N° attribué par la Bibliothèque

ACADÉMIE DE BORDEAUX |_|_|_|_|_|_|_|_|_|_|

THÈSE

présentée à

L'UNIVERSITÉ de PAU et des PAYS de l'ADOUR

ÉCOLE DOCTORALE DES SCIENCES EXACTES ET DE LEURS APPLICATIONS

par

Pierre LAFORCADE

pour obtenir le grade de

DOCTEUR

Spécialité : **Informatique**

Méta-modélisation UML pour la conception et la mise en œuvre de situations-problèmes coopératives

---

Soutenue le 15 Décembre 2004

Après avis de :

M.	**A. DERYCKE**	Professeur – Université de Lille I	**Rapporteur**
Mme	**C. SOULE-DUPUY**	Professeur – Université de Toulouse I	**Rapporteur**

Devant la commission d'examen formée des rapporteurs et de :

Mme	**D. HERIN**	Professeur – Université de Montpellier II	**Présidente**
M.	**F. BARBIER**	Professeur – Université de Pau	**Directeur de thèse**
M.	**T. NODENOT**	Maître de Conférences – Université de Pau	**Examinateur**
M.	**C. SALLABERRY**	Maître de Conférences – Université de Pau	**Examinateur**

-2004-

Remerciements

Ce mémoire de thèse représente l'achèvement de trois années d'un long, fastidieux, mais ô combien enrichissant travail qui n'aurait pu voir le jour sans la participation, l'aide, les conseils, ou encore la présence de nombreuses personnes. Je m'excuse à l'avance pour les noms oubliés et donne ci-après les non-oubliés.

Tout d'abord, mes remerciements s'adressent à Mme Chantal Soulé-Dupuy, Professeur de l'Université de Toulouse I, pour m'avoir fait l'honneur de rapporter et de juger mes travaux.

Je remercie M. Alain Derycke, Professeur de l'Université de Lille I, de m'avoir également fait l'honneur de rapporter, de juger mes travaux et de m'aider à améliorer ce mémoire lors de nos différents échanges.

Je remercie également Mme Danièle Hérin, Professeur de l'Université de Montpellier II, de m'avoir honoré par sa présence et d'avoir accepté de présider le jury de ma soutenance de thèse.

Mes prochains remerciements s'adressent à Franck Barbier qui m'a accueilli au sein du laboratoire et m'a permis de commencer ma nouvelle vie de chercheur. Je le remercie d'avoir dirigé mes travaux et d'avoir su donner les directions de recherche nécessaires.

Un immense merci à Thierry Nodenot et Christian Sallaberry pour m'avoir encadré avec autant de sérieux, de rigueur scientifique, mais aussi d'humour, de gentillesse et d'humilité. Merci de m'avoir toujours fait confiance, sans jamais m'avoir sous-estimé et de m'avoir toujours encouragé et même soutenu dans les moments obscures qui ont parsemés ces trois années. Merci pour vos relectures minutieuses du mémoire. Je n'oublierai pas non plus les longues discussions, parfois même personnelles, avec Christian pendant les longs trajets entre Pau et Bayonne.

Je remercie les autres membres de l'équipe IDÉE, Marion, Pantxika, Marie-No, Christophe, Philippe, Patrick, Mauro pour leur aide, leurs conseils, comme les échanges scientifiques lors des réunions. Je les remercie également pour leur bonne humeur, leur humour et leur soutien. Un merci supplémentaire à Marie-No et Christophe pour leur relecture de la thèse. Merci également à Cécile et Fred qui ont fait un bout de voyage avec nous, j'espère que vous avez trouvé votre voie.

J'adresse également mes remerciements à toutes les personnes qui m'ont conseillé, aidé, encouragé, ou tout simplement écouté (c'est déjà beaucoup), qu'ils soient membres du laboratoire LIUPPA ou du département informatique de Pau ou de l'IUT de Bayonne ou de l'IAE ou de la face de droit/éco, qu'il soient Palois, Bayonnais ou bien Montois, qu'ils soient Maitre de Conférences, Professeurs, Secrétaires, Ingénieurs ou bien IATOS. Merci à vous, vous vous reconnaîtrez.

Je remercie les différentes personnes rencontrées lors des séminaires, conférences ou bien école d'été avec qui j'ai pu avoir des échanges intéressants vis-à-vis de mon sujet de thèse ; avec une *tchiote* dédicace aux membres de l'équipe Noce du laboratoire Trigone.

Une petite section à part pour remercier les anciens, actuels et nouveaux DEA/doctorants avec qui j'ai partagé, entre autres, - des discussions importantes comme totalement superficielles

ou politiquement incorrectes, - des bouts de bureaux, - des parties de jeux-vidéo *vermifugées*, - des moments de galère, - des moments de détente, etc. Allez, je vous sors de l'anonymat ! : Florent, Fatou, Mouusssaaa ![1], Julien, Julien Bis, Fabien, Christelle, Arnaud, Xavier, Fred, Martin, Didier et bien sûr Nicolas.

Un merci vraiment tout particulier à M. Champagnac, qui a été celui qui m'a initié et enrôlé dans cet étrange secte de l'informatique. Vous avez été mon gourou pendant mes premiers pas ; je vous dois également l'idée, le souhait, l'envie puis l'obstination de devenir enseignant.

C'est le tour des copains. Vous qui n'avez toujours peut-être pas compris en quoi consiste la recherche en informatique ou plus particulièrement mon sujet, vous avez contribué à votre manière à m'épauler dans ce périlleux voyage ; alors dans le désordre et avec les oublis : merci à Audrey, Carole, Pépette, Véro, Vivi, Jean-Marc, Sandrine, Jean-Gui, Fred, Éric...

Je te remercie dans une section spéciale car tu le mérites : merci Mahmoud ! Tu es un «grand», un ami très cher à mon cœur, un de ceux dont la dimension spirituelle n'a d'égale que l'humilité dont tu as toujours fait preuve. Je te remercie pour tout ce que tu m'as donné et je te souhaite le meilleur du monde.

C'est un énorme remerciement que j'adresse maintenant à toute ma famille. Vous avez su à votre manière, par vos paroles et vos gestes, m'encourager et m'accompagner dans mon cœur dans tous les moments de la thèse. Un gros gros merci merci à Delphine, Liliane et Maman ! Une pensée spéciale à mon Papa qui m'aura vu débuter cette thèse sans en connaître la fin ; c'est parce que tu as toujours été fier de moi et a toujours cru en ma réussite future que j'aurai tenu par tous les moyens à finir la thèse pour honorer ta mémoire.

Enfin, je remercie en dernier ma compagne Virginie : dans mon esprit tu es celle qui occupe la première place. Un énorme merci pour ton amour et ton soutien sans faille grâce auxquels j'ai pu aboutir ce premier projet que je m'étais fixé. Je te dois beaucoup. Sans oublier le petit Benjamin qui m'a permis de développer ma patience et m'a aidé à savoir prendre du recul.

[1] À prononcer avec une voix aiguë.

À mes parents

Table des matières

Chapitre 1

Introduction générale

Sommaire

« Les enseignants, comme les formateurs, se méfient des recettes [...] car, de toutes évidences, aucune situation d'apprentissage n'est totalement reproductible [...] mais faut-il, pour autant, renoncer à bâtir des modèles ?
Pouvons nous agir sans modèle, c'est-à-dire sans un outil qui nous permette de nous saisir du réel ? Que pourrions-nous faire si nous n'étions capables de sélectionner quelques informations pertinentes dans la masse des stimuli qui nous arrivent, de repérer les éléments sur lesquels nous décidons d'agir, d'organiser nos interventions, de finaliser l'ensemble de nos activités à partir d'une représentation que nous nous donnons du « réel » ? » [Mei02a]

1.1 Contexte de la thèse

Dans les prochaines sections nous précisons le contexte de recherche dans lequel se situe ce travail de thèse.

1.1.1 Contexte initial

Cette thèse, effectuée au sein du laboratoire LIUPPA[2], matérialise une collaboration interdisciplinaire entre deux équipes : l'équipe AOC (*Agent, Objet, Composant*) et l'équipe IDÉE (*Interaction, Document Électronique, Éducation*)[3]. Ainsi, cette thèse est sous la direction du Professeur des Universités Monsieur Franck Barbier de l'équipe AOC ; l'encadrement des travaux étant réalisé par les Maîtres de Conférences Monsieur Thierry Nodenot et Monsieur Christian Sallaberry.

En parcourant ce manuscrit, le lecteur pourra alors découvrir la forte orientation génie logiciel donnée à nos travaux. En particulier, de nombreuses orientations et pistes de recherche se justifient dans l'optique de réutilisation des savoirs et savoirs-faire des résultats de recherche des deux équipes.

1.1.2 EIAH et ingénierie des EIAH

Cette thèse s'inscrit dans le courant des travaux portant sur les Environnements Interactifs d'Apprentissage Humain (EIAH). Un EIAH désigne, dans sa définition la plus large, tout environnement informatique conçu pour favoriser un apprentissage. Un EIAH intègre des acteurs humains (les apprenants, ou encore les enseignants) comme artificiels (les agents informatiques). L'EIAH offre à ces acteurs des conditions d'interactions locales ou distantes (au travers des réseaux informatiques), ainsi que des conditions d'accès à des ressources formatives (humaines et/ou médiatisées ; locales ou distribuées) [PRC97].

La recherche en EIAH est pluridisciplinaire : chercheurs des sciences humaines (didacticiens, pédagogues ou autres acteurs des sciences de l'éducation) et chercheurs en informatique sont intimement liés à la conception d'un EIAH [Tch02a].

Malgré la grande variété d'EIAH (hypermédia, système à base de connaissances, système tuteur intelligent, plate-forme de formation ouverte et à distance) la conception d'EIAH nécessite dans tous les cas de prendre en compte une multitudes de problèmes inter-reliés [Tch02a] :

- analyse didactique du contexte (analyse des conceptions des apprenants, des obstacles à l'évolution de ces conceptions, etc.) ;
- identification des objectifs d'apprentissage ;
- étude du contexte d'insertion de l'activité ;
- conception et spécification de l'activité proposée :
  - conception de la tâche à réaliser ;
  - identification des acteurs et de leurs rôles ;
  - prise en compte des dispositions personnelles, des intentions et attitudes des acteurs ;
  - articulation acteurs/outils ;
- étude des comportements émergents, étude de l'activité réelle et de l'usage effectif des outils ;
- évaluation et affinement.

L'ingénierie des EIAH vise à supporter ces problèmes. Elle est définie comme « *l'ensemble des travaux visant à définir des éléments de méthodes et de techniques reproductibles et/ou réutilisables facilitant la mise en place (conception - réalisation - expérimentation - évaluation - diffusion) d'environnements de*

[2]Laboratoire d'Informatique de l'Université de Pau et des Pays de l'Adour.
[3]Initialement appelée SIA (*Système d'Information et Apprentissage*) au démarrage de la thèse.

formation ou d'apprentissage (dans leur articulation avec les dispositifs informatiques d'aujourd'hui) en permettant de dépasser le traitement ad hoc des problèmes » [Tch02b].

Cette ingénierie a pour objectif de fournir aux équipes chargées de la mise en place d'EIAH un ensemble de méthodes et techniques capitalisées, c'est-à-dire identifiés et décrits, pour les guider dans le processus de mise en place de leur EIAH. P. Tchounikine justifie l'existence d'une telle ingénierie en présentant un certain nombre d'intérêts : « *améliorer la définition et l'échange de résultats [...] proposer un cadre spécifique pour certains travaux [...] modifier l'économie de la recherche* » [Tch02b].

1.1.3 Rôles des modèles pour l'ingénierie des EIAH

Dans ce travail, nous nous intéressons aux rôles des modèles dans la capitalisation de connaissances pour l'ingénierie des EIAH. En effet, de nombreux modèles sont manipulés dans le processus de mise en place d'EIAH (modèles didactiques, modèles de résolution collective, modèles de dialogue ou encore modèles de l'apprenant). L'ingénierie s'intéresse alors à identifier lesquels de ces modèles sont capitalisables ou réutilisables.

M. Baker propose une analyse du rôle des modèles en EIAH et plus globalement pour l'AIED (*Artificial Intelligence and EDucation research*) : « *AIED research [...] as the use of computers to model aspects of educational situations that themselves involve the use of computers as educational artefacts, some of which may incorporate computational models* » [Bak00]. Ceci l'a conduit à définir les trois rôles suivants pour les modèles :

1. modèle comme outil scientifique : les modèles (interprétables ou non par la machine) sont utilisés comme support de compréhension, communication et prédiction de certains aspects de situations d'apprentissage ; le terme *simulation* est parfois employé (par exemple un modèle décrivant les interventions d'un tuteur pour une situation d'apprentissage).
2. modèle comme composant d'un système : ces modèles interprétables sont utilisés comme composants de l'artefact éducatif (qui peut être un composant du dispositif informatique d'un EIAH ; par exemple un modèle de l'apprenant embarqué dans le dispositif).
3. modèle comme une base pour la conception : il s'agit de modèles décrivant un processus éducatif (accompagnés de leur théorie d'enseignement/apprentissage) formant la base pour la conception d'un artefact informatique pour l'éducation (par exemple un modèle représentant le système d'information d'un EIAH).

Notre réflexion concerne le rôle des modèles dans le processus de mise en place d'EIAH et plus précisément pour l'étape de conception d'EIAH particuliers que nous allons présenter. En particulier, la section suivante justifiera en quoi les modèles nous intéressant constituent, vis-à-vis de la classification de M. Baker, des modèles comme outil scientifique (point 1) et des modèles pour la conception (point 3).

1.2 Cadre de travail

La notion d'EIAH couvre une vaste diversité de systèmes de modèles. Nous précisons dans cette section le contexte précis de notre étude.

1.2.1 Champ d'étude des EIAH

Notre travail sur les EIAH est motivé par deux principaux facteurs. La première influence provient des considérations pédagogiques actuelles. Dans la littérature pédagogique, il y a une forte reconnaissance de l'importance des interactions dans l'apprentissage ; les interactions sociales entre apprenants (collaboration, coordination) comme les interactions de l'apprenant avec l'environnement d'apprentissage sont centraux. Le second facteur d'influence de nos travaux est motivé par l'avancée technologique actuelle en matière de formation à distance. La généralisation des réseaux et l'intégration des TIC[4] ont fait évolué les environnements technologiques supports aux formations vers des dispositifs de production de masse propices à la constitution d'une ingénierie.

Notre positionnement est ainsi situé dans le cadre de la formation à distance, ou *e-formation*, pour lequel la conception et la mise en œuvre de formations pour un apprentissage humain relèvent davantage d'un processus industrialisé étant donnés les enjeux sous-jacents (nombre d'apprenants visés, acteurs impliqués dans les processus de conception et de mise en œuvre, ou encore recherche de la réutilisation des formations à moindre coût).

Notre restriction des EIAH s'applique alors sur deux axes :

- Nous avons limité nos travaux de recherche à une forme particulière d'apprentissage. Il s'agit de l'apprentissage par *situation-problème*[5]. 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 [Des96].
- Nous nous intéressons aux plates-formes de formation à distance[6] comme environnement informatique support aux situations d'apprentissage.

Nous avons choisi l'apprentissage par situation-problème comme cadre théorique à nos travaux car ce type d'apprentissage est cohérent avec les théories d'apprentissage contemporaines comme le constructivisme et parce qu'il est de plus en plus populaire dans la pratique. L'équipe de recherche dans laquelle cette thèse s'inscrit a proposé de nombreux travaux relatifs à ce type d'apprentissage [Mar98, Nod01, Bes02, Sal02a, Sal02b]. De plus, l'apprentissage à distance par situation-problème est une démarche d'apprentissage cohérente comme l'évoque P. Deschênes : « *la formation à distance constitue un champ d'application privilégié des concepts fondamentaux du constructivisme* » [Des96].

Notre intérêt porté aux plates-formes de formation à distance est justifié par le nombre croissant de travaux s'appuyant sur ces dispositifs et également parce que ces systèmes sont « *orientés pédagogie* » [Tch02b] et se focalisent sur l'organisation et l'identification de contenus appropriés. Le choix des plates-formes nous permet aussi de nous affranchir de la réalisation d'un nouvel environnement informatique.

La mise en place d'un EIAH répondant à notre cadre spécifique consiste à opérer des choix sur l'objet de l'apprentissage et son organisation (vis-à-vis des situations-problèmes) et également sur son instrumentation (en articulation avec les plates-formes de formation à distance) en fonction des objectifs pédagogiques, du contexte et du public cible. Dans notre cadre restreint du domaine des EIAH, notre

[4]Technologies de l'Information et de la Communication.

[5]Le terme anglais est Problem-Based Learning (PBL).

[6]Bien que souvent appelées *plates-formes de FOAD* (Formation Ouverte et A Distance), nous préférons les appeler *plates-formes de formation à distance* qui nous semble davantage élargir leur champ d'utilisation. Par souci de concision, nous utiliserons parfois uniquement le terme de *plate-forme*.

intérêt se porte sur la **conception** de formations, basées sur l'apprentissage par situation-problème, diffusées à travers le Web par les plates-formes spécialisées.

1.2.2 Mise en place appropriée

Tout d'abord, nous donnons une terminologie essentielle de concepts liés au cadre spécifique de travail de la e-formation (section 1.2.2.1). Dans ce cadre de la e-formation, la mise en place d'une formation utilisant les plates-formes de formation à distance suit généralement deux processus identifiés sous les noms de *processus de conception* et *processus d'utilisation*. Nous synthétisons le détail de ces processus en reprenant les travaux de T. Vantroys [Van03] (sections 1.2.2.2 et 1.2.2.3). Ensuite, nous détaillons le rôle joué par les acteurs participant à la phase de conception nous intéressant (section 1.2.2.4).

1.2.2.1 Terminologie

De nombreux concepts de la e-formation seront présentés dans les sections suivantes. Nous donnons pour chacun d'eux une définition précise afin de lever toute ambiguïté pour la suite de la lecture de ce mémoire.

Dispositif de formation : ensemble d'éléments (méthodes, outils, procédures, routines, principes d'action) articulés ayant pour finalité la production de compétences individuelles et collectives ; ensemble de moyens matériels et humains destinés à faciliter un processus d'apprentissage[7]. Le dispositif ne se limite donc pas aux aspects techniques ou opérationnels mais prend aussi en compte les relations entre les acteurs (enseignants, formateurs, étudiants).

Situation d'apprentissage : selon Meirieu il s'agit d'« *une situation (ensemble de dispositifs) dans laquelle un sujet s'approprie de l'information à partir du projet qu'il conçoit. Il s'appuie alors sur des capacités et des compétences déjà maîtrisées qui lui permettent d'en acquérir de nouvelles* » [Mei02a]. Pour compléter cette définition le dispositif correspond à « *la construction didactique élaborée à partir d'une opération mentale que l'on veut faire effectuer au sujet pour l'amener à une acquisition donnée. Le dispositif met en œuvre des matériaux et des consignes-structure qui, ensemble, incarnent l'opération mentale.* »

Ainsi, le dispositif de formation représente le *moyen* pour mettre en place une situation d'apprentissage.

Module d'enseignement/apprentissage : les modules sont des éléments de structuration d'une formation qui constituent des unités pédagogiques définis en termes d'objectifs de formation, de prérequis, de contenus et d'une durée. Une formation se compose de plusieurs modules. Dans le cadre de la e-formation, le module est le résultat produit par le processus de conception.

Scénario pédagogique : défini comme « *le résultat manipulable de la modélisation d'une situation d'apprentissage* »[Dae03] ou plus précisément comme « *la description du déroulement d'une situation d'apprentissage en termes de rôles, d'activités et d'environnement nécessaire à sa mise en œuvre, mais aussi en termes de connaissances manipulées* » [Per03]. Dans un scénario, on trouve donc en général des objectifs, une planification des activités des apprenants mais aussi des formateurs (ou

[7] Glossaire de la e-formation : http ://www.educnet.education.fr/superieur/glossaire.htm

tuteurs), un horaire, une description des tâches des étudiants, des modalités d'évaluation. Tous ces éléments sont définis, agencés et organisés au cours du processus de conception.

On peut considérer le scénario pédagogique comme l'élément du module d'enseignement/apprentissage qui décrit le déroulement des activités d'enseignement et d'apprentissage ; le module étant à son tour une partie d'un dispositif de formation. Le dispositif met alors à la disposition du scénario des moyens logistiques et des ressources (techniques, humaines, administratives, etc.) pour être mis en oeuvre.

D'autres concepts plus généraux, comme ceux de *e-formation* ou de *plate-forme de formation à distance* sont définis en détail dans le chapitre 3 qui leur est consacré.

1.2.2.2 Processus de conception

Le processus de conception d'une formation à distance est un processus itératif qui comprend six phases principales.

Expressions initiales des besoins. L'*enseignant* décrit de manière informelle les différentes activités du module en fixant des objectifs globaux ou locaux et éventuellement des pré-requis pour réaliser le module. Il précise le type de contenu et les ressources pédagogiques nécessaires pour l'accomplissement du module.

Analyse et conception. Suite à la description informelle, un *ingénieur pédagogique* connaissant les particularités de la plate-forme de destination, en collaboration avec l'enseignant formalise le scénario pédagogique en le découpant en différentes phases reliées entre elles. L'expression de ce modèle est réalisée en utilisant un langage de modélisation pédagogique. Au cours de cette phase, certaines activités prévues par l'enseignant peuvent être remises en cause car elles nécessitent des caractéristiques que n'offre pas la plate-forme de destination. Ce travail d'analyse et de conception fait également appel aux *fournisseurs de ressources pédagogiques* (ils indiquent les ressources disponibles ou à développer pour répondre aux besoins de l'enseignant) et aux *développeurs de composants* (ils peuvent définir les fonctionnalités à développer pour permettre à la plate-forme de supporter complètement le module).

Implémentation. Cette phase consiste à réaliser l'ensemble des composants techniques et métiers (ressources pédagogiques) nécessaires pour le module d'enseignement, rôle joué par le *développeur de composants*. Celui-ci va collaborer avec l'ingénieur pédagogique et avec le fournisseur de contenu pour la définition des composants à développer et éventuellement avec l'*assembleur de composants* pour la modification des assemblages existants ou pour la constitution de nouveaux composants.

Déploiement. Tous les éléments développés ou récupérés précédemment sont mis en place sur la plate-forme. Cette phase concerne l'assembleur de composants et l'*administrateur de plate-forme*. À la fin de cette phase, le module peut alors être référencé.

Test. Cette phase permet la vérification du comportement de la plate-forme, la cohérence des modèles et la bonne agrégation de tous les composants de la plate-forme. Ces tests sont réalisés par l'ingénieur pédagogique qui assure ainsi la validité d'un point de vue pédagogique et fonctionnel. Le développeur de composants et l'assembleur de composants contrôlent l'absence de bogues et le bon fonctionnement de l'ensemble technique.

Étape	**Rôles associés**
Expressions initiales des besoins - description (souvent informelle) des différentes activités du module - objectifs globaux, locaux et pré-requis fixés - type de contenu et ressources pédagogiques nécessaires précisés	Enseignant
Analyse et conception - formalisation du scénario d'apprentissage - découpage en phases - révision des activités prévues - mise à disposition de ressources pédagogiques - définition des nouvelles ressources à développer - définition des nouvelles fonctionnalités de la plate-forme à développer	 Ingénieur pédagogique *idem* *idem* Fournisseur de ressources pédagogiques Développeur de composants
Implémentation - réalisation des composants nécessaires (nouvelles ressources et fonctionnalités)	 Développeur de composants
Déploiement - mise en place des composants - mise en place de la formation	 Assembleur de composants Administrateur de plate-forme
Test - vérification du comportement de la plate-forme - contrôle technique	 Ingénieur pédagogique Assembleur de composants et développeur de composants
Évaluation - évaluation du comportement des apprenants et des tuteurs	 Administrateur de la plate-forme

TAB. 1.1 – Les différentes étapes pour le processus de conception d'une formation sur une plate-forme (synthèse inspirée de [Van03]).

Évaluation. Cette étape consiste à évaluer le comportement des apprenants et des tuteurs afin de vérifier que le modèle prévu initialement correspond bien aux besoins des différents acteurs et éventuellement détecter les ajustements récurrents qui pourront par la suite être directement intégrés dans le module et ceci dans une démarche d'amélioration continue. Le comportement de la plate-forme au cours de son utilisation en condition réelle peut également être évalué, demandant peut-être alors d'apporter des solutions en reconfigurant et en redéveloppant certains composants de la plate-forme. Cette tâche concerne principalement les administrateurs de la plate-forme qui pourront ainsi faire des recommandations aux ingénieurs pédagogiques et aux développeurs de composants.

1.2.2.3 Processus d'utilisation

Ce processus détermine les différentes phases relatives à l'exécution du module de formation.

Instanciation. Cette phase consiste à déterminer les différentes personnes qui prendront part à la formation ainsi que leurs rôles (*apprenant*, *tuteur*). La constitution des groupes d'apprenants associés à la formation pourra se faire avec l'aide du conseiller en formation. Le tuteur vérifiera la présence des différentes ressources nécessaires pour débuter. La formation pourra ensuite commencer.

Exécution de la formation. La plate-forme est réellement utilisée, les différents modules et activités s'enchaînent. Un ou des tuteurs assurent le contrôle du bon déroulement de la formation. Un *référent* représente la personne responsable d'un groupe de formation. Lors de l'exécution différents problèmes d'ordre technique, pédagogique ou administratif peuvent être rencontrés. Les premiers seront principalement résolus par l'administrateur de la plate-forme. Les seconds sont les plus nombreux et les plus difficiles à résoudre. Le tuteur doit par exemple faire face au problème des compétences et du niveau des apprenants afin d'adapter le contenu du module, voire son organisation comme par exemple redéfinir l'ordre des différentes activités. Ce problème s'accentue lorsque les modifications ne s'adressent pas au groupe.

1.2.2.4 Acteurs impliqués dans la conception

L'objectif général de notre recherche est de faciliter le travail de conception d'apprentissage par situation-problème sur des plates-formes de formation à distance. Vis-à-vis des étapes de conception précédentes, notre intérêt se porte plus particulièrement sur les deux premières d'*expression des besoins* et d'*analyse et conception*. **Par souci de simplification, nous faisons référence, dans l'ensemble de ce mémoire, à ces deux étapes sous l'appellation générale de *phase de conception*.**

Les différents acteurs ou rôles intervenant dans le processus complet de conception détaillé précédemment sont décrits dans [Van03]. Toutefois, nous précisons les caractéristiques des rôles impliqués dans la phase de conception nous intéressant car les chapitres suivants de ce mémoire leur feront référence. Il s'agit des « *rôles de modélisation* »[Van03] :

L'enseignant. « *L'enseignant constitue le premier maillon de la chaîne de création d'un module pédagogique ou d'un parcours de formation. Il définit, en terme de compétences, les pré-requis et les objectifs des différentes activités pédagogiques. Il associe éventuellement des services (systèmes de discussion synchrones ou asynchrones), des outils spécifiques (éditeurs de textes, simulateur*

d'oscilloscope, ...) et des ressources pédagogiques (cours de physique d'électricité et questionnaires associés par exemple). Des zones de flou peuvent exister dans la déclaration des activités car il est souvent difficile de prévoir entièrement le déroulement d'un module. L'enseignant ne possède pas obligatoirement une vue globale du processus d'apprentissage. On peut le caractériser comme expert d'un domaine particulier. Il s'exprime de manière informelle, c'est à dire dans un langage non compréhensible directement par un système informatique. Cette description peut s'exprimer à la manière d'un scénario ou d'une pièce de théâtre en décrivant toutes les scènes (rôles, actions à réaliser, ...) et leurs enchaînements. Il doit tenir compte du contexte (de la plate-forme et administratif) pour éventuellement modifier et ajuster sa description. Il collabore avec l'ingénieur pédagogique qui maîtrise les fonctionnalités existantes et envisageables de la plate-forme, lors de la formalisation des modèles et de la mise en oeuvre des parcours. Il fait office de consultant auprès du conseiller en formation pour juger de l'aptitude d'un éventuel apprenant à suivre la formation. »

Le fournisseur de ressources pédagogiques. « *Il conçoit, développe (ou fait développer) et met à disposition de l'enseignant et de l'ingénieur pédagogique des ressources pédagogiques (numérique ou non) pouvant être utilisées lors de l'exécution d'un module. Il représente une force de proposition lors de la formalisation des modèles, il peut les orienter en fonction du contenu pédagogique et de son ordonnancement défini indépendamment des tâches d'apprentissage. Il procure les composants «métier», i.e., les différentes briques directement liées à la fonction de base de la plate-forme, l'enseignement.* »

L'ingénieur pédagogique. « *L'ingénieur pédagogique est l'homme-orchestre de la plate-forme. Il a une vue métier sur la plate-forme. Il intervient tout au long du cycle de conception. Il fait le lien entre la modélisation des modules, leurs développements et leurs utilisations. Dans les phases d'analyse et de conception, il assure la formalisation des modèles en accord avec l'enseignant et en tenant compte des caractéristiques de la plate-forme. Il est capable de discuter avec le développeur de composants afin d'établir le cahier des charges des nouveaux éléments à introduire dans la plate-forme. Il peut guider le conseiller en formation lorsqu'il souhaite mettre en oeuvre une nouvelle formation. Il peut dialoguer avec le référent et le tuteur pour réaliser des modifications du module. Ces modifications peuvent porter sur les ressources pédagogiques ou sur l'enchaînement des activités. Ses principales exigences concernent les modèles. Ces derniers doivent prendre en compte un vocabulaire et des concepts proches de ceux de l'enseignant afin que l'ingénieur pédagogique réalise facilement, i.e., de la façon la plus naturelle possible, le passage vers la formalisation. Cela nécessite notamment des outils de manipulation supportant une visualisation des modèles compréhensible par l'enseignant. L'ingénieur pédagogique désire également une certaine indépendance des modèles vis-à-vis de la plate-forme d'exécution. En effet l'évolution du système ne doit pas remettre complètement en cause des modèles opérationnels déjà éprouvés.* »

Ces trois rôles intervenant dans notre phase de conception constituent l'équipe pluridisciplinaire à qui nos travaux de recherche s'adressent. Nous précisons que le rôle d'*enseignant* est un rôle général représentant en fait l'ensemble des acteurs des sciences humaines et sociales (didacticien, sociologue, psychologue, ou encore pédagogue) pouvant intervenir dans la conception de situations d'apprentissage.

1.2.3 Modèles concernés

La phase de conception, telle que nous l'envisageons, nécessite un travail de « design pédagogique » [Paq97]. Il s'agit d'un processus ou méthode pour la production d'enseignements ou apprentissage en formation à distance. Le design pédagogique doit proposer des méthodes et techniques adaptées pour faciliter la conception d'EIAH. Dans cette thèse, nous nous intéressons plus particulièrement aux rôles joués par les modèles dans ce processus de conception. En effet, les modèles sont la base sur laquelle se greffent les méthodes et les outils.

Les modèles doivent favoriser, dans un premier temps, la conception des situations-problèmes en terme d'analyse, de description et d'organisation de la situation d'apprentissage (quelles sont les ressources ? les critères de succès à spécifier ? les activités à proposer ? les rôles à jouer ? etc.). Ces modèles servent de support de réflexion et de communication pour l'ensemble de l'équipe multi-disciplinaire chargée de la conception. Les modèles doivent être adaptés au type de situation d'apprentissage envisagé, dans notre cas, les situations-problèmes.

Les modèles doivent également faciliter la conception de la situation d'apprentissage sur la plate-forme d'apprentissage à distance. Cette articulation entre « intention didactique » et « environnement informatique » est centrale aux EIAH [Tch02b]. Pour ceci, les modèles de conception doivent donc également être adaptés aux caractéristiques générales des plates-formes. Le scénario pédagogique d'une situation-problème correspond au modèle final résultant de la phase de conception.

Le contexte de recherche de cette thèse et le cadre spécifique de travail fixé sont illustrés dans la figure 1.1.

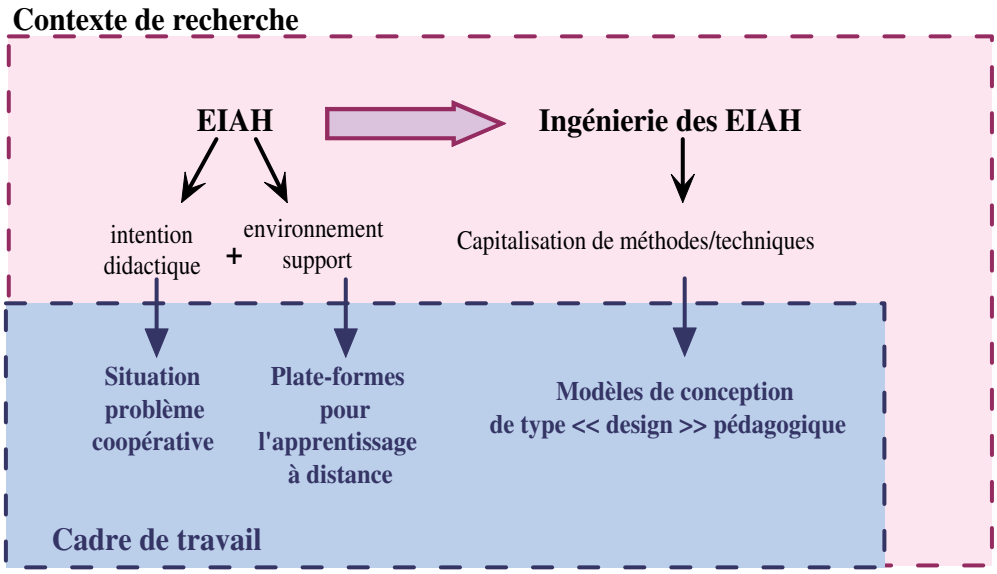

FIG. 1.1 – Le contexte de recherche et le cadre spécifique de travail.

1.3 Objectifs de la thèse

Dans cette section, nous présentons les objectifs et les orientations initiaux pour nos travaux.

1.3.1 Problématiques et défis

Notre première réflexion concerne la description des modèles de conception. Comment garantir la modélisation de la complexité d'un apprentissage de type situation-problème ? En effet, les modèles supportent et guident la phase de conception sur deux niveaux :

- horizontalement : les modèles sont différents (en termes d'objectifs, d'abstraction, de contenus et d'usages) selon les étapes dans la phase de conception (expressions initiales des besoins, analyse, conception avancée) ;
- verticalement : les modèles capturent différents points de vue ou perspectives sur la même situation d'apprentissage (pédagogique, didactique, social ou encore médiatique).

Notre travail dans cette thèse concerne principalement la mise au point d'un **langage** dédié à la mise en œuvre de nos modèles. Ce langage doit alors être adapté aux situations-problèmes et aux plates-formes génériques pour l'apprentissage à distance. Un langage se définit comme « un moyen d'exprimer une idée »[8], c'est-à-dire dans notre contexte, un moyen pour décrire les modèles de conception de situations-problèmes. Notre langage doit alors être composé d'une syntaxe et d'une sémantique ; la syntaxe définit les *sujets* (concepts/entités) et les *verbes* (liens entre les concepts), tandis que la sémantique donne l'interprétation aux sujets et verbes.

Il convient également de distinguer les utilisateurs des modèles produits avec le langage et les utilisateurs du langage. En effet, les modèles produits s'adressent à l'ensemble de l'équipe pluridisciplinaire en charge de la conception des situations-problèmes alors que le langage ne s'adresse qu'à l'ingénieur pédagogique dont le rôle est défini en section 1.2.2.4.

Nous considérons la proposition d'une méthode de design pédagogique adaptée à notre langage comme un travail postérieur à la mise au point du langage. Notre travail consiste donc à proposer ce langage indépendamment de toute réflexion d'ordre méthodologique (non traitée dans cette thèse).

De plus, le travail de cette thèse est un travail d'ingénierie des EIAH et non un travail visant la mise en place d'un nouveau type d'EIAH. Dans ce cas, nous n'envisageons pas de réaliser une nouvelle plate-forme répondant à nos besoins spécifiques. Au contraire, notre travail cherche à réutiliser et capitaliser ce que ces plates-formes proposent à l'heure actuelle.

Un second défi concerne ainsi la prise en compte de l'évolution de la recherche dans le domaine des plates-formes comme support informatique à l'apprentissage distant. Depuis quelques années, le concept de composant, issu de l'ingénierie logicielle, est apparu dans le domaine éducatif. Le domaine des EIAH, articulé entre le génie logiciel et les sciences humaines, a toujours tenté de s'approprier des techniques et méthodes du génie logiciel pour améliorer son savoir-faire. Ce concept ambigu de composant est ainsi employé de diverses manières en particulier en relation avec les plates-formes d'enseignement/apprentissage à distance. Ce second défi consistera alors plus précisément à identifier en quoi la vision composant des plates-formes d'apprentissage permet de faciliter la conception de formations.

