Activity Instantiation Model for e-Learning Engines

Jorge Torres
Distributed and Adaptive Systems Lab for Learning Technologies.
Tecnológico de Monterrey, México

Eduardo Juárez
Distributed and Adaptive Systems Lab for Learning Technologies.
Tecnológico de Monterrey, México

Jesús Reséndiz
Distributed and Adaptive Systems Lab for Learning Technologies.
Tecnológico de Monterrey, México

ABSTRACT
In Learning Process Engines, a main concern is to deliver rich pedagogical learning scenarios. However current e-learning systems are not completely capable of handling complex role structures and learning activity flows. To solve this issue, an activity instantiation model based on role access control is proposed. This model is capable of ensuring the completeness of the learning flow for each learner, and it allows the learner and the professor to know and maintain control of her learning process and also to participate with other learners in team-based tasks.

General Terms
Activity Instantiation Model, Learning Process Engines.

Keywords
Learning Process Engines; Learning Systems Platforms and Architectures; LPCEL; Activity Instantiation Model.

1. INTRODUCTION
E-Learning systems evolution [1] has brought an enhanced set of characteristics and flexibility to the learning process, moving from simple static web contents, to the creation of Educational Modeling Languages (EML) able to fully describe learning scenarios, such as IMS LD [2] which are later executed by a process engine like CopperCore for IMS LD.

Originally, EMLs were designed to describe E-Learning Systems that had all the resources, the learning scenario will need, in a self-contained mode. However, it was found that such systems were unable to provide a rich and diverse pedagogical experience for the learner, due to the lack of expressiveness of EMLs and their engines, in terms of pedagogical diversity and learning flow description [3].

To aboard these issues, the Learning Process Composition and Execution Engine (LPCEL) was proposed from the learning scenario design perspective [4]. From the architectural perspective, the Web Applications and Services Enhanced Learning Architecture (WASEL) [1] was proposed in order to integrate learning web services. To complete the vision of this new generation of e-learning systems, efforts are focused on the design and construction of Learning Process Engines (LPE).

A main concern in such types of engines is to deliver rich pedagogical learning scenarios able to handle complex role structures and complex learning activity flows. Sometimes in collaborative work, learners or even professors (if there are more than one), may perform diverse roles, which provide access to different activities, e.g. only the team leader can do the final delivery of a project, the project manager is in charge of the work plan, the student with the role quality assurance will design the user acceptance tests for the project, the professor with more modeling experience will evaluate modeling projects, among others situations.

The objective of this paper is to provide a model capable of solving these issues. The rest of this paper is structured as follows: section 2 presents our vision of e-learning systems, section 3 describes briefly the role-based access control model and current instantiation models, section 4 formally presents our Activity Instantiation Model for Learning Process Engines, section 5 shows a worked example of the model, to end with some conclusions.

2. A NEW GENERATION OF E-LEARNING SYSTEMS
Our vision for new e-learning systems is based on Learning Web Services (LWS), which need to be integrated and executed by the learners in a pedagogical sense, i.e. participants may perform learning activities by consuming many different web services, and is achieved by the means of a service-oriented-architecture (SOA) paradigm [5].

Inside a distributed learning environment, there is a great variety of resources and services that learners can use to achieve their learning objectives. Some of these services may be recovered and deployed locally; others may be executed in a distributed way, allowing the integration of new resources and services to the learning process. Today, the only EML offering a framework for the integration of web services within a distributed environment is LPCEL, and it is also the most expressive EML to represent complex learning flow structures, not even IMS LD [6].

LPCEL along with the WASEL architecture (see Figure 1) present a SOA based approach able to support complex learning scenarios within a distributed environment. In this proposed architecture, the learning scenario is designed with a Learning Process Editor and executed by a Learning Process Engine, which communicates with a Learning Services Bus where the Learning Web Services are plugged in. User interface must be Web 2.0-enabled and communication is made through an API WS-Access. Interoperability with other Learning Management Systems such as Moodle or LAMS can be achieved by Wrap ad-hocs.

