Rédacteurs	PL – NEM
Relecteurs	
Date	15/07/15
Référence	GRAPHIT-D2.4
Version	01/01/00

Rapport sur les VIDLs et les éditeurs graphiques

D2-6

■ Historique du document

Version	Date	Auteurs	Modifications
0.1	12/01/12	PL	Plan
0.2	06/03/12	PL, EL	Plan++ & contenus
0.3	24/04/12	PL	Plan revu (comparaisons inversées) & contenu ajouté
0.4	18/06/12	PL	Révision « forme » & contenu ajouté
0.5	25/09/13	PL	Contenu ajouté
0.6	31/10/14	PL	Remise en forme & contenu ajouté
0.8	16/06/15	NEM	Remise en forme & contenu ajouté
01/01/00	20/09/15	PL	Contenu ajouté et finalisation

■ Table des matières

Table des figures

1 Objectifs de ce document

Ce livrable sur les VIDLs a pour objectif d'étudier les VIDLs actuels afin de :

- détecter et approfondir ceux en relation avec les *Learning Management Systems* (LMS) ou plateformes de formation à distance qui sont au centre du projet GraphiT ;
- détecter ceux répondant aux besoins identifiés dans le projet :
  - binding (ou formalisation) vers un format informatique formel (type XML) directement ou indirectement (par export/transformation) vers un formalisme en relation avec les LMS (standards ou formats propriétaires de LMS)
  - spécification formelle de la syntaxe abstraite du langage sous une forme proche de celle des méta-modèles
  - outils-auteur/éditeur concret exploitant le VIDL.
- identifier vers quels types de VIDL centrés LMS le projet va se diriger.
- Identifier les éléments de notation et de construction « courants » que devront proposer les VIDLs du projet.

2 Introduction aux VIDLs

2.1 Définition d'un VIDL

VIDL signifie ***Visual Instructional Design Language***. Il s'agit donc, littéralement, d'un **langage de conception pédagogique** de nature **visuelle.**

L'enjeu et l'origine des VIDLs est décrit dans (Rogriguez et al., 2010). *« One of the main barriers disconnecting education from informatics and computer science is that teachers and educators are **rarely able to describe learning activities as clear-cut and precise processes and workflows**. In other words, **they tend not to formalize their activities to the degrees to which computational scientists and engineers are used to**, as they are forced to transform ideas and projects into the precise and logical world of machines. Nevertheless, teachers and educators actually design learning experiences ».*

Les VIDL sont donc à la fois des outils pour les enseignants/concepteurs comme pour les développeurs de solutions logicielles éducatives : *« **an intermediate step bringing the formalization of teaching and learning activities closer to the development and deployment of digital tools to support and facilitate it.** »* (Rogriguez et al., 2010).

Il convient donc de rappeler ce qu'est un langage de modélisation pédagogique et ce que l'on peut considérer, ou pas, comme visuel lorsque l'on parle de ce type de langage.

2.2 Langage de conception pédagogique

(Goodwin, 1994) définit un langage de conception, au sens large du terme, comme *"**a set of abstractions used to give structure, properties, and texture to solutions of design problems; Design languages provide building blocks for designs"**. Goodwin précise que tout langage de conception manipule différents termes : "actors, actions, concepts, types of relation,*

composite objects, qualities, and properties".

Pour **Gibbons (2003)** toute conception pédagogique s'appuie sur une architecture en 7 couches principales, identifiées selon une décomposition fonctionnelle de la situation à concevoir, plutôt qu'une approche orientée processus : ***content, strategy, controls, message, representation, media-logic, and data management*** **(Gibbons and Rogers, 06)**. Pour chacune de ces couches il existe de nos jours plusieurs langages de conception pédagogique basés sur des approches de conception différentes pour la même fonction. La multiplicité des langages de conception pédagogiques disponibles montre également la rapidité à laquelle opèrent les changements technologiques dans le monde éducatif : les méthodes, processus, théories, outils, styles sont intégrés à ces langages.

2.2.1 Educational Modeling Languages (EML)

Le CEN/ISSS a retenu la dernière proposition EML-OUNL, comme définition des langages de modélisation pédagogique : « *An EML is a semantic information model and binding, describing the content and process within a « unit of learning » from a pedagogical perspective in order to support reuse and interoperability* » **(Rawlings et al., 02)**. Un langage de modélisation pédagogique est donc « un modèle d'information et d'agrégation sémantique décrivant les contenus et processus engagés dans une "unité d'apprentissage" selon une perspective pédagogique et dans le but d'assurer la réutilisabilité et l'interopérabilité ». Ces langages ont la particularité de proposer un modèle d'information (décrivant les concepts, leurs relations et leur sémantique) ainsi qu'une correspondance (*binding*) vers une expression formelle de ce modèle dans un langage compréhensible par l'ordinateur (généralement réalisée en XML). Un EML décrit donc formellement des ressources éducatives et/ou des scénarios pédagogiques.

(Dessus and Schneider, 06) identifient les objectifs suivants pour les EMLs :

- Définir des scénarios pédagogiques ;
- Échanger des unités d'apprentissage (learning objects, scenarios)
- Exécuter une unité d'apprentissage sur une plateforme dédiée (LMS)
- Esquisser, concevoir, planifier, discuter des scénarios pédagogiques.

La typologie des EML s'appuie sur les critères suivants :

- Formalisme : strictement formel (exemple d'une grammaire XML) *versus* semi-formel (par exemple à travers des diagrammes UML ou une simple description verbale).
- Exécutabilité: les modèles produits sont non-interprétables ou bien sont interprétables / exécutables (avec ou sans étapes intermédiaires) ou les deux.
- Statut : standard formel / *standard-like* / expérimental.
- Couverture : plutôt global (on parle parfois de *couvrant*) / très spécialisé / entre deux.
- Orientation pédagogique (on parle de stratégie ou de « courant » pédagogique).

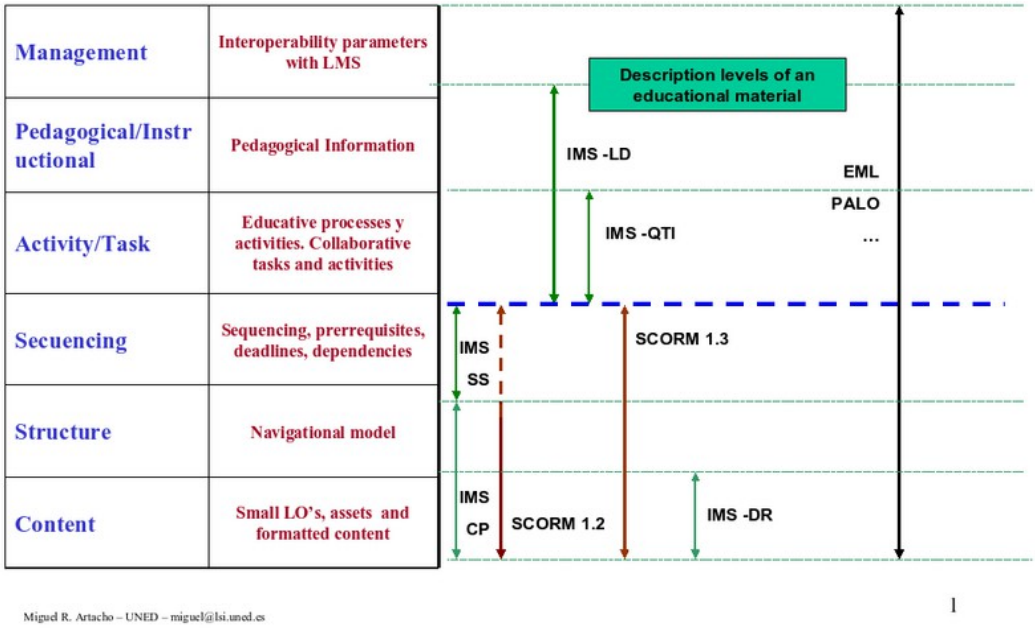


Figure 1: Positionnement des premiers EMLs par rapport aux autres standards de l'époque comme SCORM (source http://edutechwiki.unige.ch/en/Educational_modeling_language)

Les EMLs sont historiquement une réponse au manque de langages pédagogiques prenant en compte les activités pédagogiques (voir positionnement des premiers EMLs : EML(-OUNL), PALO, et le standard de facto IMS-LD basé sur EML-OUNL).

On peut donc considérer les VIDLs comme des EMLs particuliers moins enclins à proposer une formalisation strictement formelle ni une exécutabilité des modèles produits, mais au contraire à exploiter les valeurs ajoutées de l'utilisation d'une notation visuelle.

2.2.2 Langage de conception pédagogique Visuel

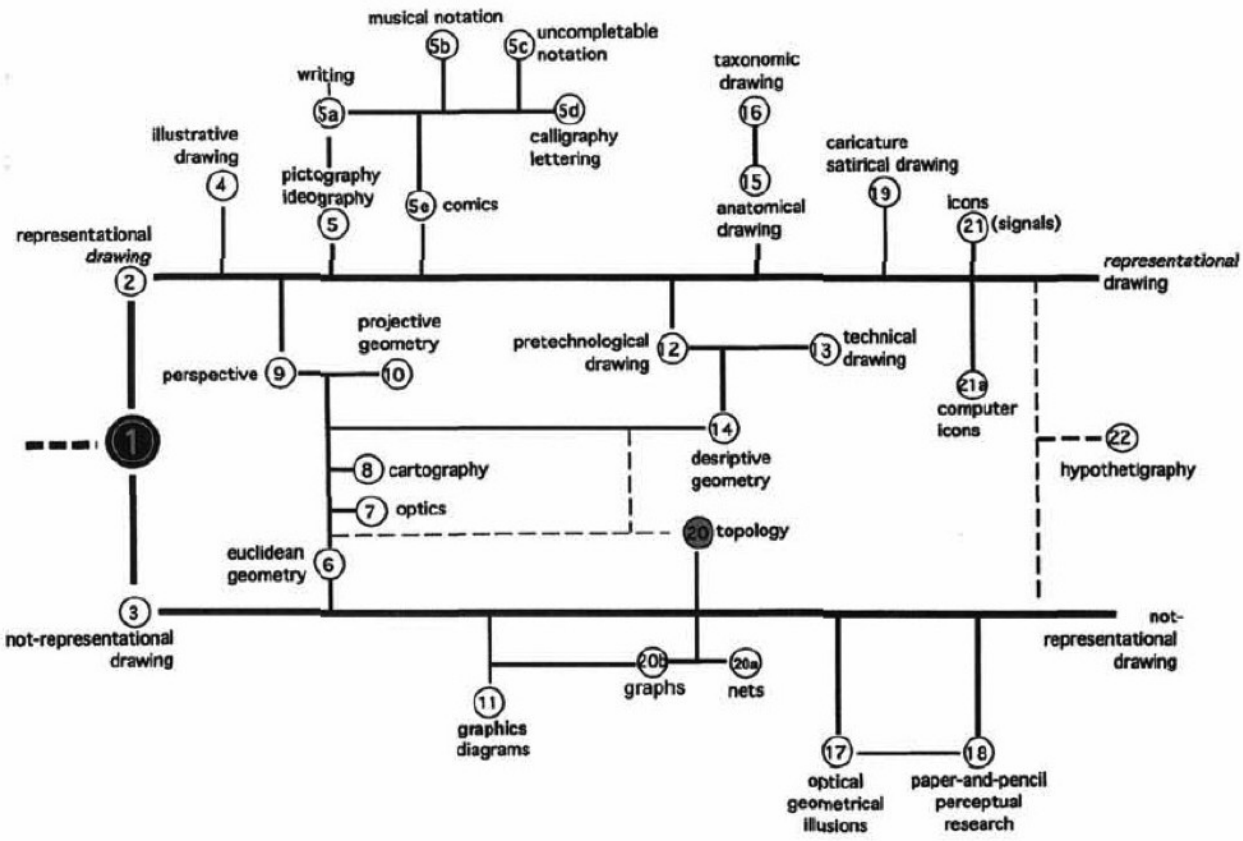


Figure 2: Taxonomie des descriptions visuelles

La figure suivante de **(Massironi, 02)** présente une taxonomie des descriptions visuelles dans le champ disciplinaire général de la conception.

On peut donc remarquer qu'il y a de nombreuses notations possibles qui sont *visuelles.* Comme nous le verrons par la suite, la plupart des VIDLs offrent une notation de type **diagramme graphique** (numéro 11 dans la figure précédente) avec toutefois des représentations de type **icônes** (numéro 21a) pour les nœuds.

2.3 VIDL et éditeurs

Un VIDL nécessite des outils concrets pour permettre à ses usagers (praticiens, ingénieurs pédagogiques, experts du domaines, etc.) de décrire ou spécifier des modèles de conception pédagogiques. Ces outils sont parfois appelés outils-auteurs, éditeurs, éditeurs graphiques, environnement-outil, *notation systems*, etc. Ils désignent tous l'environnement logiciel conçu et développé pour permettre aux usagers de réaliser les modèles.

Certains VIDLs revendiquent un éditeur associé spécifique déjà disponible, d'autres VIDLs ne sont pas encore outillés ou sont en cours de l'être. Aussi certains éditeurs sont présentés au premier plan sans mettre l'accent sur le langage implicite qu'ils exploitent et utilisent.

Il convient donc d'aborder et d'analyser également les éditeurs visuels de conception pédagogique même lorsque ceci ne référencent pas explicitement un VIDL.

2.4 Usages/valeurs ajoutées et inconvénients des VIDLs

Figl and Derntl recensent dans **(Figl & Derntl, 06)** les usages suivants pour les VIDLs :

- Better **communication** with less misunderstanding between experts due to consistent terminology.
- More intuitive **understanding** than a textual description through reduction of complexity.
- Educational requirements for tools to be used in specific e-learning settings can be defined more **easily.**
- Investigation and diagnosis of different e-learning settings and comparison with respect to main course design principles, e.g. the alignment of face-to-face and distant activities. In this way a VIDL can lead to a more profound understanding of e-learning scenarios.
- **Reuse** of already successful e-learning course designs or e-learning content.
- **Sharing** of expertise as well as best practices.
- **Training** of new designers.
- Teaching strategies as well as used materials can be **documented** in a standardized and compatible way.
- Bridging the gap between the **design** and **implementation.**

Les inconvénients suivants ont été relevés dans la littérature :

- les utilisateurs ont besoin d'un certain **niveau d'expertise** du VIDL et de ses outils ;
- ils ont besoin de se sentir à l'aise : le/les VIDL(s) doivent être **utiles** et **utilisables** *(useful and usable)* ;
- tous les avantages sus-nommés pour les VIDLs ne peuvent être abordés par un seul d'entre eux : la conception pédagogique relève généralement de l'utilisation de **plusieurs** langages et non d'un seul langage « universel » ;

2.5 Vers des VIDLs spécifiques aux plateformes de formation à distance

Dans le cadre du projet GraphiT, notre intérêt est orienté vers des VIDLs qui sont dédiés aux plateformes de formation à distance. Nous proposons donc de réduire l'étude des VIDLs existants à ceux ayant une orientation ou *a minima* une relation particulière (une correspondance par exemple) avec les environnements de formation qui nous intéressent.

Afin de ne pas écarter des VIDLs qui pourraient toutefois nous intéresser par l'étude de la notation visuelle qu'ils proposent ou bien d'autres particularités pertinentes, nous proposons de ne pas décrire en détails ces VIDLs dans la section suivante.

3 VIDLs et éditeurs existants

Cette section présente les langages de modélisation pédagogique visuels existants. Pour chacun d'entre eux, nous présentons tout d'abord une description générale du langage, de son approche, sa notation, ses enjeux, etc. selon ce qu'en disent ses auteurs.

Ensuite nous illustrons concrètement le langage par des extraits de diagramme et des précisions sur la notation visuelle. Ces éléments peuvent apparaître sous forme de capture d'écran si un éditeur spécifique au langage a été élaboré.

Enfin, nous détaillons les informations sur méta-modèle, si connu et/ou officiellement présenté par les auteurs du langage. Il est possible que ce méta-modèle ne soit pas formalisé sous une forme autant détaillée. Si des modèles « conceptuels » sont disponibles, même réduits et non exhaustif de la totalité de la syntaxe et sémantique du langage, ces derniers sont alors présentés. Nous précisons pour chaque langage si celui-ci a un ou des éditeurs spécifiques qui ont été développés. Pour finir, nous précisons les relations existantes entre ce langage/éditeur et les plateformes de formation (prise en compte, exports vers formats intermédiaires, etc.).

3.1 Approche narrative

Présentation

Un diagramme de narration permet de décrire, en conception, les objectives « empiriques » à atteindre dans la réalisation d'une activité éducative. Il inclut une orientation verticale visant à augmenter l'engagement des apprenants à travers l'apprentissage progressif **(Parrish, 08)**.

Discussion

Il s'agit d'un VIDL très marginal, pas d'outil-support, pas de méta-modèle mais tout de même une notation visuelle orientée diagramme et « *plots* ».

Exemple

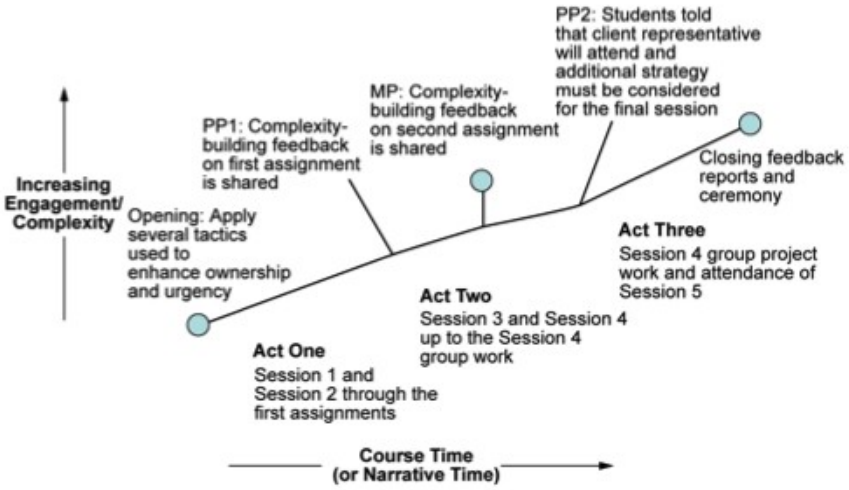


Figure 3: Diagramme de l'approche narrative extraite de **(Botturi et al, 08)**

3.2 CPM

Présentation

Le langage CPM (*Cooperative Pbl Metamodel*) **(Laforcade et al, 05)** est une proposition de langage basé sur UML pour la modélisation semi-formelle graphique (orientée diagrammes) de situations-problèmes coopératives. Ce type d'apprentissage est issu du courant constructiviste selon lequel : 1) les connaissances sont construites, 2) l'apprenant est au centre du processus d'apprentissage et, 3) le contexte d'apprentissage joue un rôle déterminant (Deschênes et al., 1996). Le langage CPM est spécifié comme une extension du langage UML conçu par la technique de méta-modélisation des profils UML. Comme le langage UML, CPM est alors composé de :

- une syntaxe abstraite : représentée par le méta-modèle CPM qui définit les concepts et leurs relations ;

- une syntaxe concrète : représentée par le profil CPM qui définit la notation des concepts et de leurs relations ainsi que leur utilisation dans les diagrammes UML ;

- une sémantique : celle-ci est définie au niveau de la terminologie dans le méta-modèle (sous la forme de contraintes OCL et de règles en langage naturel) comme au niveau de la notation (sous la forme de propositions d'usage des diagrammes).

Informations sur le méta-modèle

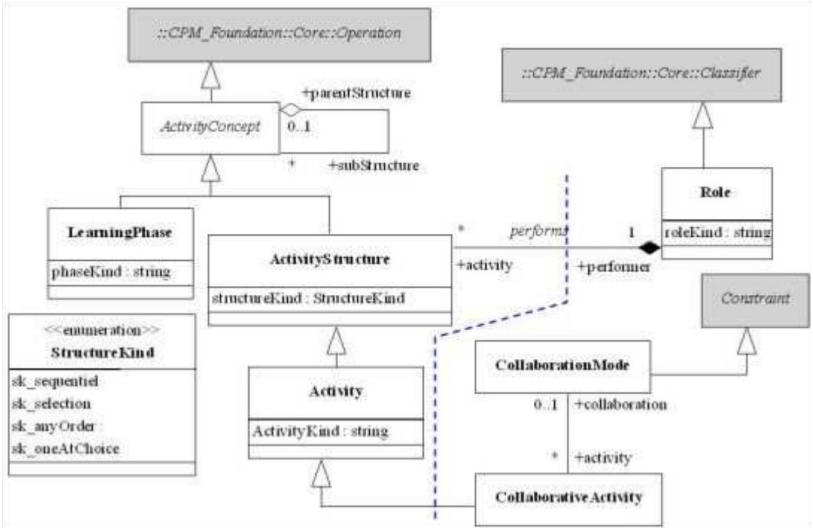


Figure 4: Extraits des paquetages structurels et sociaux du méta-modèle CPM

Le méta-modèle de CPM a été spécifié de manière similaire à celui d'UML : un *noyau* représentant les fondations (qui peut être assimilé au MOF, Meta-Object Facility) et des extensions spécifiant concrètement la véritable partie « métier » du méta-modèle. La figure suivante illustre une partie de 2 paquetages sur les 5 constituant l'extension. Les explications détaillées de chaque élément du méta-modèle sont expliquées dans **(Laforcade, 04)**.

Editeur et illustrations

L'éditeur associé au langage CPM est en réalité une personnalisation de l'atelier de Génie Logiciel UML Objecteering qui permet la création et exploitation de profils UML. Objecteering propose également un langage propriétaire J qui permet de définir et personnalisation des éléments IHM pour guider la spécification dans les diagrammes spécifiques aux profils.