Nous résumons ainsi nos objectifs :

[8] Définition du Wikipédia (encyclopédie Web gratuite), http ://fr.wikipedia.org/wiki/Accueil

1. Proposer un langage facilitant l'élaboration de modèles pour la conception de situations-problèmes sur des plates-formes d'apprentissage distant ;
2. Prendre en compte l'évolution actuelle des plates-formes (application du *savoir-faire composant* [Bar04]) et identifier les apports pouvant améliorer la conception des formations.

1.3.2 Orientation et justifications

Nous avons choisi d'orienter la mise au point de notre langage sur la base du langage de modélisation UML (*Unified Modeling Language*).

Pourquoi choisir une notation graphique pour un langage dédié à la conception ? Parce que les valeurs ajoutées de ces notations sont la communication et la compréhension [Cos02, Fer02]. Un *bon* diagramme aide à communiquer des idées sur la conception en évitant des détails superflus. Les diagrammes pourraient aider à mieux appréhender la complexité des situations d'apprentissage et pourraient ainsi communiquer cette compréhension à l'ensemble de l'équipe multi-disciplinaire.

Pourquoi choisir un langage orienté objet pour décrire nos modèles de conception de situations-problèmes médiatisées par les plates-formes ? La réponse est qu'il y a de nombreux avantages à utiliser les concepts et techniques orientés objets pour modéliser nos situations d'apprentissage :

Des concepts similaires. Une situation-problème peut être décrite comme un processus dans lequel des acteurs jouent des rôles et tentent ensemble (par coopération/collaboration) d'atteindre un but commun en utilisant différents types de ressources. Des règles définissent des conditions et des contraintes sur le déroulement du processus ainsi que sur les ressources, les rôles, les interactions entre tous ces éléments, etc. Tout ceci peut faire l'objet d'une correspondance avec les concepts d'objets, de relations entre objets ou encore d'interactions entre objets, en créant, par exemple, des modèles statiques et dynamiques orientés objets.

Des techniques largement éprouvées. La modélisation orienté objet, comme la programmation orientée objet d'ailleurs, a été utilisée depuis maintenant de nombreuses années et a su prouver son rôle dans la manipulation de systèmes larges et complexes et peut ainsi très bien convenir aux situations d'apprentissage.

Pourquoi choisir UML ? Le choix d'UML est justifié par les besoins suivants :

Intuitivité : UML propose une notation *a priori* facilement compréhensible, interprétable et utilisable qui convient à notre équipe multi-disciplinaire chargée de la mise en place des situations-problèmes ;

Usage de standards : UML est la notation standard adoptée pour la modélisation orientée-objet ; choisir un standard pour la description de nos situations d'apprentissage nous permet aussi de nous affranchir de la construction d'outils ;

Outil de communication : le langage UML est outil de communication reconnu dans les divers domaines l'utilisant ; l'équipe pluridisciplinaire chargée de la mise de place de situations d'apprentissage pourra utiliser ce langage commun dans le but de mieux communiquer aux autres membres sa vision métier ;

Passerelle vers d'autres modèles ou « systèmes » : UML permet de supporter la modélisation de tout système orienté objet de l'analyse jusqu'à la phase de codage. Nos modèles UML pour la conception et la mise en œuvre de nos situations d'apprentissage pourront alors servir de base

à d'autres modèles comme, par exemple, pour la réalisation de l'apprentissage (modélisation du système d'information) ou pour assurer la régulation de l'apprentissage (modèle de régulation).

Toutefois, UML est un langage trop général pour l'usage spécifique qui nous intéresse. Une solution consiste à spécialiser UML pour notre domaine d'EIAH : les situations-problèmes médiatisées au travers des plates-formes de formation à distance. Cette spécialisation est un mécanisme d'ingénierie logicielle identifié par le concept de méta-modélisation. Dans le cadre du langage UML, cette méta-modélisation est possible en créant des profils UML correspondant alors au regroupement d'extensions du langage (d'un point de vue notation et sémantique) et de règles. Nous orientons notre travail concernant le premier défi d'élaboration d'un langage vers la proposition d'un profil UML dédié à notre domaine.

1.3.3 Travail réalisé

Nos travaux ont conduit à l'élaboration d'un langage supportant la conception de situations-problèmes médiatisées sur une plate-forme de formation à distance : le langage CPM.

Ce langage permet de produire de nombreux modèles pour la phase de conception de nos EIAH. Le niveau d'abstraction variable de ces modèles leur permet de convenir autant pour guider l'équipe de conception dans l'analyse amont de la situation problème à mettre en place (expression initiale des besoins, analyse) que pour spécifier plus précisément l'organisation de la situation d'apprentissage sous la forme d'un scénario pédagogique (conception). Les modèles ont également la particularité d'être composés d'un ensemble de vues complémentaires leur permettant de décrire la situation d'apprentissage, à un moment donné de la conception, selon diverses perspectives (sociales, pédagogiques, structurelles).

Nos travaux ont conduit également à la proposition d'un modèle de *composant éducatif* : le composant CPL (Composant Pédagogique Logiciel). Il est dédié à réduire l'écart entre les besoins pédagogiques et les fonctionnalités logicielles rendues par les plates-formes génériques d'apprentissage à distance. Du point de vue de l'équipe de conception, un composant CPL représente une activité pédagogique réutilisable dans la phase de conception ; du point de vue du développeur de composants, le composant CPL est un nouveau type de composant logiciel « métier » que proposera la plate-forme de formation. Chacun de ces composants est spécifié et construit selon une démarche spécifique et grâce au modèle que nous proposons. Cette démarche aboutit à la mise en place d'une bibliothèque de modèles CPL d'activités réutilisables. Notre intérêt concerne alors l'usage de ces modèles d'activités dans la conception. Une deuxième démarche est proposée. Elle permet d'améliorer et d'étendre les perspectives de description des modèles de conception en intégrant les modèles de CPL. Une version étendue de notre langage CPM garantit cette extension.

Le langage CPM est décrit selon les différentes phases de sa mise en œuvre : le méta-modèle CPM, le profil UML CPM, le profil CPM implémenté dans l'atelier de génie logiciel Objecteering.

L'évaluation de nos propositions donne lieu à des vérifications réalisées sous la forme d'expérimentations (sur un cas d'étude appelé SMASH, sur la description d'un composant éducatif particulier) et de réalisations logicielles (implémentation du profil CPM dans Objecteering, personnalisations de l'AGL pour faciliter la mise en œuvre des modèles).

1.4 Organisation du mémoire

Ce mémoire est décomposé en deux grandes parties : la première regroupe les chapitres consacrés à l'état de l'art : chapitres 2 à 5 (cadre supérieur dans la figure 1.2)[9] ; tandis que la deuxième rassemble l'ensemble de notre contribution : chapitres 6 à 10 (cadre inférieur).

Le deuxième chapitre est consacré à l'étude du cadre pédagogique théorique de nos travaux : les situations-problèmes. La définition et les caractéristiques de ce type d'apprentissage sont décrites. Ceci nous permet alors d'établir une grille terminologique synthétique résumant les concepts clés de ces apprentissages. Ensuite, nous étudions le travail nécessaire à l'élaboration de situations-problèmes ; ainsi, nous connaissons les différents besoins de modélisation pour la phase d'expression initiale des besoins. Puis, nous présentons le cas d'étude SMASH en nous appuyant sur les concepts précédents.

Le troisième chapitre concerne l'étude du cadre technique de nos travaux : les plates-formes de formations à distance. Tout d'abord, nous commençons par présenter brièvement les environnements supports existants (autres que les plates-formes) adaptés à l'implémentation de situations-problèmes. Ceci nous permet d'aboutir à une liste de critères pour évaluer les plates-formes. Ensuite, au regard de ce travail, nous présentons et définissons le cadre de la formation à distance, des dispositifs informatiques supports, et étudions plus en détail les plates-formes : d'un point de vue structure d'accueil des formations manipulées, puis d'un point de vue technologique (architecture sous-jacente). Ceci nous amène alors à présenter succinctement les standards actuels pour la formation à distance pour lesquels de nombreux modèles liés aux plates-formes sont proposés. Ainsi, nous abordons la problématique d'interopérabilité et de réutilisation au cœur de ces standards mais également des composants pour les plates-formes. Nous proposons alors une synthèse de travaux et de projets s'intéressant à ces composants. Ceci nous permet de définir quels définitions et usages pour les composants éducatifs sont intéressants pour capitaliser le savoir-faire et ainsi améliorer la mise en place de plates-formes basées composants.

Le quatrième chapitre est dédié à l'étude des différentes techniques actuellement proposées pour supporter la phase de conception de tout ou partie de situations d'apprentissages articulées sur un dispositif de type plate-forme. Les méta-données, les ontologies éducatives et finalement les langages de modélisation éducatifs sont ainsi examinés au regard des concepts clés nécessaires pour la description de situations-problèmes.

Le cinquième chapitre étudie les possibilités de modélisation vis-à-vis de notre orientation de départ : le langage UML. Tout d'abord l'aspect modélisation UML est étudié en s'intéressant particulièrement aux usages possibles des diagrammes UML pour supporter nos modèles de conception. Dans un deuxième temps, l'aspect méta-modélisation, au sens large, et méta-modélisation UML, au travers des profils UML, sont définis et étudiés quant aux usages existants dans le domaine éducatif et en regard des usages pour notre cadre de travail.

Le sixième chapitre présente les différentes propositions de notre contribution. Tout d'abord, un bilan de l'état de l'art est proposé. Ceci nous permet de mettre en évidence les manques actuels pour l'élaboration de modèles adaptés aux situations-problèmes et aux plates-formes pour la phase de conception. Face à ces manques nous positionnons notre contribution sous la forme de deux propositions principales : le langage CPM et notre modèle de composant éducatif CPL. La démarche méthodologique suivie pour l'élaboration de notre langage est détaillée.

[9] Le chapitre 1 est ce chapitre d'introduction générale.

Le chapitre 7 présente notre méta-modèle CPM comme méta-modèle indépendant d'UML consacré à capturer la syntaxe abstraite et la sémantique de notre langage CPM. Pour cela, un modèle conceptuel est présenté et justifié au regard de la théorie de l'activité [Nar96]. Ce modèle conceptuel sert alors de base à la construction de tous les autres concepts et relations du méta-modèle CPM.

Le chapitre 8 poursuit la démarche d'élaboration de notre langage CPM en proposant le profil CPM. Celui-ci s'appuie sur la sémantique précédente et étend le méta-modèle d'UML de manière à réutiliser les éléments de notation appropriés. Le profil CPM correspond à la syntaxe concrète du langage CPM.

Le chapitre 9 rassemble nos travaux d'évaluation concernant notre première proposition. Cette évaluation est basée tout d'abord sur la vérification du profil CPM théorique en l'implémentant sous la forme d'un profil pour l'Atelier de Génie Logiciel Objecteering. Ce travail permet d'outiller notre proposition théorique, de la mettre en pratique et de produire des modèles répondant à notre objectif de supporter la phase de conception. Ensuite, des travaux de personnalisation de l'environnement de modélisation sont décrits. Ces travaux servent à démontrer les usages permis par la personnalisation des AGL, lorsque cela est permis, dans le but de faciliter la mise en œuvre des modèles. Ensuite, nous évaluons dans un deuxième temps notre langage par une expérimentation pratique liée à notre cas d'étude SMASH. Cette expérimentation est une mise à l'essai du langage CPM.

Le chapitre 10 est consacré à la deuxième proposition de la contribution. Nous présentons le composant éducatif CPL permettant de réduire l'écart entre les besoins pédagogiques de conception et les fonctionnalités techniques rendues par les plates-formes. Un modèle adapté au composant CPL est donné en détail. Puis, deux démarches sont décrites : une première démarche pour construire une bibliothèque de modèles de composants éducatifs CPL, et une deuxième pour expliquer comment utiliser cette bibliothèque pour proposer de nouveaux modèles de conception. Tous ces travaux sont ensuite évalués par le biais d'une application concrète visant l'élaboration d'un composant CPL particulier de gestion de conflit. Des prototypes logiciels élaborés pour valider techniquement l'implémentation des composants CPL sont ensuite évoqués. Puis une extension du langage CPM pour prendre en compte les modèles de CPL est proposée.

Le mémoire se termine par un chapitre de conclusions et de perspectives (chapitre 11).

Dans les annexes nous proposons divers éléments évoqués ou présentés plus succinctement dans le mémoire comme le paquetage **CPM_Foundation**, sur lequel est construit l'ensemble de la terminologie du langage CPM, et des documents de travail concernant la situation-problème SMASH. Des extraits de code d'implémentation sont également proposés en annexe afin d'illustrer le langage J (langage propriétaire à l'outil Objecteering) et son utilisation quant à l'outillage de notre langage CPM.

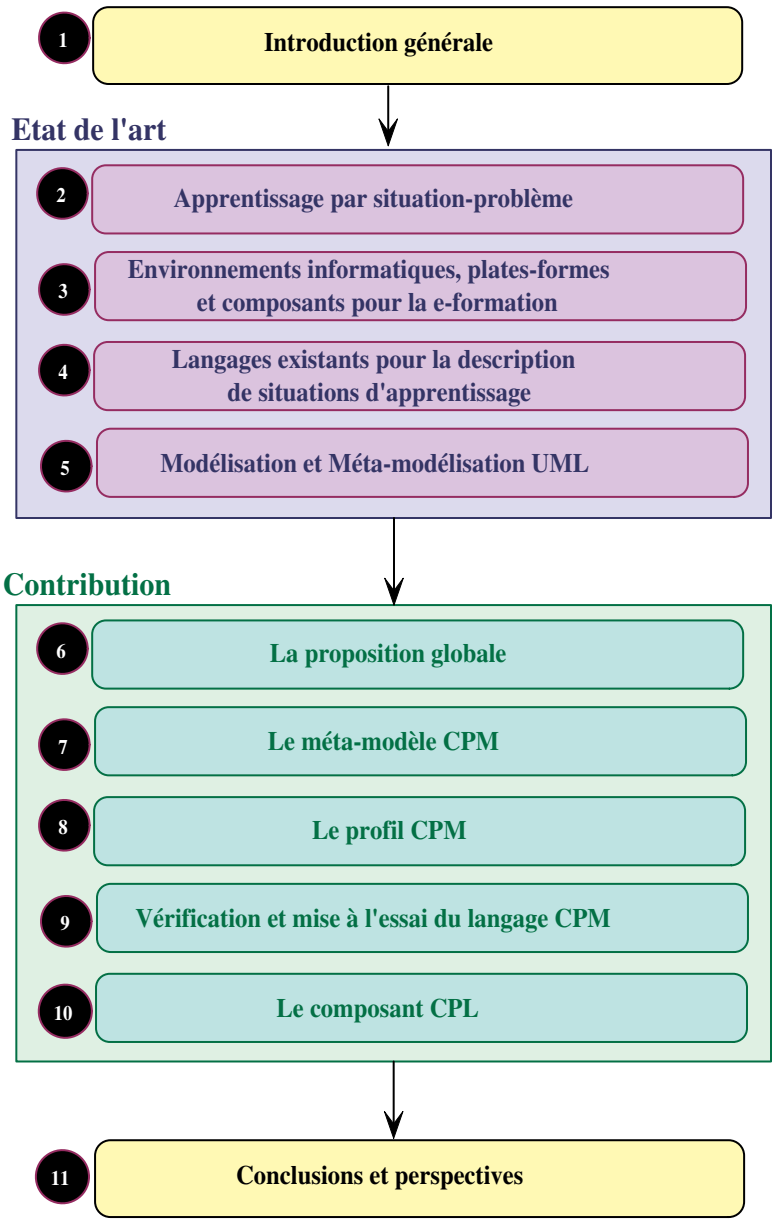

FIG. 1.2 – Organisation de la thèse en chapitres.

Première partie

État de l'art

Chapitre 2

Cadre théorique : l'apprentissage par situation-problème

« Tu me dis, j'oublie.
Tu m'enseignes, je me souviens.
Tu m'impliques, j'apprends. »
Benjamin Franklin

Sommaire

Dans le chapitre précédent, nous avons introduit l'apprentissage par situation-problème comme modèle théorique d'apprentissage au centre de nos travaux de recherche. L'objectif de nos travaux est de fournir un langage visuel de design pédagogique adapté à ce type d'apprentissage. Ainsi, ce deuxième chapitre a pour objectif d'étudier ce cadre théorique. Pour cela, nous commencerons par définir l'apprentissage par situation-problème (section 2.1). Après avoir détaillé les caractéristiques intrinsèques à ce type d'apprentissage nous développerons l'aspect coopératif des situations-problèmes (section 2.2) ; nous souhaitons mettre en avant cet aspect social de l'apprentissage entre plusieurs apprenants résolvant une situation-problème. Nous consacrons la section suivante (2.3) au processus d'élaboration d'une situation-problème. Ceci nous permet d'étudier la démarche suivie par l'enseignant pour débuter la phase de conception qui nous intéresse : l'étape d'expression initiale des besoins (Figure 2.1). Enfin, nous présentons notre cas d'étude de situation-problème coopérative : SMASH (section 2.4). Ceci nous permet d'illustrer concrètement ce chapitre théorique et également de présenter le cas d'étude sur lequel s'appuient les exemples de ce mémoire. Il sera ensuite repris pour la vérification de notre proposition. Enfin, la section 2.5 conclura ce chapitre.

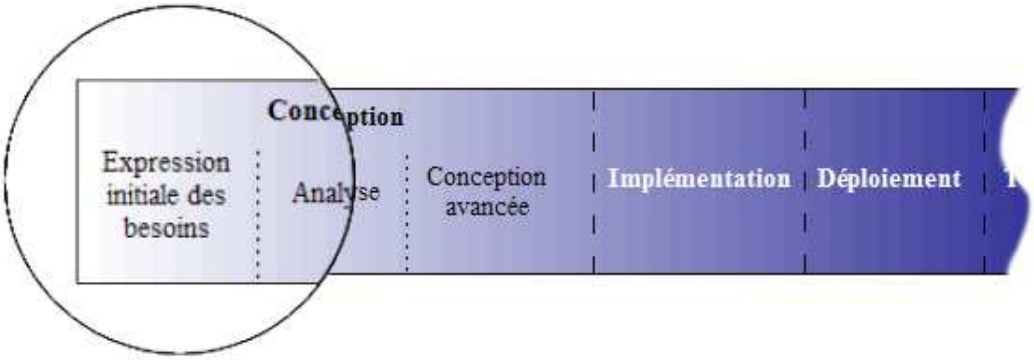

FIG. 2.1 – L'étude des PBL concerne la phase d'expression initiale des besoins.

2.1 Apprentissage par situation-problème

Dans cette section nous définissons l'apprentissage par situation-problème et présentons ses caractéristiques intrinsèques.

2.1.1 Définition

L'apprentissage par situation-problème (Problem Based Learning ou PBL[10] en anglais), encore appelé apprentissage par résolution de situation-problème, a de multiples définitions dans la littérature.

Voici quelques définitions des PBL :

[10]Par souci de simplification d'écriture nous emploierons régulièrement par la suite l'abréviation PBL en lieu et place d'« apprentissage par situation-problème » (*PBL* sera considérée du genre féminin).

Barrows *et al.* : « *...the learning which results from the process of working towards the understanding of, or resolution of, a problem* » [Bar80].

Boud et Feletti : « *Problem based learning is an approach to structuring the curriculum which involves confronting students with problems from practice which provide a stimulus for learning* » [Bou91].

Finkle et Torp : « *problem-based learning is a curriculum development and instructional system that simultaneously develops both problem solving strategies and disciplinary knowledge bases and skills by placing students in the active role of problem solvers confronted with an ill-structured problem that mirrors real-world problems* » [Fin95].

Meirieu : « *situation didactique dans laquelle il est proposé au sujet une tâche qu'il ne peut mener à bien sans effectuer un apprentissage précis. Cet apprentissage qui constitue le véritable objectif de la situation-problème, s'effectue en levant l'obstacle à la réalisation de la tâche. Ainsi, la production impose l'acquisition, l'une et l'autre devant faire l'objet d'évaluations distinctes. Comme toute situation didactique, la situation-problème doit être construite en s'appuyant sur une triple évaluation diagnostique (des motivations, des compétences et des capacités)* » ([Mei02a] p.191).

Pour compléter cette définition nous précisons également sa définition de situation didactique : « *une situation didactique est une situation d'apprentissage élaborée par le didacticien qui fournit, d'une part, des matériaux permettant de recueillir l'information et, d'autre part, une consigne-but permettant de mettre le sujet en situation de projet* ». Toujours selon Meirieu, une situation d'apprentissage est « *une situation (ensemble de dispositifs) dans laquelle un sujet s'approprie de l'information à partir du projet qu'il conçoit. Il s'appuie alors sur des capacités et des compétences déjà maîtrisées qui lui permettent d'en acquérir de nouvelles* ».

Ainsi, pour Meirieu, l'apprentissage par situation-problème est une « *pédagogie du problème et non une pédagogie de la réponse* » : l'apprentissage est lié à un obstacle à lever *implicite* dans la résolution du problème ; cet apprentissage ne correspond donc pas à la réponse *explicite* à un problème.

Une PBL est, selon ces auteurs, un modèle, un processus/démarche, un curriculum/apprentissage didactisé. Malgré ces quelques divergences, les PBL partagent toutes les mêmes caractéristiques que nous présenterons après avoir préalablement expliqué en quoi les PBL sont en cohésion avec les théories d'apprentissage contemporaines.

2.1.2 Considérations pédagogiques

L'apprentissage par situation-problème est basé sur des théories d'apprentissage contemporaines tels que le *constructivisme* (*constructivism*), *l'apprentissage situé* (*situated learning*) et *l'apprentissage adulte* (*adult learning*).

Constructivisme : dans la théorie constructiviste, les apprenants sont actifs (*learning by doing*) et construisent eux-mêmes leurs propres connaissances.

D'après Savery et Duffy [Sav95], les environnements pour les PBL sont basés sur les hypothèses constructivistes suivantes :

1. Rattacher toutes les activités d'apprentissage à un problème ou une tâche plus large.

2. Soutenir l'apprenant à s'investir personnellement dans la résolution du problème ou de la tâche.

3. Concevoir une tâche authentique[11].

4. Concevoir la tâche et l'environnement d'apprentissage de manière à refléter la complexité de l'environnement dans lequel les apprenants devront être capable de se comporter à la fin de l'apprentissage.

5. Donner à l'apprenant la possibilité de s'approprier et définir son propre processus dans le développement de la solution.

6. Concevoir l'environnement d'apprentissage de manière à permettre et encourager la réflexion des apprenants.

7. Encourager la confrontation des idées moyennant l'alternance de vues et de contextes.

8. Fournir l'opportunité et la capacité de réfléchir sur le contenu appris comme sur le processus d'apprentissage pour y parvenir.

Apprentissage situé : l'apprentissage situé, aussi appelé apprentissage en contexte, conçoit que la construction de connaissances et le développement de compétences devraient se faire dans des situations réelles ou authentiques, c'est-à-dire des situations présentant un environnement social complexe. Dans la perspective de l'apprentissage situé, la connaissance et la compréhension sont fondamentalement un produit de la situation d'apprentissage et de la nature de l'activité d'apprentissage [Lav91b]. Pour [Bro89] activité, concept et culture sont interdépendants ; aucun ne peut être compris sans les deux autres. Lave écrit : « *situated. . .does not imply that something is concrete and particular, or that it is not generalizable, or not imaginary. It implies that a given social practice is multiply interconnected with other aspects of ongoing social processes in activity systems at many levels of particularity and generality* » [Lav91a]. C'est dans un apprentissage situé que l'apprenant donnera un sens à sa démarche pour construire ses connaissances et développer ses compétences. Dans une telle orientation, on favorise les apprentissages signifiants et la transformation de nouvelles informations en des connaissances viables et transférables.

Apprentissage adulte : Camp indique que, dans sa forme la plus « pure » de réalisation, l'apprentissage par situation-problème correspond aux principes de l'apprentissage-adulte [Cam96]. L'autonomie de l'étudiant, construite sur des connaissances et expériences précédentes, et l'opportunité pour une application immédiate sont bien connus pour faciliter l'apprentissage des adultes. Elles favorisent le succès de l'approche PBL avec des élèves qui sont des étudiants adultes [Cam96]. En fait, les apprenants adultes sont davantage motivés pour apprendre quand ils savent *pourquoi* ils ont besoin d'apprendre telle ou telle chose et quand leurs connaissances et expériences passées sont utilisées comme point de départ *et* comme ressource de l'apprentissage. De nos jours, l'apprentissage par situation-problème est également pratiqué dans les écoles (maternelles, primaires).

[11] Tâche ou situation signifiante pour l'apprenant : « *an authentic learning environment is one in which the cognitive demands, i.e., the thinking required, are consistent with the cognitive demands, in the environment for which we are preparing the learner* » [Hon93].

2.1.3 Popularité des PBL

L'apprentissage par situation-problème est une méthode d'enseignement/apprentissage de plus en plus populaire. A l'origine, ce type d'apprentissage fut développé pour les étudiants en médecine au Canada [Bar80] pour les aider à améliorer leurs compétences en diagnostic. On présente aux étudiants en médecine un problème de diagnostic, généralement lié à un patient malade ou se plaignant. En utilisant une base de données et des informations test sur le patient, les étudiants sont amenés à construire un diagnostic en générant des hypothèses, en collectant des informations pertinentes selon leurs idées et en évaluant leurs hypothèses. Ils sont guidés à travers leurs tâches par un facilitateur jouant le rôle du coach ou du questionneur *socratique*.

Initialement, l'approche d'apprentissage par situation-problème était pratiquée principalement dans les écoles médicales et professionnelles. Progressivement d'autres champs d'étude ont commencé à adopter cette méthode [Rhe98] : médecine généraliste, infirmières, dentistes, pharmacie, médecine vétérinaire, santé publique, architecture, commerce, droit, ingénierie, sylviculture, sciences politiques, éducation, électronique et encore beaucoup d'autres domaines [Cam96, Sán01, For02, Nor03, Arm02, Ste02a, Lac01, Moo00]. D'après [Woo96] l'apprentissage par situation-problème peut être utilisé dans tous les sujets et à tous les niveaux.

2.1.4 Caractéristiques des PBL

Les PBL intègrent la plupart des principes généraux qui permettent d'améliorer les apprentissages. Y. Miao a étudié et listé ces principes [Mia00] :

1. Les apprenants doivent s'impliquer activement dans les activités d'apprentissage.
2. Les apprenants doivent coopérer dans leurs processus d'apprentissage pour s'aider les uns les autres.
3. Il n'est pas nécessaire que tous les apprenants apprennent de la même façon. Les activités d'apprentissage doivent être proposées de manière à ce que chaque apprenant soit capable d'avoir son style.
4. Les apprenants doivent avoir des objectifs et critères de réussite *explicites* afin de pouvoir vérifier si leurs buts sont atteints.
5. Il faut donner aux apprenants un retour (*feedback*) sur leur performance aussi vite que possible.
6. Les apprenants doivent participer à l'évaluation par le biais de rôles tels l'auto-évaluation ou l'évaluation d'un pair.
7. Un environment de travail doit être fourni dans le but précis d'aider les apprenants à réussir. Il doit prévoir que chaque apprenant ait un intérêt d'apprentissage personnel.
8. Des interactions tuteur-apprenant riches doivent être fournies par le biais de nombreux types d'événements, internes ou externes à la classe pédagogique.

Dans [Tho00], l'apprentissage par situation-problème est un modèle particulier de l'apprentissage par projet. En fait selon la littérature, les PBL sont souvent comparées à d'autres méthodes d'apprentissage : projet de recherche, étude de cas, projet de conception, confrontation expérimentale, orienté-projet, apprentissage coopératif, apprentissage basé explication, apprentissage actif ou apprentissage en petit groupe [Woo96].

Ainsi, les situations-problèmes sont plus précisément caractérisés dans la littérature comme actives, orientées adultes, centrées apprenants, collaboratives, intégrées et inter-disciplinaires. Les apprenants sont souvent en petits groupes. Les prochaines sous-sections détaillent les caractéristiques les plus importantes des situations-problèmes.

2.1.4.1 Interactions sociales riches

Dans l'enseignement traditionnel, les apprenants sont passifs et reçoivent individuellement les connaissances organisées comme une série de cours par l'enseignant. Les interactions sociales entre apprenants sont quasi-inexistantes ; l'interaction sociale principale prend généralement place sous la forme d'une communication unidirectionnelle enseignant/élève. Bien qu'assis à proximité, les étudiants coopèrent rarement. De plus, les étudiants ont des devoirs à faire à la maison et suivent des évaluations bien souvent individuelles également. L'apprentissage par situation-problème permet également une résolution des problèmes par des équipes collaboratives. Cette méthode promeut les interactions entre étudiants et aussi le travail en équipe, ce qui enrichit les compétences inter-personnelles des étudiants. Les interactions sociales n'ont pas uniquement lieu dans les salles de classe mais aussi parfois au delà selon le type d'investigation proposée par la PBL. Ainsi, les interactions sociales se réalisent entre apprenants et enseignants (agissant en tant que facilitateur ou *tuteur*) mais aussi entre les apprenants et les autres acteurs impliqués dans le scénario, *i.e.* les experts interviewés. Bien que la communication soit à la base de la plupart des interactions entre les différents acteurs de la PBL, les étudiants collaborent aussi entre eux en utilisant des outils pour définir les problèmes, générer des solutions, débattre de différentes perspectives, conduire des expériences, ou encore écrire des rapports. Les étudiants peuvent également utiliser des outils comme le téléphone pour interviewer des experts. Les élèves collectent des informations de sources identifiées variées : articles, livres, sites Web, etc. Différentes formes de médias sont utilisés (textes, images, vidéos, etc.).

2.1.4.2 Le problème est structuré autour d'un obstacle

Le problème auquel les apprenants font face est étroitement associé à des problèmes de la réalité. Les problèmes de la vie réelle sont souvent *mal structurés*[12] et souvent « *require integrated instruction, which blends disciplines into thematic or problem-based pursuits, and instruction that incorporates problem-based learning and curriculum by project* » [Jon94]. Gallagher et Gallagher [Gal94] offre une comparaison entre les problèmes bien et mal structurés (Table 2.1).

Plus les apprenants comprennent profondément le problème et plus ils ont de questions. Ils doivent identifier quels thèmes d'apprentissage ils ont besoin d'étudier et quelles informations ils devraient collecter. Les informations et connaissances rassemblées par les apprenants individuellement ou collectivement sont organisées et intégrées autour du problème à résoudre. Ainsi, ces informations et connaissances seront partagées par le groupe entier d'apprentissage. Les apprenants utiliseront les connaissances acquises pour élaborer des hypothèses et des solutions au problème. Ils ont à prendre des décisions et à fournir des solutions pour de tels problèmes. Ils appliqueront alors des lois et des preuves déduites des informations qu'ils auront collectées pour pouvoir raisonner. A travers les processus de raisonnements collaboratifs,

[12] *ill-structured* en anglais

Problèmes bien-structurés	**Problèmes mal-structurés**
Définition du problème facilement identifiable	Le problème doit être défini, voir redéfini
Toutes les informations nécessaires à la résolution du problème sont fournies	Des informations additionnelles sont nécessaires pour résoudre le problème
Focus sur la solution du problème	Focus sur la nature du problème
Une unique bonne réponse peut être identifiée	Différentes solutions sont possibles
Faible motivation à résoudre	Forte motivation à résoudre

TAB. 2.1 – Comparaison entre les problèmes bien structurés et mal structurés.

ils peuvent identifier les incohérences de leurs connaissances, découvrir les connaissances manquantes et construire des connaissances partagées. D'autres caractéristiques associées aux PBL sont liées à la structuration du problème. On parle d'*authenticité* et de *complexité*. En effet, dans un environnement de PBL, les problèmes sont présentés comme ils existent dans la réalité. Ainsi, en participant à des activités authentiques, les apprenants adoptent le comportement et les connaissances de personnes réalisant normalement ces activités dans la vie réelle. Les problèmes de la vie réelle sont généralement complexes et se raccrochent à de nombreux détails et points de vue inter-reliés ; cette complexité se retrouve également dans la structure même de la situation-problème.

2.1.4.3 Les différents rôles

Traditionnellement, les différents acteurs impliqués dans le déroulement ou l'exécution de la situation-problème adhèrent à un rôle (exemple l'enseignant et l'étudiant) dont les responsabilités sont stables sur la globalité de la situation d'apprentissage. Dans notre cas, nous privilégions les *rôles globaux* d'*apprenant* pour représenter les élèves et de *tuteur* pour l'enseignant (notre choix est en grande partie justifié par la présence de ces deux types d'acteurs comme usagers des plates-formes de formation à distance - voir chapitre 3).

Toutefois, les responsabilités d'un apprenant comme d'un tuteur peuvent varier dans différentes PBL ou au sein d'une même PBL selon sa construction. Concernant ce type de responsabilités, que nous appellerons *rôles situés*, la littérature sur les situations-problèmes fait référence aux différents rôles suivants pour les apprenants :

1. *Chercheurs de problèmes* et *résolveurs* : les apprenants anticipent, explorent, analysent et résolvent les problèmes. Ils peuvent enquêter sur les causes du problème selon de multiples perspectives et proposer des hypothèses et des solutions possibles.
2. *Planificateurs* et *producteurs* : les apprenants peuvent planifier et concevoir des méthodes ou des stratégies, pour résoudre les problèmes, aboutissant à des produits ou processus originaux.
3. *Initiateurs* et *organisateurs* : les apprenants initient, coordonnent et facilitent l'accomplissement des tâches collectives en prévoyant et en définissant les résultats visés, en décidant comment accomplir ces tâches, en anticipant les obstacles, et en *enrôlant* l'aide d'autres personnes pour atteindre les résultats.

4. D'autres rôles mentionnés dans la littérature mais que nous ne détaillons pas sont intitulés *implementors*, *performers*, *communicators*, *negotiators*, *explorer* et *partners*

Les responsabilités de l'enseignant, comme celle de donner l'information, n'ont plus leur place dans les PBL. Il devient plutôt un facilitateur ou un coach éducatif (souvent référé dans le jargon des PBL comme un *tuteur*). Jones *et al.* résument des nouveaux rôles plus précis pour les tuteurs dans les PBL [Jon94] :

1. Facilitateur. Le tuteur fournit des environnements et activités d'apprentissage riches en y incorporant des opportunités pour le travail collaboratif, pour la résolution du problème, pour la réalisation des tâches authentiques et pour le partage des connaissances et des responsabilités.
2. Guide. Dans une salle de classe collaborative, le tuteur doit agir comme un guide (un rôle complexe et varié incluant la médiation, la présentation de mode et le coaching). En servant d'intermédiaire dans l'apprentissage des élèves, l'enseignant doit fréquemment ajuster son niveau d'aide en se basant sur les besoins de l'élève. Il doit permettre à l'élève de relier les nouvelles informations avec les connaissances antérieures. Il raffine leurs stratégies de résolution du problème et leur apprend comment apprendre.
3. Co-apprenant et Co-investigateur. Tuteur et apprenants participent à des investigations en jouant des professionnels. En suivant ce modèle, les élèves explorent de nouvelles frontières et deviennent des producteurs de connaissances dans une communauté de production de savoirs. Ainsi, avec l'aide de la technologie, les élèves peuvent devenir des enseignants comme les enseignants des élèves.

Un autre type de rôle est celui lié à la situation authentique contextualisant la PBL. Ce *rôle authentique* représente une responsabilité jouée dans la réalité et permet de renforcer l'immersion des apprenants dans leur tâche (par exemple le rôle d'inspecteur de police convient pour réaliser une PBL sous forme d'enquête). Il est possible également de faire référence à des rôles joués lors des interactions entre apprenants et tuteurs dans les activités collaboratives. Ce type de *rôle collaboratif*, plus local, est développé en section 2.2.

Les différents rôles sont résumés dans le tableau 2.2. Ceux joués uniquement dans un contexte de la tâche sont souvent complémentaires à d'autres rôles. Les différents rôles peuvent être utiles dans la conception d'une PBL mais ne sont pas forcément « explicites », c'est-à-dire spécifiés précisément pendant l'élaboration de la PBL.

Type de rôle	**Temporalité dans la PBL**	**Exemple**
Rôle global	Global	Apprenant/Tuteur
Rôle situé	Global/local	Organisateur/Producteur
Rôle authentique	Global	Inspecteur/Médecin
Rôle collaboratif	Local	Questionneur/Répondeur

Tab. 2.2 – Les différents rôles d'une PBL.

