Supporting PoEML Educational Processes in Moodle: a Middleware Approach

Roberto Pérez-Rodríguez, Manuel Caeiro-Rodríguez, Luis Anido-Rifón

Department of Telematic Engineering, University of Vigo
rperez@gist.det.uvigo.es
{Manuel.Caeiro, Luis.Anido}@det.uvigo.es

Abstract. Integrating the PoEML specification into Moodle, an open-source Learning Management System (LMS), is more than a technology-related question. It has got pedagogical and philosophical implications and also usability issues to be aware of. This paper documents the study of that integration and discusses the suitability of this approach. PoEML educational processes are suitable for collaborative process-based learning. Moodle is based on a content delivery strategy and offers no support for process-based collaborative learning. In our solution we make use of the open source workflow engine JBpm in order to support process specifications in Moodle. It is worth noting that the functionality provided by this solution is beyond the current state-of-the-art of Moodle development.

Keywords. Moodle, PoEML, JBpm, Web Services

1 Introduction

The latest e-learning standards address the necessity for process-aware tools that support the execution of educational processes. Learning Management Systems (LMSs) must be able to execute educational processes to support process-based collaborative learning. Educational Modelling Languages (EMLs) were proposed as a means to model the complex interactions and relationships that arise in e-learning environments. The main idea underlying EMLs is to provide a single modelling solution rather than attempting to capture each pedagogy in a different language. If several languages are considered, each one requires specialized implementations of both design and runtime systems. Using an EML, a single set of design and runtime tools is enough to support the desired wide range of pedagogies. One of the main purposes of EMLs is to design Units Of Learning (UOLs) and to promote interoperability between different run-time environments. This corresponds to the principle of *create once, use everywhere*. The Learning Design [1] specification is the *de facto* standard EML. Nevertheless, LD has serious deficiencies dealing with process-based collaborative learning. To overcome these deficiencies a new EML based on the separation of concerns approach was proposed: Perspective-oriented EML (PoEML) [2]. In this work we have adopted a subset of the complete PoEML specification. Specifically, we have considered the Structural, Order and Data perspectives. Although the complete specification is much more rich it is worth noting that this work supposes a qualitative jump from the current content-delivery Moodle philosophy.

We consider a basic collaborative process as an example. In our example there are an ordered sequence of activities and two participants: a learner and a teacher. When a person accesses to the system in one of these two roles, she/he is provided with a list of activities to perform. Such a list is updated automatically in accordance with the realization of activities by the other participants. Activities can be arranged in a sequence, but also in parallel and in accordance with pre-defined conditions. This functionality is provided by integrating the current Moodle system with a well-known workflow engine.

This paper is organized as follows. Next section introduces PoEML and Moodle [3] and then section 3 exposes the reasons and motivation of this work. Section 4 explains the mappings between the Moodle course structure and workflow engine processes and tasks and section 5 presents the system architecture. In section 6 we expose some considerations about the flexibility of the final system. Finally, section 7 outlines our conclusions and future work proposals.

2 PoEML and Moodle

PoEML (Perspective-oriented EML) was designed following a strategy of separation of concerns. Instead of trying to solve the modelling problem as a whole, its division in several separated parts was considered. As a result, the modelling of each part may be solved "almost" independently (however, there are certain dependencies among perspectives that cannot be solved completely). The separation of concerns approach reduces the complexity of modelling Educational Scenarios (the equivalent to LD Units Of Learning). With this approach the design of authoring tools and run-time learning environments can be done in an incremental way. Due to the extraordinary complexity of the complete PoEML specification we have considered three of the thirteen perspectives that compose the complete PoEML proposal. These three perspectives correspond to the core of an LMS implementation: they are the Data, Structural and Order perspectives. The Data perspective deals with the variables that are part of the Educational Scenarios and the data flow among them. The Structural perspective models the composition of Educational Scenarios and supports nesting. Finally, in the Order perspective is contained the information necessary to sequence the Educational Scenarios.

