Learning by doing: vLab, a virtual laboratory for Computer Engineering education

Roberto Laschi
Dip. Elettronica Informatica e Sistemistica
Università di Bologna, Italy
rlaschi@deis.unibo.it

Anna Riccioni
Dip. Elettronica Informatica e Sistemistica
Università di Bologna, Italy
ariccioni@deis.unibo.it

Patrik Suzzi
Dip. Elettronica Informatica e Sistemistica
Università di Bologna, Italy
patrik.suzzi@studio.unibo.it

Abstract

In this paper we present the design and implementation of an advanced virtual lab aimed at supporting teaching and learning in Computer Engineering courses. We first outline the requirements an educational tool must meet to represent an effective and comprehensive support for students through the whole process that leads them to achieve conceptual understanding and technical skills. The described virtual lab, based on a constructivist approach and suitable for use in blended-learning contexts, responds to a general model that allows its application in different educational domains. In the second part of the paper we describe the implementation of vLab 2.0, a prototype built following the proposed directions and the results of a prior experimentation within the Information Security Technologies course hold at the Engineering Faculty of the University of Bologna. vLab 2.0 relies on Eclipse technology in order to guarantee openness and extensibility, and to access a set of advanced features which concur in providing support for building a comprehensive and sophisticated tool for teaching and learning.

1 Introduction

Information and Communication Technologies are constantly advancing and becoming more pervasive. Nowadays, one of the most challenging issues is represented by the individuation of new, complex and effective ICTs application. Education is a sector which can greatly benefit from ICTs improvements and proper adoption.

Our research project aims at defining and implementing an integrated tool to support teaching and learning in Computer Engineering courses. The ideal framework in which this instrument can be effectively used is blended learning, here intended as a combination of two different approaches: synchronous traditional face-to-face lessons and asynchronous, individual experimentations in a virtual laboratory.

One of the main skills a computer engineer must achieve during his studies is the ability to design a well dimensioned software system, able to satisfy the defined requirements in a specific domain. Once that the specifications and the domain have been settled, the engineer starts an incremental process which cycles through four main steps:

- analyze the problem
- sketch a solution
- verify how the model fits the requirements
- design, implement and test a prototype.

What we want to do is to define the requirements an educational tool must meet in order to represent an effective and comprehensive support for students through the whole process that leads them to achieve conceptual understanding and technical skills.

Modeling frameworks and simulators have a strategic importance in helping students to gain greater insight into subjects [1, 2, 3]. Many authors have pointed out that the use of simulation tools often reinforces learning, offering a proper support for verifying theory, and contributes to improve students' performances in various disciplines [2, 3]. Especially in Computer Engineering, modeling skills are traditionally taught relying on CAD tools [3, 4].

To achieve technical proficiency and to improve practical programming skills students often benefit from the employ of advanced IDEs. They can acquire better understanding of subjects by quickly designing, simulating and then testing and extending systems [2].

Our purpose has been to integrate these different functionalities in one comprehensive educational tool. In our perspective, this tool should be projected towards educational and not industrial objectives. This makes it suitable to focus on relevant didactic aspects and to suggest an educational, partial code to students as a

starting point for software development, so that they can complete, modify and reuse it in different and more complex contexts. Following this direction, our virtual lab helps users to:

- understand and solve problems related to key theoretical concepts
- solve problems through the design and construction of new artifacts or processes
- gain and improve the technical skills their future professional roles will require.

One of the main facets of the project is its suitability to be used in different formative domains. This feature is achieved through the adoption of one of the best-known abstraction in computer engineering field: the block diagram [5, 6].

