

Reactive Learning Objects for Distributed e-Learning Environments

Patrick Duval, Agathe Merceron, Michel Scholl

▶ **To cite this version:**

Patrick Duval, Agathe Merceron, Michel Scholl. Reactive Learning Objects for Distributed e-Learning Environments. 1st International Kaleidoscope Learning Grid SIG Workshop on Distributed e-Learning Environments, 2005, Vico Equense (Naples), Italy. pp.4. hal-00190586

HAL Id: hal-00190586

https://telearn.hal.science/hal-00190586

Submitted on 23 Nov 2007

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Reactive Learning Objects for Distributed e-Learning Environments

Patrick Duval, Agathe Merceron, Michel Scholl
Computer Science Department
Engineering School Leonard de Vinci
F-92916 Paris La Défense – Cedex (France)
patrick.duval@devinci.fr, agathe.merceron@devinci.fr, michel.scholl@devinci.fr

We present a concept of reactive learning objects that goes away from the hydraulic view of e-learning and gives to students activity a central place. First experimentations suggest that this concept should be enlarged to include distributed computation, distributed storage and Web services.

E-learning, Learning Object, Service.

1. INTRODUCTION

New technologies bring challenges to teaching and learning. As recalled in the LeGE-WG project [4], high drop-out rates are common for e-Learning. There is an agreement that e-Learning should go away from the hydraulic view of information transfer from the teacher to the learner and propose radically different scenarios that give to learners' activity a central place. One approach is the development of collaborative tools to create a community among learners and a space for common activity. There is a vivid research in this area, see for example the special interest group "Computer Support for Collaborative Learning" of the Kaleidoscope project, [3].
Another approach proposed in [1,2] is to make learning objects 'reactive' along two lines. The first one concerns students. Learning objects contain exercises that students have to solve for which they receive immediate feedback. The structure of exercises with immediate feedback proposed in many e-Learning platforms is currently too simple. They are either multiple choice exercises, or click and drop exercises, or fill in exercises where what has to be filled in is poor, usually a single word. However, for more complex exercises such as software programming exercises where a substantial part of a program has to be entered, to check that the answer given by a student is correct cannot be handled inside the learning object itself. It is necessary to empower learning objects with the ability of calling plug-ins to assess students' answers. These plug-ins may in turn call other services, such as a Java compiler or an SQL server.
The second line along which learning objects can be made more reactive concerns teachers/tutors. After having processed answers, in current learning objects, a score table is updated. However, this score table contains only marks, or statistics on success, failures and so on. It does not contain students' answers, including mistakes and thus the pedagogical information that can be conveyed to teachers is limited. It is necessary to return more pedagogical information to teachers about the work done by students, store in a database not only whether students have solved successfully exercises or not, but also (parts of) the answers they have submitted along with error messages that plug-ins may have produced. The learning objects introduced in [2] have this capability.
The aim of this paper is (i) to show that these reactive learning objects could benefit from distributed and service oriented architectures and (ii) to contribute to the emergence of novel scenarios for distributed e-Learning.
First we present the architecture model of our reactive learning object. Then we show how these reactive learning objects support both learners' and tutors/teachers' activity. We present limits of our architecture and discuss these limits in the light of distributed and service oriented architectures.

2. ARCHITECTURE MODEL

The layer responsible for managing the student navigation through learning materials is called the E-learning Guided Tour (EGT). This layer provides guidance and constraints to students in their training activity (and play the role of a top-level Controller in the Model View Controller model terminology). Each time a student requests a training exercise, or submits some response, the EGT delegates further processing to a Learning Object, forwarding to it the exercise identification, the student identification, and submitted data (figure 1).
Such a Learning Object (LO) is activated by the EGT, and responsible for (i) providing the learner with a given exercise (ii) managing the training process regarding an exercise description, providing feedback and a final score to the learner, (iii) logging significant training events for subsequent tutors and authors mining's, (iv) returning an

execution status to the EGT, indicating success or failure. It is up to the EGT to decide what exercises or course materials may be relevant to present next to the student.

