A Flexible Infrastructure for the Support of Distributed Learning

Manuel Caeiro, Luis Anido, Martín Llamas, Luis M. Álvárez, and Fernando A. Mikic

Departamento de Ingeniería Telemática
Universidad de Vigo
Campus Universitario s/n, E36200 Vigo (Spain)
{mcaeiro,lanido,martin,lmsabu,mikic}@det.uvigo.es

Abstract. This paper describes an infrastructure devoted to support distributed learning by Information and Communication Technologies. Our purpose is to provide a flexible solution that facilitates the execution of any kind of teaching/learning strategy, where faculty can apply their own methodology without worrying about technological issues. The IMS Learning Design specification offers the underlying semantic framework used to describe any pedagogical approach in a formal way. Our infrastructure supports the creation of final learning systems that can execute learning descriptions expressed according to this framework, providing the services and functionalities required to obtain the maximum learning support and control. Therefore, learning experiences are supported, automated, and executed distributely. We use workflow and groupware technologies in order to support coordination and cooperation in distributed environments, and an object-oriented approach to provide openness, scalability, and flexibility.

1 Introduction

The learning literature describes many pedagogical theories and approaches, each one based on different teaching and learning strategies. In daily practice, most teachers and trainers apply their own principles of learning. In any case, Information and Communication Technologies have been successfully used for decades to provide valuable automation and support, facilitating the realization of learning at distance and in time. Koschmann [1] identified four types of instructional technology paradigms: *Computer Assisted Instruction* (CAI), *Intelligent Tutoring Systems* (ITS), *Logo-as-Latin*, and *Computer-supported Collaborative Learning* (CSCL).

The main drawback is that current systems are oriented to a particular pedagogical approach with no support to use the same system for other learning strategies. Our aim is to free the teaching/learning approach from the technological infrastructure and with the control at hands of instructors. The areas of *Computer Supported Collaborative Work* (CSCW) [2], *Workflow Management*

(WFM) [3], [4], [5] and *Groupware* deal with similar problems that we take into account.

The paper is organized as follows. Section 2 analyses the problem that have guided our work, its goals and background. In section 3 we introduce the infrastructure. Finally, we present some conclusions in section 4.

2 Background

The main drawback when supporting any kind of teaching/learning approach is that it requires the provision of different functionalities and supporting mechanisms. A solution to this problem could be to develop large systems providing the whole functionalities required by the different approaches. But this drives to monolithic and heavy-weight systems that are difficult to manage and control. In this way the construction of scalable architectures enables to obtain lightweight solutions.

A scalable approach is necessary but not sufficient. In addition to provide different kinds of support it is required an integrative framework to let the functionalities and services be combined conveniently. As in WFM [3] a critical question is to push the (learning) procedures out of the applications. In this way, authorized users (teaching staff) should be able to specify the learning activities as they want, without worrying about technological aspects.

We need suggestive metaphors, adequate models and also languages with the appropriate expressiveness and modularity to adapt to a wide range of instructional requirements and styles. In addition, faculty and students should be provided with a practically open-ended set of possibilities for tailoring the technology according to their dynamic needs, so they can concentrate on teaching and learning and not on the technology itself. Specific requirements for our learning infrastructure are:

- To provide a broad technological support for learning based in a common model so it can be maintained independently of the pedagogical approach.
- To allow users (teaching staff generally) to design and control learning: processes, contents, services, activities, etc.
- To allow the construction of learning systems incrementally by meeting requirements such as openness, flexibility, integration, etc. that may be extended to provide further automation and support for learning in an integrative way.

2.1 Strategies of Learning Articulation

Basically, our problem is how to articulate learning. The problem is that there are so many different strategies for providing learning: self-paced, problem-based, group-based, etc., (each one managing different concepts and artifacts) that a common underlying model (that supports the formal description of any of them) is fundamental in order to provide an integrated support.

