Applying Model Driven Engineering Techniques and Tools to the Planets Game Learning Scenario

Nodenot Thierry [1], Caron Pierre-André [2], Le Pallec Xavier [2], Laforcade Pierre [3]

[1] Laboratoire LIUPPA
IUT de Bayonne,
2, allée du parc Montaury
64600 Anglet
FRANCE
http://liuppa.univ-pau.fr

[2] Laboratoire LIFL
Bâtiment M3,
59655 Villeneuve d'Ascq Cédex
FRANCE
http://www.lifl.fr

[3] Laboratoire LIUM
52 Rue des Dr Calmette et
Guérin, 53020 Laval Cedex 9,
FRANCE
http://www-lium.univ-lemans.fr/

Abstract: CPM (Cooperative Problem-Based learning Metamodel) is a visual language for the instructional design of Problem-Based Learning (PBL) situations. This language is a UML profile implemented on top of the Objecteering UML Case tool. In this article, we first present the way we used CPM language to bring about the pedagogical transposition of the planets game learning scenario. Then, we propose some related works conducted to improve CPM usability: on the one hand, we outline a MOF solution and an Eclipse GMF solution instead of the UML profile approach. On the other hand, we propose some explanations about transforming CPM models into LMS compliant data, and tool functionality.

Keywords: Visual Instructional Design Languages, Learning Management Systems (LMS), Metamodeling techniques and tools, UML profiles, Meta-Object Facility (MOF), Graphical Modeling Framework (GMF), Model transformation.

1 Introduction

CPM means **C**ooperative **P**roblem-based learning **M**etamodel. It is a visual design language that we developed at the LIUPPA Laboratory (France) as a specialisation of UML language. CPM language focuses on the design of Problem-Based Learning (PBL) situations. According to the Instructional Design Classification Scheme defined in Botturi, Derntl, Boot and Gigl (2006), it is a visual (notation level), layered (stratification level), semi-formal (formalisation level) language promoting multiple perspectives (more than one view) upon the same entities.

On the one hand, *educators and designers* use CPM language to draw models (similar to UML sketches) focusing on initial requirements of a PBL situation. These include the PBL domain, situated roles of learners/teachers, learners' skills, predicted obstacles which the educators want learners to overcome, goals and criteria for success within the PBL situation, resources available to learners, *etc.*

On the other hand, CPM language addresses *instructional engineers*. Their work involves designing a viable solution, coordinating all the actors involved in the development team. UML is a prerequisite for the engineers who use CPM language to draw various models which capture different points of view or outlooks on the same PBL situation (pedagogical, structural, social, operational viewpoints). This set of models makes up the learning/tutoring scenario which can be planned (in terms of steps and learning/tutoring events) but cannot be totally predetermined during the design phase since PBL addresses generative learning Allert (2005). The blueprints they produce are expressed in terms of concepts appearing in sketches produced by educators, thus facilitating discussion and agreement.

CPM sketches and blueprints prepare the detailed design stage that involves mapping those agreed CPM models with Platform-Independent Models (PIM) e.g. IMS-LD (Laforcade, 2004) or Learning Management System (LMS) abstractions (Caron, Le Pallec and Sockeel, 2006). The idea consists of mapping conceptual design models with components representing abstract views of the services provided by an LMS: such mapping leads designers to use CPM to specialize and contextualize the services supplied by an LMS according to the specificities of the activities to be fulfilled.

This paper presents some models that were produced with CPM. These models refer to the Planets game case-study proposed by Vignollet, David, Ferraris, Martel and Lejeune (2006). Assisted by a primary teacher, we used CPM to describe the conceptualisation level that 12 year-old pupils can reach whilst selecting different scientific properties of the planets: their distances from the sun, their day durations, their year durations, their compositions, their average temperatures, *etc.* This domain study led us to set more detailed learning/tutoring objectives from which we defined a learning scenario and tutoring strategies.

The article is structured as follows: in section 2, we present the way we used CPM to transpose didactically the planets game learning scenario. Section 3 presents related works conducted to improve CPM usability: on the one hand, considering the difficulties that educational designers encounter while producing CPM models with the Objecteering UML Case tool, we outline a MOF solution (section 3.1) and an Eclipse GMF solution (section 3.2) instead of the UML profile approach. On the other hand (section 3.3), we propose explanations for transforming CPM models into LMS compliant data and tool functionality.

2 Pedagogical transposition of the Planets game with CPM language

As presented in the introduction, CPM is well suited to enable designers to elicit a learning scenario and question it against a/ the knowledge to be taught and b/ the learning outcomes. Since the case study's outline gives few results about these topics, CPM could be used to represent what the case study's outline tells us (cf. Figure 1):
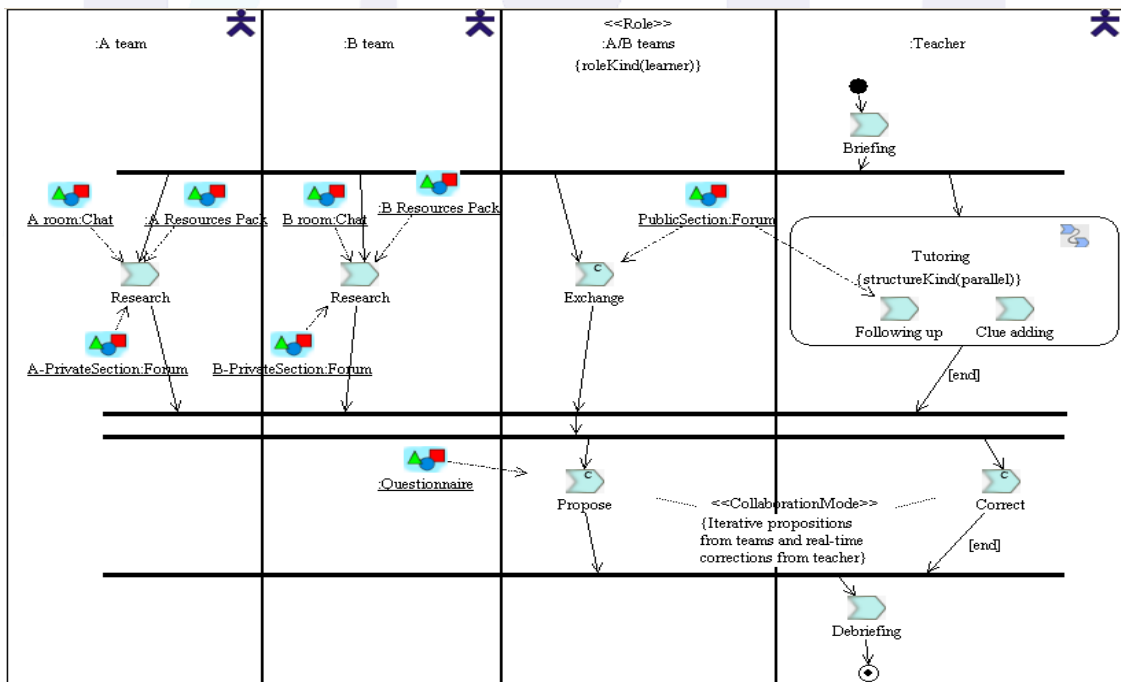

Figure 1: An activity diagram representing identified roles and activities

In this figure, four swim lanes are used to visually represent the specific activities performed by identified roles. This figure demonstrates the expressive power of CPM (Resources used and produced, individual and collaborative activities) whose stereotypes can enrich the semantics of any UML diagram. However, this activity diagram is poorly founded pedagogically because nothing is said about the knowledge that A/B teams will discover in their Resource Pack, nothing is said about their Forum functionality, nothing is said about the kind of tutoring that will be provided to learners, etc.

2.1 The knowledge to be taught

As a consequence, we had to go further in our analysis of the Planets Game. We decided to focus on the conceptualisation level that 12 year-old pupils can reach; first we tried to define the information that should be embedded in the interviews (resources pack) provided to both teams. Such information concerns the following planet properties: their names, their distance from the sun, their day length, their year length, their composition (solid / gas), their average temperature.

Name	Distance from sun (million kms)	length of day	length of year (Earth year/day)	composition	Temperature (K)	Diameter (km)
Mercury	58	59 days	88 days	solid	100-700 (mean 452)	4878
Venus	108	244 years	225 days	solid	726	12104
Earth	150	24 hours	365 days	solid	260-310	12756
Mars	228	25 hours	687 days	solid	150-310	6787
Jupiter	778	10 hours	12 years	gas	120	142796
Saturn	1427	10 hours	29 years	gas	88	120660
Uranus	2872	17 hours	84 years	gas	59	51118
Neptune	4509	16 hours	165 years	gas	48	48600
Pluto	5916	6 days	248 years	solid	37	2274

Table 1: The planets focused properties and property values

From this table, we decided that the knowledge to be learned by the A and B teams from their interview analyses should include:

- solid (vs. gaseous) planets are near (vs. far) from the sun except Pluto: this is the classification of the nine planets based on the criteria composition / distance,

- small (vs. giant) planets are near (vs. far) from the sun except Pluto: this is the classification of the nine planets based on the criteria diameter / distance,

- average temperature ranking of the nine planets is determined by their distance from the sun, except for Venus

- length of year ranking of the nine planets is determined by their distance from the sun,

- …

CPM enabled us to formalise such information about the knowledge to be taught: for example, Figure 2 is a representation of the planets classification on the criteria composition / distance while Figure 3 represents the correlation between the average temperature and the distance from the sun.
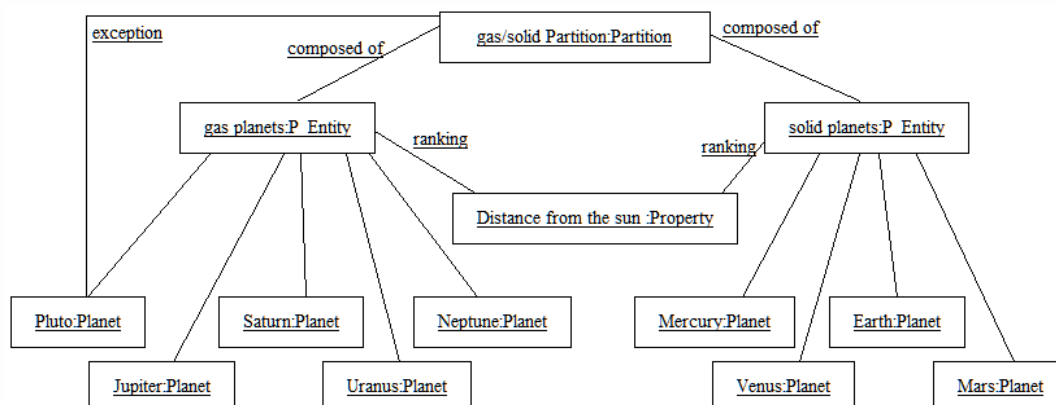

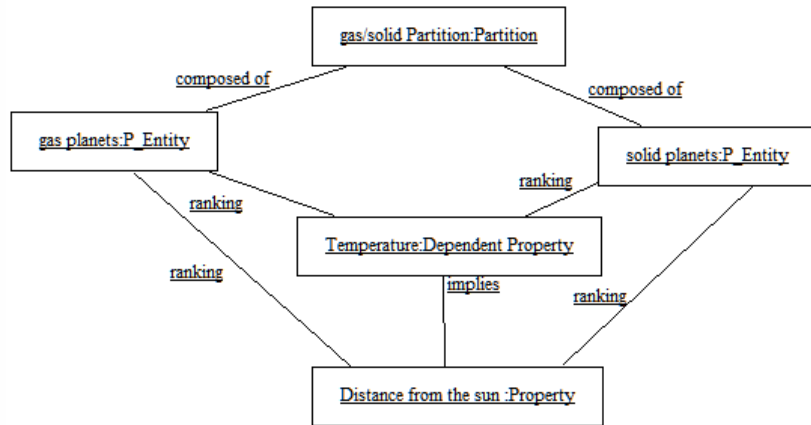Figure 2: A classification for the composition / distance criteria

Figure 3: Correlating distance and temperature properties.