2 Requirements

In 2005 we developed a first prototype, vLab 1.0, taking the course on Information Security Technologies hold at the Engineering Faculty of the University of Bologna as our reference case study.

vLab 1.0 GUI was composed by three synchronized areas: the main one shows a graphical representation of the system to be analyzed and allows students to configure and run experiments. The second area is dedicated to prompt feedback reporting: for each component part of the system whose execution successfully ended, notes about data dimension and execution time are presented. On the other hand, for those components that could not complete their execution, usually because of a wrong parameters setting, errors are shown in red. Finally, a third view displays an educational sample code related to the ongoing exercise. A two-year experimentation conducted by making vLab 1.0 available to around 200 students confirmed the effectiveness of the chosen approach and served as a successful acceptance test, but also pointed out vLab 1.0 main limit: its lacking support to students' autonomous experimentation, based on self-constructed experiments. In line with these results, we focused on constructivist theory, which assumes that students should be provided with an open and extensible environment: they have to be actors, other than active, in their own learning process [7]. For this purpose it is fundamental to provide students with various instruments aimed at covering the different functionalities which can have a central role in the learning process [7, 8, 9]. Figure 1 shows how to reap these broad goals, outlining the major actors and features involved.

The sketch represents the workbench dedicated to the graphical modeling of the current experiment as the central aspect within the virtual lab. The workbench includes a library of primitive components, that students can use to construct their own experiments. It is also possible for users to load a predefined exercitation, chosen among the available sets, and configure and run it. According to this approach, tutors and teachers are charged with the responsibility of creating a starting collection of experiments that learners can set and run; but then learners too can cover the instructor's role and modify or build their own trials from scratch. Thus saving and restoring features in a standard format, so that new built experiments can be shared between users, assume a primary importance within the virtual lab.

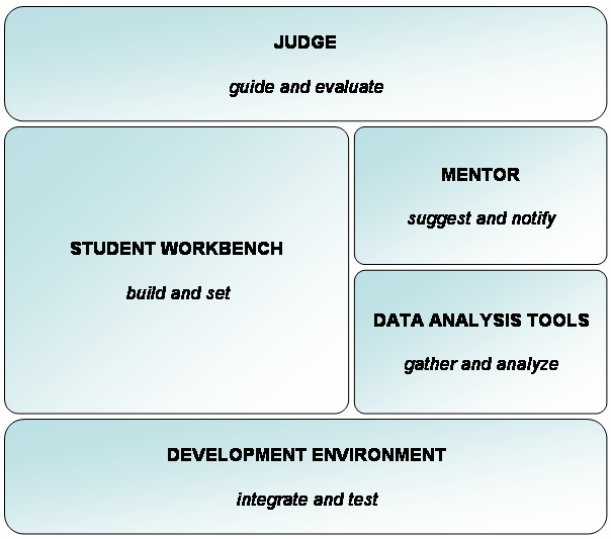

Figure 1. Actors and tools role in the virtual lab environment.

The workbench should also let users switch between different detail levels, in order to analyze different sides of the same topic. This can provide a more effective, customizable and comprehensive didactic experience and is central to gain a major compliance with different learning styles. Especially for computer science students it is important to have the ability to access both the systemic and the algorithmic view within a same solution, while referring to our case study such a facility enables to switch from a security service perspective to a mechanism-oriented one.

Another issue within the virtual lab is represented by help instruments. In fact, activities can be introduced and proposed by an intelligent tutoring system, the "judge" in figure 1, which evaluates users' performances and suggests new tasks according to students achievements and objectives. Furthermore, each step conducted within an experiment can be followed by a "mentor" aimed at pointing out possible errors and suggesting how to correct them. Users' understanding and reflection can also be improved by allowing the access to external additional resources, such as technical documentation or reference standards.

The virtual lab also includes gathering and analysis tools whose function is to help users in collecting experimental data and interpreting them. Finally, users can switch at any time to a development environment integrated within the lab to complete,

modify, debug, test and run a program, starting from a partial educational code automatically produced during an exercitation.

With these features, the virtual lab can really support the constructivist approach and learners' active and primary role in their own educational experience. Building and sharing new knowledge can, in fact, pass through:

- the construction of new experiments
- the modification of predefined experiments
- non trivial experiments settings
- the interpretation of gathered experimental data relying on the featured analysis tools
- the discovery or application of new analysis methods
- accessing multiple external information sources.

Furthermore, users can extend the virtual lab in order to comply with their changing needs and share their new knowledge with other users, for instance adding their new built schemas to shared libraries, or implementing special analysis tools for specific needs.

3 Eclipse technology

The implementation of a prototype adherent to the exposed constructivist approach has requested the design of an appropriate architecture and the adoption of a proper technology suitable for its realization.