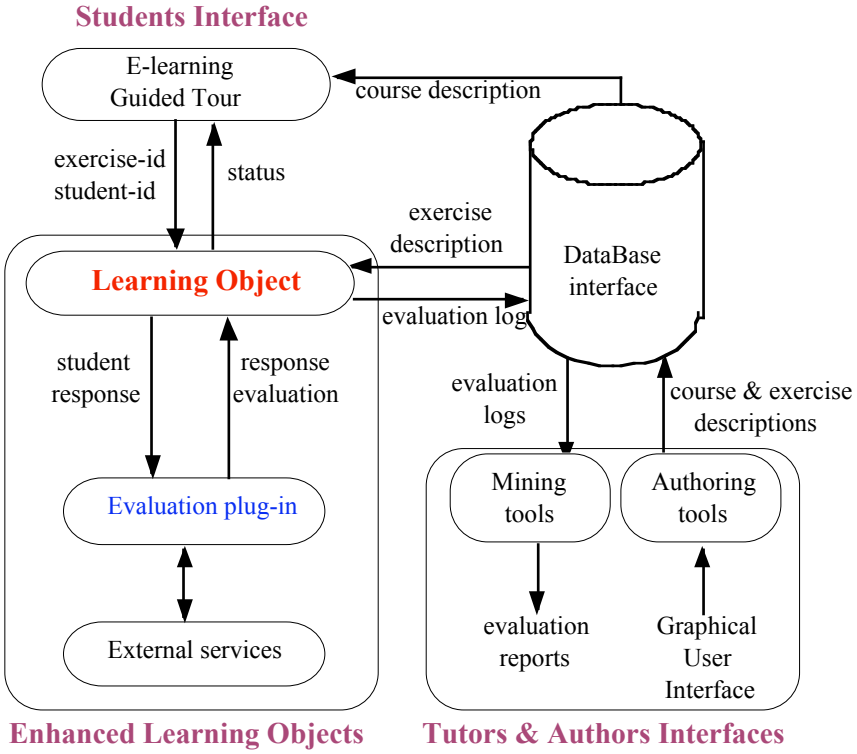


FIGURE 1: Learning Object Model

3. SUPPORTING LEARNERS' ACTIVITY

Two prototype learning objects following this model have been implemented and used by students. One protoype is an SQL course, the other one is an introductory course on Java. We illustrate how these 'reactive' learning objects give to students' activity a central place with the latter course.

The call to a plug-in and to further services is transparent to the learner. In the case of the Java course for example, students do not need to learn anything about compiler or Java virtual machine. Managing the learning platform is enough. However, learners do receive immediate feedback to exercises. Figure 2 shows a screenshot taken while a student attempts an exercise on arrays. The aim of this exercise (text not visible on the screenshot) is twofold: to practice array declaration and initialization and to reinforce parameter passing. In her answer, this student does not practice these skills. Instead, she just prints the expected result from the tester class. The feedback is a correct compilation, but a wrong code with the message “the result is not the expected one”. Note also exercises status is displayed to students. In the present case, exercise 1 has been successfully completed, other exercises have not been done yet. Storing complete answers and mistakes makes it possible for learners to look at their entire history. For example, a student can look back at the mistakes made while completing a particular exercise.

4. SUPPORTING TUTORS' ACTIVITY

Comprehensive data is stored to allow tutors to track students' activity while they progress through the course material. This data provides several views of progress to tutors: An overall class view, a student view, and an exercise view. These views can be displayed under the form of tables or graphs.

As an example, Figure 3 shows the view for chapter 'while loop' done by student Bob Smith under the graph form. This chapter has 7 exercises. For each exercise, the first column shows how many times it has been read, the second column how many mistakes have been made and the third one shows whether it has been terminated successfully. The two thicker columns in the background show left the total time spent and right the total time spent till success. Bob has read the first exercise three times, has made two mistakes on it before getting it right and it took him less than two minutes. Looking at the number of mistakes, it has not been easy for Bob to get exercise 7 correct. If the tutor wishes, she can consult the history to see all answers. In the present case, the tutor could see all answers and all mistakes Bob has made before getting exercise 7 correct. Note that time is indicative only, it is not easy to say whether a student is working or dreaming while in front of the computer.

5. FIRST EXPERIENCE AND LIMITS

The SQL course has been used as an extra resource by students taking a face to face database course. The Java course has been taken both as a distance course and an extra resource to a face to face course. Students do

appreciate this way of learning by doing, which goes away from the traditional information transfer model. They also appreciate receiving immediate feedback, which allows them to practice at their own pace since they don't have to wait for the feedback of a human scorer. However, these first experiments point out limits of the architecture.