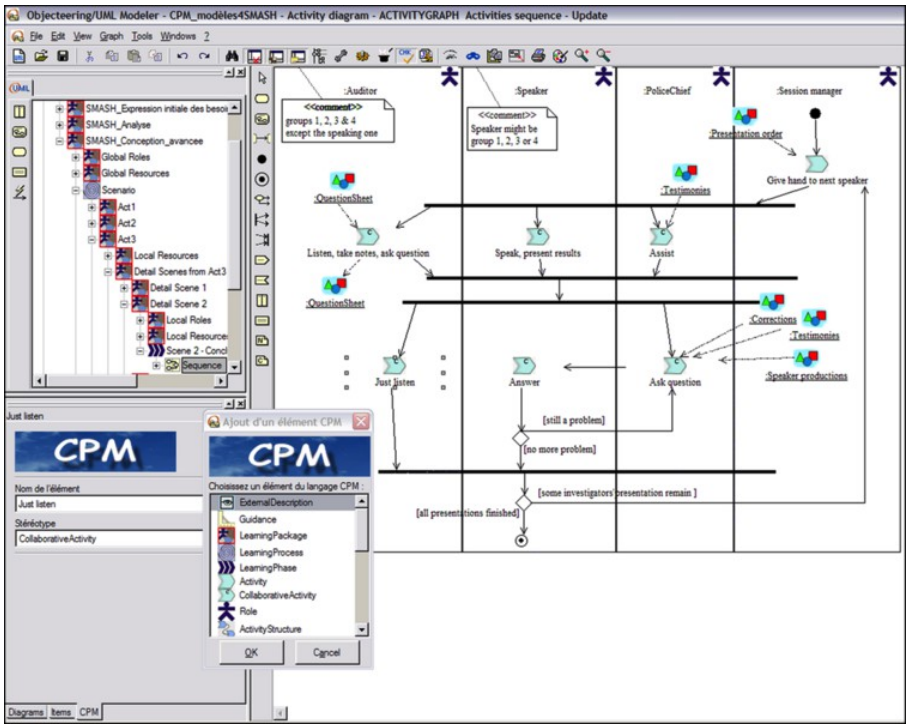


Figure 5: L'éditeur basé Objecteering du langage CPM

Lien avec les autres langages et plateformes

L'éditeur du langage CPM intègre un export vers le standard de facto IMS-LD **(IMS, 03)**. Il est ainsi actuellement possible de transformer des diagrammes d'activités modélisant des scénarios d'apprentissage à l'aide du langage CPM vers des modèles XML conformes au standard de la spécification d'IMS-LD.

CPM n'adresse pas autrement la mise en œuvre des modèles qu'il permet de décrire. Aucune référence aux plateformes de formation à distance, si ce n'est une mise en œuvre manuelle, n'est mentionnée.

3.3 E²ML

Présentation

E²ML (*Educational Environment Modeling Language*) est présentée comme une notation de modélisation semi-formelle pour créer et documenter des "*instructional designs*" **(Botturi, 2004a)**. Il est possible de décrire des objectifs d'apprentissage *(Learning goals*), des pré-requis (*requirements*) et de modéliser des activités d'apprentissage (*teaching and learning activities*). There is a more specific tool for goal classification that was developed together with E2ML: the Quail Model **(Botturi, 2004b)**, which is a visual model for the definition and classification of high-level learning goals. E2ML modeling starts with the definition and mapping of educational goals, then all available resources (actors, resources, tools) get listed and action diagrams as the core part are modeled. Finally overview diagrams are created, for example a timeline as visualization of the "course calendar" **(Botturi, 2003)**. So there are three central document sets: (1) goal definitions, (2) action diagrams and (2) overview diagrams **(Botturi & Belfer, 2003)**.

Informations sur le méta-modèle, les outils associés, et les plateformes

E²ML ne propose pas de formalisation de sa syntaxe abstraite. Il n'existe pas non plus d'éditeur supportant ce langage. E²ML ne vise pas l'implémentation des scénarios conçus sur une plateforme de formation ou tout autre environnement.

Exemples

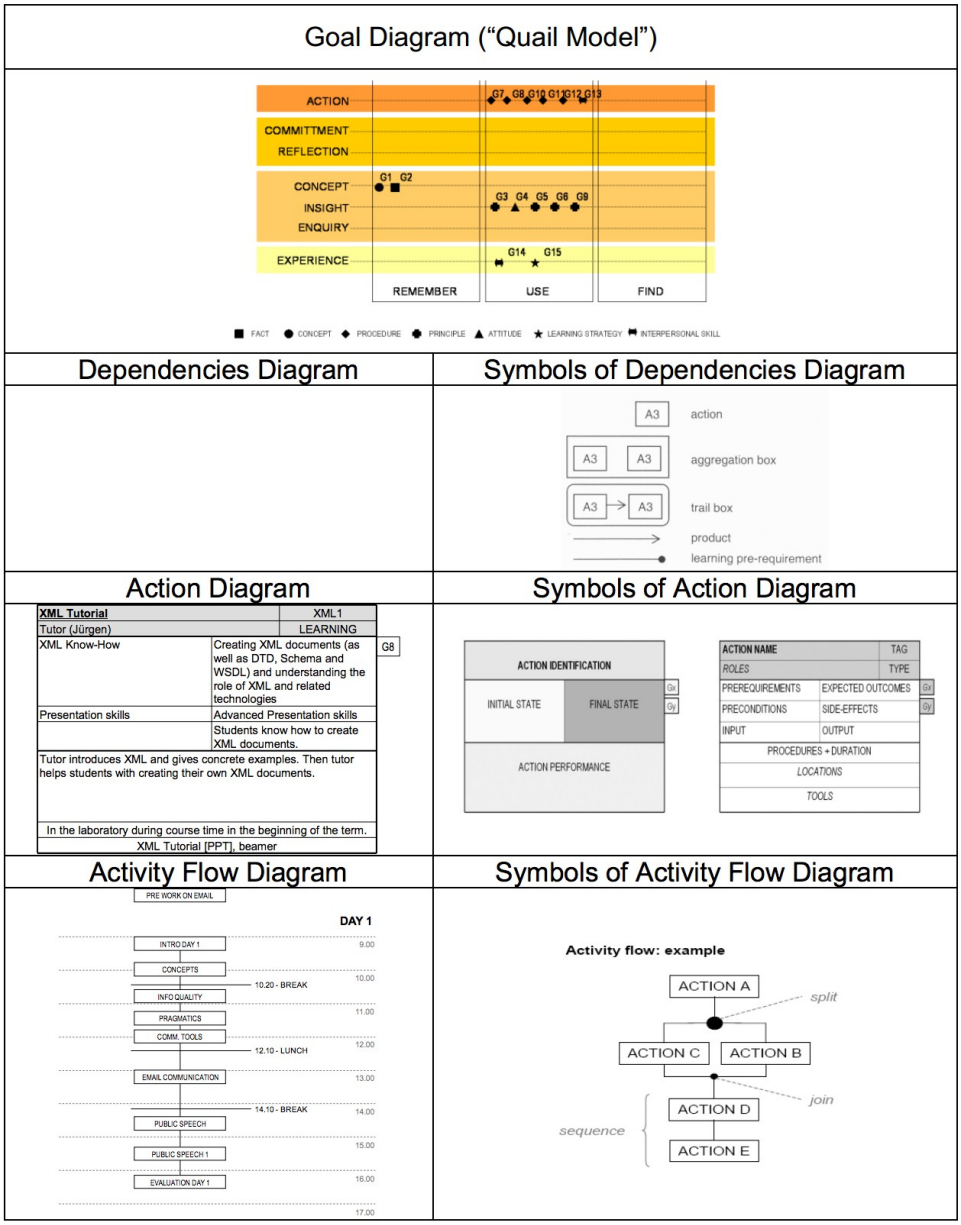


Figure 6: Les différents diagrammes et notations de E2ML

3.4 PceL

Présentation

PCeL combines a *person-centered* approach to teaching and learning with elements of e-learning. PCeL scenarios and instructional activities which facilitate meaningful learning are modeled and collected in the PCeL pattern repository **(Derntl, 2005)**. Families of related patterns are organized in packages. There are seven packages in the pattern repository:

- Assessment,
- Course Types (e.g., patterns for seminars, lectures),
- Evaluation (e.g., self- and peer-evaluation),
- Feedback (e.g., reaction sheets),
- General (e.g., meetings, team workspaces, diary),
- Interactive Elements (e.g., brainstorming, team building, chat, tutorial)
- and Project-based Learning.

The patterns are modeled using extended UML activity diagram. The most notable extensions include different stereotypes for declaring activities as face-to-face, web-based, or blended. The complementary structural models use generalization/specialization concepts, as well as dependency relations (e.g., include, derive, successor-of, or use). All patterns are described by a template style including name, intent, motivation, parameters, and further sections.

Exemples

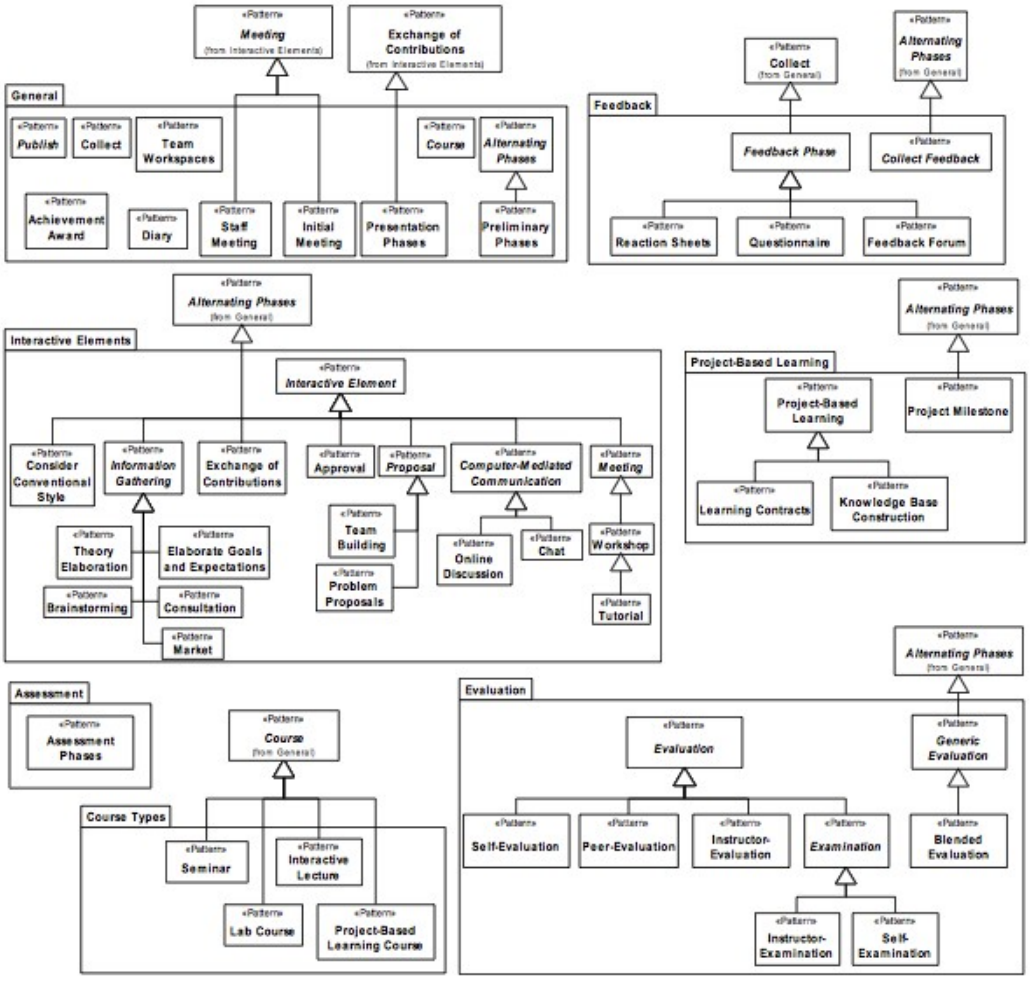


Figure 7: vue d'ensemble des patrons proposés dans PCeL

Informations sur le méta-modèle, les outils associés, et les plateformes

PCeL ne propose pas plus de formalisation que la description des méta-données communes à tous les patrons. Il n'existe pas non plus d'éditeur dédié à ce langage (pour assembler/structurer les patrons pour spécifier un scénario d'apprentissage). PCeL ne vise pas l'implémentation des scénarios conçus sur une plateforme de formation ou tout autre environnement.

3.5 EduWeaver

Présentation

EduWeaver est un framework de modélisation. EduWeaver is based on the meta-modeling platform ADONIS. EduWeaver allows defining courses and teaching processes with coherent learning units in a sequence, as well as lessons and learning objects. An XML-export interface is included, which allows to export content packages according to the IMS standards and integrate them into learning management systems.

There are four modeling levels that are hierarchically linked with each other **(Karagiannis & Bajnai)**:

1. Course: provides an overview of all courses, which will be specified in the other levels.
2. Module: One course consists of different modules, which are thematically coherent units and they can be modeled in several possible teaching paths –sequential, alternative (if previous knowledge is necessary) or parallel (e.g. theory and practice).
3. Lesson: Lessons take about 45–90 minutes of learning time. Lessons can take place sequential (e.g. weekly), in loops, or parallel (e.g. online and present) and alternative paths can be defined.
4. Learning object use: what learning objects are used in a single lesson?

Designers can either start bottom-up with learning objects, collect them in a lesson, and in modules building a whole course – or they can start top-down modeling courses first. Learning objects are small, homogenous parts of the subject and they need about 5–15 minutes to be taught. They are linked to the real multimedia material (HTML, document...) and can be used in several courses. For each of the modeled elements – courses, modules, lessons, and learning objects – the application offers a “notebook,” where further information can be supplied. For learning objects this is for example general data like description or owner and additional data like related learning objects, target group, requirements, key words or HTML documents.

Informations sur le méta-modèle

Les auteurs d'EduWeaver positionnent la plateforme ADONIS, le langage EduWeaver et les modélisation de cours selon l'architecture en couche de l'OMG.

The modeling core of eduWeaver consists of four modeling levels. Each level contains learning construct instances that correspond to the model types CourseOverview, Course, Module and Lesson. These model types are hierarchically linked to each other by internal references. Within each modeling level sequences of instruction can be graphically modeled by using according object and relation classes representing different granularities of the process level.

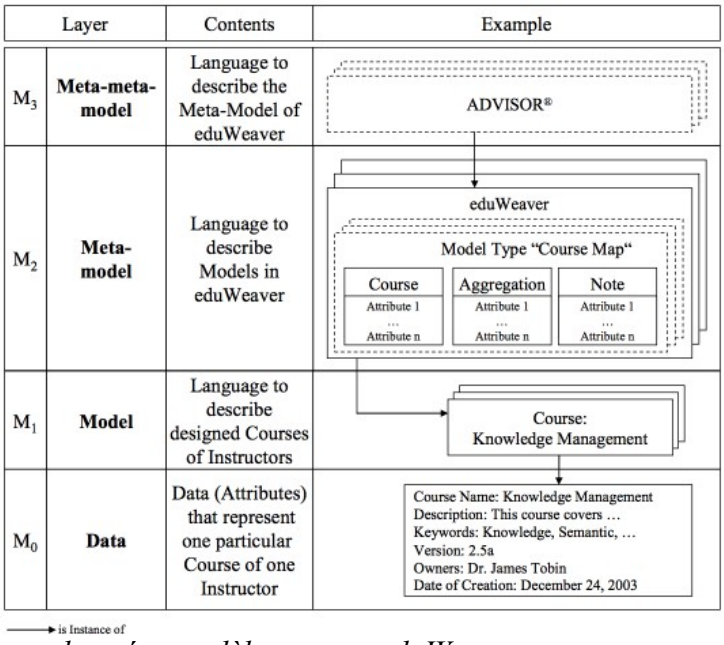


Figure 8: L'approche méta-modèle vue par eduWeaver

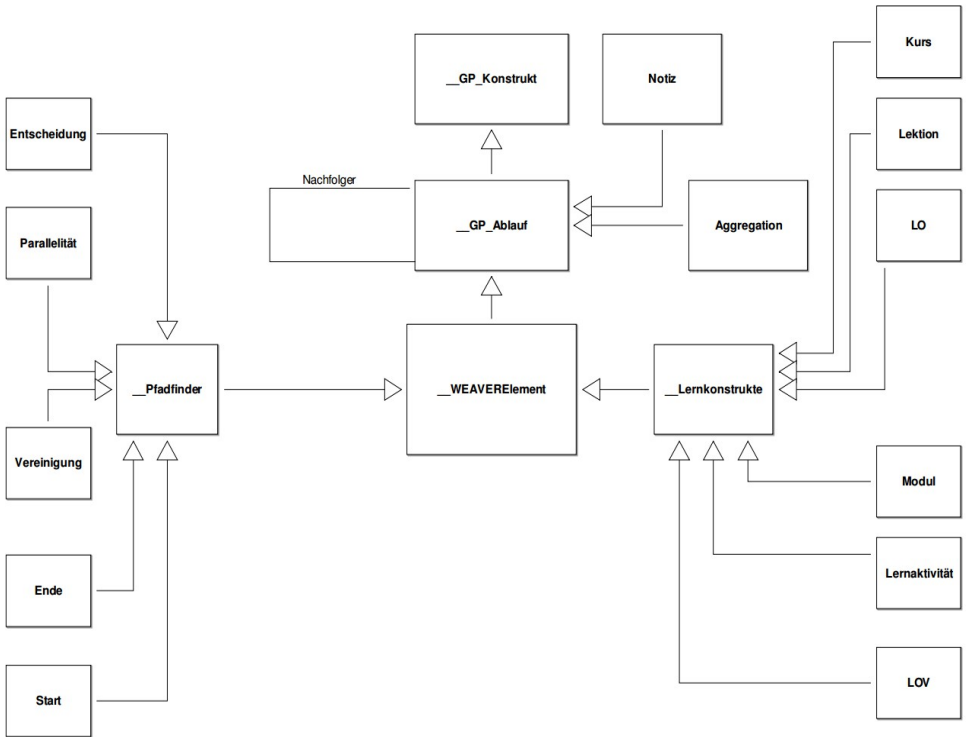


Figure 9: Le métamodèle d'EduWeaver *(Bajnai et al., 2005)*

Exemples

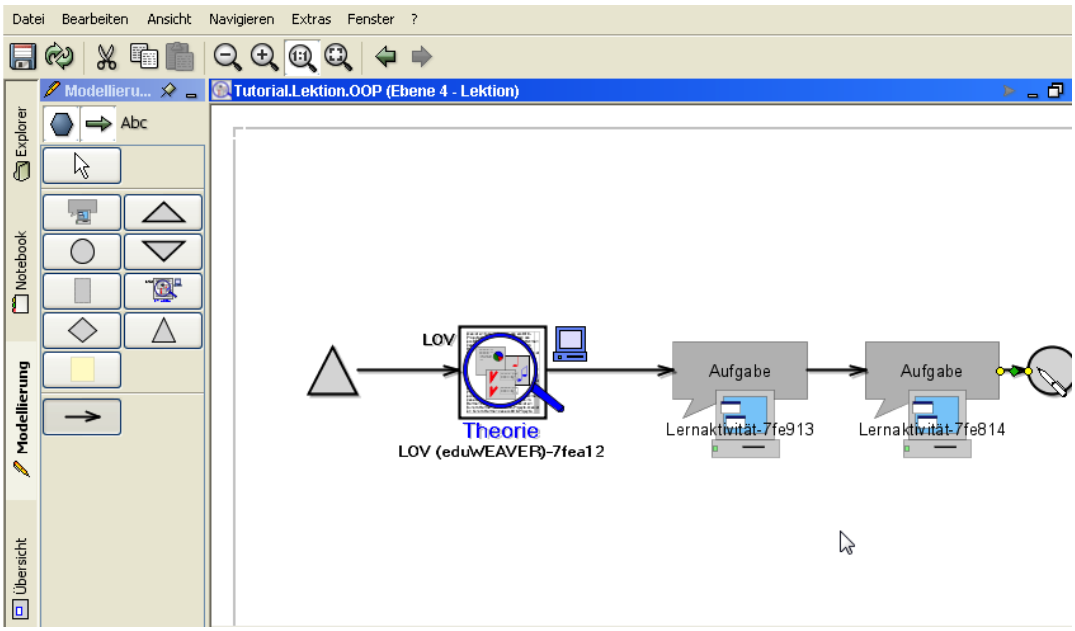


Figure 10: Capture d'écran de l'outil eduWeaver

Informations sur les outils associés, et les plateformes

eduWEAVER est présenté à la fois comme un VIDL et comme un outil. Ce dernier a été développé à l'Université de Vienne et semble n'exister que dans une version en langue allemande.

Les modules réalisés avec cet outil peuvent être exportés en HTML, SCORM 1.2, IMS Content Packaging. Un export IMS-LD était à l'ordre du jour en 2009 mais ne semble pas avoir été réalisé encore à ce jour. Selon ce que les plateformes de formation considérées sont capables d'importer, il peut donc y avoir réutilisation des modèles produits pour pré-configurer un espace-cours correspondant.

3.6 coUML

Présentation

coUML est un langage de modélisation pédagogique basé sur UML. Il vise à être utilisé à n'importe quelle étape d'un processus de conception et de développement d'un cours dans laquelle des modèles visuels et des descriptions structurées de la conception en cours sont pertinentes pour l'équipe de conception. CoUML s'appuie sur le formalisme existant d'UML et lui ajoute des extensions qui lui sont propres. D'après ses auteurs l'élaboration de coUML a émergé après plusieurs années de pratiques d'enseignants-concepteurs. Il a été publié et présenté en 2007 **(Derntl & Renate, 2007)**. Il propose d'aborder les exigences et besoin suivants :

- Support for local and temporal activities, including decisions and concurrency
- Activities can be modelled at different levels of detail
- Roles can be attached to activities
- Most learning activities "consume" or "produce" documents of various kinds
- Learning goals can be modeled and linked to activities
- Activities can be tagged either "web-based", "blended" or "presential".