This problem is very similar to the one present in the CSCW domain of articulating work. In this case two main diverging strategies are distinguished [6]:

- *Regulating interaction* (e.g.: CAI, ITS) as in WFM . This strategy aims at systems that prescribe interaction. In this case, articulation of work is achieved through action constrained by coordination artifacts and protocols. Its purpose is to make more reliable and efficient the collaborative processes, but it restricts the coordination of the interaction to a priori model of interaction.
- *Mediating interaction* (e.g.: *Logo-as-Latin*, CSCL) as in *Groupware*. This position aims at flexible systems that do not prescribe interaction. This kind of systems offers infrastructures that can provide the basis for any application. However, the users have to cope with the complexity of coordinating their cooperative activities. The articulation of work is achieved by ad hoc alignment and improvisation on the basis of mutual awareness.

In the learning domain an innovative proposal has appeared very recently (October 2002) that supports to a certain degree the definition of learning activities according to the specification of both regulating and mediating interaction. It is the *IMS Learning Design* (IMS LD) specification [7], [8], [9] which is based on the *Educational Modeling Language* (EML) [10] developed at the *Open University of the Netherlands* (OUNL). This proposal is focused on the commonalities of the diversity of learning approaches rather than in its particularities. Namely, regardless of the pedagogy involved, in practice every learning design comes down to a set of prescribed *activities* for the actors involved (*learner* and *staff* roles) that should be executed in a certain order in a specific environment. This is the conceptual model of IMS LD in its simplest form (level A). All elements are gathered in an information model accordingly to the structure represented in figure 1:

- *Learning objectives*, *Prerequisites*, and *Metadata* provide information about the learning design and its instructional usage.
- *Components* are the declarations of the different resources that participate in the learning and provide the 'building blocks' for the *method* section: *Roles*, *Activities*, and *Environments*. They are declared separately from any structure to avoid duplication in the *method* when using the same component more than once. *Environments* are composed of two kinds of elements: *Learning Objects* and *Services*. *Learning Objects* are part of the learning design, while *Services* should be provided by proper applications in the final system.
- The *Method* governs the running of the *Learning Design*. The teaching-learning process is modelled in the *Method* on the notion of a theatrical play. A *Play* has *Acts*, and in each *Act* has one or more *Role-parts*. The *Role-parts* within an *Act* associate each *Role* with an *Activity*. The *Activity* in turn describes what that *Role* is to do and what *Environment* is available

to it within the *Act*. In the analogy, the assigned *Activity* is the equivalent of the script for the part that the *Role* plays in the *Act*, although less prescriptive. In this way it regulates interaction and supports interaction.In this model two different kinds of activities are recognized: *Learning* and *Support Activities*. *Learning Activities* prescribe typical tasks. *Support Activities* prescribe activities that have to be carried out by a *Role* in order to support the activity of other *Role*.