The architecture of an effective educational tool must be open and extensible. Eclipse plug-in architecture and extension point mechanism perfectly address this major issue [10]. Relying on Eclipse technology is crucial also in order to access a set of basic and advanced features essential for providing a complete and sophisticated tool for support teaching and learning in higher education. Eclipse technology simplifies the creation and coordination of synchronized multiple editors, views and perspectives and makes straightforward to integrate a fully-featured IDE customizable for different programming languages. Furthermore, within Eclipse Community several projects are being developed as general purpose plug-in which can effectively be integrated within our educational tool.

Table 1 outlines the mapping between the requirements previously discussed and the proper technical solutions selected within Eclipse technologies.

Requirement	***Solution***
Judge	BPEL
Student workbench	GEF-based editor
Mentor	Ganymede
Data analysis tools	BIRT and special purpose plug-ins
Development environment	JDT

Table 1. Mapping between requirements and technical solutions.

Figure 2 outlines the resulting vLab 2.0 architecture, which is represented by a stack where each tier depends upon, uses and integrates the level below.

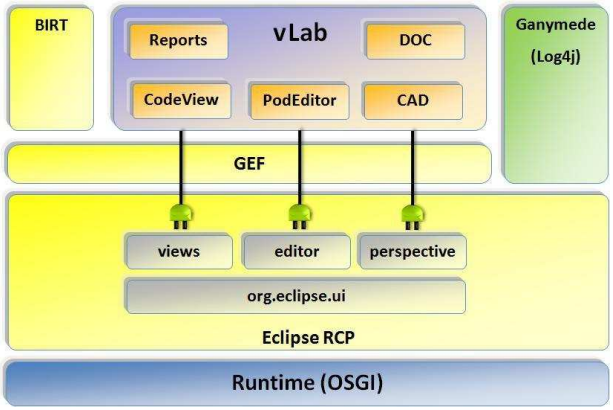

Figure 2. The virtual lab architecture based on Eclipse technology.

The OSGi runtime, responsible for the coordination of multiple and different software components, is the foundation of the framework [11]. The Eclipse Rich Client Platform relies on OSGi services to provide a specific implementation of the standard, replacing the generic OSGi bundles with Eclipse plug-ins [12]. It also offers a new set of facilities, collected in the org.eclipse.ui package, essential for the Eclipse Development Framework and tailored to support external contributes through the extension point mechanism. The vLab feature leverages different resources to provide its added functionalities: among them, the Eclipse RCP, the Graphical Editing Framework [13] and several projects, like BIRT [14], or third-parties plug-ins such as Ganymede [15], built to integrate Log4J within Eclipse applications. Accordingly to Eclipse technology main characteristics, vLab is supplied with a set of extension points in order to be open to future expansions.

3.1 The black-box model

The exercitations within the virtual lab are based on a logical model that reflects the block diagram abstraction.

The block diagram key elements are blocks and connections. The blocks have an associated function or system of functions, and can have inputs, outputs and parameters which are graphically represented by nodes. The connections are oriented and must respect some basic rules related to data flow to be considered syntactically valid: each input or parameter node can only be a target for at most one connection, while output nodes can propagate their value towards multiple destinations being the source of several connections.

To cope with increasing schema complexity, the box abstraction has been added. Box represents a

boundary which can isolate a sub-graph providing it with an external interface through pins. Box pins are called "adapters" because of their double function: other than performing data flow, they can allow users to accomplish the possibly required data type conversions. Boxes can also be obscured to hide their inner structure: thus, they represent the needed support for implementing the hierarchical design approach and for switching between different detail levels within the same schema. Finally, boxes can be used to introduce recursion and iteration.

Since blocks can execute, and so change their state, it's necessary to define a color code to discriminate between different states. Orange blocks are missing input data, while yellow blocks are fully configured and ready to be executed. A similar semantic occurs within nodes, too: a node whose associated value is null is colored in red, and becomes green when the handled data assumes a defined value.

The following figure illustrates the depicted model.