Les modèles de coUML couvrent la conception d'un cours selon 3 perspectives :

Primaires

- Le ***course activity model*** (CAM) (modèle d'activités du cours) décrit les activités des participants avec un langage dérivé des diagrammes d'activité UML.
- Le ***course structure model*** (CSM) (modèle de la structure du cours) décrit les modules d'un cours et leur dépendances).

Secondaires

- Le ***roles model*** (modèle des rôles) permet de décrire les rôles des participants et leurs relations
- Le ***goals model*** permet de décrire les objectifs pédagogiques et leurs dépendances
- Le ***documents model*** définit les documents produits et utilisés. Il est également possible d'attacher des rôles.

Auxiliaires

- Le ***course package model*** (CPM) est un simple tableau qui résume la description du cours avec une dizaine d'éléments.

Exemples

La figure suivante illustre un exemple concret pour chacun des différents modèles.

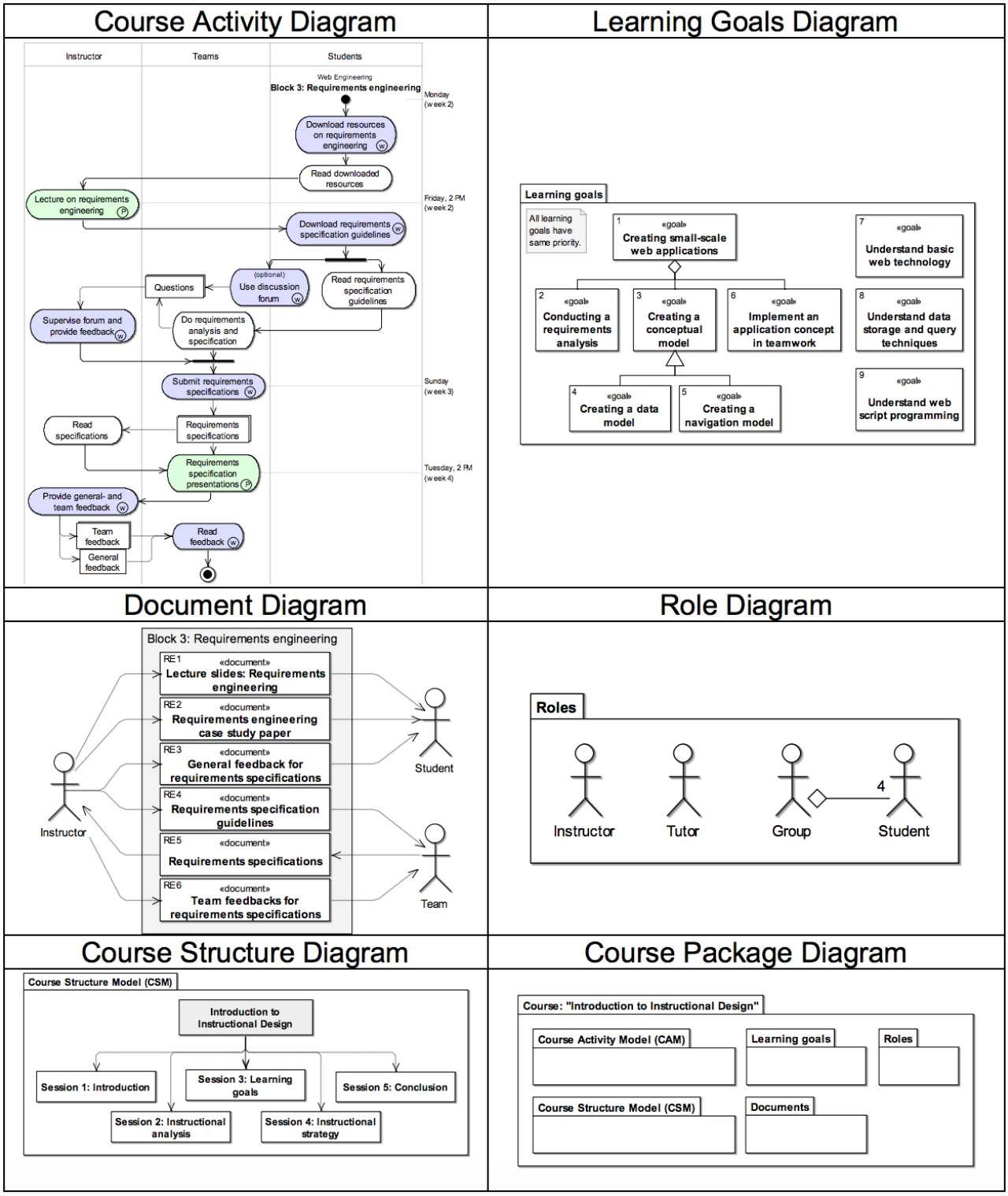


Figure 11: Exemples des différents diagrammes et notation proposés par coUML

Informations sur le méta-modèle, les outils et la relation avec les plateformes

coUML est construit comme une extension d'UML : il s'appuie donc sur ce que UML permet et propose de restreindre et détourner certains diagrammes UML pour en faire les usages définis en relation avec les différents « modèles » coUML. coUML propose donc en plus d'UML des stéréotypes qui sont concrètement des annotations que le modeleur doit préciser sur certaines entités lorsqu'il modélise. Ces stéréotypes agissent comme des étiquettes en apportant par leur simple présence une sémantique propre au VIDL coUML.

coUML ne propose pas d'outil dédié. Les exemples que l'on peut trouver sont surement basés sur un outil modeleur UML qui permet l'ajout de stéréotypes dans les diagrammes. En l'état, il n'existe donc pas d'outil permettant d'assurer à l'enseignant-concepteur qu'il réalise bien un modèle conforme à la sémantique de coUML. En comparaison avec le VIDL CPM qui s'appuyait également sur UML, coUML n'étend pas UML avec des *tagged-values* ni des icônes spécifiques. De plus CPM avait été accompagné du développement de nombreuses interfaces additionnelles à *Objecteering* afin de guider la modélisation.

En ce qui concerne les plateformes de formation, coUML ne mentionne aucune relation directe dans la sémantique du langage étant donné que coUML revendique être « pédagogiquement neutre ». Les auteurs évoquent la possibilité d'exploiter les modèles UML annotés résultants comme base informatique exploitable (fichier XML) mais ne mentionne aucune exploitation concrète.

3.7 POEML

Présentation

poEML (perspective oriented EML) is a visual educational modeling language (EML). The main idea underlying poEML is to break down the modeling of educational units into separate parts that can be specified independantly. “PoEML proposes two different kinds of orthogonal concerns: (i) vertical concerns that group related elements in separated parts (named as Perspectives); and (ii) horizontal concerns that consider different modes of characterization of those elements (named as Aspects). Based on this distinction PoEML identifies 12 perspectives and 4 aspects. In conjunction, they enable to approach the modeling of educational units in an incremental and iterative way offering advantages in expressiveness, simplicity, reusability, flexibility and adaptability.” **(Caeiro, 2008)**.

Caeiro identified 13 different perspectives for designing or looking at educational units:

- Structural. It is concerned with the arrangement of the elements involved into well-structured and self-contained components.
- Functional Goals that have to be attained in an educational unit, i.e. the work that has to be performed by participants
- Participants involved in the educational unit, defined in terms of roles.
- Environments, i.e. the combination of data elements, tools and services that can be used by participants to work and to achieve the goals.
- Data used in educational units. Data are used by other elements and there is also a data flow issue.
- Tools, i.e. applications and services that can be used in an educational unit (e.g.

simulators, editors, communication and collaboration services)

- Organizational, structure required to carry out an educational unit.
- Order (or control flow), the order in which activities are intended to be performed.
- Temporal, time specifications that tell when an activity must be initiated or finished and/or for how long it should last.
- Authorization, i.e. access rights of participants to environments' elements, mainly to the tools' functionalities.
- Awareness is concerned with the processing of runtime information (events) and the notification of relevant situations, e.g. that a teacher knows what is going on.
- Interaction concerns the invocation of operations in tools, i.e. it deals with the mechanisms required to support the invocation of operations.
- Causal. It involves competencies, metadata, learning objectives, pre-requisites, etc. It informs participants about why they should perform an educational unit.

poEML models these elements and their relationship with UML models. These models are then used to design the JPoEML graphical scenario editor.

Informations sur le méta-modèle

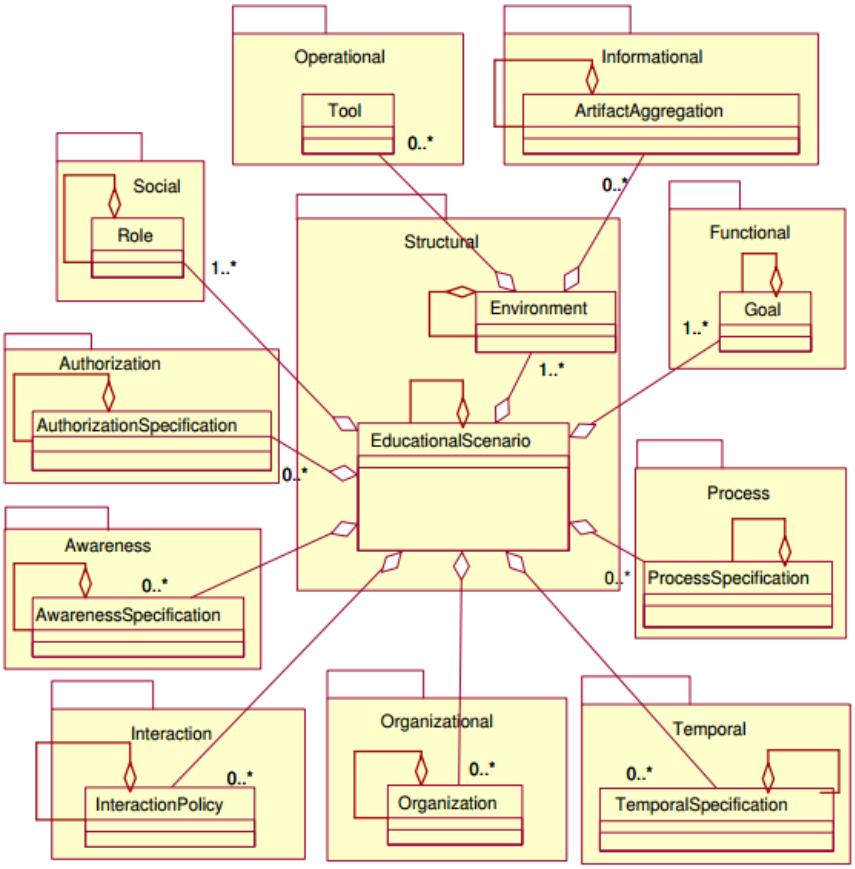


Figure 12: Métamodèle de POEML

This figure illustrates the EducationalScenario (ES) element and its main relationships. An ES is intended to support the modeling of any kind of educational practice at different aggregation levels, from simple lessons, to complete courses or curriculums. The ES element is the aggregation point where all other elements are anchored. Each ES constitutes a context of elements not accessible from other ES, except from a Parent ES.

As it is represented in the figure, an ES is intended to: (i) achieve a certain Goal or set of Goals; (ii) that have to be attained by a particular Participant in a Role; in a particular Environment composed by (iii) a set of Artifacts, (iv) and Tools that represent Applications and Services; (v) in the context of a certain Organizational Structure; (vi) considering a certain Order in the way in which Tasks are intended to be attempted; and (vii) Temporal Restrictions on their performance; and involving a set of rules that control and manage (viii) the Authorizations of the involved participants to invoke operations; (ix) the Awareness they receive during execution; and (x) the Interaction through applications and services. The different elements and specifications can be connected directly or considering a control (condition, decision or event) to select them appropriately.

Exemples

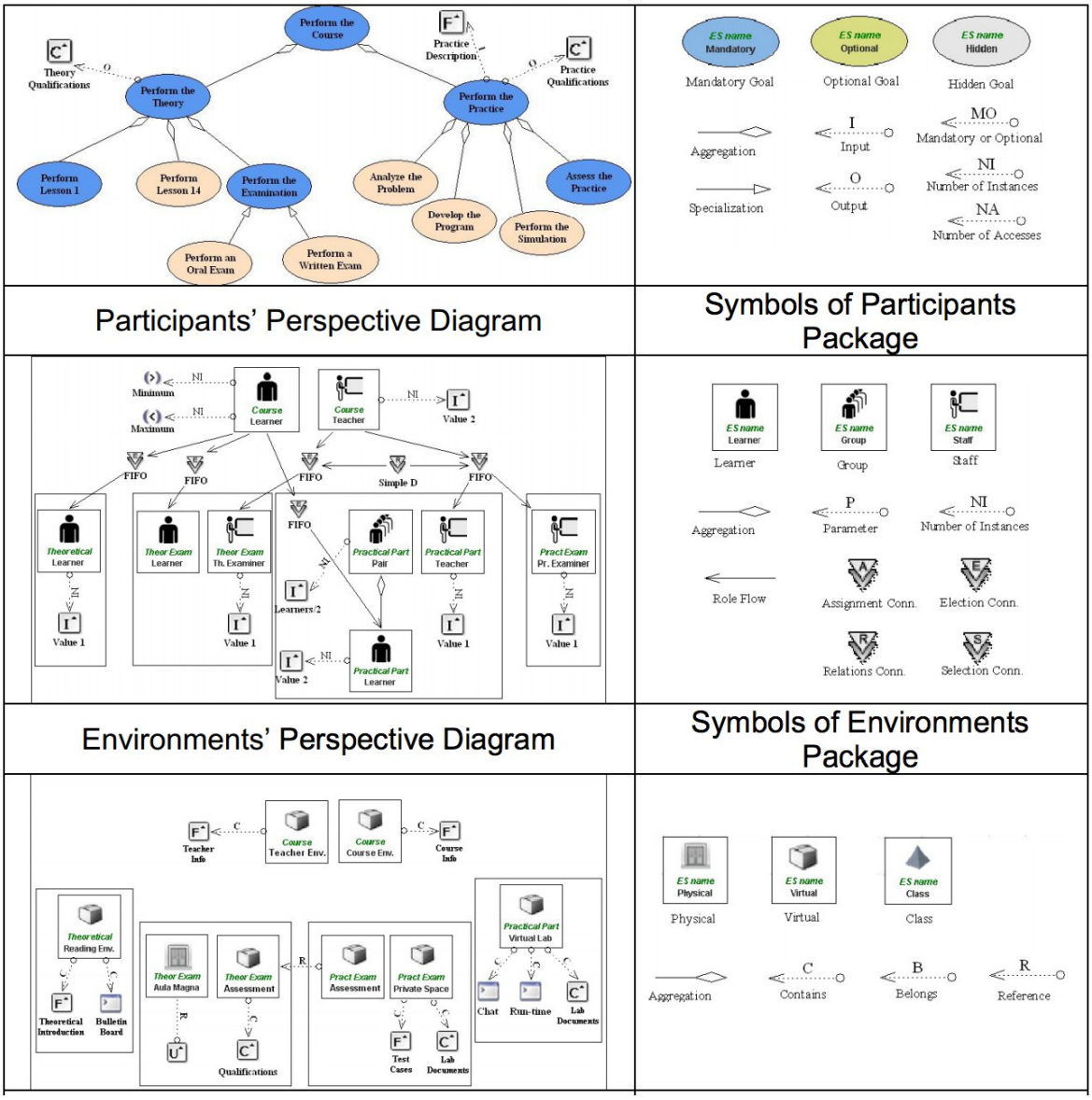


Figure 13 : exemples de modèles réalisés avec poEML

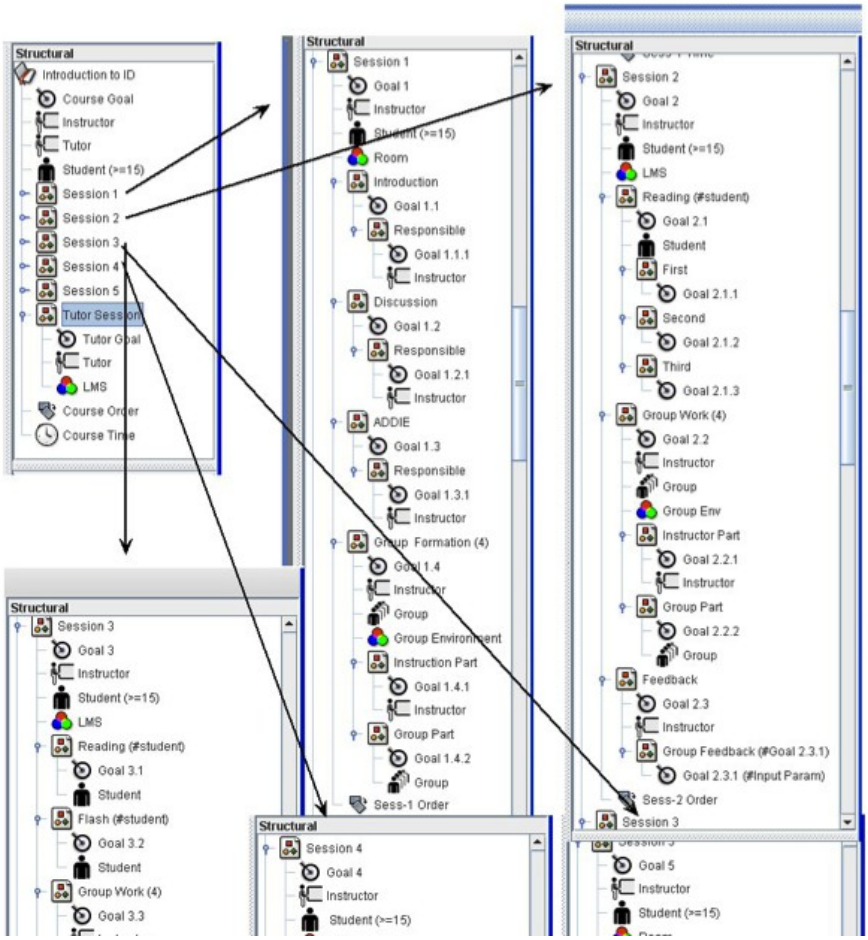


Figure 14: modèle structurel de poEML

Relations avec les plateformes

Les auteurs de poEML ont développé un moteur d'exécution spécifique aux cours poEML. L'exécution est réalisée ainsi par une sorte de plateforme de formation dédiée, élaborée selon une architecture SOA à base de services Web **(Perez-Rodriguez et al., 2010)**. L'éditeur jpoEML propose d'exporter les modèles conçus dans un format XML (manifest) qui sera ensuite importé et exploité par la plateforme.

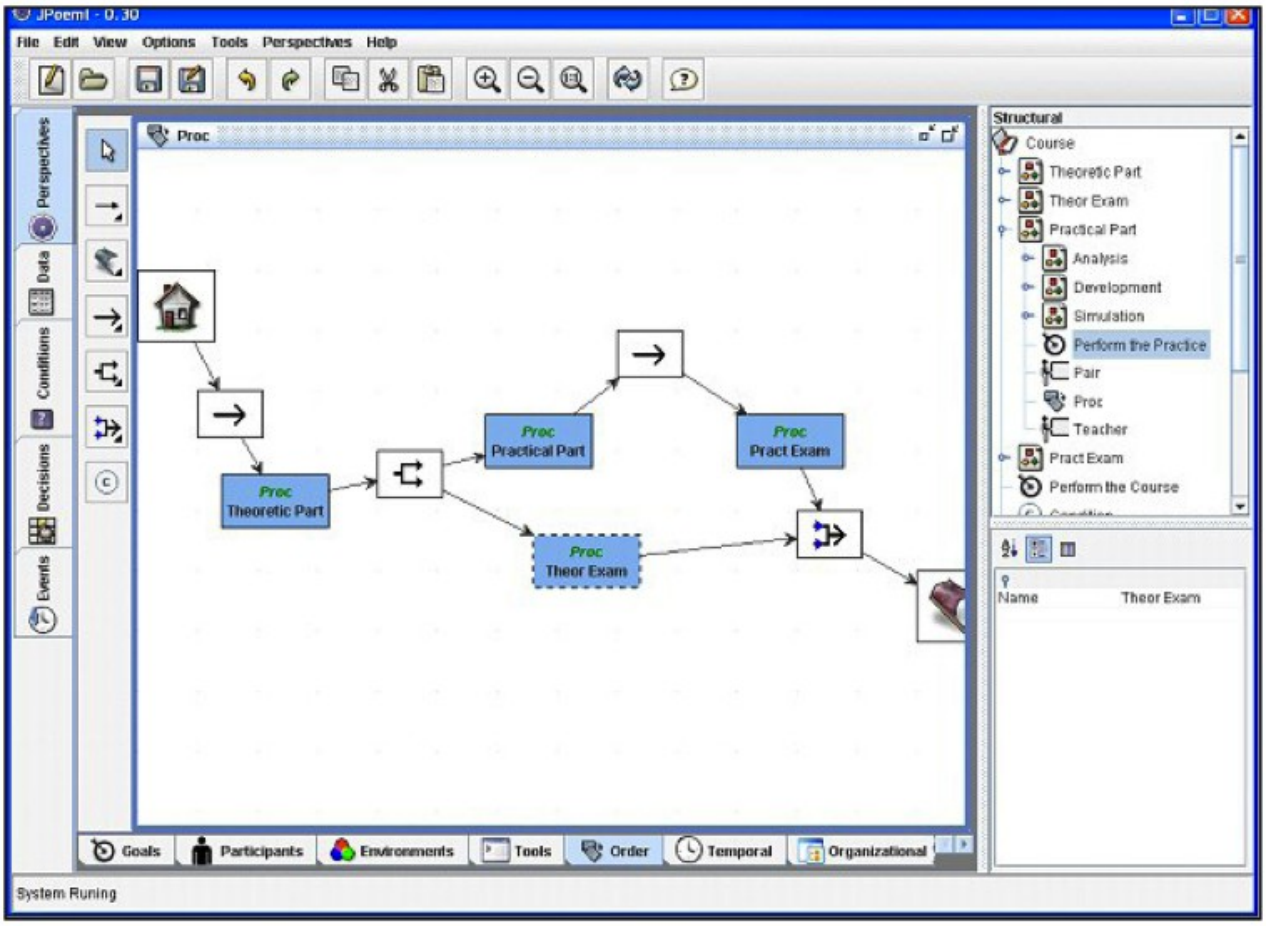


Figure 15: Vue globale de l'éditeur graphique JpoEML

3.8 LAMS

Présentation

LAMS (*Learning Activity Management System*) est un système de conception pédagogique composé d'un système-auteur et d'un système d'exécution et de suivi d'activités pédagogiques. LAMS est donc une sorte de plateforme de formation incluant son propre moteur de workflow pour gérer les flux d'activités pédagogiques. LAMS propose :

- a visual authoring interface to design and create learning sequences from a list of building blocks of individual or collective activities
- a monitoring tool through which teachers can track students' progress through an activity sequence.