Moodle is the most representative and used of the open-source Learning Management Systems. Moodle philosophy is based on constructivism in pedagogy and emphasize that learners (and not just teachers) can contribute to the educational experience in many ways. Moodle's features reflect this in various design aspects, such as making it possible for students to comment entries in a database (or even to contribute entries themselves), or to work collaboratively in a wiki. We have to note that collaboration is made at the tool level: Moodle lacks a method to develop process-based collaborative learning. Following the Learning Design terminology we can say that the collaboration is made at the Learning Content Management System level, but not at the Learning Management System level. There is not any engine that supports the execution of learning processes into Moodle. The word *Moodle* is actually an acronym for *Modular Object-Oriented Dynamic Learning Environment*. Moodle can also be considered a verb, which describes the improvisational process of doing things as it occurs to you to do them, an enjoyable tinkering that often leads to insight and creativity. As such it applies both to the way Moodle was developed, and to the way a student or teacher might approach studying or teaching an online course.

3 Problem Definition and Motivation

Moodle is one of the most used open-source e-learning platforms. Its approach is content-based, single-learner, self-paced lessons. On the other hand, the PoEML approach is process-based, multi-learner, programmed lessons. PoEML does not exclude a single-learner accessing contents in a self-paced way. With PoEML we can model a broad range of lessons: from single-learner to multi-learner, from self-paced lessons to lessons with a very rigid design.

Moodle supports collaborative learning based on **groupware** (forum, chat, wiki), but it lacks a mature approach to **process-driven collaborative work**. The inclusion of a new groupware tool into Moodle does not mean an internal restructuring of the system: we only have to add a new module. But the inclusion of a new kind of Moodle course based on collaborative multi-learner process-driven work is a major challenge in Moodle development. We have to add a process execution engine to the current Moodle features. The minimal functionality we want to reach is:

- Support for activity sequencing
- Support for multi-learner educative processes

So, in resume, our aim is to design a technical infrastructure that supports the execution of process-driven collaborative courses in Moodle: PoEML courses.

4 Mapping courses to processes

The first question we have to response is: **what do we want to sequence?** In PoEML terminology we have to identify an atomic Educational Scenario (this concept is similar to the Unit Of Learning in the Learning Design standard). We must focus our attention in the description of courses in Moodle, a course is composed by sections and resources/activities. The main difference between sections and resources/activities is that a section has a serial number into the course structure at run-time whist resources/activities are dependent of the run-time identifiers in the database. From the point of view of an structured approach we can say that a Moodle course is composed by sections and that resources/activities are hot-linked to sections. This means that it is possible to do sequences of sections but it is very difficult to try to construct sequences of resources/activities by its database-dependent nature. Sequencing of resources/activities supposes a major change in the current Moodle course structure.

Therefore, our first approach is to establish a mapping between the Moodle course structure and a process definition in JPDL. The link between the two is in the Moodle side the **section** and in the JBpm side the **task**. So we make an univocal relationship between those core elements. A section of a Moodle course corresponds to a JBpm process task.

We also have to map courses to processes. This is done as follows: we map a **course** to a **process instance**. There can be a lot of different approaches here, but we consider our main concern the process-based collaborative work, so we map a course to a process instance. Another approach could be to map the pair user-course to a process instance. The latest approach could be more suitable for a single-learner style.

5 System Architecture

The system architecture is shown in Figure 1. In order to implement the PoEML functionalities into Moodle we make use of a workflow engine as middleware. Our first task is focused on finding a suitable workflow engine to execute the learning processes. After having studied a few of them (e.g. Galaxia Workflow Engine) we have chosen JBpm [4] from the JBoss project. As the chosen workflow engine, JBpm, is written in java we cannot invoke directly its methods from the Moodle PHP code. To overcome that limitation, we have designed an architecture based on the Remote Procedure Call (RPC) mechanism.

