Pedagogical Scenario Modelling, Deployment, Enactment and Evolution

Yvan Peter, TRIGONE Laboratory, Bât. B6 – Cité Scientifique – 59655 Villeneuve d'Ascq cedex - FRANCE, +33 3.20.43.32.64, +33 3.20.43.32.79, Yvan.Peter@univ-lille1.fr

Xavier Le Pallec, TRIGONE Laboratory, Bât. B6 – Cité Scientifique – 59655 Villeneuve d'Ascq cedex - FRANCE, +33 3.20.43.32.89, +33 3.20.43.32.79, Xaviel.Le-Pallec@univ-lille1.fr

Thomas Vantroys, TRIGONE Laboratory, Bât. B6 – Cité Scientifique – 59655 Villeneuve d'Ascq cedex – FRANCE, +33 3.20.43.32.65, +33 3.20.43.32.79, Thomas.Vantroys@univ-lille1.fr

Pedagogical Scenario Modelling, Deployment, Enactment and Evolution

Abstract

The rise of the pedagogical scenario approach supported by the standardisation of IMS Learning Design is changing the focus from the pedagogical objects to the activities that support learning. With the standardisation, comes the promise of the reuse of successful designs for the pedagogical scenarios. However, the uptake of this approach relies on a sound support of the users both at the design phase and at the execution phase and the level to which successful design can be adapted for reuse in both phases. This chapter covers the whole lifecycle of pedagogical scenarios and shows the current level of support one can find in existing platforms and tools. It present also the way to enhance this support through the use of Model Driven Engineering for the design and deployment phase and implementation techniques to provide enactment engines that allow flexible runtime execution.

INTRODUCTION

The management and reuse of learning objects has now reached a certain maturity thanks to the standards such as SCORM (ADL, 2006) that enables the reuse of learning objects across platforms and LOM (LTSC, 2002) that defines metadata for their description. CORDRA (CORDRA, 2006) completes this with the means to federate object repositories and to enable the retrieval of learning objects. Thanks to this set of standards, the cost of design is lessened because of the possible reuse and the better perenniality of the resources. This level of maturity has not been reached already when one considers the pedagogical scenarios design. Indeed, the focus is evolving from the resources to the activities. The emerging standard related to these scenarios is IMS-LD (IMS, 2003). However designing pedagogical scenarios is still an expert job at least because there is still a lack of proper editors. Having however succeeded in the design of a pedagogical scenario, it is still common to reengineer it or to make it evolve slightly because the context or the hosting Learning Management System (LMS) changes or because the learners have difficulties with some activities. These modifications of the scenarios can happen between two executions (iterative design) or at runtime if the platform can support it. In this chapter, we will present solutions to support the lifecycle of pedagogical scenarios from the design time to the deployment on a specific platform and the enactment and runtime evolution. These solutions aim at keeping the pedagogical design across different contexts which will lessen the cost of the design while permitting a continuous adaptation.

This chapter is composed of two parts. In the first part, we take into consideration the fact that most Learning Management Systems do not provide any support for the enactment of pedagogical scenarios. We will first analyse the shortcomings of these platforms in the light of the IMS-LD proposal and the means to build manually a learning environment corresponding to a given pedagogical scenario. We will then show how Model Driven Engineering (MDE) helps to focus on the pedagogical scenario modelling and design while supporting the process of implementing it on a specific platform.

The second part of the chapter provides the alternative view of the current efforts to provide enactment capability to existing LMS. The focus is put on standalone enactment services which can be embedded into a LMS. We first focus on the general architecture of such systems. Next we present the research related to the flexible enactment of the pedagogical scenarios which enables adaptation according to the learners' needs. Finally, we review existing solutions for and examples of integration of an enactment engine into a LMS.

In the conclusion we will show how these two approaches complement each other and contribute to the uptake of the pedagogical scenarios by supporting their whole lifecycle.

MODELLING AND DEPLOYING PEDAGOGICAL SCENARIOS

Most of wide-spread e-learning web platforms do not support enactment of pedagogical scenarios. Underlying mechanisms for such enactment are present but are not sophisticated enough. However, if a teacher thinks about a detailed pedagogical scenario, it is still possible to create the corresponding learner(s) environment while respecting underlying pedagogical intentions of the scenario. For instance, she/he could find convenient names to items like forums (e.g., by adding a suffix referring to the corresponding activity like forum_studyChapter1), document repositories, etc. and/or could add textual help in these ones. Teacher may thus mitigate the lack of automation mechanisms and continue to use her/his regular platform which is not powerful but accessible.

Model Driven Engineering (MDE) provides theoretical frameworks that could serve for automatic construction of such learner environments based on the modelling of pedagogical scenario. This approach has many strategic benefits: pedagogical scenarios become more productive and so useful which also implies a better reusability of underlying courses; if construction is related to several platforms, there will be less barriers for a teacher to use other platforms because her/his courses will be automatically created on them … In this part, we detail one approach that combines MDE, pedagogical scenarios and e-learning platforms.

First, we review the most widespread e-learning web platforms and show their limits concerning pedagogical scenarios. We also show how to use these platforms to enact 'manually' such scenarios. Next, we show how using pedagogical scenarios changes the usual course design process. The increasing place of computational models is one of the main changes. We finally describe how such models may be used efficiently to automatically produce learner(s)'s environments and to provide other interesting benefits.

Current widespread e-learning web platforms

No support for enactment of pedagogical scenarios

E-learning web platforms[1] are a widespread means to support learning. Users of such platforms are millions (Adkins 2005). World wide consortiums (ex: ADL, IMS) create related standards like SCORM, Learning Design, QTI, etc. which are objects of many scientific works. Functionalities of current e-learning web platform has significantly increased this last decade. However, platform designers still have to implement a lot of mechanisms in order to support missing strategic types of pedagogical intention: checking knowledge related to prerequisites, objectives,

[1] May be referred to Learning Management System, Learning Content Management System, …

progression rate, sequencing activities, managing needed cognitive conflicts… As we previously mentioned, we focus in this chapter on pedagogical scenarios. The concept of pedagogical scenario is currently in the spotlight if we consider the amount of related scientific works and the IMS-LD standardisation initiative. (Schneider, 2003) defines a pedagogical scenario as "*a sequence of phases within which students have tasks to do and specific roles to play*". In the IMS-LD vision, this refers to usual workflow management system mechanisms like conditional control, parallel flows, time management, etc. E-learning platforms which support enactment of pedagogical scenarios allow constructing learning environments with a better respect of original pedagogical intentions.

Unfortunately the most-used e-learning platforms do not support enactment. We may verify this assumption by considering five features among the main ones related to IMS-LD pedagogical scenarios and five widespread e-learning web platforms (see Table 1): Blackboard (Blackboard 2006), Claroline (Claroline 2006), Ganesha (Anema 2006), Moodle (Moodle 2006), WebCT (WebCT 2006). We illustrate first the five features on an example of pedagogical scenario (see Figure 1).

The scenario is a typical framework of course that we make in our diploma (*e-services*, *IPM*). This course concerns a *tutor* and a set of *students* (two *roles*). There are two main activities that run in parallel which are respectively related to course and project. These plays are named *Play One* and *Play Two*. Play one is composed of three acts which are so executed sequentially. Act *Theory 1* is only about studying course documents related to chapter 1. While students are studying these documents, they may ask questions on a forum (*asynchronous conference service*) and tutor (but also students) may provide answers. When this act is over (see further) there is a 2 hours - chat session in order to definitively close the study (act *Discussion 1*). Students make exercises in the final act (*Practice*) while tutor may answer students' questions. This last act is very similar to the single one act of Play two. This act (*Project 1*) is dedicated to the project: students work on it and tutor help them. We have reduced the play one to one chapter for more readability, but each additional chapter may have its own three acts.

With this example, we may examine five main features of IMS-LD and their implementation on most widespread e-learning web platforms.

1. Properties management. **-Property* concepts provide one of the most powerful mechanisms of LD. A scenario may define global/local and personal/group variables, change their values or use them at control/notification points (e.g. when an act is completed). Designer also may define what properties may be viewed or changed through the *monitor* service.

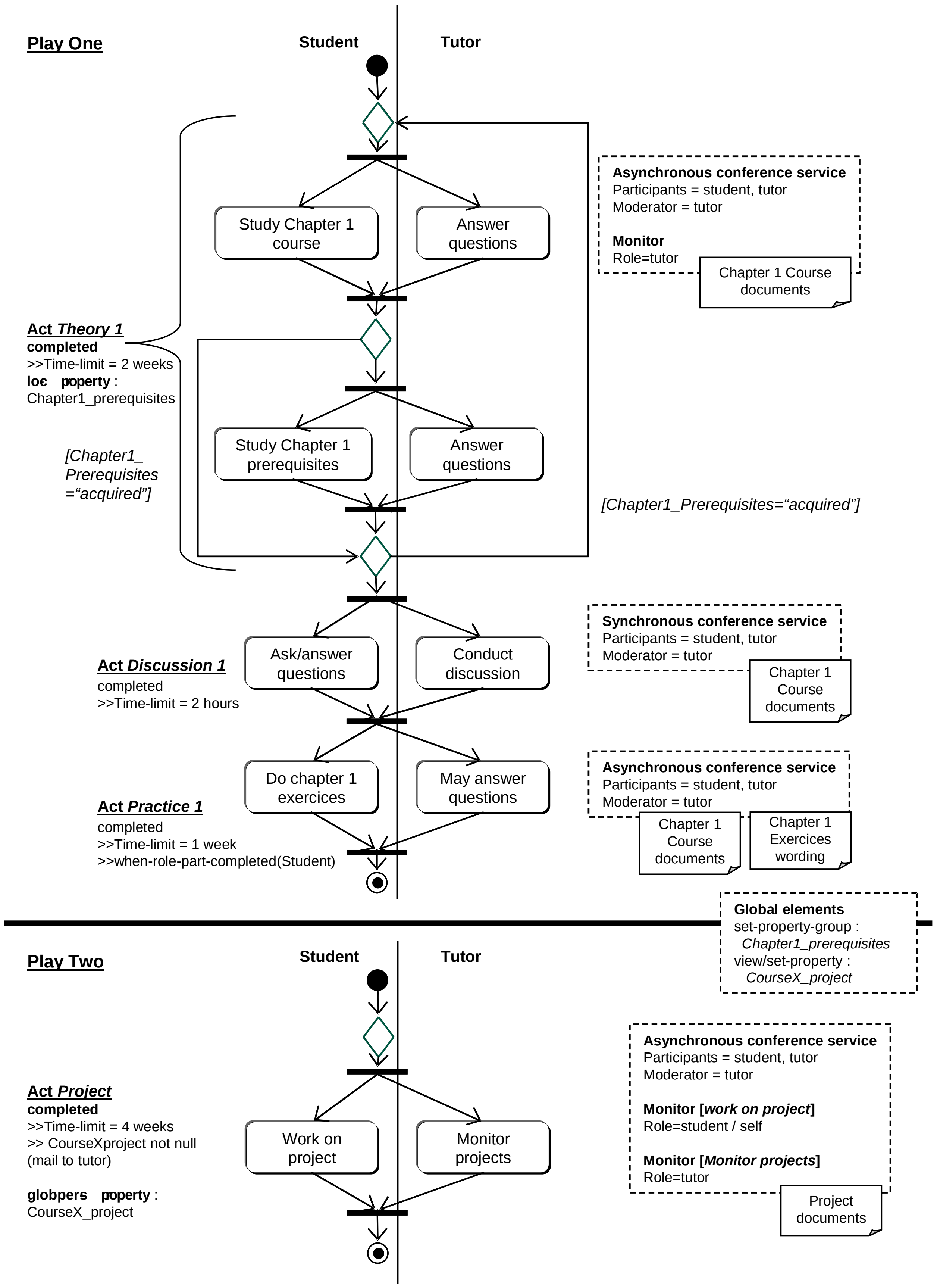


Figure 1. Example of pedagogical scenario (LD-oriented UML activity diagram)

In our scenario, we define *Chapter1_prerequisites* and *Chapter1_project* properties which are used further. *Chapter1_prerequisites* is a local property: it is associated to one execution and there is one value for all users. *Chapter1_project* is global and personal property. This one has its own life, that is to say it does not stop when the unit of learning is over, and there is one value for each participant. The underlying idea of this property is that each student may have her/his project in her/his e-portfolio.