2.1.4.4 Processus d'apprentissage auto-dirigés

Dans le contexte du scénario, ce sont les élèves qui ont à leur charge leur propre apprentissage. Ils définissent leurs objectifs d'apprentissage selon les difficultés qu'ils rencontrent en tentant de résoudre

le problème. Ils décomposent le but global d'apprentissage en sous-buts et décident d'un ensemble d'actions pour atteindre ces sous-buts. Les actions sont souvent programmées sur le temps par le groupe d'apprentissage. Ils allouent des ressources pour chaque action. En effet, les apprenants doivent planifier l'allocation des ressources pour chaque action mais aussi coordonner les personnes, actions et résultats dans le but d'accomplir les objectifs dans un temps imparti. A l'opposé de l'enseignement traditionnel où tous les élèves apprennent le même contenu dans la même salle de classe dans un apprentissage dirigé par le sujet, dans un scénario de PBL les élèves réalisent, individuellement ou en équipe, différentes actions d'apprentissage et ensuite partagent entièrement ou partiellement les informations recueillies. Ils ont le rôle actif de construire leurs connaissances partagées d'une manière significative selon leur plan d'apprentissage. Sur la manière de réaliser leur propre plan d'apprentissage, les élèves devront ajuster leurs besoins en fonction des ressources disponibles et parfois modifier leur plan. A la fin de chaque action, l'enseignant/tuteur donne un retour à chaque groupe avant qu'il ne procède à l'action suivante. Ce type d'instruction encourage les élèves à résoudre activement le problème, à conduire des recherches significatives, à s'engager dans la réflexion et à construire un ensemble de stratégies efficaces pour des apprentissages dans des contextes sociaux variés.

2.1.4.5 Les différentes étapes du processus

Le processus ou scénario d'apprentissage pour les situations problèmes suit des étapes plus ou moins similaires. Nous donnons ici plusieurs exemples de processus trouvés dans la littérature.

1. Pour [Mia00] :
   (a) Identifier le problème.
   (b) Identifier ce que savent les apprenants et ce qu'ils ont besoin de savoir.
   (c) Fixer les objectifs d'apprentissage et préparer un plan d'apprentissage.
   (d) Acquisition/apprentissage des connaissances.
   (e) Application des connaissances.
   (f) Evaluation/estimation des connaissances.
2. Pour [Nor03] :
   (a) Exploration du problème et génération d'hypothèses.
   (b) Identification des finalités d'apprentissage et des sources d'apprentissage.
   (c) Rassemblement d'informations et étude indépendante.
   (d) Discussion critique sur les connaissances acquises.
   (e) Application des connaissances pour résoudre le problème.
   (f) Retour réflexif [13] sur le processus.
3. Pour [Arm02] :
   (a) Analyse du problème.

[13] Le retour réflexif correspond à un ré-examen collectif et/ou individuel du cheminement parcouru dans la résolution du problème. Il est à caractère « métacognitif » : il aide les élèves à « conscientiser » les stratégies qu'ils ont mis en œuvre de façon heuristique, et à les stabiliser en procédures disponibles pour de nouvelles situations-problèmes.

(b) Apprentissage auto-dirigé.

(c) Ré-examen du problème.

(d) Abstraction.

(e) Retour réflexif.

Ces différents processus partagent des étapes similaires que nous résumons ainsi :

1. Analyse du problème.
2. Identifier pré-requis, objectifs et plan d'actions.
3. Acquisition de connaissances.
4. Application des connaissances.
5. Retour réflexif.

2.2 Situation-problème coopérative

Dans la littérature pédagogique [Bou00, Geo01], il y a une reconnaissance croissante de l'importance dans l'apprentissage de la collaboration, de la coopération et de la coordination. La question centrale concerne la compréhension de l'évolution des représentations mentales[14] des apprenants, de leurs connaissances et de leurs motivations à travers leurs interactions avec les autres acteurs ainsi qu'avec l'environnement d'apprentissage.

Notre intérêt se porte sur l'apprentissage par situation-problème et plus particulièrement sur les situations-problèmes mettant en valeur l'usage de la coopération entre les différents apprenants pour surmonter l'obstacle auquel ils font face.

Nous définissons, dans un premier temps, cette notion de coopération puis nous présentons brièvement ses caractéristiques ; enfin, nous justifions l'intérêt de son usage pour l'apprentissage par situation-problème.

2.2.1 Définitions préalables - terminologie

Lorsqu'on s'intéresse à la coopération ou à la collaboration, il convient de bien définir les termes employés et le contexte de leur usage. En effet, de nombreuses définitions existent dans la littérature.

De manière générale, les activités humaines faisant intervenir plusieurs personnes, simultanément ou non, peuvent être regroupées sous la dénomination de « travail coopératif » [Dav01]. Résoudre de manière coopérative une situation-problème à distance n'est qu'un champ d'application du travail coopératif assisté par ordinateur (TCAO ou CSCW[15]), et même plus précisément du domaine de l'apprentissage coopératif assisté par ordinateur (CSCL[16]).

Nous avons choisi de reprendre les définitions d'une étude réalisée dans un travail de thèse ([Geo01] - chapitre 2), illustrée par la figure 2.2. L'activité est dirigée par un objectif global et se décompose

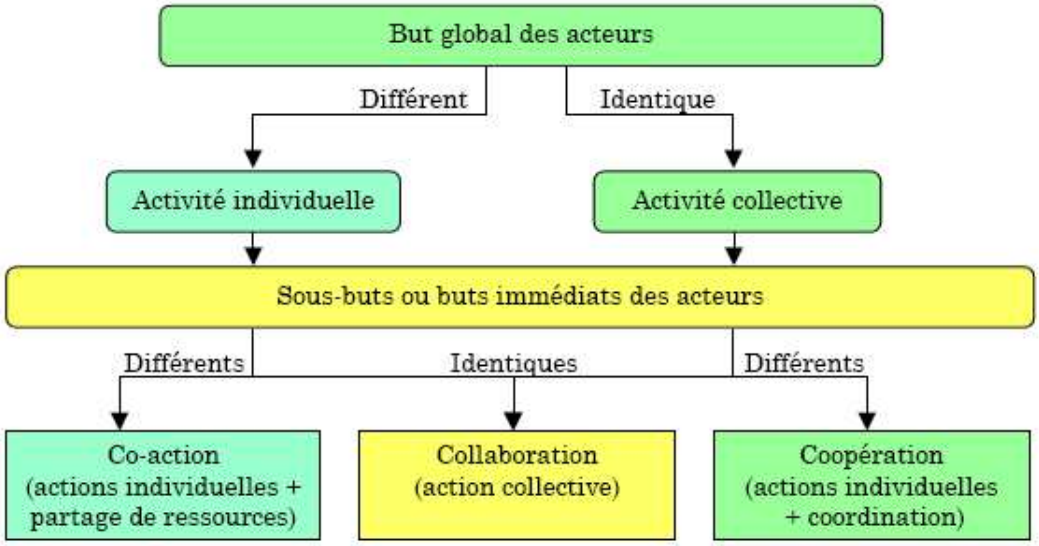

FIG. 2.2 – Activités coopératives et collaboratives [Geo01].

en actions élémentaires dirigées par des buts immédiats. Les actions sont à leur tour réalisées par des opérations de base représentant le niveau le plus élémentaire.

L'activité collective correspond à un engagement des acteurs dans un but global commun ; l'activité individuelle, quant à elle, est reliée à des objectifs globaux différents. En s'intéressant ensuite aux sous-buts immédiats de l'activité, trois cas se distinguent :

Les sous-buts sont identiques. On parle alors de collaboration pour la réalisation d'une action collective *synchrone*. Les acteurs collaborent dans le même espace de travail (réel ou virtuel) et en même temps.

Les sous-buts sont différents et l'activité est individuelle. Il s'agit dans ce cas d'une co-action, c'est-à-dire d'actions individuelles ayant un éventuel partage de ressources.

Les sous-buts sont différents et l'activité est collective. On dit que les acteurs coopèrent : actions individuelles coordonnées entre elles.

Comme le précise l'auteur, « *la collaboration et la coopération ne sont pas mutuellement exclusives et sans relation [...] les activités collectives se caractérisent par une succession de phases collaboratives et de phases coopératives...* »[Geo01].

Dans les apprentissages par situations-problèmes tous les apprenants partagent le même but global : résoudre la tâche qui leur est confiée. Leurs activités sont collectives et correspondent tantôt à des activités collaboratives, tantôt à des activités coopératives. Toutefois, il est difficile de qualifier le type d'apprentissage comme « coopératif » ou « collaboratif ». En effet, ces deux approches ont des caractéristiques communes comme celle de faire travailler les apprenants en petits groupes ou encore de considérer

[14] Désigne la conception que le sujet a, à un moment donné, d'un objet ou d'un phénomène.
[15] Le sigle anglo-saxon pour *Computer-Supported Cooperative Work*
[16] Le sigle anglo-saxon pour *Computer-Supported Cooperative Learning*

l'enseignant comme un facilitateur. Pour certains auteurs, la collaboration ou la coopération sont des propriétés émergeant lors de la réalisation des activités.

Toutefois, quatre critères majeurs différencient les deux types d'apprentissages (tableau 2.3) [Geo01]. Concernant l'apprentissage coopératif, les activités sont plus fortement structurées et les enseignants contrôlent davantage les activités ; les rôles sont assignés dès le début et on cherchera à leur enseigner des aptitudes sociales.

	Apprentissage Coopératif	**Apprentissage Collaboratif**
Structuration des activités	Forte	Faible
Contrôle de l'enseignant	Fort	Faible
Rôles des apprenants	Assignés	Négociés
Aptitudes sociales des apprenants	Enseignées	Supposées existantes

TAB. 2.3 – Caractéristiques de l'apprentissage coopératif et de l'apprentissage collaboratif [Geo01].

Notre intérêt se porte sur la résolution des situations-problèmes par les apprenants dans un cadre d'apprentissage coopératif. L'inconvénient de ce choix est lié à la structuration plus forte des activités demandées alors que les situations-problèmes s'adressent davantage à des adultes et donc proposent un apprentissage auto-dirigé : les apprenants planifient, entre autres, la réalisation de leur tâche. De plus, l'apprentissage par situation-problème se rapporte à la théorie de l'apprentissage situé selon laquelle la structuration de l'activité n'est pas quelque chose qui précède l'activité mais qui se construit dans la situation. Par contre, les avantages d'une telle structuration sont justifiées par l'utilisation de l'apprentissage par situation-problème pour des enfants moins capables de s'auto-diriger ; en effet, notre cadre d'étude concerne plus précisément l'usage de l'apprentissage par situation-problème dans les écoles primaires. La mise en œuvre de nos apprentissages sur des plates-formes de formation à distance constitue également un cadre plus propice à une forte structuration des activités. Par la suite, nous parlerons d'apprentissage par situation-problème coopérative.

2.2.2 Apprentissage par situation-problème coopérative

Alors que la résolution coopérative de PBL est considérée comme une caractéristique intégrée dans la mise en situation des PBL dans le domaine médical, tous les efforts pour la création d'environnements de PBL sont loin d'avoir utilisé des groupes coopératifs. La problématique sur l'importance ou non de la résolution collective de PBL a été traitée dans la littérature [Bou91]. L'apprentissage coopératif peut avantager l'apprentissage par situation-problème de nombreuses manières. Pour que la résolution du problème soit un succès, des rôles variés doivent être impliqués simultanément dans la résolution de la PBL : tel le résolveur qui suggère des solutions, le sceptique qui questionne et critique, l'instructeur qui explique des idées aux personnes les moins impliquées du groupe, celui qui conserve les informations récupérées ou encore le conciliateur qui résout les conflits. En assignant ces rôles à différents étudiants, les types de

réflexion impliqués dans la résolution du conflit sont différenciés, externalisés et rendus explicites. Les apprenants ont l'opportunité de pratiquer un rôle à la fois, puis de changer et d'en jouer un autre. Comme ils voient leurs collègues jouer d'autres rôles, les élèves ont des modèles de leurs pairs qui peuvent participer à les aider dans leur propre compréhension du problème. Le processus suivi par le groupe pour résoudre le problème reflète alors le processus que les étudiants doivent *internaliser* pour devenir des résolveurs du problème couronnés de succès [Col89]. Le partage des responsabilités et expertises peut rendre la complexité de la PBL plus réalisable. De même, le conflit dans un processus de groupe peut bénéficier au développement individuel de la pensée. Lorsque des différences surviennent, les étudiants doivent élaborer et justifier des points de vue. Ce processus permet alors de révéler toute mauvaise représentation mentale (*misconception*) existante, afin d'être ensuite défiée et, peut-être, corrigée. L'apprentissage coopératif conduit à une compréhension du problème plus riche puisqu'ayant de multiples perspectives.

2.3 Élaboration d'une situation-problème

Le travail de conception d'une situation-problème que nous désirons faciliter regroupe de multiples spécifications : spécification du problème, de la tâche ou encore spécification du scénario mettant en œuvre la situation-problème. Celles-ci interviennent à des moments différents du design pédagogique (expressions initiales des besoins, analyse ou alors conception avancée)

Dans cette section nous privilégions l'élaboration de la situation-problème indépendamment de l'environnement informatique qui la supportera ; l'étude technique de notre environnement informatique des plates-formes de formation à distance est reportée au chapitre 3. Notre étude concerne ici les phases amont de la conception identifiée au premier chapitre : *expression initiale des besoins* et *analyse*. Ces phases concernent principalement l'enseignant et/ou l'équipe pluridisciplinaire chargée de ces étapes. Nous nous intéressons particulièrement à ces étapes car elles s'appuient sur la théorie de l'apprentissage par situation-problème préalablement étudiée dans les sections précédentes.

L'objectif de cette section est d'étudier la phase d'élaboration d'une PBL pour extraire les besoins pédagogiques nécessaires facilitant le travail de l'enseignant. Ces besoins permettront de mieux appréhender les modèles que notre langage devra permettre de décrire : c'est-à-dire les *usages amont* du langage.

Les sous-sections suivantes sont construites sur la base des travaux de Meirieu sur l'élaboration d'une situation-problème [Mei02b].

2.3.1 Quel est l'objectif ?

Au coeur de la situation-problème se situe l'objectif du formateur. Le formateur se fixe un ou des objectifs d'apprentissage répondant à la question : qu'est-ce que je veux faire acquérir à l'apprenant qui représente pour lui un palier de progression important ?

2.3.2 Quelle tâche ?

Le formateur doit proposer une situation d'apprentissage dont la structure est en adéquation avec les objectifs d'apprentissage qu'il s'est fixé préalablement. Il doit s'assurer de l'existence d'un problème

à résoudre et de l'impossibilité de résoudre le problème sans apprendre. L'enseignant doit donc répondre à la question : quelle tâche puis-je proposer qui requiert, pour être menée à bien, l'accès à cet objectif (communication, reconstitution, énigme, réparation, résolution, etc.) ?

La structure de base d'une situation-problème est celle-ci : un sujet, en effectuant une tâche, est confronté à un obstacle. A ce stade de l'élaboration de la situation-problème le formateur définit donc la tâche qu'il proposera de poursuivre aux sujets. Cette tâche ne peut être menée à bien que si l'on surmonte un obstacle qui constitue le véritable objectif d'acquisition du formateur (obstacle=objectif). C'est grâce à un système de ressources que le sujet peut surmonter l'obstacle. Dans une situation-problème, l'objectif principal de formation se trouve donc dans l'obstacle que le sujet doit franchir et non dans la tâche qu'il réalise. Du point de vue de l'apprenant, la tâche reste longtemps la seule réalité perceptible : la tâche mobilise et oriente ses activités en lui donnant une représentation du but à atteindre.

L'apprenant a besoin de critères[17] avant d'engager la séquence d'apprentissage pour identifier la réussite de son travail : c'est son seul outil de régulation de son travail. Le véritable objectif ne lui sera identifiable qu'après la situation-problème. Une situation-problème se présente toujours à l'apprenant comme une tâche à effectuer. Néanmoins elle doit être construite par le formateur à partir de l'objectif d'acquisition qu'il s'est fixé. Ainsi, comme poursuite de tâche, la situation-problème doit faire l'objet d'une *analyse critériologique* et doit aboutir à l'élaboration d'une *fiche de tâche* ; comme poursuite d'un objectif, elle doit aboutir à l'explicitation de l'objectif en fin de séquence et à son évaluation individuelle systématique.

2.3.3 Comment l'apprenant peut-il surmonter l'obstacle ? Quelles ressources et consignes donner ?

Les pré-requis dont doivent disposer les apprenants ne garantissent en rien le franchissement de l'obstacle. C'est en procédant à des mises en relations, élimination, émission d'hypothèses, etc. que l'obstacle sera surmonté. Ces opérations mentales obéissent à cette caractéristique : « *elle est invisible pour celui qui la maîtrise et elle n'apparaît qu'en négatif à celui qui n'y parvient pas* » [Mei02b].

Le dispositif de la situation-problème doit donc être construit de manière à incarner l'opération mentale requise et permettre ainsi à ceux qui ne la maîtrisent pas encore de l'effectuer quand même. Les quatre grandes opérations mentales pouvant aider la structuration d'une situation-problème sont :

- la déduction : inférer une conséquence d'un fait, d'un principe ou d'une loi. C'est un raisonnement que l'on utilise comme moyen de vérifier et d'ajuster la plupart des acquisitions. Cette opération fait appel à ce que Piaget nomme « la décentration » [Pia78]. Il s'agit de cette faculté d' « objectiver » un instant ses actes afin d'en tirer des conséquences. C'est la situation du « SI je fais cela, ALORS il se produit cela ».
- l'induction : démarche mentale qui nous fait passer des exemples aux notions, des faits à la loi, de l'observation au concept. Le concept est un ensemble de choses ayant toutes des similarités. La conceptualisation permet à l'homme d'identifier plus facilement et rapidement les choses. Cette démarche fait donc encore appel à la faculté de *faire des hypothèses* mais dans un but différent de la déduction. Il faut ici constituer un point commun entre plusieurs éléments.

[17] Élement permettant au sujet de vérifier qu'il a bien réalisé la tâche proposée et que le produit de son activité est conforme à ce qu'il devait obtenir.

- la dialectique : si l'induction permet la formation des concepts, la dialectique permet de les placer les uns par rapport aux autres. Une connaissance fait toujours partie d'un ensemble complexe, et le dialectique est cette pensée qui chercher à comprendre la relation entre des notions différentes, afin de mieux les intégrer et les faire cohabiter dans notre esprit. La dialectique est donc la démarche intellectuelle cherchant à comprendre l'interaction des concepts.
- la divergence : opération mentale faisant appel à la faculté de création de l'esprit. D'après Meirieu la créativité « apparaît conditionnée par une opération mentale bien précise » qu'est la divergence. Elle se produit lorsque l'esprit doit confronter deux choses disparates, inhabituellement mises ensemble, et qu'il en produit une nouveauté.

Il s'agit en fait à ce niveau de réfléchir à ce qui se passe dans « la tête de l'élève » pour que l'obstacle soit franchi ; il faut instituer l'acte mental lui-même dans le dispositif de travail : pour surmonter l'obstacle, l'apprenant doit opposer, confronter, expérimenter, tirer des conséquences, faire face ou buter sur des concepts. Il s'agit alors de réfléchir aux matériaux et aux consignes qu'il faudra fournir pour que l'élève puisse y parvenir.

Il se peut à ce niveau que des contraintes matérielles (espace, temps, outils dont il dispose) ou institutionnelles (attentes des partenaires, représentation qu'ils ont d'une situation d'apprentissage) amènent le formateur à moduler le dispositif, ré-utiliser des matériaux déjà existants ou contraints, à privilégier des modalités de traitement plutôt individuelles etc. Si l'obstacle à franchir requiert la mise à l'épreuve de l'intelligibilité ou de l'efficacité sociale d'une production, on utilisera avec profit des groupes de confrontation ou de correction réciproque ; si l'obstacle requiert la construction d'un concept, on pourra procéder à des regroupements inductifs qui, faisant suite à l'étude par chacun des participants d'un exemple différent, permettra d'en dégager le point commun. Ces regroupements seront finalisés par une tâche (« production ») permettant de franchir un obstacle cognitif.

2.3.4 Structure directive mais traitement souple

Malgré les matériaux[18] de travail et les consignes-but[19], des stratégies variées et différentes peuvent être suivies par les différents apprenants : chaque sujet met en œuvre sa stratégie personnelle d'apprentissage. L'intérêt de la situation-problème tient donc dans le fait qu'elle associe une grande directivité structurelle et une grande souplesse dans le traitement individuel qui peut en être fait. Le rôle du formateur dans la résolution de la situation-problème (il y joue le rôle de tuteur) consiste davantage à aider les apprenants, à repérer les stratégies qui leur sont efficaces et aussi à les stabiliser.

2.3.5 La régulation

L'interaction sociale régulée par le tuteur (travail en petits groupes) permet d'espérer une plus grande efficacité grâce aux confrontations qu'elle suscite. Dans une situation-problème, les sujets mettent en œuvre des compétences et capacités qu'ils possèdent déjà pour en acquérir de nouvelles, grâce aux consignes et aux matériaux qui leur sont fournis. L'*évaluation diagnostique préliminaire* a pour but de

[18] Ensemble de documents, outils, ressources fournis par le formateur dans une situation didactique.

[19] Définition d'un projet à réaliser dans une situation didactique en termes de « produit fini » et renvoyant essentiellement au registre des motivations des apprenants.

diagnostiquer les pré-requis à la situation-problème. La qualité de cette évaluation est donc très importante. Ainsi, il faut garantir la possibilité d'effectuer la tâche et de surmonter l'obstacle en faisant jouer les consignes sur les matériaux, en mettant en œuvre des capacités et des compétences qui, en entrant en interaction, doivent permettre l'acquisition.

Une fois la résolution de la situation-problème lancée, l'évaluation porte différemment sur l'appréciation des processus utilisés par les apprenants : comment ils communiquent, progressent, formulent des hypothèses, tentent de résoudre le problème posé. Les interventions du tuteur recherchent moins souvent à aider la résolution du problème mais plutôt à souligner la structure du problème, à rappeler les consignes, à pointer les dévoiements du groupe, à proposer des activités intermédiaires, à soulager le travail par l'utilisation de supports facilitateurs. Cette évaluation sera *formative* si elle contribue à l'identification des procédures efficaces et à une formalisation suffisante de celles-ci pour en faciliter la réalisation : le processus est conjoncturel mais la procédure est reproductible. L'évaluation doit également porter sur l'acquisition elle-même : non plus la tâche mais l'objectif. Cette appropriation véritable nécessite une « décontextualisation » effectuée à travers un exercice différent : rédaction d'un rapport ou mode d'emploi, la confection d'un aide-mémoire ou encore fiche récapitulative. C'est une *évaluation sommative*.

2.3.6 Bilan

Nous venons d'étudier de nombreuses questions auxquelles l'équipe chargée de la conception de la PBL devra répondre (pendant l'expression initiale des besoins mais également en analyse). Notre objectif n'est pas d'en extraire une quelconque méthode mais plutôt d'en déduire les besoins en description qui pourraient être spécifiés par des modèles *via* notre langage.

Voici le résumé de ces besoins :
- définition des objectifs ;
- définition de la tâche authentique et de l'obstacle ;
- définition des rôles ;
- analyse critériologique ;
- étude du franchissement de l'obstacle ;
- prise en compte des ressources et contraintes matérielles ;
- définition d'une fiche de tâche par rôle ;
- explicitation des objectifs ;
- définition d'une fiche pour l'évaluation individuelle.

2.4 Présentation d'un cas d'étude d'apprentissage par situation-problème : SMASH

Notre travail est bâti autour d'une situation-problème particulière appelée SMASH. Elle sert de cas d'étude et d'expérimentation pour l'ensemble de nos travaux et propositions.

Notre cas d'étude a fait l'objet de multiples versions. La première version de SMASH était un produit acheté par l'équipe de recherche à la société anglaise *Actis Project Boxes* [ACT01]. SMASH a ensuite

évolué de diverses manières (ajout de la dimension coopérative, ajout de ressources - QCM, Document simplifié du code de la route, etc. -, structuration des activités plus précise, ou encore attribution de ressources selon les rôles). La version définitive est celle qui est présentée dans les sous-sections suivantes. Plus de détails sur l'évolution de SMASH sont indiqués dans [Oud03].

La présentation de SMASH décrite ci-après ne mentionne aucune référence aux plates-formes de formation à distance nous intéressant ni aucun autre environnement informatique pouvant médiatiser la situation-problème. Les aspects techniques de la réalisation de SMASH seront abordés dans le chapitre suivant.

2.4.1 Caractéristiques principales de SMASH

Nous présentons la PBL SMASH vis-à-vis des concepts et caractéristiques aux situations problèmes coopératives étudiés précédemment.

Sujets. SMASH s'adresse aux enfants de 8-10 ans du cycle 3. En cela, SMASH n'est pas une PBL dans laquelle le processus d'apprentissage sera entièrement auto-dirigé. Bien que centré sur l'apprentissage réalisé par les apprenants, le cas d'étude SMASH implique une structure directive moins souple que s'il s'agissait d'une PBL pour adultes comme celles réalisées à l'origine pour les étudiants en médecine.

Objectif d'apprentissage. Il s'agit d'initier les enfants au code de la route pour leur faire appréhender la sécurité routière pour les voitures comme pour les cyclistes et les piétons. Les enfants sont encouragés à réfléchir sur leur sécurité et leurs responsabilités vis-à-vis d'eux-mêmes et des autres.

De nombreux objectifs disciplinaires sont concernés :

- découvrir les dangers de la rue et les prendre en compte pour sa conduite (à pied ou à bicyclette, seul ou non),
- prévenir les autres,
- apprendre à connaître ses possibilités et ses limites,
- évaluer les risques,
- apprendre à prévoir les mouvements des autres et anticiper leurs actions,
- apprendre à respecter les autres,
- connaître le code de la route et comprendre son utilité,
- découvrir les codes graphiques des signalisations verticales et horizontales,
- savoir lire une carte et travailler avec un plan,
- savoir se repérer sur une carte,
- connaître et savoir entretenir son vélo,
- adapter sa tenue vestimentaire à vélo.

De même les principaux objectifs pédagogiques relevés sont :

- lecture,
- compréhension,
- écrit : synthèse, écriture,
- oral : énonciation, verbalisation des synthèses et justification des résultats obtenus,
- parler et écouter en communauté,
- justifier une action,

- expliciter une démarche,
- participer à la recherche,
- émettre des hypothèses.

Ces différents objectifs pédagogiques et disciplinaires sont issus du programme officiel pour le cours moyen selon les recommandations du Ministère de l'éducation Nationale.

Mise en situation authentique - le contexte. Un accident de vélo s'est produit dans le quartier du village *Mille-Fleurs*. Ce contexte est une situation authentique car le cycliste renversé dans la situation est un garçon du même âge que les élèves. Ceci permet de faciliter la contextualisation dans la situation : les enfants se sentent concernés par cette histoire et ainsi s'investissent davantage dans la résolution de la tâche.

Mise en situation authentique - la tâche. Il s'agit de mener une enquête sur cet accident. Les objectifs donnés aux élèves sont de reconstituer l'accident et de trouver le coupable. La tâche est authentique car elle se base sur les enquêtes policières. L'authenticité se révélera également dans la difficulté d'analyse des témoignages : indices à extraire, informations essentielles ou superflues, recoupements difficiles entre témoignages incomplets, etc.).

Mise en situation authentique - les rôles authentiques. Les apprenants jouent le rôle d'inspecteurs de police, tandis que le tuteur joue le rôle du chef de la police. Les rôles sont assignés pour la totalité de la tâche et correspondent à des métiers authentiques.

Les ressources ou matériels. Les apprenants ont à leur disposition des témoignages relevés après l'accident. Ces témoignages concernent des acteurs passifs (un piéton ou encore un conducteur arrêté à un stop) comme des acteurs impliqués dans l'accident (le conducteur de la voiture qui a percuté le vélo, le propriétaire de la voiture mal garée,... [20]). Il y a neuf témoignages. A chacun d'entre eux correspond un QCM.

Les autres ressources mises à disposition des apprenants sont : une carte complète du village des Mille-Fleurs ; un gros plan du quartier concerné par l'accident ; une fiche bilan du code de la route ; des feuilles de notes vierges ; des personnages et panneaux à positionner sur les cartes.

L'obstacle. Dans SMASH, l'obstacle rencontré par les apprenants doit être l'aboutissement à des conclusions différentes sur la reconstitution de l'accident. Après avoir confronté leurs divergences de résultats les élèves doivent surmonter l'obstacle en intégrant les autres perspectives de l'accident qu'ils ne connaissaient pas. Pour parvenir à une solution commune, ils devront donc étudier davantage le code de la route, apprendre à évaluer les risques entrepris par le cycliste et d'autres conducteurs vis-à-vis de leur vitesse, leur comportement sur la route, leur prise en compte de la météo, le respect d'autrui, etc.

Le système de contraintes. Afin de garantir la rencontre de l'obstacle et donc de garantir l'apprentissage, les différents inspecteurs joués par les apprenants doivent aboutir à des conclusions différentes. Pour cela une première contrainte, relevée par l'enseignant lors de l'élaboration de SMASH, consiste à créer et à partager les témoignages entre les inspecteurs de manière à ce que chaque inspecteur aboutisse à une conclusion différente sur les circonstances de l'accident de vélo. La

[20] Le cycliste percuté n'apporte aucun témoignage ; ceci permet de complexifier la reconstitution de l'accident car les autres témoignages sont incomplets et non objectifs. Pour anecdote, l'histoire ne dit pas si le cycliste a survécu.

deuxième contrainte est le nombre limité (trois) de témoignages par inspecteur. Ce découpage des ressources rappelle en partie la méthode d'apprentissage coopératif du *jigsaw*[21].

Les pré-requis. Les objectifs d'apprentissage fixés dans SMASH (sécurité routière ou encore responsabilisation) sont tirés du programme officiel délivré aux enseignants de cours moyen. Ainsi, les pré-requis pour les apprenants sont implicitement considérés comme présents.

SMASH fait également appel aux compétences transversales suivantes :

Parler et écouter : les élèves sont impliqués dans la discussion de groupe et l'interaction. SMASH est conçu pour permettre aux enfants de prendre librement de nombreux rôles à l'intérieur du groupe et de préparer un discours formel. On attend des enfants qu'ils écoutent, comprennent et répondent convenablement aux autres lors des débats.

Lire : tout au long de SMASH, on demande aux enfants de lire les messages, d'utiliser l'inférence et la déduction pour comprendre un texte. Les enfants utilisent leurs aptitudes à récolter des informations et à marquer la différence entre ce que les témoins disent et ce qui s'est passé.

Écrire : pendant SMASH, les enfants remplissent les feuilles de notes après chaque travail sur un témoignage et ils rédigent leur conclusion sous la forme d'un compte rendu. Grâce à cette activité, ils sont obligés de mettre en avant leurs compétences de grammaire, de conjugaison et d'orthographe ainsi que leur talent de rédacteur.

Apprendre : en plus de toutes ces aptitudes, les enfants ont également besoin de connaissances en matière de physique, de code de la route, de comportements en groupe. Par exemple si les véhicules vont vite, les dégâts sont plus importants que s'ils se déplacent plus lentement. Cela rappelle des principes élémentaires de l'énergie cinétique et de l'inertie. Une voiture doit céder le passage à un vélo qui vient de sa droite. C'est une caractéristique du code de la route.

Il n'y a pas une « bonne solution » à SMASH, d'ailleurs la vérité sur l'accident importe peu : tous les acteurs actifs impliqués dans l'accident ont un degré variable de responsabilité.

2.4.2 Autres caractéristiques

Les caractéristiques principales que nous venons de définir pour SMASH sont également accompagnées des particularités suivantes.

2.4.2.1 Répartition plus précise des rôles authentiques des apprenants

Les apprenants sont regroupés en quatre groupe distincts (le groupe est formé d'un petit nombre d'élèves : 3 à 4 élèves). A chaque groupe correspond un rôle d'inspecteurs de police différent[22] :

[21] http ://www.jigsaw.org/

[22] Un groupe correspond à un rôle et inversement. Le fonctionnement interne au groupe pour jouer le rôle du groupe n'est pas considéré dans la construction de SMASH. Les élèves ont à leur charge le fonctionnement de leur groupe : travail collaboratif, partage du travail (coopératif), etc. Ainsi, la notion de groupe disparaît au profit de celui du rôle joué.

- les rôles d'inspecteurs 1 à 3 doivent analyser les témoignages, déduire les informations clés et reconstituer l'accident pour déduire le responsable. Les témoignages sont différents pour chaque groupe ; ils doivent donc arriver à des conclusions différentes (les témoignages sont construits de manière à diriger indirectement les conclusions). Ce découpage renforce l'authenticité de la situation : dans la réalité, les inspecteurs se partagent les témoignages dans une même enquête pour faciliter leur travail.
- le rôle 4 d'inspecteur est plus particulier : il s'agit d'une sorte d'expert en cartographie dont le rôle principal est d'aider les autres inspecteurs à situer les témoins, les véhicules présents, les panneaux et autres signalisations de la route, marquer les directions et vitesse de déplacement des véhicules, etc. Cet expert partage alors certaines activités avec les autres enquêteurs : il est celui qui aura la perspective la plus générale et donc la plus complète de l'accident.

Les ressources décrites précédemment sont donc allouées en conformité avec ce raffinement des rôles.

2.4.2.2 Apprentissage coopératif

Les différentes caractéristiques d'un apprentissage coopératif se retrouvent dans SMASH : les rôles d'inspecteurs sont assignés dès le début de SMASH ; le contrôle de l'enseignant est important (il joue le rôle de co-investigateur par le biais de son rôle *authentique* de chef de la police) ; des aptitudes sociales sont également visées au travers de la résolution de SMASH (écouter et parler en communauté...) ; les activités à réaliser par les apprenants sont fortement structurées (le déroulement est donné ci-après).

Le partage des responsabilités et des expertises de SMASH complexifie sa réalisation. La présentation des conclusions de chaque rôle doit aboutir à un conflit puisque les conclusions doivent diverger. Au travers de l'élaboration et de la justification de ses points de vue différents les élèves aboutissent à une compréhension plus riche du problème et de l'obstacle. La résolution commune du problème qui s'en suit doit dépasser le conflit d'opinion initial et aboutir à une conclusion globale. Celle-ci résulte alors du recoupement des différentes conclusions, de la correction des mauvaises représentations mentales liées aux conclusions incomplètes. L'apprentissage est coopératif dans SMASH et se déroule par une succession d'activités individuelles coordonnées entre groupes, puis par une activité de résolution collective.

L'approche coopérative utilisée dans SMASH fait référence à l'approche de la cognition distribuée qui propose de voir l'apprentissage collectif sous l'angle de la communauté de pratique, c'est-à-dire dans une vision de culture professionnelle [Kos96]. Succinctement, cette théorie explique l'apprentissage comme une activité faisant participer l'apprenant à un monde réel, comme un processus d'enculturation et souligne la nécessité d'un contexte social et matériel authentique pour l'apprentissage [Dae02].

2.4.2.3 Déroulement

Le déroulement de SMASH suit une structure directive en quatre phases principales :

Présentation : pour introduire la situation problème, le tuteur va lancer une discussion avec la classe. Par exemple, il peut demander aux élèves d'énumérer les panneaux de signalisation qu'ils connaissent, de donner (selon eux) les causes possibles d'un accident, puis il peut leur présenter la carte, et leur demander de situer cinq lieux dangereux. Tout ceci va susciter des questions de la part des élèves, ce qui va permettre au tuteur de présenter le contexte de SMASH, de présenter les rôles qui seront joués par les élèves et d'organiser la classe.

Production : les différents groupes exploitent les témoignages, carte et code de la route selon leur rôle. Les inspecteurs 1 à 3 prennent des notes sur leur feuille de notes pendant l'analyse de chaque témoignage, et répondent au QCM associé. Ensuite, il passe à un autre témoignage. L'inspecteur 4 « expert » étudie les cartes du village et les ressources liées au code de la route avec précision. Il participe brièvement à l'analyse des témoignages des autres inspecteurs en proposant son expertise pour aider à localiser et analyser « géographiquement » les acteurs présents lors de l'accident.

En reprenant leurs feuilles de notes, chaque groupe va faire la synthèse de ses résultats, et préparer par écrit une présentation de ses conclusions. Leur conclusion doit être présentée clairement ; ceci nécessite de détailler : a/où étaient les gens et ce qu'ils ont vu ; b/où étaient les véhicules et comment ils se déplaçaient ; c/qui est responsable aux yeux des élèves et pourquoi ; d/quels sont les endroits dangereux ; e/leurs suggestions pour améliorer la sécurité du lieu. Finalement, leur présentation doit montrer au chef de police et aux autres investigateurs de l'accident ce qui s'est produit en utilisant leur plan (rempli préalablement à l'aide de l'inspecteur expert) et leurs explications.