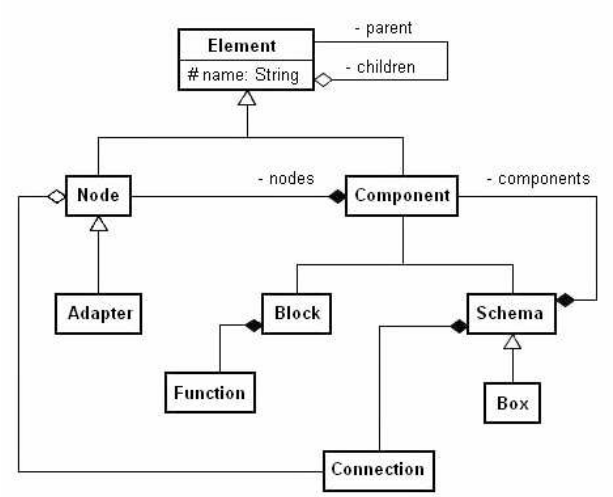

Figure 3. The underlying abstract model.

3.2 Plug-ins

The primary way users have to interact with the virtual lab is through a graphical representation of the exercitation based on the depicted logical model. For this reason, the GUI is composed by the workbench centered on a GEF-based editor, and by several views. Each view is aimed at a specific task, and synchronized with the underlying model.

The editor includes a palette that displays the available components and allows vLab users to build their schemas by mean of drag and drop operations. Users can also load a predefined exercitation and then modify it. The Eclipse outline view provides an alternative tree representation of the schema elements.

Once that the schema has been completed, users can switch from the modeling phase to the simulation of the built system, setting and then executing all of the components. The Eclipse property view supports users in configuring the required parameters, thanks to a business logic which allows to assign a value only to disconnected input or parameter nodes: in fact, a connection between two nodes represents a channel that propagates the source data value to the target.

When simulating a system behavior, it is possible to trace and record a series of characteristic values which can be used to perform tests and evaluations, other than to refine system dimensioning. Engineers usually base their efficiency evaluations on time and data dimension: for this reason the virtual lab can trace, for each block, the associated function execution time and input and output data dimensions. Once that these information have been saved, they can be displayed within a BIRT report: BIRT plug-in, which can be easily integrated with the virtual lab, allows users to represent gathered data according to various layout and to complete them with aggregated and derived information. As a consequence of the mechanism of extension points, the virtual lab can also be extended with other special purpose plug-ins, when tracing time and data dimension is not enough.

After modeling and simulation support, it is important to provide the third requested feature: automatic educational code generation. To achieve this purpose we implemented an extension of Eclipse basic view, aimed at integrating the outline of the graphical schema with the representation in form of partial code of each schema element. Saving the generated content as a file with the appropriate extension enables users to switch to the proper development perspective.

Another key feature is represented by the online interaction support, which is offered through various services: the first one is the *"judge" plug-in*, currently in a planning phase, which may rely on the BPEL project [16, 17]. A central role is also covered by the *prompt feedback reporting view*, based on Ganymede plug-in and aimed at immediately notify users with errors or significant info related to the current activity. In order to simplify the construction and test of new schemas, Ganymede displays "info"-tagged messages to give positive or neutral feedback, "warn" messages to point out incomplete configurations or not allowed operations, and "error" message to report exceptions obtained during run-time simulation. The third component involved in offering a complete interaction support is a comprehensive *help online* documentation, related both to the virtual lab usage and to the components available in the palette. Finally, it is important to provide users with a documentation *perspective*, that allows students to consult additional and external resources using a browser view and a suggested and extensible link list.

3.3 Screenshots

According to the requirements previously outlined and to the correspondent technical solutions individuated in the Eclipse technology framework, we integrated our research with the development of vLab 2.0, a prototype of advanced virtual lab.

Figure 4 shows a very simple example of how the virtual lab works. To make the screenshot more readable, it has been focused on only three views: the GEF-based editor with its palette, the code view and the XML view. These views usually appear in the exercitation perspective together with other facilities.