According to a LAMS website, LAMS includes environments for user administration, student run-time delivery of sequences, teacher run-time monitoring of student sequences and, most importantly, teacher authoring/adaptation of sequences. LAMS is inspired by, and heavily based on, IMS Learning Design and EML.

LAMS a été intégré à MOODLE comme nouveau moteur de *workflow* compatible avec les modèles spécifiés via le nouvel éditeur interne. Pour l'enseignant il peut soit choisir de bâtir son cours sur le format LAMS et non le format traditionnel de MOODLE, ou créer une activité LAMS dans son cours.

Informations sur le méta-modèle

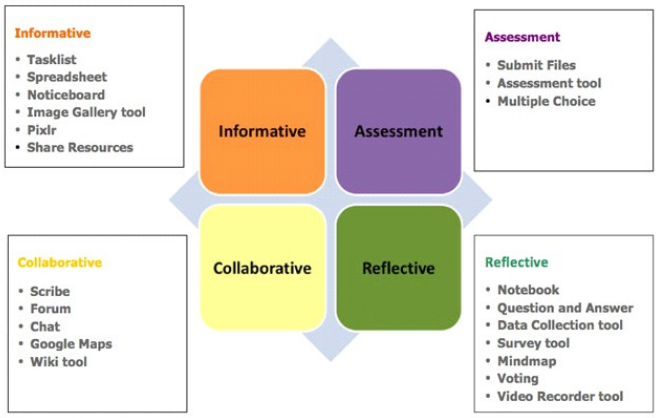


Figure 16: LAMS activities

Il n'y a pas de méta-modèle associé à LAMS présentant les concepts et relations sous-jacents. Toutefois, il est précisé que les différentes activités LAMS sont regroupées selon 4 catégories : *Informative*, *Collaborative*, *Reflective* and *Assessment.*

Exemples

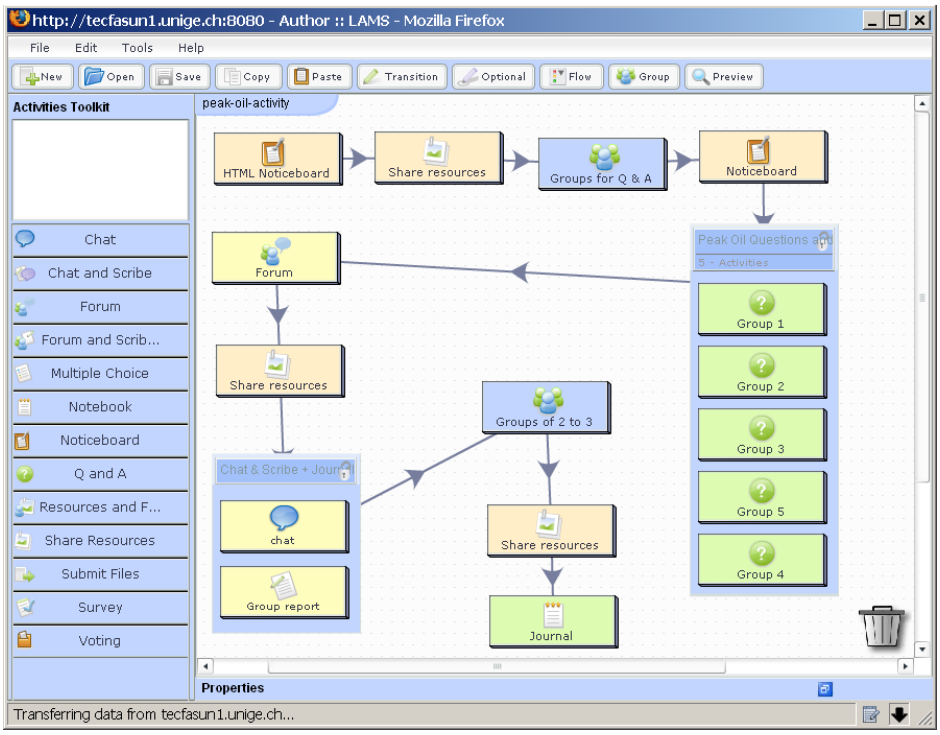


Figure 17: L'environnement online LAMS 2

3.9 LDL

Présentation

LDL (Learning Design Language) **(Martel et al., 2006)** est un langage de modélisation pédagogique (ou langage de scénarisation) qui permet de modéliser des activités collaboratives.

Plus globalement, LDL s'inscrit dans un projet de développer une chaîne éditoriale pour des activités pédagogiques en ligne, c.a.d. il s'agit de l'instrumentation des Espaces Numériques de Travail (ENT).

Contrary to IMS-LD, LDL has not been designed with the aim of supporting the industrialization of pedagogical activities. It has been defined to allow teachers themselves to design and describe these activities. Indeed, one of its general objectives is to make teachers' everyday work easier while using on-line environments and specialized software to operationalize and to execute learning scenarios. Its approach is thus teacher-centered.

Exemples

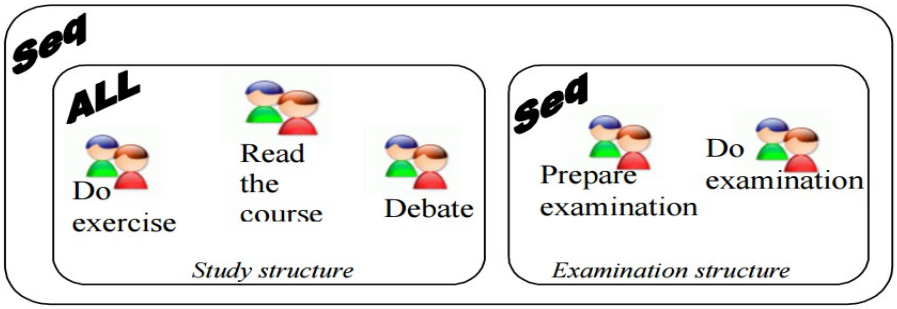


Figure 18: Exemple de modélisation de scénario avec les concepts LDL – vue structurelle

Informations sur le méta-modèle

With LDL, the scenario design starts with: 1) the identification of its interactional structure, i.e. the way the exchanges are going to be organized, 2) the definition of the roles involved, 3) the definition of the arenas which are the places where the activities will take place, 4) the definition of the rules that the participants will have to follow, 5) the definition of the participants' positions i.e. the various points of view which they would have to express during the activity.

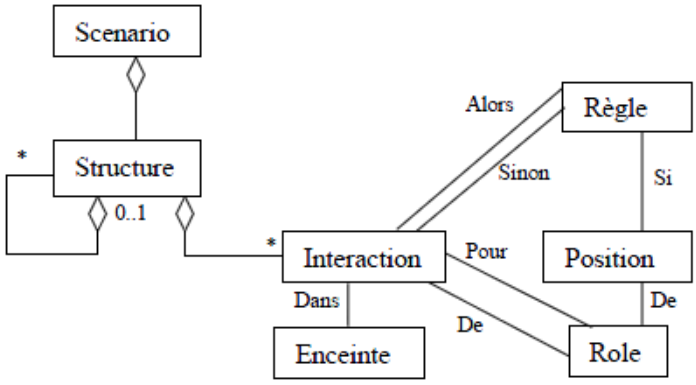


Figure 19: Représentation UML simplifié du méta-modèle de LDL

Informations sur les outils et le lien avec les plateformes

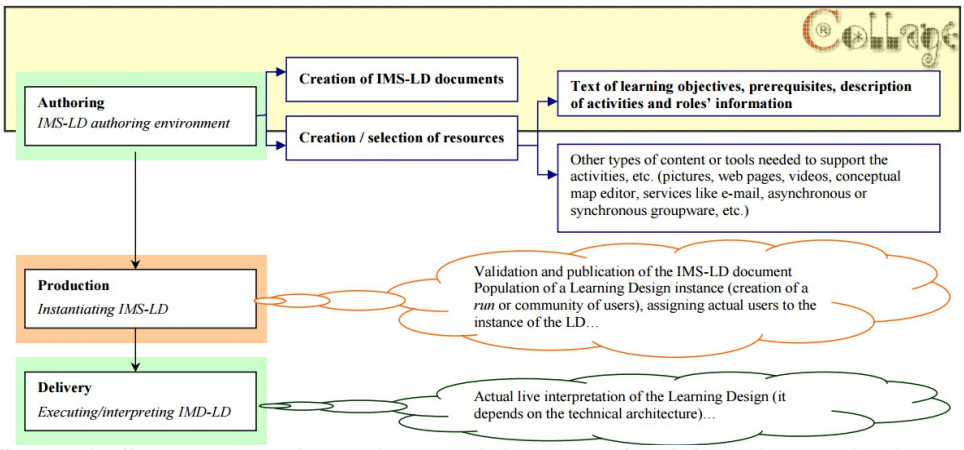
Bien qu'un éditeur dédié à LDL était à l'étude en 2007, aucun outil concret n'a été élaboré depuis. Toutefois, le méta-modeleur ModX a été utilisé pour expérimenter l'outillage de LDL. LDL devait également à l'origine être en relation avec l'infrastructure LDI, des mêmes auteurs et de la société *Pentila Corporation.* Aucune relation entre LDL et les plateformes de formation existantes de type Moodle n'est actuellement connu.

3.10 COLLAGE

Présentation

Collage is a Collaborative learning flow pattern editor. It allows a designer to configure a design pattern. The result is saved in IMS Learning Design format. It could be played with any LD player. “The Collage authoring tool guides and supports the course teacher in the process of authoring computer-interpretable representations (using the IMS learning design standard notation) of effective collaborative pedagogical designs.” **(Hernández-Leo, 2007)**.

Informations sur le méta-modèle



The Figure bellow summarizes the modules, emphasizing the authoring problem and illustrating which functions are covered by Collage. Namely, Collage is not devoted to the enactment problem: instantiating LDs, binding participants to roles, interpreting an LD, etc. However, Collage allows the authoring of LDs. A Unit of Learning (UoL) is a content package including an LD and a set of physical resources (content and tools) or their location. The resources that contain learning objectives, prerequisites, descriptions of activities and information about roles can be edited as text files in Collage. Other resources should be created with external editors.

Exemples

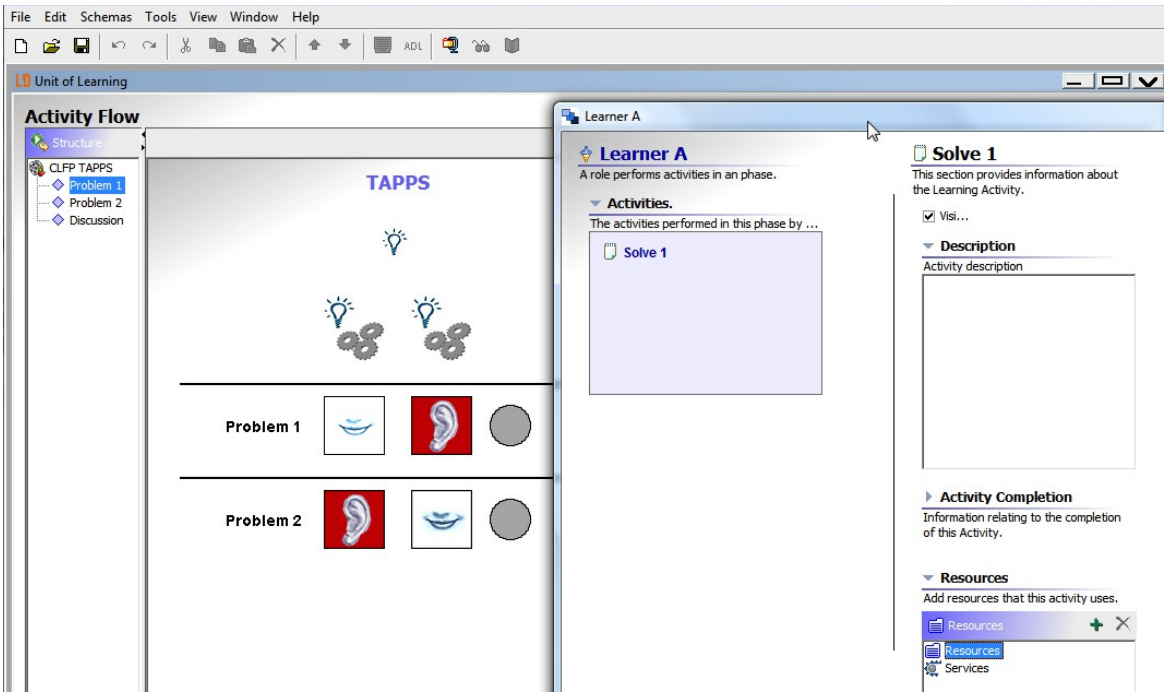


Figure 20: L'environnement COLLAGE

Outil et relation avec les plateformes

Collage est un éditeur à base de patrons de type « Collaborative learning flow ». Il permet aux concepteurs de sélectionner et configurer des patrons de conceptions. Le résultat est sauvé dans le format IMS-LD et peut donc être exécuté sur un moteur/*player* compatible. La version beta de l'éditeur en 2008 était concrètement développée comme une surcouche et une extension de l'éditeur *Reload Learning Design Editor*. Depuis 2012, une version prototype Web est disponible.

Par son lien direct avec IMS-LD, COLLAGE peut donc produire des scénarios qui seront exécutés sur les plateformes de formation compatibles IMS-LD. Malheureusement les plateformes déjà existantes avant COLLAGE (comme MOODLE) ne proposent toujours pas de support pour IMS-LD.

Il est à noter que les auteurs de COLLAGE précisent dans **(Hernandéz et al., 2007)** quelques limitations de leur éditeur. L'ajout de nouveau templates est considéré « laborieuse » : chaque template a sa propre représentation visuelle et inclue des fonctions spécifiques au patron qui lui sont propres. Les auteurs suggèrent qu'il serait intéressant d'étudier la possibilité d'utiliser un VIDL orienté diagrammes pour graphiquement représenter les solutions des patrons de conception.

3.11 MOT+

Présentation

MOT+ **(Paquette et al., 2008)** general graphical language and editor that help construct

structured interrelated visual models.

MOT permet de construire une représentation des connaissances dans divers champs de savoir et de mettre en évidence graphiquement les relations qui existent entre elles. MOT utilise un certain nombre d'icônes graphiques représentant différents types de connaissances : concepts, procédures, principes et faits, ainsi que divers types de liens : composition, régulation, spécialisation, précédence, intrant/produit, instanciation. De plus, il intègre des règles de grammaire qui régissent les types de liens permis selon les types de connaissances. MOT permet également de représenter les habiletés et de les relier aux connaissances auxquelles elles s'appliquent, définissant ainsi les compétences à atteindre.

Informations sur le méta-modèle

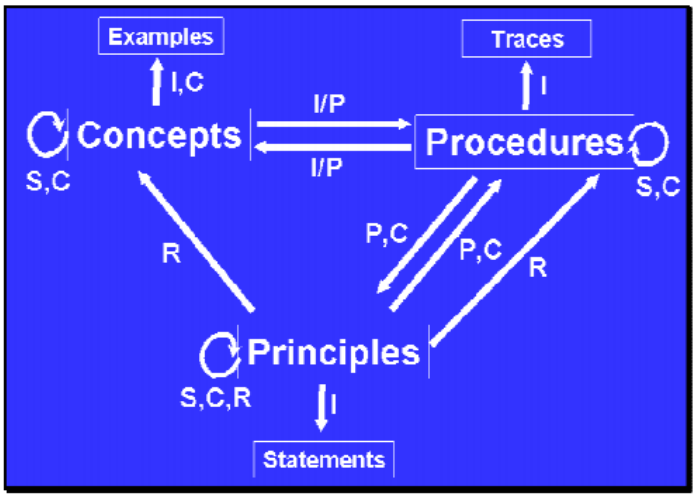


Figure 21: The MOT metamodel

There are various possible semantic interpretations of these graphic symbols.

- Concepts can be object classes (country, clothing, vehicles,…), types of documents (forms, booklets, images,...), tool categories: (text editors, televisions,…), groups of people (doctors, Europeans,…), or event classes (floods, conferences,…).
- Procedures can be generic operations (add numbers, assemble an engine,…), tasks categories (complete a report, supervise a production,…), activities (take an exam, teach a course,…), instructions (follow a recipe, assemble a device…), or scenarios (of a film,of a meeting, of a learning module).
- Principles can state properties of objects (cars have four wheels), constraints on procedures (the tasks must be completed within 20 days), cause/effect relationships (if it rains more than 25 days, the crop will be in jeopardy), laws (any metal sufficiently heated will stretch out), theories (the laws of the market economy); rules of decision (advising on an investment), prescriptions (medicinal treatment, instructional design principles), etc.

Exemples

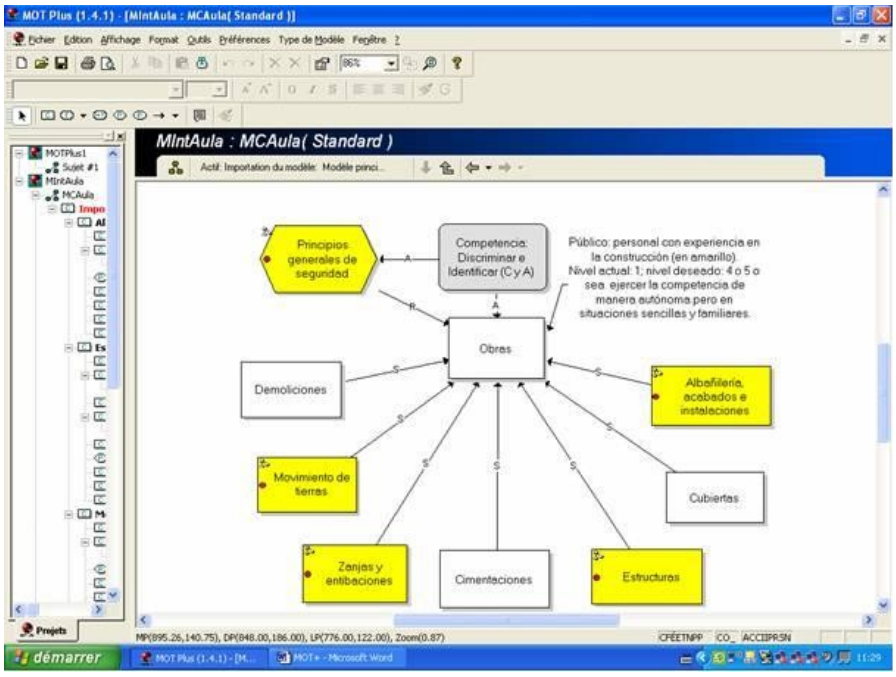


Figure 22: Illustration de l'éditeur MOT+

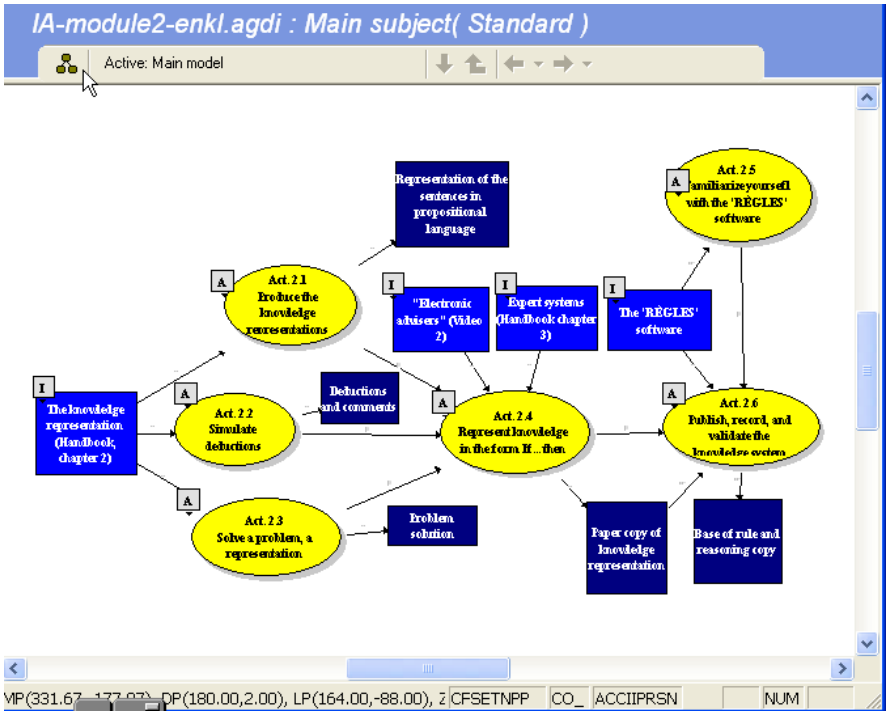


Figure 23: Autre illustration de l'éditeur dédié à MOT+

Outils et relations avec les plateformes

MOT+ est un éditeur visuel graphique. Il existe une version spécifique dédié à IMS-LD : MOT+LD. Dans cette version des primitives concepts et relations pré-configurés pour rappeler la sémantique d'IMDS-LD sont proposés (niveau A seulement). Un exportateur « XML_LD » est utilisé pour la génération du scénario au format IMS-LD. Le lien avec les plateformes de formation auraient donc pu se situer une fois de plus au niveau du standard *de facto* IMS-LD si ce dernier avait été pris en compte par les plateformes existantes.