None of the studied platforms proposes properties management.

2. Event mechanism. An event notification mechanism is available in LD through different concepts: *conditions*, *on-completion*, *when-property-value-is-set* (associated to completion of *activity*, *act* and *play*)… First, it is possible to associate processing instructions to the end of an activity. Instructions consist of changing properties value and showing/hiding some scenario elements. For example, when *Study Chapter 1 prerequisite* ends, *Chapter1_prerequisites* property is changed to “acquired”. Second, it is also possible to associate instructions to the change of a property through conditions: designer may define conditions (if-then-else) for the scenario and each one are evaluated when concerned properties change. For example, when *Chapter1_prerequisites* is set to “needed”, the *Study Chapter 1 course* activity is hidden and the *Study Chapter 1 prerequisites* one is shown. Finally the *when-property-value-is-set* feature of *completion* allows ending an activity, an act or a play when the value of a property changes. The act “Project” ends when every student has given its project (all *CourseXProject* properties are not null). The *notification* concept allows sending an email to the tutor at this point.

Studied platforms provide some trigger points. For example, Claroline allows defining some actions based on dates (rights for students to upload their works), Ganesha, WebCT and Blackboard allow running a learning step when a student passes a test. But these features are related to ad-hoc mechanisms and can not compete with generic mechanisms of IMS-LD.

3. Conditional control flow. Properties management and event mechanism are really interesting thanks to the *condition* concept. Each condition definition is essentially composed by an *if-then-else* statement.
None of the considered platforms is able to provide a generic conditional control flow mechanism. However we find specific/ad-hoc control points like the previously mentioned one related to test results.

4. Time management. *Time-limit* concept is the main LD time management mechanisms. Activities may be terminated (and following ones may be launched) according to specific time limits. Moodle, Blackboard and WebCT provide an efficient support for this feature, yet without ability to combine with properties/event mechanisms.

5. Group management. Managing learning path for a group of student is another main feature of LD. Every platform allows managing students groups (own document repositories, discussions, forums …). They also provide definition of rules and associated access rules to contents or tools items. But conditional control flow for group is generally limited to the time dimension (for each platform). There is no way, in any platform, to specify that activity B may be active when previous activity A has been completed by every students of the group.

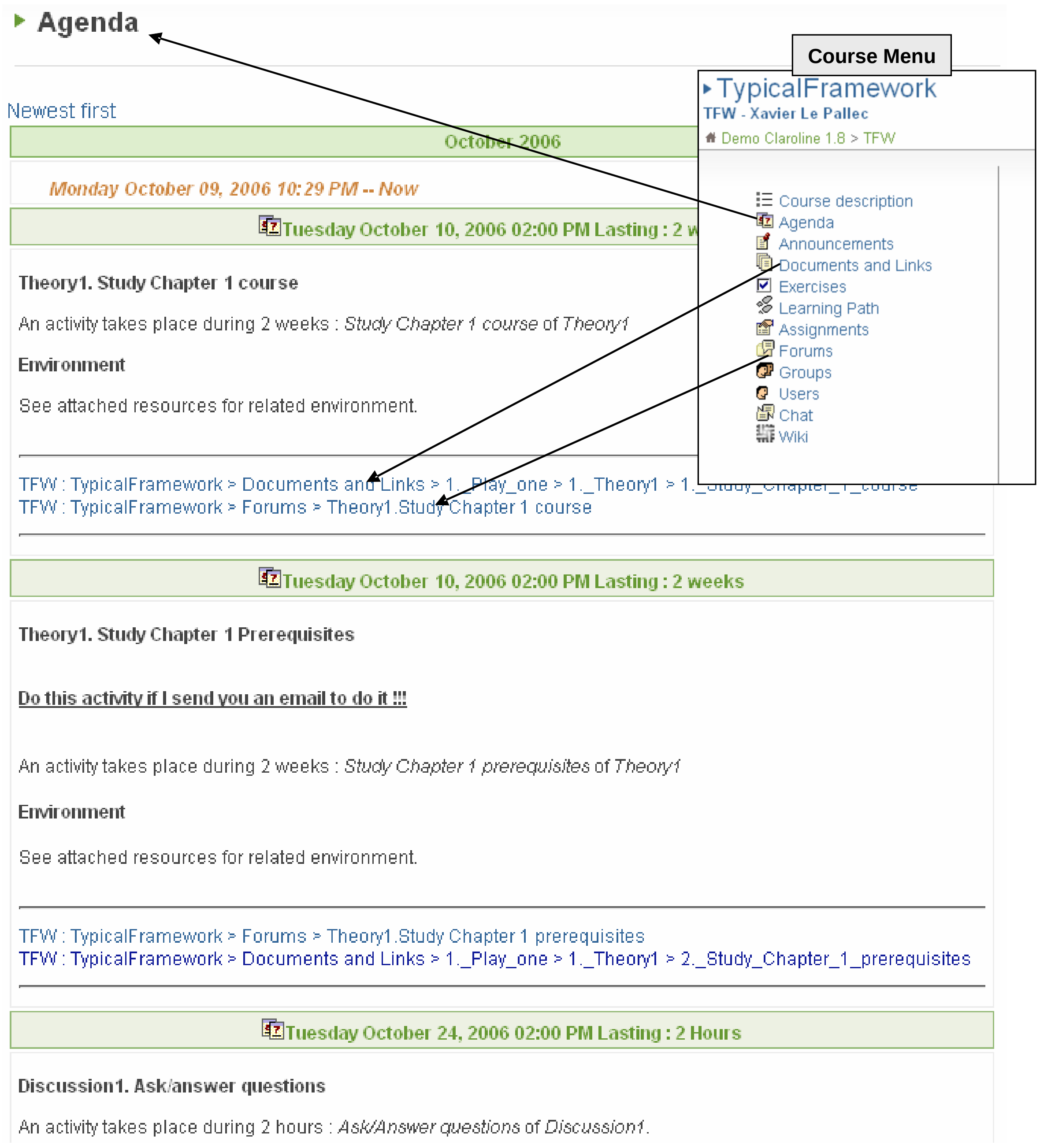
Table 1 presents a summary of our analysis for the main used platforms.

Platforms / Features	Blackboard	Claroline	Ganesha	Moodle	WebCT
Conditional control flow	Only time directed (no choice between two activities)	Only time directed (no choice between two activities)	Only test-result (no choice between two activities)	Only time directed (no choice between two activities)	Only time directed (no choice between two activities)
Time management	Effective support	Only for statistic and stopping submission	Only for statistics	Effective support	Effective support
Properties management	no	no	no	no	No
Event mechanism	no	no	no	no	No
Group management	Exists but without group control	Exists but without group control	Exists but without group control	Exists but without group control	Exists but without group control

Table 1. Platforms and their support to pedagogical scenarios enactment

How to enact manually pedagogical scenarios

If a teacher wants to or used to describe her/his courses through pedagogical scenarios, she/he does not have to banish all platform which miss enactment of such scenarios. When a course is implemented on a platform, teacher may give the scenario to students in order to explain them her/his primary pedagogical intentions. She/he may also give a map to describe correspondences between the scenario and course items on the platform. Another solution may be more efficient: every item reflect the underlying scenario by their name, their order, the presence of small textual help in each forum, document repositories… in addition to the description (somewhere) of the original scenario. This does not replace automation mechanisms but mitigate their absence. We illustrate such implementation (on the previous scenario) of the Claroline platform with two different strategies. The choice of Claroline is that it is a very user-friendly platform but unfortunately the poorest one considering functionalities.


Agenda

Newest first

October 2006

Monday October 09, 2006 10:29 PM -- Now

Tuesday October 10, 2006 02:00 PM Lasting : 2 w

Theory1. Study Chapter 1 course

An activity takes place during 2 weeks : *Study Chapter 1 course* of *Theory1*

Environment

See attached resources for related environment.

TFW : TypicalFramework > Documents and Links > 1._Play_one > 1._Theory1 > 1._Study_Chapter_1_course
TFW : TypicalFramework > Forums > Theory1.Study Chapter 1 course

Tuesday October 10, 2006 02:00 PM Lasting : 2 weeks

Theory1. Study Chapter 1 Prerequisites

Do this activity if I send you an email to do it !!!

An activity takes place during 2 weeks : *Study Chapter 1 prerequisites* of *Theory1*

Environment

See attached resources for related environment.

TFW : TypicalFramework > Forums > Theory1.Study Chapter 1 prerequisites
TFW : TypicalFramework > Documents and Links > 1._Play_one > 1._Theory1 > 2._Study_Chapter_1_prerequisites

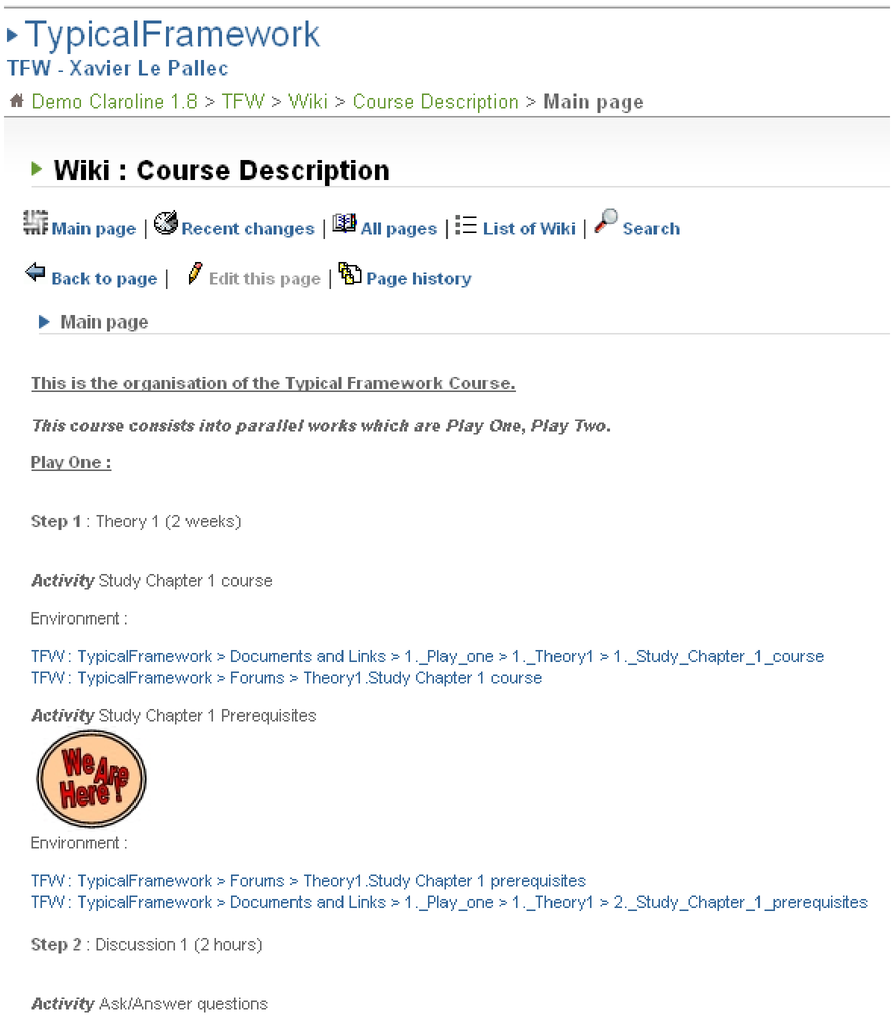
Tuesday October 24, 2006 02:00 PM Lasting : 2 Hours

Discussion1. Ask/answer questions

An activity takes place during 2 hours : *Ask/Answer questions* of *Discussion1*.