Présentation individuelle et production d'une solution commune : les présentions précédentes sont réalisées une à une aux autres groupes et au chef de la police. Ensuite, chaque groupe doit répondre et commenter les questions que les autres groupes lui posent. L'ordre de passage peut être établi par le tuteur de manière à faire passer en premier ceux dont les conclusions sont les plus incomplètes (déduction faite en analysant leurs différentes productions) pour finir ensuite par ceux ayant la vue la complète.

Après, le passage des 4 groupes, tous les inspecteurs doivent rassembler toutes leurs ressources et proposer une conclusion globale prenant en compte toutes les perspectives mises en valeur pendant les présentations.

Correction : le tuteur corrige les QCM et donc les informations importantes qu'il fallait extraire des témoignages ; il présente une perspective complète reconstituant l'accident de vélo. Les différentes responsabilités des acteurs sont abordées. Il fait ensuite un bilan sur la sécurité routière et présente les différentes compétences disciplinaires abordées par chaque groupe dans la résolution du problème. Le tuteur discute avec la classe sur le déroulement de l'activité (points positifs, points négatifs) : il questionne les élèves sur leur vision du déroulement du travail en groupe, la coopération, les aspects relationnels entre les apprenants, les temps morts, sur les conclusions auxquelles les élèves pouvaient ou ne pouvaient pas aboutir en fonction de la répartition des témoignages, etc.

2.4.2.4 Décontextualisation, extensions et perspectives de SMASH

L'extension de SMASH doit encourager les enfants à être responsables de leur propre sécurité et donc de leur comportement. Le tuteur va proposer différents exercices de renforcement (décontextualisation[23] et mentalisation[24]) visant à ouvrir le sujet sur d'autres thèmes comme par exemple :

- l'étude du code de la route pour les vélos, pour les automobiles,
- des sujets scientifiques comme la notion de masse*vitesse,
- de l'art plastique : vêtement de protection (couleur, rembourrage),

[23]Opération par laquelle un sujet utilise une acquisition dans un autre contexte que celui qui en a permis l'apprentissage.

[24]Opération par laquelle un sujet se représente une acquisition en l'absence de tout élément matériel ayant servi ou pouvant servir à son acquisition.

– des travaux d'écriture journalistique.

2.4.2.5 Les différentes évaluations dans SMASH

L'évaluation *diagnostique préliminaire* de SMASH a été abordée lors de la description des pré-requis de SMASH.

L'évaluation de SMASH en tant qu'évaluation des apprenants n'a pas été envisagée. Il s'agit plutôt d'un exercice particulier réalisé par les élèves sous la forme d'un apprentissage par situation-problème coopérative. Les rôles et tâches étant différents il serait injuste de noter les élèves. Les activités de décontextualisation ou de renforcement à partir de SMASH sont davantage susceptibles d'être notées.

Les ressources produites par les apprenants pendant la réalisation de SMASH concernent plutôt la régulation de l'apprentissage, c'est-à-dire aider le tuteur à guider l'apprentissage et la résolution de la tâche. Les ressources écrites (QCM répondus, feuilles de notes et cartes remplies, brouillons de préparation de la présentation des conclusions) permettent de suivre les progrès et les découvertes des élèves. Le tuteur peut intervenir grâce à ces indications. De même, le tuteur/chef de police guide les présentations, tour de parole, propositions de questions, etc.

La démarche ou le processus suivi par les élèves dans leur analyse des témoignages et la construction de connaissances pour la reconstitution peut être évalué de manière *formative* par une activité de décontextualisation consacrée à ré-appliquer la démarche pour un autre contexte.

Concernant l'évaluation *sommative* des connaissances acquises dans SMASH (code de la route, comportement sur la route, etc), celle-ci peut faire l'objet également d'une activité de décontextualisation comme par exemple un QCM général sur le code de la route.

2.4.2.6 Expérimentation de SMASH en classe « traditionnelle »

SMASH a pu être expérimenté dans une salle de classe de manière « traditionnelle » en présentiel, c'est-à-dire sans utilisation du médium informatique [Oud03]. Pendant quatre demi-journées le déroulement de notre PBL a pu être observé (appareil photo et caméra à l'appui ; Figure 2.3).

Cette expérimentation a permis de vérifier notre cas d'étude dans la pratique de manière à pouvoir l'ajuster avant d'envisager de le transposer à distance en médiatisant SMASH grâce aux plates-formes de formation à distance.

2.4.2.7 Vers une transposition de SMASH à distance

La mise en œuvre de SMASH pour une réalisation à distance comporte de nombreux problèmes : les ressources ou documents, les activités, les objectifs, les outils, etc. doivent être reformulés pour un contexte d'utilisation à distance et médiatisés par l'outil informatique.

La transposition didactique liée à l'élaboration de la PBL SMASH doit ainsi être suivie d'une transposition informatique. La transposition informatique désigne le transfert de la connaissance en une représentation symbolique et la mise en oeuvre de cette représentation par un dispositif informatique, qu'il s'agisse ensuite de montrer la connaissance ou de la manipuler [Bal91].

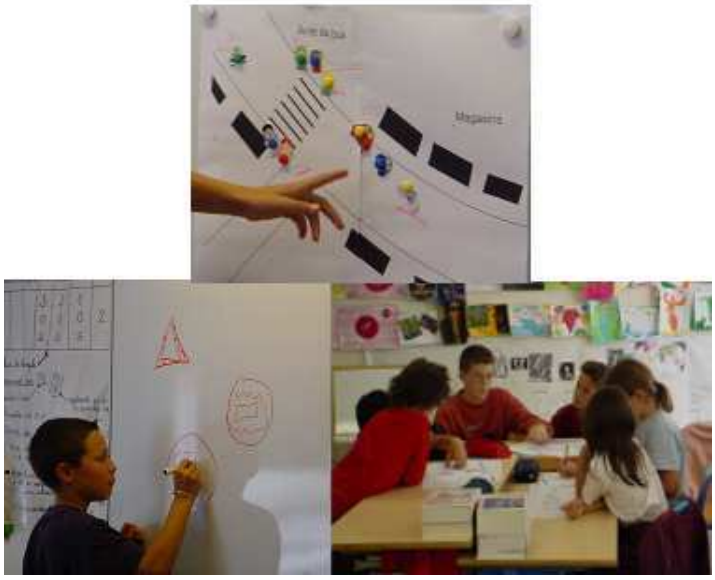

FIG. 2.3 – Expérimentation de SMASH en salle de classe.

2.5 Conclusions

Dans ce chapitre nous avons tout d'abord défini et présenté l'apprentissage par situation-problème coopérative, le modèle théorique d'apprentissage à la base de notre contexte de recherche (sections 2.1, 2.2). Nous avons ensuite consacré une section (2.3) à l'analyse de la démarche suivie par l'enseignant au début du processus de conception nous intéressant : l'élaboration d'une situation-problème. Ceci nous a permis de mettre en évidence les différentes questions que l'équipe de conception se doit de répondre en amont de la conception : phases d'*expression initiale des besoins* et d'*analyse*.

Notre langage devra supporter les différents concepts manipulés dans toute PBL (tâche, rôle, obstacle, ressources, etc.) ; il devra également faciliter la mise au point de modèles de conception supportant la phase d'élaboration de PBL. L'aspect coopératif de l'apprentissage deva également être supporté par notre langage.

Enfin, nous avons présenté notre cas d'étude de situation-problème coopérative : SMASH (section 2.4). Celui-ci a pu être expérimenté et validé dans des conditions traditionnelles d'apprentissage.

L'apprentissage par ordinateur n'a pas été étudié dans ce chapitre. Toutefois, notre contexte de recherche concerne également la réalisation de PBL sur des environnements informatiques d'apprentissage à distance de type plates-formes. L'étude technique et technologique de cette deuxième partie de notre cadre d'étude est réalisée dans le chapitre suivant.

Chapitre 3

Cadre technique : environnements informatiques, plates-formes et composants pour la e-formation

Sommaire

Dans ce chapitre, nous présentons la deuxième partie de notre contexte de travail sur les EIAH. Notre second axe de recherche concerne l'étude des plates-formes de formation à distance comme environnement support à nos PBL coopératives.

Notre objectif global est de proposer un langage adapté à la conception de situations d'apprentissage. Nous avons vu que cette conception se déroule en plusieurs étapes : *l'expression initiale des besoins*, et l'*analyse et conception*. La première étape concerne l'enseignant et son équipe ; cette étape a été

étudiée dans le chapitre précédent consacré aux PBL (2.3). La seconde concerne davantage un ingénieur pédagogique et son équipe (fournisseur de ressources pédagogiques et développeur de composants) en collaboration avec l'enseignant. L'ingénieur pédagogique et son équipe doivent connaître les particularités des plates-formes en général et surtout de la plate-forme cible afin que les modèles de conception avancée spécifient au mieux la situation d'apprentissage dans son articulation avec le dispositif informatique. Ce chapitre est donc consacré à étudier ces environnements informatiques particuliers et concerne autant la phase d'implémentation que la phase précédente de conception avancée du processus itératif identifié en 1.2.2.2.

Le chapitre se décompose en trois sections. La première section (3.1) s'intéresse aux environnements informatiques existants actuellement pour supporter les PBL. Cette étude préliminaire va nous permettre d'établir les critères fonctionnels assurant un « bon » environnement support pour les PBL. Ces critères permettront par la suite de cadrer l'étude des plates-formes. La seconde section (3.2) présente le cadre général d'utilisation des plates-formes. Après quelques définitions, les plates-formes sont présentées plus en détail. Ensuite, nous justifions en quoi ce type d'environnement informatique particulier nous intéresse dans le cadre des situations d'apprentissages spécifiques que sont les PBL. Cette seconde section s'intéresse également aux normes et standards régissant le contexte de l'apprentissage à distance. Ces travaux internationaux concernent à la fois les environnements informatiques ou technologiques pour la mise en œuvre d'enseignement et d'apprentissage à distance mais également les descriptions des enseignements/apprentissages. Enfin, la troisième section étudie le concept de composants éducatifs. Ce concept est dans la littérature étroitement lié aux différents outils de formation étudiés dans la première partie. Ces concepts sont également présents, sous des appellations différentes, dans les normes et standards étudiés dans la seconde section.

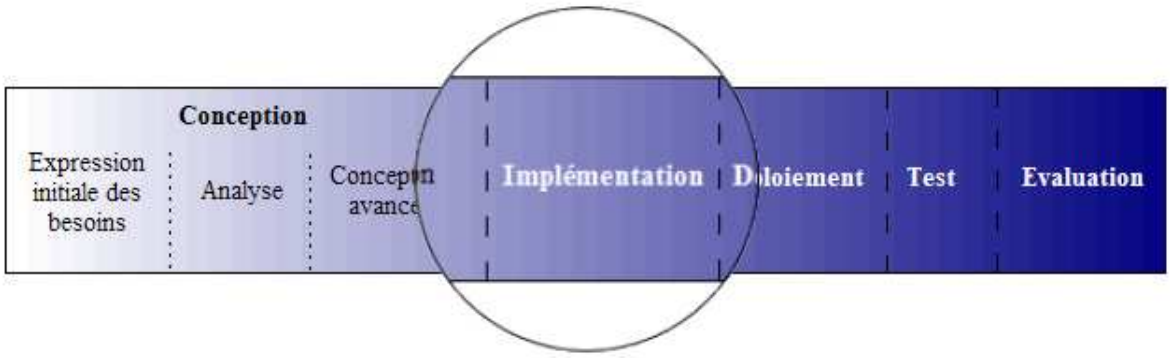

FIG. 3.1 – L'étude des plates-formes concerne les phases de conception avancée et d'implémentation.

3.1 Environnements d'exécution existants pour les PBL

Avant de nous intéresser aux plates-formes de formation à distance comme dispositif informatique pour les PBL, nous nous intéressons préalablement aux environnements informatiques existants et reconnus pour l'exécution de PBL.

3.1.1 CSCW et hypertextes/hypermédias

Depuis quelques années, de nombreuses avancées technologiques telles que le développement des hypertextes/hypermédias, des technologies pour le travail coopératif assisté par ordinateur (TCAO ou en anglais CSCW pour *Computer-Supported Cooperative Work*), et pour l'apprentissage coopératif assisté par ordinateur (en anglais CSCL pour *Computer-Supported Cooperative Learning*) ont permis la mise au point de nouveaux environnements informatiques pour l'apprentissage [25].

Pour la communauté CSCL, ces technologies sont souvent utilisées pour supporter le partage d'informations. La technologie du CSCW permet souvent d'enlever les contraintes géographiques ou temporelles, liées aux interactions en face-à-face de l'apprentissage traditionnel, en proposant des classes « virtuelles ». La technologie des hypertextes/hypermedia est en revanche souvent utilisée pour surmonter la linéarité ou séquence des structures d'informations en permettant l'intégration non linéaires d'informations pouvant être représentées par différentes formes de média : texte, image, vidéo et audio. Ces technologies peuvent potentiellement fournir un support pour le développement d'environnements d'apprentissage coopératif/collaboratif avancés. Par exemple, la technologie CSCW peut être utilisée pour développer des mécanismes informatiques guidant et contrôlant le comportement des personnes jouant des rôles différents. Les hypertextes peuvent être utilisés pour représenter hiérarchiquement un ensemble de pages structurées et connectées. L'exploitation des technologies doit être dirigée par les théories d'apprentissage pour être efficace.

3.1.2 Environnements existants pour les PBL

Il existe de nombreux environnements informatiques supportant l'enseignement et l'apprentissage. Une grande partie d'entre eux réalise une approche centrée-enseignant : ils permettent aux apprenants d'accéder à des informations en ligne déjà préparées en parcourant ou suivant une structure prédéfinie de l'information (exemples : *CBT*[26], *WBT*[27]). D'autres systèmes utilisent les technologies issues de l'intelligence artificielle pour proposer des agents jouant le rôle des enseignants, experts, ou tuteurs dans un enseignement (les *ITS*[28]). La plupart de ces systèmes ne conviennent pas pour une approche centrée-apprenant et donc ne permettent pas de supporter les PBL.

Toutefois, la littérature fait référence à de nombreux dispositifs informatiques issus du CSCL/CSCW supportant l'apprentissage coopératif/collaboratif et utilisant une approche centré-apprenant : CCL, CSILE, WebCSILE, CALE, CNB, Belvedere, McBagel, Web-SMILE, CROCODILE, STEP. Nous ne détaillons pas individuellement ces systèmes ; de nombreuses études ont porté sur ces environnements pour les PBL [Alb00, Nai96, Ros00, Ste02b]. Nous reprenons toutefois, dans la section suivante, un ensemble de pré-requis tirés d'un travail de synthèse des précédents dispositifs informatiques existants pour les PBL.

[25] La comparaison et l'étude des deux domaines CSCW et CSCL est détaillée dans le chapitre 2 de [Geo01].
[26] *Computer-Based Training*
[27] *Web-Based Training*
[28] *Intelligent Tutoring System*

3.1.3 Conditions requises pour la bonne mise en oeuvre de PBL dans un environnement informatique

Miao recense dans ses travaux des exigences d'implémentation pour un environnement informatique support à l'apprentissage par situation-problème [Mia00].

1. (R1.1) : support de l'orientation sociale. Permettre à chaque utilisateur de connaître sa position dans le scénario, où il peut aller, où il peut rejoindre ses collègues, où il peut acquérir de nouvelles informations, où il peut utiliser des outils.
2. (R1.2) : support de la conscience de groupe (*awareness*). Permettre à chaque utilisateur de se présenter aux autres et de connaître les autres utilisateurs et ce qu'ils font.
3. (R1.3) : support de riches formes d'interactions sociales. Permettre aux utilisateurs d'interagir entre eux sous des formes de coopération à des lieux/moments différents/identiques.
4. (R1.4) : support de la personnalisation des environments d'apprentissage. Permettre aux utilisateurs de personnaliser leur environnement d'apprentissage pour réaliser différents types de tâche [29].
5. (R2.1) : support de la représentation de formes variées d'idées. Permettre aux utilisateurs de représenter des formes variées d'informations ainsi que leurs intentions.
6. (R2.2) : support de la représentation des relations entre ces idées.
7. (R2.3) : support de la mise à disposition d'informations de référence. Permettre aux utilisateurs de lier une information à une autre information relative ou plus détaillée.
8. (R2.4) : support de la négociation des connaissances partagées.
9. (R3.1) : support de la définition de rôles (et par extension des responsabilités ou obligations attribuées).
10. (R3.2) : mise à disposition d'aide/directives selon les stratégies de la PBL. Il s'agit de guider les apprenants et les tuteurs dans le suivi des activités de la PBL en accord avec la stratégie pour laquelle des situations distinctes et leurs transitions sont spécifiées.
11. (R3.3) : support de la synchronisation des activités collaboratives. Ceci doit permettre aux utilisateurs de toujours réaliser leur tâche principale et d'éviter des comportements inattendus.
12. (R3.4) : support du changement entre différentes stratégies de PBL. Permettre aux utilisateurs de définir leurs propres stratégies d'apprentissage et de pouvoir en changer.
13. (R4.1) : support de la définition de plan d'action. Permettre aux utilisateurs de définir des actions et leurs relations.
14. (R4.2) : support de l'allocation de ressources.
15. (R4.3) : décharge de la difficulté des apprenants à réaliser leurs plans d'actions.
16. (R4.4) : support de l'exécution des plans d'actions définis par l'utilisateur.

Les systèmes informatiques existants qui supportent les PBL ne répondent pas à l'ensemble de ces critères mais proposent chacun des fonctionnalités correspondant à un sous-ensemble de ces critères. De

[29] Ceci correspond à la notion générale de *malléabilité*, « moyens proposés aux utilisateurs pour modifier l'environnement informatique dans son contexte d'utilisation en tant que l'une de ses fonctionnalités », étudiées dans [Bou00, Bet03].

manière générale, tous ces environnements issus des travaux de recherche en CSCW et CSCL ne sont que des prototypes développés dans des laboratoires de recherche, ce qui sous-entend une non-diffusion de ces systèmes hors du laboratoire et une évolution réduite.

Les différents critères établis précédemment nous permettront d'étudier les usages des plates-formes de formation à distance pour la mise en œuvre de PBL (section 3.2.2.7).

3.2 Apprentissage à distance : vers de nouveaux dispositifs

Les EIAH ont conduit, avec l'émergence des Technologies de l'Information et de la Communication appliquées à l'Éducation (TICE) et l'apparition des grands réseaux d'information, à la naissance de la e-formation (*e-learning* en anglais).

La formation à distance est un cadre d'utilisation particulier des situations d'apprentissage au centre de nos préoccupations de recherche. Un dispositif privilégié de la formation à distance est la plate-forme. Comme nous le verrons dans les sections suivantes, ce type d'environnement d'exécution semble peu disposé à prendre en charge la mise en œuvre et la réalisation d'apprentissage centré sur l'activité de l'apprenant comme dans les situations problèmes. Toutefois, une reflexion actuelle de la communauté de recherche internationale ainsi que des industries et organisations impliquées dans la réalisation de ces plates-formes vise à faire évoluer les fonctionnalités et les usages de ces plates-formes pour de nombreux modèles pédagogiques d'enseignement/apprentissage ; l'évolution tend vers la conception et le développement de nouveaux systèmes informatiques (plates-formes) *ouverts*, c'est-à-dire aux fonctionnalités extensibles. Comme toute évolution, la tendance actuelle est dirigée par les avancées technologiques et particulièrement celles des systèmes distribuées sur le Web. Ainsi, des plates-formes basées composants sont actuellement proposées. Notre réflexion porte alors essentiellement sur une proposition visant à guider l'extension des fonctionnalités de la plate-forme pour qu'elle permette de médiatiser les PBL.

Dans cette section, une première partie présente des généralités sur la formation à distance. Ensuite nous présentons le dispositif particulier de ce mode d'apprentissage : la plate-forme de formation à distance. Nous proposons par la suite une étude plus détaillée de ces dispositifs : aspects fonctionnels, architectures et usages. Devant la diversité des plates-formes, il est apparu indispensable de s'acheminer vers une normalisation de la e-formation. Nous abordons alors les travaux de normalisation entrepris au niveau international.

3.2.1 Apprentissage à distance

Cette sous-section est consacrée à une brève introduction du cadre d'utilisation des plates-formes qui nous intéressent. E-formation ou *e-learning*, formation à distance, formation ouverte et à distance et téléformation sont de nombreux termes à la signification proche souvent associés aux plates-formes comme cadre d'utilisation. Nous tentons de les définir et de les caractériser plus clairement.

Formation ouverte et à distance. (FOAD)[30] Voici une définition du Collectif de Chasseneuil [Col00] :

« La formation ouverte[31] *et à distance...*

[30] En anglais ODL (Open and Distance Learning).

[31] Une formation est dite « ouverte » lorsqu'il n'y a pas de condition d'accès autre que technique.

- *est un dispositif organisé, finalisé, reconnu comme tel par les acteurs ;*
- *qui prend en compte la singularité des personnes dans leurs dimensions individuelle et collective ;*
- *et repose sur des situations d'apprentissage complémentaires et plurielles en termes de temps, de lieux, de médiations pédagogiques humaines et technologiques, et de ressources. »*

. D'autres travaux détaillent la définition de FOAD (par exemple [FFF02]).

Formation à distance (FAD). Il existe peu de définitions dans la littérature [Des96]. Nous retenons toutefois celle-ci [Kee96] dans sa version traduite [Mba03] :

La formation à distance est une forme d'enseignement caractérisée par :

- la séparation quasi-permanente entre le formateur et l'apprenant tout au long du processus d'apprentissage (ceci différencie la formation à distance de la formation présentielle) ;
- l'influence d'une organisation administrative aussi bien en ce qui concerne la planification et la préparation des matériaux pédagogiques que la mise à la disposition des apprenants des services d'accompagnement et de support (ceci différencie la FAD de l'autoformation) ;
- l'utilisation de média techniques (imprimerie, audio, vidéo, ordinateurs) pour assurer le lien entre le formateur et l'apprenant et médiatiser le contenu de la formation ;
- l'existence de mécanismes de communication bidirectionnelles afin que l'apprenant bénéficie mais prenne aussi l'initiative de dialogues avec le formateur (ceci distingue la FAD des autres usages de la technologie dans l'éducation) ;
- la quasi-absence de la notion de groupe tout au long du processus d'apprentissage, de sorte que les apprenants sont toujours vus comme des individus isolés et non comme faisant partie d'un groupe, avec la possibilité d'organiser occasionnellement des rencontres, soit en présentiel [32], soit via des moyens électroniques à des fins didactiques ou de socialisation.

Pour simplifier, on peut considérer la FAD comme incluse dans la FOAD puisque la FAD repose sur la non-présence physique du formateur [FFF02]. Les termes de *e-learning*, e-formation et téléformation font référence à ces deux définitions dans un contexte d'utilisation des TICE (Technologies de l'Information et de la Communication appliquées à l'Education). Il est possible de schématiser ainsi les autres définitions :

$$\text{e-formation/e-learning} = \text{FOAD} + \text{TICE}$$
$$\text{téléformation} = \text{FAD} + \text{TICE}$$

Le *e-learning* (ou **e-formation** en français) est l'utilisation des TICE afin de transmettre ou de suivre à distance une formation complète ou une partie de celle-ci ; « *dispositif de formation faisant une large place à Internet ou à des intranets* » [Bel01]. Voici une autre définition extraite d'un rapport pour la standardisation du e-learning : « *learning or training that is prepared, delivered, or managed using a variety of learning technologies and which be deployed either locally or globally* » [Cen03]. La e-formation se décline selon différents scénarii compris entre celui de la transmission de contenu asynchrone lié ou non à une formation présentielle et celui d'une transmission de contenu et d'une interaction effectuées entièrement à distance entre le formateur et les apprenants. Elle est basée sur des technologies fiables, mais est orientée vers la pédagogie. Elle repose sur deux composantes : l'ingénierie pédagogique, qui désigne la démarche pédagogique adoptée dans la conception et la diffusion des cours, et la solution technologique, qui s'appuie sur des plates-formes aux fonctionnalités variées [Bus03]. Elle vise à concilier économie (réduction des frais indirects), réponse

[32] *En présentiel* fait référence au face-à-face traditionnel.

aux contraintes (individualisation de la formation), efficacité (rapidité de déploiement à grande échelle) et capitalisation (la gestion centrale et systématique du capital des compétences dans le contexte des entreprises [Bus03]). Ainsi, les concepteurs en ligne recherchent une plus large et plus efficace distribution de la formation, l'individualisation et l'adaptation de parcours de formations aux besoins individuels des apprenants, des possibilités de tutorat, de suivi et de démultiplication de l'offre grâce à la mise en commun de ressources, de modules de formation, de tutorat [Mba03].

La e-formation permet donc ces différentes formules :

- l'autoformation individuelle : le contenu des cours est disponible en ligne, en libre service, à n'importe quel moment.
- la formation individuelle en ligne avec tutorat asynchrone : le travail de l'apprenant est suivi par un tuteur ; ce dernier répond à ses questions et contrôle sa progression en différé dans un délai très court.
- les classes virtuelles en ligne avec tutorat synchrone : les apprenants se retrouvent en ligne pendant une plage horaire déterminée pour un séminaire ou un groupe de travaux dirigés, avec ou sans vision directe de l'intervenant.
- l'accompagnement en ligne (ou *coaching* en ligne) : la formation est entièrement personnalisée à l'occasion du tutorat synchrone pendant lequel le tuteur et l'apprenant échangent en temps réel sur le contenu présenté.

La téléformation. Elle correspond à tout dispositif de formation s'exercant par Internet, Intranet ou Extranet et ayant pour but de fournir un enseignement/apprentissage à tout individu ou groupe d'individus distants de l'organisme prestataire, du service de formation, selon des modalités souples adaptées au mieux aux possibilités de chacun des individus en formation [Mba03].

Ce mode d'enseignement/apprentissage s'appuyant sur le réseau pour communiquer est de plus en plus utilisé pour favoriser l'apprentissage actif et collaboratif, mais variable selon les spécificités des contextes éducatifs.

Notre intérêt se porte sur l'utilisation des plates-formes pour la mise en place de PBL quelque soient les configurations d'usage (tout distance, en complément, etc.).

3.2.2 Plates-formes de formation à distance

Le cadre général d'utilisation des plates-formes vient d'être présenté. Cette nouvelle section est consacrée à l'étude approfondie de ces plates-formes.

3.2.2.1 Définition générale

Plate-forme de FOAD, plate-forme de formation à distance, plate-forme de e-learning sont divers termes faisant référence à la définition générale de dispositif de formation à distance utilisant les réseaux informatiques comme support ; le dispositif de formation étant l'organisation permettant de réaliser des formations ou des cours avec des objectifs, des documents, des méthodes, des moyens humains, matériels et financiers.

Cette définition nous permet d'écarter certains types d'outils parfois appelés plates-formes comme les portails de formation qui ne sont en fait que de simples sites regroupant des offres de formation. Une autre

définition proche définit la plate-forme comme le « *logiciel qui assiste la conduite des formations ouvertes et à distance* » [ORA00, Eco01]. La « conduite » d'une formation, encore appelée suivi de formation, représente les différentes étapes de préparation/réalisation/évaluation de la formation considérée.

Il convient également de distinguer ces plates-formes du terme homonyme employé en génie logiciel pour faire référence à une technologie particulière sous-jacente à un système (Corba, J2EE, ou encore .NET sont des exemples génériques de ce type de plate-forme). On parle alors de *plate-forme applicative* ou bien d'*infra-structure* pour désigner ces plates-formes.

3.2.2.2 Typologie et usage des plates-formes

La littérature fait référence aux plates-formes de formation à distance sous divers acronymes permettant ainsi de distinguer des objectifs différents d'utilisation de ces dispositifs :

LMS : système de gestion de l'apprentissage (*Learning Management System*)[33]. Il est considéré dans bien des cas comme le cœur du dispositif de la e-formation. Il a pour but la gestion et l'organisation de la formation :
- individualisation et distribution des parcours de formation ;
- gestion des apprenants ;
- suivi de la réalisation des parcours d'apprentissage ;
- mise à disposition d'outils coopératifs pour les relations tuteur/apprenant et aprenant/aprenant.

CMS : système de gestion de contenu (*Content Management System*) [Lah04]. Il a pour but de simplifier la création et la gestion du contenu en ligne. Il permet une meilleure fréquence des mises à jour des ressources déjà publiées. Ceci repose sur deux principes essentiels de fonctionnement des CMS :
- La forme est séparée du fond : ainsi les auteurs doivent-il pouvoir se concentrer uniquement sur leur contenu. Ils disposent pour ce faire de modèles de présentation prédéfinis spécifiques à chaque élément qui compose le document (en-tête, format du titre, emplacement d'une image, intégration d'un fichier multimédia etc.). L'auteur intègre son contenu dans cette ossature. L'unité de gestion de contenu, assurée par le CMS, est le « grain pédagogique ». Le principe de fonctionnement du CMS est l'archivage et la réutilisation de ces « briques de base » dans la composition des parcours individualisés de formation.
- Il induit des procédures de publication des contenus. Deux étapes précèdent la publication : la création, la validation. Le CMS permet de les organiser selon les règles propres à l'entreprise ou l'entité l'utilisant.

LCMS : système de gestion de contenu d'apprentissage (*Learning Content Management System*). Un LCMS est un système (le plus souvent basé sur les technologies Web) qui permet de créer, valider, publier et gérer des contenus de formation.

Les LMS (systèmes de gestion de la formation) et les LCMS sont deux outils différents. La difficulté d'appréhender cette différence provient notamment du fait que bien souvent les LCMS intègrent toutes les fonctionnalités de base d'un LMS[34]. Le LCMS offre donc les services fusionnés du LMS et du CMS. Le LCMS s'appuie sur le modèle des *Learning Objects* (LO) ou encore objets de formation,

[33] La littérature fait généralement référence aux plates-formes sous le terme de LMS. Par la suite, nous emploierons également ce sigle par souci de concision.

[34] Pour comprendre ce qu'est un LCMS, on fait souvent référence à la formule suivante : LCMS = LMS + CMS

objets d'apprentissage. Un *Learning Object* est composé d'objectif de formation, d'évaluations et de contenu. Des données appelées « metadata » y sont associées. Ce sont ces données qui permettront l'individualisation des contenus selon les profils des apprenants. Le LCMS permet de créer des bibliothèques de LO, une bibliothèque de grains de contenu de formation indépendants, qui peuvent être réutilisés et associés indifféremment les uns aux autres. La logique et la capacité d'individualisation intégrées au niveau des LO sont répercutées au niveau de la publication par le biais de ces méta-données. Le LCMS pourra alors, pour un apprenant donné, gérer la distribution et le suivi de l'apprentissage à un niveau très fin : celui de l'objet d'apprentissage.

La seconde fonction des LCMS est la gestion des procédures et des flux de publication. Sur le même modèle que le CMS, le LCMS assure la mise en place d'une organisation garantissant le respect des règles de publication. Après la création d'un LO, l'auteur le soumet à la procédure de validation. S'il est approuvé, il sera publié et donc disponible, sinon il sera rejeté pour être modifié. Les nouvelles fonctionnalités des LCMS permettent de s'orienter vers de nouvelles pratiques dans la création de contenus pédagogiques. Ils favorisent le partage du travail de création et simplifient les mises à jour des ressources déjà publiées. En résumé, le LCMS est un système qui s'appuie sur les principes suivants :

- modèles de *Learning Object* réutilisables,
- une bibliothèque de *Learning Objects* centralisée,
- séparation du contenu et de la forme,
- indexation et outil de recherche,
- procédures et outils de travail (gestion du *workflow*),
- et respect des normes.

3.2.2.3 Classification des plates-formes

Le marché des plates-formes de e-formation est très fragmenté et on compte plus de 300 plates-formes différentes[35] (Docent, WebCT, Click2Learn ou encore T3W) dont un certain nombre en *Open-Source*[36] : Ganesha, Moodle, uPortal, ou encore Claroline (une liste plus exhaustive est disponible à l'adresse http ://www.fnl.ch/LOBs/LOs_Public/OpenSourcePlatf.html).

Les objectifs de formation peuvent être assez divers et c'est pourquoi les plates-formes peuvent se positionner en diverses catégories de produits : logiciels d'administration, campus virtuel, outils de visio-conférence, ou encore logiciels d'aide à la conduite des formations (la catégorie de plates-formes qui nous intéressent).

Des analyses comparatives [ORA00, DEA00, Eur02, Bus03, CSC03] étudient de manière informelle certaines plates-formes répondant à des critères préalablement identifiés et fixés. Par exemple, l'étude de l'ORAVEP [ORA00] avait pour cahier des charges que les plates-formes permettent de mettre en ligne des contenus multimédias, offrent des possibilités de communication entre les enseignants et les étudiants, et permettent l'individualisation des enseignements et le suivi des étudiants. Des exemples d'utilisation

[35] Le site de Thot http ://thot.cursus.edu/rubrique.asp ?no=12074 était célèbre pour son recensement de toutes les plates-formes ; malheureusement la célébrité a désormais un prix et leur article nécessite maintenant un abonnement payant pour être consulté.

[36] Mouvement prônant la diffusion libre des codes source.

de plates-formes pour mettre en place des formations ouvertes et à distance sont recensés par Algora sur son site[37].

Il existe donc plusieurs manières de classifier les plates-formes pour la e-formation : choix technologiques, usages pédagogiques, fonctionnalités, etc. Comme la structure des plates-formes s'articule en fonction de leur logique, et qu'une plate-forme peut avoir de multiples usages pédagogiques, une classification reposant sur les fonctionnalités proposées semble plus judicieuse.

Voici par exemple une classification reposant sur sept dimensions correspondant chacune à un besoin particulier de la formation (proposée par [Bus03]) :

- la gestion de la formation : cela recouvre toutes les fonctionnalités administratives permettant d'automatiser l'organisation de la formation, l'accessibilité des cours et le suivi ;
- la gestion des compétences : les fonctions permettant d'évaluer les compétences individuelles, de les mettre en regard des profils requis pour un poste donné, d'établir des programmes de formation ;
- la bibliothèque de formation : elle désigne la constitution de la base de données des cours et l'organisation du catalogue de formations.
- la création de cours : tous les outils permettant de créer des formations adaptées au médium Internet et de formaliser une certaine démarche pédagogique sans utiliser la programmation ;
- l'accès au cours : tout ce qui a trait à la diffusion des formations et à la mémoire que le système a de l'apprenant ;
- l'accompagnement de l'apprenant (asynchrone) : elle regroupe les différents types d'accompagnateurs et les outils qui sont mis à leur disposition pour suivre l'apprenant ;
- les classes virtuelles[38] (synchrones) : désignent les outils permettant de donner un cours de manière synchrone à plusieurs apprenants qui interagissent entre eux et avec le tuteur.

Etienne Wenger fait référence également aux plates-formes pour la e-formation dans son étude sur les plates-formes technologiques supports aux communautés de pratique[39] [Wen01]. L'apprentissage/enseignement à distance constitue l'un des catégories étudiées.

3.2.2.4 Architecture générale des plates-formes

La structure des plates-formes pour la e-formation se base généralement sur une architecture client/-serveur avec un serveur Web, une base de données et/ou un système de fichiers pour le stockage proprement dit. Toutefois, l'avancée technologique en matière d'architecture Web amène actuellement les plates-formes à proposer des architectures trois-tiers. Grâce à ce modèle d'architecture, il est possible de séparer la présentation, le traitement et le stockage des données. Il est aussi possible d'utiliser les fonctionnalités des navigateurs Internet. Le découpage suivant est observé :

- un client, généralement le navigateur de l'utilisateur, assure la partie présentation et fournit l'interface de commande ;
- un intergiciel garantit, par l'intermédiaire d'un serveur Web, la génération et la mise en ligne des pages correspondantes aux réponses des applications spécifiques ;

[37] http ://www.algora.org/fr/index.asp

[38] Une classe virtuelle désigne un dispositif de formation à distance synchrone répliquant dans le virtuel le concept de classe physique et qui peut être suivie en mode synchrone ou asynchrone [dR02].

[39] Une communauté de pratique est un groupe de personnes partageant un intérêt commun pour un domaine d'effort humain et s'engageant dans un processus d'apprentissage collectif créant leur inter-relations.

– un serveur de base de données stocke les documents et/ou les méta-données associées.

De nouvelles architectures « composants » sont utilisées actuellement pour les plates-formes. Elles sont étudiées dans la section 3.3.

3.2.2.5 Plates-formes et situations d'apprentissage

Une plate-forme permet de structurer des situations d'apprentissage comportant différentes activités pour l'apprenant, ces situations étant isolées ou s'enchaînant les unes après les autres. L'ensemble constitue un « scénario » dont l'apprenant aura une vision et les formateurs une autre vision.