In such architecture, for learning process engines as in Workflow Management Systems for workflow engines [7], the control of process and activity instances, and role management

of participants, are main concerns. In a pedagogically rich environment, LPEs must be able to handle complex role structures and complex learning activity flows, because learners and professors may perform diverse roles in the same learning scenario. Each role provides a different access level to activities based on specific learning objectives. Current e-learning systems lack these kinds of controls. This situation causes confusion to the learners about which activities they need to perform in an individual manner or which activities are going to be performed only by one learner in the team, or by everybody in a collaborative way. In most cases, instances for every activity are created for each learner even if the learner does not perform the activity. In other cases, the e-learning system acts only as a file repository and it does not provide any pedagogical help for the learners to perform learning activities [8].

In most current e-learning systems the learner progress is only tracked by the results of assessment activities, which does not necessarily represent the state of the learning process of a learner.

In order to propose a solution for Learning Process Engines about these issues, as a starting point, we reviewed the Role-Based Access Control Model and instantiation models in current EML engines.

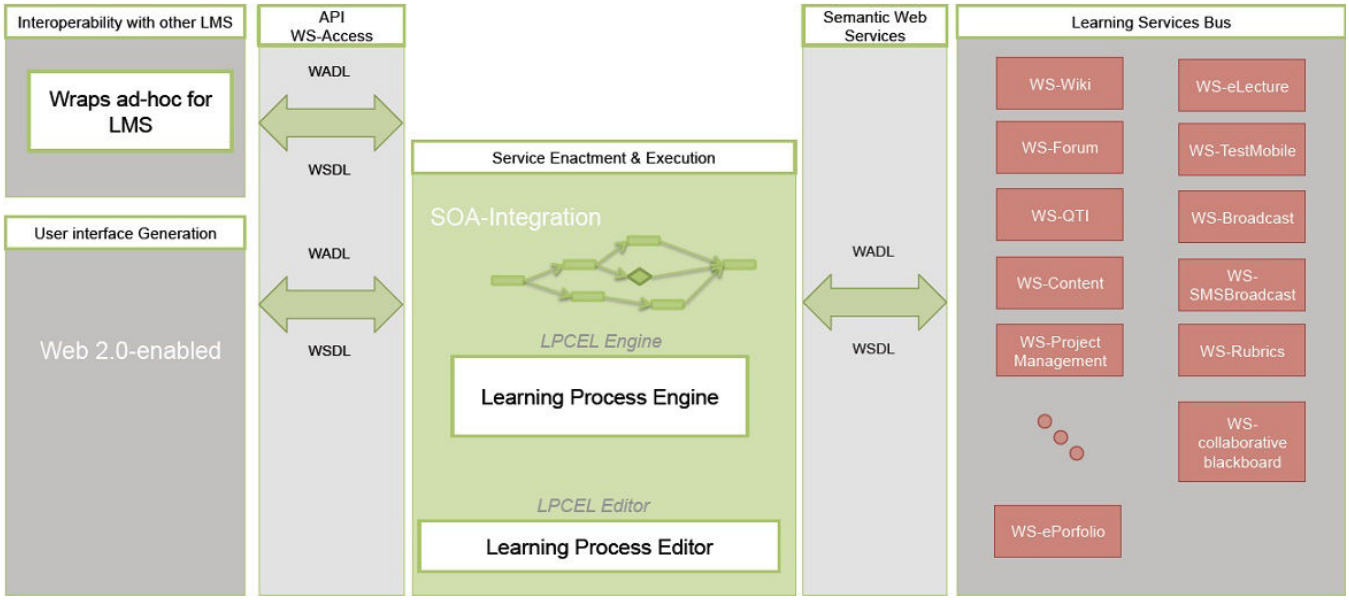

Fig 1: WASEL Architecture [1].

3. CURRENT ACCESS CONTROL AND INSTANTIATION MODELS

3.1 Role-based Access Control

Distributed learning scenarios are multiuser environments with a strong need for access control. The Role-Based Access Control Model (RBAC) [9] provides a framework for authorization management including functions to grant access to resources; also has a strong acceptance in enterprise environments for its capability to support security management and policies. It has been found that is better to specify access control in terms of roles rather than in terms of individuals.

