by Alessi and Trollip in [7]. Therefore, we considered providing this feature by implementing a slideshow-based activity, where instructors can make annotations in order to complement the contents presented in traditional slides.

Continuing with this review, we observed that one of the basic features that almost all the systems reviewed support is the submission of the work done by the students, so instructors can evaluate it. This is a feature that the system presented in this document will offer, with an added improvement: the instructor will not only be able to check the final result of the work carried out by a group, but also explore step by step the process followed to construct that solution. Additionally, we have also considered useful allowing the exchange between different groups of their respective solutions, in an effort to provide students with different views and answers to a particular problem, so they can contrast their knowledge with that of their fellow students.

Another important feature supported by several systems reviewed, and that we considered very important for group collaboration, is the use of a shared artifact that can be manipulated by all the members of a group. At the same time, the modifications made by every member of a group are broadcast to the rest of them, so every person in the group

is aware of the latest state of the shared artifact. The usage of a shared artifact also has a side benefit for collaboration: as every component of the group has access to the artifact that represents the problem to be solved, each person in the group is able to manipulate it, avoiding in some degree the emergence of "dominators", thus encouraging shy students to take active part in the activity. The identification of the user who performs every action on this artifacts would allow the instructor to identify who are the most active members of the group, and even know if their contributions were correct or not.

A feature only supported by DyKnow, which we considered also necessary for our system, is the ability to monitor the students' use of the tool. Including this capability, instructors will be able to monitor and control the state of the groups' work at any given time, preventing distractions as a result.

With respect to the platform used to build the system, the use of web standards allows us to build a solution that doesn't need the use of specific hardware or software. This enabled us to overcome the main problem that systems like Dabbleboard or Blackboard Collaborate™ suffer: while they can be used using a regular web browser in a laptop computer, both need third party plug-ins to work (Adoble Flash in needed to run Dabbleboard, Java SE or Java Web Start are required to execute Blackboard Collaborate™). Unfortunately, none of these plug-ins are supported by Apple's iPad.

With respect to the system user interface, we aim to provide a good user experience in touchscreen enabled devices like Apple's iPad; all the tools provided can be used using a HTML5 compatible web browser, like Webkit-based browsers (Safari), Chromium-based browsers (Chrome) or Gecko-based browsers (Firefox), and eventually future versions of Microsoft Internet Explorer. Additionally, we were able to easily modify and optimize the layout of the application for different devices with the use of specific CSS style sheets.

Also, as a consequence of its web-based nature, the tool presented in this document can be used either in a classroom environment, with the on-site supervision of an instructor, or remotely via Internet, enabling group-based distance learning. To enhance the latter scenario, each group is provided with a computer-mediated communication system (a chat room), where its members can discuss the next steps to take in order to provide an answer to the problem they are working on, or even ask directly the instructor any question they might have.

3.1 Activities

Almost as important as the features the system provides, we had to find the kind of activities and exercises that are best suited for the platform we have built, in order to address the second and third phases for successful instruction that we mentioned in chap-

ter 2: guidance and practice. Additionally, we want to help performing these stages of instruction following a constructivist approach to education and learning, thus emphasizing teamwork and communication among students, and also between students and the instructor.

To find the activities that are best suited for group collaboration, and also identify additional features that could be worth including in the system, we followed the guidelines proposed in [22], which aim to avoid three main problems inherent to group work. The first of these problems is related to the role that each of the members of the group assumes: normally, one or two individuals tend to dominate discussions; at the same time, the opinion of quieter people is often unexpressed or simply ignored. The second problem comes with the difficulty that groups have to stay focused on their main task, diverting their efforts in irrelevant details. The third problem arises when groups share their solution with the rest of the class, as in many cases this discussion doesn't really contribute to the common knowledge.

By requiring high individual accountability, an instructor who is in charge of a group-based activity minimizes the risk that all the work in a group is done by one or two individuals. At the same time, this fact causes members of a group who don't contribute to be seen by their group peers in a negative way. This fact can positively affect the results achieved, as all the group members are expected to provide some input. In our case, we consider that the monitoring capabilities of the system, in combination with the recording of the interactions made by each member of the group in order to modify the shared artifact in which the group works on, will make the system comply with this first condition for successful group work.

Other factor we had to keep in mind is that the activities proposed have to present an appropriate level of difficulty, so collaboration among the members of a group is needed to accomplish the task's goal. At the same time, we should avoid exercises which require inherently individual activities like writing, as the only actual group activity undertaken will be how to divide the work to be done. Using an interaction model that tries to minimize the need for writing, we will minimize the risk associated with task fragmentation. On the other hand, it will be the instructors' responsibility to devise activities that have a suitable level of difficulty, spurring the students' motivation to work as a group.

Additionally, in order to improve bonding among the members of the group, is also interesting to provide students with external feedback that challenges the group's knowledge. This is usually best done by direct comparison with the output of other groups, creating situations where the results of the work of a group is scrutinized and challenged by peers from other groups, feeding the basic human need for social validation [22]. As the system presented in this paper allows instructors to expose a group's solution to the rest

of the class, leading to discussion and comparison, we consider the system accomplishes this precept. It also noteworthy that the comparison not only involves the final solution achieved, but the whole process undertaken, as it can be replayed from beginning to end.

Having the previous points in mind, we have considered that the following types of activities can be used to foster communication and exchange of ideas between the members of the groups that work on them.

3.1.1 Annotated slideshows

Although the system we present in this document is focused on group activities, we have considered that instructors should have the possibility to present a set of basic contents that the students will require to address the subsequent group activities. To do so, the application allows instructors to display a series of slides in order to explain the basic concepts to be used. These slides can be annotated to highlight particular points; both the slides and the annotations made are transmitted in real-time to the students, who can ask the instructor questions using the computer-mediated communication channel provided (text chat).

Additionally, the annotated slides will also be available to students for later revision, providing the possibility to replay the whole presentation step-by-step, walking through the slides as the annotations were made.

3.1.2 Diagrams

Many concepts in computer science (and also other knowledge fields) can be explained making use of graph based diagrams. In most cases, these pictorial representations take a collection of items and relationships between them, and express them by giving each item a 2D position, while the relationships among them are denoted as connections between the items.

As it can be observed in figure 3.2, the students that enroll in this type of activity are provided with a shared work area (in white in the image) where they can add different types of basic shapes and nodes that can be manipulated in order to change their position, size and label. These shapes can also be connected, using directed or non-directed arrows.

All the manipulations made by the students who form a group are broadcast immediately to the rest of the groups' peers, so all group members have an up-to-date view of the state of the diagram. Similar to the case of the slideshows, if the students aren't physically in the same location they can exchange ideas using the computer-mediated communication channel provided (text chat). Additionally, they can ask for help directly to the instructor, who will be notified of this request. As a response to this help request,

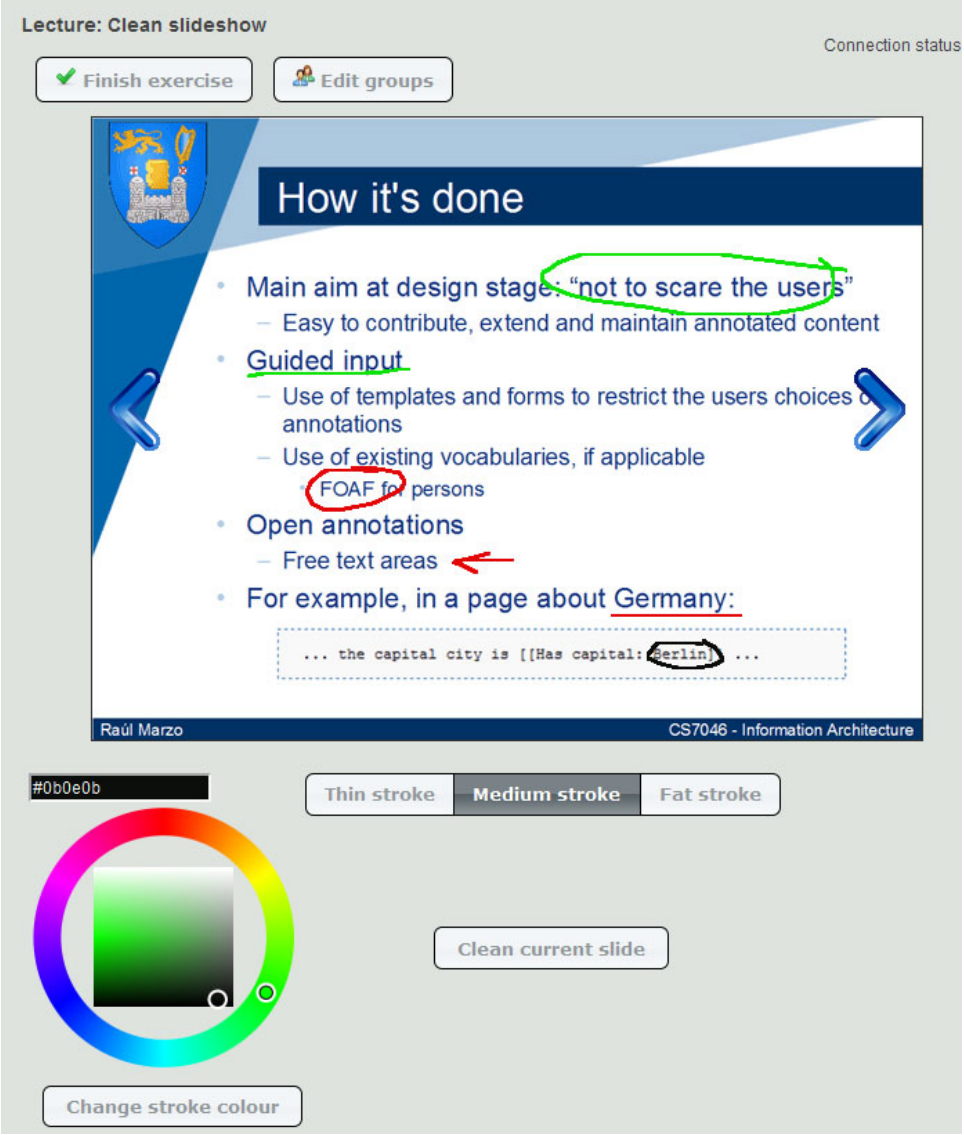


Figure 3.1: Slideshow, instructor side: students will receive in real-time the annotations made by the instructor.

instructors can send a textual tip to the requesting group, to all the groups, or alternatively join the group's shared workspace in order to manipulate directly the elements of the diagram. We further explain these cases in a later section of this document.

3.1.3 Parson's puzzles [23]

This kind of puzzle, proposed by Dale Parsons, is an interactive tool that allows training in basic programming principles via drills, exposing students to good programming practices. Its operation is actually very simple: students are provided with the description of the program to be built using the pieces of code they are provided. Then, they have to

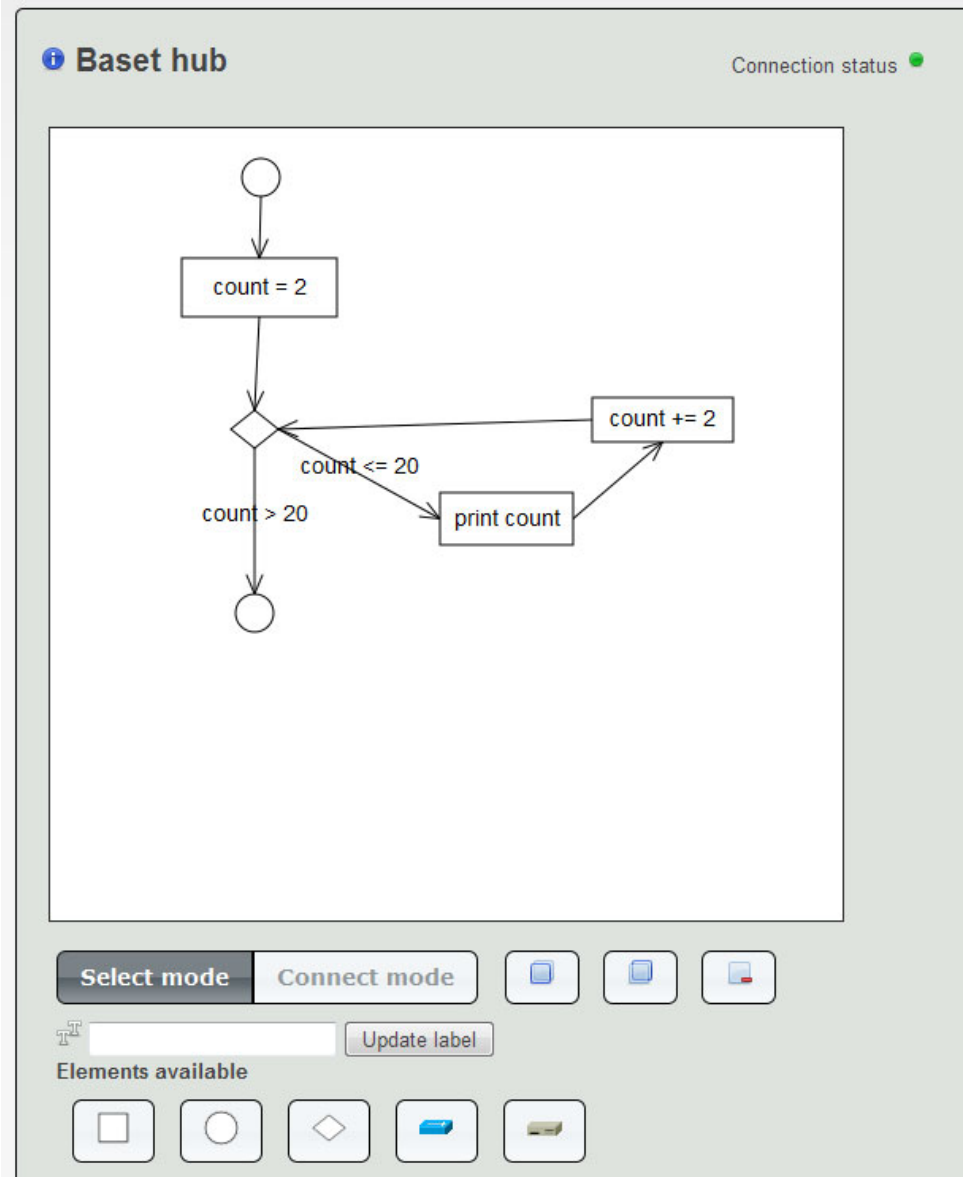


Figure 3.2: Diagram activity.

choose the correct pieces of code, and sort them in the correct order to achieve a complete program or subroutine that implements the program description given.

We considered the implementation of this particular tool because it fits two of the main targets of the system: it's well suited to promote group collaboration, and at the same time provides an artifact that doesn't require direct text input, as the manipulation of the pieces can be done with simple drag & drop gestures.

Just like the previously described activities, the students are provided with a computer-mediated communication channel to enhance discussion in case they are not in the same physical space. They can also ask the instructor for help at any given time.

Figure 3.3: Parson's Puzzle, initial state.

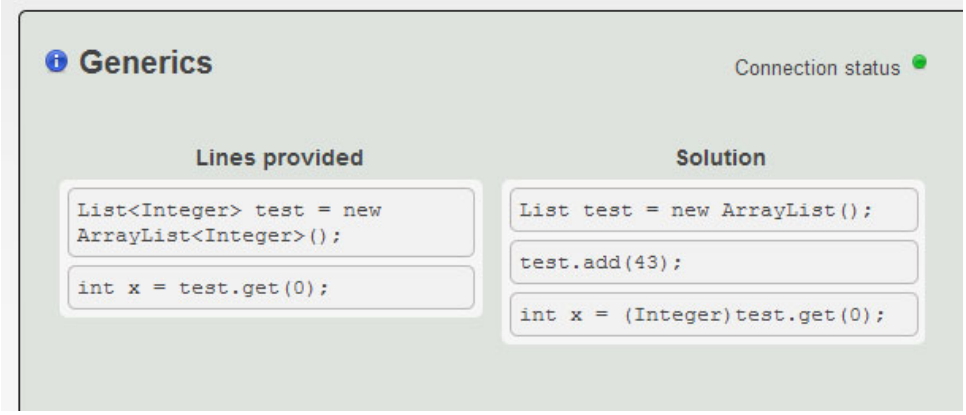


Figure 3.4: Parson's Puzzle, finished.

3.1.4 Categorization activity

A third type of exercise that we believe suits the guidelines for successful group activities, consists in a categorization activity where the students have to cluster together a set of words that define or are somehow related to a particular term.

An example of use of this particular kind of activity may consist, as it's shown in figure 3.5, in the identification of legal or illegal identifiers, given a set of guidelines about their validity. The students would be given a list of possible identifiers, which they have to drag and drop into the legal or illegal zones, which are placed on the sides of the activity layout.