3.12 CADMOS

Présentation

CADMOS **(Katsamani et al., 2012)** is a user friendly IMS LD level A & B compliant graphical learning design tool addressed to novice learning designers, i.e. practitioners/teachers with basic computer skills and knowledge of learning standards. CADMOS enables a practitioner to design a learning script in layers : first by specifying the learners and teachers activities and the associated learning resources and services required thus creating a learning activity conceptual model, and then by orchestrating the activities per human actor and adding rules and constraints using the metaphor of swim lanes which are all depicted in the learning activity flow model. Thus, not only can a practitioner determine in which order the students should perform the activities but also to specify conditions, preconditions or rules that will be associated to these activities, i.e. A student must study the theory before doing a self-assessment or the student must score at least 70% before proceeding to another activity.

Informations sur le méta-modèle

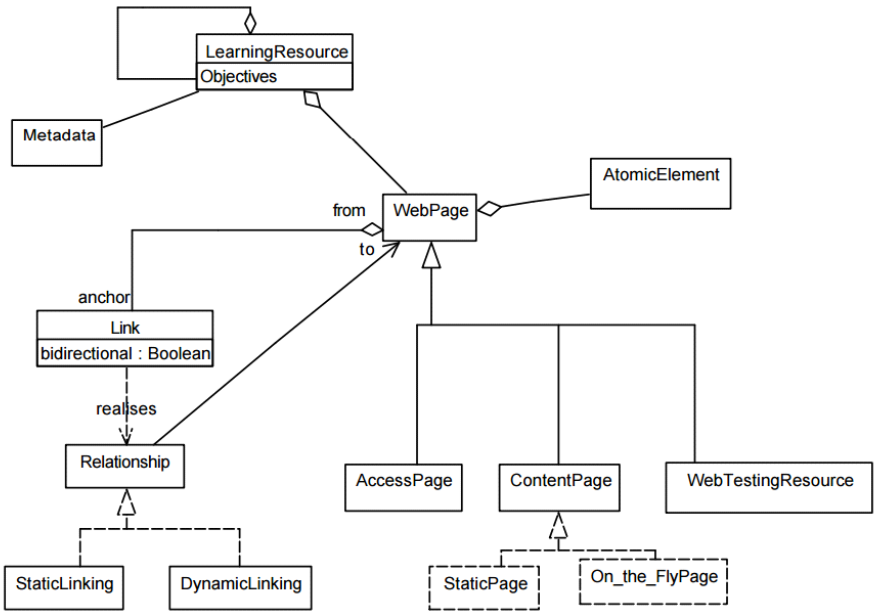


Figure 24: Learning Resources detailed meta-model.

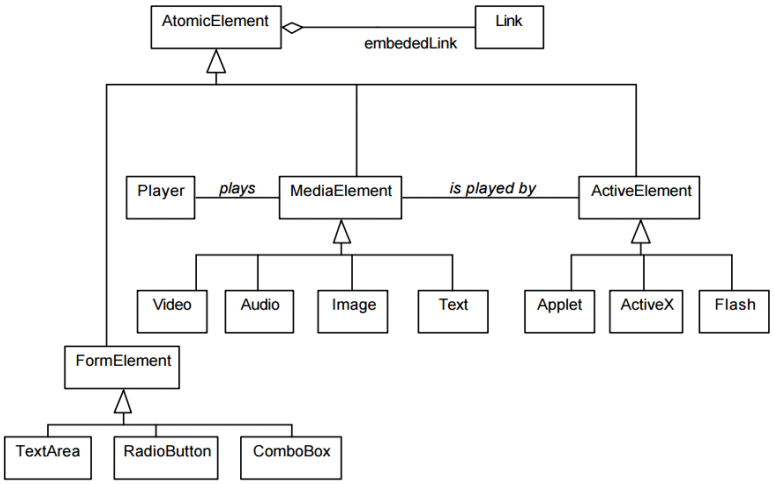


Figure 25: Structure of the web pages' atomic elements

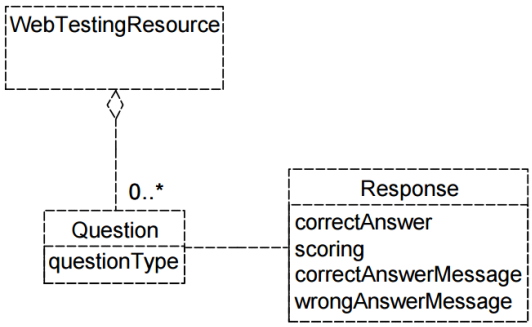


Figure 26: Structure of web-testing resources

Outils et relations avec les plateformes

CADMOS est un outil qui s'adresse à des enseignants-concepteurs novices en termes de compétences et connaissances informatiques. Le processus global de conception supporté par l'outil est considéré comme incrémental. Comme CADMOS s'appuie sur son propre méta-modèle, les scénarios conçus sont sauvés dans un format propriétaire (.cdm). Un export est possible vers IMS/LD (niveaux A et B) : un fichier *manifest* peut être généré et exécutés par des éditeurs ou players compatibles IMS-LD. L'outil offre également la possibilité d'importer un fichier *manifest* IMS-LD existant, et de le traduire dans son propre format. Ainsi, un enseignant peut ré-utiliser un scénario IMS-LD existant.

CADMOS propose également un export compatible avec la plateforme MOODLE. Un script dédié gère la traduction des concepts de CADMOS vers MOODLE et génère concrètement fichier *backup* (.mbz) prêt à être déployé sur la plateforme. Pour importer le scénario sous forme de cours il faut donc que l'enseignant utilise la fonctionnalité de Moodle de *restauration* de cours. Cette dernière n'est pas forcément accessible aux enseignants : cela dépend fortement de la politique d'administration de la plateforme. D'après les auteurs de CADMOS le cours déployé peut être ajusté directement dans MOODLE ou dans CADMOS (bien que cela semble exiger un nouvel export/restore).

Les figures suivantes présentent les correspondances CADMOS=>MOODLE utilisées.

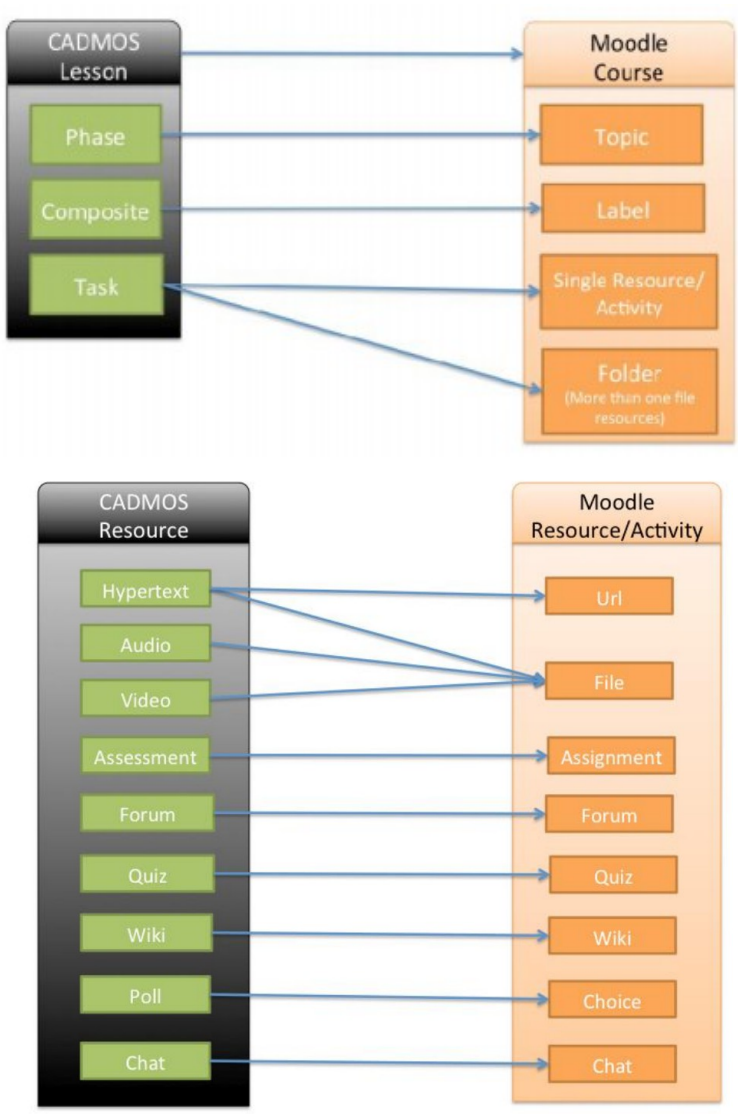


Figure 27: Correspondances entre les concepts CADMOS et ceux de MOODLE

Exemples

Figure 28: Conceptual model

Figure 29: Flow model

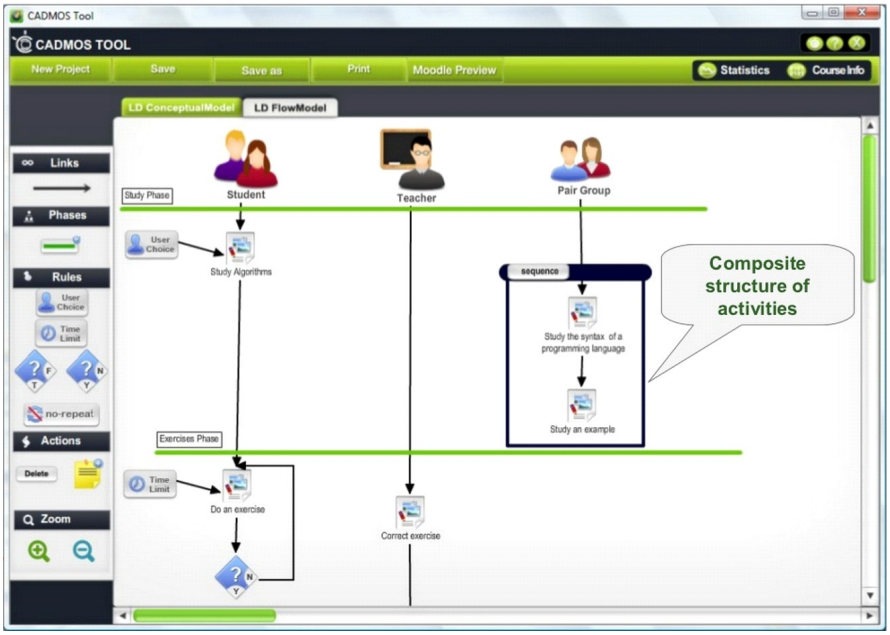


Figure 30: Flow model with rules

3.13 GLUE!-PS

Présentation

GLUE!-PS **(Alario Hoyos, 2012)** is a middleware architecture that allows a lightweight interoperability among different learning design languages and different Virtual Learning Environments. This interoperability allows teachers and other practitioners to particularize (i.e. to customize the "abstract" learning design to the particular classroom, participants and choice of tools where it is going to be enacted), deploy (i.e. to semi-automatically create the needed VLE activities, resources, etc. for the enactment of the particularized design), and even to perform certain changes in run-time (e.g. change group composition, add or remove resources from activities, etc.). All this starting from learning designs expressed in a variety of computer-interpretable learning design languages such as IMS-LD specification.

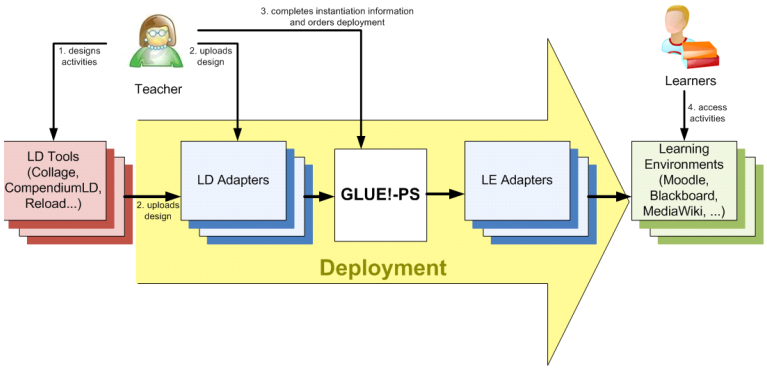


Figure 31: GLUE!-PS architecture

Moreover, GLUE!-PS also allows to deploy and manage learning designs into VLEs that integrate external learning tools (such as a Moodle integrating Google Docs and many others). This is achieved thanks to the GLUE! architecture for the integration of VLEs and external tools.

GLUE! Is another alternative route that allow the deployment of CSCL situations in VLEs, with different design goals. While GLUE!-PS aims at facilitating the management of structures of groups, activities, resources and tools to be deployed in VLE courses or lessons, GLUE! aims at increasing the set of tools teachers may select when deploying these situations.

The GLUE!-PS prototype includes two LD adapters (IMS-LD and WebCollage & and the Pedagogical Pattern Collector) and two GLUE!-PS VLE adapters (Moodle and MediaWiki).

The deployment for the specific Moodle VLE is done by generating a Moodle course backup with all the information, mapping the GLUE!-PS data model concepts to Moodle data model concepts (e.g., GLUE!-PS activities are mapped to Moodle topics); this backup is imported and deployed within a Moodle course using the Moodle restauration process for a course.

Informations sur le méta-modèle de Glue!PS

Glue!PS propose un modèle de données dont l'objectif n'est pas d'être un nouveau VIDL mais plutôt de représenter les fonctionnalités les plus communes dans les divers modèles de conception qui sont supportés dans les environnements d'apprentissage.

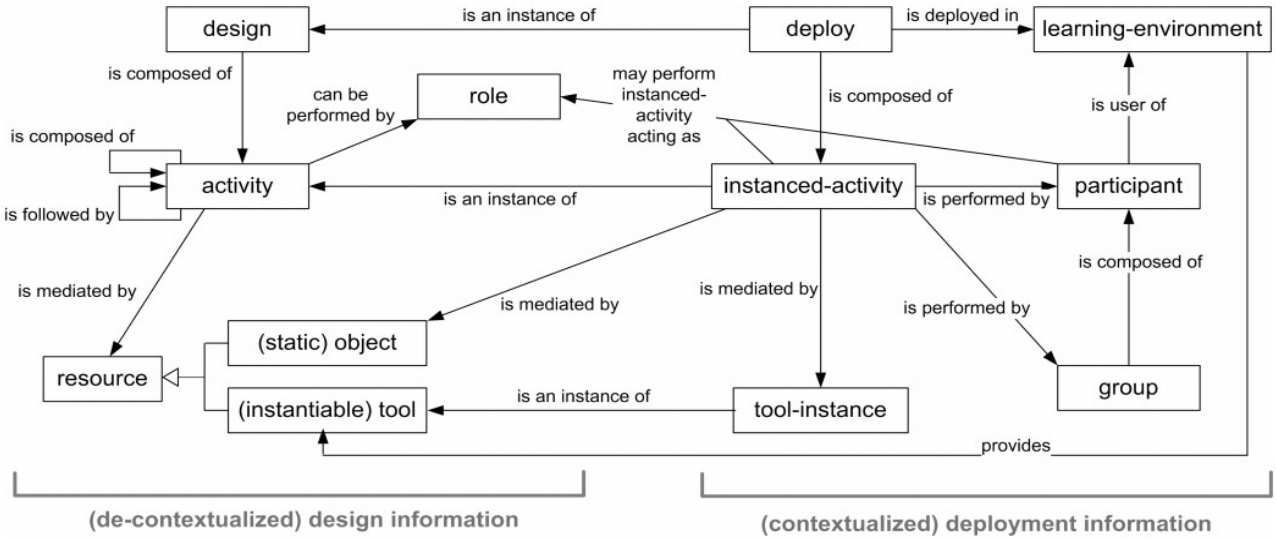


Figure 32: GLUE!-PS data model

Les auteurs de Glue! Ont conscience que ce modèle de données « intermédiaire » est un facteur contraignant de leur approche puisqu'il détermine quelles conceptions peuvent être exprimées et déployées avec Glue!PS (et quel niveau de fidélité entre le modèle initial et le modèle final). Ils considèrent (discuté dans certaines de leurs communications) que ce modèle de données est suffisamment expressif pour fournir des traductions « acceptables » des conceptions initiales

de l'enseignant en conceptions pour un VLE spécifique. Ils reconnaissent une « légère » perte d'information en général mais que dans la plupart des cas les déploiements correspondants possèdent les fonctionnalités essentiels du modèle de conception initial (à condition que le VLE soit compatible).

Exemple

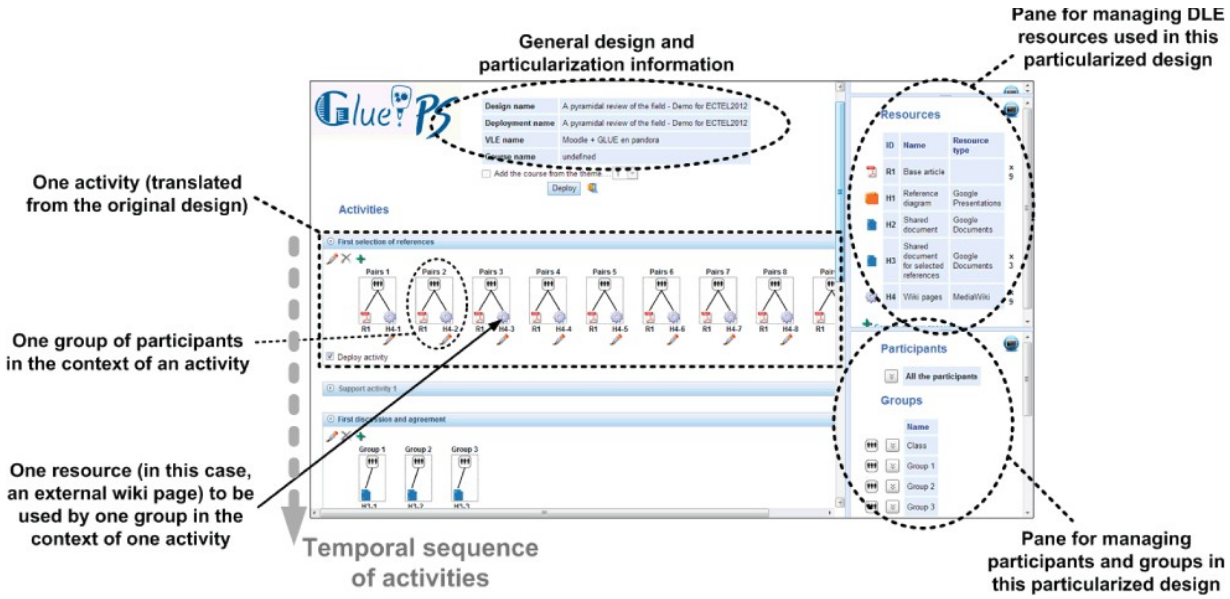


Figure 33: Interface du prototype d'éditeur (développé avec Java + AJAX + Web services) conforme au modèle d'implémentation de référence pour Glue!PS (source : **(Luis et al., 2013)**)

Discussions

Les adaptateurs LD de Glue!PS permettent d'ajouter des concepts/propriétés (sémantique) à des modèles de conception qui en sont dépourvus (par exemple ajouter des concepts de groupes). Cela permet d'enrichir les conceptions avec des éléments que le futur VLE saura prendre en compte. En revanche, dans le sens inverse, si le modèle de conception est plus riche (du point de vue « sémantique de conception pédagogique ») que le VLE alors les informations seront perdues. Ces pertes d'information sont reconnues et assumées, considérées comme faibles. C'est donc bien le principe du "langage pivot" qui est employé : perte pour aller vers ce langage puis perte pour aller vers le VLE. Le langage pivot derrière l'éditeur Glue!PS n'est pas voué à être utilisé pour élaborer à partir de zéro un modèle de conception : il s'agit ici d'adapter un scénario produit par d'autres VIDLs/éditeurs.

Glue!PS et Glue ! ont été expérimentés avec Moodle. Concrètement, l'opérationnalisation est réalisée, comme pour le projet précédent CADMOS, via la restauration d'un backup de cours Moodle, ce dernier étant généré par l'outil Glue!PS.

3.14 Flexo

Présentation

The FLEXO LD (Dodero et al., 2010) approach is based upon the definition of a generic LD language with a core set of elements related to the description of activities and learning flows, plus a number of add-ons that enable extended capabilities concerning the representation of roles, target language specific extensions and web service integration.

The FLEXO language is available in a dual environment, one that includes both visual as well as text-based course editing. It can generate and export EML- or LMS-specific deliverable courses. The solution provided by FLEXO is grounded on the conceptual model below.

Informations sur le méta-modèle

The FLEXO model is a framework in which IMS LD, Moodle and other formats can be readily understood and integrated. The core concept of the model is the *Activity* class. An activity can be associated with a set of *Resources*, which represent the learning objects and services required to deploy the activity in a web environment. Because the usual way to adapt the learning flow is through assessments, the *AssessmentActivity* does just that—it associates an *activity* with an *assessment* holding the *Parameters* required for adaptation on the flow of activities. The *Flow* concept represents the conditional *expressions* used to alter the flow. A set of *roles* can be defined and mapped to the *assessment activities* in order to provide advanced models of *assessment* (e.g. self-assessment, peer assessment and co-assessment). The *Assessment* class is a special kind of resource that is associated with an *Assessment Activity*. The operational semantics needed for evaluating flow conditions and choosing the correct conditional branch are the same as those of regular flow charts.

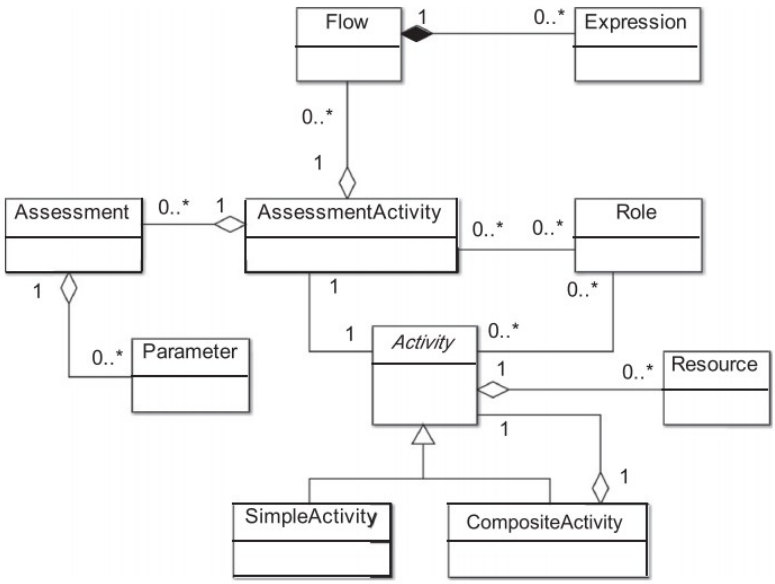


Figure 34: The FLEXO conceptual model.