Both figures conform to the following class diagram that we also formalised with CPM language. In Figure 4, concepts with the **Subj** stereotype are learning subjects, that is to say that the pupils will have to determine/investigate them from the interviews at their disposal. The figure also provides information about the contents of the different expert interviews (cf. the Resource stereotype with the ▲■ icon). All of them talk about specific criteria which are either properties (diameter, day length, composition, distance from the sun, etc.) or dependent properties (average temperature, etc.). But the provided constraint enforces to provide information about the values of some planet properties (nothing said about classifications / nothing about Groups of planets).
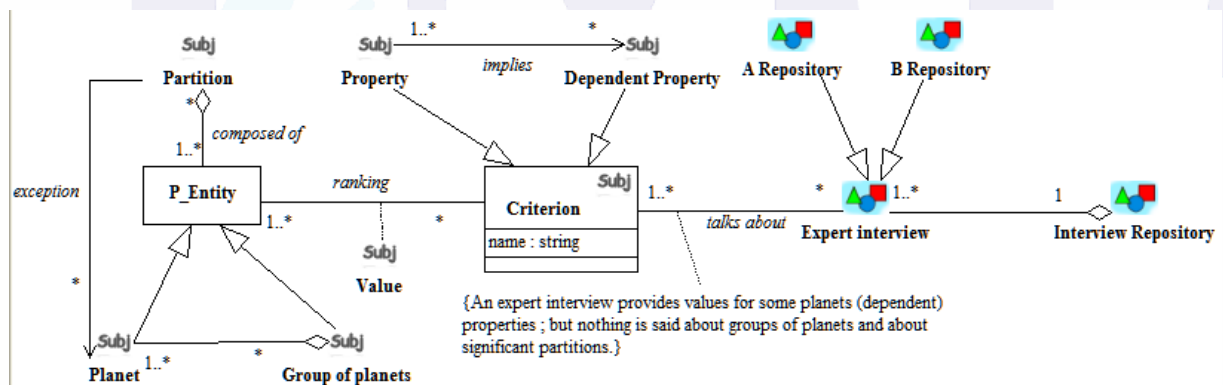

Figure 4: The CPM diagram used to represent the knowledge to be taught in the planets Game

2.2 The expert interviews embedding the knowledge to be taught

It was then quite easy to exploit such a diagram to represent the knowledge embedded within a particular interview. For example, Figure 5 presents an extract of a CPM diagram which states that the repository for A team is composed of different interviews. One of them talks about the particular distances from the sun and average temperatures of Earth, Mars, etc.
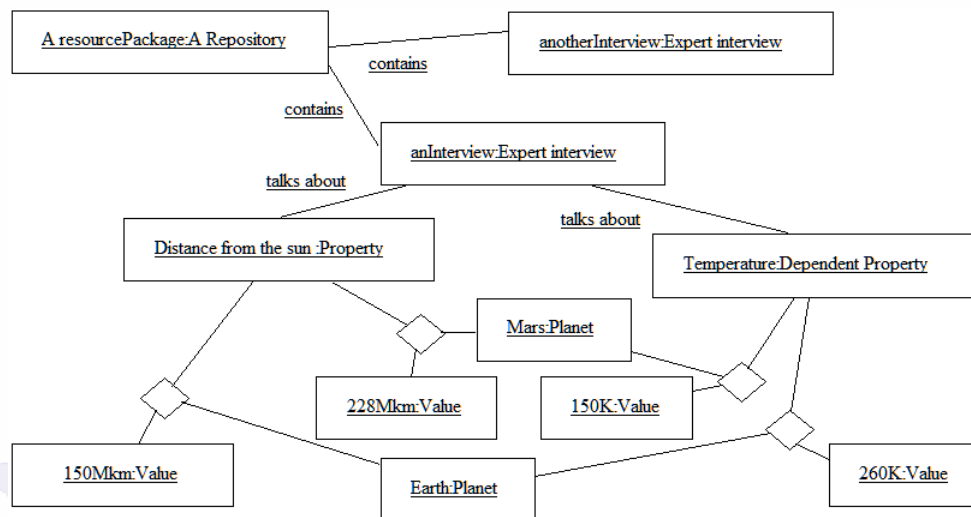
Figure 5: An object diagram describing some contents of A repository

It is therefore the same semi-formal domain language that can be used by designers to describe the knowledge to be taught, the contents of A's and B's repositories, etc. We also used such a DSL (Domain Specific Language) to analyze the activities that the pupils had to do.

2.3 Our didactic choices to help learners reach such knowledge embedded in the expert interviews

Since CPM is well suited to design constructivist learning scenarios, we decided to distribute the available planet information (cf. Table 1) among different expert interviews allocated to A or B. As a consequence, A and B teams had to share out information to reach their particular learning objectives. From its set of interviews, the A team had to (i) find the names of all planets (in the interviews, two of them were called X and Y planets), (ii) discover some [distance / year length /temperature] values, (iii) correlate year length / distance properties, (iv) correlate distance properties / temperature properties, (v) correlate year length / distance properties, (vi) identify giant/solid planets from others, and (vii) formulate exceptions. As regards to the B team, its provided learning goals were to (i) discover some [distance / length of year /temperature] values, (ii) correlate year length / distance properties, (iii) correlate distance properties / temperature properties, (iv) identify giant/solid planets from others, and (v) formulate exceptions. To enable each team self organization, we decided to specialize the forums to let each team document and identify/share its agenda within a workspace.

2.3.1 Workspace modeling

To answer the didactic aims presented above, we drew some CPM diagrams to represent the cooperative functionality provided by the forums. Figure 6 describes the workspaces provided for A and B teams:
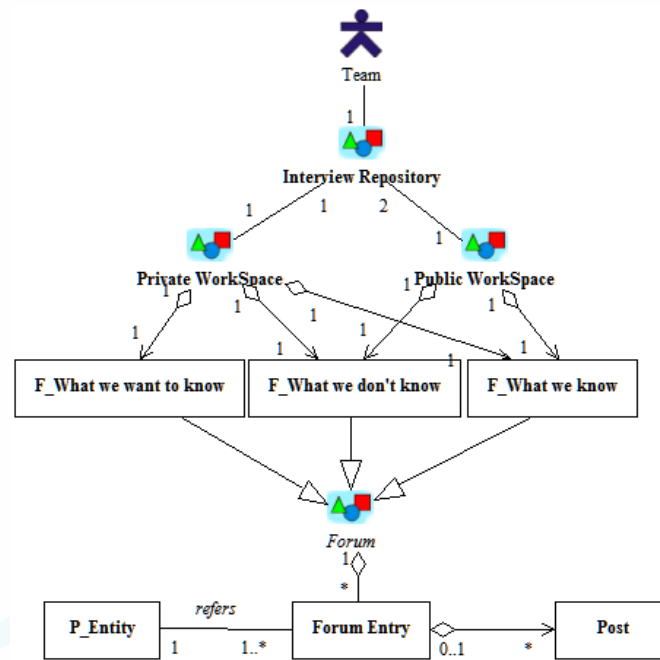
Figure 6: The Workspaces allocated to each team

The CPM diagram states that each team is provided with an interview repository composed of one private workspace and one public workspace (any information put within the public workspace can be read by the other team). Each workspace contains dedicated forums, the "F_What we want to know" forum is only available in the private Workspace since both team are competing (cf. the case study outline). Each forum provides a list of forum entries which are either Planets or groups of planets (cf. P_Entity in Figure 4). Any number of posts can be written by pupils for a given entry.

2.3.2   *Focus on the Planets Game Learning scenario*

To go further in the analysis, we also decided that each team had to assign specific tasks to dedicated roles: a forum manager that will have the rights to add entries and posts, and investigators that will analyze the interviews put at their disposal. Finally, reporting on the game results will be done as a team (specific login needed and specific time to log in). From the tutoring viewpoint, we also decided the different roles of the human teacher: he behaves as a solar system expert and his objective is to lead pupils correlate properties and find exceptions. He is also the group manager who can post messages when needed (to suggest further analysis of an interview for example). The last role is the Timer that has a very important role in our educational game since we want students add posts in the forum at regular intervals.