« *Il faut relativiser cette image du « scénario », dérivée de l'audiovisuel. Elle sera pertinente pour suggérer le fait qu'il s'agit d'abord d'une activité de conception et que la première qualité d'un scénario pédagogique est sa lisibilité pour les parties prenantes. Elle le sera moins dans la mesure où les « degrés de liberté » des acteurs dans le scénario sont souvent très importants et les démarches adoptées par les apprenants le plus souvent non prévisibles* » [Eco01].

Ainsi, les scénarios pédagogiques mis en place sur les plates-formes vont du programme structuré à l'espace de découverte individuelle. Les plates-formes ont souvent été développées pour répondre à des pédagogies directives. Dans cette perspective, la représentation adoptée par les concepteurs de l'outil est celle d'un apprenant qui part d'un état et va vers un autre en passant par une série d'étapes fixées par des formateurs-concepteurs du parcours pédagogique, à partir de leur connaissance des objectifs visés et des acquis des apprenants.

L'application privilégiée en formation à distance est la prescription d'activités guidées à deux niveaux, celui des documents et celui des parcours : les documents sont structurants et des parcours sont définis de la façon la plus complète possible. Des évaluations intermédiaires permettent de modifier les prescriptions initiales quand la durée du parcours le permet. L'apprentissage est plutôt conçu dans le cadre d'un parcours individuel.

Bien qu'exprimée de façon récurrente, cette perspective n'est pas la seule envisageable. Les plates-formes, aux réserves techniques près, permettent de mettre en œuvre une démarche pédagogique sans document structuré *a priori*, une approche éducative fondée sur des tâches de production, sur de la recherche d'information et sur des dialogues (voir [Eco01]). Elles semblent donc pouvoir convenir à la mise en œuvre de nos PBL coopératives. Les grands traits de cette démarche pédagogique sont les suivants : un niveau mesuré d'intervention des enseignants, un usage obligatoire des outils de communication, des documents de cours préparés par les apprenants eux-mêmes, la constitution de groupes d'apprentissage. Cette perspective ne supprime pas toute règle d'organisation mais celle-ci devient un paramètre de conception au lieu d'être une donnée qui s'impose.

3.2.2.6 Plate-forme et apprentissage coopératif

Nous avons vu que certaines plates-formes permettent de mettre en œuvre des démarches pédagogiques proches de nos préoccupations. En effet, les plates-formes tendent à évoluer vers un support plus large de modèles pédagogiques, tout en gardant leurs propriétés intrinsèques. Les plates-formes proposent aux usagers impliqués dans la formation (apprenants et tuteurs) des outils de plus en plus nombreux et proches de ceux des environnements dédiés au CSCL ; toutefois, les usages et les enjeux du CSCL et

de la formation à distance sont différents, bien que leurs outils-supports aient de nombreux points communs. Certaines plates-formes sont par construction même dédiées à l'apprentissage collaboratif/coopératif (exemple d'ACOLAD[40] - Apprentissages COLlaboratifs A Distance - plate-forme dont l'interface graphique privilégie la métaphore spatiale en recréant les lieux habituels de formations : amphithéâtres, salle des professeurs, bureaux personnels, salles de séminaires, salles de travail).

Les caractéristiques fonctionnelles nécessaires à l'apprentissage coopératif et à nos PBL (gestion des rôles, des droits, des ressources, des outils asynchrones et synchrones, des activités individuelles et collaboratives, de la production...) semblent plus ou moins prises en compte par certaines plates-formes au regard des outils proposés.

Il est donc possible d'étudier les plates-formes vis-à-vis des fonctionnalités offertes en termes d'outils mis à disposition des apprenants et tuteurs. Pour cela, une classification fonctionnelle s'appuyant sur un modèle semblable à celui des trois « C » [Ell94] ou sur un modèle dérivé [Tar97, Dav01] en quatre dimensions est possible :

L'espace de production : propose les objets qui correspondent à une activité pendant laquelle les participants, individuellement ou en groupe, réalisent des travaux.

L'espace de coordination : exprime les relations entre les utilisateurs et leurs activités. La coordination assure l'efficacité, dans la réalisation de la tâche, du groupe. Elle permet la production collective d'un groupe en assurant la synchronisation relative au travail commun, en gérant, en particulier, les conflits qui apparaissent.

L'espace de communication : permet aux participants d'émettre ou de recevoir des données persistantes. Elles sont obligatoirement liées à la tâche du groupe et sont émises à partir d'un espace privé du participant à un ou plusieurs espaces communs ou privés. L'activité de communication est considérée comme un outil au service de la production et de la coordination.

L'espace de conversation : permet aux participants de dialoguer sans s'échanger des données persistantes. Il s'agit d'une activité informelle qui peut être réalisée de différentes manières synchrone ou asynchrone, textuelle, orale ou visuelle.

D'autres auteurs ont également proposé d'ajouter des dimensions orthogonales au trois espaces initiaux. Par exemple, C. Ferraris propose dans [Fer00] l'espace de *régulation* qui permet de décrire les règles de coopération entre les individus impliqués dans un apprentissage collectif.

La figure 3.2 illustre les quatre dimensions de B. David [Dav01] et décrit les tâches principales associées à chaque cercle et ses intersections. Nous avons numéroté ces différentes parties comme les différents usages possibles pour des apprenants comme des tuteurs. Nous avons ensuite étudié de nombreuses plates-formes et extrait leurs outils pour les positionner vis-à-vis de ces usages[41] : voir le tableau 3.1 (la liste d'outils n'est pas exhaustive).

Au travers de ce travail, nous remarquons qu'il est assez difficile de catégoriser les outils étant donné les différents usages qui peuvent en être fait. De plus, de nombreux autres outils ne sont pas mentionnés dans le tableau car ils correspondent à des outils similaires à ceux déjà listés. Des outils/logiciels indépendants de la plate-forme mais pouvant s'intégrer dans l'environnement de formation (par exemple l'utilisation d'un tableur comme outil de simulation) ne sont pas listés car nous les considérons hors de propos. Nous

[40] http ://acolad.u-strasbg.fr/

[41] L'inverse, c'est-à-dire donner pour chaque usage l'outil le réalisant, est possible.

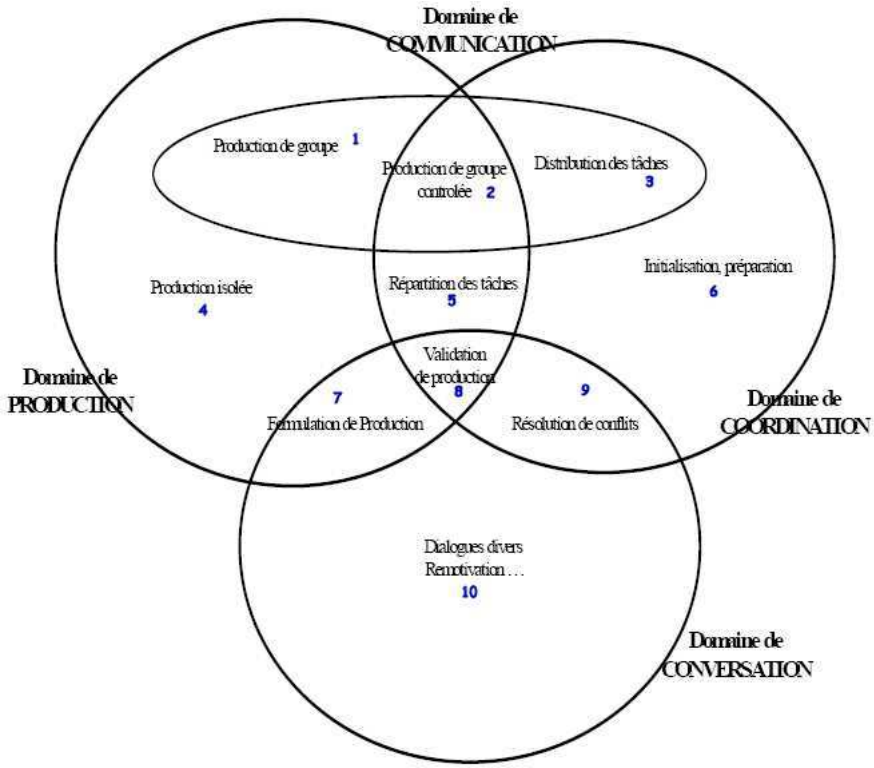

FIG. 3.2 – Le modèle en quatre « C » (repris de [Dav01] et annoté).

Chat	7,8,9,10	Partage d'application	1,2,3
Forum	7,8,9,10	Messagerie instantanée	7,8,9,10
Messagerie électronique	7,8,9,10, 1,2,3,5,6	Visioconférence	7,8,9,10, 1,2,3,5,6
Espace de travail partagé	1,2,3	FAQ	9
Quizz, QCM	4	Présentation de vidéos	4,3,6,5
Liste de diffusion	7,8,9,10,1,2,3,5,6	Tableau blanc	1,2
Accès Web	1	Questionnaires, sondages	1
Agenda	5,6	Tableau de bord	5
Outils d'historiques	4	Outils de suivi (temps, statistiques, etc.)	4

TAB. 3.1 – Outils des plates-formes et usages vis-à-vis du modèle en quatre « C » (les numéros font référence à ceux de la figure 3.2).

remarquons également que l'outil de *courriel* ou messagerie instantanée a de nombreux usages : il peut servir en plus de la conversation à coordonner des actions et à échanger des documents pour la production. De même, certains outils (comme la visioconférence) sont parfois dotés de services liés à d'autres outils. Il

faudrait donc, pour être plus exhaustif dans cette classification, préciser les services rendus par ces outils (distinction outil/services rendus). Les outils d'historiques permettent de conserver les conversations, les connexions, etc. ; ils peuvent être perçus comme des services rendus par les autres outils. En revanche, les outils de suivi sont considérés ici comme des outils servant à la « production » des tuteurs : ils les aident à guider, à assister, à évaluer et à réguler la formation, l'apprentissage des apprenants et les actions des apprenants.

3.2.2.7 Expérimentation de plates-formes

Nous avons expérimenté SMASH sur deux plates-formes : MOODLE[42] et OpenUSS[43]. Nous avons choisi ces deux plates-formes en nous basant sur les nombreux comparatifs/études existants ainsi que sur des études internes à notre groupe de recherche. Ces deux plates-formes répondent aux critères suivants : elles sont gratuites, elles proposent des fonctionnalités/outils « classiques », elles ne suivent pas un modèle pédagogique particulier et elles proposent une structuration modulaire représentative de la majorité des plates-formes éducatives actuelles (le choix d'OpenUSS est également justifié par son architecture composant présentée plus en détail dans la section 3.3). Nous n'avons pas expérimenté ACOLAD bien qu'elle permette la mise en œuvre de PBL[44] [Fae02, Fae03] car elle n'est pas représentative des plates-formes de formation à distance traditionnelles (le manque de scénarisation des situations d'apprentissage mises en œuvre ne convient pas au cas d'étude SMASH).

Notre intérêt est ici porté sur le potentiel offert par les plates-formes en terme de support pour des usages pédagogiques plus complexes que la simple transposition du présentiel à distance et pour des modèles pédagogique d'apprentissage à forte connotation sociale (ici, nos PBL coopératives). Nous analysons notre expérience au regard des critères nécessaires à un environnement informatique support détaillés dans la première section de ce chapitre (3.1) :

- ⋆ R1.1, R1.2, R1.3 : les plates-formes proposent parfois[45] une orientation sociale (noms et profils des apprenants accessibles) ainsi que l'*awareness* des autres participants (elle est généralement liée à l'utilisation des outils synchrones de type chat, tableau blanc mais elle est également parfois intégrée à l'environnement proposé dans le cadre des activités). Les interactions sociales entre apprenant/tuteur (le plus souvent) et entre apprenants (moins souvent) sont supportées par de nombreux outils.
- ⋆ R1.4 : l'environnement proposé par les plates-formes n'est pas personnalisable par les apprenants (c'est au concepteur de modifier la plate-forme en termes d'interface, de fonctionnalités, etc.).
- ⋆ R2.1, R2.2 : les plates-formes ne proposent pas d'outils de représentation d'idées et de leurs relations. Toutefois il existe, dans le domaine plus général des EIAH, des outils de ce type, construits généralement sur les fonctionnalités des tableaux blancs (exemple de DREW [Qui03]). Il est donc

[42] http ://moodle.org/

[43] http ://openuss.sourceforge.net/openuss/

[44] Voir les sites Web : http ://acolad.u-strasbg.fr/html/pedagogie1.htm et http ://web.ccr.jussieu.fr/urfist/ gremi/gremi10juin04/plateforme_acolad.ppt

[45] A l'origine, l'un des enjeux principaux de la formation à distance est l'individualisation de l'apprentissage ; ainsi, les plates-formes les plus anciennes ne proposent pas d'interaction avec d'autres apprenants mais uniquement avec les enseignants/tuteurs.

envisageable que cet outil d'expression soit « ajouté » aux plates-formes ou bien que les fonctionnalités des tableaux blancs soient étendues (nous verrons dans la section suivante consacrée aux plates-formes et composants ces possibilités d'extensions).

- ⋆ R2.3 et R2.4 : l'ajout d'informations rattachées à un référentiel n'est pas une fonctionnalité élémentaire des plates-formes ; toutefois, certains outils comme les forums de discussion, où un message peut contenir des fichiers, peuvent être utilisés dans ce but (les espaces de travail partagés peuvent être utilisés également). La négociation de connaissances partagées peut se réaliser de multiples manières sur la base des outils de conversation/communication.
- ⋆ R3.1 : les plates-formes ne définissent en général que deux rôles externes : apprenant et tuteur.
- ⋆ R3.2 : les apprenants peuvent être guidés dans leurs activités si la plate-forme prend en compte l'affichage direct d'informations sur le contexte dans lequel se trouvent les apprenants ; ou bien indirectement par la mise à disposition de ressources (vidéos, textes, etc.) que l'utilisateur doit « lire » pour obtenir cette aide.
- ⋆ R3.3 : la synchronisation des activités collaboratives n'est pas encore proposée par les plates-formes : les apprenants sont en général libres de ne pas suivre les activités (présentées sous la forme de cours/modules/exercices/etc.) dans l'ordre prévu, comme de travailler sur plusieurs activités alternativement (des travaux actuels de recherche proposent des plates-formes basées sur le principe de *workflow* afin de supporter l'ordonnancement des activités [Van03]).
- ⋆ R3.4 : les plates-formes ne permettent pas aux apprenants de définir explicitement des stratégies d'apprentissage et d'en changer à tout moment, sauf si l'on considère qu'un scénario très ouvert (décomposition en peu d'activités non directives et proposant un grand nombre d'outils/fonctionnalités), proposé par la plate-forme, répond indirectement à ce critère.
- ⋆ R4.1, R4.2,R4.3, R4.4 : tout ce qui concerne la définition, ou encore le suivi de plan n'est pas proposé par les plates-formes actuelles.

Ces différents résultats sont synthétisés dans le tableau 3.2 ("+"/"-" indiquent une forte/faible prise en compte ; Ø indique que le critère n'est pas supporté).

R1.1	R1.2	R1.3	R1.4	R2.1	R2.2	R2.3	R2.4
-	-	+				-	-
R3.1	R3.2	R3.3	R3.4	R4.1	R4.2	R4.3	R4.4
+	-						

TAB. 3.2 – Critères de la section 3.1.3 pris en compte par les plates-formes de formation à distance .

3.2.3 Constat actuel sur les plates-formes

La multiplicité des plates-formes existantes témoigne d'une tendance au développement de plates-formes répondant à des besoins spécifiques sans chercher à adapter les plates-formes existantes (car leur architecture et implémentation technologique ne s'y prêtent pas). En effet, de nombreux travaux actuels de recherche consistent à développer de nouveaux environnements et prototypes de plates-formes de formation à distance répondant à des besoins pédagogiques bien précis (exemples : soutien d'une

pédagogie à base de projet [Geo01], suivi d'activités en mode synchrone à l'usage des tuteurs [Des01]). Ceci provient du fait que c'est au concepteur de s'adapter aux caractéristiques de la plate-forme ainsi qu'au modèle pédagogique sous-jacent.

De manière générale, les activités d'apprentissage collaboratives/coopératives exploitent les possibilités d'interaction à distance offertes par les outils de communication/conversation synchrones et asynchrones [Leg01]. « *Le travail collaboratif et la communication sont souvent invoqués comme des procédés à valeur ajoutée favorisés par des outils de type plate-forme* » [Eco01]. La mise en œuvre s'avère par contre délicate. Les raisons expliquant les échecs sont nombreux : outils insuffisamment opérationnels, absence de référence aux compétences professionnelles visées, absence de référence dans le domaine ou la discipline d'étude, absence d'habitude, de pratique, etc.

Les outils et services proposés par les plates-formes sont soit transversaux aux formations proposées (c'est-à-dire indépendants de la formation, non-intégrés, accessibles en permanence), soit intégrés dans le scénario de l'apprentissage, donc accessibles dans un contexte précis de la formation. Dans ce dernier cas, les services proposés sont des services de communication, de production, ... génériques et non des services « pédagogiques » (c'est-à-dire des services à connotation pédagogique comme *poser une question au tuteur* ou *partager ses résultats avec un autre groupe*). Ce constat s'explique par la conception actuelle pendant laquelle les outils sont associés aux activités pédagogiques de manière statique (les actions et événements renvoyés par la réalisation effective des activités n'affectent pas les outils et services proposés). En réalité, les outils ne sont pas forcément très adaptés à la formation en raison de leur généricité et de leur usage pédagogique neutre. Du point de vue de la conception qui nous intéresse, la difficulté de la mise en œuvre d'un apprentissage coopératif tel que nous envisageons les PBL réside également dans la difficulté de structuration des activités (individuelles et collectives). Le travail de conception doit s'intéresser alors à la définition d'un scénario pédagogique en accord avec les possibilités de la plate-forme.

Les expérimentations de plates-formes ont prouvé que chaque fournisseur/éditeur de plate-forme définit son propre formalisme et que les concepts utilisés ne sont pas homogènes entre plates-formes, obligeant les utilisateurs à adopter le modèle pédagogique de la plate-forme [Lec03]. Toutefois, des travaux actuels cherchent à modifier et enrichir les plates-formes. Par exemple dans [Lec03], le modèle pédagogique[46] sous-jacent à Ganesha a été enrichi de nombreuses fonctionnalités comme, par exemple, l'amélioration de la décomposition des modules. De manière similaire, nos travaux sur les plates-formes OpenUSS et Moodle ont porté également sur l'extension des fonctionnalités proposées par le biais de la conception et du développement d'outils supplémentaires (tableau blanc synchrone partagé entre apprenants, afficheur d'étapes pour indiquer aux apprenants leur progression, etc.). Ceci reste toutefois un travail très technique, non reproductible et peu capitalisable car trop dépendant de l'architecture des plates-formes (si elle est monolithique aucune modification n'est possible).

Il est peut-être alors possible d'aborder le développement de dispositifs de support à la FAD, pour nos PBL en particulier, sur la base d'autres perspectives : le développement d'outils suffisamment génériques pour être couplés à des plates-formes existantes ([Mba03]) ou bien en appliquant le savoir-faire composant aux plates-formes. Notre intérêt se porte sur cette deuxième perspective développée dans la section 3.3. En effet, cette perspective correspond à l'évolution actuelle des architectures des plates-formes qui tendent vers des architectures fonctionnelles et technologiques distribuées et ouvertes ; ceci afin de les rendre, dans un premier temps, interopérables. Auparavant, nous nous intéressons aux problèmes de normalisation des

[46] Il est implémenté dans la base de données de la plate-forme.

plates-formes qui constituent actuellement le cadre international au centre de tous les travaux abordant les notions d'objets et composants réutilisables pour les plates-formes.

3.2.4 Normes et standards pour le e-learning

Plusieurs travaux et études portent sur la réalisation des standards, normes et labels au niveau international et français. On peut citer la définition d'une norme ISO (International Organisation for Standardisation), les standards techniques tels qu'AICC, SCORM, ou encore la labelisation OPQF sur la e-formation. Ces efforts de normalisation et de standardisation visent à répondre aux questions relatives à l'interopérabilité des technologies et des contenus entre les plates-formes de e-formation. Les principaux enjeux de la standardisation et de la normalisation sont donc de rendre inter-opérables les techniques entre elles afin de faciliter l'usage des ressources éducatives quel que soit la plate-forme ou l'environnement technologique utilisés [Mba03] (et indirectement de protéger et d'augmenter le retour sur investissements dans les technologies). Les standards sont un moyen crucial de garantir entre autres [Cen03] :

L'interopérabilité : vis-à-vis du contenu manipulé/échangé par différents systèmes et vis-à-vis de la communcication/interaction de plusieurs LMS.

La ré-utilisation : vis-à-vis de l'assemblage rapide de contenus et de codes et vis-à-vis de l'assemblage et de l'utilisation dans de nouveaux contextes des objets incorporant les contenus.

L'adaptabilité : le système peut être configuré pour avoir des fonctionnalités étendues visant de nouveaux buts.

3.2.4.1 Les organismes et acteurs impliqués

Les principaux travaux de standardisation sont soutenus par les organismes suivants (la liste n'est pas exhaustive) :

- AICC (*Airline Industry CBT Consortium*) [AIC00],
- ADL (*Advanced Distributed Learning Network*) [ADL],
- ARIADNE (*Alliance for Remote Instructional and Authoring and Distribution Networks for Europe*) [ARI],
- CEN/ISSS/LT (le sous-groupe *Information Society Standardization System* de la *Commission Européenne pour la Normalisation*) [CEN],
- DUBLIN CORE [Cor],
- GESTALT (*Getting Educational Systems Talking Across Leading Edge Technologies*) [GES],
- IEEE LTSC (le *Learning Technology Standards Committee* de l'IEEE) [LTS],
- IMS (*Instructional Management System project*) [IMS],
- ISO/IEC JTC1/SC36 (le *Standards Committee* du *International Standards Organization and International Engineering Consortium Joint Technology Committee*) [SC3],
- PROMETEUS (*PROmoting Multimedia Access to Education and Training in EUropean Society*) [PRO],
- et autres organismes : ALIC [ALI], EdnA [EdN], eduSource [edu], NATO ITD [ITD].

IEEE LTSC est l'institution qui rassemble actuellement les recommandations et propositions venant des autres institutions et projets de standardisation de l'apprentissage. L'ISO/IEC JTC1 a créé

le *Standards Committee for Learning Technologies* SC36 [16], qui continue les travaux pour la création de standards internationaux ANSI ou ISO. Pour la France, c'est l'AFNOR [AFN] qui a la charge de la standardisation.

3.2.4.2 Normes/Standards et spécifications actuelles

Dublin Core : la norme de métadonnées[47] du Dublin Core est un ensemble d'éléments pour décrire une grande variété de ressources. Elle comprend 15 éléments dont la sémantique a été établie par un consensus International de professionnels provenant de diverses disciplines. Chacun des éléments est optionnel ; il peut être répété et associé à un nombre limité de « qualifiants » afin d'affiner la description sémantique. Des terminologies précises sont utilisées pour permettre l'interopérabilité. Ces 15 éléments forment aujourd'hui le *Dublin Core Metadata Element Set*[48].

RDF (*Resource Description Framework*) : cette norme[49] accroît les possibilités d'exploitation des méta-données. Elle a été créée par le W3C (*World Wide Web Consortium*) qui facilite le traitement des méta-données. Ce format fournit l'interopérabilité entre les applications qui échangent de l'information non compréhensible par les machines du Web (RDF Recommendation, 1999). RDF permet d'insérer des méta-données qui permettent aux utilisateurs de prendre d'avantage contrôle de l'utilisation de leurs données personnelles lorsqu'ils visitent des sites Web.

LOM (*Learning Object Metadata*) : le standard LOM [LOM02], issu du groupe LTSC et basé sur les travaux d'IMS et ARIADNE, spécifie la syntaxe et la sémantique des métadonnées pédagogiques et définit les attributs nécessaires à une description complète des ressources pédagogiques à partir d'un ensemble de neuf catégories de descripteurs[50]. L'objectif des comités de normalisation est de spécifier la syntaxe et la sémantique des métadonnées décrivant un objet pédagogique, ou *Learning Object* (LO). LOM est devenu un standard *de facto*.

L'architecture LTSA : spécification de l'IEEE [LTS01] concernant l'architecture d'une plate-forme de formation en ligne à l'aide de composants en définissant les rôles de chacun à l'intérieur d'une architecture appelée LTSA (*Learning Technology System Architecture*). Il s'agit d'une architecture de haut niveau pour les systèmes de formation, d'éducation, et d'apprentissage. LTSA est un modèle fonctionnel composé de diagrammes de flots de données (*Data Flow Diagram*). Il fournit un cadre de référence pour comprendre l'existant et le futur système, favorise la portabilité et inclut un horizon technique tout en restant adaptable aux nouvelles technologies et aux systèmes de formation en ligne.

[47] On peut définir simplement une métadonnée comme une donnée sur une donnée

[48] http ://dublincore.org/documents/dces/

[49] http ://www.w3.org/RDF/

[50] Les neuf catégories sont les suivantes : ***General*** qui rassemble des informations générales telles que *identifier*, *title*, *language*, *keywords*, etc. ; ***Life Cycle*** qui inclus *version* et *status* ; ***Meta-Metadata*** qui exprime la forme générale pour les méta-données et donne des informations sur la création de ces méta-données ; ***Technical*** avec *format*, *size*, *system requirements*, *location of the resource*, etc. ; ***Educational*** qui inclus *interactivity type*, *learning resource type*, *interactivity level*, *difficulty*, et *intended user* ; ***Rights*** qui inclus *costs* et *copyright* ; ***Relation*** pour les relations entre l'objet cible et d'autres objets ; ***Annotation*** qui inclus des informations sur l'auteur de l'objet d'apprentissage et la date de création ; et ***Classification*** avec *purpose*, *kind of taxonomy*, *keywords*, etc.

SCORM (*Sharable Content Object Reference Model*) : cette spécification[51] a été développée par l'ADL. Les recommandations et spécifications techniques définies dans le modèle de référence SCORM peuvent être regroupées en trois thèmes majeurs :

- le *Content Aggregation Model* ou modèle d'agrégation de contenus pour la pédagogie par Internet : il contient la démarche à suivre pour identifier et organiser des ressources en contenus pédagogiques structurés. Sont fournis également une nomenclature d'objets pédagogiques et la description d'un modèle d'organisation des contenus (*SCORM Content Packaging*) référençant le modèle de méta-informations pour les ressources pédagogiques mis au point par l'IMS ainsi que la spécification LOM.
- le *Run-time Environment* ou environnement d'exécution des objets pédagogiques : il contient les directives pour exécuter des objets pédagogiques, communiquer, et garder les traces dans des plates-formes de e-formation. Il est basé sur les directives d'interopérabilité contenues dans le CMI001 mis au point par l'AICC.
- le *SCORM Sequencing and Navigation* : cette spécification concerne les responsabilités essentielles d'un LMS pendant le *run-time* pour assurer le séquençage de contenus et la navigation dans ces séquences. Il est basé sur la spécification *IMS Simple Sequencing*.

IMS : le consortium IMS a réalisé un ensemble de spécifications également appelées IMS[52] :

- *IMS Learning Resources Meta-Data Specification*, pour la description des ressources d'apprentissage pour chercher et découvrir.
- *IMS Enterprise Specification*, pour partager les données sur les apprenants, les cours, les performances, etc., avec des applications administratives et des services au travers de multiples plateformes et interfaces.
- *IMS Content Packaging Specification*, pour créer et partager des objets de contenus réutilisables.
- *IMS Question & Test Specification*, pour partager des articles de test et autres outils de contrôle.
- *IMS Learner Information Package Specification*, pour l'organisation des informations d'apprenant de manière à ce que les systèmes d'apprentissage puissent mieux réagir aux besoins spécifiques des utilisateurs.
- *IMS Reusable Competency Definition Specification*, pour décrire, référencer et échanger les caractéristiques clés d'une compétence.
- *IMS Simple Sequencing Specification*, pour la spécification de la manière dont les objets d'apprentissage sont ordonnés et présentés à l'apprenant.
- *IMS Accessibility* (qui ne devrait jamais être une spécification réelle), pour la mise à disposition d'une ligne directrice à d'autre groupes de travail IMS et s'assurer que chaque spécification offre la plus grande accessibilité à tout type d'apprenants.
- *IMS Learning Design Specification*, pour la définition de scénario d'apprentissage et d'interaction pour les créateurs de contenu ou de cours.
- *IMS Digital Repositories Specification*, pour l'intégration de l'apprentissage en ligne avec des ressources d'information pertinente.

[51] http ://www.adlnet.org/index.cfm ?fuseaction=DownFile&libid=648&bc=false
[52] http ://www.imsglobal.org/specifications.cfm

3.2.4.3 Bilan

Les approches de standardisation sont centrées sur la réutilisation de ressources ; elles sont orientées contenu. Les normes facilitent le travail du *fournisseur de ressources pédagogiques* (rôle décrit en 1.2.2.4), en donnant accès à de larges dépôts d'objets ou ressources pédagogique réutilisables, réduisant ce faisant le besoin de développer un produit en fonction de plusieurs systèmes. Les normes incitent également à créer des contenus modulaires plus faciles à maintenir et à mettre à jour.

La plupart des spécifications considère une formation comme le résultat de l'assemblage de grains pédagogiques (objets d'apprentissage ou LO). Toutefois, certaines spécifications (IMS-LD par exemple) s'orientent vers un nouveau découpage scénario/objets d'apprentissage et visent à spécifier formellement ces scénarios indépendamment des LO afin de les rendre plus facilement reproductibles et réutilisables. La spécification d'IMS-LD est étudiée en détail dans le chapitre suivant (4.3) consacrée à l'étude des langages existants pour modéliser des situations d'apprentissage.

La modification de l'architecture même des plates-formes constitue une autre approche centrée sur la réutilisation d'« éléments » : les composants.

3.3 Plates-formes et composants

Dans cette section, nous nous intéressons aux plates-formes de formation à distance basées composants mais également aux composants pour les plates-formes au sens le plus large. Pour cela, nous nous intéressons dans un premier temps à synthétiser diverses approches éducatives actuelles mentionnant le terme ambigu de « composant éducatif ». Ceci nous permet de mieux appréhender les diverses caractéristiques de ces composants et de distinguer les différents « composants » intervenant dans les plates-formes de formation à distance.

3.3.1 Objets et composants éducatifs pour les environnements éducatifs

La réutilisation, comme le terme de *composant*, sont sujets à de multiples définitions et préoccupations. L'interprétation la plus fréquente dans la communauté éducative se réfère à tout type de *matériau* pouvant être réutilisé dans une formation éducative : applet, image, vidéo, ou encore morceau de code. Cette vision du composant se rapproche de celle d'*ingrédient* et se rapporte davantage à la définition d'*objet d'apprentissage* (*Learning Object*). Les problèmes inhérents à l'utilisation de tels composants sont, d'un point de vue technique, des problèmes de base de données et d'interface utilisateur [Spa02]. La sous-section 3.3.1.1 détaille ces objets d'apprentissage. Lorsqu'il s'agit de réutiliser des composants sous la forme de bout de code qui devront être assemblés par la suite pour former une application ou un système éducatif, le terme de composant se rapproche alors de celui de *composant logiciel*. Cette vision du composant est détaillée dans la sous-section 3.3.1.2.

3.3.1.1 Les objets d'apprentissage (*Learning object* - LO)

Les LO ont été normalisés au travers des différents travaux de standardisation dont est issu le modèle LOM (vu dans la section précédente 3.2.4.2). Un objet d'apprentissage (*Learning object* - LO) se définit comme :

« *...any entity, digital or non-digital, which can be used or referenced in technology supported learning* » [LOM02].

Un LO peut donc être physique (par exemple un texte, un livre, un CD-ROM) ou numérique (comme du texte électronique, une image graphique GIF, une vidéo *Quicktime* ou encore une applet Java). La granularité d'un LO varie donc de quelques bits de textes aux petites applications délivrées sur le Web. Un LO se rapporte également à des ressources plus larges comme des pages Web entières ou de larges applications.

En utilisant des méta-données de description appropriées, les LO deviennent des unités modulaires, souvent comparées au bloc « Lego » ou aux atomes [Wil00], qui peuvent être assemblées pour former des leçons, cours, etc. Un exemple d'assemblage de LO est proposé dans la spécification de SCORM : les *asset* (la forme la plus basique de ressources d'apprentissage) sont assemblées en collection pour former des SCO (*Shareable Content Object*) qui représentent la plus petite forme de ressource d'apprentissage utilisant l'environnement d'exécution (SCORM RTE) et communicant avec le LMS. Un autre exemple est celui proposé par Cisco [Cis01] avec les RLO (*Reusable Learning Object*) et RIO (*Reusable information Object*) illustré par la figure 3.3.

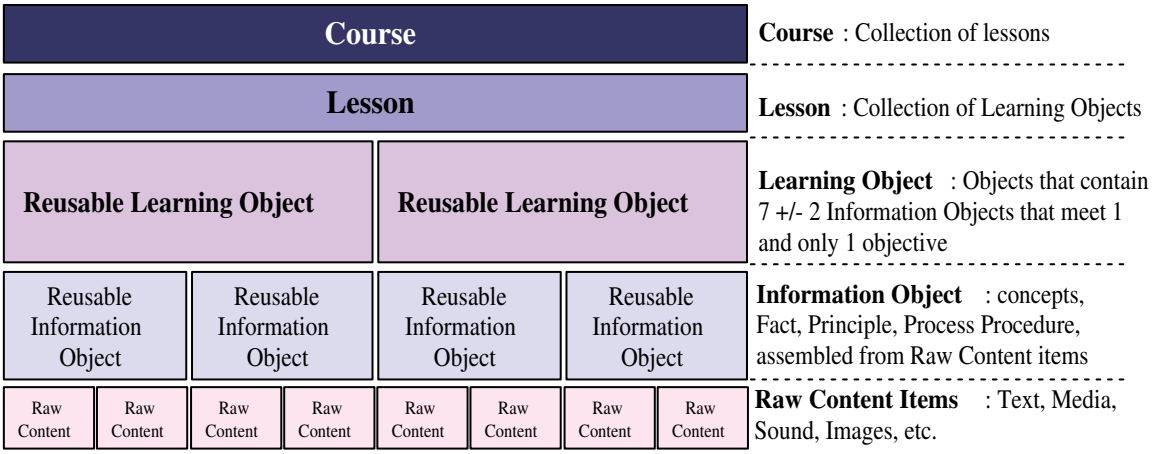

FIG. 3.3 – Exemple d'un modèle d'assemblage de LO [Cis01].

Une approche formelle de l'assemblage est proposée dans [Bou03] qui fournit un modèle formel de composant éducatif pour réutiliser des contenus déjà développés.

Les LO « numériques » répondent aux enjeux suivants :

Accessibilité : permettre la recherche, l'identification, l'accès et la livraison de contenus et composantes de formation en ligne de façon distribuée.

Interopérabilité : permettre l'utilisation de contenus et composantes développés par une organisation sur une plate-forme donnée par d'autres organisations sur d'autres plates-formes.

Ré-utilisabilité : permettre la réutilisation des contenus et composantes à différentes fins, dans différentes applications, dans différents produits, dans différents contextes et *via* différents modes d'accès.

Durabilité : permettre aux contenus et composantes d'affronter les changements technologiques sans la nécessité d'une ré-ingénierie ou d'un re-développement.

Adaptabilité : permettre l'élaboration sur mesure des contenus et des composantes.

3.3.1.2 Les composants logiciels éducatifs (*Educational Software Component* - ESC)

La réutilisation est un enjeu central dans la communauté logicielle mais également pour la communauté éducative. De manière très générale, cette réutilisation doit permettre de réduire les temps et les coûts nécessaires à la création de logiciels éducatifs de grande qualité [Rep01]. Roschelle explique l'origine des besoins croissants des composants dans les applications éducatives [Ros99].

Le composant logiciel éducatif fait référence par son nom au composant logiciel traditionnel. Une définition de *composant logiciel* largement référencée et reconnue dans le domaine de l'ingénierie logicielle est celle de Szyperski : *unité de composition avec des interfaces contractualisées et un contexte de dépendance exprimé explicitement* [Szy02]. Il ajoute ensuite qu'un composant peut être déployé indépendamment et qu'il est sujet à composition par une tierce personne. Un composant logiciel éducatif peut alors être présenté comme un « composant métier » puisqu'il prend en compte explicitement le domaine d'application éducatif [Bar04].

Comme pour les LO, les ESC ont une granularité variable. Koutlis et Roschelle proposent une méthode à deux niveaux pour déterminer la bonne taille de composant pour un domaine particulier d'application du futur logiciel éducatif [Ros99].