Informations complémentaires sur l'utilisation du langage Flexo

The architecture of the Flexo authoring system is based on a visual language specification plus a text-based intermediate language (both consider the FLEXO conceptual model to be the basis of representation for the learning design of a course).

- The visual editing tool is the users' front-end when authoring a learning design. The visual language provides instructional designers with a source abstraction to edit common elements of a learning design, such as activities, roles, learning flows and assessment-based adaptations.
- The output of the visual tool is a text file described in an intermediate language generated from the conceptual model. These files can also be edited using a text-based language.

The authoring tool follows a forward and reverse engineering approach, in which text-based course specifications can be generated from visual designs and vice-versa.

From a textual LD specification, a set of target languages can be generated. The component in charge of this is a parser that follows the meta-model of the target language in which the course must be delivered.

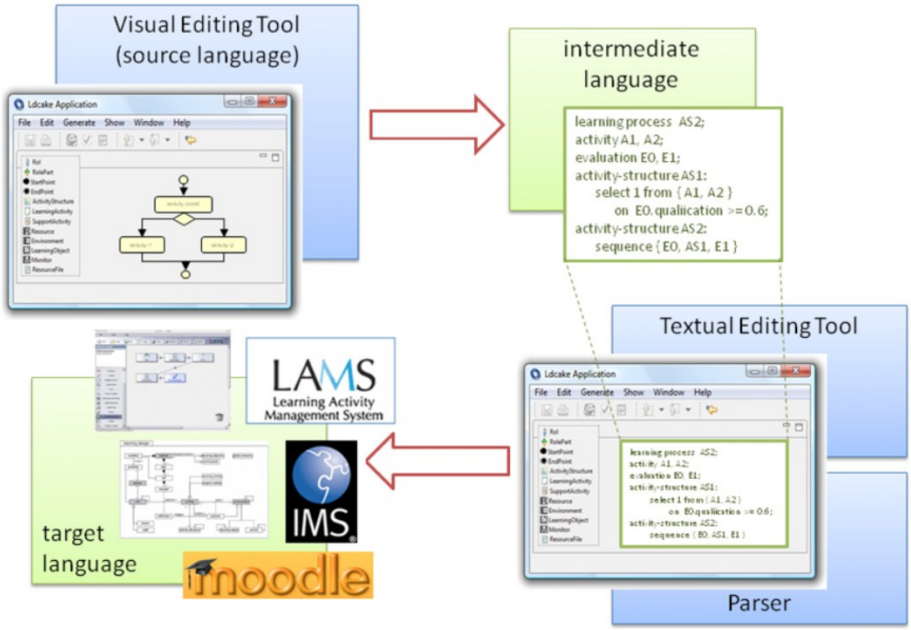


Figure 35: Main architectural components of the visual and generative LD editing system.

Exemples

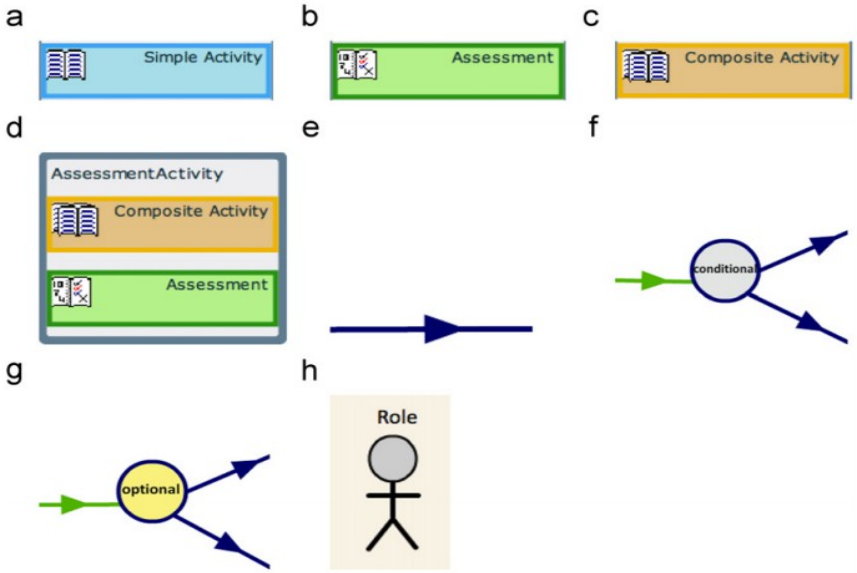


Figure 36: Summary of the FLEXO visual language elements

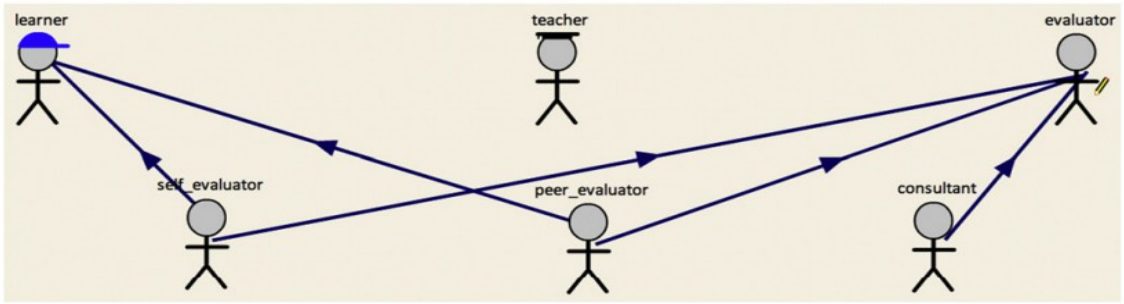


Figure 37: Defining a new set of roles from the default roles for a course

Relations avec les LMS

Flexo utilise un système d'annotation dans le format textuel de ces modèles de conception. Ce format permet d'exprimer des caractéristiques qui sont spécifiques au langage cible (langage d'un LMS ou d'un autre EML). Le mécanisme d'annotations sert également à intégrer des ressources ou services Web externes à travers leur URI ou interface ReST.

```
course courseID {
  assessment_activity assessmentActivityID {
    simple_activity simpleActivityID {
      title "Activity title"
      description "Activity description"
      resource "http://..."
    }
    assessment {
      //Evaluation params
      num grade [0,10]
      boolean finished
      collection scale ["good", "regular", "bad"]
    }
    flow:
    if(grade > 5 && finished) {
      then: otherActivity
    } else {
      flow: anotherActivity
    }
  }
}
```

Figure 38: Specification of activities & learning flow with the textual-based FLEXO language

Si le langage cible n'inclut pas une élément valide pour correspondre avec un concept d'un modèle de conception Flexo source alors cet élément est ignoré lors de la transformation. Dans leurs expérimentation sur Moodle ce cas a été rencontré à propos des concepts de « assessment activities » et de « conditional flows » car la version de Moodle qu'ils utilisaient ne permettaient de représenter que des séquences d'activités. En revanche, rien n'est précisé quant au sens opposé, c-à-d.si Flexo ne permet pas d'exprimer un concept exprimable dans le langage cible. Dans la logique de Flexo, le processus va d'un EML/VIDL vers un LMS. Si l'EML/VIDL d'origine ne prend pas en compte des éléments de conception du LMS alors il est toujours possible de les ajouter au niveau du modèle Flexo intermédiaire sauf si ce dernier n'a pas un équivalent dans le modèle de conceptuel de Flexo. Ces équivalences avec les LMS sont gérés par des extensions d'annotation spécifiques (cf. figure suivante).

```
course courseID {
  assessment_activity assessmentActivityID {
    @moodle:chat
    @moodle:text "This is a moodle activity"
    @moodle:visible yes
    @moodle:view_past_sessions yes
    simple_activity simpleActivityID {
      title "Activity title"
      description "Activity description"
      resource "http://..."
    }
    assessment {
      //Evaluation params
      num grade [0,10]
      boolean finished
      collection scale ["good", "regular", "bad"]
    }
    flow:
      if(grade > 5 && finished) {
        then: otherActivity
      } else {
        flow: anotherActivity
      }
  }
}
```

Figure 39: Specification of a Moodle-specific course extension.

Les auteurs de FLEXO ont théoriquement étudié les correspondances entre FLEXO et Moodle mais sur une version de Moodle sans restriction d'accès / suivi d'achèvements ou autres concepts récents permettant d'aller au delà du simple séquencement d'activités et ressources. Cette correspondance n'est toutefois pas très précise dans leurs communications et de nombreux points restent obscurs. Concernant le déploiement concret une expérimentation de transformation vers le format XML de backup de Moodle est précisé sans non plus davantage de précisions.

3.15 Le cas spécifique d'IMS-LD

Présentation

Il s'agit du standard *de facto* des EML. Il ne propose pas de notation visuelle car il est basé sur un binding XML. Toutefois, de nombreux éditeurs ont été créés pour faciliter la spécification de modèles compatibles. Certains de ces outils proposent alors parfois des notations visuelles similaires à une couche supplémentaire (*top-layer*) par dessus le méta-modèle d'IMS-LD.

The IMS learning design is a XML-based language for specifying learning content and processes. IMS refers to the *Instructional Managements Systems* Project which develops and promotes technical specifications for learning technology **(IMS Global, 2003)**. Historically the basis of IMS/LD is EML-OUNL (educational modeling language) **(Tattersall & Koper, 2003)** which was developed by the Open University of the Netherlands. It is independent of any specific pedagogy and its main focus is learning content, which is one important aspect of blended learning.

IMS/LD is a modeling language for elements and structure of learning units and learning processes; it is modeled who does what, when and which materials or learning services are used to achieve learning objectives **(Tattersall et al., 2003)**. Elements like resources, instructions for learning activities, templates for interactions and pedagogical models like problem-based learning, learning goals and outcomes as well as assessment tools are included **(IMS Global, 2004)**.

The specification gives a binding in XML, so that a XML document can be created for each learning process. This XML document can then be interpreted by an IMS/LD application.

Learning Design consists of three parts: Level A, B and C. There are different XML schemas provided for each level and each level extends the previous one. At level A time ordered activities which are performed by teachers and learners (role) are specified within an environment of learning objects and services **(IMS Global, 2003)**. At level B there are also properties (additional information about persons or roles and conditions) and conditions. Notifications, which are added at level C, can trigger new activities, e.g. a teacher has to answer if a question of a student occurs. Usually one starts the analysis of a didactical scenario with a narrative. In the next step semi-formal UML diagrams are drawn. They are more rigorous than a narrative, but less detailed than an XML document. On the basis of the UML activity diagram the XML document instance is created. Actual physical content is finally created based on the XML file, and the physical resources and the design are linked to each other and packaged in content files.

Informations sur le méta-modèle

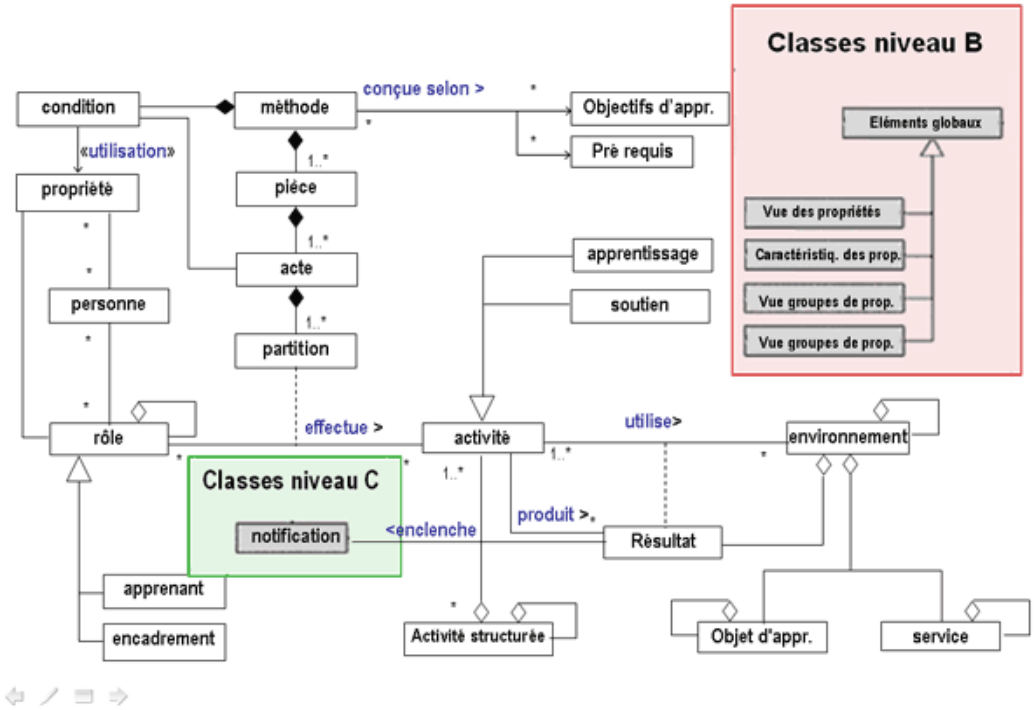


Figure 40: Le méta-modèle d'IMS-LD sur 3 niveaux

Exemples

Il convient de distinguer les *éditeurs* des *players* (Coppercore, Sled, EduBox, Reload LD player, GRAIL – pour .LRN). Parmi les éditeurs nous ne recensons que ceux développés explicitement pour supporter IMS-LD : l'éditeur MOT+ par exemple est parfois considéré comme un éditeur LD mais il s'agit plutôt d'un éditeur permettant un export vers le standard.

Reload Learning Design Editor

Développé à l'Université de Bolton par Phillip Beauvoir et Paul Sharples.
Il propose un environnement graphique permettant la conception de packages conformes à la spécification IMS LD.
La version 2 supporte les niveaux A, B et C.
Il permet l'importation et la création de packages IMS LD.
Il est possible de gérer plusieurs unités d'apprentissage simultanément.
Pour chaque unité d'apprentissage, on retrouve les onglets correspondants aux concepts principaux d'IMS LD : les rôles, activités, propriétés, environnements, ainsi que le scénario structuré en pièce, actes et partitions, correspondant au déroulement de l'unité d'apprentissage. Concevoir une unité d'apprentissage consiste ensuite à renseigner tous les champs, selon la spécification IMS LD.

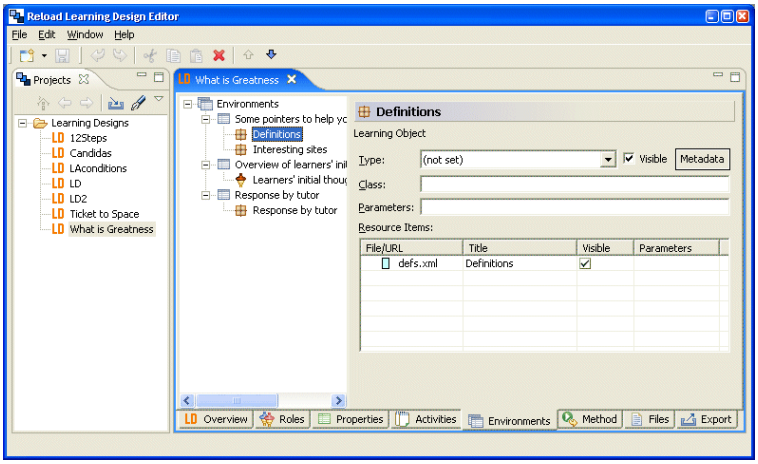


Figure 41: Reload Learning Design Editor

Recourse

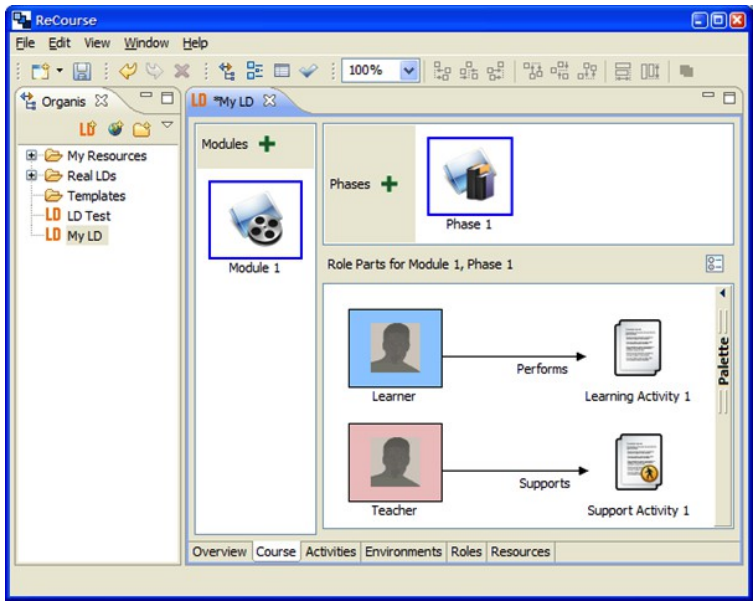


Figure 42: L'éditeur IMS-LD Recourse

ALFANET LD Editor

Développé pendant le projet ALFANET de l'UNED.
Fournit une interface permettant de concevoir, éditer et exporter des unités d'apprentissage compatibles avec IMS LD, niveau A seulement.
Comme dans Reload, concevoir une unité d'apprentissage consiste à compléter tous les champs, selon la spécification IMS LD.

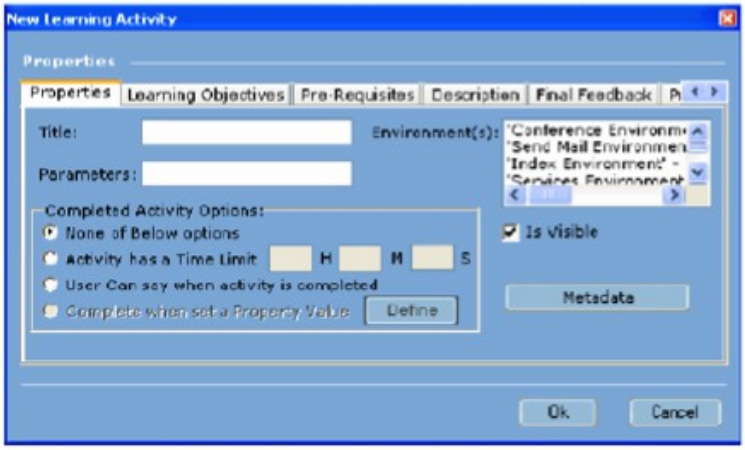


Figure 43: Alfanet LD editor

COSMOS Editor

Collaboration Script Modeling System était à l'origine conçu et implémenté pour modéliser les processus d'apprentissage collaboratifs.
Dans sa version actuelle, il permet de concevoir, éditer et sauvegarder des unités d'apprentissage conformes aux niveaux A, B de IMS LD.
A terme il devait être possible de charger et valider des unités d'apprentissage.
Il permet également de visualiser le code XML correspondant aux éléments décrits dans l'unité d'apprentissage.

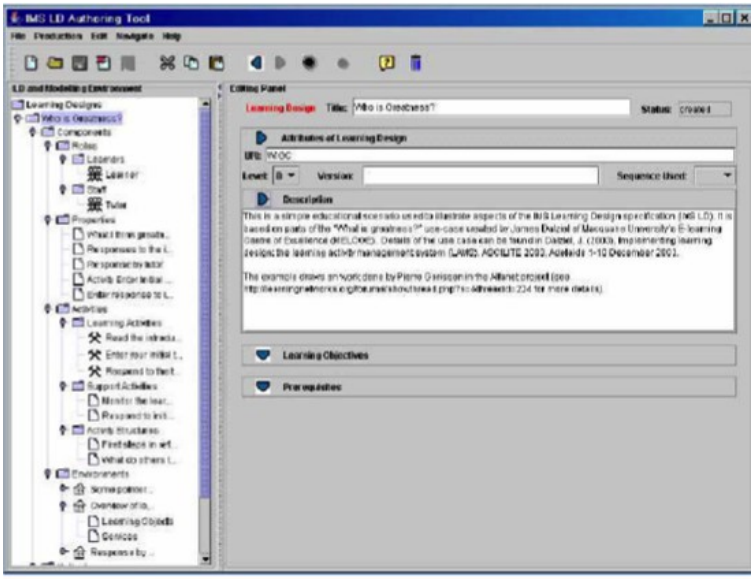


Figure 44: Cosmos Editor

CopperAuthor

Editeur IMS LD développé par l'Open University of Nederland - OUNL.
Il fournit une interface graphique permettant la conception, la validation d'unités d'apprentissage, la visualisation du code XML obtenu, et également la fusion d'unités d'apprentissage incomplètes.