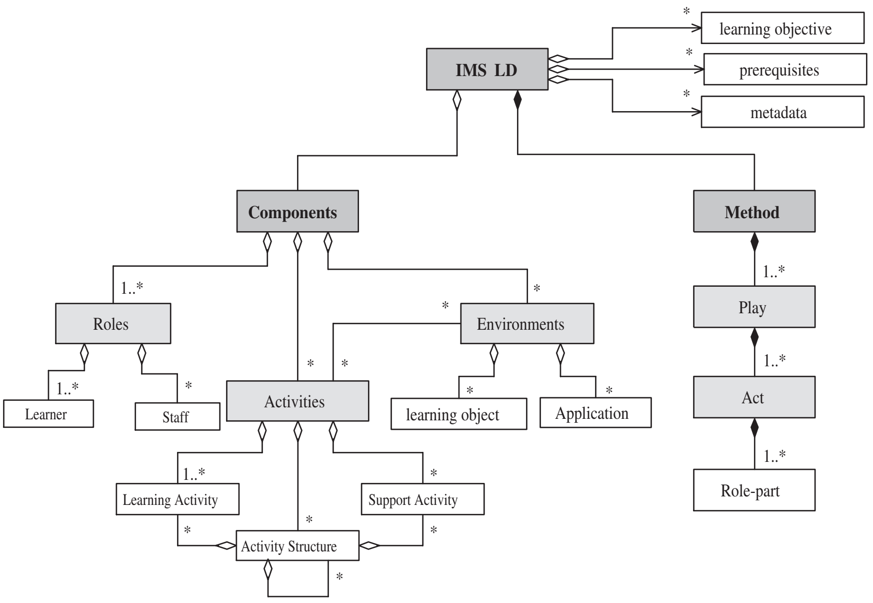

Fig. 1. IMS LD information model level A

The IMS LD specification provides a model to describe both the processes to be carried out and the environments where they are executed, supporting in this way the description of any teaching-learning approach. Our infrastructure is mainly devoted to execute and control these learning designs. To do it, we adopt a solution that allows to combine WFM and *Groupware* functionalities [11], so we support learning interaction by regulation and mediation mechanisms.

2.2 Control at Hands

The problem with the previous analysis is that learning in the real world is not always driven according to a set description of activities and environments. *Ad hoc* alignment and improvisation on the basis of learning status is inexorably interlaced with the execution of predefined procedures and vice versa.

Therefore, we need a mixture of regulating the learning but enabling its modification by authorized users (usually teachers). Schmidth and Simone [4] identify two different kinds of elements:

– *Protocols and artifacts for coordination.* Used to regulate the interactions in the collaborative work, regulating or mediating. They can be templates, maps, scripts, rules, permissions, etc.
– *Awareness.* It is information achieved in the course of doing the learning about its status. It is obtained through the emission and dissemination of signals which somehow indicate the state of the learning activities.

In order to support this kind of behaviour we consider the incorporation of both mechanisms in our infrastructure. WFM technology has developed several solutions to control the state of a process execution as its modification during runtime.

2.3 A Scalable Solution

We do not look for a final learning system but for an infrastructure that enables the construction of such final learning systems. These systems will contain the applications and services required to provide or support certain kinds of learning-teaching approaches, which may be modified and extended. To obtain a scalable solution we focus our attention in object and component-oriented technologies.

Two different kinds of extendable functionalities are devised: (1) the learning applications that may be used during the execution of the activities of the learning designs. This requires the introduction of the appropriate applications, that need to be described adequately to support their configuration (e.g.: communication facilities, simulators, dictionaries); (2) the facilities provided by the infrastructure in order to control and manage the execution of the learning designs (e.g.: monitoring, awareness). In this case we are interested in the results by Manolescu [12], [13] regarding the construction of an object-oriented workflow architecture. His solution provides a stable architecture able to grow with increased usage, allowing components to be updated without destabilizing other parts of the environment.

Authorized users choose which functionalities will be used among the available ones defining appropriately the learning designs and modifying them during their execution.

3 The Infrastructure

Figure 2 depicts the whole view of the processes that we consider in supporting generic teaching-learning approaches. It is based in WFM ideas [5]. This section shows how the different elements considered in our solution are related.

The diagram starts with the *Definition Tools* that are used to describe the learning designs. These learning designs include the formal description of actors, activities, environments, rules, procedures, etc., that may participate in a learning experience. In this way, automation is possible and reusability facilitated, because the same design may be used several times. An *Organizational Model*, and an *Application Model*, enable the separation of the particular learning design from the context in what it is going to be executed (concrete participants and applications used). These learning designs are described according to IMS LD (c.f. section 2.1).