Figure 2. Example of Claroline implementation of a LD scenario with Time-Strategy

The first strategy may be named time-strategy because it relies on time-limit of each act. It first creates in *Documents and Links* a directory for each play, a subdirectory for each act and a sub-subdirectory for each activity where documents will be placed. Second, if a forum is needed in one LD (learning-) activity, a forum is created in the Claroline course with the name *[ActName].[Activity_Name]*. This forum will be in the forum category created for owner play. The final point is the main one: each LD activity is transformed into an entry in the course agenda (as illustrated in Figure 2). Each entry contains URLs of associated learning objects or services in order to present an environment which corresponds to the activity. Duration is also present. This time-strategy works fine if no time-limit is missing. It has the benefit of being very simple. But if one or more time-limit is missing, it does not work. We may then use a Wiki-strategy which uses Claroline Wiki. A Wiki page is created which represent the organisation of all the activities with links to related resources (learning objects and services). Benefit of this strategy is that tutor has a structural description of her/his course and she/he has great expressive capabilities to exploit it thanks to Wiki.


TypicalFramework
TFW - Xavier Le Pallec
Demo Claroline 1.8 > TFW > Wiki > Course Description > Main page
Wiki : Course Description
Main page | Recent changes | All pages | List of Wiki | Search
Back to page | Edit this page | Page history
Main page
We Are Here!

This is the organisation of the Typical Framework Course.

This course consists into parallel works which are Play One, Play Two.

Play One :

Step 1 : Theory 1 (2 weeks)

Activity Study Chapter 1 course

Environment :

TFW : TypicalFramework > Documents and Links > 1._Play_one > 1._Theory1 > 1._Study_Chapter_1_course
TFW : TypicalFramework > Forums > Theory1.Study Chapter 1 course

Activity Study Chapter 1 Prerequisites

Environment :

TFW : TypicalFramework > Forums > Theory1.Study Chapter 1 prerequisites
TFW : TypicalFramework > Documents and Links > 1._Play_one > 1._Theory1 > 2._Study_Chapter_1_prerequisites

Step 2 : Discussion 1 (2 hours)

Activity Ask/Answer questions

Figure 3. Example of Claroline implementation of a LD scenario with Wiki-Strategy

We have seen that if most-used e-learning web platforms have a lot of functionalities they do not provide a support to enact pedagogical scenarios. However, designer or tutor may implement a scenario in such platforms by applying mapping rules according to her/his context (scenario complexity, her/his skills about concerned platform…). Resulting learning environment is far from being as powerful as with a scenario runtime engine but it remains correct and has the benefit to provide familiar environment to users (tutors and students).

Design learning course

We have previously mentioned that teachers or tutors may define pedagogical scenarios for courses in order to better express their pedagogical intentions. However it is important to discuss about the relation between teachers and pedagogical scenarios in e-learning web platforms context.

Typical lifecycle when designing and building courses through pedagogical scenarios

Current platforms are well supplied but they are not really powerful. This has a positive consequence: if a teacher wants to create an environment on such platform for a particular course, she/he makes a draft concerning learning activities and relation with pedagogical contents on a sheet of paper and creates corresponding ‘items’ on the selected platform. In other terms, due to their low-level complexity, usual platforms are accessible by most teachers: they can “easily” implement their courses themselves. We verify such a simple lifecycle within our on-line diplomas named *e-services* and *IPM* (Elearning 2006) and through our partnership with other universities (AUF 2006) (Tempus 2006) and our participation to related conferences (Unfold

2005), (Le Pallec 2006). It seems to be a current practice for learning environments with few students (i.e. less than 50).

“Accessible by teachers” means that teachers generally need about one or two hours of training to become familiar with a particular platform. This is totally different when one considers e-learning platforms supporting the enactment of pedagogical scenarios. The expressiveness increase goes unfortunately with a complexity increase. The construction of an electronic learning environment from a paper draft is no longer so trivial. As in software engineering, the definition of a computational model (like a UML model) is required before the implementation. But writing computational scenarios need skills that few teachers have and that few ones are ready to spend time to acquire. This is the reason why design/construction processes, which may be associated to such platforms, integrate someone with these particular skills: generally an *instructional designer*, that is, a professional whose competence is two-fold, feeling at ease with both pedagogy and computer related subjects (particularly in the e-learning arena). UML activity diagrams on Figure 4 and Figure 5 represent two lifecycles which are associated to “evolved” platforms. The five-step *Instructional System Development* (ISD) process is a proposition related to designing a course for LD-compliant platforms (Koper 2005). It contains very technical steps. The *design* and *development* steps need someone who is familiar with formal activity diagrams and LD concepts and can write XML instance documents. The six-phase *MISA* process is currently associated to the platform *Explor@* (Paquette 2005). MISA includes definition of workflow-like models (called here *Learning Event Network*) but also knowledge or maintenance models. Expressive capabilities of MISA imply the presence of context experts, media producers, training managers … and instructional designers (particularly for Learning Event Network).

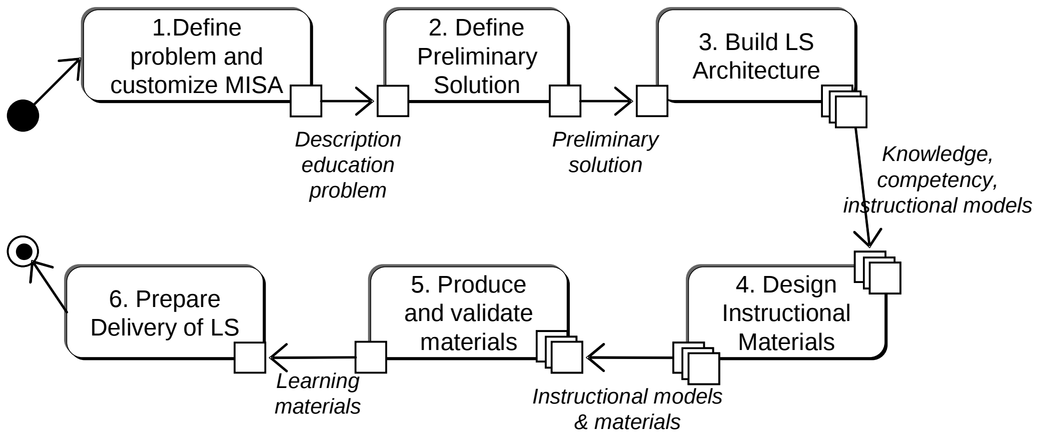

1.Define problem and customize MISA
2. Define Preliminary Solution
3. Build LS Architecture
Description education problem
Preliminary solution
Knowledge, competency, instructional models
6. Prepare Delivery of LS
5. Produce and validate materials
4. Design Instructional Materials
Learning materials
Instructional models & materials

Figure 4. The five-step Instructional System Development (ISD) process

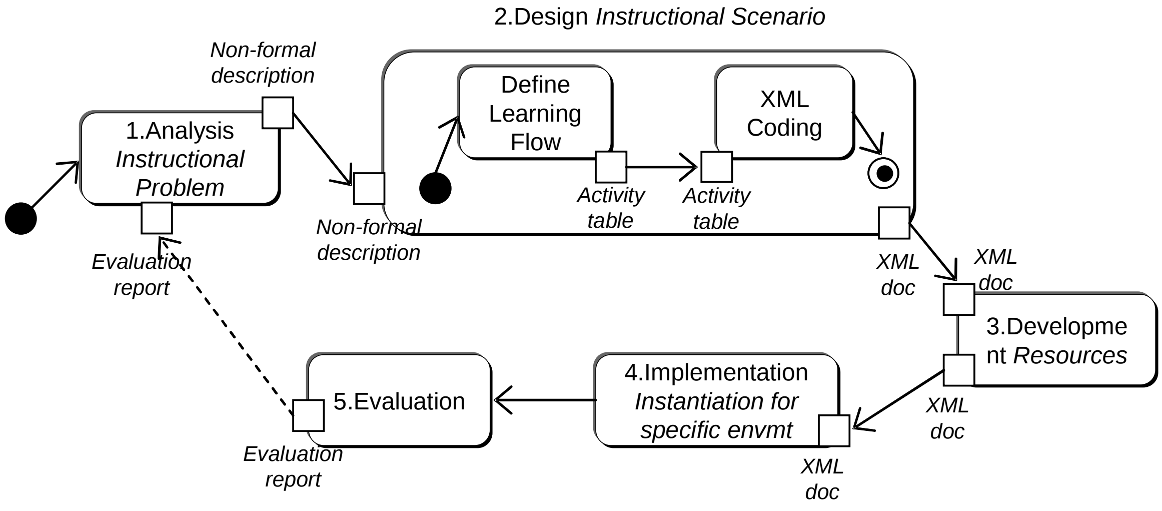


Figure 5. The six-phase MISA process

Pedagogical scenarios are generally defined through computational models. Most of software engineering areas use models, because they constitute a convenient way to transform informal analysis into software implementation. (Caron 2005) underlines that they are efficient boundary objects between teacher and instructional designers in e-learning context. (Pernin 2006) also demonstrates that models are well-suited to re-use pedagogical scenarios. Modelling concepts or modelling tools which are used to define pedagogical scenarios are then important because they may improve or degrade discussion during design process.

Accessible model editors to keep teacher in design process

The degree of teacher participation in the pedagogical scenario definition is related to the respect of her/his primary pedagogical intentions. The used authoring tool plays a strategic role because it may reduce complexity and difficulty to define scenario elements and then it may increase action abilities of teachers. Such tools should allow teachers to define themselves their scenarios because employing an instructional designer is far from being economically possible for all e-learning structures.

So, authoring or modelling tools are a fundamental aspect of the current e-learning approach based on pedagogical scenarios.