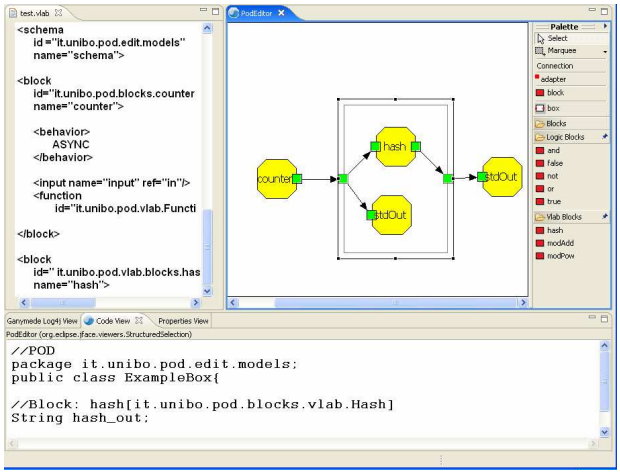

Figure 4. vLab 2.0 screenshot (exercitation perspective): editor, code view, XML view.

The schema represented in the editor is composed of four primitive components and a box with two adapters, which could be obscured to hide its content. The editor palette is organized in different categories related to libraries which can pertain to various domains; a special category can be reserved for the student's self-constructed new components.

Accordingly to the Model-View-Controller pattern, the various views within the virtual lab all rely and operate on the same underlying model in order to keep themselves synchronized. During a work session within vLab, the shared model structure or the data it holds change following users' actions: the system must then propagate the suitable notifications to the interested recipients, in order to make them able to properly respond to the modifications occurred. The common model can be represented through an abstract tree that, when building the XML description of a graphical schema, is translated into a corresponding Document Object Model.

Figure 5 focuses on a different combination of views, still referring to the exercitation perspective: the right part of the screenshot shows the "mentor", implemented by a log view based on Ganymede plug-in. The log view contains messages tagged as "info", "warn" and "error", each one represented in a different text color, respectively green, orange and red. These messages have been interactively produced during the modeling and the simulation phases, in consequence of specific actions performed by the user.

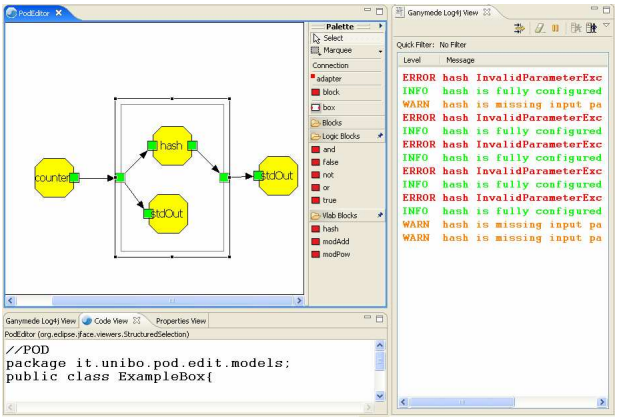

Figure 5. vLab 2.0 screenshot (exercitation perspective): editor, XML view, log view.

vLab 2.0 offers also other perspectives, each one providing different features through various plug-ins. For example, a BIRT report can be used to display the data collected during simulations. Furthermore, the virtual lab will soon integrate a special purpose plug-in aimed at performing FIPS-140-2 tests to evaluate bits casuality and another one focused on frequency distributions analysis.

3.4 Delivery

vLab 2.0 will be extensively tested within the 2008 edition of the course on Information Security.

Students will be able to access vLab 2.0 through AlmaChannel, the University of Bologna e-learning platform [18, 19]. Different distributions will be available for download: a feature-based one, for those users who already have Eclipse platform installed on their computers, and a Rich Client Platform version, suitable for who prefers to deal with a ready-to-run product [12]. The RCP will probably integrate SourceForge Dr.Java plug-in in order to simulate a simplified Java Development Environment [20]. The availability of different distributions, one of which simpler to use, is desirable in the experimental context: the course on Information Security Technologies has a mixed target composed of students with various starting informatic competences.

4 Conclusions and future work

In Computer Engineering education a central issue is to make students achieve strong design and programming skills. Providing students with advanced tools aimed at support them during modeling, testing, evaluation and implementation of software systems can

help reaching this objective. Such a tool, to be effective, must be complete, highly customizable and offer an integrated support for the various step the learning process goes through. Our research led us to design vLab, an advanced tool aimed at satisfying the described requirements.