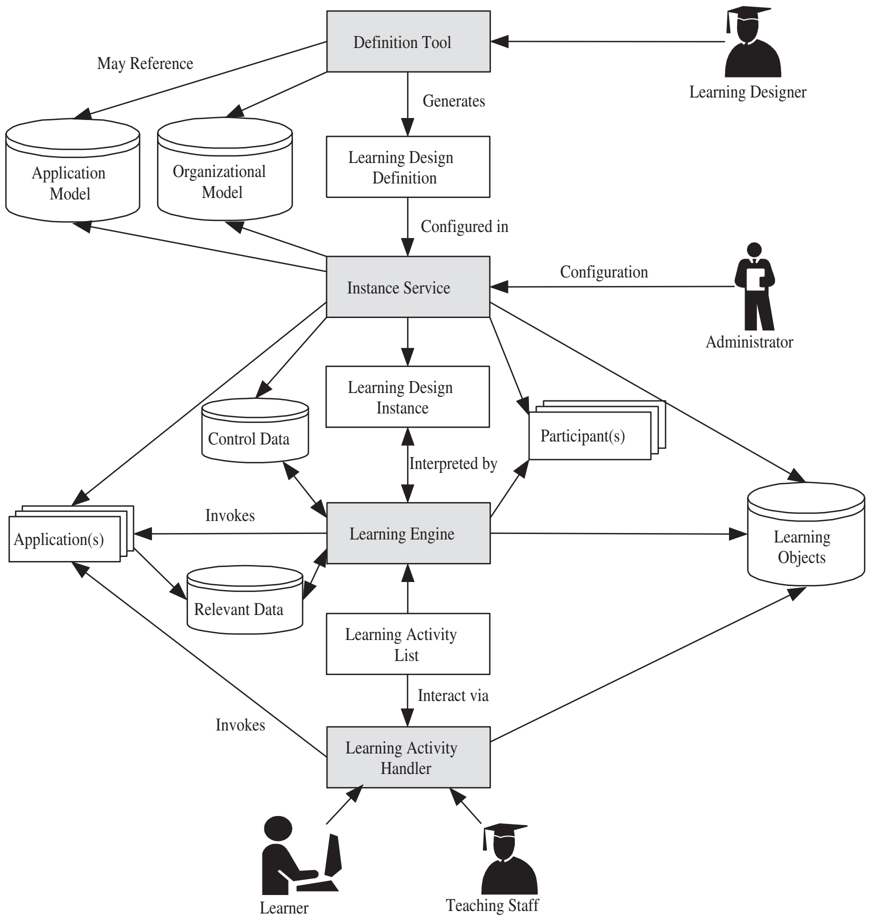

Fig. 2. The whole picture in the support of generic teaching-learning approaches

In the next stage the learning design is put into context ready to be executed. The *Instance Service* is used to include the concrete *Participants*, initiate and configure the proper *Applications* that will participate in the learning design, and the *Learning Resources* (contents, multimedia, exercises, etc.). *Control Data*, as the participants permissions and application use rules, are also fixed.

The *Learning Engine* uses all the information generated in the instance process to propose the learning activities to be realized by the each *Participant*, initiating the appropriate *Applications* and *Learning Objects* that constitute the environment. To do it, a *Learning Activity List* for each participant is generated.

Finally, *Learning Activity Handlers* are used to deliver the proposed activities and environments to the participants (*Learners*, *Teaching Staff* or appropriate *Applications*). These handlers facilitate the distribution of the learning activities.

3.1 The Definition Tools

The *Definition Tools* are authoring tools that will be used by *Learning Designers* to create the Learning Designs. The IMS LD specification is the underlying model used to realize such designs.

These *Definition Tools* may provide specific interfaces to facilitate the design of learning experiences according to certain pedagogical approaches (for example, a self-paced authoring tool would enable the organization of resources to be experienced by a single learner, considering navigation conditions). The difference is that the final learning designs developed by authoring tools following different pedagogical approaches are described using the same model (IMS LD).

3.2 The Instance Service

This service is devoted to adapt the learning design to the context where it is going to be executed. It supports enrolling of the *Participants* (learners, teaching staff, and applications), configuration of the *Applications* (including the required *Learning Engine* functionalities, setting the appropriate permissions and rules of operation, etc.), configuration of the *Learning Objects*, and the creation of an executable *Learning Design Instance*, specifying the *Control Data* to manage the execution of the design. An *Administrator* can interact with this service to manage and control this process.