The RBAC model is organized in four levels of capabilities: Core RBAC, Hierarchical RBAC, Static Constrained RBAC and Dynamic Constrained RBAC. For our activity instantiation model, Hierarchical RBAC is our main interest for its capability to represent a hierarchical structure of roles, which is necessary in learning scenarios for collaborative work. The H-RBAC model is depicted in Fig. 2. As it can be seen, the model is composed by the elements Users, Roles, Subjects, Operations, Objects, and Permissions. The Role Hierarchy is established by inheritance relationships between Roles, where a junior role inherits the permissions of its senior role.

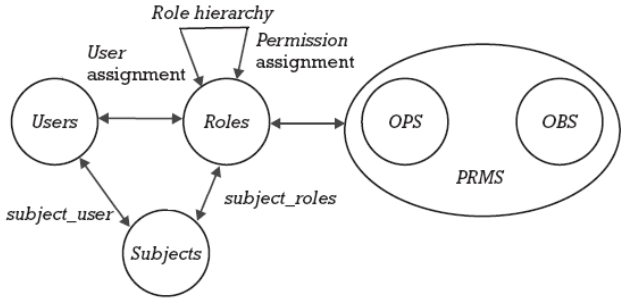

Fig 2: Hierarchical RBAC [9].

3.2 Instantiation Models

In order to execute and reuse a process many times in an execution engine, and to reuse its activities in different points of the process, it is necessary to properly instantiate the process and its activities. To achieve these goals, a model of states is needed to characterize the behavior of the process and its activities and to guarantee a reliable execution of the process.

In Workflow Systems Theory [7], an execution engine is seen as a state transition machine, where processes or activity instances

modify their state as a response to external events, or specific control decisions taken by the engine. There are six basic states for a process instance (see figure 3): (1) initiated: a process instance has been created but it has not fulfilled the conditions to start execution; (2) running: the process instance has started execution; (3) active: one or more of process instances activities have been started; (4) suspended: the process instance is quiescent and no activities are started until it returns to the running state; (5) completed: the process instance has fulfilled the conditions for completion; and (6) terminated: execution of the process instance has been stopped before its normal completion.

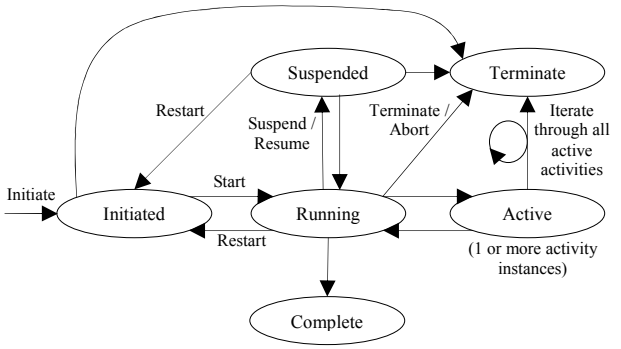

Fig 3: State transitions for process instances.

Not only processes need to be instantiated, but also their activities. In Fig. 4, the four basic state transitions of activity instances are: (1) inactive: the activity instance has been created but has not yet been activated; (2) active: a workitem has been created and assigned to the activity instance for processing; (3) suspended: the activity instance is quiescent and will not be allocated a workitem until returned to the running state; and (4) completed: execution of the activity instance has completed.

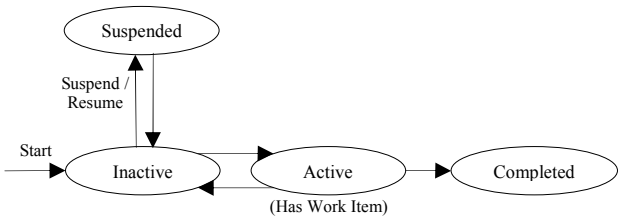

Fig 4: State transitions for activity instances.