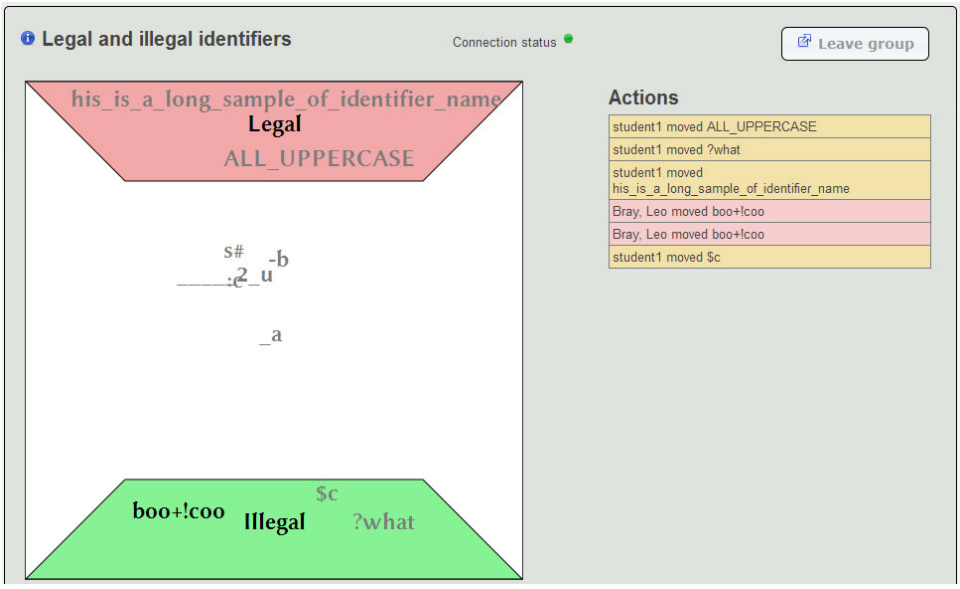


Figure 3.5: Categorization activity.

3.2 Student side

The previously described activities are available for the students that take part on a course which uses this web-based system to enhance group collaboration. When students successfully log in to the system, they are shown a menu where, in the first place, the user can select to join in one of the active activities that are associated with any of the courses the student is enrolled. A second tab allows students to access past exercises that have been previously completed, which are available for revision any time. Finally, the interface also allows student to check general information on the courses they have enrolled.

When a student joins his designated group for a particular activity, he will be shown with one of the interfaces presented in the previous section, depending on the kind of activity he or she is joining. This interface is composed of the shared working area, the basic set of controls needed for its manipulation, a complete log of the actions undertaken by every student in the group, a list of the members of the group (including the activity's supervisor), and a text chat for computer-mediated communication. It's also noteworthy that, in case a student joins an activity that has already started, he or she will be able to receive all the changes made prior to that point, having an up-to-date view of the shared artifact.

In case the user wants to review an activity completed in the past, he or she will be shown a slightly modified interface (figure 3.6) that includes several changes with respect to the one used for live exercises. Like in that case, it consists of the shared

artifact the group members worked with, and the log of both the actions undertaken and the conversations held during the course of the activity. The main particularity of this interface comes with the controls that can be seen over the shared work area: using these controls, the student will be able to revise, step-by-step, how the shared artifact evolved over time. It's also possible to go backwards and forwards action-by-action or directly to the start/end of the exercise, and even to the point when a particular manipulation was performed, clicking the corresponding log line in the action log.

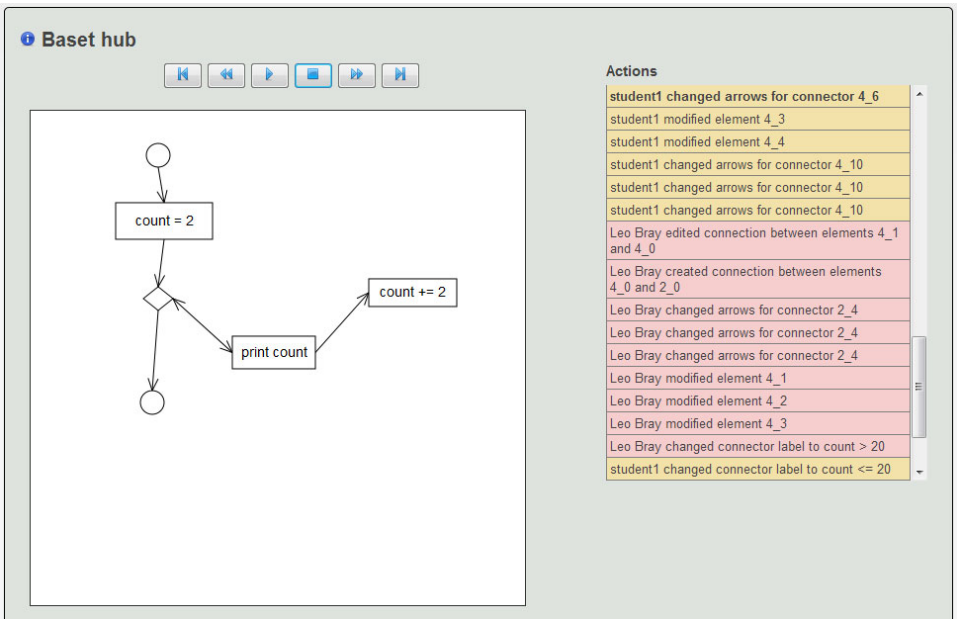


Figure 3.6: Exercise replay.

3.3 Instructor side

From the instructor's point of view, the system offers a bigger set of tools that allow a lecturer to create new instances of each of the four types of activities described in section 3.1., organize work sessions that include several activities and the creation of stable student groups, and also manage those activities while a work session is in progress.

Once an instructor has logged in into the system, is shown a customized home page that includes a list of active exercises he or she has launched, a log of the exercises done in the past and marked as finished, a list of the work sessions previously created and an index of all the activities available, both created by the user or by other instructors.

3.3.1 Creation of activities

In the following lines we give a short explanation of how an instructor can create new activities like the ones previously described.

Slideshows In order to create a new slideshow, we opted to keep things simple: slides should be created with any specific program for this task, like Microsoft PowerPoint, and each of the slides has to be exported as an image. These images can be later uploaded to the website, and sorted accordingly to shape the slideshow.

Diagram The preparation of a diagram activity is quite simple, as the instructor only has to specify the goal of the exercise and establish the set of icons that the students will be able to use. The icons available always include by default a basic set of shapes (square, ellipse, diamond), which can be complemented by with specific icons added to the system. To extend the set of available icons, instructors can upload new images to the server.

Parson's puzzle The creation of new Parson puzzles is also trivial: the lecturer only has to type, line by line, the pieces of code that the students will be able to manipulate in order to build a program that complies with the program description specified. To increase the level of difficulty of this kind of activity, a recommended practice suggests the inclusion of "distractors" among the lines provided, which are erroneous pieces of code added to help the instructor illustrating a particular point, like commonly repeated errors.

The instructor can also include the correct answer to the program, which can be released once the activity is finished, so the students can compare it with their own answer.

Categorization activity To create a new categorization activity, an instructor has to type the activity goal, define the words or terms to be classified (one per line) and specify up to four categories, which correspond to the sides of the shared work area that will be later shown using different background colours.

3.3.2 Exercise monitoring

One important feature that the system includes is the monitoring of the groups' work while it's being carried out. When an instructor releases an activity so the students can work on it, he or she has the ability to supervise the state of the groups' shared artifacts, as the collaboration among the students develops.

The system offers instructors with the possibility to gather a general view of the state of the class, having access to a screenshot of the current state of all the groups' shared artifacts. These screenshots can be updated any time, so the instructor can always have a clear picture of how the students are doing.

Additionally, in case a group of students has asked for help, the instructor or lecturer has three options to satisfy this request: first, he can provide some feedback in the form of a textual pointer, specifically sent to the requesting team; secondly, if the problem can affect several groups, this message can be sent to all of them. Finally, if the students who asked for help are really far from reaching the exercise's goal, the instructor can join the group's work area, manipulate the shared artifact as a regular member of the group, and provide feedback using the corresponding text chat.

Figure 3.7: Live exercise monitoring.

When an exercise has been flagged as finished, the instructor has several new options available. On the one hand, he can replay any of the groups work in order to grade it, and assign marks to their effort. Alternatively, the shared artifact of a group can be shared with the rest of class to promote discussion. Again, the students in the receiving groups can walk through the whole process undertaken to get to the achieved solution step-by-step, gathering a valuable knowledge about how the resolution of the problem developed over time.

3.3.3 Session management

To ease the workload at the instructor side, we have considered necessary the definition of "work sessions", which tie together a set of exercises that are planned to be executed

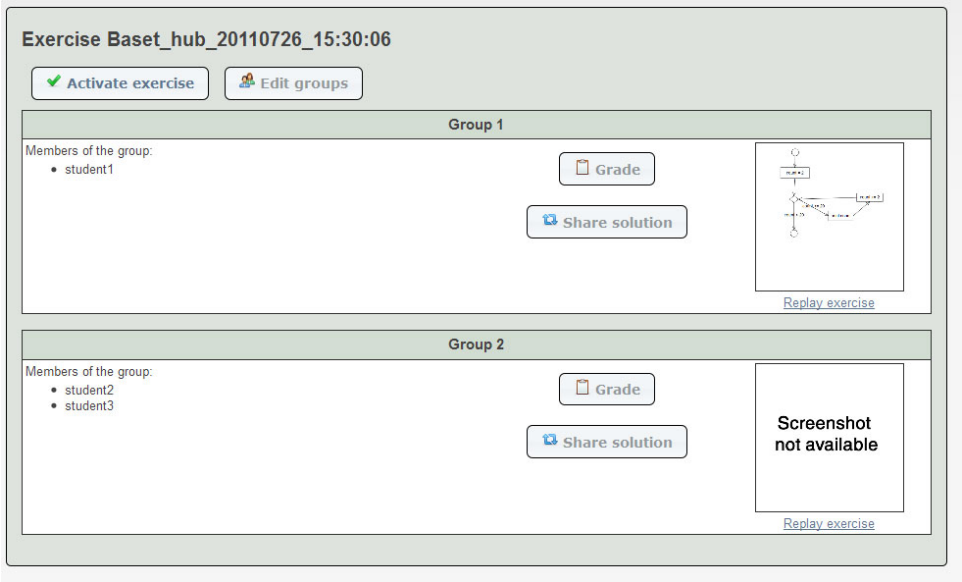


Figure 3.8: Post exercise monitoring.

during a working period. Additionally, to allow a seamless transition between activities, and with the aim to improve group bonding, the students are organized into groups that remain unchanged throughout a session.

Once these groups are established and the set of activities to be performed is defined, the instructor is able to manage the whole work session in a centralized fashion using the following interface.

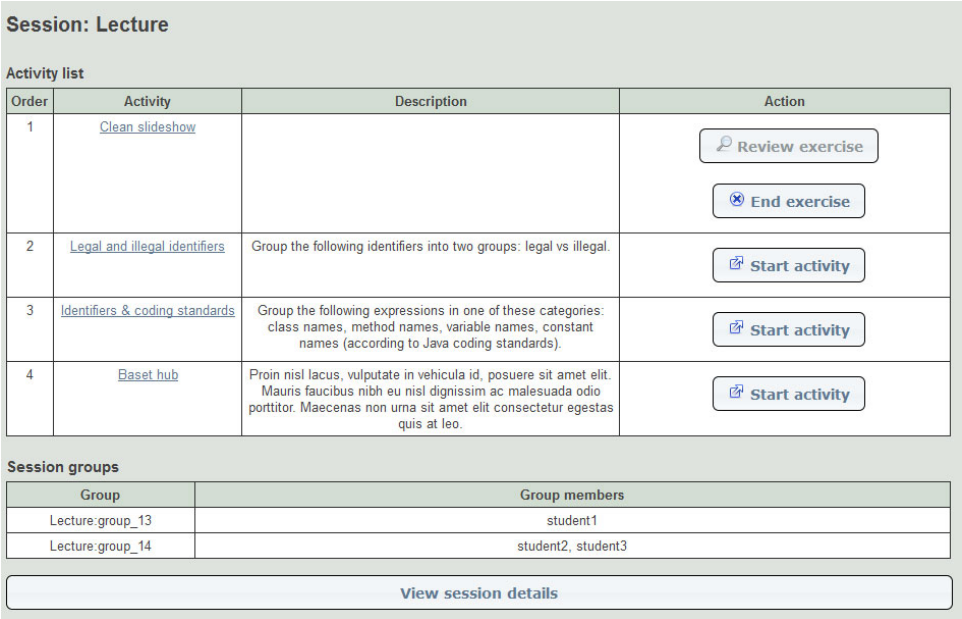


Figure 3.9: Session management.

As it can be seen in figure 3.9, the session management section allows instructors to start and finalize the activities that compose the session, and also access the monitoring area that corresponds to each of those activities.

3.4 Session example

In the following paragraphs we are going to describe how a work session should be prepared and developed. For this example, we decided that the session will comprise a series of activities that present the learners with the basics of binary search trees, and a set of drills designed to help them understand and practice the concepts introduced.

First of all, the instructor will have to choose the activities that will be carried out during the lesson. As we mentioned before, we recommend to start presenting the basic concepts needed to successfully complete the exercises and drills that compose the guidance and learning sections of the lesson.

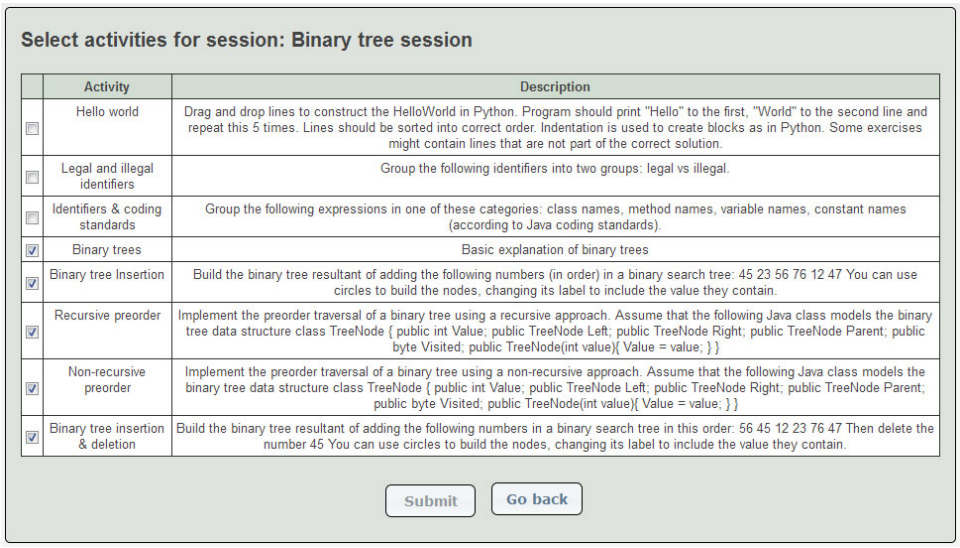


Figure 3.10: Activity selection for a session.

Consequently, in the case we are describing in these lines, the session should start with a slideshow activity that presents the basic theoretical concepts that defines binary search trees, including how to insert and delete elements in this particular data structure, and the algorithms used to traverse the information contained in one of these trees. Then, the instructor can add a series of drills that will give the students the opportunity to practice those concepts by themselves. Figure 3.10 shows the activity selection form that allows instructor to pick the activities for a session. It's worth mentioning that all exercises stored in the system, regardless of who created them, are available for all instructors at the time of selection.