Unfortunately, if we consider the LD community, current tools (Kew 2006) are not suitable for teachers. Reload does not provide a user-friendly interface because it is based on forms. MOT+ is more intuitive thanks to its graphical interface but it does not hide complexity of LD concepts. LAMS may seem to be more accessible but it does not allow defining complex scenarios as with IMS-LD (LAMS is not a full LD-compliant editor). Finally the Collage prototype (Hernandez-Leo et al, 2006) seems to be an interesting initiative: it aims to define scenarios through using pattern. This hides efficiently complexity. Unfortunately creating patterns currently still needs coding in Java. We may hope that, considering these are only the first LD authoring tools, a lot of improvements will be made and particularly concerning their complexity level.

Building automatically pedagogical scenarios-based learner environments through Model-Driven engineering

If mapping LD scenarios into usual e-learning web platforms is interesting, manual mapping/building is not. As LD scenarios are computational models, it is possible to use mechanisms from Model Driven Engineering (MDE) to automatically construct learning environment from a LD scenario in such platforms. We present an approach which is inspired by MDE in this part. We first summarize the principles of MDE.

Model-Driven Engineering (MDE) principles

Rather that giving a new definition of what is Model Driven Engineering, let’s use the one given by (Van Der Straeten, 2005): “*Model-driven engineering (MDE) is an approach to software development where the primary focus is on models, as opposed to source code. Models are built representing different views on a software system. Models can be refined, evolved into a new version, and can be used to generate executable code. The ultimate goal is to raise the level of abstraction, and to develop and evolve complex software systems by manipulating models only*”. Models, model transformation and code generation are the three main aspects of MDE.

Models. There has been a lot of works to increase the relevance of models when considering their (code) productivity. Among them, we find mechanisms or specifications which allow defining formal or strict modelling concepts tuned for a specific business area (like IMS-LD for pedagogical scenarios). Standards like OMG’s Meta-Object Facility (MOF), UML Profile and Eclipse Modelling Framework (EMF) (Abouzahra 2005) allow defining such *metamodel*s (i.e. sets of modelling concepts). Generic graphical editors were developed so as to permit designers to define specific domain oriented *models* using UML notation (rather than writing XML documents). Graphical Modeling Framework (GMF, 2006) or TopCaseD (Farail, 2006) are such generic editors for EMF but there are also Magic Draw (MagicDraw, 2006) for UML Profile and ModX for MOF (Le Pallec, 2005). Other interesting tools for models are generic programming languages (Perez 2006) like Kermeta for EMF or MOFScript for MOF. They allow testing models validity (ex: are there never-reached activities in our previous LD scenario?) or simulating their future production. Finally, works like (Marvie, 2006) focus on a step-by-step model definition.

Code generation. One of the main requirements of MDE is that models have to be productive. Productivity is mainly possible thanks to code generators. Generation may be made by generic engines with set of rules dedicated to a specific domain or by domain dedicated engines. Generation from a model may also be done through the definition of an intermediary platform/technological model (MDA 2003). For example, a designer may model a system by adopting a service oriented approach. Before generating WSDL and corresponding Java implementation, she/he may define a WSDL model in order to refine her/his previous abstract service-oriented model.

Model transformation. The concept of intermediary model between abstract model and implementation code has been one of the main innovations of MDE. But such models are economically interesting if they are automatically generated. Model transformation engines like YATL (Patrascoiu, 2004) allow this kind of generation. But the abstract/concrete dimension must not be considered as the only one interest of model transformation. Model merging (Brunet 2006) is also relevant: for example we could imagine merging a LD pedagogical scenario with several SCORM individual learning paths in order to product a Claroline-specific model. UML templates, step-by-step model definition and aspect-oriented modelling (AOM) also belong to model transformation area and demonstrate its wide scientific interest.

The Model Driven Architecture (MDA 2003) from OMG consortium is the most famous MDE initiative. It currently insists on the abstract/concrete dimension.

Implementing pedagogical scenarios on e-learning platforms through MDE

The work we present is one of the BRICOLES project results (Caron, 2005) (Caron, 2006) whose main objective is to suggest MDE-based solutions to reintroduce teacher in e-learning courses design. This work deals with BUILDING from a LD scenario one corresponding learning environment in Claroline, Ganesha or Moodle. The designer (instructional designer or teacher) follows a four-step production cycle (see Figure 6):

1. She/he defines the LD scenario.
2. She/he chooses the targeted platform, the way to map her/his scenario and get a model of her/his scenario which is expressed with the concepts of the platform. We called this model a *technological model* or a platform specific model (PSM).
3. She/he refines the technological model because it was not possible to express some subtleties related to the final realisation in the LD scenario.
4. She/he uses a deployment tool to implement the technological model on her/his platform.

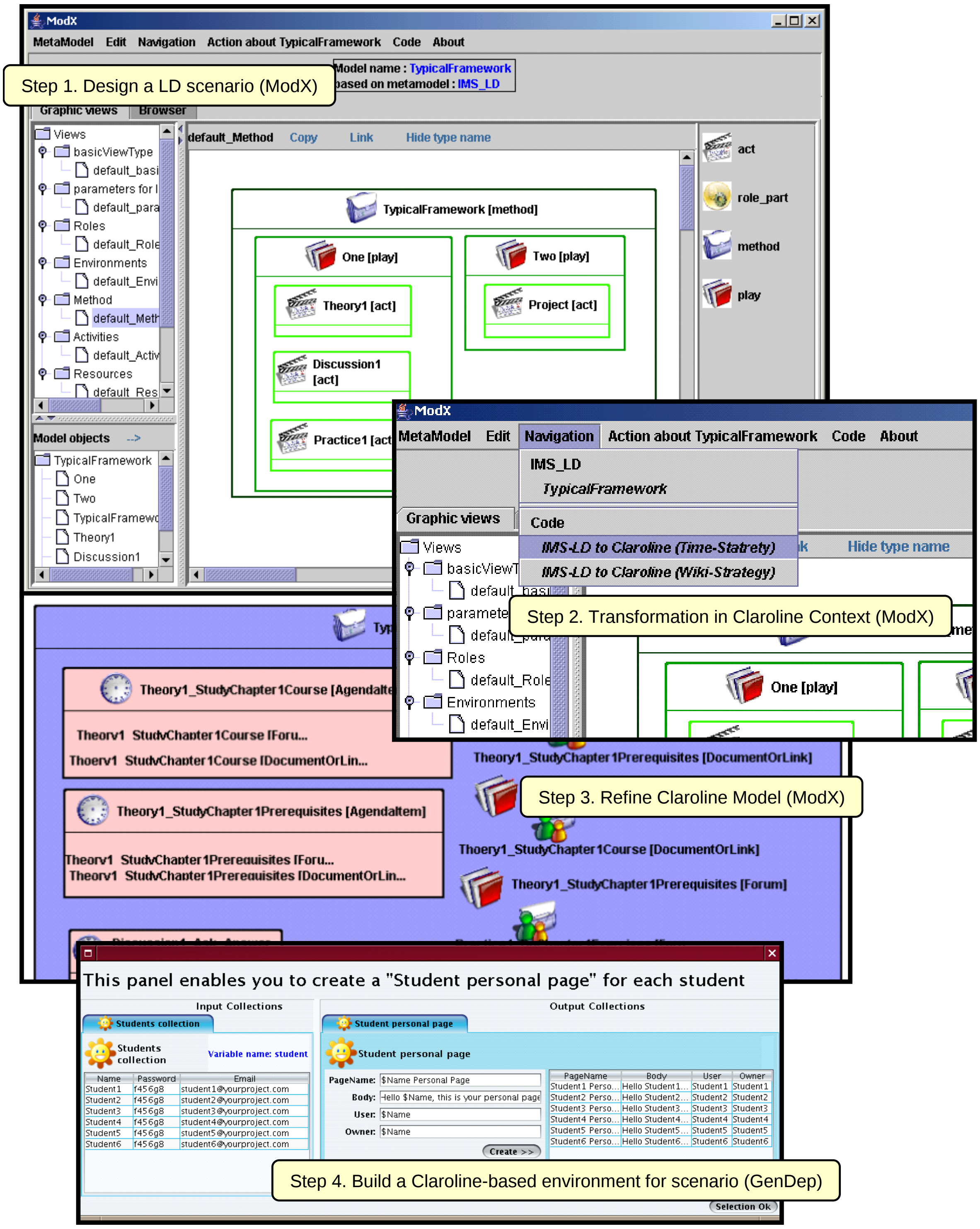
For the first step, the designer may use any LD authoring tool or our modelling tool ModX. ModX is a graphic tool used to create both model and MOF-based metamodel (ModX 2006). For pedagogical scenario concern, we have defined the MOF metamodel for IMS-LD and associated a graphical syntax to it. We have made a LD dedicated version of ModX – with functions which are related to import/export LD native documents - where we have also included the Best-Practices methodology in order to assist designer when she/he defines a LD model (ICALT 2006).

For the second step, designer must use ModX in order to benefit from its transformation model abilities. The designer may transform her/his scenario into

Claroline, Moodle or Ganesha technological context. For each platform, there is one or more kinds of mapping. The resulting model is expressed in a metamodel related to the selected platform. For this, we have defined one MOF metamodel for each platform.

For the third step, the designer goes on using ModX because it allows editing and so refining the previous technological model.

Finally, the designer runs GenDep (GenDep 2006) in order to implement the previous refined technological model on her/his platform. GenDep is a Java software which is based on the XMI library of ModX and on SOAP communication. It automatically builds a SOAP communication layer for a platform from its metamodel. This layer is used when GenDep builds one learning environment from one model from previous metamodel. In other words, it will build a learning environment on an instance of Claroline, Ganesha or Moodle from the technological model. GenDep will interact with the designer to know which students and tutor(s) - already present in the platform – have to be associated to the created learning environment.


Step 1. Design a LD scenario (ModX)
Step 2. Transformation in Claroline Context (ModX)
Step 3. Refine Claroline Model (ModX)
Step 4. Build a Claroline-based environment for scenario (GenDep)
ModX
MetaModel Edit Navigation Action about TypicalFramework Code About
Model name : TypicalFramework
based on metamodel : IMS_LD
Graphic views Browser
default_Method Copy Link Hide type name
TypicalFramework [method]
One [play]
Two [play]
Theory1 [act]
Project [act]
Discussion1 [act]
Practice1 [act]
act
role_part
method
play
Views
basicViewType
parameters for
Roles
Environments
Method
Activities
Resources
Model objects
TypicalFramework
One
Two
Theory1
Discussion1
IMS_LD
TypicalFramework
Code
IMS-LD to Claroline (Time-Statrety)
IMS-LD to Claroline (Wiki-Strategy)
Theory1_StudyChapter1Course [AgendaItem]
Theory1_StudyChapter1Course [Foru...
Thoerv1_StudvChapter1Course [DocumentOrLin...
Theory1_StudyChapter1Prerequisites [AgendaItem]
Theorv1_StudvChapter1Prerequisites [Foru...
Theorv1_StudvChapter1Prerequisites [DocumentOrLin...
Theory1_StudyChapter1Prerequisites [DocumentOrLink]
Thoery1_StudyChapter1Course [DocumentOrLink]
Theory1_StudyChapter1Prerequisites [Forum]
Create >>
Selection Ok

This panel enables you to create a "Student personal page" for each student

Input Collections

Students collection

Variable name: student

Name	Password	Email
Student1	f456g8	student1@yourproject.com
Student2	f456g8	student2@yourproject.com
Student3	f456g8	student3@yourproject.com
Student4	f456g8	student4@yourproject.com
Student5	f456g8	student5@yourproject.com
Student6	f456g8	student6@yourproject.com

Output Collections

Student personal page

PageName: $Name Personal Page
Body: Hello $Name, this is your personal page
User: $Name
Owner: $Name

PageName	Body	User	Owner
Student1 Perso...	Hello Student1...	Student1	Student1
Student2 Perso...	Hello Student2...	Student2	Student2
Student3 Perso...	Hello Student3...	Student3	Student3
Student4 Perso...	Hello Student4...	Student4	Student4
Student5 Perso...	Hello Student5...	Student5	Student5
Student6 Perso...	Hello Student6...	Student6	Student6

Figure 6. Bricoles project: 4 steps to automatically build a claroline environment from a LD scenario

Other benefits

Automatic building of learning environment is not the only one benefit of our approach. First, as models become productive, they also become more relevant in actual distant learning practice. So it may contribute to a better use/reuse of models and, as a result, to a better distant learning experience sharing. Second, we have noted that tutors have difficulties to adopt a new e-learning web platform. From our first experiences with Bricoles, we have seen that tutors grasped faster a new platform if their own course was implanted into it with, in addition, an overview of it thanks to the corresponding technological model. Finally, we have also noted that, after one or two uses of the complete production cycle, tutors generally prefer to work directly with technological model. It may seem to be a problem. But, it may be easier to define computational models related to a (accessible) platform than IMS-LD models which are complex. They somehow get learning staff used to define *computational models*. However we think that technological models may be more relevant than we planned at the beginning. They provide a good flexibility point in our production cycle. We plan to study what may be the new practices brought by technological models.