To facilitate these operations it is necessary that the applications that execute the intended services would be developed according to well-defined *Application Model*. We are extending the application models already defined in the IMS LD specification to provide the description of more services (currently services defined in IMS LD are: E-mail, Conference, Monitor, and Index-search). These *Applications* may require certain functionalities from the *Learning Engine* (e.g. a tutoring component may require a monitoring service of the Learning Engine) that should be initiated too.

3.3 The Learning Engine

The *Learning Engine* is devoted to execute instances of *Learning Designs*. It has to control the execution of the designs, generating the Learning Activity List for each participant, and supporting the functionalities required by the environment where each participant is going to interact with.

The *Learning Engine* was designed according to he lightweight approach adopted by Manolescu [12], [13] in the development of a flexible workflow architecture. He follows an object-oriented architectural style in order to encapsulate workflow features in separate components. This compositional approach lets developers plug in a component *only when their applications require support for it.* In this way, he decouples the workflow core functionalities from other issues, supporting the idea of extending through composition. At the core of the architecture three components provide basic workflow functionality:

– The *Execution Component* provides the mechanism that executes workflow based on their instance.
– The *Process Component* is based on an activity-based process model. It provides the abstractions required to build workflows.
– The *Synchronization Component* uses *Event-Condition-Action* (ECA) rules to allow developers to define dependencies within the workflow domain.

Additional components implement advanced workflow functionalities: *Monitoring*, *History*, *Persistence*, *Manual Intervention*, *Work list*, *Federated Workflow.* These components implement a wide variety of workflow features that we adapt to satisfy the requirements of our learning infrastructure. On the one side, they provide support for the control and management of the learning processes: *Execution*, *Synchronization*, *Monitoring*, *Persistence*, *Work lists*, and *Federated Workflow.* On the other side, they enable the dynamic modification of the prescribed designs: *Process*, *History*, and *Manual Intervention.*

The compositional approach enables to tailor the workflow functionality to our requirements. In this way we introduce the following functionalities:

– An *Awareness Component* in order to maintain a record of on-line users, activities attained, last accesses, etc. This could be a basic component with extensions for tutoring, evaluation, scheduling, etc.
– A *Notification Component* in charge of notifying the relevant situations to the specified actors.
– A *Group Management Component* to enable the grouping of participants and their management.
– A *Timetabling Component* to plan the activities of a user involved in the execution of several learning designs.
– A *User Management Component* to support the modification of the properties associated with users: permissions, belonging to groups, etc.

3.4 The Learning Activity Handlers

The *Learning Activity Handler* is the software that manages the interaction between each participant and the *Learning Engine*, supporting the execution of the activities contained in the *Learning Activity List.* It enables learning activities to be passed from the *Learning Engine* to each participant and status conditions to be passed between each participant and the Learning Engine.

An important aspect of this component is to provide the environment where the participant has to operate, supporting access to applications, providing required resources and information, and enabling the contact with the other participants that collaborate in the same learning experience. As in WFM we consider two different kinds of applications:

– *Client Applications.* They are software programs that provide support for the realization of the user activities (e.g.: communication tools, agendas, annotation facilities). They may request facilities and services from the *Learning Engine* to provide the appropriate service required.
– *Invoked Applications*, which support the processing of particular learning activities (e.g.: exercise evaluation, automatic tutoring). They have appropriate *Learning Activity Handlers* that are used to dynamically invoke the appropriate applications and transfer the required information and parameters.

This component is essential in order to distribute the execution of the learning activities. Each person (or software application) participating in the execution of a learning design may interact from different locations and in different time, accordingly to the requirements of the learning design. In addition, the *Learning Activity Handler* enables that a participant can communicate with several *Learning Engines*, consolidating her/his activities into a single list of tasks.