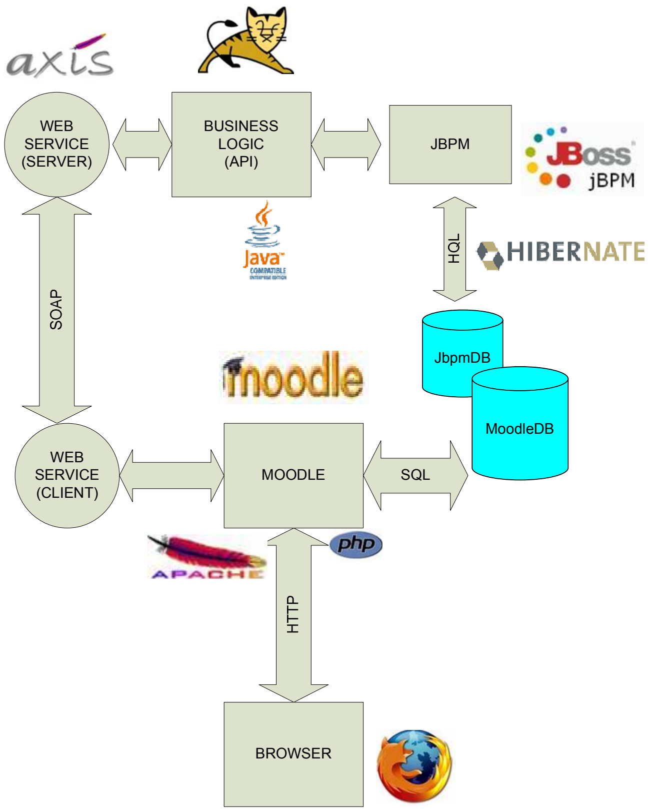


Figure 1. System architecture

JBpm needs to be executed into a servlet container. We have chosen Tomcat [5] because it is more light-weight than the complete application server: JBoss. We have designed a WebApp called BussinessLogicAPI with the methods we need to deploy the process-definition, to get the work-to-do list and to signal the end of an activity.

To be able to invoke the remote methods of the BussinessLogicAPI from the Moodle system it is necessary to publish the methods using some RPC technology. We have chosen Web Service technology because we only need some textual messages to accomplish the work. We publish the methods using Apache Axis [6] and we consume the Web Services from Moodle using NuSOAP [7].

5.1 The system at build-time

The build-time is about the actions that have to be performed to prepare the course, before the participants (learners and teachers) can access to the system. The main task we have to face at build-time is to deploy the course specification into JBpm and

Moodle. Figure 2 shows the procedure to do this. The method of the BusinessLogicAPI `deployProcessDefinition()` receives the course specification in PoEML format and transforms it into two separate archives: the JPDL archive, with the sequencing information to be deployed into JBPM, and the archive with the course specification in Moodle Backup Format ready to be deployed into Moodle. The main idea behind this deploy mechanism is the decomposition of one specification into two specifications dealing each one with separate concerns. While the sequencing information is deployed into JBPM, the content of the course sections is deployed into Moodle.

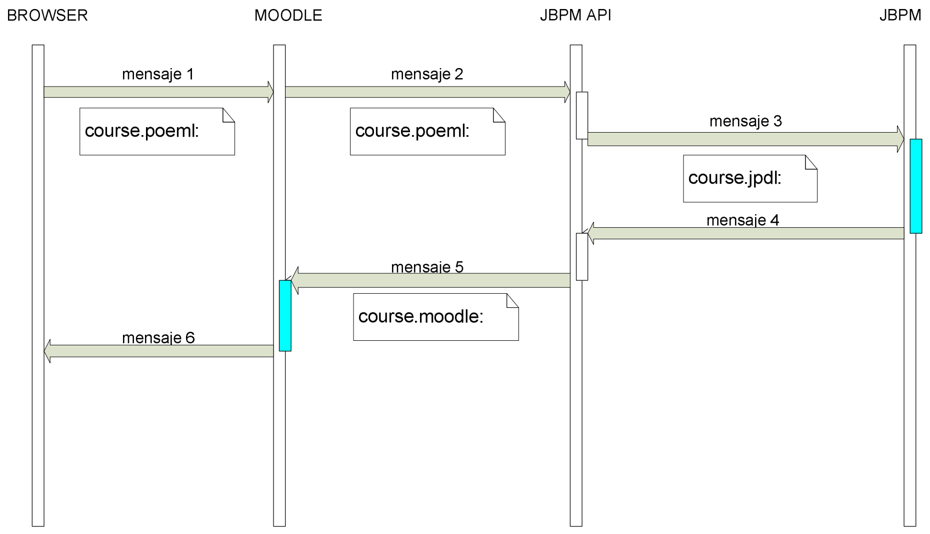


Figure 2. Deployment of the PoEML course

5.2 The system at run-time

At run-time the system behaviour is very simple and straightforward. When an user logs into Moodle and navigates to a PoEML course the system displays the work-to-do list for that user, in accordance with her/his role. As it was said at the mapping section, this work-to-do list is a list of course sections. To do it, anytime the user navigates to the PoEML course home page, Moodle performs a request of the work-to-do list (of sections) for that user in the course. It corresponds with the work-to-do list (of tasks) of a JBpm user in a certain process instance.