Les ESC peuvent également se distinguer par leur niveau d'abstraction [Spa02] :

- les composants « **boîte noire** » : ces composants éducatifs sont distribués sous la forme compilée ou binaire et ne peuvent pas être facilement modifiés ; les fonctionnalités du composant sont documentés et accessibles mais le code contenu ne peut être changé ; le composant peut être facilement utilisé et testé mais peut ne pas répondre à toutes les attentes du programmeur ;
- les composants « **boîte blanche** » : ils sont plutôt distribués sous forme de code source et peuvent ainsi être entièrement modifiés ; si de nouvelles fonctionnalités peuvent être ajoutées il est en revanche difficile de garantir la maintenance et la qualité du composant ;
- les composants « **boîte grise** » : la structure interne des composants plus complexes est visible et peut être modifiée par ré-écriture ou bien en y replaçant des petits composants *boîtes noires*.

Spalter énonce plusieurs problèmes quant à la construction et l'utilisation des composants éducatifs [Spa02] :

- la masse critique : combien de composants garantissent l'utilité d'un entrepôt ? quelle granularité des composants est la plus effective ?
- les propriétés intellectuelles ;
- les spécificités des plates-formes et systèmes ;
- programmation dans l'environnement universitaire : l'équipe chargée de la conception d'un logiciel éducatif est souvent composé d'étudiants et chercheurs universitaires qui, malgré leurs compétences, perçoivent difficilement si leurs composants pourraient interagir avec d'autres provenant d'autres sources[53] ;

[53] Le projet E-Slate a montré qu'il fallait deux ans à un programmeur confirmé pour concevoir un véritable composant réutilisable.

- l'assurance de la qualité : comment garantir aux programmeurs téléchargeant un composant qu'il fonctionnera comme prévu ?
- la recherche et les méta-données pour des larges entrepôts ;
- l'issue sociale : les équipes multi-disciplinaires chargées de travailler à assembler ces composants ont des perspectives sociales différentes et donc appréhendent différemment la conception du composant.

3.3.2 Projets et recherches sur les composants éducatifs

De nombreux projets éducatifs se sont intéressés au problème de la création de composants logiciels ré-utilisables pour les applications éducatives.

Le projet Exploratories [EXP] : création d'applets Java pour l'enseignement de l'introduction au graphisme par ordinateur (les applets utilisent la librairie Java3D).

Le projet ESCOT [ESC] : ce projet teste et rassemble de nombreux composants et outils pour créer des applications éducatives dans le cadre de l'apprentissage des mathématiques. Les outils listés permettent la création d'applets (AgentSheet [Age]), SimCalc, SketchPad, etc.).

E-Slate [E-S] : E-Slate est un environnement d'apprentissage expérimental [Kou99, Bir00, Kyn02]. Il fournit un cadre pour la création d'applications éducatives dynamiques et riches en fonctionnalités (micro-mondes) par des non-programmeurs. Ces micro-mondes sont conçus en assemblant et reliant des composants éducatifs fournis par E-Slate comme un kit d'objets pré-fabriqués et interopérables. Les micromondes peuvent être facilement construits en assemblant les composants selon différentes configurations. Le comportement des composants comme des micromondes peut être programmé dans un langage de script basé sur Logo [Kou97, Kyn01]. E-Slate est basé sur les technologies et la plate-forme Java.

CREATE (*Component Repository and Environment for Assembly of Teaching Environments*) [CRE] : ce projet a pour objectif de catalyser la création d'un large ensemble d'expériences d'apprentissage actives en ligne, et d'organiser ces activités de manière à faciliter le travail de localisation pour les enseignants.

Le projet Boxer [diS] : il a été créé afin de supporter et encourager le développement de ressources et d'outils ouverts afin de permettre à des éducateurs, de différents niveaux d'expertise, de modifier, d'étendre, ou de construire ces ressources.

NSF NSDL (National Science Foundation's National Science Digital Library) [NSF] et le projet **Web/-COMP** [WEB] rassemble et harmonise de nombreux projets (dont certains mentionnés précédemment).

Les ESC concernent, pour la plupart des projets, des composants logiciels basés sur des technologies Web de type applet ou JavaBeans. En effet, la méthode généralement suivie pour créer des ESC réutilisables amène chaque composant à proposer des fonctionnalités réutilisables dans les applications éducatives. Pour cela, le composant doit être bâti sur une implémentation technologique du patron modèle-vue-contrôleur (MVC).

Toutefois, nous avons recensé des approches composants différentes :

- l'approche visant à faire coopérer des prototypes de logiciels éducatifs implantant des fonctionnalités diverses : chacun de ces prototypes est perçu comme un composant. Les fonctionnalités rendues par ces composants sont les briques de base. L'objectif est de faciliter la conception et la réalisation d'expérimentations en permettant à l'utilisateur de choisir les fonctionnalités dont il a besoin [Ros03].
- l'architecture SimulNet : il s'agit d'un modèle de composants en couches pour le support d'applications interactive et collaborative basées sur le Web (framework) ; le développement d'une application éducative basé sur ce modèle a été réalisée [Ani01]. Chaque composant a pour but de fournir une fonctionnalité des applications collaboratives et interactives basées sur le Web (exemple : tableau blanc). Ils sont *plug&play*, codés en Java, et directement invocables par un appel de méthode. Ils sont stockés côté serveur et téléchargés sous forme d'applets java. Ils sont reliés aux applications coté client par l'API[54] de la couche SSC (*Simulnet Software Component*). Ils collaborent entre eux grâce à la couche sous-jacente des services de SimulNet. Chaque composant a sa propre interface.
- l'approche visant à contribuer à la standardisation de la e-formation au niveau métier (*business logic tier*) d'une application 3-tiers. Le principe repose sur une identification des services communs de la e-formation permettant de proposer une « architecture de référence ». Celle-ci sera composée de composants logiciels réutilisables avec des interfaces ouvertes et clairement identifiées. Une instance concrète de cette architecture est proposée en se basant sur la technologie CORBA [Ani02].
- l'activité *Shared Virtual Laboratory* du récent projet européen d'excellence Kaléidoscope[55] vise à proposer un *portail* pour aider les membres de sa communauté à partager, échanger et réutiliser des logiciels, applications mais aussi des moyens et des techniques. L'objectif est de promouvoir l'interopérabilité entre différents environnements d'apprentissage et outils.

3.3.3 Typologie des différents composants éducatifs

Nous synthétisons les deux sous-sections précédentes en proposant une classification des composants éducatifs basée sur l'orientation du composant : orienté contenu pour l'un, orienté conteneur pour l'autre (tableau 3.3). Ceci réduit la granularité des LO aux micros-applications (de type applet) réutilisables pour l'élaboration d'une formation car elles embarquent des connaissances. A l'opposé, les ESC concernent ces mêmes micro-applications mais également toute partie d'un système logiciel éducatif. Ces deux catégories ne sont pas exclusives : certains composants embarquent des connaissances mais peuvent être utilisés dans plusieurs applications (exemple des composants du projet E-SLATE). Du point de vue de l'assemblage, les LO permettent d'élaborer des contenus plus gros jusqu'à former des modules ou unités d'apprentissage ; en revanche, les ESC sont assemblés pour former des modules fonctionnels, voire des systèmes ou applications éducatives complètes. Les ESC se distinguent par un niveau d'abstraction variable entre boite noire, blanche ou grise. Pour les LO, l'abstraction n'a pas la même sémantique ; elle correspond davantage au degré de généricité de l'objet d'apprentissage : plus il est abstrait et plus il est réutilisable mais il est moins spécifique au domaine, ce qui réduit finalement son intérêt ; par contre, plus le LO est spécifique à un domaine et moins il est réutilisable (niveau bas). Enfin, les différents acteurs participant au processus itératif de conception d'une formation à distance (voir 1.2.2.2) ne sont pas concernés par l'usage des mêmes composants : les LO s'adressent davantage au *fournisseur de ressources pédagogiques* qui les

[54] API : *Application Protocol Interface.*
[55] http ://www-kaleidoscope.imag.fr/

décrit ou les crée comme à l'*ingénieur pédagogique* et à l'*enseignant* qui les manipule et les assemble ; les ESC correspondent davantage aux *développeurs* et aux *assembleurs de composants*. Pour reprendre la classification des composants métiers (proposée dans [Bar04]), les ESC correspondent à des composants métiers « verticaux » : c'est-à-dire composants utilitaires, processus métier et entité métier ; les LO rappellent plutôt les objets métiers du génie logiciel.

	Orientés contenu	Orientés conteneur
Appellations	LO	ESC
Exemples de granularité	texte...applet	applet, micro-application
Résultat d'assemblage	module/unité de formation	système/application
Niveau d'abstraction	généricité vs domaine	boite noire vs boite blanche
Rôles concernés	fournisseur de composant, enseignant et ingénieur pédagogique	assembleur de composants et développeur de composants
Analogie GL	objets métiers	composants métiers « verticaux »

TAB. 3.3 – Classification des composants éducatifs selon l'orientation du composant.

3.3.4 Vers des plates-formes ouvertes et flexibles : les plates-formes basées composants

Les plates-formes de formation à distance sont des applications éducatives particulières. Elles sont concernées par la manipulation des LO, comme nous l'avons étudié dans la section 3.2.2, mais également par les ESC. Les ESC concernent moins des composants de type interface que les applications éducatives traditionnelles. Dans le cas des plates-formes, l'assemblage de composants pour le développement d'une plate-forme spécifique relève beaucoup de l'assemblage de composants architecturaux. Ces composants sont liés à la technologie spécifique de la plate-forme développée.

3.3.4.1 Vers des architectures de plates-formes basées composants

Les plates-formes de formation à distance traditionnelles ont pour la plupart une architecture monolithique lié aux technologies Web utilisées mais surtout à leur choix de conception (l'évolutivité n'était pas un critère commercial). Actuellement, des plates-formes basées sur les technologies Web distribuées font leur apparition. Par exemple OpenUSS (plate-forme *Open source*) ou encore Cybeosphere II[56] (plate-forme commerciale) ont choisi l'architecture J2EE. D'autres plate-formes utilisent l'architecture .NET comme OpenPortal. D'un point de vue commercial, ces choix permettent de mettre en avant la modularité de la plate-forme et ses possibilités d'évolution en terme de nouvelles fonctionnalités ou services (pris en charge par l'éditeur de la plate-forme). Cette évolutivité est parfois traduite sous le terme de plate-forme « ouverte ». D'un point de vue plus technique une plate-forme basée composants est davantage

[56] http ://www.cybeosphere.fr

flexible et ajustable : la plate-forme peut s'adapter plus facilement à différents modèles pédagogiques, différents besoins pédagogiques, différentes organisations. Les interfaces utilisateurs peuvent parfois aussi être personnalisées. L'ajout de nouveaux composants par les utilisateurs n'est permis que par les plates-formes gratuites. Par exemple, OpenUSS propose une méthode pour ajouter des composants « *plugable* » (voir la couche des composants *plugable* dans la Figure 3.4 et le détail de la méthode dans [Dew00])[57].

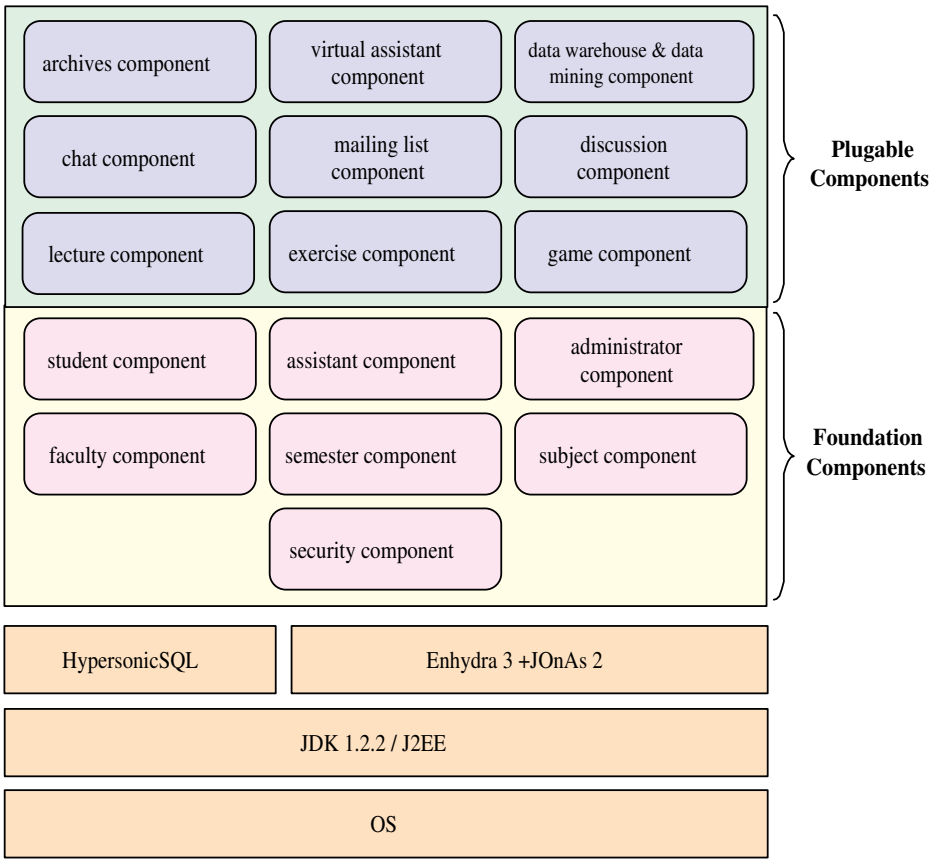

FIG. 3.4 – Les différents composants d'OpenUSS.

3.3.4.2 Typologie des composants logiciels de ces nouvelles plates-formes

En général les composants mis en œuvre sur ces plates-formes peuvent être perçus selon deux dimensions :

– la dimension technique/technologique : les composants sont perçus selon leur positionnement dans l'architecture distribuée multi-tiers. Cette dimension concerne davantage les développeurs de la plate-forme (développeurs de composants, assembleurs de composant).
Par exemple dans la plate-forme OpenUSS (Figure 3.5), les composants de la couche présentation sont appelés PO (*Presentation object*), ceux de la couche « métier » BO (*Business Object*) et ceux

[57] Les explications détaillées de chaque composant d'OpenUSS apparaissant dans la figure 3.4 sont disponibles à l'URL http ://openuss.sourceforge.net/openuss/developer/groups/architecture/architecture.html

de la couche données DO (*Data Object*). Les BO se décomposent également en trois catégories :

1. *Entity Components* (BO - *Business Objects*) : « *Assistant* », « *Faculty* » sont des exemples de ces BO. Ils sont implémentés par des EJB[58] entités.
2. *Association Components* (RO - *Relationship Objects*) : par exemple « *AssistantFaculty* ». Les RO sont implémentés également avec des EJB entités.
3. *Workflow - Helper Components* (WO - *Workflow Objects*) : par exemple « *AssistantHelperWO* ». Les WO sont implémentés avec des EJB sessions.

Le découpage précédent rappelle la classification des composants métiers présentés dans [Bar04] : processus métier et entité métier.

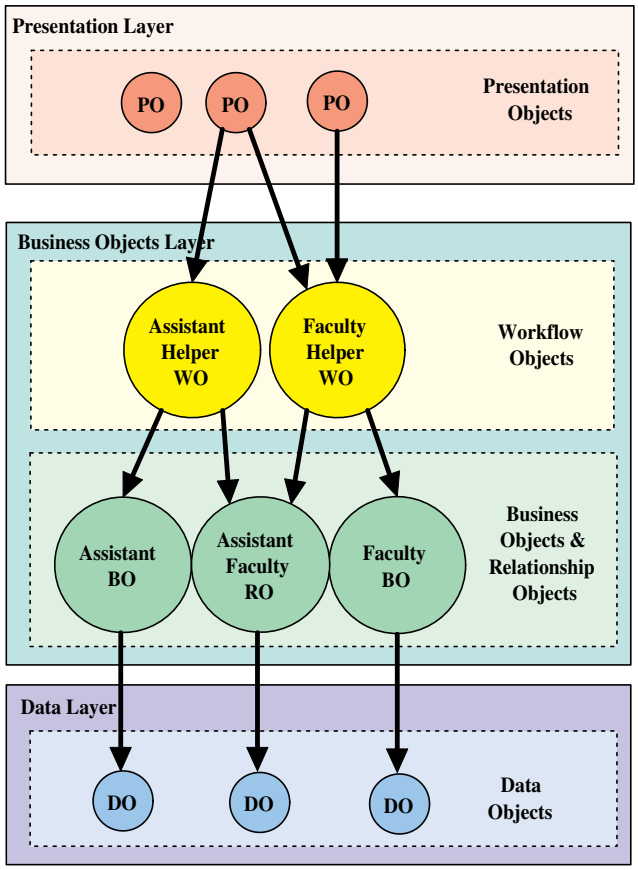

FIG. 3.5 – Les différents composants dans l'architecture multi-tiers d'OpenUSS.

– la dimension fonctionnelle : la plate-forme se décompose en couche fonctionnelle. Plus la couche est basse plus elle est dirigée vers les fondations de la plate-forme. Cette dimension intéresse davantage les utilisateurs de la plate-forme (administrateurs, concepteurs, fournisseur de ressources pédagogiques).
En exemple, ce découpage s'effectue en deux niveaux sur Cybeosphere II : noyau et composants fonctionnels. Sur OpenUSS le découpage est également sur deux niveaux : *Foundation Components* et *Plugable Components* (Figure 3.4).

[58] *Enterprise Java Bean*

Les deux dimensions sont orthogonales : un composant des couches fonctionnelles hautes peut se décomposer en composants d'interface et composants d'application. D'ailleurs, le problème de l'ajout de composant réside dans la combinaison de ces deux dimensions.

On peut remarquer que les composants des couches « hautes » se réfèrent aux services/outils rendus traditionnellement par les plates-formes. Ce sont ces composants qui nous intéressent car ils concernent indirectement l'équipe de conception : ils fournissent les fonctionnalités à intégrer dans la conception des scénarios.

3.4 Bilan

Les plates-formes de formation à distance (ou LMS) sont des dispositifs de plus en plus utilisés et reconnus quant au support des différents besoins des enseignants et des apprenants. Les premiers LMS, apparus lors des années 1997-1999, se sont positionnés comme des outils permettant une gestion de la formation simplifiée et efficace, visant une réduction des coûts. Ces produits permettaient surtout une gestion du présentiel (gestion des sessions et des inscriptions) et la diffusion de cours asynchrones. Avec l'arrivée de nouvelles technologies synchrones, ces logiciels se sont ensuite enrichis de composants permettant l'interactivité (chat, classe virtuelle), permettant une individualisation des parcours et un meilleur suivi des apprenants. En 2001 sont apparus les LCMS visant à gérer la création de contenu. Actuellement, les éditeurs se consacrent entre autres à l'amélioration des systèmes de *reporting* permettant un meilleur suivi financier et humain de formation, ainsi que la gestion du capital humain et la gestion de la performance [CSC03]. Les plates-formes s'enrichissent continuellement afin de proposer davantage de fonctionnalités ; leur usage s'élargit de plus en plus de l'individualisation de l'apprentissage au support d'apprentissage coopératif (support de différents types d'apprentissage : enseignement traditionnel comme activités socio-constructivistes).

Des spécifications et des normes sont actuellement proposées par des organismes de la e-formation pour les LMS. Ces standards visent à permettre : une communication entre différents systèmes (interopérabilité) et la réutilisation des contenus en cas de changement de plate-forme. Toutefois, l'application de tous ces standards manque actuellement de maturité pour être réellement fiable : une interopérabilité efficace et complète entre les normes est limitée [CSC03].

La tendance actuelle, dans le domaine de la recherche pour les plates-formes, est de faire évoluer ces dispositifs vers des architectures composants. Les objectifs visés sont dans un premier temps d'appliquer les savoirs-faire technologiques (J2EE, EJB, Web Services par exemple) et fonctionnels (structuration en couches) en terme de composants logiciels traditionnels ; ceci afin d'étendre, dans un deuxième temps, les fonctionnalités et usages des LMS.

Pour chaque fonctionnalité rendue actuellement par les LMS, il existe d'autres systèmes éducatifs adaptatifs[59] basés Web la réalisant mieux mais ces systèmes ne sont généralement que des prototypes confinés dans des laboratoires et ne permettant pas de mener une activité d'apprentissage complète. Alors qu'une piste actuelle de recherche consiste à faire évoluer, communiquer et interopérer ces systèmes[60] de

[59] *Adaptative Web-Based Educational System* (AWBES).
[60] Le prototype est alors perçu comme un composant.

type hypermédias adaptatifs [Bru03, Ros03, Bru04], nous pensons qu'une seconde approche consiste à étendre et à enrichir les fonctionnalités des plates-formes grâce aux architectures composants.

Chapitre 4

Langages existants pour la description de situations d'apprentissage

Sommaire

Les deux premiers chapitres ont permis d'étudier en détail le contexte de nos travaux : étude des PBL (chapitre 2) et étude des plates-formes de formation à distance (chapitre 3). Nous rappelons que l'un de nos défis concerne la proposition d'un langage de modélisation dédié à l'élaboration de modèles pour aider à la scénarisation de PBL coopératives. Nous avons initialement limité notre approche en écartant tout travail d'ordre méthodologique comme l'étude sur les méthodes d'ingénierie pédagogique.

L'objectif de ce chapitre est de prospecter les langages existants actuellement pour décrire tout ou partie de scénarios de situations d'apprentissage. Le terme de langage désigne ici tout moyen d'expression permettant de décrire ces éléments. Dans ce cadre, nous avons étudié trois catégories de langages : les langages à base de métadonnées (section 4.1), les langages à base d'ontologies (section 4.2) et les langages de modélisation pédagogique (section 4.3).

4.1 Langages à base de métadonnées

Nous définissons dans un premier temps les caractéristiques et usages des méta-données (sous-section 4.1.1), puis les usages actuels des langages à base de méta-données pour décrire des scénarios d'apprentissage (sous-section 4.1.2).

4.1.1 Terminologie

Les métadonnées sont traditionnellement définies comme « des données décrivant d'autres données ».

Leur champ d'application est vaste. Dans la e-formation leur usage est croissant. En effet, les sources d'information disponibles et appropriées pour l'apprentissage sont de plus en plus nombreuses. Il devient donc de plus en plus difficile de trouver et d'assembler des informations pertinentes. Les métadonnées sont utilisées dans la e-formation parce qu'elles fournissent la capacité de décrire et d'identifier précisément les contenus. Ainsi, il devient possible de rechercher, d'assembler et de délivrer le « bon » contenu à la bonne personne et au bon moment. Les métadonnées ne s'appliquent pas pour autant uniquement au contenu des formations mais également à toutes les ressources mises en jeu : codes logiciel, équipements, ou encore individus et leur compétences. Elles peuvent êtes simples, objectives et invariables comme l'auteur d'un livre, la taille d'un fichier, l'emplacement d'un fichier dans une base de données. Elles peuvent également être complexes, subjectives et variables comme les préférences d'apprentissage ou les styles d'un apprenant.

Les métadonnées sont donc un moyen efficace pour complètement décrire et identifier tout contenu et ressource utiles pour l'apprentissage ; elles permettent, pendant la conception mais également l'exécution de l'apprentissage, de rechercher, sélectionner, retrouver, combiner, utiliser et modifier des ressources.

Les métadonnées ont quatre principales utilisations permettant une réduction des coûts et du temps et une amélioration des performances humaines :

Catégorisation : les métadonnées permettent d'organiser les informations en catégories. Ces catégories doivent être standardisées pour être efficaces.

Taxonomies : il s'agit de structurer et d'organiser les catégories de métadonnées en groupes ordonnés de relations. Ceci permet d'organiser les contenus et aussi de capturer les relations entre catégories. De cette manière, les taxonomies de différentes formations ou structures peuvent être reconnues, comprises et faire l'objet de correspondances avec d'autres taxonomies.

Ré-utilisation : lorsque les contenus et leurs métadonnées deviennent davantage structurés et que leur granularité décroît, la réutilisation du contenu augmente. Il est même possible de réutiliser des blocs d'information dans des buts complètement différents ou en les mettant dans des contextes différents en choisissant seulement pour ceux qui sont nécessaires, le bon médium.

Assemblage dynamique : l'information peut seulement être réutilisée selon son degré de flexibilité mais également assemblée dynamiquement dans le « bon matériau » pour la bonne personne, dans le bon format de média, dans le bon langage, délivré au bon endroit, sur le bon dispositif et au bon moment.

4.1.2 Métadonnées et conception de situations d'apprentissage

Les métadonnées permettent d'étiqueter les ressources pédagogiques enjeux de l'apprentissage en conception (appelées *objets d'apprentissage*).

Les métadonnées ont fait l'objet de nombreux travaux de standardisation : LOM d'IEEE ou encore la spécification IMS *Learning Resource Metadata* (voir la section consacrée aux normes et standards de la e-formation 3.2.4). Des organismes s'intéressent également à leur utilisation (exemple : CanCore[61]).

Certains travaux ont porté sur l'ajout d'informations didactiques aux objets d'apprentissage [All02]. Partant du constat que le standard LOM n'inclut aucune information sur les rôles pédagogiques joués par les objets d'apprentissage dans un cours, leurs travaux ont donc ajouté ces informations en se basant sur le concept de rôles et de relations pédagogiques.

Grâce aux métadonnées, on souhaite pouvoir repérer plus efficacement les diverses ressources éducatives sur l'Internet en facilitant leur recherche par descripteur ou marqueur. Mais l'objectif premier des métadonnées qui accompagnent les diverses ressources du Web c'est de permettre à divers logiciels ou systèmes dédiés à la formation en ligne de pouvoir « interpréter » la fiche descriptive d'une ressource et de pouvoir traiter la ressource conformément aux exigences ou particularités qui y sont énoncées [Cre02].

Plusieurs ensembles de métadonnées sont en cours de développement actuellement et visent plus spécifiquement certaines fonctions comme, par exemple, la description du profil de l'apprenant ou la désignation et l'assemblage des contenus ou objets pédagogiques.

L'utilisation des métadonnées s'adresse davantage au *fournisseur de ressources pédagogiques* qu'à l'enseignant ou l'ingénieur pédagogique impliqué dans la conception de situations d'apprentissage. Les métadonnées actuelles concernent la description des objets d'apprentissage manipulables dans diverses situations d'apprentissage comme dans nos PBL coopératives. Ces langages de description ne concernent toutefois pas directement les PBL pour faciliter leur analyse, ni plus généralement la conception avancée de scénarios d'apprentissage.

4.2 Ontologies éducatives

Les motivations ou besoins reconnus pour l'utilisation des ontologies sont [Col03] :

- avoir une représentation et une structure des concepts de façon explicite (externe et accessible) pour la machine et pour l'humain ;
- partager une représentation à facettes multiples pour une communauté plus ou moins spécialisée ;
- avoir une représentation indépendante partagée par des logiciels permettant ainsi leur interopérabilité (communication sur des concepts communs) ;
- avoir une représentation faisant référence ;
- avoir une représentation servant de banc de comparaison ;
- avoir une représentation guidant la conception des systèmes.

Les ontologies sont une réponse au besoin de guidage dans la conception de systèmes mais sont-elles adaptées pour faciliter cette conception ?

[61] http ://www.cancore.ca

Dans un premier temps nous définissons et caractérisons les ontologies, puis nous nous intéressons à l'étude des ontologies éducatives.

4.2.1 Ontologie

Il existe de nombreuses définitions d'une ontologie (philosophique, étymologique, systémique...) ; nous préférons proposer la définition formelle de Gruber : *une ontologie est la définition explicite d'une conceptualisation* [Gru93]. Une conceptualisation se définit quant à elle selon Guarino [Gua95, Gua98] comme (repris de [Col03]) :

- une structure formelle de la réalité telle que perçue et organisée par un agent indépendamment du vocabulaire utilisé et d'une situation particulière ;
- un ensemble de modèles d'un langage logique L qui décrivent les interprétations acceptables des symboles de L : une ontologie est une axiomatisation (peut-être incomplète) d'une conceptualisation.

Une ontologie est alors composée de concepts, de définitions de ces concepts, de liens hiérarchiques, de relations et d'axiomes formalisant les définitions et les liens/relations des concepts.

Une ontologie peut être considérée comme un méta-modèle [Miz00] : elle fournit des concepts et des relations qui sont utilisés comme des blocs de construction pour les modèles. Les axiomes contraignent la sémantique des concepts. Ainsi, une ontologie spécifie les modèles à construire en fournissant un guidage et des contraintes à satisfaire.

Les ontologies partagent les caractéristiques suivantes :

- leur indépendance des systèmes informatiques ;
- le consensus d'une communauté qu'elles représentent ;
- les différents points de vue exprimés (structurel, fonctionnel, types, propriétés, ...) ;
- leur stabilité dans le temps ;
- elles sont explicites.

Il est possible de catégoriser différentes ontologies selon leur niveau d'abstraction ou le niveau formel employé (semi-formel vs formel). Mizoguchi définit trois niveaux pour les ontologies [Miz00] :

Niveau 1 : une collection structurée de termes.

Niveau 2 : ajout au niveau 1 de définitions formelles des concepts et ajout de relations et contraintes formelles[62] définies également par un ensemble d'axiomes.

Niveau 3 : niveau d'exécution. Les modèles construits sur la base de l'ontologie s'exécutent en utilisant des modules fournis par les codes abstraits associés aux concepts dans l'ontologie.

4.2.2 Ontologie et conception de situations d'apprentissage

Les EIAH manipulent de nombreuses connaissances pouvant faire l'objet d'ontologies. Pour les enseignants, l'ontologie peut permettre de déterminer les connaissances enjeux d'apprentissage et de positionner les connaissances acquises par rapport aux connaissances en jeu. L'ontologie représente alors un consensus de la communauté enseignante (les programmes scolaires en sont un exemple). Pour les

[62] Pour pouvoir être interprétées par la machine

EIAH et les plates-formes de formation à distance plus particulièrement, il est également possible d'avoir une représentation de référence pour les fonctions ou services rendus par la plate-forme, les ressources manipulées, etc.

La littérature fait référence à des exemples d'ontologies employées dans les EIAH :

- pour la composition de document par association d'ontologies (ontologie du domaine et ontologie pédagogique) [Ran00, Ran02] ;
- pour : a) structurer des documents techniques (ontologie générique de document) ; b) analyser et segmenter en fragments les documents techniques structurés (ontologie du document technique) ; c) indexer les fragments (ontologie du domaine) ; d) rechercher des informations et construire des scénarios (ontologies pédagogiques) (dans le cadre d'un projet européen IMAT [Des02]).
- des ontologies pour le CSCL : ontologie des objectifs d'apprentissage pour un usage de formation de groupe opportuniste [Ina00] ; une ontologie du CSCL pour définir et créer des environnements CSCL, analyser et identifier des caractéristiques de collaboration et guider les activités de CSCL [Bar02].

4.2.2.1 Le projet IMAT

Notre intérêt concerne l'usage des ontologies pour scénariser des apprentissages. Dans les travaux du projet IMAT, plusieurs ontologies pédagogiques sont utilisées pour la scénarisation :

- objectifs d'apprentissage en maintenance (Figure 4.1) ;
- types de description en maintenance ;
- stratégies pédagogiques ;
- type de connaissance ;
- activités pédagogiques (Figure 4.2) ;
- utilisation de ressource pédagogique.

L'outil IMAT de scénarisation de séquence pédagogique a pour objectif de faciliter la conception par des non spécialistes de séquences pédagogiques. Il permet d'extraire de la base de fragments IMAT des éléments pour chacune des activités/actions de la séquence. Cet outil, basé sur les ontologies IMAT, fonctionne ainsi :

- l'auteur explicite progressivement ses choix pédagogiques, en sélectionnant des concepts dans les ontologies pédagogiques ;
- l'outil le guide dans sa construction : les relations entre les concepts des différentes ontologies restreignent les choix possibles de l'auteur en fonction des choix précédents ;
- l'outil lui propose des fragments adaptés à chaque activité de la séquence en fonctions des ontologies les qualifiant.

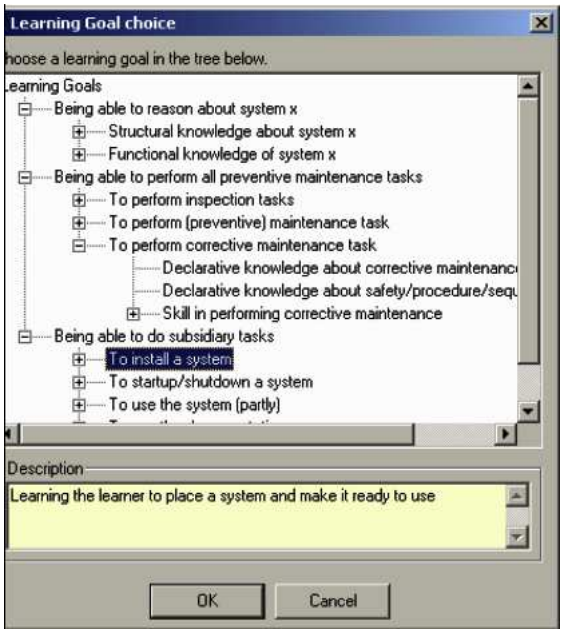

FIG. 4.1 – Ontologie des objectifs d'apprentissage du projet IMAT.

Activity		
Setting goals	Instructor:	Present the 'current' learning goal
	Learner:	Comprehend and accept learning goal
Illustration	Instructor:	Present illustration
	Learner:	Study illustration; Elaborate
Presentation	Instructor:	Present the 'content'
	Learner:	Memorise (learn by heart) content presented
Explanation	Instructor:	Provide explanation
	Learner:	Read, Listen to explanation; Study explanation; Relate explanation to existing knowledge
Asking questions	Instructor:	Construct and ask questions
	Learner:	Comprehend question, Recall content or reason and answer question
Providing workspace	Instructor:	Provide workspace
	Learner:	Use workspace
Instruction	Instructor:	Give instructions
	Learner:	Comprehend instructions and memorise them
Demonstration	Instructor:	Show how a task has to be done
	Learner:	Observe how a task is done and memorise
Presenting tasks	Instructor:	Give assignments
	Learner:	Comprehend assignments and doing them
Providing examples	Instructor:	Give example
Providing references	Instructor:	Provide references
	Learner:	Act on references
Summarising	Instructor:	Present a summary

FIG. 4.2 – Ontologie des activités pédagogiques du projet IMAT.

4.2.2.2 Le système *Learning Design Palette*

Il s'agit d'un système-auteur basé sur l'exécution d'ontologies (*ontology-aware*) [Ina04]. Les utilisateurs ciblés par cet outil sont les concepteurs d'apprentissage (*Learning Design*) : enseignants, instructeurs, concepteurs de cours ou même apprenants auto-dirigés.

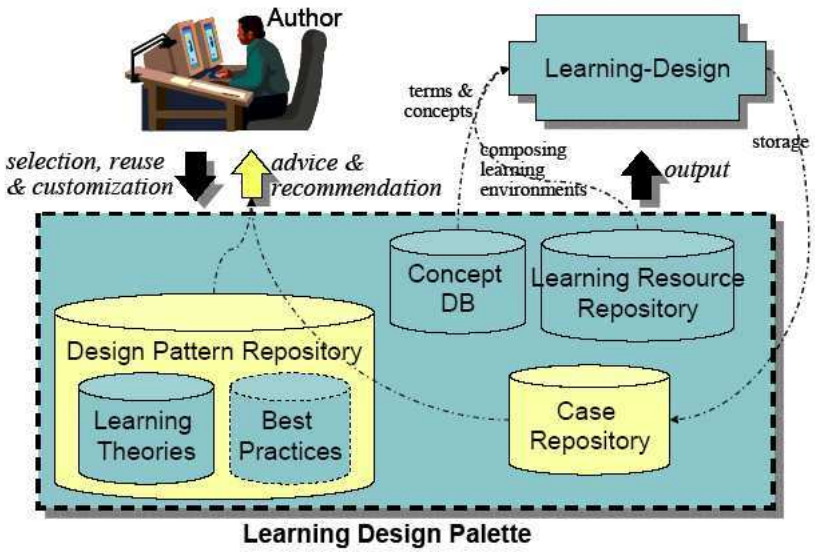

FIG. 4.3 – Le système-auteur *Learning Design Palette* basé ontologies.