The system we are currently developing, vLab 2.0, relies on Eclipse technology. The plug-in architecture is essential to supply extensibility and to reuse third-parties tools compliant with our needs. Finally, Eclipse provides support for designing different solutions for delivery and customizing packages with different features dedicated to specific targets.

Our future efforts will be addressed both at improving vLab adding new advanced features and at dealing with new open and challenging issues. A key point is represented by users' activities evaluation, and is strictly connected to the presentation of customized feedback and the suggestion of new tasks based on users' profiles, formative objectives and activity history. This research direction will also imply an integration with the standard SCORM [21]. Furthermore, we are planning to analyze BPEL suitability to define and manage users' portfolios and curricula.

An interesting and useful achievement would be to build new component libraries to adapt vLab 2.0 for being adopted also in contexts other than Information Security Technologies. A suitable new frame is represented by Logic Circuits, as we experimented following Eclipse GEF tutorial based on logic diagrams. Finally, it is possible to make vLab 2.0 provide educational code in various programming languages, other than Java.

Bibliography.

1 Chung G. K., Harmon T. C., Baker E. L., (2001). The Impact of a Simulation-Based Learning Design Project on Student Learning, IEEE Transactions on Education 44 (4): 390-398.

2 Büchner P., Nehrir M H., (1991). A Block-Oriented PC-Based Simulation Tool For Teaching and Research in Electric Drives and Power Systems, IEEE Transactions on Power Systems 6 (3): 1299-1304.

3 Allwood J. M., Cox B. M., Latif S. S., (2001). The Structured Development of Simulation-Based Learning Tools With an Example for the Taguchi Method, IEEE Transactions on Education 44 (4): 347-353.

4 Ma J., Nickerson J. V., (2006). Hands-on, Simulated, and Remote Laboratories: A Comparative Literature Review, ACM Computing Surveys 38 (3): art. n. 7. DOI= http://doi.acm.org/10.1145/1132960.1132961.

5 Mano M. M., Kime C. R., (2004). Logic and Computer Design Fundamentals, Pearson Prentice Hall.

6 Stallings W., (2006). Cryptography and Network Security, Pearson Education.

7 Black J. B., McClintock R. O., (1995). An Interpretation Construction Approach to Constructivist Design, in B. Wilson (Ed.) Constructivist Learning Environments.

8 Guidorzi R., (2003). e-Learning Projects at Bologna University: an Overview, Proceedings of the International Conference on Networked e-Learning for European Universities.

9 Guidorzi R., Diversi R., Colin M., Lodoli G., (2006). A Constructivist Approach in Designign an e-Learning System Identification Course, Preprints of the 7th IFAC Symposium on Advances in Control Education.

10 Gamma E., Beck K., (2003). Contributing to Eclipse: Principles, Patterns, and Plug-Ins, Addison-Wesley.

11 OSGi Alliance, (2005). OSGi Service Platform specifications, Release 4. Available from http://osgi.org.

12 McAffer J., Lemieux J. M., (2005). Eclipse Rich Client Platform : Designing, Coding, and Packaging Java Application, Addison-Wesley Professional.

13 Moore B., Dean D., Gerber A., Wagenknecht G., Vanderrheyden P., (2004). Eclipse Development Using the Graphical Editing Framework and the Eclipse Modeling Framework, IBM RedBooks.

14 Peh D., Hannemann A., Hague N., (2006). BIRT: A Field Guide to Reporting, Addison-Wesley Professional.

15 Ganymede plug-in: http://sourceforge.net/projects/ganymede/.

16 BPEL project: http://www.eclipse.org/bpel/index.php.

17 Eclipse Italian Community: http://www.dis.unina.it/ECLIPSE/index.php?option=com_content&task=view&id=12&Itemid=28&lang=en.

18 Vicari D., (2004). AlmaChannel Project, Citam internal report, University of Bologna.

19 AlmaChannel portal: https://www.almachannel.unibo.it/portale/index.htm.

20 Dr.Java plug-in: http://drjava.sourceforge.net/.

21 Advanced Distributed Learning (2006). SCORM 2004 3rd Edition. Available from http://www.adl.gov.