4. ACTIVITY INSTANTIATION MODEL

In order to design an activity instantiation model for learning process engines, it is necessary to emphasize in the difference between workflows and learning flows. In workflows it is common for a user to complete a task, but for her it's not relevant the current state of the workflow, because her responsibility is limited to the sub-process where she is the owner; meanwhile the chief owner of the whole process is who cares about everything, but it is uncommon that the chief will be performing a certain task in the workflow. Also, users are limited to perform only a certain type of activity in just one part of the process. In the other hand, in a learning flow, the learner is the owner and is responsible of her entire learning process. Based on the above, the activity instantiation model considers the benefits of workflow systems about monitoring the process state, level of execution of activities based on roles (RBAC), and also the learning process implications to provide pedagogical sense to the execution of activities in a learning scenario.

First, we will define the sets of entities related to the activities that are going to be instantiated, which are the users, the roles, and the teams.

$$US = \{us_1, us_2, \ldots, us_n\}$$

Where *US* is the set named *Users*, and represents subjects with access to the learning process engine. Elements of *US* are defined by the tuple:

$$us = tuple(n, u, p)$$

Where *n* is the real name of the subject, *u* is the username, i.e. the user id in the engine, and *p* is the password of the subject.

$$R = \{r_1, r_2, \ldots, r_n\}$$

Where *R* is the set named *Roles*, and represents the responsibilities of a user in a course. The elements of *R* are defined in design time by the tuple:

$$r = tuple(t, d)$$

Where *t* is the title of the role, and d is the description.

$$T = \{t_1, t_2, \ldots, t_n\}$$

Where *T* is the set named *Teams*, and represents the teams participating in a course. The elements of *T* are defined in execution time by the tuple:

$$t = tuple(tn, e)$$

Where *tn* is the name of the team, and *e* is the execution instance of a course.

$$RH \subseteq R \times R$$

Where *RH* is the set named *Role Hierarchy*, and is used to establish different levels of responsibilities in a course. These relationships are defined in design time.

$$UR \subseteq R \times US$$

Where *UR* is the set named *User roles*, and is a subset of many to many mapping between *Users* and *Roles*, established in execution time.

$$TR \subseteq R \times T$$

Where *TR* is the set named *Team roles*, and is a subset of many to many mapping between *Roles* and *Teams*, established in execution time.

With the previous sets defined, now we will proceed to the presentation of the sets related to activity descriptions, including individual and group activities.

$$A = \{a_1, a_2, \ldots, a_n\}$$

Where *A* is the set named *Activities*, which represents descriptions of learning activities. An activity is defined in design time by the tuple:

$$a = tuple(t, d, i, r) \mid \forall_{a \in A} p(i) : i = \{0,1\}$$

Where *t* refers to the title of the activity, *d* is the description of the activity, *i* indicates if the activity is an individual or a team activity, and *r* is the role that can execute the activity.

$$IA \subset A \mid \forall_{a \in A} p(i) : i = 1$$

Where *IA* is the set of *Individual activities*, which is a subset of *Activities*, where *i* means that the activity has to be performed individually.

$$TA \subseteq A \mid \forall_{a \in A} p(i) : i = 0$$

Where *TA* is the set of *Team activities*, and is a subset of Activities, where *i* means that the activity may be performed only once by a team of learners.

Therefore:

$$IA \cup TA \Leftrightarrow A$$

Finally, the sets of activity instances are generated as follows.

$$II \subseteq UR \times IA \mid \forall_{ur \in UR} p(t) \wedge \forall_{ia \in IA} p(r) : t = r$$

Where *II* is the set named *Individual activity instances*; is the set of activities performed individually by each user with a specific role (the same of the activity description). If the activity produces an assessment value, the value is only for the user who performed the activity, and if the activity requires the generation of a product (e.g. project, paper), the user does it herself.

$$TI \subseteq TR \times TA \mid \forall_{tr \in TR} p(t) \wedge \forall_{ta \in TA} p(r) : t = r$$

Where *TI* is the set named *Team activity instances*; is the set of activities performed in teams by users with a specific role (the same of the activity description). If the activity produces an assessment value, the value is for all the team members, and if the activity requires the generation of a product, it can be done by all the users in the team.

$$I = II \cup TI$$

Where *I* is the set named *Instances*, which is the set of all activity instances.

$$I \forall_{i \in I} p(s) : s = \{in, a, s, c, f\}$$

Where each instance of *i*, has the property state *s*, which is the execution state of the instance. The execution states of the instance may be: (1) *in-inactive*; (2) *a-active*; (3) *s- suspended*; (4) *c-completed* with success; or (5) *f -failure*, which means that the instance has been completed but with failure, due to a technological matter or because the learner did not achieved the learning objectives, such as failing a test.