EXECUTION AND RUNTIME EVOLUTION OF PEDAGOGICAL SCENARIOS

At the moment, there are very few solutions compatible with IMS-LD standard or at least targeted at pedagogical scenarios (Britain, 2004). This is particularly true when one considers the enactment of these scenarios which is considered in this part. In a first step, we will consider the architecture of IMS-LD enactment engines. Next, we will consider how flexibility can be provided by such engines so as to permit a continuous adaptation of the pedagogical scenario. Finally, we will see how enactment engines can be integrated into Learning Management Systems.

Enactment of Pedagogical Scenarios

The enactment of pedagogical scenarios requires two elements: *a learning design engine* that can interpret the IMS-LD scenario and keep track of the activities so as to provide the right activities with the related resources and services to the participants and a client or *player* that will provide a learning environment for the learners and tutors. The learning design engines can be developed in two ways:

- **Embedded in a Learning Management System** as in the .LRN platform (.LRN 2006). This approach has the benefit of permitting a tight integration with the LMS services which enables a rich learning environment, in terms of services and user interface. However, this means an ad hoc development that can hardly be reused in another setting.
- **As a standalone service** with well defined interfaces that permit its integration into a LMS. This approach provides the advantage of a reuse of the enactment service. However, since the service is generic, one has to take care of the integration issues both with the support services (e.g. chat, mail) and with the player.

In the following, we will concentrate more on the second type of approach based on examples such as the Coppercore engine and its extensions as well as the Cooperative Open Workflow engine. Based on these, we will delve into the technological basis and the means to provide flexibility. We will also cover the issues related to the integration into a LMS.

Developing a learning design engine: design and technologies

The design

Users of the pedagogical scenarios will interact with the player for different purposes: to get the activities they have to perform, to manage the scenarios… The player then relies on the learning design engine to manage these tasks. Hence, the LD engine should provide the following services:

- **Runtime services** to provide the right activities, resources and pedagogical services to the actors. The main interaction mode between the player and the engine is pull mode, i.e., the player retrieves the information from the engine.
- **Administration services** to provide the means to administrate and instantiate the pedagogical scenarios. This means defining the students and support staff participating in a particular instance, creating groups…
- **Monitoring services** to provide the means for a tutor to monitor the progress of the students/groups in a particular instance. This type of service is also interesting to be able to put in evidence problems related to the scenario itself, which would trigger a reengineering of the scenario.

The engine must take care of the models but also of the instances (called *runs* in Coppercore). This corresponds to the actors and groups, the resources but also to keep track of the position of everybody in the scenario (current activities…). Since a pedagogical scenario can last for a long time (e.g., weeks or months), the engine should take care of the persistence of these information.

Technologies

Since the engine must be integrated into an existing learning environment, it should provide interfaces for this. Middleware technologies such as CORBA (Common Object Request Broker Architecture), Java RMI (Remote Method Invocation) and DCOM (Distributed Component Object Model) provide the basis for a remote integration and the means to define the interfaces. However, while solving the integration problem, these technologies do not support the developer in providing robust software since every service such as persistence or security has to be managed explicitly. Component standards have appeared to solve the problem by a declarative management of these non functional properties. The definition and configuration of the services to provide is then used by a component container to provide a suitable runtime environment. The Enterprise JavaBeans (EJB) component model has been used to develop both CopperCore and COW, so we will provide a brief description for it. The EJB model defines three types of components:

- **Entity beans** provide an object oriented view of the persistent information managed by the software. Each modification to an entity is persisted in a permanent storage, typically a relational database.
- **Session beans** are associated to a client and can be seen as the providers of the service. Sessions can be stateless or stateful. In the latter case, they can manage the interactions with the user/client. Sessions typically provide the business logic and manipulate the entities according to the business rules.
- **Message Driven beans** (MDB) allow asynchronous communication based on the Java Messaging Service (JMS) standard. Upon receipt of a message, the MDB can interact with sessions or entities.

During the deployment, the container uses an XML descriptor provided with the beans to provide a suitable runtime environment. This deployment descriptor defines the configuration of the beans and non functional services. After deployment the beans interfaces are available through Java RMI and eventually as Web Services for the session beans. Indeed, it is still difficult to integrate third party services which may not be developed with the same model. To cope with this heterogeneity problem Web Services are the current solution. The principle is to use existing standards like HTTP or XML to realize remote procedure call. SOAP (Simple Object Access Protocol) (SOAP, 2003) allows an easy integration by using an XML language to realise the method invocation. The exposed interfaces are described with an XML language called WSDL (Web Service Description Language) (Christensen, 2001).

Based on this, one can envision an enhanced reuse of existing services which can be combined to provide new services. This is the Service Oriented Architecture (SOA) vision. One example in the domain of technology enhanced learning is the E-Learning Framework (Wilson et al, 2005). This framework intends to standardise all the useful services of a learning environment so as to allow for a greater reuse of existing implementations. Since the LD engine is an enactment service, it should be provided as a Web Service to enhance integration. This is the case of both Coppercore and COW.

Existing implementations

In this part, we will focus on CopperCore and SLeD implementation and the Cooperative Open Workflow engine which are the main examples of standalone enactment services. Among the few implementations we can also mention works presented in (Chen et al, 2005) and (Hagen et al, 2006). The former follows the same architectural principles as the ones we will see hereafter. Based on the implementation work they raise questions about the interpretation of some IMS-LD constructs. The latter is also a prototype to question the implementation of the standard.

The Coppercore Engine

The CopperCore engine is one of the results of the aLFanet (Active Learning for Adaptive Internet) IST project. The objective is to provide a reference implementation of IMS-LD. Coppercore handles the three levels of IMS-LD, that is static (level A) and dynamic scenarios based on properties and conditions (level B) as well as notifications (level C). Level B and C support are in fact a contribution from the SLeD project presented hereafter. Coppercore has been built as a proof of concept and demonstrator and for this reason it provides a command line and a web based client that should not be useful in an embedded setting. It is based on J2EE (Java 2

Enterprise Edition) technologies and provides three levels of interfaces for its integration into a LMS (cf. Figure 7):

- A set of java objects that shield the developer from the intricacies of J2EE development, in particular the access to the session beans' interfaces.
- The Remote Method Invocation (RMI) interfaces provided by the Enterprise JavaBeans component framework which is also restricted to java environments.
- The Web Service interfaces based on WSDL that can be used at a distance and in a heterogeneous setting.

For these three levels, we will find a learnflow interface corresponding to the runtime services describe beforehand and an administration interface.

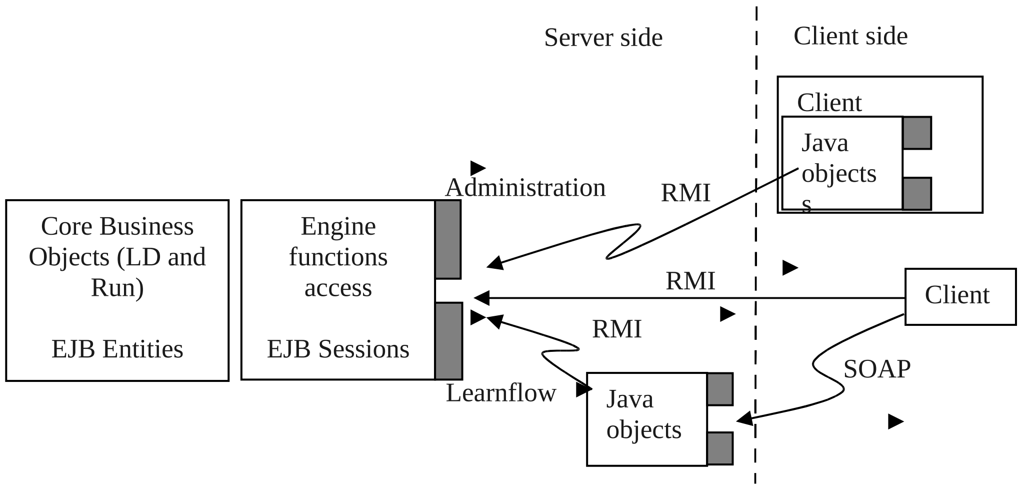


Figure 7: CopperCore architecture and interfaces

The SLED Project

The Service-based Learning Design system (SLeD) project started in 2004 and had multiple follow-up to extend the work (SLeD 2) and to assess its use in real settings (SliDe project, Barret-Baxendale (2005)). In addition to contributing to Coppercore, the aim of the project was to provide another implementation that enable the integration of new services which can be used in the pedagogical scenarios (e.g., QTI engine, e-portfolio…). This work is made in the spirit of the Service Oriented Architecture (SOA) and E-Learning Framework. The objective of the project was to explore the tension between generic service description as used in IMS-LD that facilitate reuse of the design and a rich service description which would provide a higher integration and functionality. Indeed, to fully take the benefit of SOA, it is necessary to have standardised services which truly enables to switch implementations transparently and permit to define a service with few information (identifier and configuration data). This has lead to the proposition of the architecture presented in Figure 8 as an output of the SLeD2 project. The IMS Content Package provides the definition of the needed services within the IMS-LD scenario. This is used in the integration layer to manage the relevant services implementations. The integration layer is responsible for calling the right service and provide the input data based on the performed activities and user inputs as well as catching notifications from the services and forwarding to other services as necessary (e.g., to notify the end of an activity to the LD engine). Since there is scarcely a standard for the service, an adaptation layer is necessary to accommodate different implementations with different APIs. For each service type (e.g., forum, e-portfolio…) it is necessary to define a common interface and for each service implementation, one has to develop a translator that will implement this interface and make the right calls on the actual service.