Cet outil est conçu pour réduire le coût de création de scénarios d'apprentissage privilégiant l'apprentissage collaboratif. L'architecture de l'outil est la suivante (Figure 4.3) :

- Le *Concept DB* représente le vocabulaire pour représenter des *Learning Design*.
- Le *Design Pattern Repository* rassemble des structures partielles de conception de deux types : a) inspiré par les théories d'apprentissage, et b) issues des *Best Practices* d'autres concepteurs/enseignants.
- Le *Learning Resource Repository* contient les ressources d'apprentissage : matériaux et outils avec leurs métadonnées.
- Le *Case Repository* contient les *Learning Designs* de type scénario d'apprentissage et la description de leurs configurations éducatives et techniques.

Ce système-auteur interprète les spécifications SCORM et LOM pour rechercher, réutiliser, et recommender des ressources existantes aux auteurs.

La création d'un *Learning Design* avec le système-auteur est réalisée par le biais de nombreuses interfaces ; grâce à ces interfaces, l'utilisateur manipule des boites de dialogue dynamiques dissimulant l'utilisation et l'exécution des ontologies.

Cet outil-auteur pour la spécification de scénarios d'apprentissage propose également une méthode, sous la forme d'un ensemble de phases de spécification, pour guider les utilisateurs (Figure 4.4) ; par exemple, la première phase (plan supérieur de la figure 4.4) concerne la spécification du processus global d'apprentissage et de ses objectifs ; la phase suivante (plan inférieur) précise la structure du scénario en terme d'activités d'apprentissage, d'environnement d'apprentissage et de l'organisation des apprenants

(apprentissage collaboratif ou individuel) ; ainsi, chaque nouvelle phase affine la spécification du scénario vis-à-vis du plan précédent.

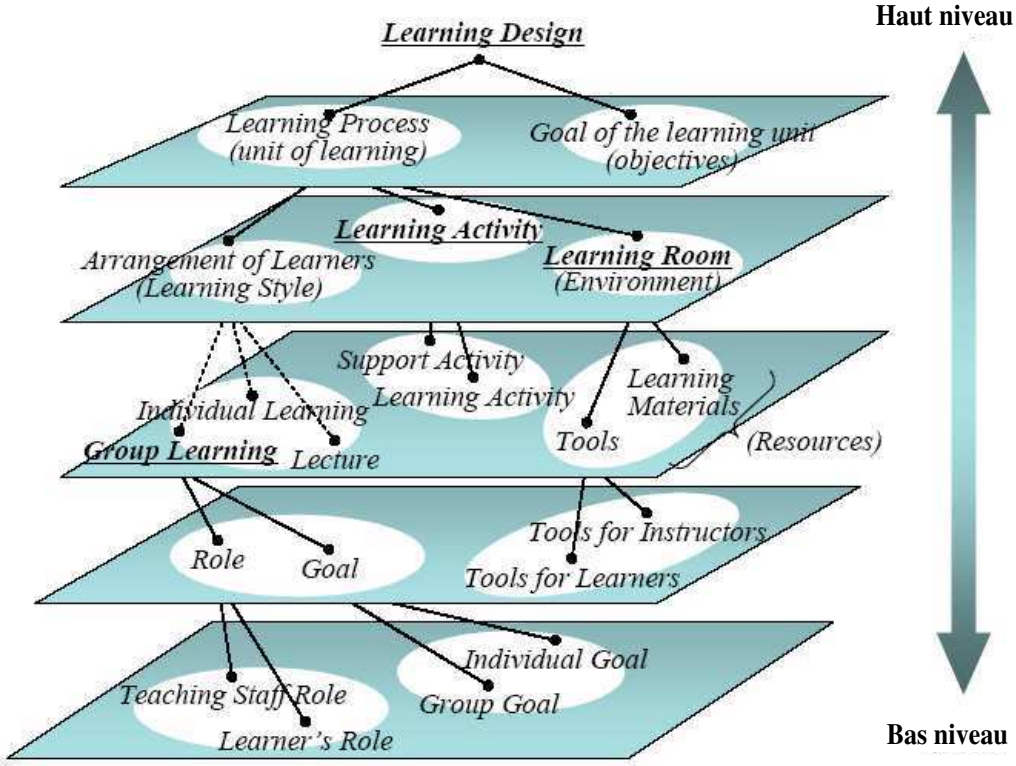

FIG. 4.4 – Les phases de spécification du *Learning Design Palette*.

4.2.3 Bilan

Nous avons étudié dans cette section les ontologies comme un langage possible pour aider notre équipe de conception à spécifier des situations d'apprentissage de type PBL. Une ontologie du domaine des PBL est envisageable pour proposer une conceptualisation du domaine nous intéressant. Cette ontologie permettrait alors d'identifier les différents éléments complexes de nos situations d'apprentissage spécifiques (rôles, activités, objectifs, etc.).

Toutefois, le problème de l'outillage et de la méthode pour le guidage se pose. Des outils comme *Protégé-2000*[63] permettent la création d'interfaces adaptées pour permettre à l'utilisateur de préciser les caractéristiques des concepts qu'il a identifiés grâce à l'ontologie. Cependant, l'outil est trop spécifique pour servir de système-auteur. Nous avons vu avec le *Learning Design Palette* qu'il est possible de bâtir des systèmes auteurs plus adaptés basés sur l'utilisation d'ontologies. Toutefois, ce type d'outil demande un effort de développement très important. Concernant la méthode de guidage, notre expérience avec *Protégé-2000* ([Mot03]) a mis en évidence la difficulté de garantir pour un non-expert de l'outil qu'il

[63] http ://protege.stanford.edu/

aboutira à une conception précise de sa situation d'apprentissage. Encore une fois, un outil-auteur adapté permettrait de pallier ce défaut.

Comme langage de conception de situations d'apprentissage en général, les langages ontologiques correspondent davantage au support et au guidage d'une conception avancée dans laquelle la représentation externe et explicite de l'apprentissage, intégrée dans l'ontologie, permettrait de formaliser toute situation d'apprentissage particulière ; cette modélisation formelle a l'avantage d'être basée sur des concepts et relations partagés par l'équipe de conception mais surtout d'être interprétable par la machine.

Une ontologie semi-formelle du domaine des PBL pourrait permettre d'assister la conception dans des phases plus amont comme l'analyse voire même l'expression initiale des besoins. Toutefois, l'inconvénient principal repose sur la difficulté d'élaboration de ces ontologies : une ontologie représente un consensus qui doit être partagé par les différentes équipes de conception et les acteurs qui les composent.

Ces ontologies sont elles stables ? Peuvent elles évoluer, s'adapter aux concepteurs tout en garantissant une base minimale de consensus permettant toujours l'interopérabilité et la réutilisation de situations d'apprentissage provenant d'autres équipes, d'autres projets ? Si de multiples ontologies plus ou moins abstraites peuvent guider les différentes étapes de conception nous intéressant, alors doit-on obtenir des modèles de l'évolution de la situation d'apprentissage en conception sans inter-relation ? Finalement, peut-on automatiser et guider le processus de conception avec des ontologies ?

4.3 Langages de modélisation pédagogiques

Depuis quelques années, les travaux de recherche ont montré que les plates-formes de formation à distance ne peuvent se contenter de seulement mettre à disposition des ressources d'apprentissage. Des langages de modélisation pédagogiques (ou *éducatifs* selon les auteurs) sont apparus dans le but de gérer les ressources mais surtout les activités pédagogiques les utilisant. Ces langages se focalisent davantage sur l'utilisation des ressources que sur les ressources elles-mêmes.

4.3.1 Introduction

Une étude à l'initiative du CEN/ISSS WS/LT (Workshop on learning Technology) [Raw02] avait pour but d'analyser, étudier et comparer des candidats au titre de *Educational Modeling Language* (EML). La définition d'EML utilisée à travers ce projet et utilisée comme définition de référence maintenant est la suivante :

« *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*** » [Raw02]

Plus simplement, Rob Koper parle de *notation sémantique pour des unités d'apprentissage utilisées dans le cadre du E-learning* (traduit de [Kop02]). 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). Le concept d'unité d'apprentissage (***unit of learning*** aussi appelée *unit*

of study ou *unit of instruction*) représente la plus petite unité définissant un modèle d'apprentissage (accompagnée des ressources et des services requis pour atteindre un ou plusieurs objectifs d'apprentissage). Cette granularité minimale est atteinte lorsqu'il n'est pas possible de réduire l'unité à une sous-partie de ses composantes internes sans perdre la sémantique et l'efficacité recherchée vis-à-vis des objectifs pédagogiques. Ces unités d'apprentissage peuvent correspondre, par exemple, à des cours, à des modules de formation, à des leçons comme à une PBL. Un EML a également la particularité de pouvoir décrire ces unités d'apprentissage selon une **perspective pédagogique** particulière : ceci implique donc que l'EML soit relativement indépendant des diverses théories d'apprentissage. Pour terminer l'explication de la définition d'un EML, les notions de **réutilisation** et d'**interopérabilité** impliquent que les modèles produits avec les EML ne doivent pas contenir d'éléments contextuels (comme le type des médias utilisés) afin de les rendre réutilisables pour différentes configurations/choix d'exécution et interprétables par différentes plates-formes LMS.

Six candidats ont été étudiés par l'étude (CDF, EML-OUNL, LMML, PALO, Targeteam, TML/Netquest). Seuls deux (PALO et EML-OUNL) ont retenu particulièrement l'attention. Cette étude a conclu que le langage EML-OUNL correspondait au mieux à cette définition (PALO se limitant à la description de tâches individuelles). EML-OUNL a depuis fait l'objet d'une intégration dans les travaux de standardisation engagés par le consortium IMS ; le résultat a donné lieu à la spécification *IMS-Learning Design* (IMS-LD), déjà évoquée en 3.2.4.2.

Dans les prochaines sous-sections nous présentons le langage EML-OUNL (4.3.2), puis l'intégration réalisée de ces travaux dans le standard IMS-LD (4.3.3). A ce propos, nous remarquons que certaines illustrations des travaux d'EML-OUNL intégrés et modifiés dans la spécification IMS-LD sont parfois reprises dans la littérature comme illustration d'EML-OUNL. Ceci n'est pas erroné mais pour étudier et synthétiser l'ensemble des travaux portants sur les EML, nous avons choisi dans les sous-sections suivantes de présenter EML-OUNL tel qu'il l'était avant son intégration dans IMS-LD, puis de présenter la spécification d'IMS-LD et donc aussi les travaux plus avancés d'EML-OUNL. Dans la dernière section (4.3.4) sont présentés les travaux de l'équipe du LICEF portant sur l'élaboration d'une méthode d'ingénierie éducative. Bien que nous ayons écarté dès le départ les travaux d'ordre méthodologique, les travaux autour de MISA ont beaucoup évolué ces dernières années et cherchent maintenant à s'intégrer dans la définition des EML.

4.3.2 EML-OUNL

EML-OUNL[64] a été développé à l'*Open University of the Netherlands*[65] dans le cadre de l'utilisation de la e-formation[66]. Après 20 années d'existence, cette université ouverte est maintenant reconnue comme la spécialiste de l'auto-apprentissage et comme experte en e-formation ; son budget s'élève à 57.1 millions d'euros pour 791 membres du personnel et 21 004 étudiants actifs [Ope04].

[64]Le terme d' « EML » a été créé à l'origine pour représenter les travaux réalisés à l'université ouverte des Pays-Bas. Puis, EML a désigné le concept générique de langage de modélisation éducative. Depuis, pour distinguer les deux, on désigne l'EML de l'OUNL par EML-OUNL.

[65]Site web http ://www.ou.nl/

[66]Site web http ://eml.ou.nl/eml-ou-nl.htm

Ce cadre *industriel* justifie les besoins et les enjeux de la modélisation de scénario pédagogique pour des formations visant un très large public. La spécification d'EML-OUNL s'adresse ainsi à des nouveaux acteurs, experts dans la formation à distance, comme l'ingénieur pédagogique défini en section 1.2.2.4.

4.3.2.1 Pré-requis pour le modèle d'information d'EML-OUNL

Les onze pré-requis que le langage EML-OUNL devait atteindre étaient [Kop00, Raw02] :

1. Formalisation : il doit décrire les unités d'apprentissage de manière formelle, de manière à ce qu'un processus d'automatisation soit possible.
2. Flexibilité pédagogique : il doit être capable de décrire des unités d'apprentissage basées sur des théories et des modèles d'apprentissage/enseignement différents.
3. Méthode d'instruction explicite (dans [Kop00]) ou typage explicite des objets d'apprentissage (dans [Raw02]) : il doit explicitement exprimer la sémantique des différents objets d'apprentissage dans le contexte d'une unité d'apprentissage. Il doit fournir la structure sémantique des contenus ou fonctionnalités des objets d'apprentissage typés dans une unité d'apprentissage.
4. Complétude : il doit être capable de décrire entièrement une unité d'apprentissage, incluant tous les objects d'apprentissage typés, les relations entre ces objets et les activités, et le flot de travail (*workflow*) de tous les étudiants et membres de l'équipe d'encadrement avec les objets d'apprentissage.
5. Reproductibilité : il doit décrire un modèle d'apprentissage abstrait de telle manière qu'il permette une répétition de l'exécution dans différentes configurations et avec différentes personnes.
6. Personnalisation : il doit être capable de décrire des aspects personnalisés, afin que le contenu et les activités situés à l'intérieur d'une unité d'apprentissage puissent être adaptés en se basant sur les préférences, le dossier personnel, les connaissances antérieures, les besoins éducatifs et les usages contextuels des utilisateurs. De plus, le contrôle du processus d'adaptation doit être donné, selon le cas, à l'étudiant, à un membre de l'équipe pédagogique, à l'ordinateur ou à un concepteur.
7. Indépendance du médium et du cadre d'utilisation : il doit décrire le contenu des ressources indépendamment du médium utilisé afin que l'unité d'apprentissage puisse être réutilisée pour des formats différents de publication (exemples : Web, papier, e-books, mobile, etc.) et aussi pour des cadres d'utilisation différents (enseignement/apprentissage à distance, apprentissages mixtes,...).
8. Interopérabilité : conserver autant que possible une indépendance entre les standards utilisés pour la notation des unités d'apprentissage et les techniques utilisées pour interpréter la notation des unités d'apprentissage.
9. Compatibilité : il doit utiliser les différents standards existants lorsque cela est possible.
10. Réutilisabilité : il doit permettre d'identifier, d'isoler, de dé-contextualiser et réutiliser les modèles dans d'autres contextes.
11. Cycle de vie : il doit rendre possible de produire, modifier, conserver, distribuer et archiver des unités d'apprentissage ainsi que tous les objets d'apprentissage contenus.

4.3.2.2 Le méta-modèle pédagogique d'EML-OUNL

Le système de notation d'EML-OUNL est basé sur un vocabulaire pédagogique commun aux différents modèles pédagogiques possible de décrire. Comme l'indique Rob Koper [Kop00], la difficulté est de trouver un bon niveau d'abstraction du langage EML-OUNL : un niveau d'abstraction trop haut n'aurait rien révélé lors de la conception de modèles d'apprentissage ; un niveau trop bas/concret aurait signifié que le système de notation aurait été spécifique et donc aurait seulement permis de décrire un assortiment de modèles d'unités d'apprentissage. Ainsi, le choix a été fait de concevoir un méta-modèle d'instruction (*instructional meta-model*) basé sur une analyse des théories éducatives existantes dans la littérature et d'autres travaux (voir [Kop00] p.27).

Ce méta-modèle est exprimé sous la forme d'une structure statique décrite par des diagrammes UML. D'un point de vue conceptuel, le méta-modèle pédagogique se décompose en quatre paquetages :

1. Le modèle d'apprentissage : il décrit comment les apprenants apprennent selon un consensus sur les théories d'apprentissage (le méta-modèle se veut *pédagogiquement neutre*).

   La Figure 4.5 décrit un résumé de ce modèle. Ce modèle est basé sur différents axiomes compatibles selon leur auteur avec toutes les théories d'apprentissage [Kop01].

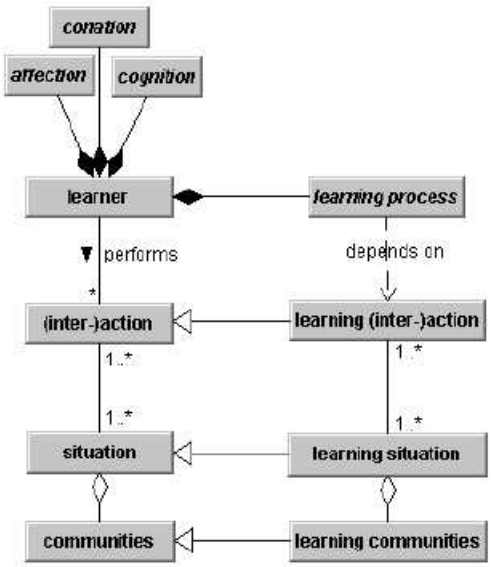

FIG. 4.5 – Le modèle d'apprentissage.

2. Le modèle de l'unité d'apprentissage : il décrit comment les unités d'apprentissage, applicables dans la pratique, sont modélisées, selon le modèle d'apprentissage et le modèle d'instruction donnés (Figure 4.6). L'unité d'apprentissage représente le résultat d'un processus de conception d'apprentissage. Cette unité doit prendre en compte :
   - les rôles de *learner* et *staff* dans le processus d'apprentissage,
   - les objectifs d'apprentissage et le groupe cible,
   - les pré-requis sur les apprenants,
   - d'autres caractéristiques sur les apprenants (style d'apprentissage, préférences, etc.),
   - le domaine d'apprentissage (e.g. mathématiques),

– le contexte de l'apprentissage (apprentissage à distance ou « mixte », etc.),
– l'évaluation de l'apprentissage.

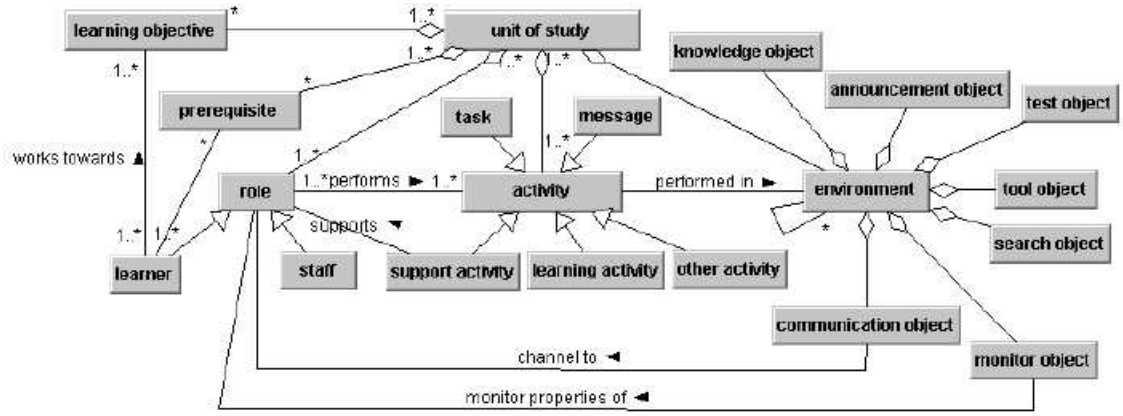

FIG. 4.6 – Le modèle de l'unité d'apprentissage.

3. Le modèle du domaine : il décrit le type des contenus et son organisation. Chaque domaine a sa propre structuration des savoirs, aptitudes et compétences ce qui donne souvent un modèle pédagogique spécifique pour ce domaine.
4. Théorie d'apprentissage/enseignement : il décrit les théories, principes et modèles d'apprentissage et enseignement tels qu'il sont décrits dans la littérature ou tels qu'ils sont conçus dans l'esprit des praticiens.

La Figure 4.7 donne une vue globale de l'ensemble des modèles composants le méta-modèle. Dans cet exemple, le focus est sur le rôle de l'apprenant uniquement. Cet exemple permet de représenter les dépendances entre les différents paquetages dans le méta-modèle. On voit également dans ce modèle que les termes *activity* et *environment* sont reliés aux termes *actions* et *situation*. Ceci exprime en fait la différence entre le modèle de l'unité d'apprentissage qui concerne la conception de processus d'apprentissage (c'est-à-dire la planification d'activités) et le modèle d'apprentissage qui désigne la façon dont l'apprentissage prend place en réalité.

4.3.2.3 Implémentation du méta-modèle dans un schéma XML

Le méta-modèle précédent a été implémenté dans un schéma XML (selon la recommandation XSD du W3C). Un exemple de la structure de base du langage EML-OUNL est montré sous la forme d'un arbre dans la Figure 4.8. Il ne s'agit que d'une sélection d'éléments et relations sans attributs tirée de [Kop01]. On peut remarquer dans cette figure une structure d'arbre du canevas (*framework*) contenant les objets d'apprentissage. Tous les objets d'apprentissage sont mentionnés derrière le nom de leur type. Les relations entre objet d'apprentissage sont exprimées en conformité aux règles de XML : « ? » signifie *optionnel* ; « * » signifie *zéro ou plus* ; « + » signifie *un ou plus*. Du point de vue de la notation, les listes séquentielles d'éléments sont représentés par les traits rectilignes tandis que les sélections d'un seul élément sont représentés par les traits en diagonale.

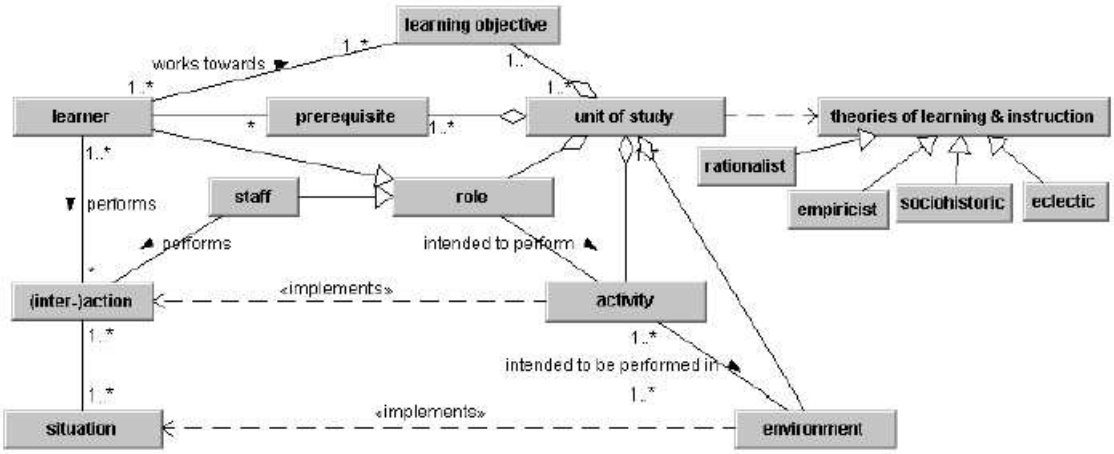

FIG. 4.7 – Une illustration du modèle intégré.

4.3.3 IMS-Learning Design

Le travail sur les EML a été intégré dans la spécification *IMS Learning Design* (IMS-LD). Il s'agit de la plus importante initiative en date ayant pour but d'intégrer des préoccupations de conception éducative (*Instructional Design*[67]) dans le mouvement des standards internationaux. La spécification d'IMS-LD [IMS03c] repose sur un modèle d'information de conception d'apprentissage. Celui-ci intègre donc les travaux d'EML-OUNL mais également d'autres spécifications existantes du consortium IMS : *IMS Content Packaging* [IMS03a], *IMS Meta-Data* [IMS01] et *IMS Simple Sequencing* [IMS03d].

Le méta-modèle d'information d'IMS-LD se décompose en trois composants principaux (Figure 4.9) :

- Le modèle conceptuel : il représente le vocabulaire, les relations entre les concepts et les relations avec la spécification *IMS Content Packaging*.
- Le modèle d'information : il décrit les éléments d'IMS-LD sur trois niveaux :
  - le niveau A décrit le vocabulaire au coeur de la spécification afin de concevoir des unités d'apprentissage supportant diverses pédagogies ;
  - le niveau B ajoute au niveau A de nouveaux éléments permettant une personnalisation ainsi que des scénarios et des interactions plus élaborées ;
  - le niveau C ajoute un nouveau mécanisme de notification permettant d'étendre la dynamique des scénarios construits.

  La description de chacun des trois modèles d'informations s'appuie sur une version restrictive du modèle conceptuel précédent selon le niveau concerné.
- Le modèle de comportement : il décrit un ensemble de comportement d'exécution (*runtime*) que des systèmes délivrant les unités d'apprentissage devront implémenter.

[67] De nombreux termes sont employés dans la littérature américaine comme européenne : « Instructional Design », « Instructional System Design », « Instructional Science », « Pédagogie Scientifique », « Ingénierie Pédagogique », « Ingénierie Éducative ».

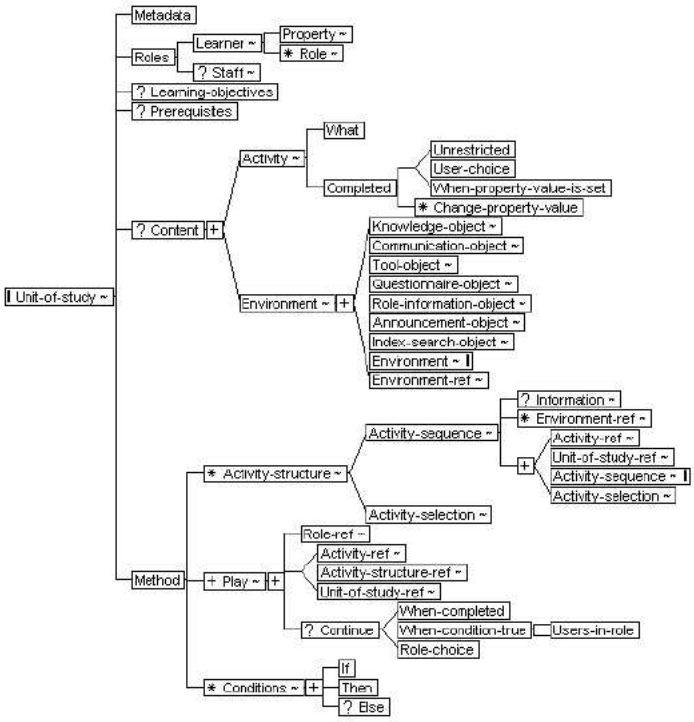

FIG. 4.8 – Implémentation (*binding*) de la structure de base du canevas contenant les objets d'apprentissage dans un schéma XML.

4.3.3.1 Le modèle conceptuel

L'objectif principal de la spécification IMS-LD est de proposer un canevas retenant tous les éléments pouvant décrire toute conception de processus d'apprentissage-enseignement de manière *formelle*. La spécification est basée sur un cahier des charges [IMS03c] reprenant une partie des pré-requis d'EML-OUNL : complétude, flexibilité pédagogique, personnalisation, formalisation, reproductibilité, interopérabilité, compatibilité, réutilisabilité.

Le modèle conceptuel est exprimé sous la forme d'un ensemble de diagrammes de classe UML et est accompagné de définitions précises du vocabulaire utilisé. Le modèle conceptuel est composé du modèle d'agrégation, du modèle de structure et du modèle représentant l'intégration des conceptions d'apprentissage avec le *IMS Content Packaging* [IMS03a] pour obtenir une « unité d'apprentissage » (voir la figure 4.9). Seuls les deux premiers modèles sont étudiés ; le troisième étant au delà de nos préoccupations.

Le modèle d'agrégation Cette vue du modèle conceptuel, illustrée par la Figure 4.10, ne représente que les relations d'agrégation (ce qui inclut celles de composition) et de spécialisation pour les classes abstraites (leur nom est en *italique* dans la figure).

Cette vue exprime le fait que tout modèle de conception d'apprentissage fournit une collection de ressources d'un coté et de l'autre les intègre dans une *method* spécifiant les aspects dynamiques de l'apprentissage (scénario). On peut analyser ce modèle d'agrégation selon deux niveaux d'agrégation :

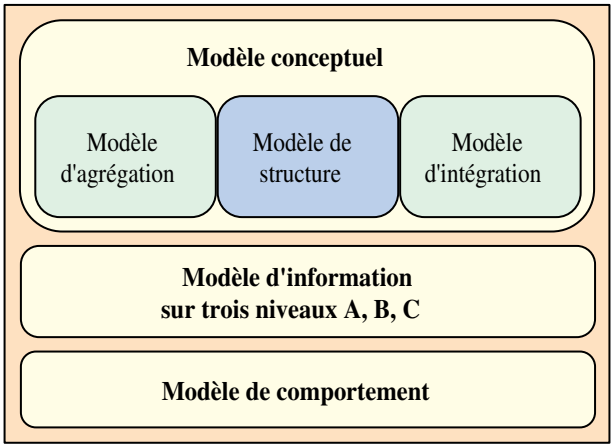

FIG. 4.9 – Décomposition du méta-modèle d'IMS-LD.

Composant	Correspondance avec un type de ressource	Étape(s) pour la correspondance
Role	Person	instantiation ou exécution (*runtime*)
Objective/prerequisite	Web content	conception, instantiation ou exécution
Objective/prerequisite	Imsld content	conception
Property	Dossier	instantiation
Learning object	Web content	conception, instantiation ou exécution
Learning object	Imsld content	conception
Service	Service facility	conception, instantiation ou exécution
Activity	Web content	conception, instantiation ou exécution
Activity	Imsld content	conception

TAB. 4.1 – Étapes de correspondance entre composants et ressources d'IMS-LD

- niveau 1 (le plus haut) : la conception d'un apprentissage concerne la définition d'un *learning design* comme une collection de composants, d'objectifs/pré-requis et d'une méthode.
- niveau 2 : les composants précédents s'agrègent en ressources ; la méthode en *plays*, conditions, et *notifications*.

Un composant peut être de sept types différents : rôle, propriété de groupe, propriété, structure d'activité, activité, environnement, et *outcome*[68]. Les correspondances entre composants et ressources peuvent s'effectuer à plusieurs moments de la conception d'un apprentissage :

[68] À l'exception de *outcome*, tous les autres concepts font partie du modèle d'information d'IMS-LD ; *outcome* n'est utilisé que conceptuellement.

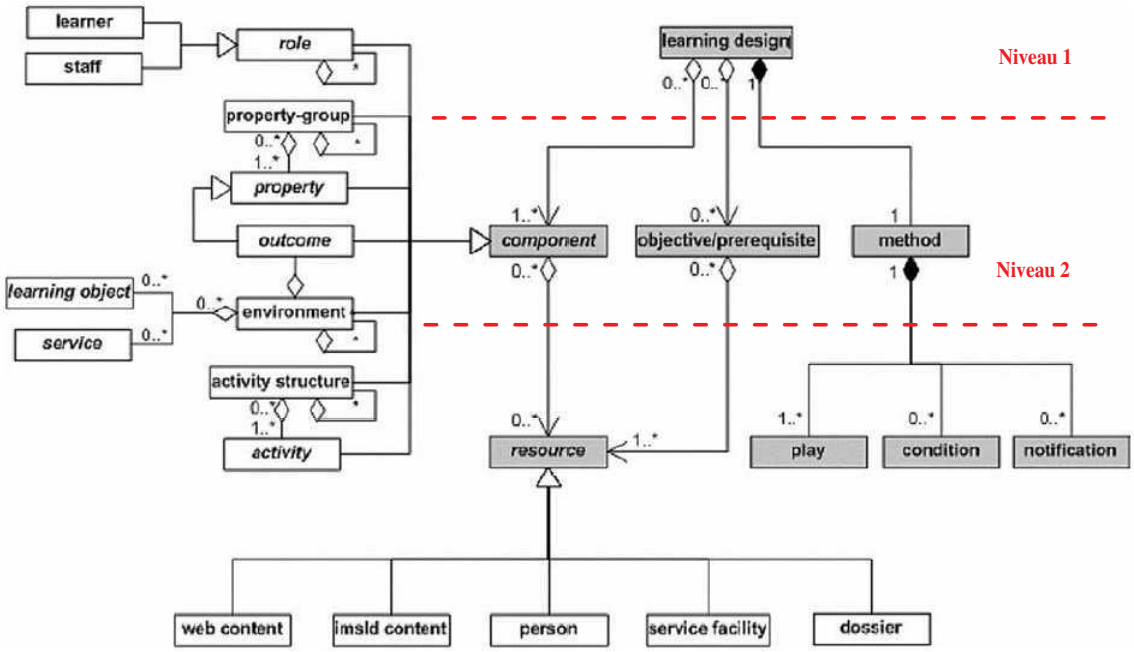

FIG. 4.10 – Les niveaux d'agrégation sémantique d'IMS-LD (tirée de [IMS03c] puis annotée).

Le modèle conceptuel de la structure d'un Learning Design Il s'agit d'une autre vue du modèle conceptuel global (Figure 4.9), celle-ci mettant l'accent sur les relations de fonctionnalités entre les classes (Figure 4.11).

Le concept au coeur de cette spécification pour la conception d'apprentissage est qu'une personne (*person*) joue des rôles (*role*) dans un processus d'apprentissage/enseignement (typiquement, un rôle d'apprenant *learner* ou de personnel *staff*). En jouant ce rôle la personne cherche à aboutir à des résultats (*outcomes*) en réalisant des activités d'apprentissage (*learning activity*) ou de soutien (*support activity*) plus ou moins structurées (par le biais des structures d'activités *activity-structure*). Ces activités se réalisent dans le contexte d'un environnement (*environment*). Cet environnement est constitué d'objets d'apprentissage (*learning object*) et de services (*services*) qui seront utilisés pendant la réalisation des activités associées à cet environnement. La méthode (*method*) permet de déterminer quel rôle assurera quelles activités à tel moment du processus. Le mécanisme de notification (*notification*) permet également de préciser certaines structures du processus de l'apprentissage ; ce mécanisme appartient au niveau C. La méthode est conçue pour atteindre des objectifs d'apprentissage (*learning objectives*) qui correspondent à la spécification de résultats pour l'apprenant. La méthode est décrite sur l'hypothèse de certains pré-requis (*prerequisites*) correspondant alors à la spécification des pré-requis d'entrée pour les apprenants.

La méthode est constituée d'une ou plusieurs pièces de théâtre (*play*) concurrentes (leur réalisation est en parallèle). Une *play* est constituée d'un ou plusieurs actes (*act*) séquentiels cette fois-ci. Chaque acte fait référence à un ou plusieurs *role-part* concurrents également ; ces *role-part* permettent d'associer un rôle avec une activité ou à une structure d'activité.

Au niveau B, une méthode peut contenir des conditions (*condition*) ou règles (par exemple du type *If-Then-Else*) qui permettent alors de raffiner la visibilité des activités et des environnements pour

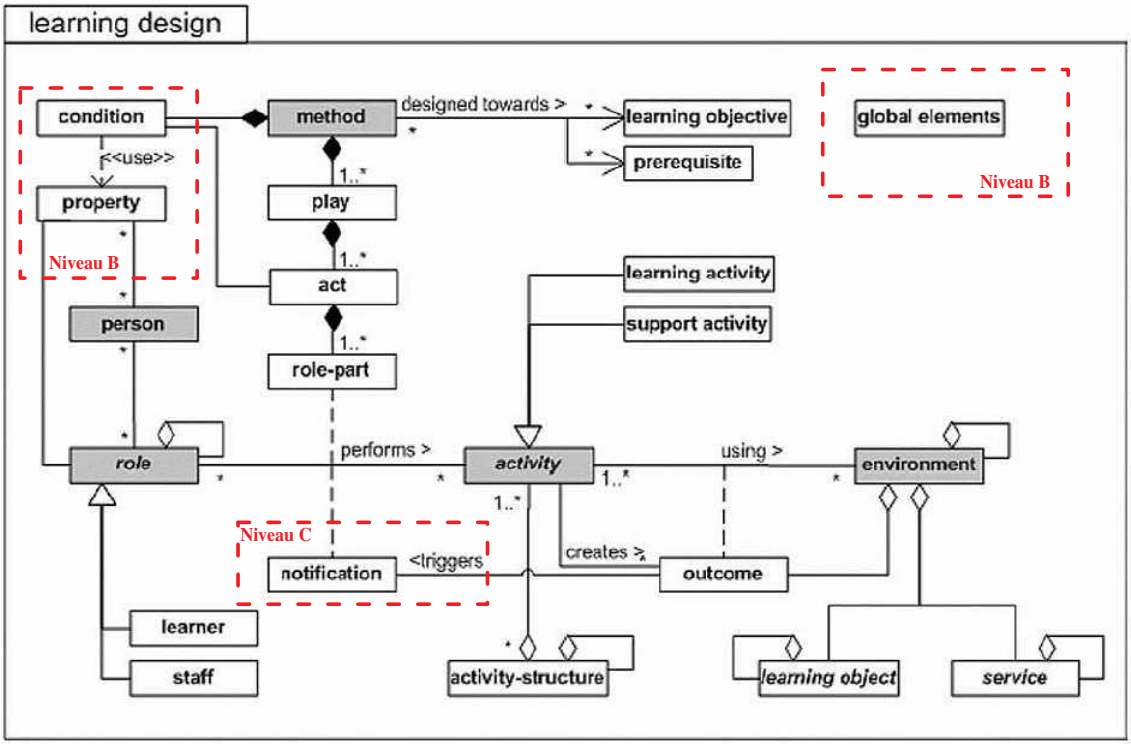

FIG. 4.11 – Modèle conceptuel de la structure de tout Learning Design.

les personnes ou rôles. Ces conditions définissent alors des expressions booléennes sur les propriétés de ces personnes ou rôles. Une propriété peut être regroupée dans des *property-groups* qui sont un exemple de propriétés globales (*global elements*) pouvant être définies. Les propriétés peuvent être de différents types, locales comme globales pour les personnes comme pour les rôles.