Figure 3 shows the actions that conform the basic interaction of an user with the system.

- **Default course view**. The course page view must be the pending tasks for the user currently logged in.

- **Activity view**. The visualization of the page of an Educational Scenario. At the basic level this would be a text resource.

- **End of activity signalling.** The confirmation of an user having finished the current Educational Scenario. At this point we must to signal the end of the activity to the workflow engine. Following this signalling the workflow engine evolutes to the next state. As a result the systems presents the user the course page refreshed to the new state of the pending tasks of the user.

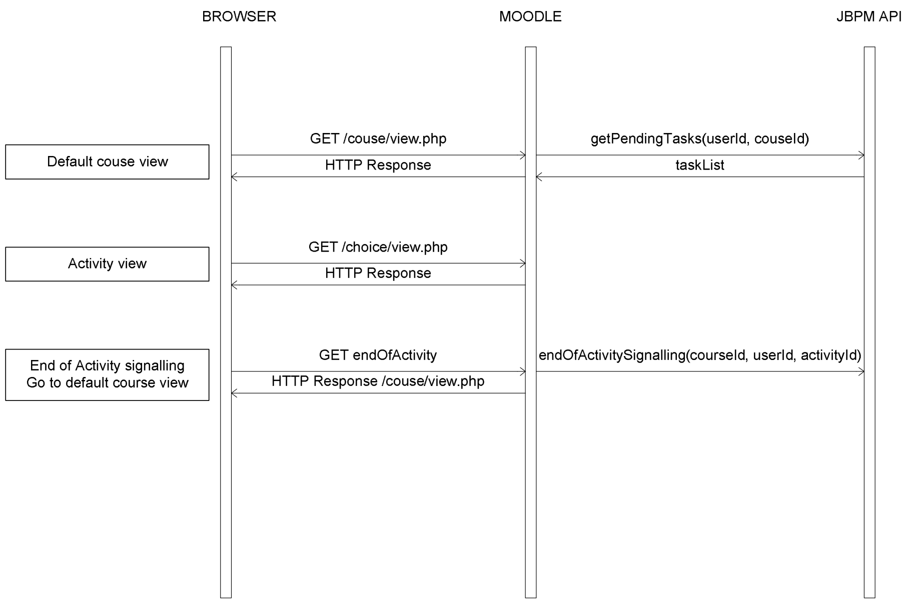
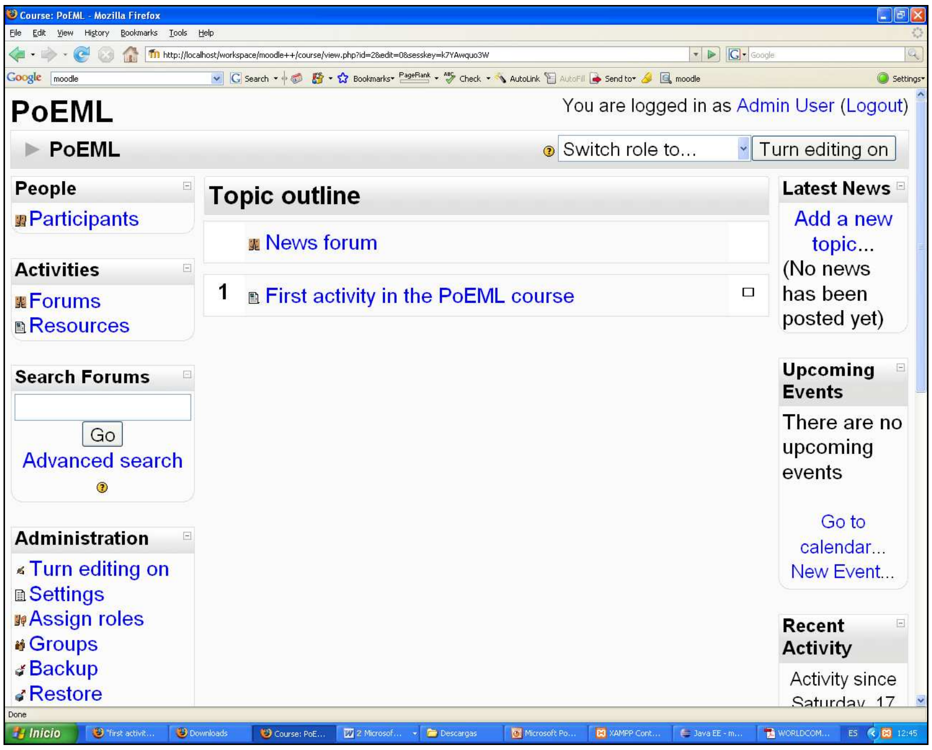