So far, the activity instantiation model has been defined. Following, a worked example of the activity instantiation model is presented.

5. A WORKED EXAMPLE

In the Tecnológico de Monterrey, México, inside the Master program of IT Administration, there is a course of IT Services Continuous Improvement[1]. The course designed by M.S. Teresa de Jesús Lucio, focuses the fifth module to the topic of Problem Management. A part of the module has the following activities:

1. Make critical reading (individual mode, role: learner).
2. Case study reading (individual mode, role: learner).
3. Technical analysis of the case (collaborative mode, role: engineer).
4. Administrative analysis of the case (collaborative mode, role: administrator).
5. Case study conclusions (collaborative mode, role: team member).
6. Delivery of the case study (collaborative mode, role: team leader).

From the above we define the following roles:

$$r_1(learner), r_2(team\ member), r_3(engineer), r_4(admin), r_5(team\ leader)$$

Which constitute the set of roles:

$$R = \{r_1, r_2, r_3, r_4, r_5\}$$

With the following hierarchy:

$$R = \{(r_1, r_2), (r_2, r_3), (r_2, r_4), (r_2, r_5)\}$$

The activities are defined as follows:

$$a_1\begin{pmatrix} Make\ critical\ reading, \\ Book\ ITIL\ Service\ Operation\ Ch.4(\ldots), 1, learner \end{pmatrix}$$

$$a_2\begin{pmatrix} Case\ study\ reading, \\ Case\ study\ No.3(\ldots), 1, learner \end{pmatrix}$$

$$a_3\begin{pmatrix} Technical\ analysys\ of\ the\ case\ study, \\ Analyse\ the\ case\ study(\ldots), 0, engineer \end{pmatrix}$$

$$a_4\begin{pmatrix} Administrative\ analysys\ of\ the\ case\ study, \\ Analyse\ the\ case\ study(\ldots), 0, administrator \end{pmatrix}$$

$$a_5\begin{pmatrix} Case\ study\ conclusions, \\ Make\ conclusions(\ldots), 0, team\ member \end{pmatrix}$$

$$a_6\begin{pmatrix} Delivery\ of\ the\ case\ study, \\ Make\ delivery(\ldots), 0, team\ leader \end{pmatrix}$$

The above elements constitute the following sets:

$$A = \{a_1, a_2, a_3, a_4, a_5, a_6\}$$

$$IA = \{a_1, a_2\}$$

[1] http://www.ruv.itesm.mx/portal/promocion oe/m/mti/plan/homedoc.htm#TI5018

$$TA = \{a_3, a_4, a_5, a_6\}$$

Where

$$IA \cup TA \Leftrightarrow A$$

At runtime, the following learners are registered to the course:

$$us_1(Alice,1,***), us_2(Bob,2,***), us_3(Carol,3,***),$$
$$us_4(David,4,***), us_5(Eve,5,***), us_6(Fran,6,***)$$

Who constitute the set of learners:

$$US = \{us_1, us_2, us_3, us_4, us_5, us_6\}$$

According to the number of learners enrolled, the teacher decided to create two teams:

$$t_1(team\ 1, ago - dic\ 2010), t_2(team\ 2, ago - dic\ 2010)$$

Which constitute the set of teams:

$$T = \{t_1, t_2\}$$

The teacher decides to assign the following roles to the learners:

$$UR = \{(r_3, us_1), (r_4, us_2), (r_5, us_3), (r_3, us_4), (r_4, us_5), (r_5, us_6)\}$$

And by hierarchy inheritance, the final role assignation stands as follows:

$$UR = \begin{Bmatrix} (r_3, us_1), (r_4, us_2), (r_5, us_3), (r_3, us_4), (r_4, us_5), (r_5, us_6), \\ (r_2, us_1), (r_2, us_2), (r_2, us_3), (r_2, us_4), (r_2, us_5), (r_2, us_6), \\ (r_1, us_1), (r_1, us_2), (r_1, us_3), (r_1, us_4), (r_1, us_5), (r_1, us_6) \end{Bmatrix}$$