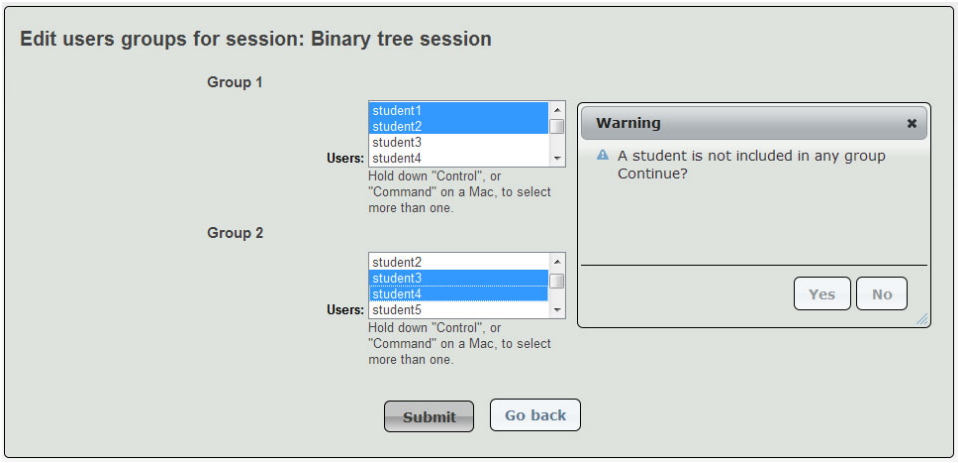
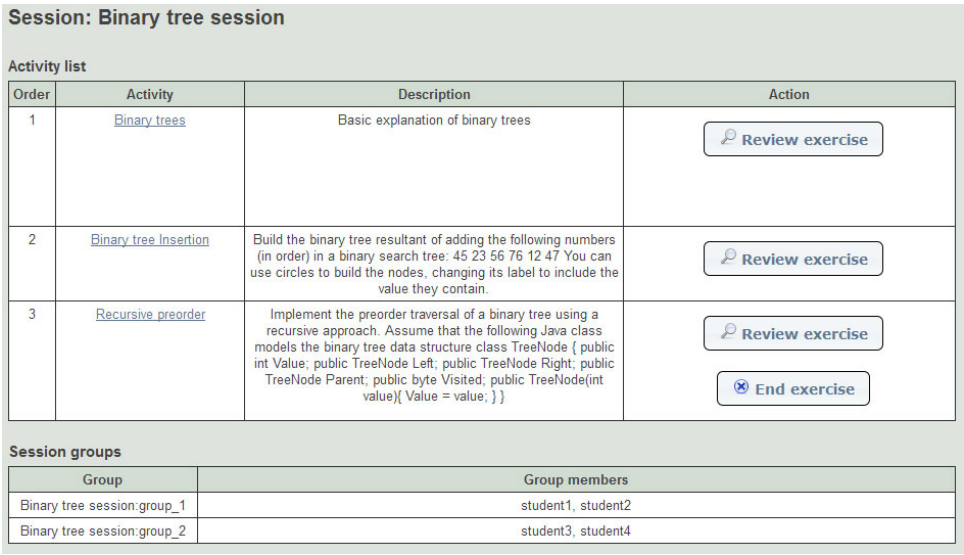


Figure 3.11: Distributing students in groups.

Once the activities for the session had been selected, the instructor has always the possibility to add or delete exercises from it, and sort them according to the order he or she plans to present them to the students. There is just one more step before the lecture can effectively start: distributing the students into groups. This feature is quite convenient for instructors, as these groups can be maintained during the course of the whole session, thus avoiding further configuration at this respect. To perform this task, the instructor is provided with a form where he or she can select the students to be assigned in each group (figure 3.11). When this form is submitted, the system will check that all the students that take part in the course are assigned to a group, and also that there's no student assigned to several groups at the same time.



Session: Binary tree session

Activity list

Order	Activity	Description	Action
1	Binary trees	Basic explanation of binary trees	Review exercise
2	Binary tree Insertion	Build the binary tree resultant of adding the following numbers (in order) in a binary search tree: 45 23 56 76 12 47 You can use circles to build the nodes, changing its label to include the value they contain.	Review exercise
3	Recursive preorder	Implement the preorder traversal of a binary tree using a recursive approach. Assume that the following Java class models the binary tree data structure class TreeNode { public int Value; public TreeNode Left; public TreeNode Right; public TreeNode Parent; public byte Visited; public TreeNode(int value){ Value = value; } }	Review exercise End exercise

Session groups

Group	Group members
Binary tree session:group_1	student1, student2
Binary tree session:group_2	student3, student4

Figure 3.12: Example of session management.

When these first steps are completed, the instructor will be ready to start the lecture, having access to the session management section (see figure 3.9). From here, the instructor can launch the execution of the activities —delivering them to the students—, access the monitoring section that corresponds to active exercises, and conclude the activities in progress.

As we mentioned before, we recommend to start with a slideshow where the lecturer presents the basic information the students need to later practice in a set of related exercises. For example, in the case of binary search trees, some of that information is related to the terminology needed to define the parts of the tree. In figure 3.13, we can see how the instructor highlighted with different colours the parts of a tree that corresponds to each of the terms defined in the slide:

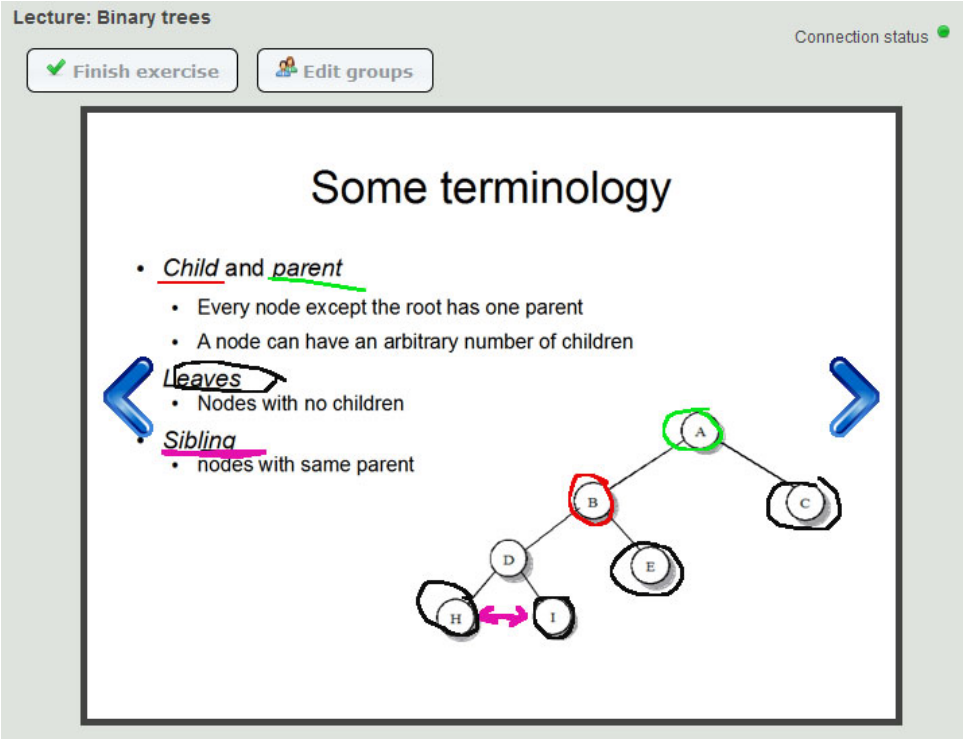


Figure 3.13: Binary tree definitions: annotated slide.

- The instructor used the green and red colours to denote a pair of parent and child nodes, underlining the terms in the text, and drawing a circle over the corresponding nodes in the example tree.
- Black was used to remark the term "leaves", using the same color to highlight the leaves nodes in the tree.
- Purple is used to underline the term "sibling", and it's also used to draw an arrow that connects two sibling nodes in the example tree.

When the slideshow is completed, the activity can be marked as concluded using the session management section of the system. At this point, the instructor can start providing the students with activities where they can practice the contents they have just received.

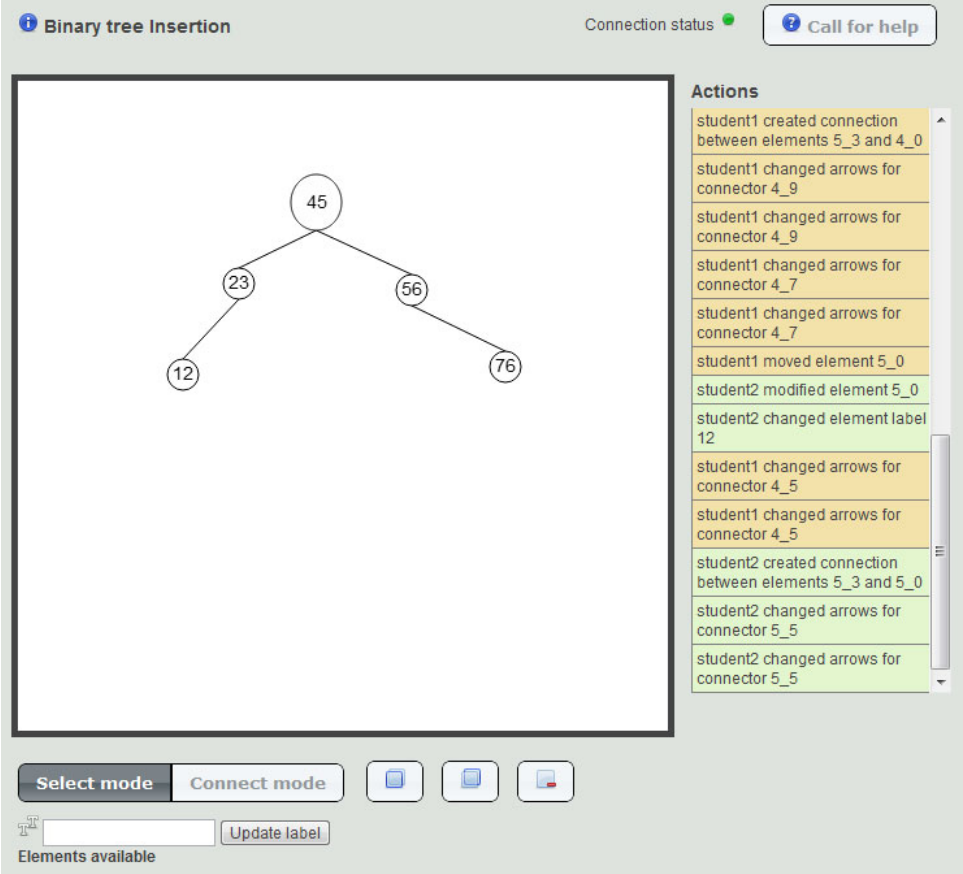


Figure 3.14: Diagram activity: insertion in binary search tree.

Continuing with the session dedicated to binary search trees, the lecturer can offer his students the possibility to perform a diagram-based activity in which they will have to build a tree, inserting in order a set of numbers (see figure 3.14).

When this exercise is activated, the instructor will be able to monitor the progress of the groups of students as we mentioned in an earlier section of this document, checking the state of their shared artifact and interacting with the student groups using some of the communication channels provided by the system. Additionally, this diagram-based activity could also be complemented on the fly: if all the groups have managed to build the binary search tree correctly, the instructor can ask them to delete some of the nodes, thus further practicing their knowledge in relation with the construction and update of binary search trees.

As the goals for the previous activities are achieved, the instructor can choose to

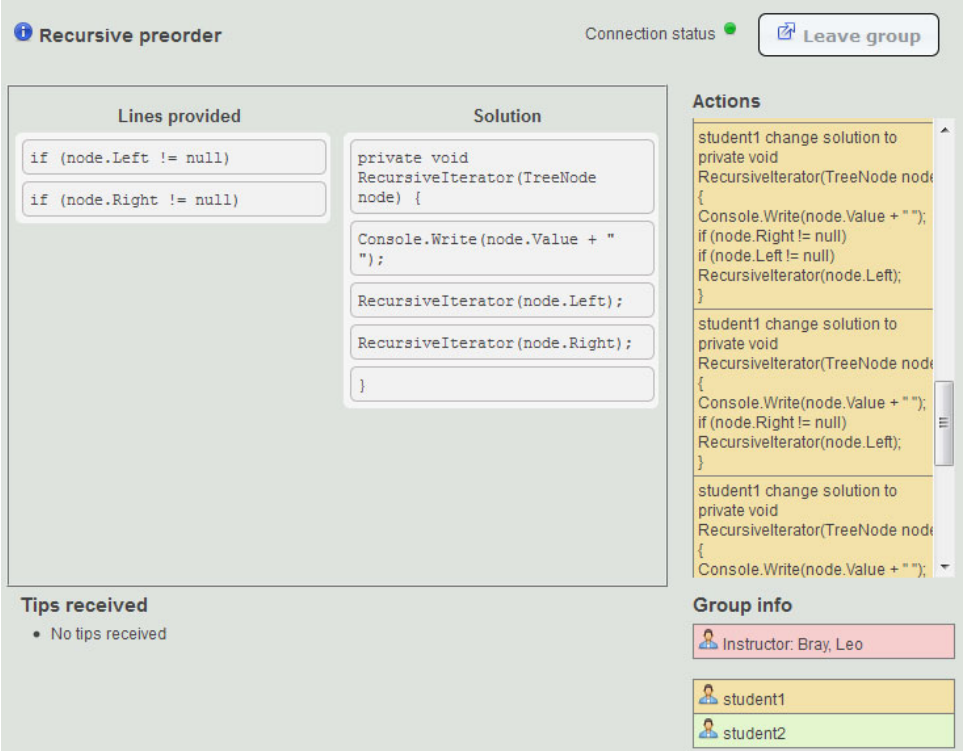


Figure 3.15: Parson's puzzle: Preorder tree traversal.

present the students with other kind of exercises. For example, the lecturer can ask the students to build an algorithm to traverse a binary search tree in a particular order, providing a basic implementation of this data structure as part of the activity description. To do this, he or she can disclose a Parson's puzzle that provides the pieces of code needed to implement that algorithm using recursion (see figure 3.15).

This last activity could be complemented with the inclusion of another puzzle that provides the pieces to build an iterative version of this program. This way, the instructor can provide the students with an alternative to the recursive algorithm, and show them the differences between these two approaches to the problem of traversing a binary search tree.

Chapter 4

System architecture

As we mentioned in the introduction of this document, we have tried to build the system described in the previous chapter following a web-based approach. The main reason to follow this course of action is that the resultant application is platform-independent, so there is no need to run the system on specific hardware, or create different versions of it to be executed in different devices; any computer equipped with a modern web browser can be used to get access to the system. At the same time, some of the new features introduced by new web standards, like HTML5, provide the basis to build a solution that includes an enticing graphical interface and, at the same time, allows real-time communication at a reasonable cost.

Therefore, given its web-based nature, we opted to implement the system following a classic client-server architecture. At the beginning of the project we also took into consideration the use of an infrastructure in which users' devices would communicate directly, establishing an ad hoc network. Considering the fact that we had to know beforehand the IP address of the devices in order to build the corresponding URL needed to establish a WebSocket connection between the nodes, we concluded that it would be impractical to implement the system this way in a real world environment.

With respect to the development of the basic web interface, we opted to use the Python [6] based Django web framework [2], which implements a variation of the well-known model-view-controller (MVC) architectural pattern. In this pattern, the "model" refers to data access layer; the "view" is in charge of rendering the model, shaping the user interface; and the "controller" refers to the part of the system that decides which view is displayed given a particular user input, manipulating the model as needed.

In the particular case of Django, the "model" is handled by Django's database layer, which offers convenient methods to access, validate and define relationships between the data manipulated; the "view" is handled by views and templates, which build the web

contents displayed; finally, the "controller" is handled by the framework itself, matching the current URL with the appropriate Python function, as declared in its "urls" configuration file.

Because of its particular approach to implement the "controller", combined with the fact that most of the framework functionality is done by models, templates and views, Django has also been referred as an MTV framework. According to this development pattern, M stands for "model", which again corresponds to the data access layer; T stands for "template", that corresponds to the presentation layer (how the contents are presented to the users); finally, V stands for "view", which relates to the application's business layer: it defines how the model is manipulated, and which templates are going to be used in order to display the data retrieved from the model, working as a bridge connecting those layers.

This separation of concerns, which distributes the work to be done between models, views, and controllers, provides a high level of modularity, which we used in our favour. In particular, we organized the different parts of the system in Django modules, which we could extend or modify according to our needs. We will discuss the modular organization of our system later in this document.

In summary, the use of Django in order to deliver the contents of the application has proved itself to be of great help, speeding up the development of the whole system.

Client-server architecture From a technical point of view, we decided to run the Django-based website on top of an AMP stack, composed by an Apache web server [1], equipped with the mod_wsgi module required to run Python scripts, in combination with a MySQL database server [5]. In parallel, we used a Jetty [3] based WebSocket server in order to improve the performance of real-time data transmission among the users of the system. This Java-based HTTP server and servlet container provides a ready to use WebSocket servlet that abstracts the low-level details of this technology, which as a result enabled us to build a robust implementation of a WebSocket server. The technical reasons to choose WebSockets over other existing alternatives for the implementation of real-time communication —like long polling or streaming, both based on regular HTTP connections—, are discussed in depth in section 4.1.2.

It's worth mentioning that we also explored the possibility of using a Python-based implementation of a WebSocket server, but its Java-based equivalent proved itself to be more robust, performing better during the process of connection and disconnection of the clients, which is critical for the correct broadcast of messages between the members of a group.