BROWSER
MOODLE
JBPM API
Default couse view
GET /couse/view.php
HTTP Response
getPendingTasks(userId, couseId)
taskList
Activity view
GET /choice/view.php
HTTP Response
End of Activity signalling
Go to default course view
GET endOfActivity
HTTP Response /couse/view.php
endOfActivitySignalling(courseId, userId, activityId)

Figure 3. Default course view, activity view and end-of-activity signalling

In the **Default course view** when the user sends the course visualization request the Moodle system invokes the `getPendingTasks()` method of the BusinessLogicAPI. The system only shows the links to the user's pending tasks in the current course.

When the user clicks at the activity link (**Activity view**) Moodle presents the activity page. It is not necessary the participation of the workflow engine to accomplish that.

When the user finishes the activity throws the **End of activity signalling**. Moodle communicates the end of the activity to the workflow engine.


Course: PoEML - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://localhost/workspace/moodle++/course/view.php?id=2&edit=0&sesskey=k7YAwquo3W

PoEML

You are logged in as Admin User (Logout)

▶ PoEML

Switch role to... | Turn editing on

People
Participants

Activities
Forums
Resources

Search Forums
Go
Advanced search

Administration
Turn editing on
Settings
Assign roles
Groups
Backup
Restore

Topic outline

News forum

1 First activity in the PoEML course

Latest News
Add a new topic...
(No news has been posted yet)

Upcoming Events
There are no upcoming events

Go to calendar...
New Event...

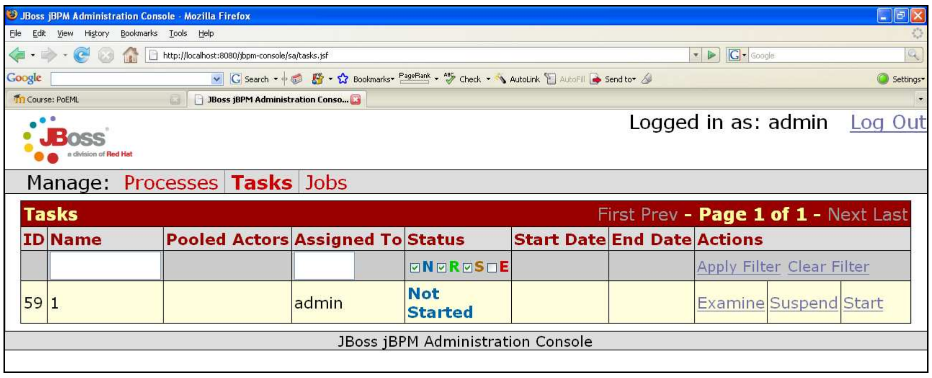
Recent Activity
Activity since Saturday, 17