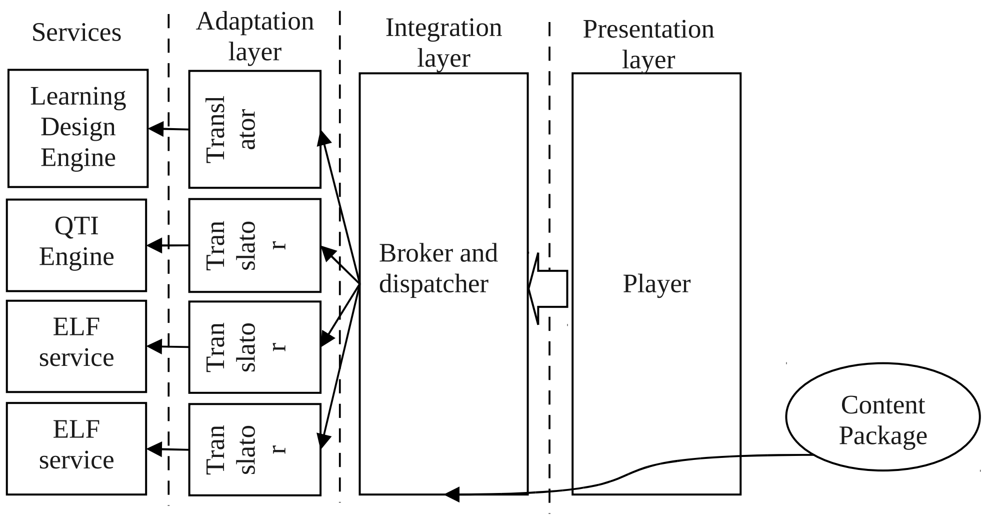


Figure 8: SLeD architecture for a service oriented LD

The adaptation service is only necessary because of the low maturity of the field regarding standardisation. In the long term, the situation should evolve driven by the work done by standardisation bodies like the IMS consortium. An example of this is the IMS Enterprise Services Specification (IMS, 2004).

The COW project

The Cooperative Open Workflow (Peter, Vantroys 2005) developed by the TRIGONE laboratory since 2001 aims at providing a bridge between the concept of activity in the EMLs (Educational Modelling Language) and in workflows systems in order to enact learning scenarios. COW has been built using workflow technologies with the idea that learning scenarios are only a certain kind of processes. This workflow based enactment service has been design around the following principles:

- To design a platform centred on the user (Bourguin & al, 2001) which allows a continuous adaptation to provide the best learning experience.
- To recognise the importance of the scenario model and to make it constantly available and easy to handle.
- To provide flexibility mechanisms allowing the realization of the preceding principles.

To provide an easy integration of this enactment service into a LMS and to foster reusability the development is based on existing standards at both technical and pedagogical levels. As the project begun before the rise of IMS-LD, the main concepts come from the workflow world but there are several links between classical workflow processes and learning scenarios. Based on a metamodelling approach, we have designed a translator that can map IMS-LD concepts into workflow concepts (Vantroys, Peter, 2003), which shows the two approaches are compatible.

At the technical level, the development is based on J2EE standards in particular using EJB components and useful interfaces are also published as Web Services to enhance integration. The implementation of the service follows the Workflow Management Facility (WMF) specification (WMF, 2000) of the Object Management Group (OMG, 2006) and the workflow reference model of the Workflow Management Coalition (WfMC, 1995). The WMF describes the architecture of a workflow engine by defining the interfaces of the different objects. Each object represents a concept in a workflow model (process, activity …).

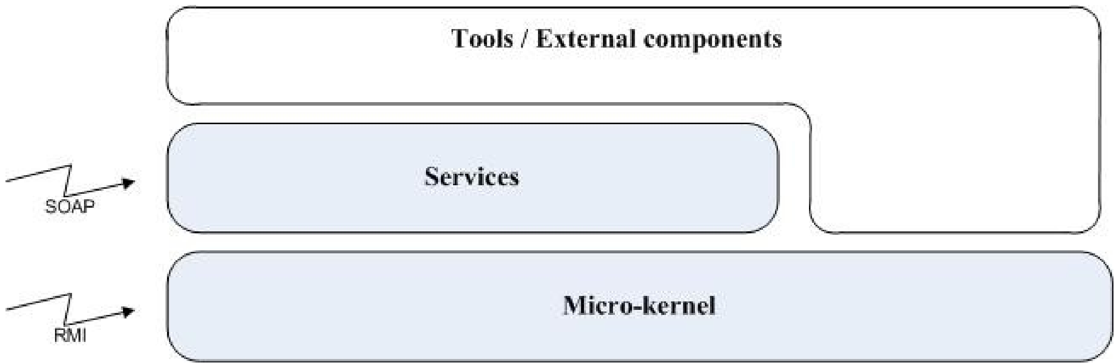


Figure 9: COW global architecture

The global architecture of COW is depicted in Figure 9 the technical architecture is show in Figure 10. It is decomposed into three specific layers:

- The microkernel represents the heart of our system. The philosophy of this type of architecture is to offer a minimal set of specific functions (Tanenbaum, 1994). This minimalism allows this architecture to be very flexible because it implies few constraints to add dedicated functionalities at higher level. Within the framework of a workflow system the basic functionalities relate to the rudimentary management of the processes and the activities, the flow of data between activities and event notification that enable external systems to be aware of the enactment evolution. This minimalist approach was also employed for the design of *micro-workflow* (Manolescu, 2000). For COW that corresponds to the implementation of the WMF interfaces (activity, process…), made with EJB entities to ensure the persistence of the data (see Figure 10).
- The services layer offers an additional level of abstraction compared to the core by offering added value services which correspond to the runtime, administration and monitoring services. These interfaces abstract the complexity of the kernel based on the façade design pattern (Gamma & al, l995). The basic services are implemented as EJB sessions.
- The external tools and components correspond to the clients of the services offered by the platform. This level corresponds to the player and offers user

interfaces to the monitoring tools, model management, and activities for the learners and staff.

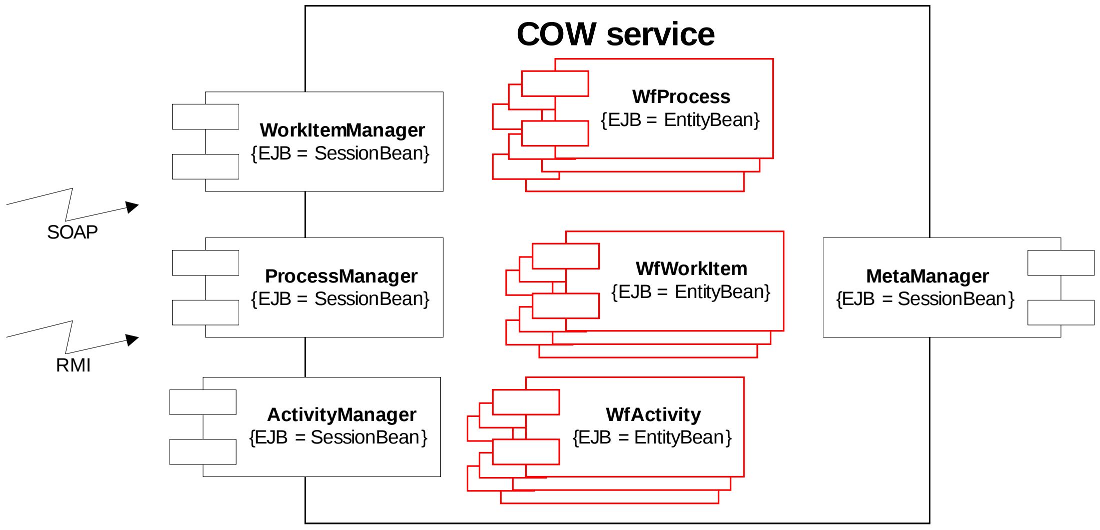


Figure 10: COW internal architecture

Support for the flexible enactment of pedagogical scenarios

Many adaptations to the pedagogical scenario will be done at design time when one reuses a scenario. The adaptations to the actual context may also be done during deployment (e.g., to link a service to a specific implementation). However, even with these late adaptations, one cannot envision all the events that can occur during the execution of the scenario nor the particular difficulties of the learners or differences in the way they approach the learning activities (e.g. learning styles, knowledge level…). There are even some times when one does not know beforehand all the activities needed. Because of this it is necessary to support runtime modification of the scenario.

The aLFanet project which is at the origin of CopperCore has also developed an adaptation component for IMS-LD. The runtime adaptations provided are targeted at guiding the users to improve their learning experience according to a constructed user model (Boticario et al, 2004). This is based on machine learning to detect user categories and to keep the memory of the successful interactions related to the different categories. The learning part relies on a multi-agent system. The adaptation of the scenario relies on modifications to the properties in level B to change the course of the activities.

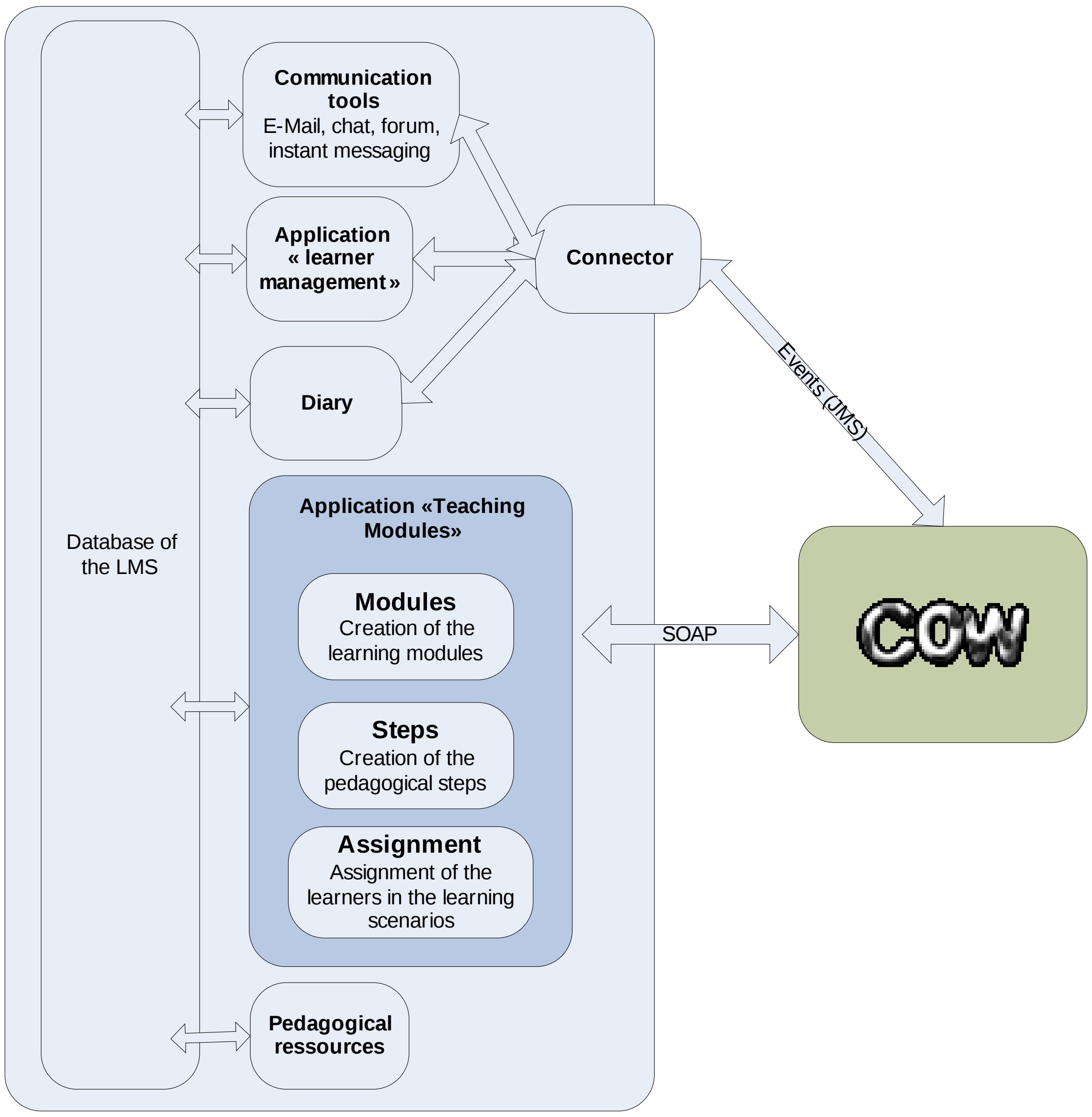
Another approach which is more related to the context of learning rather than the learners themselves is presented in (Zarraonandia et al, 2006). Design patterns and aspect oriented programming are used to extend the Coppercore implementation with adaptation features. Adaptations are defined using a language that describes the manipulations of the scenario. To this end, they have defined the set of modifications that can be achieved for the elements in level A and B (e.g. change activity title, add an activity, change a property value…). Modifications are expressed within an adaptation command file which may be accompanied with a manifest that provides IMS-LD fragments used in the modifications. Adaptations can then be defined for specific contexts which will be triggered based on the acquisition of contextual information. However, for this approach to work, the considered context should be limited otherwise the combination of values and corresponding adaptations could not be handled.