As it can be observed in figure 4.1, both servers (Apache and Jetty) have access to the

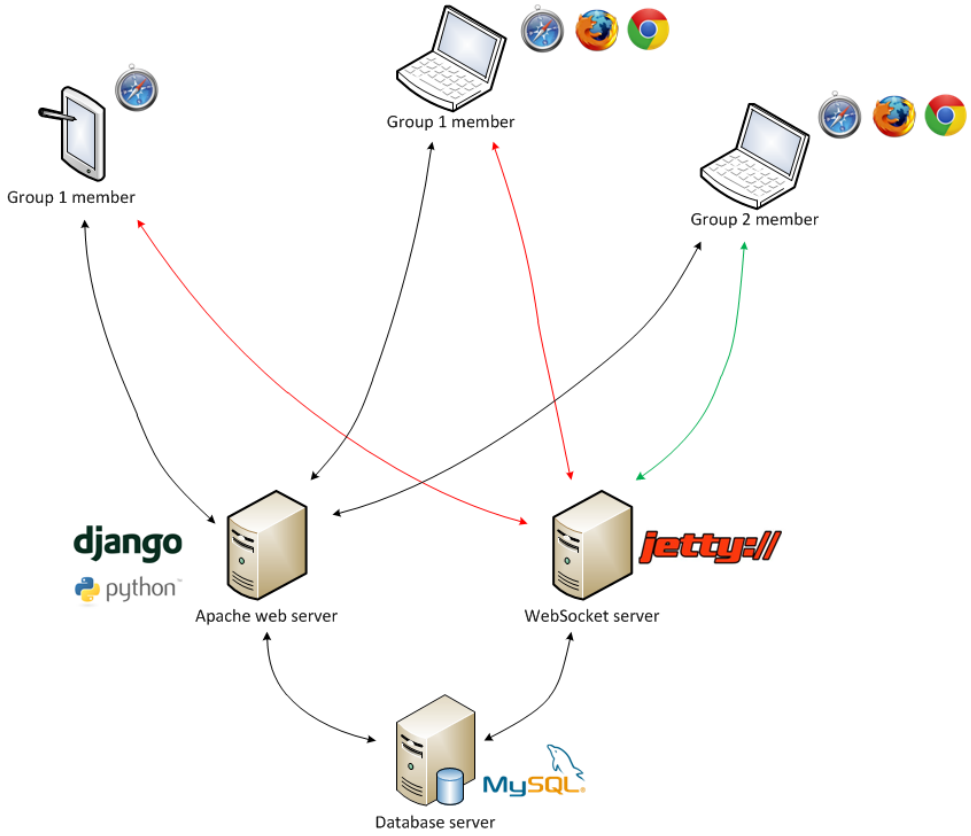


Figure 4.1: Client-server architecture.

same MySQL database, so all the basic information gathered is shared between the two.

At this point, we are going to take a close look at the modules that compose the different parts of the system, and the relations between them. In particular, we will discuss the components of the Django-based web application, and the packages that build the WebSocket server.

Web server components As we mentioned before, Django follows a model-view-template or MVT pattern, which implies that each of the modules includes a view.py file, which defines the business logic of the unit; a models.py file, that contains the models used to describe the underlying database tables for the corresponding entities, expressed as Python classes, which can be used to create, update, or delete records in the database, abstracting the associated SQL queries; and a set of HTML files enhanced with the use of a template language that provides basic logic statements, as iterators or conditional expressions. Additionally, the common urls.py file specifies the relationship between a URL and the view that has to be executed; for example, the URL "/activity/new/" will be handled by the "new_activity()" function, defined in the view.py file included in the activity module.

In our particular case, the system was organized in the following modules, as described in figure 4.2.

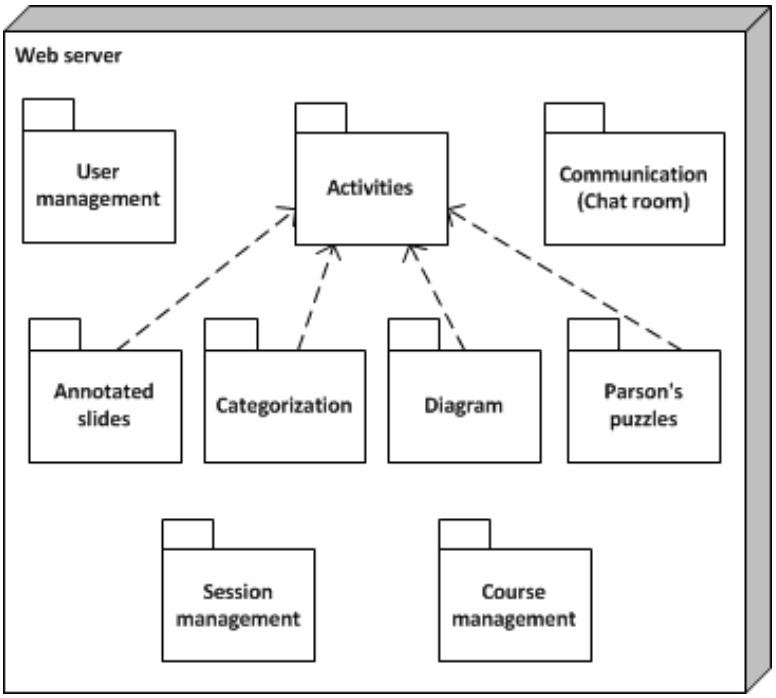


Figure 4.2: Web server modules.

The functionality of each of these modules is briefly explained below:

User management: this module defines two user profiles (instructor and student), which have different user privileges. It also provides the tools to prevent that users get access to parts of the site to which they don't have access rights.

Communication: implements the models and views that correspond to one of the computer-mediated communication channels used in the system (chat room).

Activities: this module implements the entities related with the management of activities, including the creation of particular instances of those activities, the definition of student groups, the sending of instructor tips or hints, the storage of the actions done by the students on shared artifacts, and the functionality needed to share those artifacts between different groups.

It also includes the description of a basic activity model, which was later extended in order to build the modules that implement the activities described in section 3.1 (annotated slides, categorization, diagrams and Parson's puzzles). Two of these extended packages also enhanced the functionality of this module by adding a submodule that provides image management (specifically, slides and diagram's icons).

Session management: allows the creation and edition of sets of activities launched in succession during a lecture or time slot, using the same student groups, as described

in section 3.3.3.

Course management: for practical reasons, we also considered necessary the creation of a module to manage all the information related to courses, which include the list of students enrolled in each course, and the instructors assigned to impart the corresponding lectures.

WebSocket server components As we mentioned earlier in this chapter, we opted to use a WebSocket server in order to provide real-time communication between the users of the system. Based on the libraries included in the latest Jetty distribution —a Java-based HTTP server and servlet container— the WebSocket server is composed of a series of packages that help to achieve its main functionality: forward the messages that codify the actions performed by a user to his or her group mates, in order to keep their shared work area up-to-date. The functionality of each of those modules is explained below.

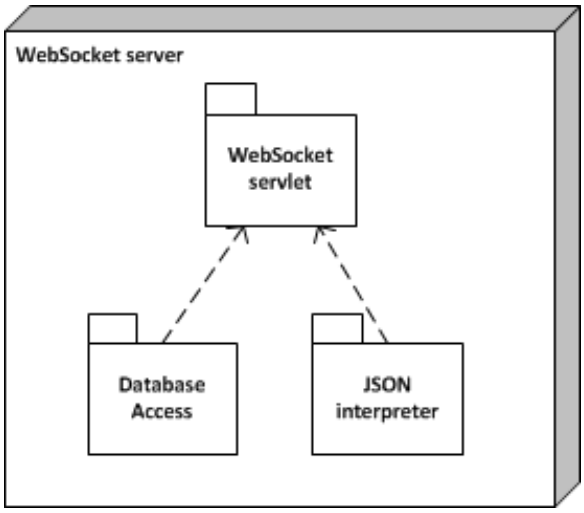


Figure 4.3: WebSocket server architecture.

The WebSocket server main class extends the WebSocketServlet class included in the libraries provided with the latest distribution of Jetty. This class manages the creation of a new HTTP connection that is asking to be upgraded into a WebSocket link. It also keeps a list that includes all active connections, identified by their corresponding group identifier, which is used to determine the recipients of the messages forwarded to the members of a particular group.

To deal with these WebSocket-enabled links, we created a class that implements the WebSocket.OnTextMessage interface, which is specifically in charge of dealing with the reception of text messages, like the JSON encoded messages we are exchanging (we explain the exact format of these messages later in this section). An instance of this class is called when all fragments of a text message have been received. Then, the message is parsed,

checking in the first place if the message encodes an activity-related action performed by a user, or if it is a connection management message (ping). The next step is checking if the corresponding activity it's currently active. If it is, the action is timestamped, saved in the database, and forwarded to the group members other that the sender of the message. If it is not, the message is ignored, sending a "inactive activity" message to the user.

To perform this procedure, the main class uses two auxiliary packages. The first one, labeled as "JSON interpreter" in figure 4.3, is in charge of parsing and translating the messages received from the clients, codified using JSON, into the corresponding Java classes in order to proceed to further manipulation. Specifically, we implemented a generic class, which is able to retrieve the messages' general data (group_id, exercise_id, method called), independently of the type of message received. This information is then used to discriminate which exact Java class corresponds to the function call encoded in the received message, leading to a more specific manipulation.

The second one, named "Database access" in figure 4.3, enables an efficient access to the database, implementing the singleton design pattern, which restricts the number of connections that the server establishes with the database. The use of this pattern makes database access more efficient because the creation and destruction of those connections is one of the most expensive operations needed to implement data retrieval from a database.

Message codification From the point of view of an ongoing activity, the WebSocket server is in charge of broadcasting the messages that contain the parameters that define the actions undertaken by the students on their shared artifact, forwarding these messages only to those users who are part of their group. This information is codified for transmission using the JSON standard (JavaScript Object Notation), which enables easy manipulation of the data at reception, independently of the programming language used.

The following listing shows the actual encoding of an action, in particular the movement of an element in a shared diagram.

Listing 4.1: JSON codification of diagram's move element action.

```
{'user_id':'6',
 'exercise_id':'1',
 'group_id':'2',
 'call':{'method':'moveElement',
         'parameters':{'elementId':'6_3',
                       'toX':228,
                       'toY':112,
                       'fromX':203,
```

```
                    'fromY':203,
                    'width':24,
                    'height':24,
                    'prevWidth':24,
                    'prevHeight':24
                }
        }
}
```

All messages used to transmit an action performed on a shared artifact carry the information that defines who carried out that manipulation (user_id), the group that person is assigned to (group_id), and the activity instance in which the group takes part (exercise_id). This data is used by the WebSocket server to check that a user who performed an action is actually part of the specified group, avoiding the possibility that users could manipulate a shared object other than theirs.

The fourth common parameter carried in every action-related message refers to the function call that defines the actual manipulation made on the shared artifact. In listing 4.1, we can observe that a call has two main parameters: the "method" (moveElement), that denotes the action performed on the work area, and the "parameters" (elementId, toX, toY, fromX, fromY, width, height, prevWidth, prevHeight) that define the particular values needed to perform the execution of that method. It's worth noting that in most cases, the parameter-related data includes the new values that define the modification done on the working area, and also the values previous to that manipulation, which are later used to empower exercise replays, both forward and backward.

When this kind of message is received by a client, the corresponding call is reconstructed using the method and parameters values included. In this example, the reconstructed call will look like this: "moveElement('6_3', 228, 112, 203, 203, 24, 24, 24, 24)". At this point, the expression is evaluated using Javascript's eval() function, which will actually perform the corresponding update to the shared artifact.

There are other types of messages, not related with the manipulation of shared artifacts, which are also transmitted through the WebSocket server. Specifically, ping packets and text chat messages are also sent using a WebSocket-based connection.

Ping messages are extremely simple, being their function maintaining the WebSocket connection between a client and a server open. A ping packet only includes group and activity related information (group_id, exercise_id), so the server can include in its reply information related to instructor hints sent to a group, or signal the finalization of the activity.

Listing 4.2: Ping message.

```
{'group_id':'47',
 'exercise_id':'47',
 'call':{'method':'ping'}
}
```

In the case of a chat room message, its codification and manipulation by the server, and also by clients at reception, is equivalent to the process followed with packets that include data related to changes performed on a shared artifact.

Listing 4.3: Chat room message.

```
{'user_id':'4',
 'exercise_id':'26',
 'group_id':'27',
 'call':{'method':'write',
         'parameters':{'message': 'student1: fine'}
        }
}
```

It's also worth mentioning that all messages forwarded by the WebSocket server are timestamped and stored in the database, allowing us to establish the order in which the actions were performed on the groups' shared artifacts. Once the system has settled the chronological order in which these actions were executed, this information can be later queried in order to provide activity replays and activity late-joining.

Unique naming of elements If we revisit the activities described in section 3.1, we observe that three of the four kinds of activities we devised (annotated slideshows, categorization activities and Parson's puzzles) provide users with a set of elements to be manipulated, but there's no option to create new items or delete existing ones. That is not the case of diagram-based activities, where members of a group can add and remove items from the shared artifact. This feature requires the use of unique identifiers for the nodes and connectors that compose a diagram, in order to unequivocally identify which elements are manipulated by a user, and transmit this information to the rest of the members of a group to update their view of their shared work area.

To achieve this uniqueness, each of the instances of the shared diagram uses the numeric identifier assigned to each user as a prefix to build the name of a new item, which is completed with a counter value that keeps track of the nodes and connectors added by that user. In case a user leaves an ongoing activity and later rejoins it, the

value of that counter is reset to the largest value that corresponds to the prefix assigned to the elements in the diagram created by that user.

Front-end implementation As we mentioned before, each of the activities we have implemented needs the creation of a new Django module that extends a generic Activity package, which acts as a skeleton for new types of exercises. This back-end module is then complemented by the code needed to provide its user interface, which changes depending on the activity. For example, in the case of Parson's puzzles, a jQuery-based script was developed for this task, based primarily on "drag-and-drop" actions. In contrast, annotated slideshows, categorization and diagram based activities were built using the Javascript port of the Processing programming language, which enables the renderization of complex graphics using the standard HTML <canvas> element.

The Processing programming language was originally conceived to teach fundamentals of computer programming within a visual context, being specifically designed to provide a clean interface, with a simplified syntax and graphics programming model that helps creating and modifying images. It also includes a handful of input events that need to managed (mouse pressed, mouse released, key pressed, etc.), and a base skeleton to implement animations. Additionally, as Processing builds on the Java programming language, it allows the use of object-oriented features, like class inheritance and polymorphism, which are really helpful during the development of large scale projects like this.

In contrast, Processing.js ports Processing functionality into web browsers using the canvas element for rendering. It basically translates Java code into Javascript code, and as a result, some of that object-oriented features present in the original implementation of Processing have no direct translation, and thus had to be discarded in our code. Other interesting point about Processing.js is that it only translates Processing code, which allowed us to mix in the same file Processing and Javascript code, being those Javascript snippets available at the end of the translation process.

4.1 Use of standards

As we mentioned in the introduction, one of the main targets we aimed at the start of this project was achieving a web-based platform-independent solution that provided an enticing graphical interface and, at the same time, allowed real-time communication at a reasonable cost. We do this in the belief that web standards like HTML5 will be predominant in a near future, providing an added benefit: this approach reduces developing costs, as it only needs the implementation of one version of the system, instead of specific versions depending on the target device. In the downside, there is a trade-off associated

with using a cross-platform solution versus a platform dependent implementation, as the use of the tools provided by hardware vendors would help extracting the best performance these devices have to offer.

In the following paragraphs, we discuss in depth the two main features of the still evolving HTML5 standard that allowed us to build an enticing graphical interface coupled with real-time communication: WebSockets and the <canvas> element.

4.1.1 Canvas element

HTML5 standard defines the <canvas> element as "a resolution-dependent bitmap canvas which can be used for rendering graphs, game graphics, or other visual images on the fly" [13]. Drawing in a canvas is achieved using specific Javascript functions designed for this purpose, which manipulate graphics at a pixel level.

In our particular case, we didn't use these functions directly but, as we mentioned before, used a JavaScript-based port of the Processing programming language that provided us with a simplified syntax and graphics programming model. This port uses the canvas element for rendering, allowing Processing to be executed natively in modern web browsers without the installation of any extra plugin.

Other important feature provided by this element is that contents drawn in a canvas can be exported as an image file, both in PNG or JPEG format. We used this feature to implement the monitoring capabilities that are part of the instructor side of the system: thanks to this, we captured the content displayed on the canvas element on which we based the shared artifacts used by the student groups, and sent it to the web server, which makes these images available for the instructor.

These capabilities made the <canvas> element defined by the HTML5 standard a sound base to build a relatively complex interface, runnable in any device that is equipped with a compatible web browser.

4.1.2 WebSockets

As we mentioned in chapter 2, the WebSocket protocol provides a full-duplex bidirectional communication between a client (a web browser) and a server (in our case, a Jetty based WebSocket server). This connection is established by executing a specific exchange of HTTP messages: first, the client sends a handshake request, which is replied to by the server with a handshake response. Once this process is finalized, the connection is upgraded, creating a full-fledged TCP connection between the two.

Having a bi-directional communication means not only pushing real-time messages from the server to the client, but also the reverse. Additionally, this connection offers other

advantages, like the reduction of unnecessary network traffic and latency in comparison to other HTTP-based solutions, which require a full round-trip to send messages from the client to the server, a constraint imposed by how browsers and proxies are implemented.

At the time of writing of this document, the WebSocket specification is still in draft status (the latest version is draft-ietf-hybi-thewebsocketprotocol-09) and its adoption is still quite immature and fragmented. The level of support for this feature varies depending on the browser: according to [19], Chrome (version 4.0 and later), Safari (version 5.0 and later) and Firefox (version 5.0 and later) are the only web browsers that give support to this particular feature, and the JavaScript libraries that implement this feature use the older WebSocket specification described in the draft-ietf-hybi-thewebsocketprotocol-00 document.