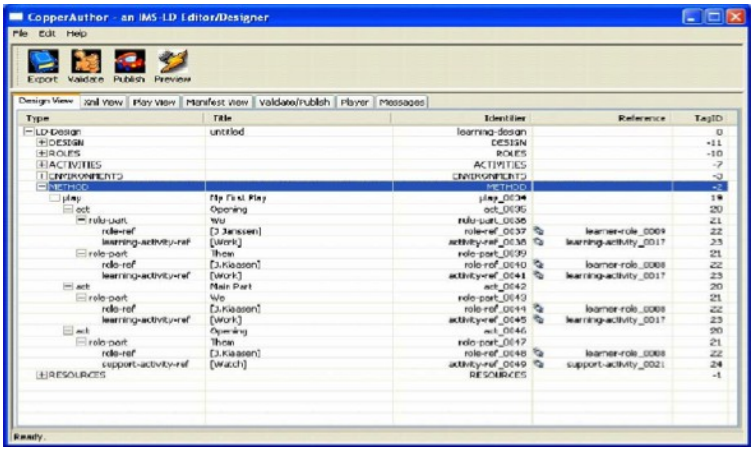


Figure 45: CopperAuthor

openGLM

Open Graphical Learning Modeller (OpenGLM) is an open-source learning design authoring tool which supports IMS Learning Design (LD) at levels A and B. The tool was conceived to facilitate non-IMS LD experts in creating, sharing and reusing units of learning. To achieve this, OpenGLM focuses on two features that differentiate it from most other IMS LD authoring tools. First, it adopts a visual modelling metaphor that conceals the complex and unintuitive elements and structures of IMS LD from the graphical user interface. Second, it provides built-in search, import and export access to an open repository which hosts more than 80,000 educational resources ranging from single learning objects to full online courses.
OpenGLM est en réalité le successeur de Prolix GLM (graphical learning modeller).

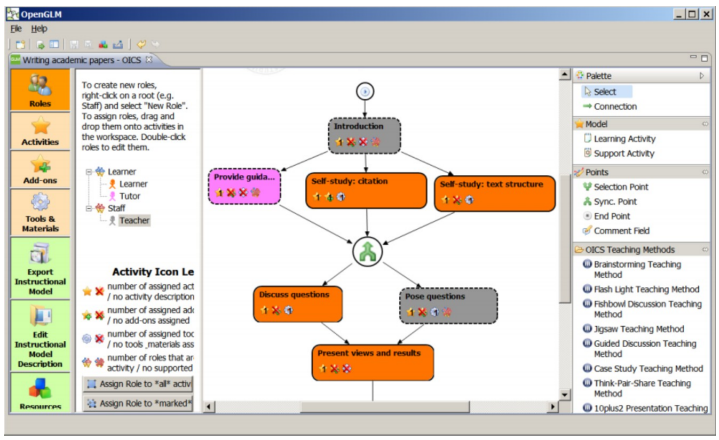


Figure 46: l'éditeur LD openGLM

3.16 Autres éditeurs

Nous proposons de rassembler dans cette section les éditeurs de scénarisation ne référençant pas directement un langage de modélisation pédagogique sous-jacent. Toutefois, cette absence de langage explicite n'implique pas l'absence de *modèle* de conception pédagogique en ce qui concerne l'approche suivie par l'outil.

3.16.1 CeLS

CeLS (Collaborative e-Learning Structures) is a web-based environment for creating and conducting structured asynchronous collaborative activities and incorporating them in the existing

instructional setting for all subjects and levels. CeLS is a web-based system designed to create and reuse activity structures; runable formats reflecting various collaborative instructional strategies e.g., creating and analyzing a common database, reaching an agreement, peer-product evaluation, contest, creating a group product.

CeLS is an executable XML-based model for collaborative Activity Structures, consisting of stages that are interconnected and based on each other.

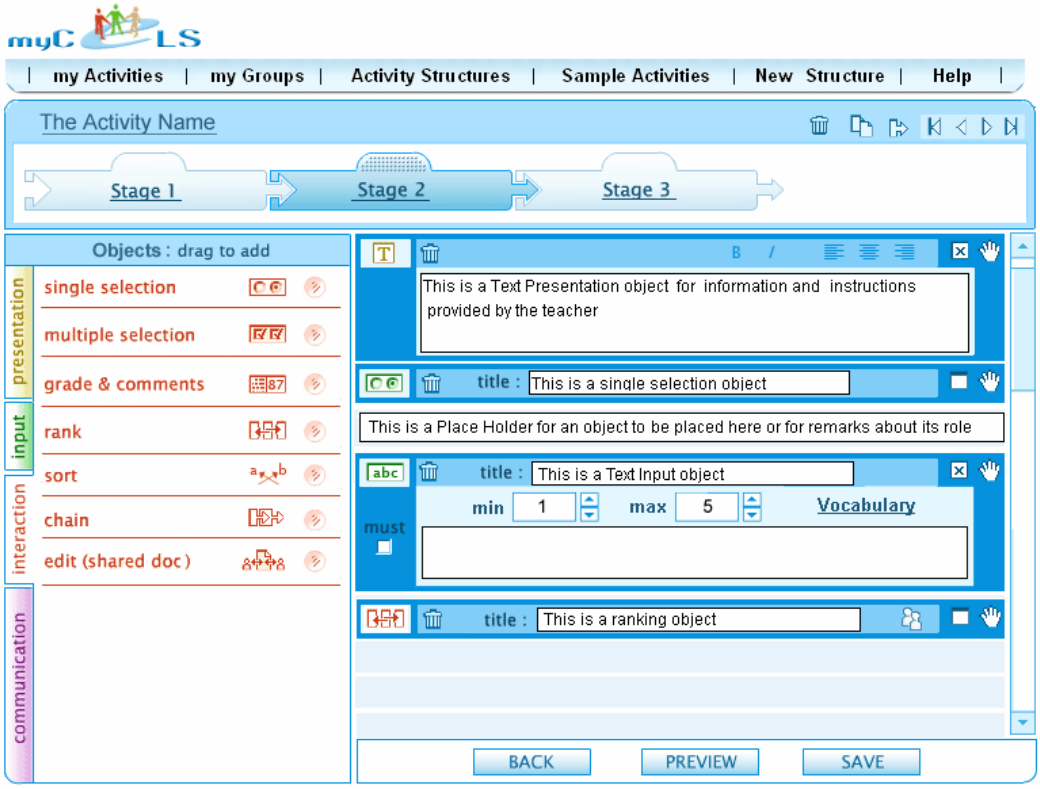


Figure 47: L'éditeur CeLS

3.16.2 CompendiumLD

Compendium LD or CompendiumLD is a learning design editor. It has been implemented on top of the Compendium mind mapping and argumentation software and can be used as a design tool.

Compendium LD is a rather high level design tool. It will not generate executable code like MOTPlus or the ReCourse editor, but it is much easier to use. However, since concepts maps can be exported as HTML pages with a menu, it might be possible to create a web site for an activity in the spirit of learning design, however the result would need some hand editing.

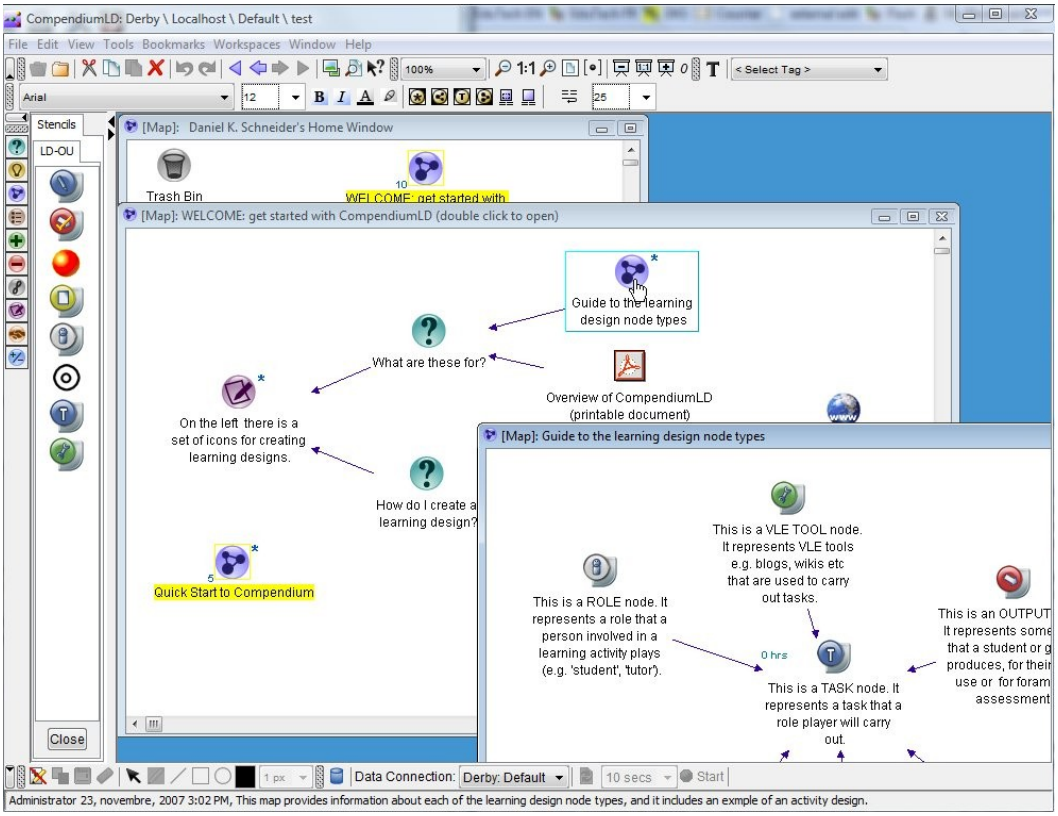


Figure 48: Compendium LD

3.16.3 DialogPlus Toolkit

The DialogPLUS Toolkit was a tool made for guiding and supporting teachers as they create, modify, and share learning activities and resources. DialogPLUS is an online browser-based application. This tool is partly inspired by IMS Learning Design and somewhat related toolkits like LAMS and MOT. The DialogPLUS learning design toolkit should guide practitioners through the process of creating pedagogically informed learning activities which make effective use of appropriate tools and resources. DialogPlus can export to IMS Learning Design.

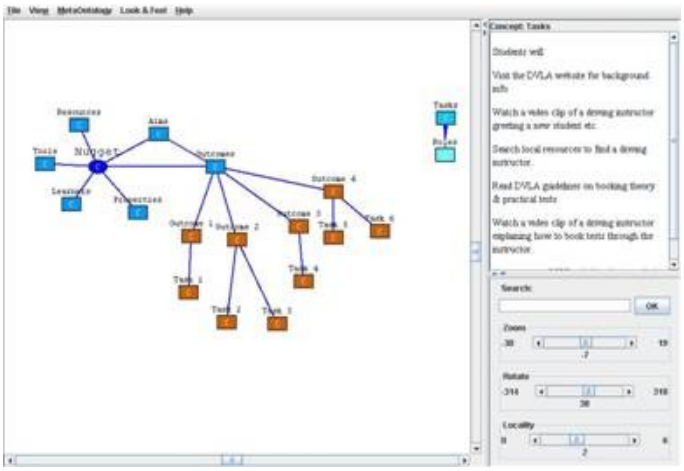


Figure 49: DialogPLUS Toolkit

4 Critères de catégorisation / taxonomie

Devant la multitude de VIDL de nombreux auteurs ont proposé différentes catégorisations et taxonomies afin de mieux les caractériser, les comparer mais également pour mieux les positionner les uns par rapport aux autres. La classification la plus utilisée est celle proposée dans (**Botturi et al., 06**). La prochaine section 4.1 présente la génèse de cette catégorisation qui sera présentée dans la section 4.2. Enfin, nous complétons cette section par deux autres tentatives de classification sous d'autres angles.

4.1 Vers une classification des VIDL

Gibbons **(2003)** a décrit un ensemble de couches pour représenter les différents sous-problèmes fonctionnels abordés globalement par la conception pédagogique. En effet pour lui la conception pédagogique n'est qu'un type de conception spécifique qui peuvent toutes être perçues comme des structures en couches. Pour Gibbons, chacune des couches qui suit correspond à un ensemble de problèmes et d'objectifs de conception spécifiques. Chaque couche peut être exprimée à l'aide de langages de conception qui lui sont dédiés. Il estime que des principes de sélection/arrangement et correspondances de structures sont définissables pour chaque couche ; des principes de relations structurels sont possibles également entre couches.

- **Content** layer. A design must specify the structures of the abstract subject-matter to be taught, must identify the units into which the subject/matter will be divided, and must describe how elements of subject-matter will be made available to instructional functions performed by other layers.
- **Strategy** layer. A design must specify the physical organization of the learning space, social organizations of participants, their roles and responsibilities, instructional goals, allocation of goals to timed event structures, and strategic patterns of interaction between the learner and the instructional experience.
- **Message** layer. A design must specify the tactical language of message structures through which the instructional experience can communicate content-derived information to the learner in conversational form.
- **Control** layer. A design must specify the language of control structures through which the learner expresses messages and actions to the source of the learning experience.
- **Representation** layer. A design must specify the representations that make message elements visible, hearable, and otherwise sense-able: the media representation channels to be used, the rule for assigning message elements to media channels, the form and composition of the representation, the synchronization of messages delivered through the multiple channels, and the representations of content.
- **Media-logic** layer. A design must specify the mechanism by which representations are caused to occur in their designed or computed sequence.
- **Data management** layer. A design must specify data to be captured,archived, analyzed, interpreted, and reported.

Le concept des couches de conception constitue une théorie structurante pour la création de conception pédagogique. Chaque couche a ses propres questionnements mais participe à la conception globale.

Layer	Objective	Examples of activities	Examples of design outcomes
1. Content	Define the content and structure of the domain (learning goals in terms of know how and know that)	Task analysis, content analysis, concept mapping, model analysis	Task hierarchies, mental model descriptions
2. Strategy	Define the instructional design (how should be learned)	Identification of a global instructional design model, definition of and sequencing of learning tasks, definition of social relationships during instruction, definition and sequence of time-event structures, definition of roles, goals and initiative-sharing during instruction	Task classes, case descriptions, feedback mechanisms
3. Control	Define the ways a learner with the system, i.e. his "command language"	Identification of user actions, definition of control space, flow planning	Content controls, strategy controls, administrative controls
4. Message	Define the message design (what should be sensed)	Definitions of message structure, composition of elements and rules	Message standards design for content
5. Representation	Define the representation design (how should it be shown)	Media selection, selection of production tools and methods	Layout standards, media channel assignment, media synchronisation methods
6. Media-logic	Define the software architecture (how should the programme be structured)	Definition of logic structure, algorithms creation, learning objects definition	Modularity plan, Packaging method, software platform selection, maintenance plan
7. Data management	Define the data management (how should information, captured during instruction, be organised, analysed, stored and reported).	Defining administration processes, data base selection, definition of data items, capture, filtering, storage, analysis, interpretation, compilation and sharing	Security plan, billing methods, metadata assignment

Un framework similaire, plus simple, a été décrit dans le modèle conceptuel de la méthode MISA dans lequel la conception relevait de 4 couches : Content, Pedagogical specifications, Materials et Delivery.

Gibbons and Brewer (2005, p. 113) ont décrit plusieurs dimensions ou variantes pour les langages de conception :

- Complexité – simplicité
- Précision – non-précision
- Formel – informel
- Personnalisation – partage
- Implicite – explicite

- Standardisation – non-standardisation
- Opérationnalisable – non opérationnalisable

All of these characteristics can be seen in instructional design languages. The languages related to high-tech media design and production tend to be complex, precise, formal, shared, explicit, standard, and computable. Languages that describe instructional strategy structures tend to be relatively simple, non-precise, informal, often personalized or stylized, implicit, non- standard, and non-computable.

4.2 La classification de Botturi

Luca Botturi, en plus d'être à l'origine du VIDL E2ML, a proposé l'esquisse d'un cadre de classification pour les VIDL repris dans (**Botturi et al., 06)**. Cette classification s'appuie sur cinq critères :

Stratification (nominal: *flat*, *layered*). A layered language offers a set of tools or representations for describing entities of different types, such as people and roles, activities, or learning materials. On the other hand, a flat language would collect entities of all types into a single representation. For example, UML takes a layered perspective.

La stratification aide donc une équipe de conception à déterminer les relations entre le niveaux d'analyse, permettant ainsi à chaque concepteur de situer son regard à un niveau donné (plus pédagogique ou plus fonctionnel/technique).

Formalization (interval: *formal*, *informal*). A formal language defines a stringent, closed set of concepts and rules for composition of concepts in order to describe designs. For instance, XML or UML are formal languages, while sketches or dialogs are more open and informal. Other design languages may combine formal and informal descriptions.

Elaboration (ordinal: *conceptual*, *specification*, *implementation*). Each particular design language is able to provide more or less detail of a specific artifact. The three levels of elaboration are taken from Fowler (2003): The conceptual level allows for a general, aggregate view on the design, indicating its rationale and main elements; the specification level provides means for a more comprehensive description, including all elements; the implementation level represents the highest level of detail achieving maximum precision.

Perspective (nominal: *single*, *multiple*). While layered languages foresee the use of multiple representations for different entities, multiple-perspective languages exploit different tools for representing more than one view on the same entities. For example, E2ML offers two overview diagrams, one for chronological relationships among learning activities, and one for structural relationships.

Notation System (nominal: *none*, *textual*, *visual*). If a language exposes a notation system, this can be primarily non-visual (=textual, e.g. IMS/LD) or visual (e.g. UML).

Ci-après une illustration du positionnement de plusieurs VIDL par rapport à cette classification.

	Niveau de stratification	Niveau de formalisation	Niveau d'élaboration	Nombre de perspectives	Notation
E^2ML	plat	semi-formel	conceptuel	plusieurs	visuel
PCeL	plusieurs couches	semi-formel	conceptuel	une	visuel
AUTC	plat	informel	spécification	plusieurs	visuel
IMS-LD	plusieurs couches	formel	spécification	une	textuel
PoEML	plusieurs couches	formel	implémentation	plusieurs	visuel
UML	plusieurs couches	formel	conceptuel / spécification	plusieurs	visuel

Figure 50: Exemples de classification des VIDL

A ces cinq critères de classification sont également ajoutés 2 autres critères par le même auteur dans des travaux plus récent. Ces critères additionnels concernent une classification bi-dimensionnelle des usages possibles des VIDLs à supposer que chaque langage est toujours développé dans un but bien précis à l'esprit.

Communication. The first axis in the application framework concerns the main objective of the ID language, with two values: (a) Reflective (personal) means that the language is used primarily for personal creative thinking. This is useful for formally-bent or visually-oriented people and for designers in the first conceptual stages of design in which they do not yet collaborate with other designers and stakeholders; (b) Communicative (community) indicates that the language is used to communicate with other designers or stakeholders. This is useful for interdisciplinary design teams involving different views/roles.

Creativity. The second axis describes the relationship between the design language and the generation of design solutions: (a) Generative means that the language can be used as a means of exploring the design space and creating and refining design solutions and alternatives, e.g. during redesign. (b) Finalist means that it is used to formalize and “freeze” the final design solution, e.g., for creating a final IMS/LD specification of an e-learning module.

La figure suivante montre quelques comparaisons de VIDLs à travers ces 2 critères.

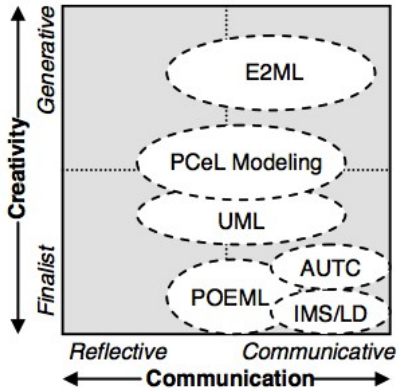


Figure 51: Exemples d'utilisation des critères communication et créativité pour comparer des langages de conception.

4.3 Autres catégorisations

4.3.1 Différentes représentations et notations

T. Nodenot propose dans (Nodenot, 07) sept types de représentations visuelles pour les VIDL :

1. La représentation des **rôles et responsabilités** que prennent les acteurs dans les situations d'apprentissage décrites (différenciation apprenant/tuteur, *a minima*, à la hiérarchie de rôles contextualisés par exemple);
2. La représentation des **modalités d'apprentissage** prescrites pour chaque module d'enseignement identifié (le déroulement dans le temps des activités du module, la synchronisation des activités conduites par différents acteurs, la différenciation des activités d'apprentissage médiatisées par l'outil informatique, etc.).
3. La représentation des **connaissances** sous-jacentes aux activités d'apprentissage prescrites (par exemple relier les activités prescrites aux concepts du domaine, aux savoirs et savoir-faire liés à ce domaine) ;
4. La représentation précise des **buts d'apprentissage** (objectifs, prérequis/postrequis, etc.);
5. La représentation de la **structure d'un module d'apprentissage** sous forme d'activités inter-reliées par des relations de dépendance fonctionnelle ou de composition;
6. La représentation des **collaborations synchrones/asynchrones** entre acteurs participant à une situation d'apprentissage (pour leur assigner leurs responsabilités respectives par exemple);
7. La représentation de la **diffusion** au sens de la façon dont les activités d'apprentissage ainsi prescrites vont finalement être regroupées en paquetages puis mises en œuvre sur une infrastructure logicielle donnée grâce aux composants et services fournis par cette infrastructure.

   La figure suivante, extraite de (Nodenot, 07), positionne plusieurs VIDLs par rapport à ces 7 représentations.

	E²ML	CPM	coUML	MOT+	PoEML	ASK-LDT	COLLAGE
Rôles et responsabilités		+	+		+	+	+
Modalités d'apprentissage	+		+			+	
Domaine, connaissances	+	+	+	+			
Objectifs et buts d'apprentissage	+	+		+			
Structure du cours	+			+	+	+	
Collaborations entre acteurs	+	+	+			+	+
Liens avec l'infrastructure		+	+	+	+	+	

Figure 52: Positionnement de plusieurs VIDL selon les différentes représentations proposées par **(Nodenot, 07)**

4.3.2 La catégorisation selon la centration « métier »

Parmi les différentes catégorisations possibles des scénarios pédagogiques / langages de scénarisation pédagogique, **(Laforcade et al., 07)** propose trois types de scénarios selon une séparation de préoccupations orientée " métier ". Chacun des trois modèles s'adresse à des communautés différentes et répond à des enjeux différents : ils correspondent à des " métiers " différents.