The following CPM diagram is a use-case diagram that provides the general picture of different use-cases represented by ovals.
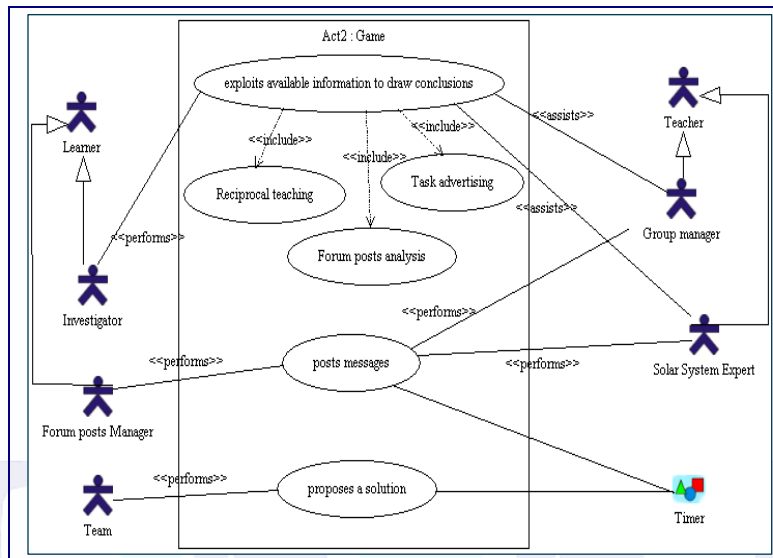
Figure 7: The CPM use-case diagram of the Game

This figure focuses on Act2:Game; this act occurs after the presentation of the game that consists of building two teams and providing them with required functionality to read interviews and game goals. In our learning scenario, Act2:Game is followed by a debriefing session conducted with the whole class and an individual assessment session: Figure 8 is a CPM diagram that describes such a sequencing:
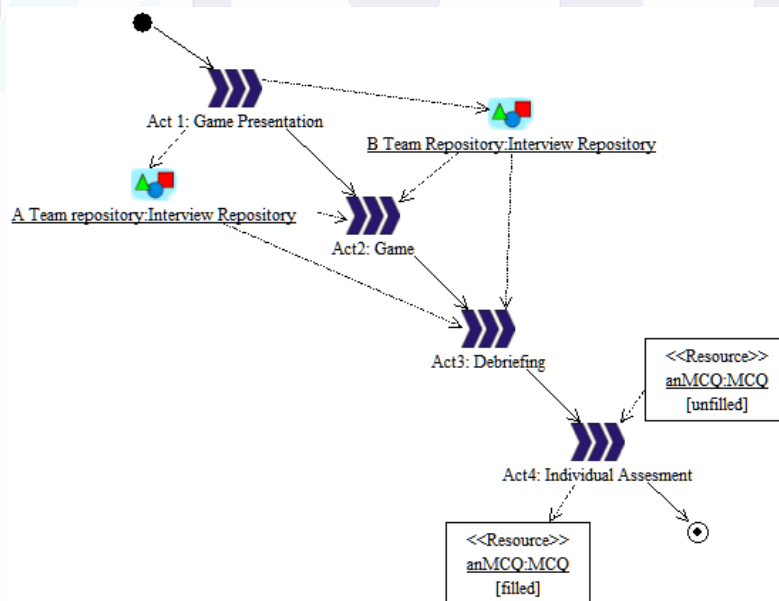

Figure 8: The context in which Act2:Game occurs

Figure 9 details the "posts messages" use-case from Figure 7. Here, this part of the scenario details what happens when the "Forum Posts Manager" is asked by the Timer to post a new message. Within a team, the posts manager and an investigator have to agree (cf. the cooperative activity stereotype) on the information that will be copied from the "F_What we know" forum to the public "Workspace" (and as a consequence made available to the other team). To do that, the Forum Posts Manager" uses tools functionality (cf. the CPL stereotype) made available to him.
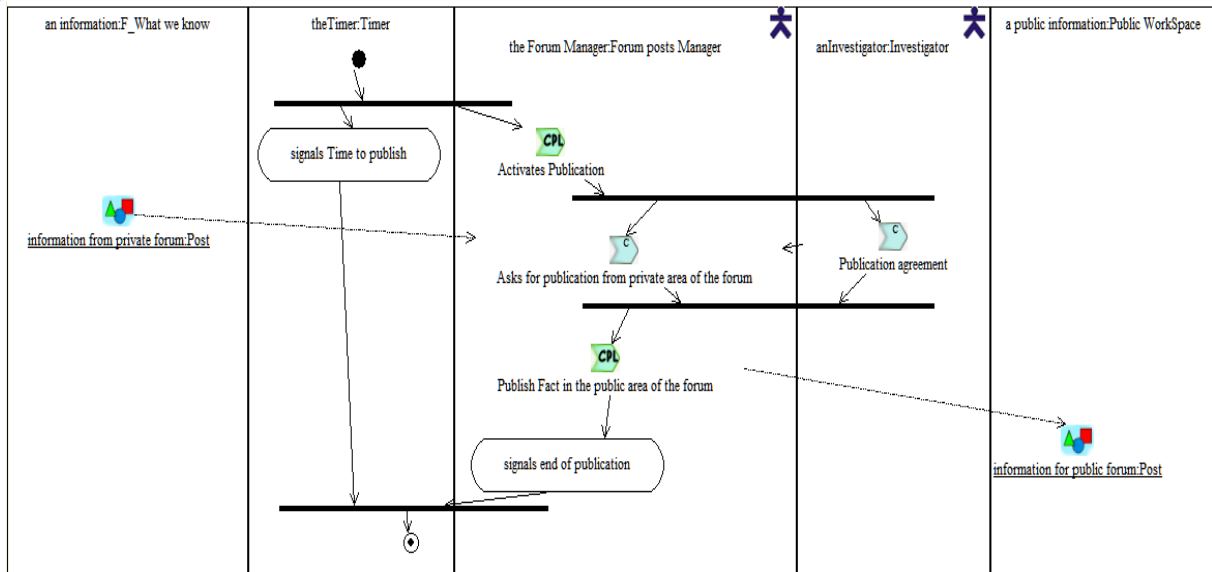
Figure 9: An activity diagram for the "posts messages" use-case of Act2:Game

2.3.3 The CPM Browser

All CPM diagrams (from Figure 2 to Figure 9) are reachable from a unique starting point provided for the designers: the Browser.
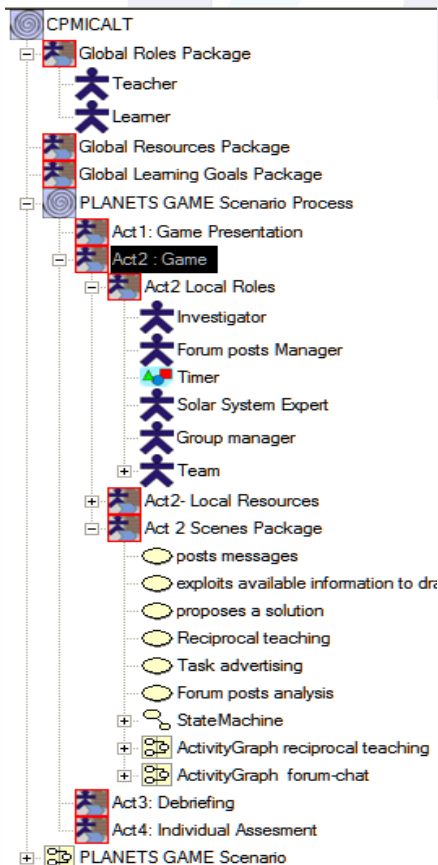

Figure 10 is a snapshot of the Browser which enables a designer to edit the Planets game Learning Scenario. At root level, experience led us to create three Learning Packages whose model elements are exploited by the *Learning Process* Package called the *Planets Game Scenario Process.* At the bottom of the figure, worth noting is the *Planets Game Scenario* denoted as an activity diagram used to generally describe how the different acts of the *Planets Game Learning Process* are sequenced.

The model elements (and graphical views) of these four acts are then detailed within the *Planets Game Scenario Process*.

In the snapshot, the details of the *Act2 Process* were expanded. At this level, note that the package structure is the same as the one at root level: Act2 shows a *Local Roles Package*, a *Local resources Package*, a *Local Learning Roles Package*, and an *Act2 Scenes Package* which contains all the scenes within Act2. This structuring promotes the contextualisation of roles, learning goals, resources and learning activities. For example, the expanded *Act2 – Local Roles Package* shows different *Actor* stereotypes, which are model elements used during Act2 to specialize the tutor role and the Learner role (*cf.* the *Global Roles Package*).

Figure 10: The Planets Game browser

3   Related works to enhance CPM usability

UML and its profile mechanism propose a framework which may be quickly efficient. First it provides several types of diagrams which allow describing many aspects. Second, several design processes have emerged from UML community over the last decade. They describe best-practices related to navigation between previous types of diagram. Nevertheless, there are some weaknesses. First, defining new modeling concepts with UML profiles requires the use of (through inheritance) existing UML concepts/meta-class like Class or Actor. Inheriting from core UML concepts is also a way to reuse graphical facilities of diagrams. A UML profile designer does not necessarily want all inherited attributes or methods. She/he has to block access to these undesirable properties in conceptual AND graphic ways (respectively through OCL constraints and J code with Objecteering) to respect the semantics of the language she/he wants to design. This is a complex and tedious process if we consider the definition of graphical languages for complex, condensed and non-software engineering metamodels (like CPM or IMS-LD). In addition, an efficient profile (that is to say, with conceptual and graphic filtered accesses) generally works only with the UML tool used to define it.

Finally, it is currently difficult to propose a free stand-alone model editor for practitioners with UML tools: efficient tools are generally expensive; using profiles means using the entire environment; environments are often software engineering oriented which is not a quality for practitioners.

So it is important to explore alternatives like OMG-MOF (OMG 2007) or Eclipse/EMF (EMF 2008) environments. Based on a meta-modeling approach, they present some advantages. For example, defining a language first starts with the definition of a metamodel (abstract syntax) which is not created from existing concepts but from scratch. So there is no need to filter access to model elements because of undesirable inherited features. Furthermore, there are currently powerful graphic tools such as GMF (GMF 2008), which allow the definition of efficient graphical syntax for a language (concrete syntax). There are other facilities which are not so efficient in UML community like model transformation engines (GMT for Eclipse (GMT 2008), ATL for MOF-based models (see Patrascoiu, 2004) and code generation engines like JET (JET 2008)). Another useful functionality of EML is that creating a metamodel may be done just by analysing a XML schema or a DTD. We present in this part two technological methods with which we can create a CPM authoring tool: ModX and GMF.

3.1   ModX

ModX is a graphic tool used to create both model and MOF-based metamodel. It supports XMI format (import/export) in order to be compatible with other MDA tools. Use of ModX may be summarized through the following steps:

- Load an existing metamodel or create a new one,
- Define the graphic notation associated to the metamodel,
- Create models from the metamodel,
- Change the metamodel if necessary (associated models will change accordingly).

ModX should be used to graphically define models, where there is no graphical editor for the underlying metamodel, to design a new metamodel or to improve an existing one, to manipulate models and to remain compliant with MDA standards (MOF, XMI, UML) or to grasp and to master a complex metamodel provided in XMI format by experimenting with it.