Done

Figure 4. PoEML course home page

Both Figure 4 and Figure 5 correspond with the same work-to-do list. Figure 4 is the default course view with the pending sections for the user ‘admin’ in the course ‘PoEML’. Figure 5 is the view of the pending tasks from the JBpm administration console. We can see that in both cases the user ‘admin’ has to do the section/task ‘1’. When the user completes the section 1, in both cases the work to do list will be section/task ‘2’ (if the user ‘admin’ has the two roles staff/learner). This behaviour is obtained in accordance with our example expressed in JPDL (the JBpm process definition language):

```
<process-definition name="poeml">
        <swimlane name="alumno">
                <assignment class="es.teleco.jbpmapi.assignments.Alumno"/>
        </swimlane>
        <swimlane name="profesor">
                <assignment class="es.teleco.jbpmapi.assignments.Profesor"/>
        </swimlane>
        <start-state name="start">
                <transition name="" to="1" />
        </start-state>
        <task-node name="1">
                <task name="1" swimlane="alumno" />
                <transition name="" to="2" />
        </task-node>
        <task-node name="2">
                <task name="2" swimlane="profesor" />
                <transition name="" to="1" />
        </task-node>
        <end-state name="end" />
</process-definition>
```


JBoss jBPM Administration Console - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:8080/jbpm-console/sa/tasks.jsf
Course: PoEML
JBoss jBPM Administration Conso...
JBoss a division of Red Hat

Logged in as: admin Log Out

Manage: Processes Tasks Jobs

Tasks — First Prev - Page 1 of 1 - Next Last

ID	Name	Pooled Actors	Assigned To	Status	Start Date	End Date	Actions
				☑N ☑R ☑S ☐E			Apply Filter Clear Filter
59	1		admin	Not Started			Examine Suspend Start

JBoss jBPM Administration Console

Figure 5. Work-to-do-list in JBpm

6 Some questions about flexibility

In our approach, the JBpm administration console works as a course manager. This means that it is possible to signal tasks, to delete course instances, etc. from that user interface. These issues have obvious positive implications on the flexibility of the final system. We have to remark an interesting characteristic of our system: it is possible to change at run-time the resource associated with a certain section. This is a very noticeable and positive characteristic regarding flexibility.

These flexibility characteristics are direct consequences of the separation-of-concerns approach.

7 Conclusions and future work

This paper introduces a software development project aiming to extend the current Moodle system to support process-based collaborative learning units. A main conclusion of this project is that it is not realistic to try to implement the full PoEML specification in Moodle due to its lack of an process-aware approach. There are issues like scalability, adaptation and flexibility that imply a complete refactoring of Moodle. Additionally there are questions regarding usability that must be faced in order to reach a stable/production system, as the response of the system to the navigation go-back and go-forward buttons. There must be an adequate control of access to contents based in security checks against the workflow engine. In spite of these considerations, we introduce the system exposed in this paper as the basic proof of concept of an integration of PoEML sequencing capabilities into Moodle.

It is interesting to mention that there have been other projects aiming to support process-based collaborative learning in Moodle. In concrete it is remarkable the LAMS system, which offers an integration solution based on offering a LAMS sequences as a Moodle activity. Our approach is rather different from the LAMS one, because we are sequencing directly functional components of a Moodle course: sections.

We are currently working on developing a PoEML-based execution engine from scratch that will be the core of a complete PoEML-based Learning Management System. We are studying the state-of-the art of workflow management systems with the aim of collecting some ideas about designing and implementing a perspective-oriented execution engine. We hope that the great modularity of the PoEML specification will be an aid to develop an execution engine incrementally perspective by perspective.

Acknowledgements

We thank *"Ministerio de Educación y Ciencia"* for its support under project *"Servicios Adaptativos para E-learning basados en estándares"* (TIN2007-68125-C02-02), and "Consellería de Innovación e Industria" under project *"Diseño y desarrollo de un marco semántico para el modelado de servicios en la administración pública. Aplicación a la provisión de servicios frente al ciudadano."* (PGIDIT06PXIB322285PR).

References

1. R. Koper, B. Olivier, and T. Anderson, editors. *IMS Learning Design Information Model.* IMS Global Learning Consortium, Inc., 2003
2. Manuel Caeiro Rodríguez (2007): *Contribuciones a los Lenguajes de Modelado Educativo*. Tesis doctoral. Universidad de Vigo.
3. Moodle (2008): http://moodle.org
4. JBpm, http://www.jboss.com/products/jbpm
5. Tomcat, http://tomcat.apache.org
6. Apache Axis, http://ws.apache.org/axis
7. NuSoap, http://sourceforge.net/projects/nusoap