4 Conclusions

Nowadays, there are many applications and systems that provide functionalities and services valuable for the provision of learning. These solutions are adequate for the provision of certain kinds of pedagogical approaches, but not for others. Maybe, the reason of this problem is the fast development of the technology itself: at the same time that the technology enabled the realization and support of activities, new applications and systems were developed to support additional learning features and processes. The problem is that new systems do not combine and integrate the different possibilities from a pedagogical perspective. In our solution, the IMS LD specification plays a central role. Mainly, because it is an active part providing support for the description of learning independently of the learning approach.

Currently, there are some initiatives in the learning domain concerned this problem. The main purpose of the IMS LD specification is to provide a framework that supports pedagogic diversity and innovation, while promoting the exchange and interoperability of e-learning materials. In this way they propose a set of elements that can describe any design of teaching-learning process in a formal way. IMS LD enables to separate the definition of the learning designs from the logic needed to realize them. In this way it is allowed to modify one without affecting the other. Teachers can define and manage their own learning strategies while software engineers support the technology.

Acknowledgments. We want to thank *Ministerio de Ciencia y Tecnología* for their partial support to this work under grant *CORBALearn: Interfaz de Dominio guiada por Estándares para Aprendizaje Electrónico* (TIC2001-3767).

References

1. Koschmann, T.: Paradigm Shifts and Instructional Technology: An Introduction. In T. Koschmann (Ed.) CSCL: Theory and Practice of an Emerging Paradigm. Mahwah, NJ: Lawrance Erlbaum Associates (1996) 1–23
2. Guareis de Farias, C.R., Ferreira Pires, L., van Sinderen, M.: Frameworks for the Development of CSCW Systems (1999)
3. van der Aalst, W.M.P.: The Application of Petri Nets to Workflow Management. The Journal of Circuits, Systems and Computers, 8(1) (1998) 21–66
4. Georgakopoulus, D., Hornick, M., Sheth, A.:An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. Kluwer Academic Publishers. Distributed and Parallel Databases, 3 (1995) 119–153
5. WfMC: Workflow Management Coalition Terminology & Glossary (WFMC-TC-1011). Issue 3. Hampshire, United Kingdom (1999)
6. Schmidt, K., Simone, C.: Mind the Gap! Towards a Unified View of CSCW. Proceedings of the COOP 2000. The Fourth International Conference on the Design of Cooperative Systems. Sophia Antipolis, France (2000)
7. Koper, R., Olivier, B., Anderson, T. (Editors): IMS Learning Design Best Practice and Implementation Guide. Version 1.0. Public Draft, IMS Global Learning Consortium (2002)
8. Koper, R., Olivier, B., Anderson, T. (Editors): IMS Learning Design Information Model. Version 1.0. Public Draft, IMS Global Learning Consortium (2002)
9. Koper, R., Olivier, B., Anderson, T. (Editors): IMS Learning Design XML Binding. Version 1.0. Public Draft, IMS Global Learning Consortium (2002)
10. Koper, R.: Modeling Units of Study from a Pedagogical Perspective: the Pedagogical Meta-model behind EML.Version 2. First Draft, Educational Technology Expertise Centre Open University of the Netherlands (2001)
11. Ben-Shaul, I., Kaiser, G.: Integrating Groupware Activities into Workflow Management Systems. In Proceedings of the Seventh Israeli Conference on Computer Systems and Software Engineering (1996) 140–149
12. Manolescu, D.A.: Micro-workflow: A Workflow Architecture Supporting Compositional Object-Oriented Software Development. PhD thesis, University of Illinois, Urbana-Campaign. Availables a Computer Science Technical Report UIUCDCS-R-2000-2186 (2000)
13. Manolescu, D.A, Johnson, R.: A Micro-workflow Component for Federated Workflow. OOPSLA2000 Workshop on Implementation and Applicationn of Object-Oriented Workflow Management Systems III (2000)