3.1.1   Main principles

Figure 11 describes activities that one can do with ModX and their relations.
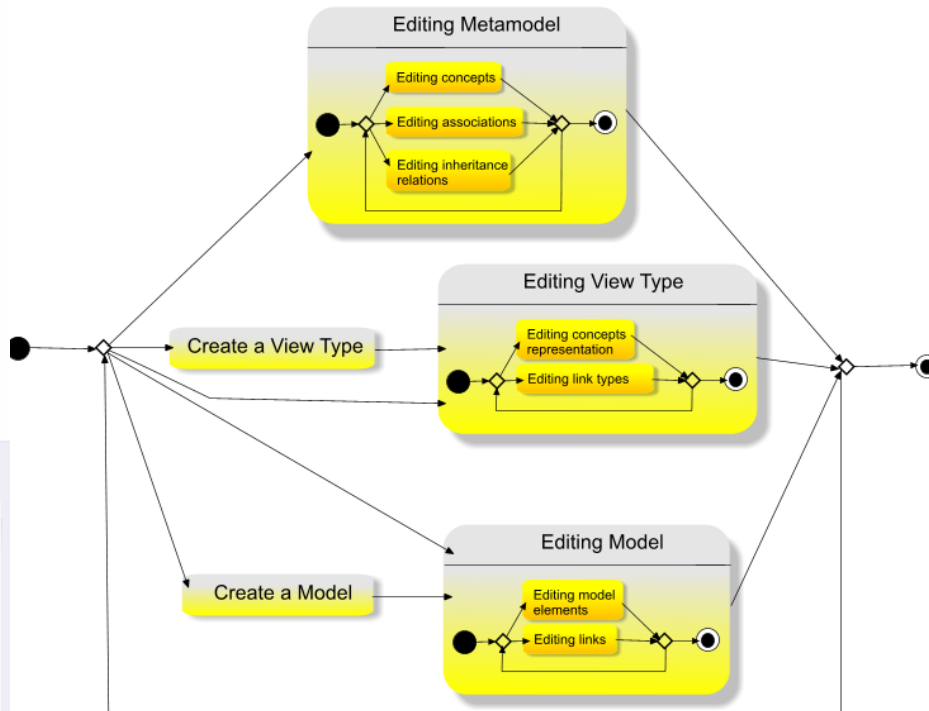
Figure 11: ModX Activities

Reflection is the main feature of ModX: it is possible to change the underlying metamodel or one of its graphic formalisms (called "View Type") during editing a related model. A ModX metamodel is MOF 1.4 compliant and it is composed of classes and associations (and inheritance links).

A metamodel may have several view types. With ModX, a view type (illustrated in figure 12 and figure 13) corresponds to a concrete syntax (MDE term) while a ModX metamodel corresponds to an abstract syntax. A graphic notation may be more appropriate here. A view type specifies whether instances of a concept will be represented by an icon or a graphic shape. It also specifies whether instances of an association will be represented by simple line, juxtaposition or the fact that one of the two ends will be nested in the other one.

Figure 12: Graphical representation of roles in CPM Activity Diagram

Figure 13: Graphical representation of links between actors (and so roles) and activities in CPM Activity Diagram

A model is made of model elements and links. It is represented through one or several views. Each view respects its assigned view type. Figure 14 shows the use case diagram for the planet game using CPM metamodel and notations.
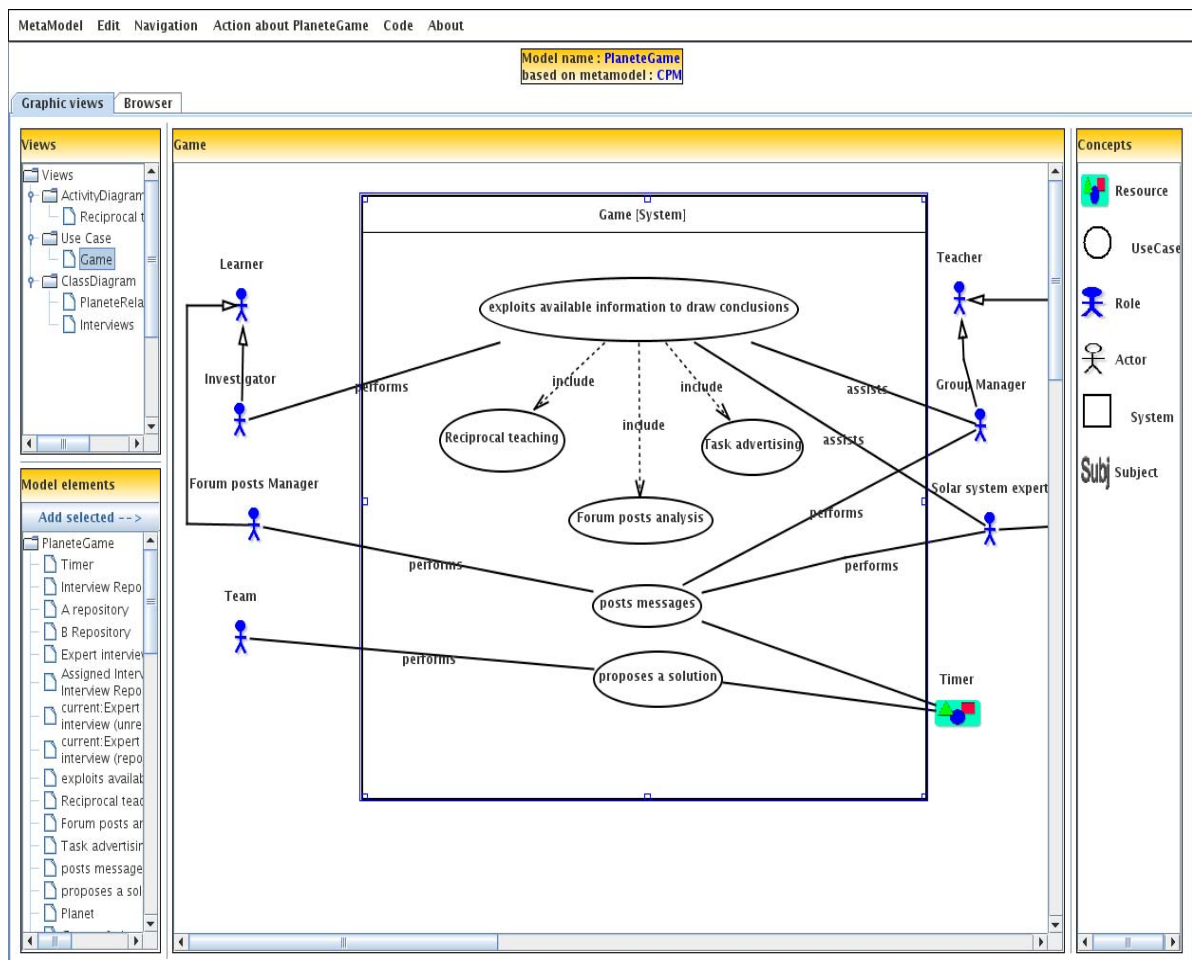
Figure 14: Planets game on ModX: a CPM use case diagram

Models can be "instantiated" from a metamodel at any time, even if the metamodel has only one concept: models dynamically change according to modifications of related metamodel and associated view types.

3.1.2   CPM with ModX

If we want to associate three different types of diagrams for CPM metamodel, we need to define three view types in ModX.

We used Modx graphic abilities to define the view types which conform to the UML representations of CPM version of the case study (cf. figure 15).For example, in the view type related to use case diagram, an actor is represented by an icon and all links in which an actor may occur are simple lines or arrows. On the other hand, within an activity diagram, an actor appears as a rectangle which may contain activities: the Activity end of the association plays is graphically noted « isNested ».