Advantages with respect to previous techniques

As we mentioned before, HTML5 WebSockets provides a full-duplex bidirectional communication between a client and a server. This feature enables an efficient implementation of up-to-date real-time transmission of information. Before this protocol was available, developers had come up with techniques to implement real-time web applications that in most cases involved polling periodically the server for new updates. In the following lines we describe some of these solutions, and we ponder the advantages that the WebSocket protocol provides in comparison.

Polling The first and most simple approach consists in polling the server for new information, sending HTTP requests in regular intervals, with the server replying to these requests as they are received. This was the first approach tried in order to implement real-time communication, but it only works efficiently if the client can synchronize its request interval to the availability of new data in the server, and that is only possible assuming that new data is available in the server with precise periodicity. Given the fact that real-time information production is often unpredictable, this approach causes the sending of lots of unnecessary requests. And this problem is even worse in situations where these updates are sparse.

Long-polling A variation of the previous approach involves the client sending a request to the server that provides the real-time data, which maintains this connection open for a relatively long period of time. If an update is available while the connection is still open, the data is sent to the client. If there is no data to send during the time the connection is kept open, the server will just send a response to terminate the request. Although the

number of unnecessary requests is greatly reduced in comparison to simple polling, there is no real gain if the volume of messages transmitted is relatively high.

Streaming In this third approach, a connection established between a client and a server is kept open indefinitely (or during a long period of time). When new up-to-date information is available in the server side, a message containing this data is sent to the client, but the connection is kept alive as the server never signals a request completion, being the server able to use this same connection to send later messages. The main drawback this approach suffers is related to the fact that the connection is still encapsulated in HTTP, which might cause that intervening firewalls and proxies opt to buffer the data transmitted, thereby adding latency to the communication.

There are two other problems that affect the previous approaches to real-time up-to-date sending of information. Firstly, if the goal is establishing a full-duplex communication channel, like the one provided by the WebSocket protocol, we would have to establish two half-duplex HTTP-based connections (upstream and downstream) that have to be coordinated and maintained. This coordination adds complexity to the implementation, and requires the use of more computing resources in both sides of the communication channel. Secondly, these methods involve the sending of HTTP request and response headers, which are irrelevant for the endpoint applications, but need to be transmitted along with the actual data. This introduces added latency to communication, specially if the amount of data exchanged is small in comparison to the size of those headers.

WebSocket shortcomings

Although it is clear that using WebSockets it's appropriate to provide real-time communication in the system described in this document, there are a number of shortcomings that we had to address to implement robust communication using this technology.

One of the most important problems of using a stateful connection using WebSockets comes from the fact that any connection established might drop for different reasons. For example, the server or an intermediary might choose to close the connection due to an idle timeout. This might happen for example because there was no need for communication on the client side. To address this problem, we devised two complementary measures: ping-pong messages and automatic reconnection.

In the first place, we used a ping-pong scheme, with the client sending periodically a short message to the server in order to keep the connection alive. Unfortunately, we don't have any indication about the frequency at which pings would need to be sent in order to maintain the connection alive at the server or at the intermediaries, so we chose an arbitrary time interval of 20 seconds. Once we had this infrastructure in place, we

actually used that ping and pong messages to carry relevant information: ping messages will contain the data associated to the user who is using the connection and the exercise in which he or she is working on. In response to the ping message, the server will reply with a pong message containing the latest tip send to the users carrying out that exercise, if there is any available.

It's worth mentioning that the JavaScript API included in current web browsers does not have a specific function to send ping messages, although they are included in the WebSocket specification it implements. Meanwhile, the latest specification of the WebSocket protocol (draft-ietf-hybi-thewebsocketprotocol-10) provides ping and pong control frames, so it would be no surprise if in the near future the JavaScript API that implements this version of the protocol includes a pair of functions to exchange this kind of message.

Complementing the previous measure, we chose to implement an incremental back-off reconnection policy in order to reopen the link between the client and the server, in case the client receives an onClose event caused by a broken communication channel. To prevent falling in an endless loop of reconnection attempts if the server or the network are temporarily down, we have used a reconnection policy that implements binary exponential back-off, which after an unsuccessful reconnection attempt will multiplicatively decrease the rate of this process until a predefined limit is reached, which in our case performs one connection attempt per minute.

Finally, we had to take into consideration what to do with the messages produced by the users while the connection with the server is down. Having in mind that the actions carried out on the artifact that the group members are sharing depend directly on the current state of the workspace, we considered that there was no need to keep a queue or buffer to store the actions performed while the connection is broken, as the result of the operations carried out during this off time will not be consistent with the actual state of the shared artifact. Therefore, we thought that it would be better for a user who suffered from a broken connection to simply rejoin the group when the communication is reestablished.

WebSocket browser support

By the time of the writing of this document, two of the most important players in the browser community had announced the implementation of a more advanced specification of the WebSocket protocol, which fixes a security issue [14] discovered in an earlier version of the protocol (draft-ietf-hybi-thewebsocketprotocol-00).

Specifically, this vulnerability occurs under a particular network configuration that includes the use of transparent (or intercepting) proxies. The main issue comes from

the fact that in many occasions, these proxies don't understand the semantics of the data they forward. Taking this into account, transparent proxies might not understand the Upgrade-based handshake exchange of messages needed to establish the WebSocket connection in earlier versions of the protocol, and therefore they don't know that the following bytes transmitted don't correspond to an HTTP request. As a result, proxies treat this data as HTTP requests, allowing an attacker to circumvent firewalls or poison the proxy's HTTP cache, depending on how the proxy was configured.

To prevent this kind of attack, web browser developers have taken some preventive measures. In particular, the Mozilla foundation, responsible of the development of Gecko-based browsers like Firefox, announced [4] that this family of browsers will implement draft-ietf-hybi-thewebsocketprotocol-07 version of the WebSocket protocol in Gecko v6.0, and draft-ietf-hybi-thewebsocketprotocol-10 version in Gecko v7.0. Earlier versions of this browser included an implementation based on the draft-ietf-hybi-thewebsocketprotocol-00 specification of the protocol, but this feature was disabled by default because of the vulnerability we previously mentioned.

Meanwhile, the development team of the Chromium project had announced [30] that Chrome 14 will implement draft-ietf-hybi-thewebsocketprotocol-10 version of the WebSocket protocol. Prior to this release, Chrome browser included an implementation of the WebSocket protocol based on its draft-ietf-hybi-thewebsocketprotocol-00 specification.

Unfortunately, this revamped implementations of the WebSocket protocol are not backwards compatible, which means that, while the client-side implementation will remain functional, the server-side implementation will have to be revised to maintain the current functionality of the system. We expect that an upgraded version of the WebSocket libraries included by the Jetty web server will be enough to keep the implemented system working correctly.

Before the end of this section, we would like to mention that we also considered the use of other new features included in the HTML5 specification, like web storage, using this feature to build a buffer that could store the actions undertaken while the WebSocket connection is down. The distributed nature of the system and the need for real-time updates deterred us from implementing this added capability.

4.2 User interface considerations

An important issue in groupware and collaborative systems is **awareness**, which involves knowing or having a basic idea of how other people are interacting with the system at any given time, and also knowing what are they doing or have been doing [26]. To give this sense of awareness to the users who are manipulating a shared artifact, we have used

several visual cues.

To begin with, both live and replayed activities have a list of the users that can potentially modify the shared artifact: the students that form the group, and the instructor who launched the activity, who can also join the group and manipulate the shared work area at any given time. In this list, each member of the group is assigned a color, which will be used to denote the user who performed each of the actions listed in the action log. This register will be updated every time a user performs an action and its broadcast is received by the rest of the members of his group.

Additionally, in categorization activities, diagram-based exercises and Parson's puzzles we try to maintain user awareness using specific visual cues.

Categorization activity When the movement of one of the terms in the shared artifact is performed by one of the students in the group, the rest of them will see how the term moves accordingly in its screen. To help users to grasp who exactly did that action and how it develops, the moving element will temporarily change its colour to the one assigned to the user who performed that action. Also, the frame around the canvas element will briefly flash with that colour.

Diagrams For this kind of activity, the number of possible manipulations that can be performed on the shared artifact increases, but we have considered that movement and resizing of shapes are the most important ones. As a result, if the action received corresponds to the movement or resizing of an element in the diagram, this action will be progressively performed, and the handles of the updated element will be colored with the colour that corresponds to the student who executed the action.

Additionally, every time an action is performed by a student, his or her group mates will see how the frame around the canvas element briefly flashes with the colour associated to that student.

Parson's Puzzles In this case, users will see how the background of the layout where the pieces of the puzzle rest briefly flashes with the colour associated to the student that performed the corresponding action.

4.3 Usage of tablet PCs

The rapid popularization of a device like Apple's iPad during the last few months (it was launched in April 2010 and has sold since then nearly 30 million units), coupled with the apparition of a large number of new players in the tablet PC market, has increased

the penetration rate of this kind of device. At the same time, its ease of use has also improved the notion users have of tablet PCs, which are now viewed by some authors [20] as a perfect substitute of notebooks in classrooms. As any new technology that it's introduced in a new environment, the use of these devices in an educational environment has several advantages and drawbacks:

Advantages

- **Screen size:** Although smaller than the screen we can use in a regular laptop, the screen size of Apple's iPad and other similar tablet PCs is more than enough to provide good visualization of contents.
- **Battery life:** Specially in the case of Apple's iPad, the duration of its batteries is long enough to last for a whole school day (6-8 hours).
- **Size and weight:** given its relatively small dimensions (243 mm x 190 mm x 13 mm) and weight (ranging between 600g and 730g, depending on the model), these devices are quite manageable, especially when we considered the size and weight of the equivalent books and notebooks they replace. Sadly, finding suitable textbooks for these devices could be problematic, as it seems that editors are not interested in keeping up with technology.
- **Predictable user experience:** thanks to its polished user interface, new users can start using this kind of device right away, as applications behave in a predictable way. In addition, the time needed to find and run an application is relatively short, which has a result a reduction in the time needed to set up a working session.
- **Communication capabilities and multimedia support:** As this kind of device can easily connect to the Internet using a Wi-Fi network, the students can browse the web, having access to multimedia contents, e-mail or download e-books to read.
- **No need for dedicated labs:** because of its wireless communication capabilities, Apple's iPad and other similar devices don't need the use of dedicated labs, as there is no need for complex wiring or specific computer desks. In return, it's needed a strong and reliable connection to the web, which relies on a stable incoming bandwidth available, and a tried and tested wireless infrastructure.

Drawbacks

- **Sharing devices:** These devices work better in a one student per device ratio, as users usually leave a data "footprint" in them when saving their work. This fact makes sharing an iPad between two or more students impractical.
- **Touch interface:** Although it allows a natural interaction with the device, the touch interface has an onscreen keyboard that makes typing text somewhat impractical, specially when editing long documents.
- **Printing:** At least with the first version of the operating system included in this device, there's no possibility to print documents.
- **Adobe's Flash and Java are not supported:** A corporate decision of Apple, Adobe's Flash and Java applications are not supported by its products. This fact leaves out the possibility of using contents based on these technologies.

When comparing this new generation of tablet PCs with previous implementations of this kind of device, mostly based on the use of digital ink (which were used extensively in this field, as we mentioned in chapter 2), the main difference is that they are controlled by a multitouch display, which enables users to interact with the device directly with their hands, in an even more natural way.

Figure 4.4: Apple's iPad.

As a result of using this multitouch display, which is designed to be controlled by bare fingers, we had to look for activities that don't require very accurate aim. This problem could also be minimized using a stylus specifically designed to work in combination with this type of touchscreen, but it requires an extra expenditure.

Additionally, this iteration of the tablet PC device uses by default a virtual onscreen keyboard instead of a physical keyboard, which makes text input somewhat slower and

more cumbersome. This fact encouraged us to look for activities in which writing does not take an important role.

Given the size constraints imposed by the smaller screen size available in these devices (for example, Apple's iPad has a 19.1x14.8 cm liquid crystal display, with a screen resolution of 1024x768 pixels), we considered necessary to adjust the website interface dynamically depending on the orientation of the display. Thanks to some new features included in the CSS3 standard, the style sheet used can be adjusted automatically depending on the width and height of the screen, or the orientation of the device, thus improving the user experience.

Apple also provides a custom software development kit specially designed to build native applications, which helps extract the best performance this device has to offer. Although we could have opted to use this approach to implement the system, we discarded tying our system to a particular device in favor of creating a cross-platform solution that provides better portability.

It's also noteworthy that, in the particular case of Apple's iPad, the device is equipped with a custom version of the Webkit-based Safari web browser that includes some notable differences with respect to its laptop equivalent. For example, small informative texts displayed when using a pointer device, like the *title* attribute in a *img* HTML tag, are no longer available. Also, JavaScript's mouse related events are also replaced by touch related events. Consequently, we had to transform touch events into the equivalent mouse events, so that the corresponding JavaScript/Processing handlers could be fired. Furthermore, jQuery custom events related to drag and drop gestures had to be emulated in order to keep the basic functionality of the user interface, specially in the case of the Parson's puzzles shared artifact.

Although Apple products are widely regarded to have an intuitive user interface, it's worth mentioning a counter-intuitive feature of its browser that manifested itself when a div contains scrollable content: in a regular web browser, we would be indicated with a scroll bar that there are hidden contents that we can display with a scroll gesture. In the iPad's Safari version, this scroll bar does not appear, and only with a two-finger drag motion we are be able to scroll the content, which is not easy to discover by a casual user.

Chapter 5

Evaluation

In order to evaluate the system implemented as a result of this project, we organized several evaluation session were different kinds of users were asked to try out the application, and rate different aspects of it. These sessions were conducted with the approval of the Research Ethics Committee of the School of Computer Science and Statistics.

The process of evaluation was based on a questionnaire that was handed to the participants, who went through a complete work session similar to the one described in section 3.4 of this document. In particular, we used paper-based questionnaires consisting of two main parts. The first part is composed by a set of questions presented using a five point rating scale of 1=strongly disagree, 2=disagree, 3=neither agree or disagree, 4=agree, 5=strongly agree. These questions were also organized in two subsections, in order to establish the participants' opinion depending on the role they would take while using the collaboration system: either as instructor or as students.

The second part of the questionnaire contained a set of open answer questions, so the participants could explain themselves in detail, providing more meaningful feedback about certain characteristics of the software evaluated.

5.1 Student evaluation

An initial set of evaluation sessions was performed with the collaboration of fellow members of the M.Sc. in Mobile and Ubiquitous Computing course, and several undergraduate students.

In these sessions, the volunteers were organized in groups of two members, having each participant access to a laptop. Then, they were asked to pretend that they were students about to receive a lecture focused on the basics of a particular data structure —binary trees—, while the researcher took the role of the instructor. This "lecture" would follow

the development of the work session described in section 3.4 of this document, which includes a slideshow presenting the basic concepts that describe this kind of data structure, and several related activities where participants could apply their newly acquired knowledge.

As we mentioned before, after the conclusion of the practice activities, participants were presented with a questionnaire where they could express their attitudes toward using the collaboration system.

The full set of questions included in the first part of the questionnaire, corresponding with likert-style items, is given in table 5.1, along with the average responses calculated.

Survey item	Response average
Q1. The overall experience of using the system was satisfactory	4.00
Q2. The system provides adequate tools to present information	4.50
Q3. The system provides adequate tools in order to organize group activities	5.00
Q4. The system helps monitor group activities	5.00
Q5. The system improves communication with the students	4.00
Q6. The system helps manage activities during a session	4.50
Q7. The system helps grade the students' work	5.00
Q8. The system can be used for distance learning	5.00
Q9. The system improves collaboration among group members	4.25
Q10. The system improves communication among group members	3.75
Q11. The system improves communication with the instructor	3.50
Q12. The system provides adequate tools to review the work of other groups	4.50
Q13. The system provides adequate tools to review previous work	4.75
Q14. The system can be used for distance learning	4.25
Q15. The use of the software was enjoyable	3.75

Table 5.1: Mean survey responses: user evaluation, 4 respondents.

If we look at the aggregated results, we can observe that these participants were satisfied with their overall experience using the system (Q1, avg=4.00).

Focusing on the questions related with the features the application provides to ease the work of instructors while organizing group-based activities, we can observe that they strongly agree that the system provides adequate tools to present information to students (Q2, avg=4.50), organize group activities (Q3, avg=5.00), and monitor the students' progress while the exercises develop (Q4, avg=5.00). Participants also agree that the collaboration system improves communication between an instructor and the students attending a lecture (Q5, avg=4.00), and strongly agree that it helps the instructor to manage activities during a working session (Q6, avg=4.50) and rate the students' work after it has been completed (Q7, avg=5.00). With respect to the last question of this first

block, we can observe that the participants strongly agree that the system could be used to impart distance learning courses (Q8, avg=5.00).