Then the teacher assigns the following roles to the teams

$$TR = \begin{Bmatrix} (r_2, e_1), (r_2, e_2), (r_3, e_1), (r_3, e_2), \\ (r_4, e_1), (r_4, e_2), (r_5, e_1), (r_5, e_2) \end{Bmatrix}$$

When the execution of the course starts, the system generates the following instances of activities:

$$II = \begin{Bmatrix} ((r_1, us_1), a_1), ((r_1, us_2), a_1), ((r_1, us_3), a_1), ((r_1, us_4), a_1), \\ ((r_1, us_5), a_1), ((r_1, us_6), a_1), ((r_1, us_1), a_2), ((r_1, us_2), a_2), \\ ((r_1, us_3), a_2), ((r_1, us_4), a_2), ((r_1, us_5), a_2), ((r_1, us_6), a_2), \end{Bmatrix}$$

$$|II| = 12$$

$$TI = \begin{Bmatrix} ((r_3, e_1), a_3), ((r_3, e_2), a_3), ((r_4, e_1), a_4), ((r_4, e_2), a_4), \\ ((r_2, e_1), a_5), ((r_2, e_2), a_5), ((r_5, e_1), a_6), ((r_5, e_2), a_6) \end{Bmatrix}$$

$$|TI| = 8$$

$$|I| = 20$$

In total, 20 activity instances will be generated, of which 12 are individual instances and 8 are team instances.

6. CONCLUSIONS

The main contributions of our activity instantiation model are: (1) the model is capable of generating the instances needed for each activity, which ensures the completeness of the learning flow for each learner; (2) the model only generates the activity instances needed, i.e. there are no instances generated for activities that are not in the interest for certain learners, e.g. a user with an specific role does not have an activity instance for an activity that she will not perform; (3) there are two types of activities, individual activities and team activities, this distinction allows the learner to know and maintain the control of her learning process and also to participate with other learners in team-based tasks; (4) the way the instantiation is generated in the model, allows the professor to know the state of the learning flow of each learner and the whole class.

7. ACKNOWLEDGMENTS

The work is partly funded by the Avanza subprogramme (project TSI-020501-2008-53), the PCI subprogramme (project A/018126/08) of the Spanish Government and the Distributed and Adaptive Systems Lab for Learning Technologies Development, DASL4LTD (C-QRO-17/07) from the Tecnólogico de Monterrey, México.

8. REFERENCES

[1] Torres, J., Cárdenas, C., Dodero, J.M., Juárez, E. 2010. Educational Modeling Languages and Service-Oriented Learning Process Engines. In-teh.

[2] Koper, R., Tattersall, C. 2005. Learning Design, a handbook on modeling and delivering networked education and training.

[3] Dodero, J.M., Torres, J., Aedo, I., Díaz, P. 2005. Beyond descriptive EML: Taking control of the execution of complex learning processes. In Proceedings of the Multidisciplinary Symposium on the Design and Evaluation of Digital Content for Education.

[4] Torres, J., Dodero, J., Aedo, I., Diaz, P. 2006. Designing the execution of learning activities in complex learning processes using LPCEL. In Proceedings of the 6th International Conference on Advanced Learning Technologies.

[5] Erl, T. 2004. Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services. Prentice Hall.

[6] Torres, J., Juarez, E., Dodero, J.M., Aedo, I. 2009. EML learning flow expressiveness evaluation. In Proceedings of the Ninth IEEE International Conference on Advanced Learning Technologies.

[7] Hollingsworth, D. 1995. The Workflow Reference Model - Issue 1.1. Technical Report. Workflow Management Coalition.

[8] Holgado, C. 2010. Luces y sombras de la plataforma Moodle: Valoración y experiencia didáctica en lenguas extranjeras. In Proceedings of the Multidisciplinary Symposium on the Design and Evaluation of Digital Content for Education.

[9] Ferraiolo, D., Kuhn, D.R., Chandramouli, R. 2007. Role-based Access Control. Artech House Inc.