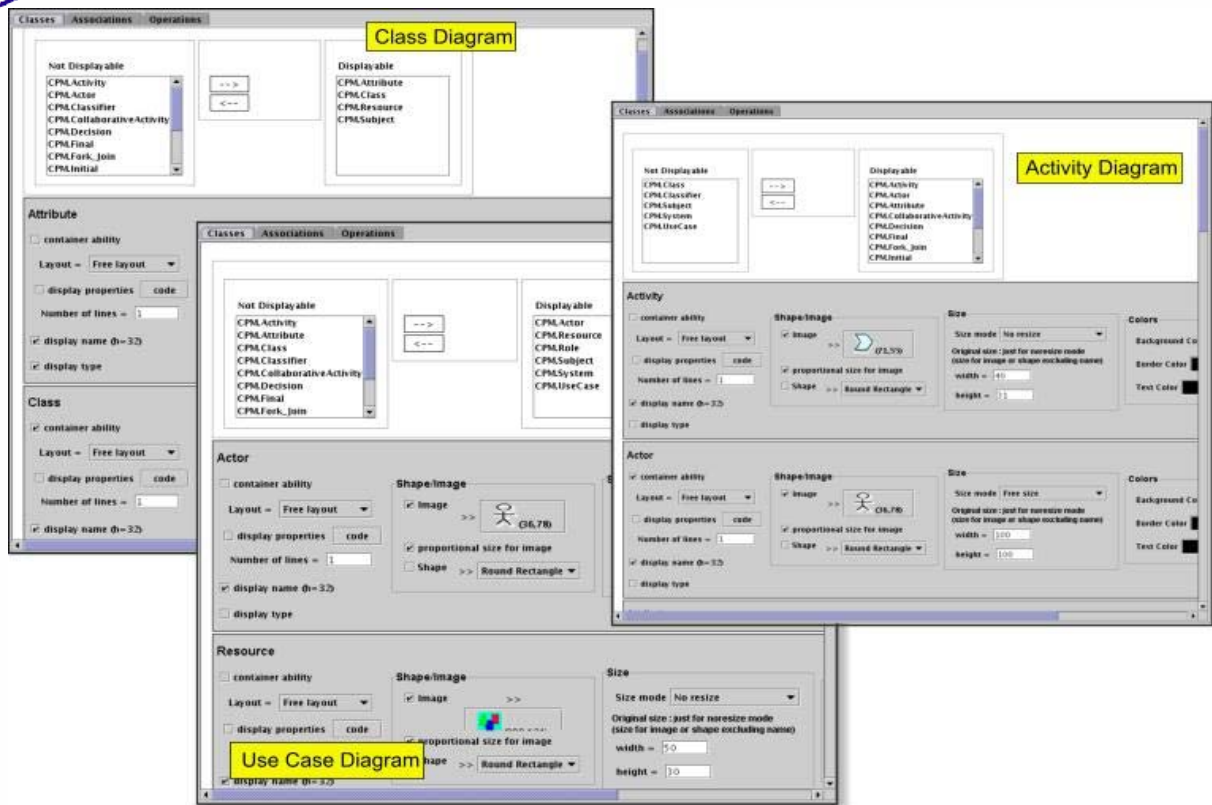Figure 15: View types in ModX

Thanks to the previous metamodel and the three associated view types, ModX provides a basic CPM model editor. We have duplicated original diagrams for the planets game with ModX (cf. figure 16).
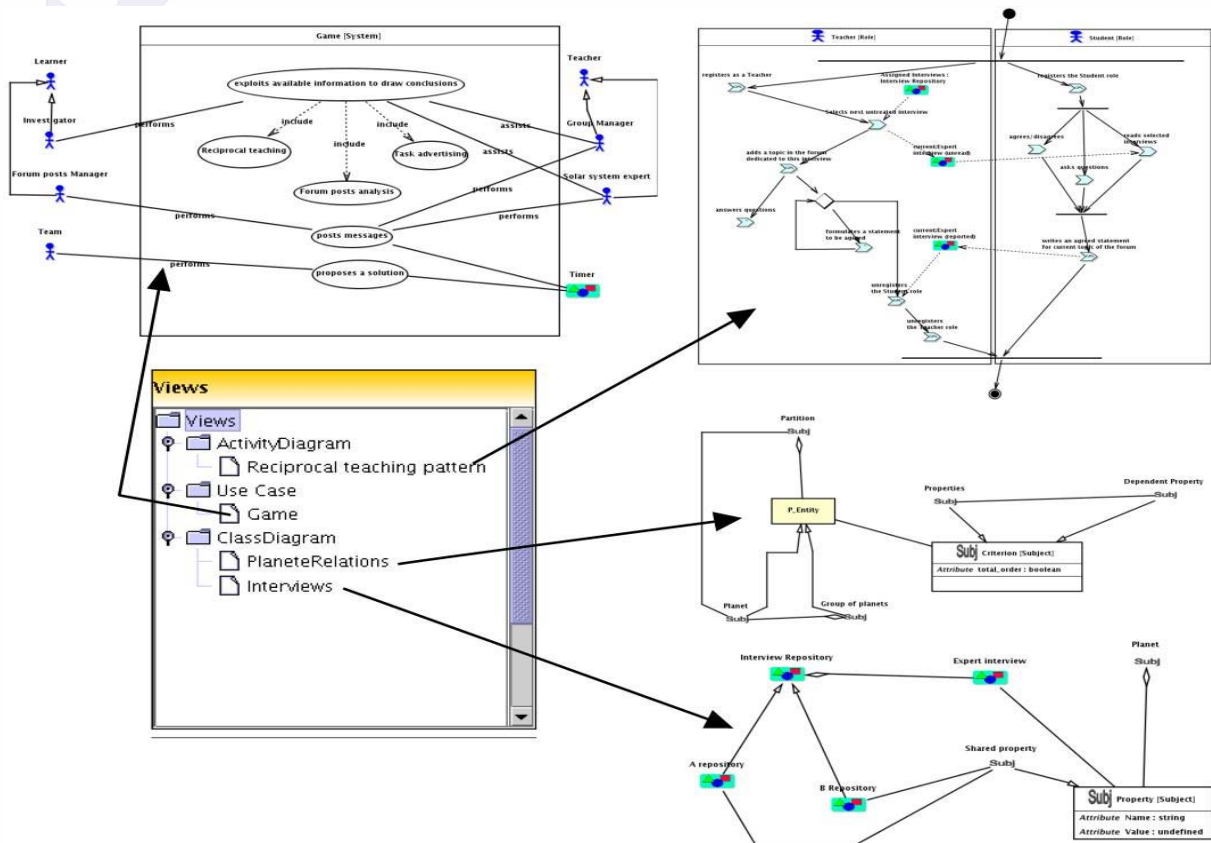

Figure 16: Diagrams for the planets game realized with ModX.

3.2   Use of Domain-Specific Modeling tools from the EMF project

The other approach that we outline here goes one step beyond the UML-based approach (CPM) and the MDE approach of ModX. It concretely deals with the application of *Domain-Specific Modeling* (DSM) theories and practices (see Kelly and Tolvanen, 2008) for instructional design.

3.2.1   Domain-Specific Modeling

DSM is a part of the MDE domain. It is a software engineering methodology for designing and developing systems. It involves the systematic use of a graphic *Domain-Specific Language* (DSL) to represent the various facets of a system, also called Domain-Specific *Modeling* Languages (DSML). Several technical approaches coexist currently to support the specification of DSML (see Jouault, Bézivin, Consel, Kurtev and Latry, 2006): commercial products like *MetaCase/MetaEdit+,* the Microsoft DSL tools (based on the *Software Factories* vision), and academic propositions or open-source projects like VMTS, TIGER, EMF, GEF, GMF, etc. All these DSM tools propose metamodeling techniques capable of expressing domain-specific vocabularies (abstract syntaxes), and propose facilities to construct various notations (concrete syntaxes). These editing frameworks offer techniques for multiple customizations with minimal programming. As a result, these tools can generate powerful and user-friendly dedicated editors for DSM languages. They are kind of meta-CASE editors capable of generating CASE tools. The final editors give domain-designers the ability to graphically specify models from their domain, and propose some persistence facilities to load and store these models in a machine-interpretable format. This machine-directed format is always independent from the notation used to visually represent the model.

As the ModX tool, DSM tools are *meta*-tools that can allow specific editors to be used for instructional purposes. The DSM approach can help the emergence of communities of interests or practices sharing the same domain-vocabulary and formalisms. Some DSM tools can be used in this instructional design context to support the emergence of Visual Instructional Design Languages (VIDL), with their specific editors. We can imagine various uses of this new approach: specific editors focusing on teacher-designer practices (for example in relation to some pedagogical theories, didactic fields as well as specific references to the LMS they use; a CPM editor could be used as well), other editors specific to EML standards (like an visual editor providing a graphic notation for the IMS-LD standard), and other editors dedicated to specific platforms (like a visual editor for Moodle, Ganesha, etc.). Such a DSM approach can cover all the three steps of an instructional design process: conception, specification and implementation activities. Also, other DSM tools focusing on model transformations (e.g. the Eclipse ATL tool, described in Allilaire, Bézivin, Jouault and Kurtev (2006)) can be used to support the transformation of instructional scenarios from VIDL to another.

We are in the process of experimenting with the *Graphical Modeling Framework* (GMF) to support the DSM approach for learning scenarios. We propose to detail an experiment about a UseCase-centric VIDL and editor for the Planets game in the next sub-section.

3.2.2   The Planets game case study: a new use case-based representation

We choose to focus also on one CPM diagram detailed in section 2.3.2: the UseCase-based view of the main activities and roles involved in the Planets Game learning scenario. This previously depicted diagram illustrates 'Act 2'. The notation is UML for this kind of diagram: actors, usecases, one system, and various relations (inheritance between actors, communication links between actors and usecases, includes and extends relations between usescases, etc.). The CPM notation for this diagram is limited to the addition of stereotypes for the UML meta-concepts and meta-relations involved into UseCase diagrams: actors have been extended to represent « roles » and a dedicated icon has been provided to illustrate such stereotype application. Also, the expressiveness of this CPM diagram is limited to the addition of semantics to show concepts and relations via the stereotype whose semantics is defined in natural language.

Since UML UseCase diagrams are not made for expressing time-related relationships between usecases, it is not possible to link main learning activities, derived from the usecases, between each other to specify a precedence/following relationship. Our experimental work will consist in providing

a pluridisciplinary learning design team with a dedicated visual editor able to express such scenario representation. Also, we aim to provide them with a specific VIDL guaranteeing that the produced models will be both human-readable and machine-interpretable.

3.2.3   EMF/GMF framework presentation

The Eclipse *Graphical Modeling Framework* (GMF) (GMF 2008) provides a generative component and runtime infrastructure for developing graphic editors based on EMF (*Eclipse Modeling Framework*) and GEF (*Graphical Editing Framework*).
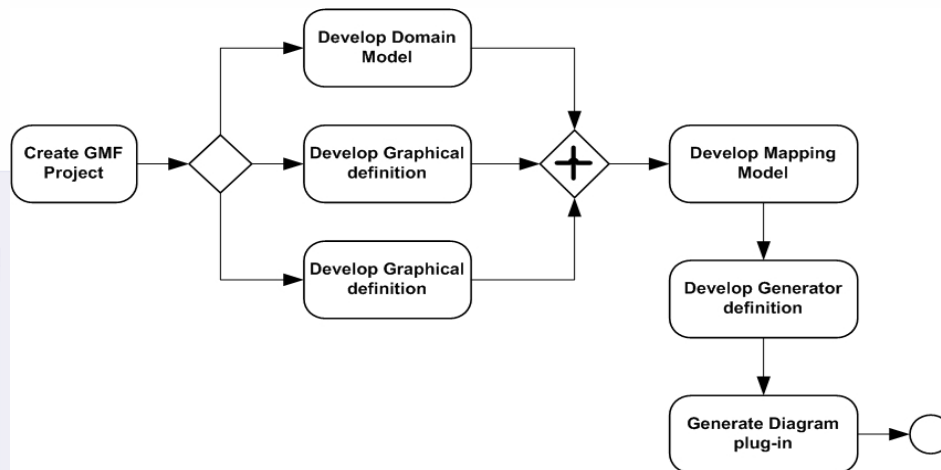

Figure 17: The GMF process

The above figure is a diagram illustrating the main components and models used during GMF-based development. One of the GMF core concepts is the graphic definition model. This model contains information related to the graphic elements that will appear in the future diagrams produced by practitioners. These graphic elements have no direct connection to the domain models for which they will provide representation and editing. Also, a tooling definition model is used to design the palette and other periphery (menus, toolbars, etc.). GMF allows the graphic definition to be reused for several domains by using a separate mapping model to link the graphic and tooling definitions to the selected domain model(s).

Once the appropriate mappings are defined, GMF provides a generator model to allow implementation details to be defined for the generation phase. The production of an editor plug-in based on the generator model will target a final model; that is, the diagram runtime (or "notation") model. The runtime will bridge the notation and domain model(s) when a user is working with a diagram, and also provides services for the persistence (XMI as well as XML) and synchronization of both. The final generated editor can also be provided as a standalone RCP (*Rich Client Platform*) application (independent from Eclipse).