The second block of likert-scale items includes a set of questions related with the student's point of view of the application. In this case, participants tend to agree that the system improves collaboration and communication between the members of a group (Q9-Q10, avg=3.75). The level of agreement is a bit lower with respect to the improvement the application provides in communication between the student and the instructor (Q11, avg=3.50). They also agree that the system provides adequate tools to review both students' previous work and the work of other groups (Q13-Q14, avg=4.00) and that the application could be used to receive a distance learning course (Q14, avg=4.00). Finally, participants tend to agree that the use of the software during the evaluation session was enjoyable (Q15, avg=3.50).

Finally, we have summarized below the participants' answers to the second part of the questionnaire, which presented them several open-ended questions.

Most useful features

From the point of view of regular users, the most useful feature the system provides is the possibility to review their work, in a step-by-step basis once is completed, and the ability to replay lecture slides with the annotations made by the instructor during a slideshow.

Some users also indicated the improvement the system provides in terms of collaboration among group members, and the use of complementary communication channels, like the chat room available for students while they are manipulating a shared object, or the sending of "tips" by the instructor during the activities.

Drawbacks of using this kind of software in a classroom environment

Among the possible drawbacks that the users detected in the software when used in a classroom environment, the users pointed out the fact that one student could dominate the shared artifact, doing all the work required to complete a particular activity, thus preventing others to participate. Another problem related with the students' behaviour can arise if one of the group members starts sabotaging the shared artifact, blocking the group's work. Both issues were taken into consideration during the designing stages of the system, and we believe that the fact that instructors and students alike can spot "who did what" during an activity will prevent this kind of situation.

Also the chat room was considered by several users as a possible cause of distraction, leading to messy conversations or even some sort of "spam".

Other drawback pointed out are related with the lack of pointers or textual instructions indicating the function of the system's controls.

Subject areas which are more appropriate to be taught using this kind of system

The users mentioned a wide range of possible knowledge fields where this kind of system could be used, with several users pointing out the fact that the activities already included would suit Math instruction, or any other algorithm related subject.

Also, thanks to the inclusion of annotated slides, the system was considered by some users to be suitable for any other kind of subject, enhancing the presentation of contents.

Suggestions received

Most of the suggestions made by the users during the evaluation sessions are related with user interface issues. Some participants pointed out that the question or task of the exercise was not visible by default at the start, a problem which has already been fixed.

Another suggestion that some users provided is related to the need for guidance during the first use of the system. A fix for this situation can consist of a "first time" use tutorial that explains the function of the controls.

It was also mentioned the possibility to make the system more colourful for younger students, and improve the layout of the Parson's puzzles activities, highlighting keywords and providing text indentation.

5.2 Expert evaluation

In collaboration with Tim Savage, we were also able to organize an evaluation session in which members of the Centre for Research in I.T. in Education, and other experts in this particular field of knowledge tested the application built. The aim of this session was gathering the point of view of professionals in education, trying to spot any area of the system that could be improved that had not been previously detected.

In this case, the four participants were organized in one group, providing an iPad for every two people. Again, participants were asked to pretend that they were students about to receive a lecture focused on the basics of binary search trees, while the researcher took the role of the instructor. The development of the working session loosely followed the one described in section 3.4 of this document, including once again a slideshow and other activities (diagrams, Parson's puzzles).

The results of the likert-scale based part of the questionnaire for this group of participants is given in table 5.2, along with the response average computed.

Survey item	Response average
Q1. The overall experience of using the system was satisfactory	4.00
Q2. The system provides adequate tools to present information	3.75
Q3. The system provides adequate tools in order to organize group activities	3.75
Q4. The system helps monitor group activities	4.00
Q5. The system improves communication with the students	3.00
Q6. The system helps manage activities during a session	4.00
Q7. The system helps grade the students' work	4.00
Q8. The system can be used for distance learning	4.25
Q9. The system improves collaboration among group members	3.75
Q10. The system improves communication among group members	3.75
Q11. The system improves communication with the instructor	3.50
Q12. The system provides adequate tools to review the work of other groups	4.00
Q13. The system provides adequate tools to review previous work	4.00
Q14. The system can be used for distance learning	4.00
Q15. The use of the software was enjoyable	3.50

Table 5.2: Mean survey responses: expert evaluation, 4 respondents

The results show that the users that took part in this particular evaluation session were satisfied with their overall experience using the system (Q1, avg=4.00).

If we focus on the questions related with the features the system provides for instructors to organize group-based activities, we can observe that they relatively agree that the system provides adequate tools to present information and organize those activities (Q2-Q3, avg=3.75). Participants agree that the application would help an instructor to monitor group activities (Q4, avg=4.00), but don't agree or disagree that it improves communication between an instructor and the students (Q5, avg=3.00). They also agree that it helps the instructor managing activities during a working session and rating the students' work after it's completed (Q6-Q7, avg=4.00). To end with this first block of questions, we can see that the participants strongly agree that the system could be used to impart distance learning courses (Q8, avg=4.25).

The second part of the likert-style questionnaire includes a series of questions related with the student's point of view of the application. In this case, participants tend to agree that the system improves collaboration and communication among members of a group (Q9-Q10, avg=3.75). The level of agreement is a bit lower with respect to the improvement the application provides in communication between the student and the instructor (Q11, avg=3.50). They also agree that the system provides adequate tools to review both

students' previous work and the work of other groups (Q13-Q14, avg=4.00) and that the application could be used to receive a distance learning course (Q14, avg=4.00). Finally, participants tend to agree that the use of the software during the evaluation session was enjoyable (Q15, avg=3.50).

Below we include a summary of the responses to the open-ended questions formulated in the second part of the questionnaire.

Most useful features

During this evaluation session, the participants were almost unanimous about the feature of the system they considered most useful: the fact that the system tracks the manipulations made on shared objects.

Other feature that was also mentioned by two of the participants as important is how easy is for instructors to "jump" from group to group, assisting student groups as required, either sending a textual hint or interacting with the group's shared artifact.

Finally, the possibility to playback the actions performed during an activity was seen as a very useful feature that the system provides to students and instructors alike.

Drawbacks of using this kind of software in a classroom environment

The main drawback pointed out is related to the level of usability of the system's user interface. According to the participants in this evaluation session, it's really important, specially in instruction-related settings, that users are able to use the system seamlessly and with a high degree of satisfaction, or it won't be used in the long run.

Also, it was mentioned that it could be useful the implementation of some kind of turn-taking protocol, in order to organize group work, thus allowing all students in a group to collaborate. Although implementing this measure would mean that all students would have to collaborate during an activity, we consider that this restriction is not really necessary. As one of the goals of the system is developing communication between group members, we think that in most cases groups will discuss and organize themselves in order to complete the task or tasks they are presented in a cooperative way.

It was also mentioned that it could be difficult to provide all students with iPads or laptops in order to access the system, although this is a problem that escapes to our control.

Subject areas which are more appropriate to be taught using this kind of system

Again, the participants considered that a system like the one presented in this document could be used in a wide range of different knowledge fields, but specially in science-related subjects. It was also pointed out that the use of a more generic drawing tool could make the system even more versatile.

Suggestions received

Most suggestions made by this group of participants were related to the level of usability of the system, which should be excellent if it's going to be used in classrooms successfully: in this environment, if users had problems during their first contacts with the system, it's probable that they will discard its use altogether.

The participants also considered that the system is in a phase in which it needs to be evaluated by educators, so the system is optimized to match their actual needs, adapting the activities currently included to other fields or education, or developing new ones that could take advantage of the synchronization and communication features already provided by the software.

Comparing the results obtained in both evaluation sessions, we can observe that the average scores for the questions related with the instructor side of the application are higher in the evaluation session performed with students than those obtained in the expert evaluation session. This result can be explained by the fact that experts participants already have experience as instructors. On the other hand, it is remarkable that the answers for questions relating the student side of the system are very similar in both cases, and as a result, probably more significant.

As for the open-ended questions, we can observe that both groups of respondents found that the ability the system provides to review their work step by step is its most useful feature. Expert participants further emphasized the fact that they could know "who did what" during revision, which again shows us their experience as instructors. It is also remarkable to see how both groups of users think that the system is versatile enough to be applied in different fields of knowledge other than computer science.

Also, both groups of participants consider that there is still room for improvement with respect to the user interface, but while the students made suggestions for specific parts of it, expert users stressed the need to improve the interface as a whole in order to provide the best user experience possible.

Finally, it's worth noting that, specially in the case of the evaluation sessions conducted

with students, we must be cautious about the positive results obtained: they must be regarded as tentative because of the small number of participants. Therefore, we consider that we could extract more significant conclusions if this same evaluation protocol is implemented after using the software in an actual course, with a larger number of students taking part in an evaluation session at its conclusion.

Chapter 6

Conclusion and future work

6.1 Conclusion

From a technical point of view, the result of the project has been quite satisfying. The system we have implemented has proved to be useful during the evaluation sessions, being robust enough to provide a good user experience while keeping shared objects always synchronized thanks to the communication mechanisms used. We believe that this project shows that web standards provide a good base to implement collaboration tools, both at the back-end of the application, providing robust real-time communication, and at its front-end, enabling the construction of an enticing user interface.

We also think that implementing the system using web standards, instead of tying its implementation to a particular platform, was the right call. We believe that web standards like HTML5 will be predominant in a near future, providing an added benefit: this approach reduces developing costs, as it only needs the implementation of one version of the system, instead of specific versions depending on the target device. We may be sacrificing the ability to provide better graphics, but this is not that important in an application like the one created in this project.

If we focus on the HTML5 standard, we can now conclude that it provides different levels of support to implement the features we wanted to supply to users in a classroom environment.

On the one hand, the HTML <canvas> element has a stable and mature implementation, and we could count on drawing libraries (like Processing.js) that abstract the details of low-level Javascript drawing functions. On the other hand, we now know that the WebSocket protocol provides a good basis to implement a real-time up-to-date communication system that relies on web-based technologies. On the downside, it should be noted that the WebSocket protocol doesn't have at the moment the same level of maturity as

other components of HTML5, as its specification is still evolving. This fact is even more pronounced if we take into consideration that future implementations of the protocol —which are supposed to be included in forthcoming releases of widely-used browsers like Firefox or Chrome— are not backwards compatible, forcing current WebSocket server implementations to be updated so they can keep working.

With respect to the use of tablet-styled devices like Apple's iPad, I think that they can be really useful in a classroom environment, but only if the right contents and applications are used, in an attempt to change the way instruction is performed. The potential for change is there, as this type of device can be more than a fancy way to give students access to the Internet. A coordinated effort that includes content creators, application developers and educators is needed to make this change possible. In that sense, we consider that this project has been an effort that points in the right direction, having established a solid technical foundation to create a learning environment based on collaboration. It's now the time to get in touch with educators and content creators, and find activities and artifacts to improve the functionality currently provided in the system, and extend the scope of application of this software to a broader range of knowledge areas.

Hopefully, the system's modular nature will allow us to complement and enhance its functionality with low implementational costs: the addition of new activities only requires the construction of new modules, which are completely independent of existing ones. Also, improving and modifying the ones already included won't affect the basic functionality of the system. Moreover, the fact of having two separate servers differentiated by their functionality (one delivering the application, the other providing real-time communication) means that changes in one of the servers do not affect the operation of the other, and we will be able to completely change a server implementation without having to make drastic modifications to the one that remains unchanged.

Additionally, during the conduct of the evaluation sessions we could observe that the selection of activities that we chose to illustrate the use of a collaborative system in a learning environment was also correct: participants started discussing and negotiating how to tackle the tasks to perform, and instructors could guide them easily, monitoring their progress in real-time and providing hints as necessary.

Finally, from a personal point of view, I consider that this project has been quite enriching, giving me the opportunity to work with cutting-edge technology —like the still evolving WebSocket protocol—, learn how to get the most of different programming languages —like Java, Python, Processing or Javascript— while applying the right tool to tackle the problems I encountered, and organize myself to face a project of a considerable dimension. Furthermore, this project gave me the chance to approach a field of knowledge like pedagogy which I didn't previously know about, but that turned out to be

quite interesting. This project has confirmed me that, like Edsger Dijkstra once stated, "Computer science is no more about computers than astronomy is about telescopes."

6.2 Future work

The system built as a result of this project has proved to be useful providing higher levels of collaboration and communication among students working in a shared artifact while using tablet-styled devices. Unfortunately, time and resource constraints have meant that several possible improvements and new features were not included in the final version of this groupware system. We review some of these features below in no particular order.

Automatic feedback In order to ease the workload of instructors, an interesting feature to include in the future would consist of providing automatic feedback to students as they work with their shared artifact.

For example, when a group is manipulating a Parson's puzzle, a group asking for automatic feedback would know if they have achieved a correct subroutine. If this was not the case, the system would indicate which lines of code are placed incorrectly.

In diagram activities, a possible way to provide feedback or check the validity of the solution achieved by a group could consist on comparing their solution against a base solution provided by the instructor, trying to match the graph structure of both solutions.

Finally, in a categorization activity, the system could check which terms had been grouped around a specific concept, indicating if they have been correctly classified.

Activity deadlines & student submission Specially if the system is going to be used for distance learning, it could be useful to provide instructors with the possibility to assign a deadline to active exercises, which would remain in this state until that date.

Complementing this feature, it could also be interesting to provide students with the ability to notify the instructor if they consider they have achieved a valid solution. As we are dealing with groups, not individuals, the students in a group will need a voting system so they can reach a consensus on the state of the solution obtained.

Improved activities In addition to the provision of automatic feedback, there are particular improvements that could be done to some of the activity types previously presented.

Annotated slideshows Currently, the slideshows included in the system only allow instructors to annotate the slides. We could complement this activity by enabling students

to add their own private annotations to the contents they are provided.

These private annotations could be done in two alternative ways: firstly, they could be performed directly on the slides, in the same way the instructor's strokes are currently stored in the system. Alternatively, students could also complement the contents presented in the slides with their own notes, stored in textual form.

Also, the implementation of undo and redo actions would be a valuable addition to the current functionality that the system provides.

Parson's puzzles Following the example of [15], we consider that, depending on the programming language used to codify the lines of code provided, it could be interesting to control the indentation of the pieces a group puts together as a solution. This is particularly important if the programming language used is Python, as it uses white spaces to delimit program blocks.

Additionally, these puzzles could be further improved by highlighting the corresponding programming language keywords, providing this useful information to students.

Diagrams An improvement to the current implementation of diagram-based activities would consist in giving instructors the possibility to provide students with a starting diagram, which they could manipulate to complete a particular task. For example, an instructor could provide an initial binary search tree, which the students should manipulate to add or delete nodes to it.

New activities For practical reasons, the activities we devised as a proof of concept are closely related with computer science. We consider that it could be helpful to get in touch with educators in other disciplines, who could suggest new types of activities to be integrated in the system, and find new uses for the ones already included while applied in other fields of education.

Usage statistics In an effort to improve the information available for instructors when they evaluate the work done by each of the members of a group, we considered the possibility of generating use statistics. This data would indicate, for example, the number or percentage of actions performed by each student on the group's shared artifact, or other relevant factors that could be statistically measured.

WebSocket server update & fallback implementation As we mentioned in chapter 4, the latest specification of the WebSocket protocol is not backwards compatible. This won't require any changes on the client side, but it will make the current WebSocket

server implementation obsolete. Hopefully, thanks to the abstraction layer provided by the Jetty web server, we think that the Java classes that shape our implementation won't need any changes, and only updating the Jetty-related libraries will be necessary.

Also, if we wanted to enable standard compliant communication to users accessing the system with a web browser that doesn't support the WebSocket library, we have considered the possibility of providing fallbacks based on Comet technologies, using techniques like Ajax long polling or HTTP streaming.

Sound notifications We have also considered the inclusion of sound-based notifications that complement the visual cues used to notify users when the shared artifact has been updated. Thanks to the audio tag included in HTML5, this improvement could be performed in standard-compliant fashion.

Inclusion of other communication channels Specially if we consider the possibility of using this software in distance learning courses, the use of a textual chat might not be enough to provide a fluid communication between users. For this reason, we have considered the possibility of using our collaboration system in combination with VoIP applications that provide voice and video chat, like Google Talk or Skype, in order to improve communication among participants.

Alternatively, future implementations of the HTML5 standard, combined with WebSocket-based data transmission, could provide the tools to build an embedded voice and video chat in our system.

Improved user awareness One particular feature that was initially included, but was later discarded, allowed instructors to know the moment a student interacted with the system for the last time. This data, in combination with the timestamps assigned to the messages received by the WebSocket server, could be used to improve the notion a user might have about which other users are actually using the system at any given time, or if any of them has stopped interacting with it altogether.

On the one hand, it will indicate an instructor reviewing a live activity the time that has elapsed since the members of a group manipulated their shared object for the last time. To do this in a graphical manner, we could use a colour-based frame that surrounds the instructor's vision of the groups' shared artifacts (figure B.3).