- Scénario à niveau connaissance (SC) : son vocabulaire " métier " est celui d'une équipe pluri-disciplinaire de conception, des communautés d'enseignants-concepteurs : un langage à niveau connaissance doit pouvoir capturer leur " métier ", leur vocabulaire commun, qu'il soit lié à des approches pédagogiques particulières, à des plates-formes de formation connues, etc. L'objectif d'un tel language est de faciliter la définition même du scénario pédagogique, c.-à-d. de guider l'activité de conception.
- Scénario abstrait (SA) : son vocabulaire cherche à être indépendant de celui de la plate-forme de formation (ou de tout autre dispositif) dans le but de faciliter sa réutilisation ; il est donc possible que le vocabulaire d'un EML abstrait cherche aussi à s'abstraire d'approches pédagogiques trop spécifiques. L'objectif principal est l'interopérabilité des dispositifs et l'abstraction des EML spécifiques à un domaine.
- Scénario spécifique (SS) à un dispositif/plate-forme de formation : son vocabulaire est conforme à celui d'une plate-forme bien précise. L'objectif est alors de servir de guide pour la configuration du dispositif (manuellement ou automatiquement selon le " format " du scénario).

A ces 3 catégories est également proposé 2 types de notation pour chacune d'entre elles :

- **visuel**, ayant une notation textuelle ou graphique dans le but de rendre le scénario

interprétable par les membres de la communauté visée (enseignants, concepteurs, etc.)

- **forme informatique standard** dans le but de rendre le scénario spécifié interprétable sans ambiguïté par la machine (explicitation maximale des concepts) et/ou qu'il soit facilement stocké, recherché et échangé.

Pour le second cas, la notation doit être basée sur un format “standard”, non propriétaire (généralement XML), permettant d’expliciter les concepts/relations embarqués. De ce point de vue, un scénario IMS-LD est un fichier XML conforme au schéma proposé par IMS, tandis qu’un fichier XML décrivant un diagramme en termes de primitives graphiques (par exemple le format SVG) n'en est pas pour autant un modèle de scénario productif.

Deux types de transformations sont également proposées

- la transformation de l'un de ces scénarios vers une autre catégorie (transformation **extra-domaine**) permet à la fois d'atteindre les objectifs de l'autre catégorie mais aussi de toucher des communautés “métiers” différentes (dans un souci de partage et de réutilisation) ;
- transformations intra-domaine afin de représenter visuellement un scénario décrit dans un format informatique standard tout aussi bien que l'inverse, c.-à-d. retranscrire le scénario visuel dans un format informatique standard indépendant de toute représentation visuelle.

A titre d'illustration, la figure suivante présente les trois types de scénarios métiers, les deux formalismes par catégorie, ainsi qu'une proposition de positionnement des EML actuels (langages et outils).

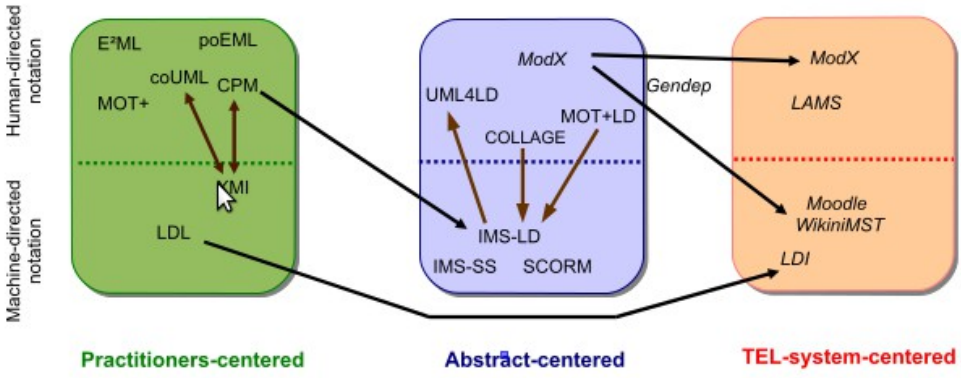


Figure 53: Catégorisation selon la centration métier et l'intention de la notation

4.3.3 Le modèle des 3D pour la documentation des conceptions

Le modèle « 3D » pour documenter les conceptions pédagogiques est proposé dans **(Boot et al., 2007)**. Il s'ait d'une sorte de framework pour que des concepteurs de langages/éditeurs de conception pédagogique puissent discuter et améliorer leurs langages et documents : “*Designers and producers of instructional materials lack a common design language. As a result, producers have difficulties translating design documents into technical specifications. The 3D-model is introduced to improve the stratification, elaboration and formalisation of design documents*”.

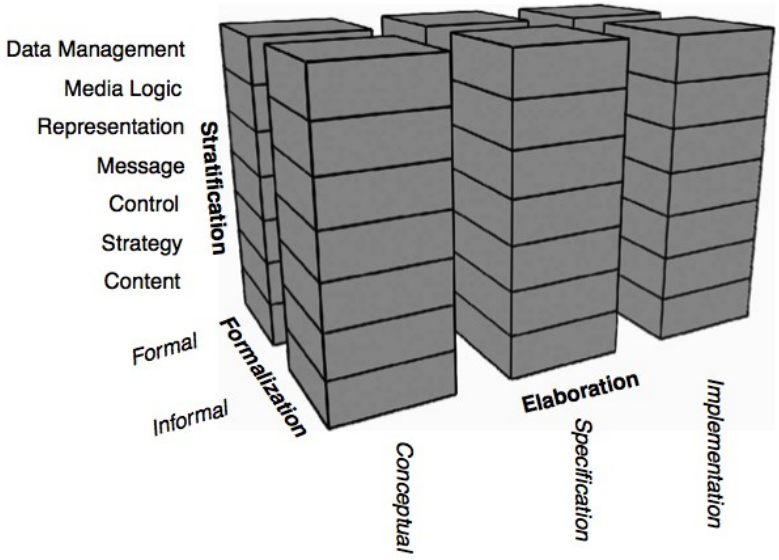


Figure 54: Le modèle 3D de documentation de conception **(Boot et al., 2007)**

Ce modèle propose 3 dimensions complémentaires pour décrire où se situe un langage/outil de conception pédagogique :

- la **stratification** : elle correspond aux 7 niveaux de **(Gibbons & Brewer, 05)**
- le degré **d'élaboration** : les 3 niveaux de **(Fowler, 03)**
- le niveau de **formalisation** : formel/informel.
- Ce modèle s'intéresse davantage à la qualification de toute forme de production abordant la conception pédagogique. Ce modèle est très proche de la classification de **(Botturi et al., 06)** dont Boot est l'un des auteurs.

5 Conclusion : vers quels types de VIDLs centrés LMS ?

5.1 Selon la classification de VIDL la plus usitée

Si l'on se réfère à la classification de VIDL la plus complète et utilisée, c'-à-d. **(Botturi et al., 06)**, les VIDLs que cherchent à produire le projet devraient avoir les caractéristiques suivantes :

- **Stratification** : en **couches.** Il est important que le langage de conception que l'on proposera offre la possibilité aux concepteurs de ne pas devoir spécifier tout leur scénario sur un seul et unique diagramme. L'utilisation de plusieurs couches comme différents diagrammes inter-reliés permettra de séparer les points de vue sur la conception comme de séparer les points de vue « analyse » en proposant des couches de différents niveaux (plus pédagogique ou plus fonctionnel/technique).

- **Formalisation** : elle devra être **formelle** au sens où les modèles produits devront être manipulés informatiquement sans ambiguité. En revanche cela n'empêche pas que les langages de conception qui seront proposés puissent embarquer des descriptions plus ou moins formelles (au sens « label ») qui seront traitées comme tel.
- **Elaboration** : si l'on considère le niveau d'informations du futur modèle celui-ci devra être à un niveau **implémentation** : il devra inclure tous les éléments de conception à un niveau de détail permettant la réalisation automatique des correspondances pour les plateformes de formation. Si l'on considère le niveau d'informations perçu par l'enseignant-concepteur alors il s'agit des niveaux **spécification** et **implémentation**, au sens où les concepteurs pourront spécifier certaines portions de leurs modèles à un niveau proche de celui de la plateforme cible, tandis que d'autres portions du modèle seront spécifiées en détail à un niveau plus abstrait de la plateforme (la correspondance vers la plateforme étant en partie réalisée pendant la conception (au runtime), l'autre lors de l'export du modèle vers un format compréhensible par l'API d'import de la plateforme).
- **Perspective** : sur ce point nous n'avons pas de contraintes au préalable. Une perspective multiple pourrait permettre de séparer différents points de vue d'un même élément de conception (comme les aspects structurels des activités pédagogiques, d'une part, et leurs aspects temporels, d'autre part). Toutefois, une perspective simple pourrait suffire si la notation et l'ergonomie finale n'apportent pas de confusion pendant l'utilisation du VIDL.
- **Notation** : **visuelle** (et plus précisément orientée diagrammes) puisque à destination principale d'enseignants-concepteurs mais également **textuelle** au sens où les modèles produits devront être spécifiés dans un format qui fait sens à la machine.
- **Communication** : les langages et éditeurs prévus par le projet sont essentiellement à usage **personnel.**
- **Créativité** : les langages et éditeurs prévus par le projet seront outillés selon un cadre technologique DSM (EMF, GMF, SIRIUS, ATL...) qui sont destinés à formaliser et « fixer » les modèles plutôt qu'à créer/raffiner des solutions : **finalisation.**

5.2 Selon les autres classifications

Si l'on se réfère aux niveaux de **stratification** de Fowler, repris dans le modèle 3D de Boot, l'objectif des VIDLs du projet est de couvrir un maximum de précisions sur la conception de la situation d'apprentissage souhaité sur la plateforme de formation. Les VIDLs que nous envisageons devraient donc aborder des aspects relevant des couches **Content** / **Strategy.**

En revanche nous savons d'ores et déjà que le positionnement des VIDL envisagé dans le projet suit les contraintes suivantes :

- le modèle de conception doit être formalisé concrètement dans un format XML qui sera manipulée par l'API d'import (développée dans le cadre du projet pour plusieurs plateformes) : il n'est donc pas envisagé de considérer les ressources physiques dans les futurs modèles de conception (les noms et descriptions des ressources *logiques* mais pas les ressources *physiques*) ;
- l'objectif de la conception est de pré-configurer l'espace-cours d'une plateforme de formation : le résultat final prend en compte le point de vue des usagers finaux (tuteurs, apprenants) ;

- un focus sur la régulation de la situation pédagogique n'est pas envisagé dans le projet, idem pour un focus sur l'observation pendant et *a posteriori* du déroulement de la situation d'apprentissage.
- l'expressivité embarquée dans le VIDL sera dépendante et limitée à celle permise par la plateforme cible.

Les dimensions **Message** / **Control** / **Representation** / **Media-logic** / **Data management** pourraient donc être partiellement abordées.

Selon les différentes notations proposées par **(Nodenot, 07)**, les VIDLs visés par le projet devraient aborder les dimensions **rôles et responsabilités**, **modalités d'apprentissage** prescrites, **buts** d'apprentissage, **structure** d'un module d'apprentissage, bien entendu en relation étroite et cohérente avec ce que la plateforme de formation est capable d'exprimer. Selon l'expressivité que nous rencontrerons pour les plateformes étudiées, nous essaierons d'aborder également la représentation des **collaborations** synchrones/asynchrones entre acteurs participant à une situation d'apprentissage. En revanche nous avons déjà écarté la représentation des **connaissances** car celle-ci n'est généralement pas fonctionnellement couverte par les plateformes de formation. Enfin, la représentation de la **diffusion** devrait également être prise en compte dans nos VIDLs en permettant aux concepteurs de spécifier ou de visualiser tout ou partie de leurs conception en terme de configurations plateformes correspondantes.

Selon la classification basée sur la centration du métier de conception, les VIDLs que le projet vise à élaborer sont centrés sur le métier de conception de la plateforme mais ne sont pas limités à l'expressivité de celui-ci : l'objectif est de s'abstraire de ce métier orientée infrastructure concrète de réalisation pour se diriger vers un métier de conception plus abstrait, mettant davantage en avant des dimensions pédagogiques. Les VIDLs seront donc **centrés plateformes mais dirigés vers les besoins de conception** des enseignants utilisant la plateforme cible. Les notations seront visuelles, graphiques et orientées diagrammes. Toutefois les outils-auteurs qui seront développés pour supporter les VIDLs devront proposer un formalisme de **persistance** afin que les modèles de conception visuels soient traduits dans un format complètement interprétables par la machine (fonctionnalité de l'outillage DSM qui sera exploité dans le cadre du projet) mais également exploitables par la plateforme pour configurer l'espace-cours correspondant : une sorte de **transformation externe** sera donc nécessaire.

6 Références

Alario Hoyos, C. (2012). GLUE!: An Architecture for the Integration of External Tools in Virtual Learning Environments, PhD Thesis, School of Telecommunication Engineering, June.

Bajnai J., Karagiannis D., Steinberger C. (2005) eduWeaver – the Courseware Modeling Tool. In: Akoka, J. et al. (ed.) Perspectives in Conceptual Modeling, ER 2005 Workshops. Springer

Botturi, L. (2003). E2ML - Educational Environment Modeling Language. ED-MEDIA 2003: World Conference on Educational Multimedia, Hypermedia & Telecommunications, Honolulu, Hawaii, USA.

Botturi, L. (2004a). Visual Languages for Instructional Design: an Evaluation of the Perception of E2ML. ED-MEDIA 2004: World Conference on Educational Mulitmedia, Hypermedia & Telecommunications, Lugano, Switzerland.

Botturi, L. (2004b). The Quail model for the classification of learning goals. Lugano: Università della Svizzera italiana.

Botturi, L., & Belfer, K. (2003). Pedagogical Patterns for Online Learning. E-Learn'03 - World Conference on E-Learning in Corporate, Government, Healthcare, & Higher Education, Phoenix, Arizona, USA.

Botturi, L. (2005). A Framework for the Evaluation of Visual Languages for Instructional Design: the Case of E2ML. Journal of Interactive Learning Research. 16 (4), pp. 329-351. Norfolk, VA: AACE.

Botturi, L. (2006). E2ML. A visual language for the design of instruction. Educational Technologies Research & Development, 54(3), 265-293.

Botturi, L., Derntl, M., Boot, E., & Figl, K. (2006). A Classification Framework for Educational Modeling Languages in Instructional Design. Proceedings of The 6th IEEE International Conference on Advanced Learning Technologies, July 5–7, Kerkrade, the Netherlands, 1216-1220.

Botturi, L., Stubbs, T. (eds.) (2007). Handbook of Visual Languages in Instructional Design: Theories and Pratices. Hershey, PA: Idea Group. 28 August 2008 (UTC). ISBN 978-1-59904-729-4

Botturi, L., Burgos, D., Caeiro, M., Derntl, M., Koper, R., Parrish, P., Sodhi, T. & Tattersall, C. (2008). Comparing Visual Instructional Design Languages. In L. Botturi & T. Stubbs (eds.), Handbook of Visual Languages in Instructional Design: Theories and Pratices. Hershey, PA: Informing Science Reference, 315-343.

Caeiro, M. (2008). PoEML: a separation-of-concerns proposal to instructional design}}, Handbook of Visual Languages for Instructional Design: Theories and Practices. IDEA Group Inc.

Derntl, M. (2005). Patterns for Person-Centered e-Learning. Unpublished PhD Thesis, University of Vienna, Vienna, Austria.

Derntl Michael & Renate Motschnig (2007). coUML - A Visual Language for Modeling Cooperative Environments. in L. Botturi & T. Stubbs, Handbook of Visual Languages for Instructional Design: Theories and Practices, Information Science Reference, 155-184. ISBN 1599047292.

Dessus, Philippe et Schneider, Daniel Scénarisation de l'enseignement et contraintes de la situation, (2006) In J.-P. Pernin & H. Godinet (2006). (Eds.), Colloque Scénariser l'enseignement et l'apprentissage : une nouvelle compétence pour le praticien ? (pp. 13-18). Lyon : INRP.

Dodero, J.M., Martinez Del Val, Á. and Torres, J. (2010): An extensible approach to visually editing adaptive learning activities and designs based on services. Journal of Visual Languages & Computing 21(6): 332–346.

Figl, K. & Derntl, M. (2006). A Comparison of Visual Instructional Design Languages for Blended Learning. In E. Pearson & P. Bohman (Eds.), Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2006 (pp. 941-948). Chesapeake, VA: AACE.

Fowler, M. (2003). UML distilled: a brief guide to the standard object modeling language. Boston, MA: Addison-Wesley Professional.

Perez-Rodriguez R., Caeiro-Rodriguez M., Anido-Rifon L., Llamas-Nistal M. (2010). Execution Model and Authoring Middleware Enabling Dynamic Adaptation in Educational Scenarios Scripted with PoEML, in Journal of Universal Computer Science, vol. 16, no. 19 (2010), 2821-2840

Gibbons, A. S. (2003). What and how do designers design? A theory of design structure. TechTrends, 47(5), 22–27.

Gibbons, A. S., Brewer, E. K. (2005). Elementary principles of design languages and notation systems for instructional design. In J. M. Spector, C. Ohrazda, A. Van Schaack, & D. Wiley (Eds.), *Innovations in instructional technology : Essays in honor of M. David Merrill*, Mahwah, NJ : Erlbaum.

Gibbons, A. S. and Rogers, P. C. (2006). Coming at Design from a Different Angle: Functional Design. Paper presented at the AECT Summer Research Symposium, June 22–25, Bloomington, IN.

Goodwin, C. (1994). Professional vision. Am. Anthropol., 96(3), pp. 606–633.

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

Hernández-Leo, D., Villasclaras-Fernández, E.D., Asensio-Pérez, J.I., Dimitriadis, Y. (2007). Diagrams of

learning flow patterns solutions as visual representations of refinable IMS Learning Design templates in *Handbook of Visual Languages for Instructional Design: Theories and Practices*, Botturi, L., Stubbs, T. (Eds.), IGI, pp. 394-412.

IMS Global. (2003). IMS Learning Design Best Practice and Implementation Guide. Retrieved Nov 21, 2003, from http://www.imsglobal.org/learningdesign/ldv1p0/imsld_bestv1p0.html IMS Global. (2004).

IMS (2003). IMS Learning Design Version 1.0 Final Specification, Technical report.

Karagiannis, D., & Bajnai, J. (2001). ADVISOR - An Educational Management Tool. Symposium Towards the New Education Society, Zvolen, Slovakia.

Karagiannis, D., & Bajnai, J. (2004). eduXX - The Instructional Design Platform. International Conference on Education and Information Systems, Orlando.

Katsamani, M., Retalis, S., (2012). Designing a Moodle course with the CADMOS learning design tool, Educational Media International, Taylor & Francis Group, 49:4, pp. 317-331.

Laforcade, P. (2004), Modélisation et méta-modélisation UML pour la conception et la mise en œuvre de situations-problèmes cooperatives, Ph.D. Thesis, University of Pau et des Pays de l'Adour, http://www.univ-pau.fr/~laforcad/publications/Pierre_Laforcade-These.pdf

Laforcade, P., Nodenot, T., Sallaberry, C. (2005) Un langage de modélisation pédagogique basé sur UML, Revue STICEF, Volume 12, 2005, ISSN : 1764-7223, mis en ligne le 08/11/2005, http://sticef.org

Laforcade P, Barré V, Zendagui B. (2007) Scénarisation Pédagogique et Ingénierie Dirigé par les Modèles : Cadre d'étude pour la définition de langages et environnements-outils de scénarisation pédagogique spécifiques à des domaines. Actes d'EIAH'07, Lausanne (Suisse), 27-29 juin.

Luis P. Prieto, Juan I. Asensio-Perez, Juan A. Munoz-Cristobal, Yannis A. Dimitriadis, Ivan M. Jorrin-Abellan, Eduardo Gomez-Sanchez, "Enabling Teachers to Deploy CSCL Designs across Distributed Learning Environments", IEEE Transactions on Learning Technologies, vol.6, no. 4, pp. 324-336, Oct.-Dec. 2013, doi:10.1109/TLT.2013.22

Martel, C., Vignollet, L., Ferraris, C., David, J.P., Lejeune, A.: Modelling collaborative learning activities on e-learning platforms. In: the 6th IEEE International Conference on Advanced Learning Technologies, ICALT 2006, 5-7 July 2006, Kerkrade, The Netherlands.

Massironi, M. (2002). The psychology of graphic image :Seeing, drawing, communicating. Mah- wah, N.J.: L. Erlbaum.

Motschnig-Pitrik, R., & Derntl, M. (2005a). Can the Web Improve the Effectiveness of Person-Centered Learning? Case Study on Teaching and Living Web-Engineering. IADIS International Journal of WWW/Internet, 2(1), 49-62.

Nodenot, T. (2007).. Scénarisation pédagogique et modèles conceptuels d'un EIAH : Que peuvent apporter les langages visuels ? Revue Internationale des Technologies en Pédagogie Universitaire(RITPU) / International Journal of Technologies in Higher Education (IJTHE), 2007, 4 (2), pp.85-102.

Paquette, G., Léonard, M., & Lundgren-Cayrol, K. (2008). The MOT+ visual language for knowledge-based instructional design. Handbook on Virtual Instructional Design Languages.

Parrish, P. (2008). Plotting a Learning Experience. In L. Botturi & T. Stubbs (Eds.), Handbook of Visual Languages for Instructional Design: Theories and Practices (pp. 91-111). Hershey, PA: Information Science Reference.

Rawlings, A., van Rosmalen, P., Koper, R., Rodríguez Artacho, M. and Lefrere, P. (2002). CEN/ISSS WS/LT Learning Technologies Workshop: Survey of Educational Modelling Languages (EMLs).

Rogriguez, M. , Derntl, M. , & Botturi, L. (2010). Visual instructional design languages. Journal of Visual Languages and Computing, 21(6), 311-312.

Stubbs, S.T. and Gibbons, A.S. (2008), The power of design drawing in the other design fields, in L. Botturi and T. Stubbs (eds), Handbook of Visual Languages for Instructional Design. Theories and Practices, Hersey PA: IGI Global.

Tattersall, C., & Koper, R. (2003). EML and IMS Learning Design: from LO to LA Educational Technology Expertise Centre, The Open University of the Netherlands.

Tattersall, C., Manderveld, J., Hummel, H., Sloep, P., Koper, R., & de Vries, F. (2003). IMS Learning Design Frequently Asked Questions: The Open University of The Netherlands.