3.2.4   Rationale for using GMF to develop a CPM editor

A basic **domain model** for our « Learning Design Use Case » view has been defined. It is illustrated in the following figure (it is a diagrammatical view of the concrete domain model whose native format is XML). We can observe the domain concepts and relations that are not related to UML meta-concepts but are fully related to the domain we want to specify. We keep references to the UML constructs we decided to maintain on purpose: « includes » and « extends » relationships between « HighLevelActivities ». The inheritance link between « actors » has also been maintained. However, we add a new relationship between activities so as to be able to express precedence/following relations between activities.
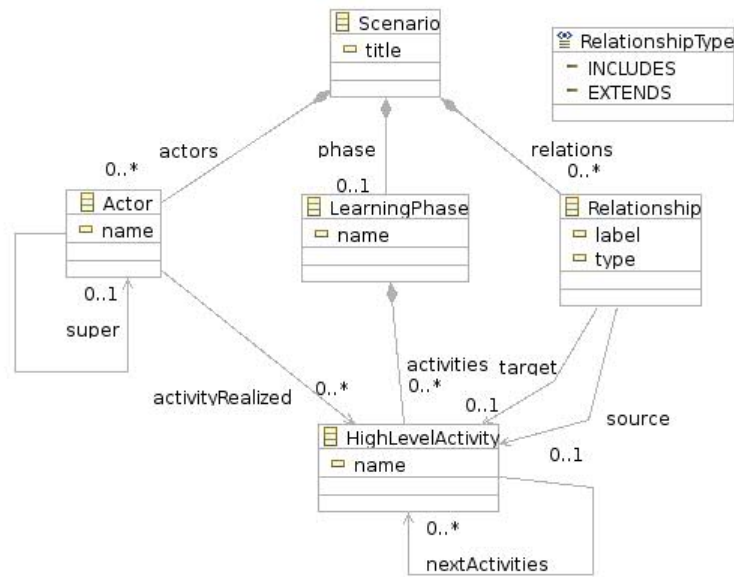
Figure 18: The domain model

A **graphic definition model** has been designed; it defines the figures, nodes, links, etc. that we want to draw in the final diagrams. An extract of this model is presented in the following figure (depicted here by the user-friendly tree-view proposed by GMF). If you examine this model, you will find reference to a *Canvas* at the root with a Figure gallery containing basic *Rectangle, Label,* and *Polyline Connection* elements. These are used by corresponding *Node, Diagram Label,* and *Connection* elements to represent our Actors, LearningPhase, etc. from the previous domain model.
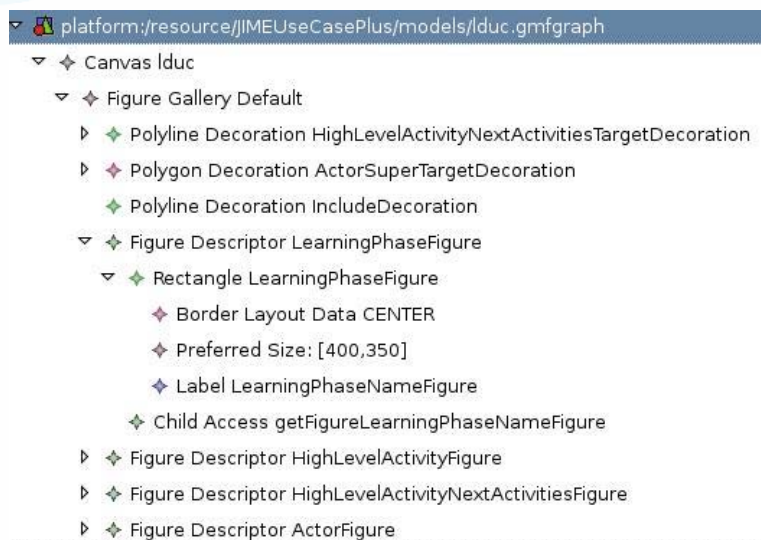
Figure 19: The graphic definition model

We then designed the **tooling definition model** which is used to specify the palette, creation tools, actions, etc. for our graphic elements. Next, the **mapping definition model** will let us bind the three previous models: the domain, the graphic definition, and the tooling definition. The following figure is an image of this model (tree-view). The mapping definition is a key model in GMF development because it will be used as an input to a transformation step which will produce the final code.

Figure 20: The mapping definition model

3.2.5 Back to the Planets Game example

Thanks to the editor generated by GMF we can now use a specific and visual editor that proposes us to draw graphically scenarios conformed to both the domain and the notation we discussed before. The next figure sh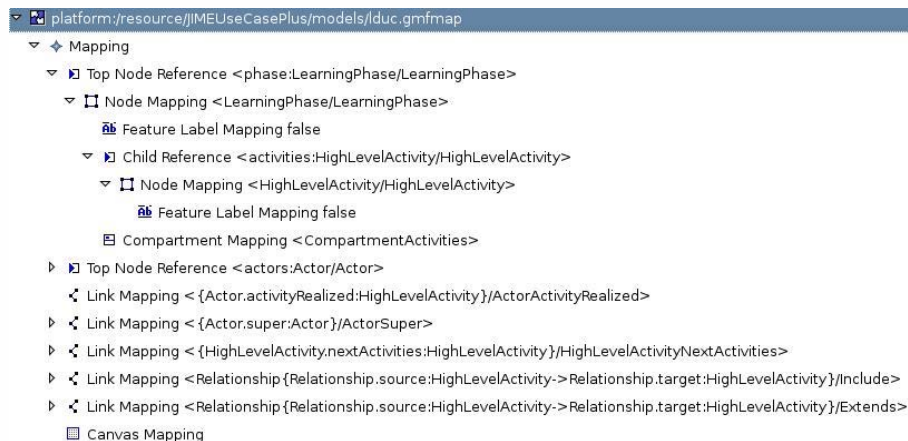ows an extract of the new version for the « Act2 » of the Planets Game. In this new view, we can notice the new relation between activities to express the precedence relationship.
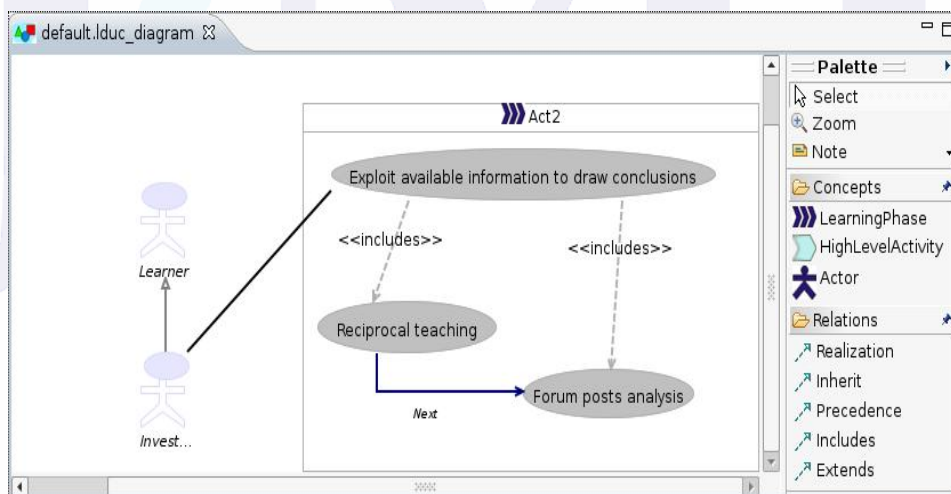
Figure 21: Example of model designed with a GMF-based editor

It is important to not reduce the scenario to the visual representation depicted into the previous figure. This is just a human-readable « view » of the model that we concretely designed. Indeed, the scenario is serialized into a machine-interpretable format (cf. figure 22). Such a concrete model can be edited with a simple text editor (in this case, we customized the GMF based editor to serialize models as XMI resources but a XML version is available if we prefer a serialization conformed to a given XML schema).

Figure 22: The produced model in its computer-readable view

The previous XMI serialization of the produced scenario is based on the domain model we designed: direct references to concepts and relations are made. This example emphasizes the added-value of having XMI serialization of domain models rather than XMI serialization showing some references to UML constructs. It is also important to notice that the EMF framework (used with GMF) automatically generates code in order to load and store such serialized resources.

3.2.6   Synthesis

In this subsection, we presented ongoing work on porting CPM language to the Eclipse Community. For the time being, such effort provides no added-value to instructional designers and pedagogues since the more visible result is a GMF-based editor that enables these actors producing CPM like diagrams. However on the backstage, the EMF specification of the produced learning scenario can then be exploited by any EMF-compatible tool, particularly by Model Driven Engineering tools to transform such a specification into executable code. Moreover, CPM usability could also be improved from such an approach since it enables engineers to more easily develop dedicated editors fitted to the users' needs (we recall that our efforts –cf. section 2- to specialize the Objecteering UML case tool for CPM language did not completely succeed because specialization was limited by the UML profile mechanism).

Yet, such an approach can only fit experimented teams with high-level technological skills. Diving into Eclipse EMF techniques and tools is not straightforward. Producing GMF editors, transforming EMF specifications into code is anything but simple and facing such complexity could bring instructional engineers round to forget that the emphasis must remain on the educational added-value of such editors and tools.

3.3   Operationalization through the Bricoles project

The Bricoles project proposed by Caron (2007) aims to provide, for small scale teaching teams who could not benefit from the help an Instructional Designer, an infrastructure to construct pre-structuring device on e-learning platforms. We speculate that pre-structuring devices are learning objects that are sufficiently weak to be handled within the frame of controlled teaching improvisation. We believe that such objects can be modeled, manipulated, constructed and technically implemented in e-learning Platforms.

3.3.1   The conceptual framework

The conceptual framework of our proposition is made up of two processes. **The first process** aims to create a workshop adapted to a specific type of pedagogical concern. The stakeholders of this process are: the teacher, the instructional designer and the computer engineer.
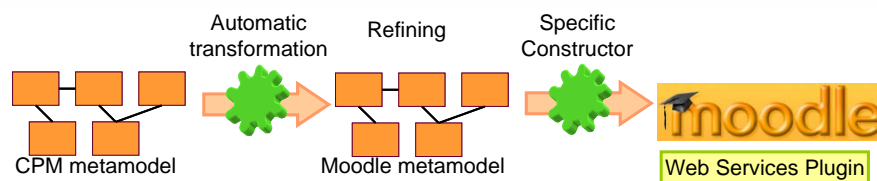
Figure 23: First process - pedagogical concerns, transformation rules, and web service plugin.

- The first step of this process is the description of types of pedagogical concerns by the teacher and the instructional designer. It is possible to perform this work within the CPM metamodel.

- The second step is the definition of the transformation rules for the CPM metamodel to Moodle metamodel by all the stakeholders. We take it for granted that the Moodle metamodel and the Web service plugin, needed to communicate with Moodle, already exist (Hoogstoel and Caron, 2008). Otherwise, they are produced by both the computer engineer (s) and the instructional designer(s). This work and the implementation of the specific constructor are detailed in next section.

The goal of the second process is to carry out the pedagogical aims of a teacher. Its principle is simple:

- First the teacher defines his didactic purpose in a CPM model with graphic language.

- Second he transcribes it into a Moodle model.

- Third he uses Specific Constructor to deploy this model on the moodle platform via the Web Service plugin.

3.3.2   The software support

The software support for the conceptual framework is made up of two tools developed by the LIFL laboratory: **ModX** proposed by Le Pallec (2001) and used for the definition of metamodels and the creation of models (cf. section 3.1), and **GenDep** proposed by Caron (2007) for implementation purposes. **GenDep** - Generic Deployer – has been developed for the Bricoles project. For a given source model, it communicates with the e-learning platform (selected through its web address) to create matching target elements. A plugin must be inserted in each target web-based tool, so that GenDep can interact with it. This plugging enables GenDep to create tangible technological model elements.