On the other hand, students would be provided with up-to-date user status information related to their fellow group members and the instructor that is supervising the activity they are performing, indicating if they are also present and actively manipulating the shared artifact, or if instead they have stopped interacting with it. This information

could be displayed using a colour-based activity scale associated to each member of the group. Most users are already familiar with this kind of display because it is widely used by instant messaging applications, like Skype or Google Talk, to denote users' status.

User interface evaluation In order to provide the best user experience possible and improving the usability of the system, it would be interesting to perform some user interface evaluation sessions, including the measurement of user's performance while performing a set of common tasks, like browsing their activity archive or prepare a work session. Thanks to this evaluation process, we can gather valuable feedback that we could later use to build a top quality UI, thus improving the user perception of the system.

Improved activity management Aiming to facilitate the instructor's work, we suggest the addition of further management options, like the removal of outdated activities from the system. Currently, this action can only be performed by system administrators.

Appendix A

Abbreviations

Short Term	Expanded Term
API	Application programming interface
CSS3	Cascading Style Sheets, version 3
DOM	Document Object Model
HTML5	HyperText Markup Language, version 5
HTTP	Hypertext Transfer Protocol
K-12	Designation for the sum of primary and secondary education
MVC	Model-View-Controller architectural pattern
MVT	Model-View-Template architectural pattern
TCP	Transmission Control Protocol
UI	User interface
URL	Uniform Resource Locator
XHTML1	eXtensible HyperText Markup Language, version 1

Appendix B

Main user controls

The following pages give a short explanation on the different controls that the system's user interface provides to instructors and students, in order to get access to the features this software offers. Only the most relevant functionalities are included in this appendix.

- Instructor home.
- Student home.
- Instructor live activity review.
- Instructor inactive activity review.
- Activity: annotated slideshow.
- Activity: categorization.
- Activity: Parson's puzzle.
- Activity: diagram.
- Activity replay.
- Session management.

B.1 Instructor home

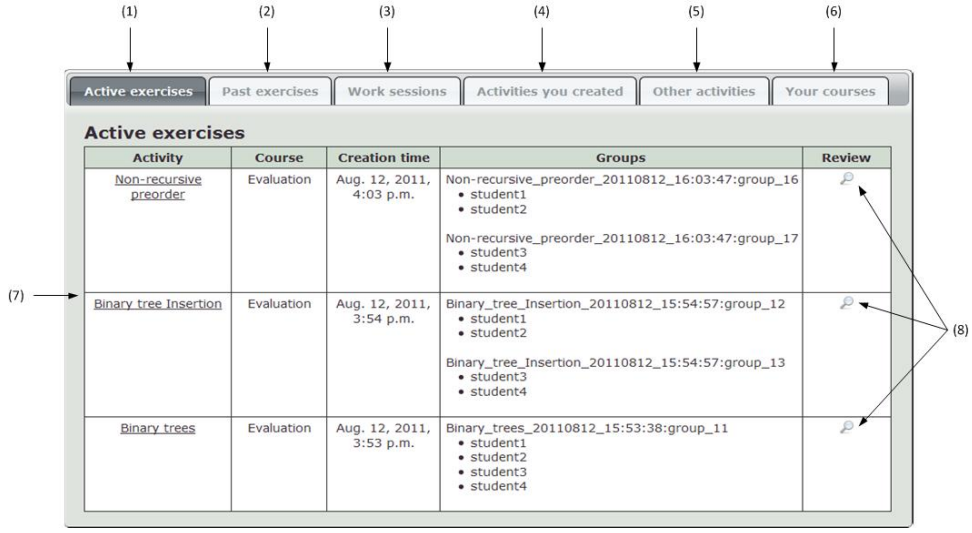


Figure B.1: Instructor home.

Main controls:

1. "Active exercises" tab: lists all live activity instances launched by the instructor.
2. "Past exercises" tab: lists the inactive and concluded activities previously launched by an instructor.
3. "Work sessions" tab: shows a catalog of the work sessions managed by the instructor who is currently logged into the system.
4. "Activities you created" tab: list of the activities created by the instructor who has logged into the system.
5. "Other activities" tab: lists other activities available, created by instructors other than the one that is accessing the site.
6. "Your courses" tab: lists basic information about the courses where the user is assigned as an instructor.
7. Main display: it shows the contents of each of the previous tabs.
8. Review activity icon: gives access to the corresponding activity review area.

B.2 Student home

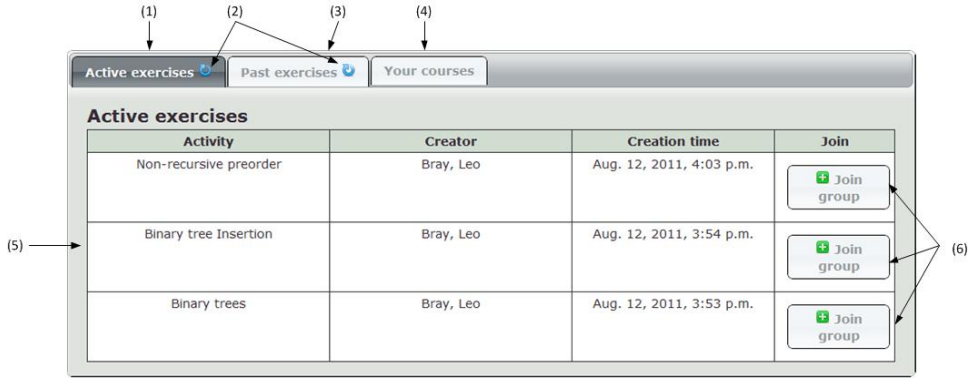


Figure B.2: Student home.

Main controls:

1. "Active exercises" tab: lists all active activities in which the student is included.
2. Refresh icons: allows the student to update the lists of active and past exercises.
3. "Past exercises" tab: lists the inactive and concluded activities in which the student took part.
4. "Your courses" tab: lists basic information about the courses in which the user has enrolled.
5. Main display: it shows the contents of each of the previously described tabs.
6. "Join group" buttons: these buttons give access to the corresponding group activities.

B.3 Instructor live activity review

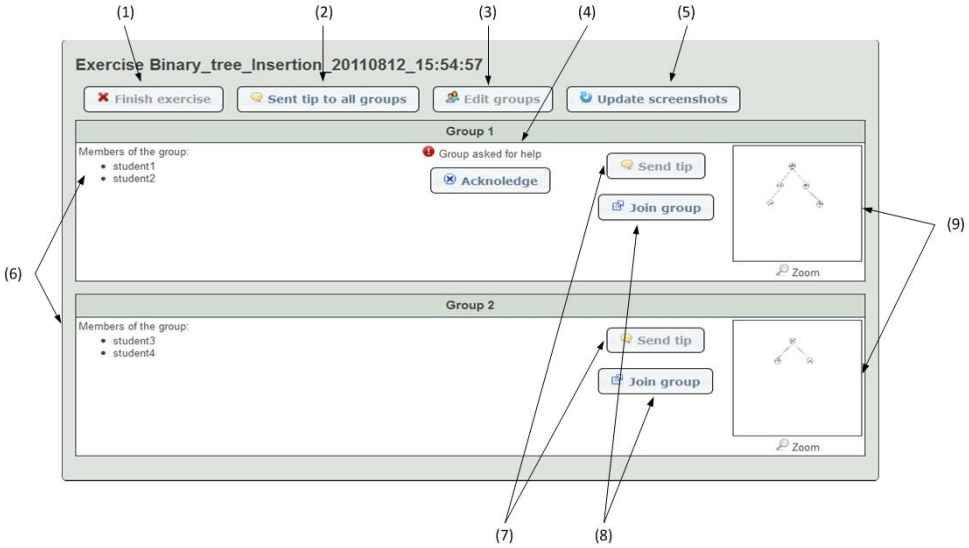


Figure B.3: Instructor live activity review.

Main controls:

1. "Finish exercise" button: allows the instructor to mark the activity as concluded.
2. "Send tip to all groups" button: shows the instructor a dialog to send a textual hint to all the groups undertaking the activity.
3. "Edit groups" button: allows the instructor to modify the student groups once the activity has started.
4. Help notification: when a student group asks the instructor for help, the display shows that condition in this way. Pressing the button "Acknowledge", the system hides this message until further requests are made by the students.
5. "Update screenshots" button: updates the images that show the current state of the shared artifact in which each group is working on.
6. Group information: lists the details of the groups taking part in the activity.
7. "Send tip" buttons: the instructor can send textual tips to particular groups using these buttons. When the instructor clicks one of these buttons, the system will display a dialog where the textual hint can be typed and confirmed to be sent.

8. “Join group” buttons: if necessary, the instructor can join one of the groups to manipulate their shared artifact directly, or communicate with the students in that particular group using the chat room provided.

9. Shared artifact current state: shows a thumbnail of the current state of the shared artifact used by each group. When the user clicks on any of these thumbnails, the system displays the corresponding image in full-size.

B.4 Instructor inactive activity review

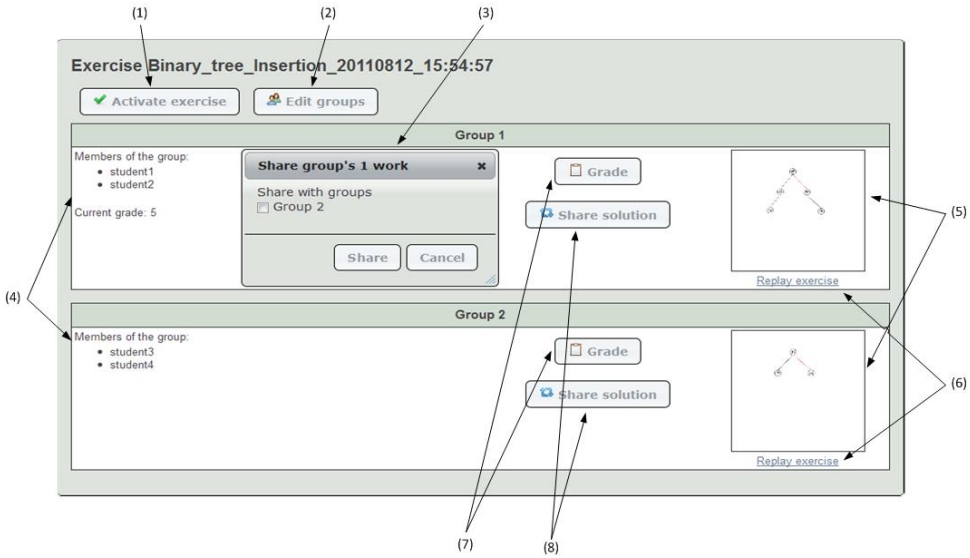


Figure B.4: Instructor inactive activity review.

Main controls:

1. "Activate exercise" button: allows the instructor to activate the exercise.
2. "Edit groups" button: gives access to the instructor to modify the student groups.
3. "Share group work" dialog: this dialog allows the instructor to indicate with which student groups he or she wants to share the work of a particular team.
4. Group information: lists the general information of the group, including the students that form the group and the marks given by the instructor (if available).
5. Shared artifact state: shows a thumbnail of the last available state of the shared artifact used by each group. When a user clicks on any of these thumbnails, the system displays the corresponding image in full-size.
6. "Replay exercise" links: gives the instructor access to the replay of any group's work.
7. "Grade" buttons: launches a dialog that allows the instructor to mark the group's work in a scale from 1 to 10.
8. "Share solution" buttons: launches the "Share group work" dialog that corresponds to that group.

B.5 Activity: annotated slideshow

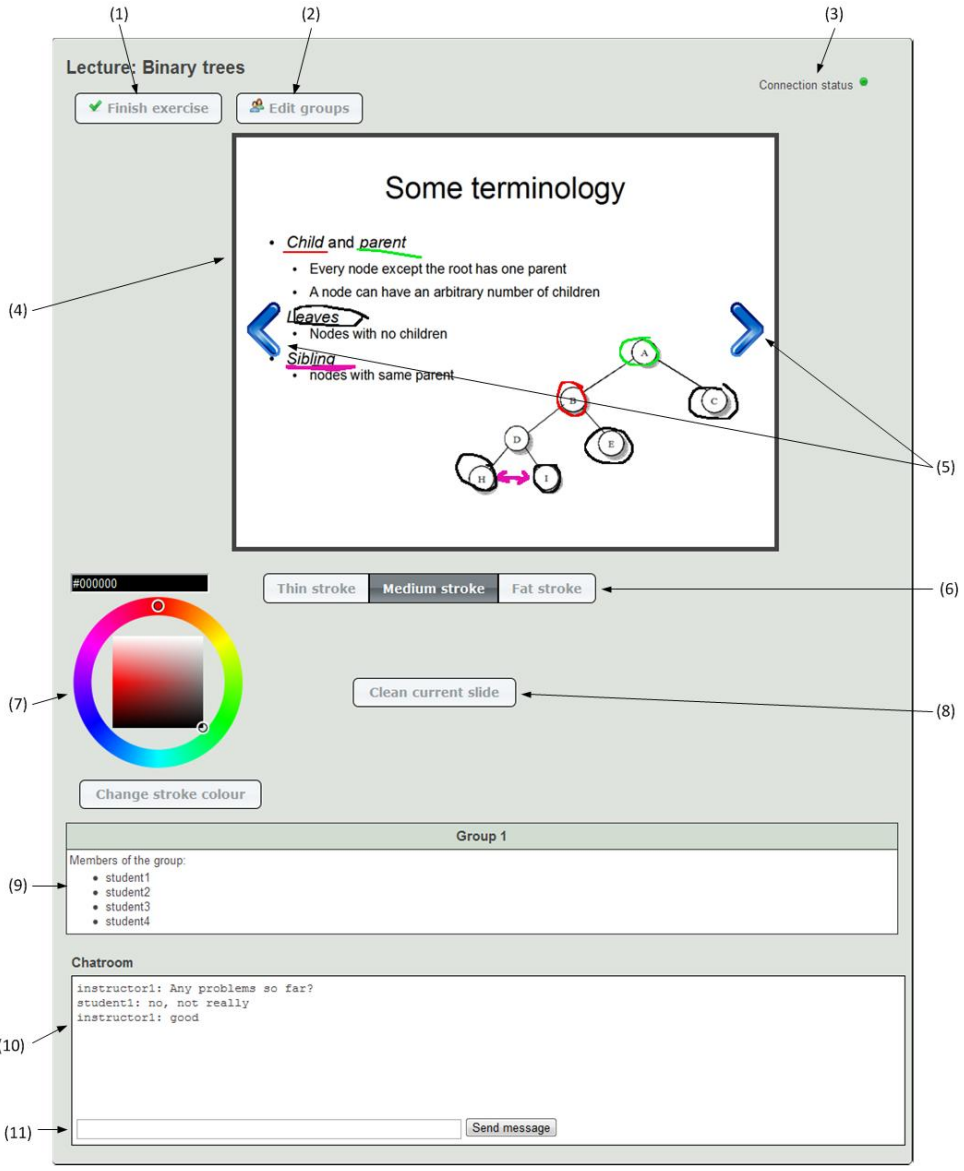


Figure B.5: Controls for annotated slideshow.

Main controls:

1. "Finish exercise" button: allows the instructor to mark the activity as concluded.
2. "Edit groups" button: allows the instructor to rearrange the student groups once the activity has started.
3. Connection status: indicates the current status of the WebSocket-based connection. This indicator is gray if the connection is being established, green if the connection is up, red if it breaks.

4. Main display.
5. These handles allow the instructor to browse the slides.
6. Stroke weight selector.
7. Stroke colour selector.
8. “Clean current slide” button: erases all the strokes drawn on the slide currently displayed.
9. Group information: lists the students that are going to receive a live feed of the slideshow.
10. Chatroom: displays the messages interchanged between the members of the group.
11. Chatroom input field.