The flexibility brought by COW is found according to two axes: the first relates to the possibility to modify the model of the scenario in the course of execution, the second relates to the possibility to modify the interpretation of the scenario (i.e., the behaviour of the engine). The first axis rests on mechanisms of introspection and intercession and the principle of open implementation (Kiczales *& al.*, 1991) which make it possible to handle the model in order to add/withdraw/modify activities and to modify the sequencing of these activities. One can also modify the activity model (e.g., changing tools or learning objects, modifying activities assignment to roles).

These changes can be realized for one learner or for a group of learners. The second axis of flexibility relates to the behaviour of the engine itself. Indeed, it is impossible for the programmer to foresee all the use cases of the system. The modification of the behaviour relies on the use of the strategy design pattern to be able to provide ad hoc behaviour depending on the context. It is then necessary to identify the flexibility points within the engine and to use the strategy pattern at these points. The way of managing the time constraints is a typical example of behaviour which one must be able to adapt. Indeed, the management of time is one of the other properties of COW. Two temporal concepts are provided: duration and deadlines. The first represents the minimal or maximum time to perform an activity. The second concept specifies the periods during which work must happen. In order to manage the calculation of the durations and the deadlines, it is possible to use various “strategies”, such as for example taking into account the weekends or not. One can thus dynamically adapt calculation according to the context of use and the wishes of the users. The strategy pattern is also used to allow various behaviours when constraints are violated, for example, to send an email to the learner or the tutor, or to validate in an automatic way the activity.

Integration of learning scenario enactment services in LMS

As indicated previously, there are few LMS integrating support for learning scenarios. The solution is thus to integrate a third party enactment service like CopperCore or COW in the LMS. From a technological point of view that does not seem to be a problem because these services have been designed to be integrated. They use in particular SOAP for the access to their API. By exploring this question a little, one note that it is not so simple because integration requires necessarily modifications of the LMS and it is also necessary to determine the desired level of integration, from slightly coupled to strongly integrated into the LMS. This choice will also depend on non-technical parameters. It will be directed by the level of knowledge of the users and by their level of adoption of IMS-LD and the learning scenarios approach. If the users have little knowledge and where one simply wishes to add a new functionality to the LMS without changing its architecture, a weak coupling can be selected. It is for example the case of (Bergren et al, 2005). They have integrated Coppercore player into Moodle in order to offer a new functionality to the users of Moodle. CopperCore player is regarded as a simple resource of the type “Web link”. Thus the users are redirected on the Web interface of the player when necessary. This approach is simple but presents disadvantages. The first one relates to the user interface which does not correspond to the traditional ergonomics of the LMS. The second point relates to the communication. The engine can use learning resources provided by the LMS, but it has no possibility to return information to the LMS. Because of these limitations, the authors plan a stronger integration in order to have a bidirectional communication. It is to obtain a better integration that (Harrer et al, 2005) had an approach based on the development of a communication module called “remote control component”, to integrate the CopperCore engine with Cool Mode (Pinkwart, 2003). They extended CopperCore to produce events related to the enactment. These events are listened by the LMS. Thus a bidirectional communication is possible. A third integration more strongly coupled is explained in (Vantroys, Peter, 2006). COW was integrated into a commercial platform, the “Campus Virtuel” of the French enterprise Archimed (Archimed 2006). The “Campus Virtuel” already contained a learning scenario service which did not fulfil all the requirements of the users, in particular in term of flexibility of the learning path. Here what was needed was a strong integration with bidirectional communication so as to maintain the existing modes of operation. There was also the additional constraint of not changing the user interfaces and to avoid to the maximum modifying the other existing functionalities. Figure 11 summarizes the integration. For the existing components managing the learning scenarios, a large development was carried out to replace the old service based on COM components. The integration was carried out by using SOAP. For the communication with the other services (chat, forum…), an event-driven type of connector was developed in the same spirit as the approach described above.



Communication tools
E-Mail, chat, forum, instant messaging
Application « learner management »
Diary
Connector
Events (JMS)
Database of the LMS
Application «Teaching Modules»
Modules
Creation of the learning modules
Steps
Creation of the pedagogical steps
Assignment
Assignment of the learners in the learning scenarios
SOAP
COW
Pedagogical ressources

Figure 11: COW integration in the Campus Virtuel

CONCLUSION

(Hummel et al, 2004, pp 125) conclude that "*…staff and educational developers can already benefit from the philosophy of IMS-LD by focusing on learners' activities and objectives...*" and that the final objective is the emergence of educational best practices expressed as reusable learning designs which can then be adapted for use in particular contexts and enacted in IMS-LD compliant environments. To achieve this vision, it is necessary to provide a suitable support to the designers of the learning activities and to have means to enact these learning designs. Moreover, since there are many good reasons for an evolution of the design it should be possible to adapt it even at runtime to accommodate specific contexts or learners' problems. However, the world of learning design and IMS-LD has not reached that maturity yet. In this chapter, we present the current state of support at both design time and runtime. Concerning design time, we have presented how one can mitigate the absence of IMS-LD support in most LMS while still benefiting from the effort of a careful design of the learning process. We have also presented an approach based on Model Driven Engineering to provide a better support for design and to ensure a faithful and easy implementation in a specific LMS.

Concerning runtime support for pedagogical scenarios, we have presented the design and technologies for an enactment engine and the main existing services. It is important to note that the main implementations comply with existing standards from the technical perspective and from the pedagogical perspective: EJB components, Web Services, IMS QTI… This approach provides the opportunity to take the benefit of existing tools and services, a good basis for reuse of the developed services and facilitates the integration with other services. This prepares the way for the uptake of the Service Oriented Architecture as seen in the E-Learning framework (Wilson et al, 2005). In the long term, this will allow the deployment of customised learning environment based on the integration of existing services. We have then presented some specific implementation targeted at providing a flexible runtime execution that enables the evolution of the scenario. In one case, the adaptation relies on IMS-LD properties within the level B which implies the range of adaptation is known beforehand. The approach of the two other examples is to enable an explicit manipulation of the model to change it at runtime. Finally, we have analysed on different examples the technical solutions for the integration of an enactment service into a LMS.

The work presented in the design and runtime supports both provide a contribution to the uptake of the Learning Design approach. They can be seen has two alternative views at the moment, but this is due to the low maturity of the existing platforms and tools. When enactment support will be more common, we will have integrated solutions to cover the whole lifecycle of the pedagogical scenarios. The last missing link which is still to build will be to feed runtime adaptations into the design tools so as to avoid loosing good evolutions and their rationale. This will enable a continuous evolution of the scenarios.

REFERENCES

.LRN (2006) .LRN platform web site http://www.dotlrn.org/, last visited September 2006.

Abouzahra, A. & Bézivin, J. & Didonet Del Fabro, M. & Jouault, F. (2005). *A Practical Approach to Bridging Domain Specific Languages with UML profiles*, In: Proceedings of the Best Practices for Model Driven Software Development at OOPSLA'05, San Diego, California, USA.

Adkins S. (2005), *Wake-Up Call: Open Source LMS*, ASTD's source for e-learning, Learning Circuits, http://www.learningcircuits.org/2005/oct2005/adkins.htm, October 2005.

ADL (2006). ADL web site for SCORM related specifications, http://www.adlnet.gov/scorm/, last visited June 2006.

Anemalab.org (2006) *Description de la plateforme Ganesha*, http://www.anemalab.org/ganesha/descriptif.htm, Last visited October 2006

Archimed (2006). Archimed web site, http://www.archimed.fr, last visited October 2006.

AUF (2006), *FOAD Formations Ouvertes et A Distance, Master pro (M2) E-Services*, http://foad.refer.org/rubrique59.html

Barrett-Baxendale M. & Hazelwood P. & Oddie A. & Anderson M. & Franklin T. (2005). SLeD Integration Demonstrator – Final Report, retrieved September 2006 from http://www.hope.ac.uk/slide/.

Berggren A. & Burgos D. & Fontan J.M. & Hinkelman D. & Hung V. & Hursh A. & Tielemans G. (2005). *Practical and Pedagogical Issues for Teacher Adoption of IMS Learning Design Standards in Moodle LMS*, Journal of Interactive Media in Education, 2005(02).

Blackboard (2006) *Blackboard Learning System*, http://www.blackboard.com/products/as/learningsys, Last visited October 2006.

Boticario, J.G. & Santos, O.C. & Barrera, C. & Gaudioso, E. & Hernandez, F & Rodriguez, A. & van Rosmalen, P. & Koper, R. (2004). *Defining adaptive learning design templates for combining design and runtime adaptation in aLFanet*, retrieved June 2006 from http://hdl.handle.net/1820/210.

Bourguin G., Derycke A., Tarby JC. (2001) *Beyond the interface: Co-evolution Inside Interactive Systems - A Proposal Founded on Activity Theory*. In Jean Vanderdonckt, Ann Blandford, et Phil Gray, editors, People and Computers XV - Interaction without Frontiers, pages 297 - 310, Lille, France, September 2001. ISBN: 1-85233-515-7.

Britain, S. (2004). A Review of Learning Design: Concept, Specifications and Tools, report for the JISC E-learning Pedagogy Programme.

Brunet, G &, Chechik, M. & Easterbrook, S. & Nejati, S. & Niu, N. & Sabetzadeh, M. (2006), *A manifesto for model merging*, In : Proceedings of the 2006 international workshop on Global integrated model management, International Conference on Software Engineering, Shanghai, China, May 22 - 22, 2006.