3.3.3   Technical approach

We propose to plug a pre-structuring model onto an e-learning platform facilitating its pre structuring use.
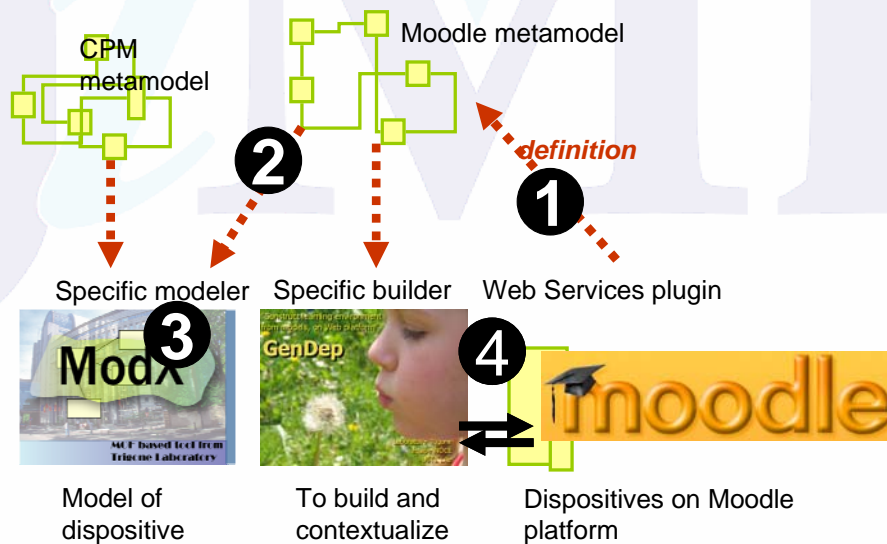
Figure 24: From modeling to deployment

Our approach is founded on the definition of an infrastructure that allows the building of a pre-structuring device on an e-learning platform via the Web Service call. To achieve our goal, we have defined a pluggable web services module for each application. This plugin is inspired by the IMS enterprise standard and is a wrapper for the e-learning platform (Caron, Hoogstoel, Le Pallec and Warin, 2007). From this plugin, we have defined the metamodel of Moodle (1). This metamodel, expressed in Meta Object Facilities language, enables us to parameter a specific modeler of the application, and to build a pre-structuring device constructor (2). Then, it is possible to express a specific pre-structuring device model of the application (3). The engineering process we want to set up is able to structure an application in order to promote its pedagogical use. E-learning platforms single out emerging structuring mechanisms, which is why our infrastructure identifies a context definition phase. This phase allows a dialog between pre-structuring device models and the emerging structures (4).

Building a device on Moodle requires the identification of its construction functions. We propose to wrap these functionalities with services. This solution avoids application code modification. It also provides a standard solution to address an application, which doesn't depend on our solution.

It is possible to define the e-learning platform functionalities metamodel from the expression of the services. (see next section). We propose to use this metamodel for two different purposes:

- The first one consists of using it as a metamodel to get a specific modeling tool for this web application. Thus the models created are compliant to this metamodel.

- The second one implements the service constructor, compatible with the models produced before. Within this constructor, it is possible to define a context for a model before it is constructed.

The choice of Model Driven Engineering (MDE) enables us to define a fast and formal method to generate these two specific tools. In the learning context, the evolution of practices and the lifecycle development of e-learning platforms can bring teachers to redefine the modeling tools frequently. The strength of MDE and our infrastructure is allowing these changes to happen dynamically. The MDE and pre-structuring device approach of e-learning platforms promote usage-centered designs which are important in an educational approach.

3.3.4   Web services Plugin

E-learning platforms do not always provide native service interface. So, our proposition is to wrap their functionality in a Web service plugin (cf. Table 2). Such services ease the contextualization step. The dialog these services promote allows the Pre-Structuring Device to take into account -in the application context- already established structures. As Web Services are built on industry standard protocols, and because Web Services are easy—even trivial—to implement with PHP and Java, we decided to implement the services to wrap the Moodle application as Web Services (Hoogstoel et al., 2008). They are indeed a useful technology for connecting a learning environment to organizational data and processes, and for integrating various kinds of common functionalities. In our experimentation, we address a Moodle application but our work is also conducted on four of the most popular Open Source E-learning platforms (Ganesha, Claroline, Dokeos, Atutor). There is no standard defining how to address a specific e-learning platform. E-Learning Framework initiatives present the different service-oriented architectures applied to the development and integration of computer systems in the learning domain. Amongst them, "IMS Enterprise specification: People and Groups" are the definition of how systems manage the exchange of information describing people, groups and memberships within the context of learning. Our proposition uses and expands this specification to define the services adapted to our goal.

about	createNewLabel	getAllWikis	affectSectionToAssignment
auth	getAllCourses	getAllAssignments	affectSectionToLabel
createNewUser	getAllTeachers	affectUserToGroup	renameCourse
createNewGroup	getAllUsers	affectGroupToCourse	renameCategory
createNewCategory	getAllGroupsOfACourse	affectCourseToCategory	removeCourse
createNewCourse	getAllGroups	affectTeacherToTheCourse	removeCategory
createNewSection	getAllStudents	affectStudentToTheCourse	removeUser
createNewAssignment	getAllCategories	affectSectionToCourse	removeWiki
createNewWiki	getAllSectionsOfACourse	affectSectionToForum	removeAssignment
createPageOfWiki	getAllSections	affectSectionToDatabase	removeStudentFromCourse
createNewForum	getAllForums	affectSectionToWiki	removeTeacherFromCourse
createNewDatabase	getAllDatabases		

Table 2: Our proposition of a generic web services API

3.3.5   Moodle Metamodel

It is generally difficult to define the technological metamodel of a domain. In an E-learning context, this problem is reinforced by the diversity of platforms. Our approach defines the features that such a metamodel must have and is organized as follows:

- Limitation of the functionalities.

- Identification of the element factories.

- Definition of the factorization mechanism.

The first point consists of restricting the modeling domain to the web services considered, excluding for example the global setting (security, display pattern…). From the analysis of these services, we can't identify element factories and their capacity to set elements which can be used by the web services. These represent the concepts of the metamodel.
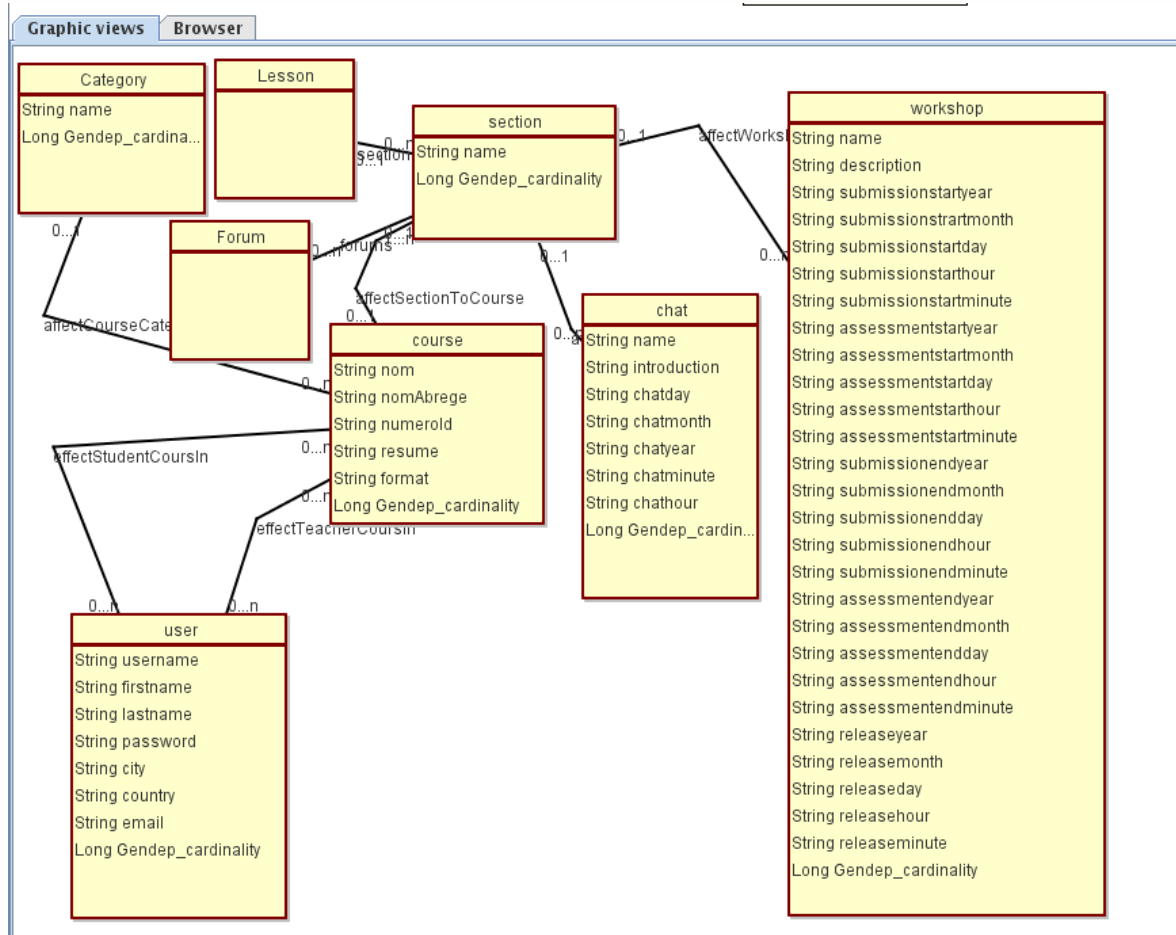

Figure 25: Part of the Moodle metamodel.

The definition of a factorization mechanism is the subject of our third part. The mechanism we propose is based on the fact that a model is a simplified view of a system. Therefore a model element can factorize the system collection of elements. This profiling mechanism is more general than the mechanism of cast definition that we use to generate our specific constructor.

3.3.6   From CPM to Moodle

The transformation rules from a CPM model to a Moodle model have not been written. It is a very difficult task:

- A lot of information in a CPM model is only for documentary purposes,

- Underlying CPM orchestration mechanisms have no equivalent on Moodle platforms,

- As a consequence of the second point, such rules would be very subjective, that is to say, depending on pedagogical practices of the rule writer.

We only present in this part a small part of what "it" could be. We start from figure 6 which represents workspaces that include different forums. The idea is to create forums functionalities in tune with the forums offered by the Moodle platform. Figure 6 shows what a designer has to create: a resource called "Forum" and for each forum that he/she wants to create, a class that inherits from the previous resource must be created too. So here the principle of forum mapping rules is to search all resources containing "Forum" and to look for classes that inherit from them.