B.6 Activity: categorization and Parson's puzzles

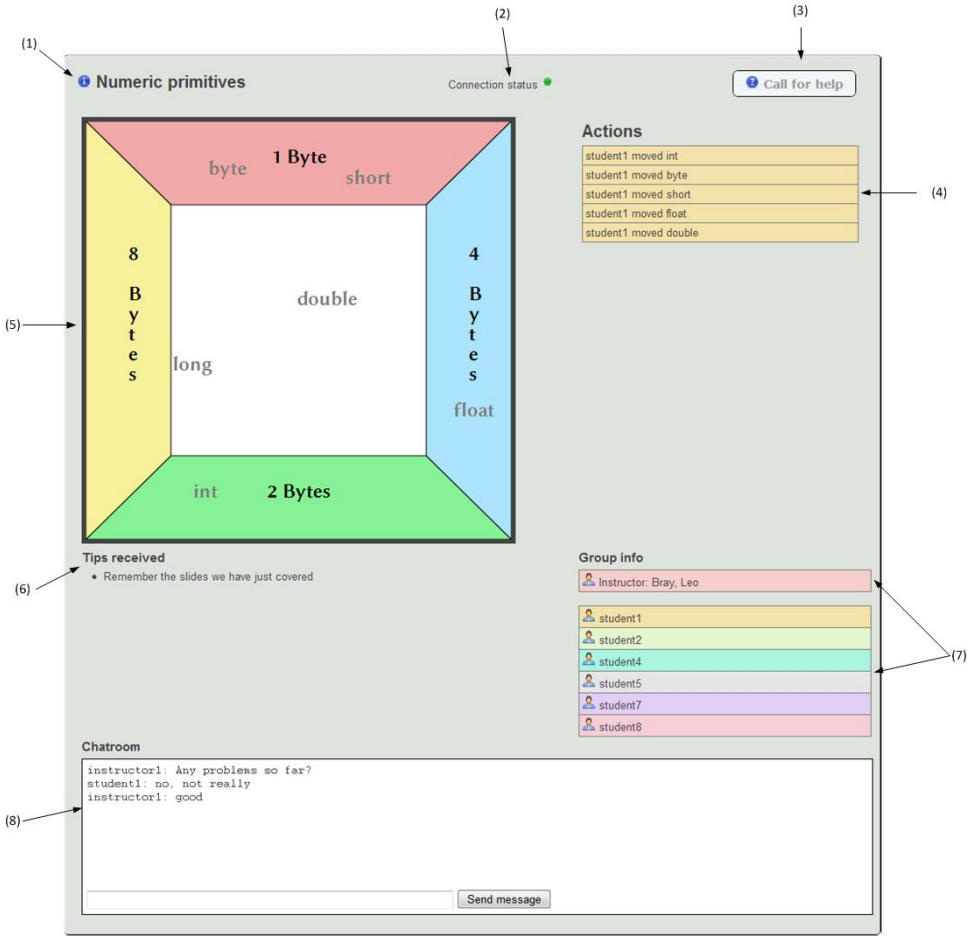


Figure B.6: Controls for categorization activity.

Main controls:

1. Activity description: when this icon is clicked, the system displays a short description of the activity and its goals.

2. Connection status: indicates the current status of the WebSocket-based connection. This indicator is gray if the connection is being established, green if the connection is up, red if it breaks.

3. "Call for help" / "Leave group" button. Regular students will be displayed with a "call for help" button that allows them to request help from the instructor. Instructors, on the other hand, will be shown with a "leave group" button, that will take them out of this section.

4. Action log: lists all the actions performed by the users in order to modify the shared artifact. Each action has its background colored with the colour that corresponds to its performer.

5. Shared artifact.

6. Tips received: list of all the textual hints sent by the instructor to the group.

7. Group information: lists the users who are taking part in the activity. A colour-based legend is displayed here, associating each user with a colour.

8. Chatroom: displays the messages interchanged between group members. It also provides an input field to send new messages.

B.7 Activity: diagram

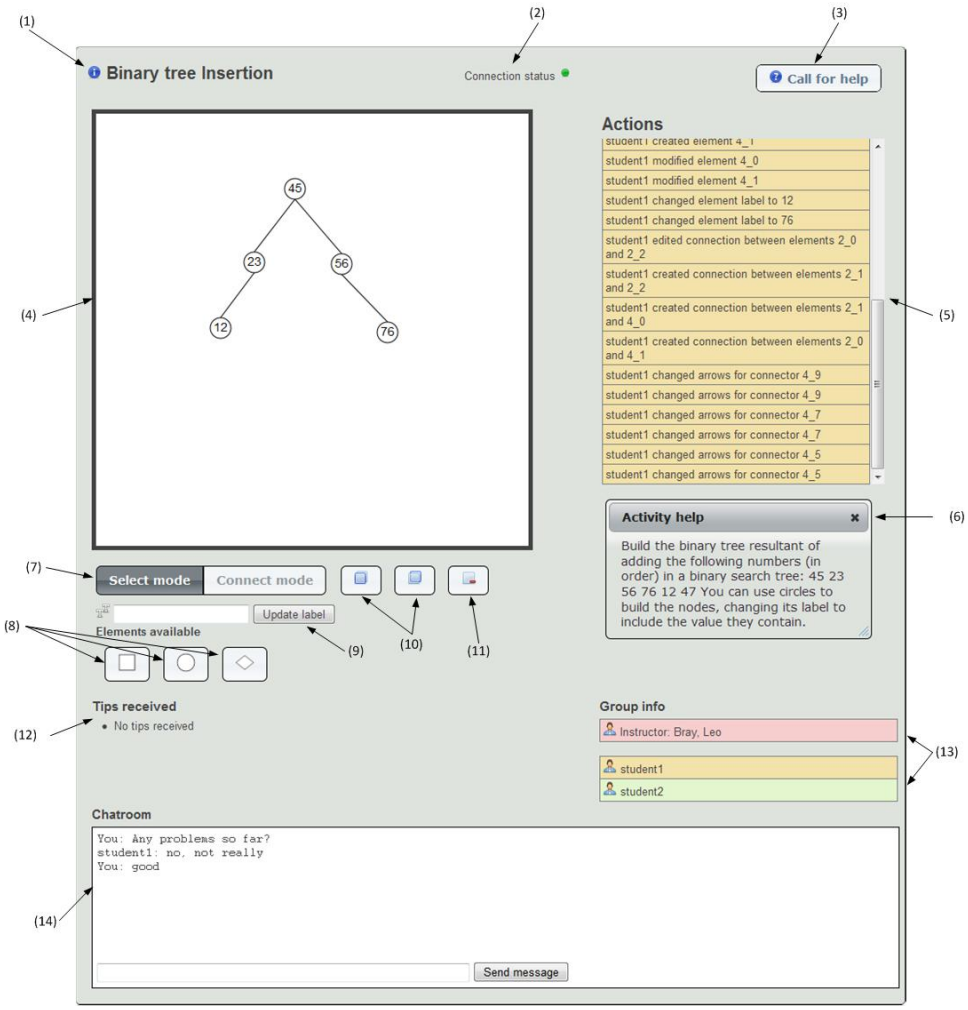


Figure B.7: Controls for diagram activity.

Main controls:

1. Activity description: when this icon is clicked, the system displays a short description of the activity and its goals.

2. Connection status: indicates the current status of the WebSocket-based connection. This indicator is gray if the connection is being established, green if the connection is up, red if it's broken.

3. "Call for help" "Leave group" button. Regular students will be displayed with a "call for help" button that allows them to request help from the instructor. Instructors, on the other hand, will be shown with a "leave group" button, that will take them out of this section.

4. Shared artifact.

5. Action log: lists all the actions performed by the users in order to modify the shared artifact. Each action has its background colored with the colour that corresponds to its performer.

6. Dialog that shows the instructions of the activity.

7. Mode selector: allows the user to alternate between selection and connection modes. In selection mode, the user can select items, move and resize them, change their label, take them to the front or back layer of the diagram, or delete them. When the diagram is in connection mode, the user will be able to connect items with directed or non-directed links.

8. Add new shape: adds a new element to the diagram.

9. Update element label: changes the text associated to a particular item in the diagram.

10. Send to front/back: sends a selected item to the front or back layer of the diagram.

11. Delete element: if an item of the diagram is already selected, this button performs its deletion.

12. Tips received: list of all the textual hints sent by the instructor to the group.

13. Group information: lists the users who are taking part in the activity. A colour-based legend is displayed here, associating each user with a colour.

14. Chatroom: displays the messages exchanged between group members, and provides an input field to send new messages.

B.8 Activity replay

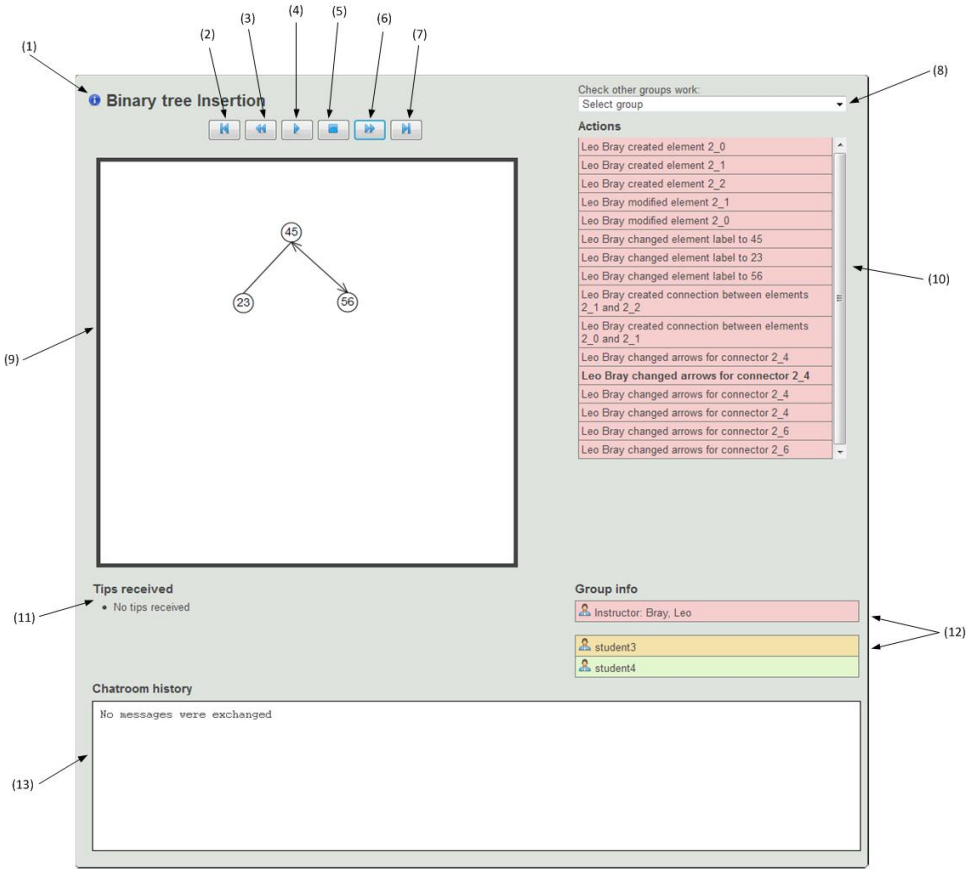


Figure B.8: Controls for activity replay.

Main controls:

1. Activity description: when this icon is clicked, the system displays a short description of the activity and its goals.
2. “Go to start” button: takes the shared artifact to its initial state.
3. “Previous” button: goes back one action.
4. “Play” button: starts a step-by-step replay of all the actions performed on the shared artifact.
5. “Stop” button: stops the replay.
6. “Next” button: executes the next action in the registry.
7. “Go to end” button: takes the shared artifact to its final state.

8. Shared groupwork: using this selection box, group members can access the work of other groups which have been shared by the instructor.

9. Shared artifact.

10. Action log: lists all the actions performed by the users in order to modify the shared artifact. Each action has its background colored with the colour that corresponds to its performer. Clicking a particular action in this registry takes the shared artifact to this point of execution.

11. Tips received: lists all the textual hints sent by the instructor to the group.

12. Group information: lists the users who took part in the activity. A colour-based legend is displayed here, associating each user with a colour.

13. Registry of the messages exchanged between the group members.

B.9 Work session management

Figure B.9: Controls for session management.

Main controls:

1. List of the activities that the work session consists of.
2. "Review exercise" button. The instructor can access the review section that corresponds to an activity by clicking this button, either if it is active or inactive.
3. "End exercise" button: marks an activity as inactive or concluded.
4. "Start activity" button: launches the corresponding activity.
5. Group information: shows the composition of the student groups arranged for this working session.
6. "View session details" button: shows detailed information of this work session.

Bibliography

[1] Apache HTTP server project. `http://httpd.apache.org/`, 2011.

[2] Django Web framework. `https://www.djangoproject.com/`, 2011.

[3] Jetty WebServer. `http://jetty.codehaus.org/jetty/`, 2011.

[4] MDN Mozilla Documentation. `https://developer.mozilla.org/en/WebSockets`, 2011.

[5] MySQL database. `http://www.mysql.com/`, 2011.

[6] Python programming language. `http://www.python.org/`, 2011.

[7] S.M. Alessi and S.R. Trollip. *Multimedia for learning*. Allyn and Bacon, 2005.

[8] Richard Anderson, Ruth Anderson, Peter Davis, Natalie Linnell, Craig Prince, Valentin Razmov, and Fred Videon. Classroom presenter: Enhancing interactive education with digital ink. *Computer*, 40:56–61, September 2007.

[9] Ian Beatty and William Gerace. Technology-enhanced formative assessment: A research-based pedagogy for teaching science with classroom response technology. *Journal of Science Education and Technology*, 18:146–162, 2009. 10.1007/s10956-008-9140-4.

[10] Dave Berque, Terri Bonebright, and Michael Whitesell. Using pen-based computers across the computer science curriculum. In *Proceedings of the 35th SIGCSE technical symposium on Computer science education*, SIGCSE '04, pages 61–65, New York, NY, USA, 2004. ACM.

[11] A.L. Bishop, R.K. Dinkins, and J.L. Dominick. Programming handheld devices to enhance learning. *Educause Quarterly*, 26(1):50–54, 2003.

[12] J.D. Bransford, A.L. Brown, and R.R. Cocking. *How people learn*. National Academy Press, 2000.

[13] I. Hickson. HTML5 Draft Standard, 3 November 2009. `http://www.whatwg.org`, 2009.

[14] L.S. Huang, E. Chen, A. Barth, E. Rescorla, and C. Jackson. Talking to yourself for fun and profit. *Proceedings of W2SP*, 2011.

[15] Petri Ihantola and Ville Karavirta. Open source widget for parson's puzzles. In *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education*, ITiCSE '10, pages 302–302, New York, NY, USA, 2010. ACM.

[16] Sam Kamin, Michael Hines, Chad Peiper, and Boris Capitanu. A system for developing tablet pc applications for education. In *Proceedings of the 39th SIGCSE technical symposium on Computer science education*, SIGCSE '08, pages 422–426, New York, NY, USA, 2008. ACM.

[17] Kenneth L. Kraemer, Jason Dedrick, and Prakul Sharma. One laptop per child: vision vs. reality. *Commun. ACM*, 52:66–73, June 2009.

[18] Patricia Lasserre and Steven Smithbower. A proposal for a new communication medium in the classroom. In *Proceedings of the 15th Western Canadian Conference on Computing Education*, WCCCE '10, pages 5:1–5:5, New York, NY, USA, 2010. ACM.

[19] P. Lubbers, B. Albers, F. Salim, and R. Smith. *Pro HTML5 Programming: Powerful APIs for Richer Internet Application Development*. Apress, 2010.

[20] R.C. Meurant. Providing every student with an ipad as a means of helping develop korean efl digital literacy. In *Networked Computing and Advanced Information Management (NCM), 2010 Sixth International Conference on*, pages 242 –247, 2010.

[21] Z.R. Mevarech. Who benefits from cooperative computer-assisted instruction? *Journal of Educational Computing Research*, 9(4):451–464, 1993.

[22] L.K. Michaelsen, L.D. Fink, and A. Knight. Designing effective group activities: Lessons for classroom teaching and faculty development. *To improve the academy*, 16:373–398, 1997.

[23] D. Parsons and P. Haden. Parson's programming puzzles: a fun and effective learning tool for first programming courses. In *Proceedings of the 8th Austalian conference on Computing education-Volume 52*, pages 157–163. Australian Computer Society, Inc., 2006.

[24] Jennifer Pearson and George Buchanan. Real-time document collaboration using ipads. In *Proceedings of the third workshop on Research advances in large digital book repositories and complementary media*, BooksOnline '10, pages 9–14, New York, NY, USA, 2010. ACM.

[25] Chad Peiper, David Warden, Ellick Chan, Boris Capitanu, and Sam Kamin. efuzion: development of a pervasive educational system. In *Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education*, ITiCSE '05, pages 237–240, New York, NY, USA, 2005. ACM.

[26] J. Preece, Y. Rogers, and H. Sharp. *Interaction design: Beyond human-computer interaction*. Wiley Dreamtech, 2011.

[27] B. Rosenshine and R. Stevens. Teaching functions. *Handbook of research on teaching*, 3:376–391, 1986.

[28] Nicolai Scheele, Anja Wessels, Wolfgang Effelsberg, Manfred Hofer, and Stefan Fries. Experiences with interactive lectures: considerations from the perspective of educational psychology and computer science. In *Proceedings of th 2005 conference on Computer support for collaborative learning: learning 2005: the next 10 years!*, CSCL '05, pages 547–556. International Society of the Learning Sciences, 2005.

[29] Michelle Wilkerson, William G. Griswold, and Beth Simon. Ubiquitous presenter: increasing student access and control in a digital lecturing environment. In *Proceedings of the 36th SIGCSE technical symposium on Computer science education*, SIGCSE '05, pages 116–120, New York, NY, USA, 2005. ACM.

[30] T. Yoshino. New WebSocket Protocol: Secure and Extensible. `http://blog.chromium.org/2011/08/new-websocket-protocol-secure-and.html`, 2011.