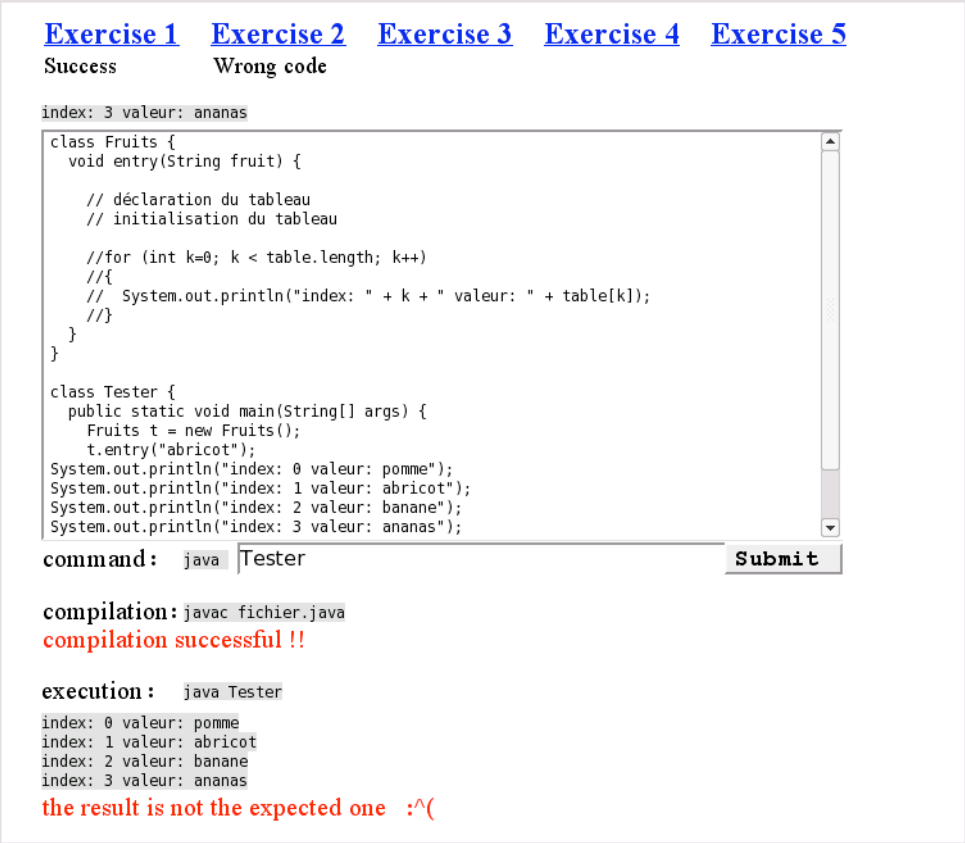


FIGURE 2: Attempting a Java exercise.

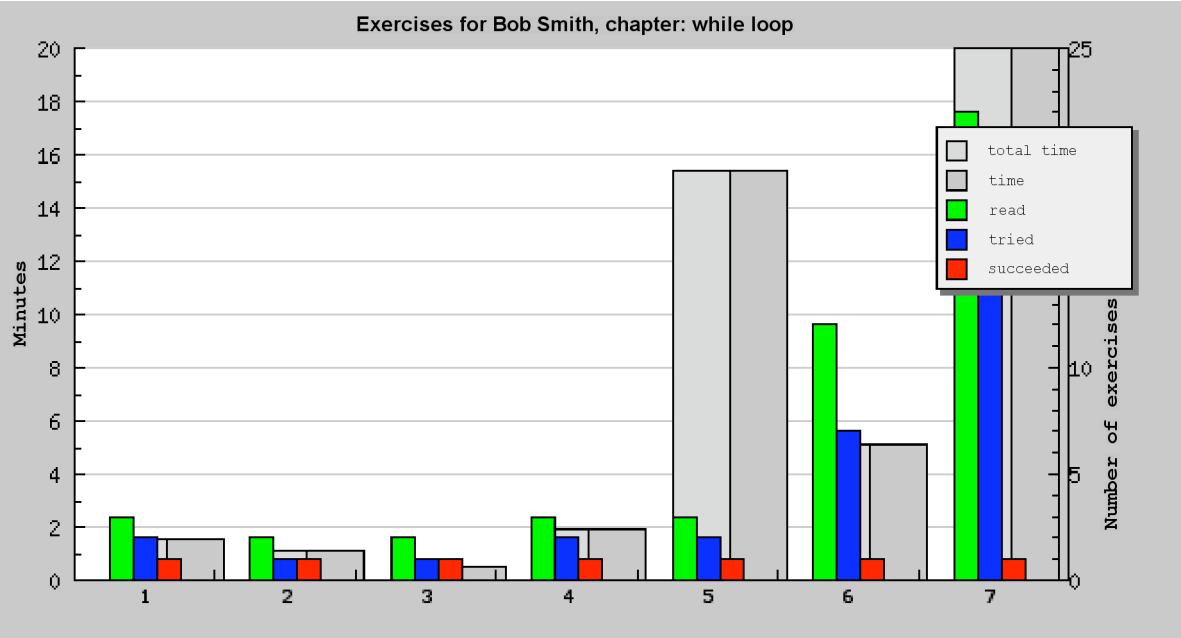


FIGURE 3: While loop chapter view for Bob Smith.