Within ModX, rules are written in Java. Generally rules start by referencing CPM and Moodle metamodels :

```
modx.MOF.Model.Package CPM=getMetamodel("CPM");

modx.MOF.Model.Package Moodle=getMetamodel("Moodle");
```

Then, in the script, the user is asked to choose the source mode; then, our rules create a corresponding target model:

```
//choose CPM source model

modx.MOF.Instance.Model cpmModel=chooseModel(CPM);
if (modelCPM==null) return;

//create target Moodle model

java.lang.String moodleModelName=askName("Moodle model
    name ?",cpmModel.getName()+"_moodle");
modx.MOF.Instance.Model moodleModel =
    modx.MOF.Instance.Models.newModel(moodleModelName,CPM);
```

The rules load all resources of the selected CPM model and look for "*forum*" ones:

```
//Load resources

modx.MOF.Instance.Object[] resources =
  getObjects(getClass("CPM.contents.Resource"),cpmModel);

//Select *Forum* resources

java.util.Vector cpmForums=new java.util.Vector();
for (int i=0;i<resources.length;i++)
 if (resources[i].getName().toUpperCase().matches(".*\\bFORUM\\b.*"))
   cpmForums.add(resources[i]);
```

the rules search for classes inheriting (directly or not) from previous selected resources:

```
//Select also forum-like classes

modx.MOF.Instance.Object[] classes =
  getObjects(getClass("CPM.contents.Class"),cpmModel);
boolean doARoundAgain=true;
while (doARoundAgain) {
    doARoundAgain=false;
    for (int i=0;i<classes.size();i++) {
      if (!(cpmForums.contains(classes[i])))
        if (cpmForums.contains(classes.get("superClassifier"))) {
            cpmForums.add(classes[i]);
            doARoundAgain=true;
        }
      }
    }
}
```

Finally, the rules create corresponding forums in the Moodle target model:

```
//Create Equivalent Moodle forums

for (int i=0;i<cpmForums.size();i++) {

  modx.MOF.Instance.Object currentCpmForum =
      (modx.MOF.Instance.Object)cpmForums.get(i);
  modx.MOF.Instance.Object moodleForum =
    createObjectInModel( currentCpmForum.getName(),
          getClass("Moodle.contents.Forum"),moodleModel);
}
```

3.3.7 *Specific Constructor implementation*

For each application, we have to create a specific constructor which allows the context definition of the modeled Pre-Structuring Device. From the Pre-Structuring Device model, this constructor generates a contextual Pre-Structuring Device through the platform Web services plugin. The implementation of the specific constructor is almost exclusively generated from the metamodel interpretation and from the file describing the different Web Services. Only the methods allowing the deployment and the retrieving of the object on the e-learning platform must be completed to address the correct service.
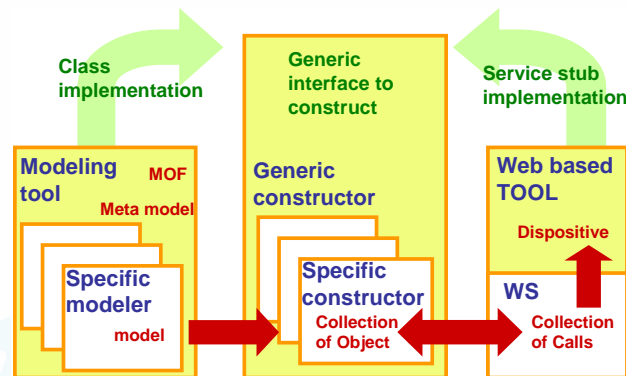
Figure 26: Gendep Technical process

3.3.8 *Synthesis*

In this subsection, we presented ongoing work on porting CPM scenarios to the Moodle Community. For the time being, such effort showed that technology enables developers to map learning activities with Moodle components and services, and as a consequence to deploy CPM learning scenarios on the Moodle platform. This advance makes possible to spread CPM language abroad the Moodle community, particularly those users who try to deploy constructivist learning scenarios with the Moodle platform. Since the design and development of Moodle were guided by social constructivist pedagogy with an emphasis on tools that promote collaboration and self evaluation (Dougiamas and Taylor, 2003), CPM and the Bricoles toolkit can clearly help Moodle users draw and deploy a scenario that makes use of such collaborative and self evaluation tools.

Yet, such an approach can introduce further complexity: Moodle is a Course Management System that is specialized in editing and using online-courses; not to design such online-courses. While Moodle uses Web forms to edit courses, CPM favours diagrammatical tools. As a consequence, usability of a platform that would embed CPM in Moodle (thanks to the Bricoles Toolkit) is far from being guaranteed.

4   Conclusion and perspectives

CPM is a visual, layered, semi-formal, multi-perspective language dedicated to the description of collaborative learning scenarios with special emphasis on Problem-Based Learning (PBL). It is not a drawing tool; it is based on the CPM profile that defines the syntax and semantics of such a visual language.

In this paper, we first presented in section 2 the way designers can model the planets game learning scenario with CPM language. We showed that, by means of the layering mechanism, designers may tackle more easily a complex situation using this graphical and conceptual feature of CPM language: they start with a coarse-grained description to grasp the global situation (cf. Figure 8 and Figure 7) and can then decompose each element to get a complete and detailed description. Next, with the multi-perspective mechanism, the designers may focus on the same objects in different perspectives (for example, the *interview repository* in Figure 4 and in Figure 6) and at different levels of abstraction (for example, the *Act2:Game* in Figure 7 and in Figure 8). Thanks to such language, they may focus on the sequencing of activities, the behaviour of a particular activity, role responsibilities, *etc*. The browser provides a uniform access to all objects of the learning scenario (cf. Act2:Game in Figure 10).

The different case-studies that we conducted over three years with this language demonstrated (Nodenot, Laforcade, Le Pallec, 2007) that even though most pedagogues are not able to produce, without methodological support, a set of CPM coherent models, both pedagogues and developers can contribute to and benefit from such design models. Among the recurrent difficulties faced by designers, we underline two important points:

1. The concepts offered by CPM language enabled designers to map the services supplied by an LMS according to the specificities of the activities to be fulfilled within a learning scenario (*cf.* the CPL stereotypes of Figure 9 that represent different functionalities offered by a forum tool). But the CPM toolset did not offer any functionality to help them deploy such activities in a concrete LMS form. As a consequence, there was an important gap between the time for designing a learning scenario and the time for its evaluation on a concrete LMS. Such a gap was somewhat frustrating since the models produced at design time could have been transformed into code (cf. Model-Driven Engineering techniques and tools).

2. The CPM toolset provides the designers with a modeler built on top of the Objecteering UML case-tool. We developed different CPM assistants, wizards and macros to help the designers select and use CPM concepts according to the diagram being produced, to automatically generate an adequate hierarchy of packages (cf. figure 10) when a new project was being created. Yet, pedagogues and designers still felt that the toolset was not usable enough for people without computer-science skills. They were not confident in the interaction process enforced by the underlying UML case-tool and they often asked us to propose the CPM in a dedicated visual environment.

In section 3, we presented different works to answer such implementation weaknesses. Thus, we outlined a MOF solution (section 3.1) and an Eclipse GMF solution (section 3.2) instead of the UML profile approach. We also proposed in section 3.3 some indications to transform CPM models into LMS compliant data and tools functionality. This work in progress already demonstrates that going towards more usability for CPM is a reachable objective. The usability of the next versions of the CPM editor will then need to be assessed by pedagogues in real-world case studies as we have done with the current CPM editor.

5   References

Allert, H. (2005). *Modeling Coherent Social Systems for Learning*. Thesis Dissertation, Hannover University (Germany).

Allilaire, F., J. Bézivin, F. Jouault and I. Kurtev (2006). ATL - Eclipse Support for Model Transformation. *Proceedings of the Eclipse Technology eXchange workshop (eTX), ECOOP 2006 Conference* , Nantes, France.

Botturi, L., M. Derntl, E. Boot and K. Gigl (2006). A Classification Framework for Educational Modeling Languages in Instructional Design. *6th IEEE International Conference on Advanced Learning Technologies (ICALT 2006)*, Kerkrade (The Netherlands).

Caron, P.-A. (2007). *Ingénierie dirigée par les modèles pour la construction de dispositifs pédagogiques sur des plateformes de formation*. Doctorat en Informatique de l'Université de Sciences et Technologies de Lille.

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

Caron, P. A. (2007). Web services plug-in to implement "Dispositives" on Web 2.0 applications. *EC-TEL 07 International Conference, Springer LNCS*, Crete (Greece).

Caron, P. A., F. Hoogstoel, X. Le Pallec and B. Warin (2007). Construire des dispositifs sur la plateforme Moodle. *Moodlemoot 2007 Conference*, Castres (France).

Dougiamas, M. and P. C. Taylor (2003). Moodle: Using Learning Communities to Create an Open Source Course Management System. *EDMEDIA 2003 Conference,* Honolulu, Hawaii (USA).

EMF. (2008). *Eclipse Modeling Framework Project*. Accessed online on May 2008, at http://www.eclipse.org/modeling/

GMF. (2008). *GMF Project*. Accessed online on September 2008, at http://www.eclipse.org/gmf/

GMT. (2008). *GMT Project*. Accessed online on June 2008, at http://www.eclipse.org/gmt/

Hoogstoel, F. and P.-A. Caron. (2008). *projet GenDep-Moodle, Espace dédié au développement WS Moodle*. Accessed online on September 2008 at http://www.assembla.com/wiki/show/MoodleWSPourGenDep .

JET. (2008). *JET Project*. Accessed online on September 2008 at http://www.eclipse.org/modeling/m2t/?project=jet

Jouault, F., J. Bézivin, C. Consel, I. Kurtev and F. Latry (2006). Building DSLs with AMMA/ATL, a Case Study on SPL and CPL Telephony Languages. *Proceedings of the 1st ECOOP Workshop DSPD*, Nantes, France.

Kelly, S. and J. P. Tolvanen (2008). *Domain-Specific Modeling*. Wiley-IEEE Computer Society Press.

Laforcade, P. (2004). *Méta-modélisation UML pour la mise en oeuvre de situations problèmes coopératives*. *LIUPPA*. Doctorat en informatique de l'Université de Pau et des Pays de l'Adour (France).

Le Pallec, X. (2001). RAM3 : un outil dynamique pour le Meta-Object Facility. *LMO2001: Langages et Modèles à Objets, numéro spécial de la revue L'Objet, Hermes, 2001, 79-94*.

Nodenot, T., P. Laforcade and X. Le Pallec (2007). *Visual Design of coherent Technology-Enhanced Learning Systems: a few lessons learnt from CPM language*. *Handbook of Visual Languages in Instructional Design; Theories and Practices*. L. Botturi and T. Stubbs, Hershey, PA: IDEA Group**:** 254-280.

OMG. (2007). *Meta-Object Facility (MOF) Specification*. Accessed online on February 2007 at http://www.omg.org/mof

Patrascoiu, O. (2004). YATL:Yet Another Transformation Language. *1st European MDA Workshop*, University of Twente, The Netherlands.

Vignollet, L., J.-P. David, C. Ferraris, C. Martel and A. Lejeune (2006). Comparing Educational Modeling Languages on a Case Study. *Workshop in conjunction with the 6th IEEE International Conference on Advanced Learning Technologies (ICALT 2006)*, Kerkrade, The Netherlands.