Caron, P.-A. & Derycke, A. & Le Pallec X. (2005). *The Bricoles project: support socially informed design of learning environment*, In: International Conference on Artificial Intelligence in Education (AIED 2005), Amsterdam, IOS Press, 2005, p 759 – 761.

Caron, P.-A. & Derycke, A. & Le Pallec, X. (2005). *Bricolage and Model Driven Approach to design distant course*, In: E learn 2005, world conference on E-learning in corporate Government, Healthcare & higher education, Vancouver, Association for the Advancement of Computing in Education (AACE), p 2856- 2864.

Caron, P.-A. & Le Pallec, X. & Sockeel, S. (2006). *Configuring a Web-based tool through pedagogical scenarios*, in IADIS Virtual Multi Conference on Computer Science and Information Systems (MCCSIS 2006)

Chen, M.C. & Chen, C.T & Cheng, Y.C. & Hsieh, C.Y. (2005). *On the Development and Implementation of a Sequencing Engine for IMS Learning Design Specification*, 5th IEEE International Conference on Advanced Learning Technologies, 2005, 5-8 juillet, Kaohsiung, Taiwan.

Christensen, E. & Curbera, F. & Meredith, G. & Weerawarana, S. (2001). Web Services Description Language (WSDL) 1.1. Retrived from http://www.w3.org/TR/wsdl.

Claroline (2006), *Claroline.net - Open Source eLearning*, http://www.claroline.net/, Last visited October 2006

CopperCore (2006). Coppercore platform home page, http://coppercore.org, last visited June 2006

CORDRA (2006). *What's "CORDRA"*™, http://cordra.net/docs/info/whatscordra/v1p00/info-whatscordra-v1p00.php, last visited June 2006.

Elearning (2006) *Master professionnel Ingénierie Pédagogique Multimédia, Elearning Actu*, http://www.elearning.fr/index2.php?option=content&task=view&id=46&pop=1&page=0, Last visited October 2006

ELF (2006). E-Learning Framework web page: http://www.elframework.org/ last visited in June 2006

Farail, P. & Gaufillet, P. & Canals, A. & Le Camus, C. & Sciamma, D. & Michel, P. & Crégut, X. & Pantel, M. (2006). *The TOPCASED project: a Toolkit in Open source for Critical Aeronautic SystEms Design*, In : Eclipse Technology eXchange workshop (eTX) at ECOOP 2006, Tuesday 4th July 2006, Nantes, France.

Gamma, E. & Helm, R. & Johnson, R. & Vlissides, J. (1995). *Design Patterns: Elements of reusable Object-Oriented Software*, Addison-Wesley, ISBN 0-201-63361-2.

GenDep (2006), *GenDep Web site*, http://noce.univ-lille1.fr/projets/ModX/index.php?option=com_content&task=view&id=15&Itemid=35

GMT (2006), *GMT Project*, http://www.eclipse.org/gmt/, Last visited October 2006.

Hagen, K. & Hibbert, D. & Kinshuk (2006) Developping a Learning Management System based on the IMS Learning Design Specification, Sixth International Conference on Advanced Learning Technologies (ICALT'06), 5-7 july 2006, pp420-424.

Harrer, A. & Malzahn, N. & Hoeksema, K. & Hoppe, U., (2005). *Learning Design Engines as Remote Control to Learning Support Environments*. Journal of Interactive Media in Education (Advances in Learning Design. Special Issue, eds. Colin Tattersall, Rob Koper), 2005/05. ISSN:1365-893X [jime.open.ac.uk/2005/05].

Hernández-Leo,D & Villasclaras-Fernández, E. D. & Asensio-Pérez, J. I.& Dimitriadis Y. &, Jorrín-Abellán, I. M.& Ruiz-Requies,I. Rubia-Av, Bartolomé (2006). COLLAGE: A collaborative Learning Design editor based on patterns. Journal of Educational Technology & Society. Vol 9(1), p 58-71.

Hummel, H. & Manderveld, J. & Tattersal, C. & Koper, R. (2004). *Educational Modelling language and learning design: new opportunities for instructional reusability and personalised learning*. International Journal of Learning Technology, Vol 1, No1, 2004.

IMS (2003). IMS, *IMS Learning Design Information Model - version 1.0*. IMS Global Learning Consortium, Inc., retrieved from http://www.imsglobal.org/learningdesign/index.html.

IMS (2004). IMS, Enterprise Services specification - Version 1.0 Final Specification, IMS Global Learning Consortium, Inc., retrieved from http://www.imsglobal.org/es/index.html

Kew, C. (2006), *Learning Design tools currently available or under development*, web page, UNFOLD, http://www.unfold-project.net/developers_folder/dev_resources/tools/currenttools/, Last visited October 2006.

Kiczales, G. & des Rivières, J. & Bobrow, D. (1991). *The art of the Metaobject Protocol*. The MIT Press. ISBN: 0-262-61074-4

LTSC (2002). IEEE LTSC, Draft Standard for Learning Object Metadata, IEEE 1484.12.1-2002.

Koper, R. & Tattersall, C. (2005), *Learning Design – a handbook on Modelling and Delivering Networked Education and Training*, Springer-Verlag Berlin Heidelberg 2005.

Le Pallec, X. & Renaux, E. & Moura, C. O. (2005). *ModX*, Tools Exhibition in European Conference on Model Driven Architecture - Foundations and Applications, Nuremberg, Germany, November 2005.

Le Pallec, X. & de Moura Filho, C. & Marvie, R. & Nebut, M. & Tarby, J.-C. (2006), *Supporting generic methodologies to assist IMS-LD modeling*. In: The 6th IEEE International Conference on Advanced Learning Technologies (ICALT'2006), July 5-7, 2006, Kerkrade (The Netherlands), pp 923 – 927.

MagicDraw (2006), *MagicDraw*, http://www.magicdraw.com/, Last visited October 2006

Manolescu, D. A. (2000). *Micro-Workflow: A Workflow Architecture Supporting Compositional Object-Oriented Software development*. Ph'D Thesis, University of Illinois. http://micro-workflow.com/PhDThesis.

Marvie, R & Nebut, M. (2006), *Processus de modélisation incrémentaux*, in 2nd Journées sur l'Ingénierie Dirigée par les Modèles (IDM'06), Lille, Juin 2006.

McAndrew, P. & Woods, W.I.S. & Little, A. & Weller, M.J. & Rob Koper, R. & Hubert Vogten, H. (2004). *Implementing Learning Design to support web-based learning*. AusWeb 2004, Gold Coast, Australia 3-7 July 2004.

McAndrew, P. & Nadolski, R. & Little A. (2005). *Developing an approach for Learning Design Players*. Journal of Interactive Media in Education (Advances in Learning Design. Special Issue, eds. Colin Tattersall, Rob Koper), 2005/14. ISSN:1365-893X [jime.open.ac.uk/2005/14].

MDA (2003) *MDA Guide Version 1.0.1*, http://www.omg.org/docs/omg/03-06-01.pdf

ModX (2006), *ModX Web site*, http://noce.univ-lille1.fr/projets/ModX/.

Moodle (2006) *Moodle - A Free, Open Source Course Management System for Online Learning*, http://moodle.org/, Last visited October 2006.

OMG (2006) Object Management Group official web site, http://www.omg.org last visited October 2006.

Paquette, G. (2005). *The IMS Learning Design Conceptual Model*, http://www.licef.teluq.uquebec.ca/gp/docs/Article%20EML-MISAedited.doc

Patrascoiu O. (2004), *YATL:Yet Another Transformation Language*, In : the 1st European MDA Workshop, MDA-IA, University of Twente, the Nederland, January 2004.

Perez C. E. (2006), *Open Source Model Driven Translators Written in Java*, Manageability Website, http://www.manageability.org/blog/stuff/open-source-model-translators-java/view.

Pernin, J.-P. & Lejeune, A. (2006), *Models for the re-use of scenarios of training*, IFIP - WG 3.1, 3.3, & 3.5 Joint Conference, "Imagining the future for ICT and Education" - 26th-30th June 2006 in Ålesund, Norway.

Peter, Y. & Vantroys, T. (2005). *Platform support for pedagogical scenarios*, Educational Technology and Society Journal, International Forum of Educational Technology & Society – ISSN 1176-3647, 8(3), 122-137.

Pinkwart, N. (2003). *A Plug-In Architecture for Graph Based Collaborative Modeling Systems*. In U. Hoppe, F. Verdejo & J. Kay (eds): Proc. Of Artificial Intelligence in Education, Amsterdam, IOS Press.

Schneider, D. & Synteta, P. & Frété, C. & Girardin, F. & Morand, S. (2003) *Conception and implementation of rich pedagogical scenarios through collaborative portal sites: clear focus and fuzzy edges*, ICOOL 2003 - International Conference on Open and Online Learning December 7-13, 2003 University of Mauritius.

SOAP (2003) SOAP specification, http://www.w3.org/TR/2003/REC-soap12-part1-20030624/, last visited October 2006.

Tanenbaum, A. (1994). *Systèmes d'exploitation :Systèmes centralisés, systèmes distribués*. Informatique Intelligence Artificielle, InterEdition, Paris. ISBN :2-7296-0706-4.

Tempus (2006) *TEMPUS MEDA Programme - Lebanon*, The European Union's Cooperation with Lebanon - Programmes and Projects 2005-2006, Call for proposals 2003 - International distance training e-services, http://www.dellbn.cec.eu.int/en/eu_and_lebanon/project4.htm, Last visited October 2006.

UNFOLD (2005), *Atelier sur les usages et les outils d'IMS Learning Design*, UNFOLD Paris Workshop April 2005.

Vantroys, T. & Peter, Y. (2003). *COW, a Flexible Platform for the Enactment of Learning Scenarios*, 9th Conference Groupware (CRIWG 2003), Springer-Verlag LNCS 2806 France, 2003

Vogten, H. & Martens, H. & Nadolski, R. & Tattersall, C. & van Rosmalen, P. & Koper, R. (2006). *CopperCore Service Integration - Integrating IMS Learning Design and IMS Question and Test Interoperability*. Sixth International Conference on Advanced Learning Technologies (ICALT 2006). 05-07 July 2006 Page(s):378 – 382

WebCT (2006) *WebCT.com*, http://www.webct.com/, Last visited October 2006.

Weller, M. & Little, A. & McAndrew, P. & Woods, W. (2006). *Learning Design, generic service descriptions and universal acid*, Educational Technology & Society Journal, 9 (1), 138-145.

Wilson, S. & Olivier, B. & Jeyes, S. & Powell, A. & Franklin, T. (2005) A Technical Framework to Support e-Learning, retrieved from http://www.elearning.ac.uk/frameworks/resources

Workflow Management Coalition (1995) ”The Workflow Reference Model”, WfMC-TC-1003, Version 1.1, 19 january 1995. http://www.wfmc.org/.

WMF (2000) Object Management group. ”Workflow Management Facility”,Version 1.2, http://www.omg.org/technology/documents/formal/workflow_management.htm last visited October 2006

Yongwu M. (2005). *Enabling learning designers to model dynamic learning processes*. Fifth IEEE International Conference on Advanced Learning Technologies, (ICALT 2005). 5-8 July 2005 Page(s):399 - 401

Zarraonandia, T. & Dodero, R.M. & Fernandez, C. (2006), *Crosscutting Runtime Adaptations of LD Execution*, Educational Technology & Society Journal, 9 (1), 123-137.