A first limit, especially for the Java course, is the size of the database. Indeed, storing students' answers mean storing programming code. With 30 students, the database grew to 10 mega-bytes. With bigger numbers of students and a complete course -the present prototype does not cover all chapters- the size of the database could become a problem.

A second limit of the present architecture model is that it is not distributed. Though this did not happened in our experiment, if many students hit the submit button at the same time, the server could be overloaded.

A third limit is that there is no specific support in our model for collaborative work. In the forum of the platform that hosted the prototype, there was a dynamic discussion of students sharing their programming problems and solutions. This suggests that collaborative work on bigger projects requiring each learner to do a specific part would be a good complement to the present approach where students solve exercises individually.

6. TOWARDS A SCENARIO FOR DISTRIBUTED E-LEARNING ENVIRONMENTS

Distributed storage and computing, and Web services come could come into play in several places for our learning object model.
First distributed storage could be necessary when the database of students gets too large.
Second, to check students' answers the evaluator plug-in could call various Web services. In the present case of our two implementations, the evaluator of the Java course calls the service of a Java compiler and a Java virtual machine and the evaluator of the SQL course calls the service of a MySQL server. However, this is not implemented as Web service calls in the sense of [6]. Changing the architecture to allow the call of a Web service would help to cope with the overloading problem of the server mentioned earlier. Further, one single course containing several kinds of exercises and tests, could call several kinds of Web services to check students' answers, not just one or two as it is presently the case.
Third, as suggested in [5], one could integrate in our learning objects tests offered as Web services, or make our whole learning object accessible as a Web service. This leads to the concept of a Web service relying itself on other Web services.

7. RELATED WORK

Numerous papers report experiments on interactive environments for teaching programming languages. As examples of such environments, Ceilidh [7] is used for automatic assessment of homework assignments on ML programs, [8] describes a system for automatic analysis of Scheme programming assignments, ELP [9] is an interactive environment for teaching Java to first year students. Only a few systems exist which provide an automatic analysis of SQL answers, for example [10,11]. Our approach shares with some of the aforementioned ones the goal of finding and understanding the errors of the students, in particular for a better tuning of the e-learning tools to the student population and to the teaching objective. However, our approach focuses on a more comprehensive collection of data related to the student answer and suggests a different architecture relying on distribution of e-learning material and services.

REFERENCES.

[1] Duval P., Merceron A., Rinderknecht C. and Scholl M. “LeVinQam: A Question Answering Mining platform”. ITHET04, Proceedings of the 5th International Conference on Information Technology Based Higher Education and Training, Turkey. IEEE Press. 250-255, 2004
[2] Duval P., Merceron A., Scholl M. and Wargon L. “Empowering Learning Objects: an experiment with the ganesha platform”, ED-MEDIA 2005, Montreal, Quebec, June 27-July1, 2005.
[3] Kaleidoscope.Computer Support for Collaborative Learning.http://wwwkaleidoscope.imag.fr/pub/cscl/index.html 2004
[4] Le-GE-WG www.lege-wg.org/ 2004.
[5] Reklaitis V., Baniulis K., and Aukstakalnis N., “Building Assessment Web Services from Question Type learning Objects”, 4th International LeGE-WG Workshop - Towards a European Learning Grid Infrastructure: Progressing with a European Learning Grid. Stuttgart, Germany. 27 - 28 April 2004.
http://ewic.bcs.org/conferences/2004/4thlege/session2/paper3.htm
[6] Web services. http://www.w3.org/2002/ws/
[7] S. Foubister, G. Michaelson, and N. Tomes, “Automatic assessment of elementary standard ML programs using Ceilidh”, Journal of Computer assisted Learning, 1996
[8] C. Beierle, M. Kulas, and M. Widera, “Automatic analysis of programming assignments“, Proc. Of DeLFI 2003, Köllen Verlag, Bonn, 2003
[9] N. Truong, P. Bancroft, and P. Roe, “A Web based Environment for learning to program”, ACSC2003, Adelaide, Australia, 2003
[10] D. Radosav, Z. Kazi, L. Kazi, « SQL E-learning System », http://www.timsoft.ro/ejournal/article-radosav.htm, 2005
[11] G. Russell, A. Cumming, “Improving the student Learning experience for SQL using Automatic Marking“, http://www.soc.napier.ac.uk/publication/op/getpublication/publicationid/7436710, 2005