Une *notification* peut être générée par un résultat et peut ainsi créer une nouvelle activité disponible pour un rôle donné (la personne recevant la notification n'est pas forcément celle qui l'a émise). Par exemple, si un apprenant termine une activité (ce qui correspond au fait d'avoir atteint un résultat), alors un autre apprenant ou bien un enseignant peut se voir attribuer une nouvelle activité en conséquence. Ce mécanisme peut aussi être utilisé pour des conceptions d'apprentissage dans lesquelles la ressource d'une activité est dépendante du type de résultats obtenus à des activités précédentes (utiles pour des apprentissages coopératifs comme nous envisageons les PBL).

Deux rôles explicites sont spécifiés dans IMS-LD : *learner* et *staff*. Ces rôles peuvent être spécialisés en sous-rôles d'après la spécification d'IMS-LD mais cela reste à la charge du concepteur d'apprentissage. Nous avons vu que les activités pouvaient être assemblées dans des structures. Celles-ci sont alors une agrégation d'un ensemble d'activités dans une unique structure ; les activités sont alors associables à un rôle grâce au *role-part*. Une structure peut modéliser une séquence d'activités, qui devront toutes êtres complétées dans l'ordre fourni, ou bien une sélection d'activités pour laquelle le rôle associé décidera du nombre d'activités qu'il voudra réaliser. Les structures d'activités peuvent également faire référence à d'autres structures comme à d'autres unités d'apprentissage externes.

Les environnements peuvent contenir deux types de base :

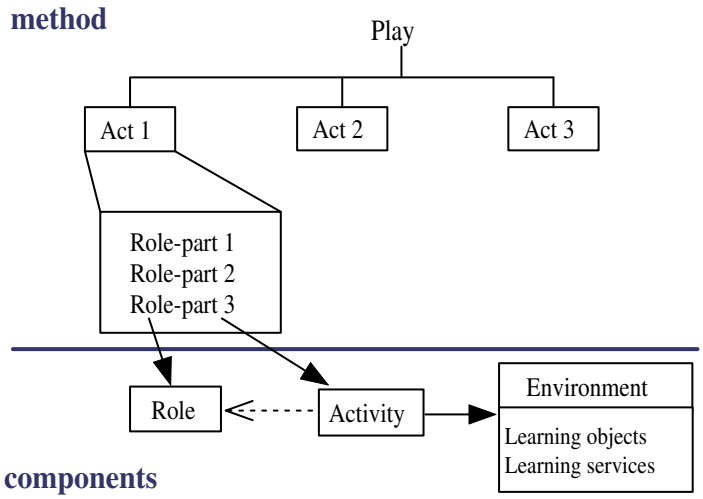

FIG. 4.12 – Illustration de la dynamique des scénarios construits avec IMS-LD.

- des objets d'apprentissage localisés (typiquement spécifiés avec une URL accompagnée parfois de métadonnées)
- des services génériques relatifs à des services concrets disponibles à l'exécution.Ces services n'ont pas d'URL associée à la conception. L'URL sera donnée quand la situation d'apprentissage conçue (*Learning Design*) sera instanciée à l'exécution.

La plupart des services proposent des droits d'accès différents selon de nombreux rôles (i.e. un groupe de discussion spécifie quels rôles ont tel type d'accès : participant, observateur, modérateur, etc.). La spécification d'IMS-LD propose de résoudre ce problème à l'intérieur de la définition du service lors de l'instanciation.

4.3.3.2 Le modèle d'information

Ce modèle est le cœur de la spécification IMS-LD : tous les éléments d'informations et leurs relations sont représentés. Le modèle d'information est défini pour chaque niveau A, B et C et s'appuie sur le modèle conceptuel étudié précédemment. Les éléments de ce modèle sont décrits sous la forme de diagrammes en arbre comme ceux utilisés dans les travaux d'EML-OUNL [Kop02]. Chaque élément du modèle conceptuel peut être décrit sous la forme d'un extrait d'un unique arbre (*learning design and global elements tree*). Le modèle d'information a pour vocation de servir de guide quant à la conception réelle d'apprentissage avec IMS-LD. Il est moins abstrait que le modèle conceptuel et permet de distinguer les prémisses du modèle XML résultant de la conception.

La figure 4.13 illustre un exemple de ces diagrammes. Cette illustration montre qu'une activité d'apprentissage est composée de nombreux autres éléments aux multiplicités variés comme un titre, une description de l'activité, etc. On remarque également qu'une activité d'apprentissage peut avoir localement des objectifs d'apprentissage et pré-requis qui n'apparaissaient qu'au niveau global dans le modèle conceptuel. Ces pré-requis et objectifs d'apprentissage peuvent être décrits en utilisant un autre format standard des spécifications IMS : *IMS Reusable Definition of Competency or Educational Objectives*

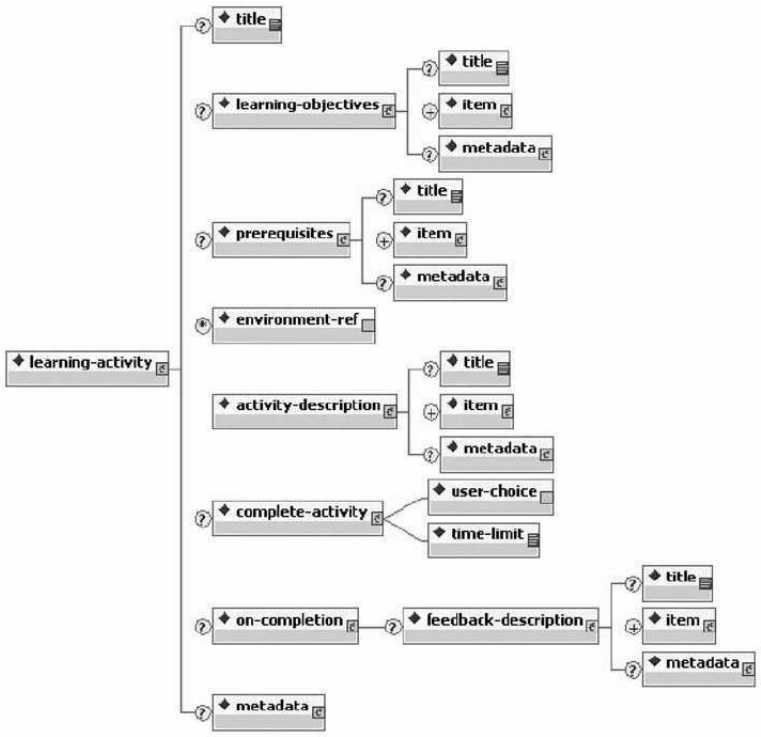

FIG. 4.13 – Diagramme en arbre décrivant l'élément *learning-activity*.

(RDCEO) ; ils peuvent également faire référence à une ressource textuelle les décrivant.

On peut remarquer également que certains concepts du modèle conceptuel ne sont pas référencés dans le modèle d'information. Nous venons de voir que *outcome* était devenu un objectif d'apprentissage. *Person* n'existe plus car, pour IMS-LD, la conception d'une unité d'apprentissage ne doit pas faire référence à des personnes réelles avant le *binding*, c'est-à-dire avant l'instantiation de l'unité d'apprentissage en vue de son exécution. En fait, comme indiqué dans le modèle comportemental ([IMS03c] p.68), l'instantiation est principalement consacrée à l'association de personnes particulières aux rôles spécifiés pendant la conception ainsi qu'à l'instantiation des services génériques par des services locaux (pour lesquels il faudra également définir des rôles/droits d'utilisation des services).

4.3.3.3 Exemples et applications d'IMS-LD

Un exemple de scénario simplifié est illustrée par la figure 4.14. Une pièce de trois actes est décrite avec uniquement un étudiant et son enseignant.

Method

Play 1

Act	Role	Role-part assigned
1.1	Teacher	support-activity: teacher-introduction
1.2	Student	learning-activity: introduction
complete act when all individual students are finished		
2.1	Student	activity-structure: lessons&discussions
2.2	Teacher	activity-structure: teaching
complete act when all teachers complete activity		
3.1	Student	learning-activity: assessment
3.2	Teacher	support-activity: closing-activities
complete act when teacher has completed		

complete play when last act has been completed

complete method when play 1 completed

FIG. 4.14 – Exemple d'un *play* spécifié grâce à la table précédente (re-dessiné à partir de l'exemple de [IMS03c].

Lorsque cet exemple est spécifié avec le langage d'IMS-LD, le code XML suivant est obtenu :

```
<method>
  <play id="play1">
     <act id="act1">
       <role-part id="part11">
            <role-ref ref="Teacher"/>
            <support-activity-ref ref="teacher-introduction"/>
       </role-part>
       <role-part id="part12">
            <role-ref ref="Student"/>
            <learning-activity-ref ref="introduction"/>
       </role-part>
       <complete-act><when-role-part-completed ref="part11"/></complete-act>
     </act>
     <act id="act2">
       <role-part id="part21">
            <role-ref ref="Student"/>
            <activity-structure-ref ref="lessons&discussions"/>
       </role-part>
       <role-part id="part22">
```

```
                <role-ref ref="Teacher"/>
                <activity-structure-ref ref="teaching"/>
            </role-part>
            <complete-act><when-role-part-completed ref="part22"/></complete-act>
          </act>
          <act id="act3">
            <role-part id="part31">
                <role-ref ref="Student"/>
                <learning-activity-ref ref="assessment"/>
            </role-part>
            <role-part id="part32">
                <role-ref ref="Teacher"/>
                <support-activity-ref ref="closing-activities"/>
            </role-part>
            <complete-act><when-role-part-completed ref="part32"/></complete-act>
          </act>
          <complete-play><when-last-act-completed/></complete-play>
        </play>
        <complete-unit-of-learning>
          <when-play-completed ref="play1"/>
        </complete-unit-of-learning>
      </method>
```

La documentation d'IMS-LD suggère l'utilisation des diagrammes d'activités UML ou des diagrammes de Gantt[69] pour décrire initialement la conception de l'apprentissage. La figure 4.15 est une illustration proposée dans la spécification d'IMS-LD [IMS03c] ; elle correspond à l'exemple simplifié de la figure 4.14.

Dans le guide du concepteur [IMS03b], la spécification IMS-LD est positionnée sur les phases de **conception** et de **développement** du processus itératif consistant à développer une unité d'apprentissage (les phases mentionnées sont : analyse, conception, développement, implémentation et évaluation) : ce processus est similaire à celui présenté en 1.2.2.2 ; la phase de développement correspond à celle que nous appelons implémentation. Pour les phases plus amont, le guide propose de suivre ces différentes étapes :

1. Pour la phase d'analyse : un problème éducatif concret est analysé entre différents acteurs (enseignant, didacticien ou encore pédagogue). Cette analyse a pour résultat la mise au point d'un scénario didactique capturé dans un texte narratif (langage naturel), ou bien sous forme de *cheklist*.
2. La narration est alors décrite sous la forme d'un diagramme d'activité UML dans le but d'ajouter davantage de rigueur à l'analyse (ce modèle UML est considéré comme la première étape de conception). Le diagramme d'activité UML sert alors de base pour la conception du document XML d'instance conforme à la spécification IMS-LD. Il s'agit de la seconde étape.

[69] http ://www.ganttchart.com/

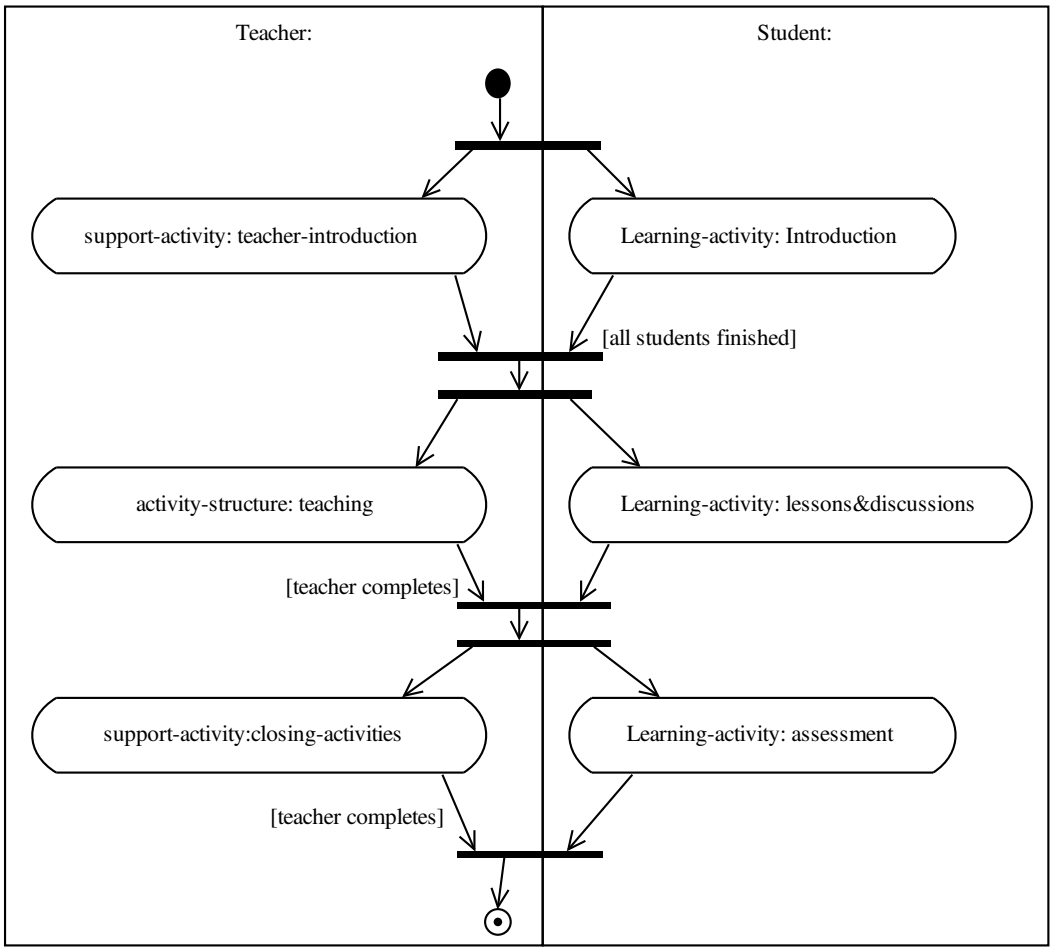

FIG. 4.15 – Utilisation du diagramme d'activité UML pour représenter le scénario de la figure 4.14 (re-dessiné de [IMS03c]).

3. Ce document d'instance sert alors à son tour comme base pour le développement du contenu réel (les ressources) dans la phase de développement.
4. Le *content package* contenant les ressources et le *learning design* peuvent alors être évalués.

Les diagrammes UML, qui sont au cœur de notre orientation pour la mise en œuvre de notre langage, servent donc à modéliser les situations d'apprentissage en amont de leur formalisation avec le langage IMS-LD.

La Figure 4.16 illustre l'utilisation des diagrammes d'activités d'UML dans le but d'analyser un cas d'étude utilisant l'approche qui nous intéresse : l'apprentissage par situation-problème. Cet exemple est tiré du *Best Practice Guide* d'IMS-LD [IMS03b].

Ce diagramme d'activité fait référence à la seconde étape précédente (ajout de rigueur à l'analyse) ; il est construit sur la base de la narration suivante :

- Le *coordinateur* du cours met à disposition des apprenants la description du problème sous la forme d'un fichier accessible sur le site Web.
- Tous les apprenants, ainsi que le facilitateur, lisent le problème.
- Les apprenants élisent un *responsable* (*via* un système de conférence synchrone incluant le facilitateur), qui parlera au nom du groupe et sera responsable de la prise de notes sur les décisions du groupe.
- Les apprenants communiquent ensemble et tentent de clarifier le problème ; ils ont la possibilité également de discuter avec le facilitateur.
- Le responsable rédige la description du problème dans un fichier déposé sur le site ; le groupe continue à identifier les solutions possibles et les pistes pour résoudre le problème.
- Ces pistes sont alors découpées en petit nombre afin d'être explorées plus tard par les apprenants.
- Ces pistes à poursuivre sont listées dans un fichier déposé sur le site.
- Le groupe identifie alors les ressources/apprentissages nécessaires pour résoudre le problème puis, les apprenants effectuent les recherches requises individuellement.
- Éventuellement, le groupe se réunit à nouveau (par le biais du système de conférence synchrone utilisé précédemment) pour discuter de leurs résultats de recherche, à nouveau assisté du facilitateur.
- Le responsable résume les découvertes dans un autre fichier déposé sur le site.
- Finalement, un évaluateur discute avec le facilitateur des résultats et des performances du groupe ; l'évaluateur fournit alors une évaluation du groupe (toujours dans un fichier déposé sur le site).

Nous constatons que l'analyse ayant abouti à l'écriture de cette narration était déjà bien avancée : les besoins d'apprentissage en termes de rôles, d'activités et d'environnements étaient déjà très précis. Le contexte lié à l'environnement d'exécution est déjà connu (système de communication).

En analysant le diagramme d'activité UML proposé (Figure 4.16), nous pouvons également remarquer que l'utilisation des barres de synchronisation (*fork* et *join*) permet de modéliser le parallélisme des activités implicitement mentionné dans la narration. Par contre les flèches traversant les couloirs expriment un besoin de précédence entre les deux activités qu'elles relient. Sur la base du document XML correspondant à la spécification de cet exemple avec IMS-LD ([IMS03b] p. 61 à 65), nous avons annoté la figure de manière à indiquer la position des 13 actes du modèle XML obtenu (voir 4.16). Les cadres entourant des activités successives réalisées par le même rôle font référence aux *structure d'activités* spécifiées dans le document XML.

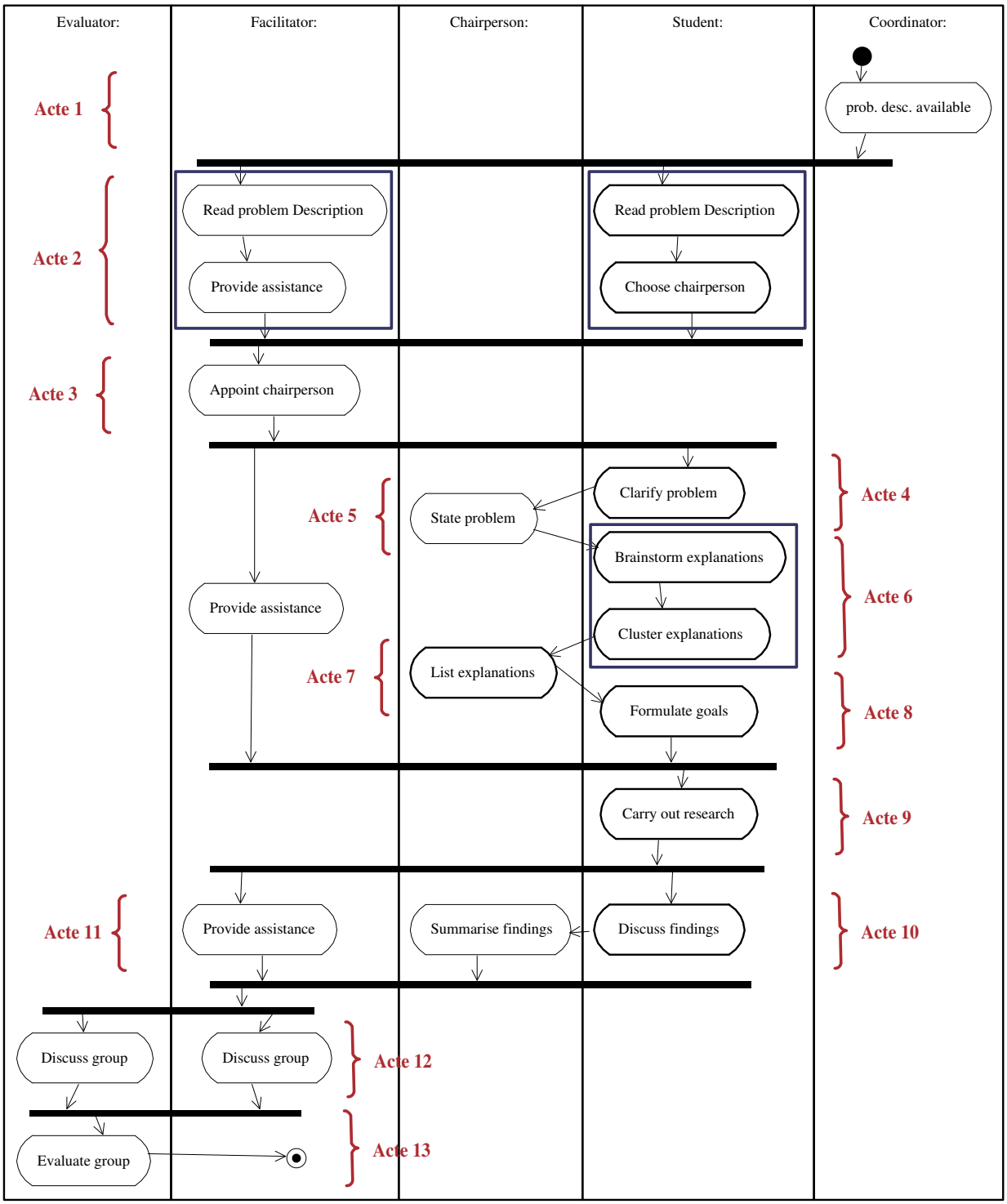

FIG. 4.16 – Exemple d'analyse d'un cas d'étude (PBL) avec les diagrammes d'activités UML (re-dessiné et annoté à partir de [IMS03b]).

4.3.4 MISA

Depuis 1992, de nombreux travaux de recherche au Centre de recherche CIRTA[70] du LICEF ont porté sur la mise en œuvre d'une nouvelle approche à la conception pédagogique fondée sur les sciences cognitives qu'ils ont baptisé « Ingénierie pédagogique » (*Instructional Engineering*) : « *A method that supports the analysis, the design and the delivery planning of a learning system, integrating the concepts, the processes and the principles of instructional design, software engineering and cognitive engineering* »[Paq04a]. De ces travaux ont abouti une méthode d'ingénierie pédagogique *MISA*[71] [Paq97], une plate-forme/LMS de formation à distance *Explor@*[72] et un environnement auteur pour les concepteurs *ADISA* (Atelier distribué d'ingénierie de systèmes d'apprentissage)[73].

L'objectif de MISA est d'aider des concepteurs dans l'élaboration de la conception de systèmes pédagogiques. Cette conception pédagogique pourra alors être utilisée pour produire un système pédagogique exécutable (*runtime*) qui peut être délivré sur un système de diffusion comme Explor@ ou tout autre LMS et LCMS. Ainsi, MISA se positionne (comme EML-OUNL et IMS-LD) dans un processus industriel de la e-formation pour lequel de nouveaux experts représentent le public visé pour l'application de ces méthodes d'ingénierie pédagogique.

4.3.4.1 La méthode d'ingénierie pédagogique

Une approche de modélisation des connaissances a été suivie afin de définir la méthode MISA (Figure 4.17) et également ses concepts, ses processus et ses principes. Afin d'aboutir à la production d'un modèle d'un système d'apprentissage, la méthode propose de suivre six phases, chacune développant progressivement les quatre modèles de conception et les descriptions des objets. MISA 4.0 comprend plus de 35 sous-tâches, chacune produisant un élément de conception. Les quatre modèles sont :

- Le modèle de connaissances : c'est une représentation graphique du contenu du domaine du système d'apprentissage visé. Dans ce modèle, les compétences cibles ou pré-requises sont associées aux unités de connaissance en fournissant les objectifs d'apprentissage du modèle d'apprentissage.
- Le modèle d'apprentissage : c'est essentiellement un réseau d'unités d'apprentissage (*Learning Unit* - LU) et d'événements auxquels sont attachés les connaissances et compétences cibles. Chaque LU est décrit par un scénario d'apprentissage graphique décrivant les activités d'apprentissage et de support reliées à des ressources. Les ressources manipulant du contenu (appelées *instruments*) sont associées à un sous-ensemble du modèle de connaissances.
- Les modèles de matériaux d'apprentissage (optionnels). Chacun de ces modèles regroupe les *instruments* dans un *objet d'apprentissage*, en décrivant les composantes médiatiques, les documents source et quelques principes de présentation et autres spécifications pour construire ou agréger les objets d'apprentissage.
- Le modèle de livraison : il rassemble les matériaux d'apprentissage et tous les autres types de ressources tels que les outils, les liens de communication, les services, etc., décrits dans le modèle d'apprentissage. Chacun de ces modèles de livraison représente un processus de flot de travail

[70] http ://www.licef.teluq.uquebec.ca/cirta/
[71] Actuellement dans sa version 4.0.
[72] http ://explora2.licef.teluq.uquebec.ca/demo/
[73] http ://www.cogigraph.com/fr/adisa.htm

Problem definition		
100 Organization's training system	102 Training objectives 104 Target populations	106 Actual situation 108 Reference documents
Knowledge Modeling	**Instructional Modeling**	
210 Knowledge modeling principles	220 Instructional principles	
212 Knowledge model	222 Learning events network	
214 Target competencies	224 Learning units properties	
310 Learning unit content	320 Instructional scenarios	
410 Learning instrument content	322 Learning activities properties	
610 Knowledge and comptency management	420 Learning instruments properties 620 Actors and group management	
Learning Materials Modeling	**Delivery Modeling**	
230 Media development principles	240 Delivery principles	
330 Development infrastructure	242 Cost-benefit analysis	
430 Learning mateirials list	340 Delivery planning	
432 Learning materials models	440 Delivery models	
434 Media elements	442 Actors and user's materials	
436 Source documents	444 Tools and telecommunication	
630 Learning system and ressource management	446 Services and delivery locations	
	540 Assessment planning	
	542 Revision decisions log	
	640 Maintenance and quality management	

FIG. 4.17 – Les principales tâches de la méthode d'Ingénierie Pédagogique MISA [Paq04b].

(*worflow*) multi-utilisateurs où les acteurs utilisent et produisent des ressources en réalisant leurs différents rôles.

Au coeur de la méthode se trouve la modélisation graphique. Toutes les représentations des modèles de la méthode sont réalisées grâce à la notation MOT [Paq99]. Tous les modèles MOT et propriétés d'objets sont traduits vers XML pour être interprétables par la machine.

Le modèle qui nous intéresse le plus dans cette méthode pour ses ressemblances avec les EML est le modèle pédagogique.

4.3.4.2 Le modèle pédagogique

La Figure 4.18 tirée de [Paq04a] montre, à un niveau élevé, le modèle pédagogique comme un réseau d'événements d'apprentissage (*learning events*). Ce concept désigne le terme générique de leurs travaux ; il est la base de toutes les représentations ; il permet de décrire des modules, des cours, etc.

Tout réseau d'événements d'apprentissage est composé d'événements d'apprentissage, de ressources, de liens et de règles. Les *C links* (Composition) permettent de spécifier la hiérarchie entre les événements d'apprentissage tandis que les *P links* (Précédence) décrivent leurs pré-requis. Les ressources peuvent être utilisées ou produites par les événements d'apprentissage en les reliant avec des *I/P links* (Input/Product). Les règles gouvernent l'utilisation des événements *via* l'utilisation des *R links* (Rules).

Il est possible de construire un scénario d'apprentissage pour chaque unité d'apprentissage. Ce scénario correspond alors à un événement d'apprentissage non décomposable en sous-réseau d'événements. Il a des

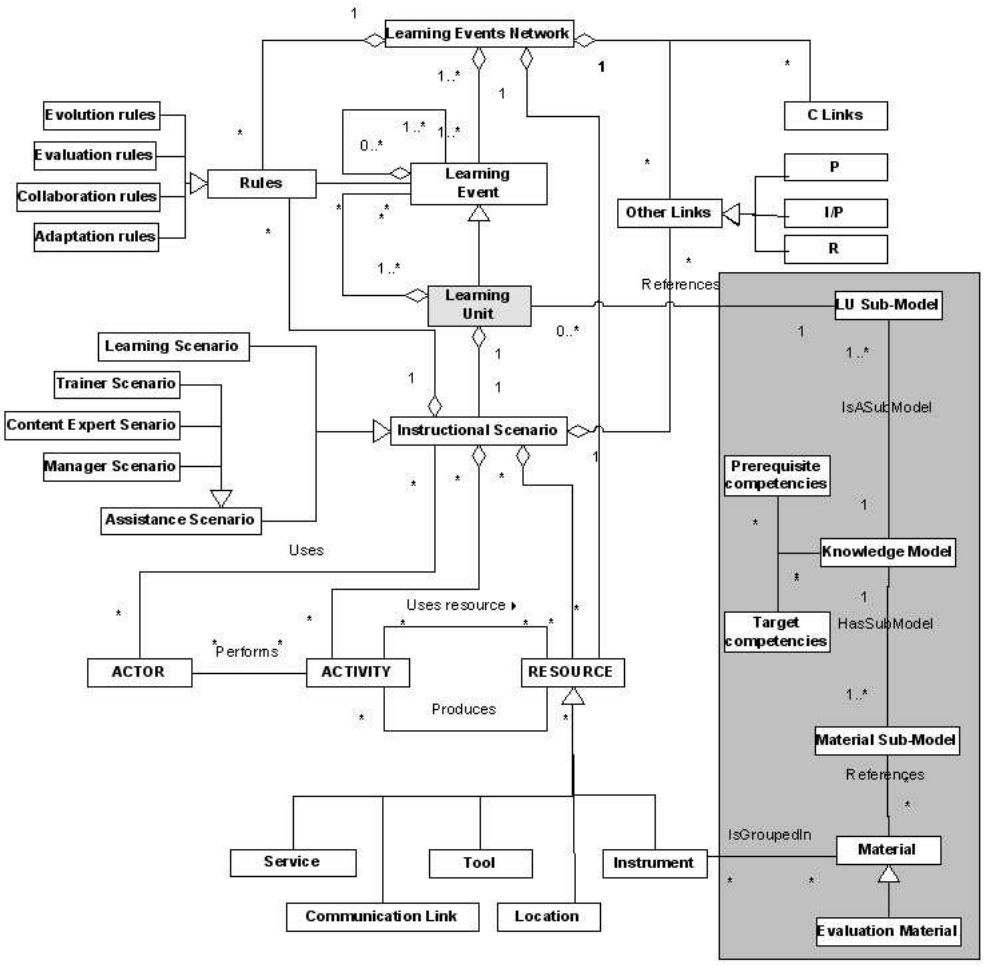

FIG. 4.18 – Diagramme de classe du modèle pédagogique de MISA 4.0.

objectifs et des pré-requis d'apprentissage, fournis par les *target* et *prerequisite competencies* définis dans le modèle des connaissances (partie droite de la figure 4.18).

Une unité d'apprentissage fait référence à un unique scénario d'apprentissage regroupant les acteurs, les ressources, les activités et les liens. Les acteurs réalisent les activités (*R link*) qui utilisent ou produisent des ressources (*I/P links*). Il est possible d'exprimer la précédence entre activités (*P links*) mais la composition d'activités (*C links*) n'est pas permise [Paq04a]. Les règles d'exécution, d'évaluation, de collaboration, etc., sont reliées aux activités par les *R-links*. Le sous-ensemble d'un scénario regroupant les activités et leurs ressources réalisées par un apprenant est appelé scénario d'apprentissage (*Learning Scenario*) ; de même pour les facilitateurs dont le sous-scénario est appelé *Assistance Scenario*.

Cinq types de ressources peuvent être utilisées : *instruments*, *tools*, *services*, *locations*, et *communication links*. Les *instruments* sont les seuls représentant un contenu (ils sont reliés aux matériaux du modèle de connaissance).

La Figure 4.19 présente un exemple de scénario d'apprentissage. Le modèle graphique utilise la notation MOT. Les ovales représentent des activités réalisées par les acteurs (L pour *learner* ou F pour

facilitator). Les rectangles représentent des ressources (I pour *instruments*, T pour *tools*, S pour *services*, C pour *communication* ou L pour *location*). Les ressources non marquées sont des résultats produits pendant une activité. Les hexagones représentent les règles (X pour exécution, E pour évaluation, C pour collaboration et A pour adaptation).

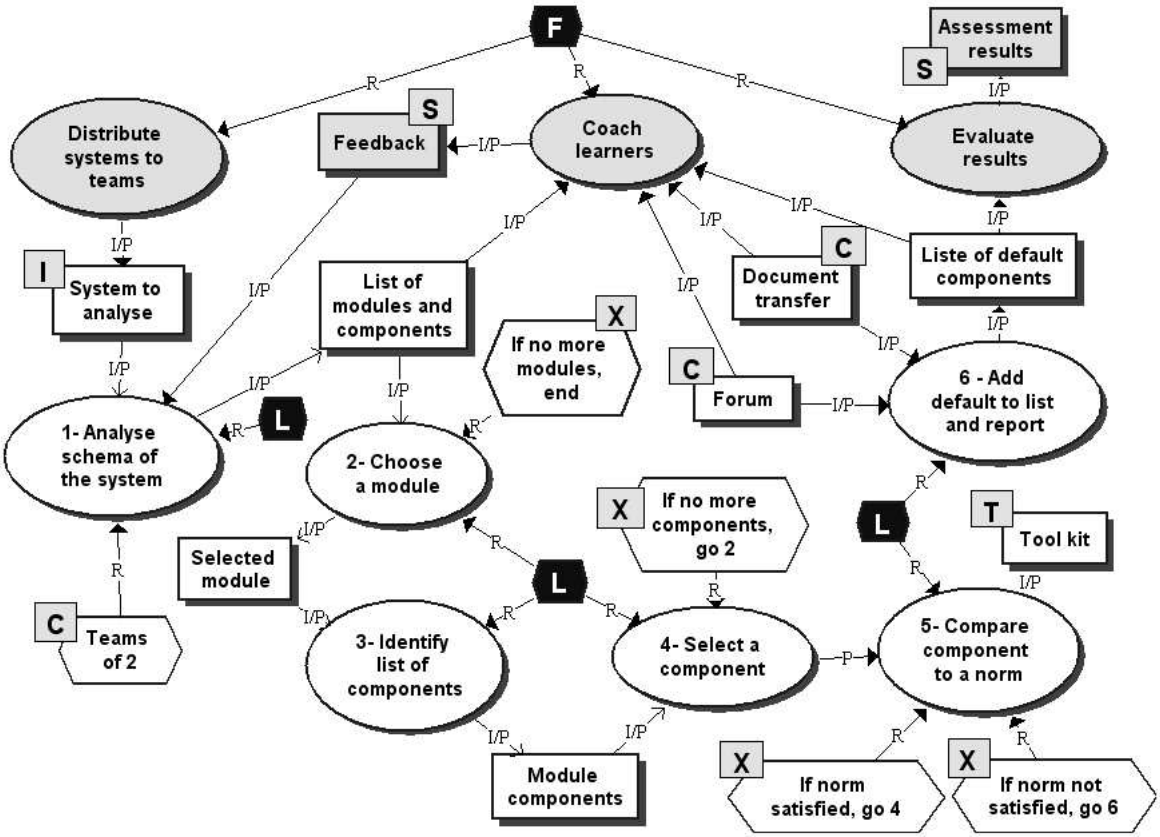

FIG. 4.19 – MISA.

La narration correspondant à cet exemple est disponible dans [Paq04a].

4.3.4.3 MISA et les EML

De nombreux concepts du modèle pédagogique de MISA sont très similaires à ceux d'IMS-LD bien que leurs terminologies respectives diffèrent. Pour G. Paquette [Paq04a], MISA constitue un EML dans le sens défini au début de la section consacrée aux EML (4.3.1) : le modèle d'information sémantique est constitué dans MISA de 1/l'ensemble des modèles MOT groupant le réseau d'événements d'apprentissage (*Network of learning events*) ; 2/ des scénarios d'apprentissage décrits pour chaque unité d'apprentissage (*learning unit*) ; 3/ et de leurs modèles de connaissances associés. La transformation de ces modèles MOT en des fichiers XML constitue le *binding* d'informations. Ces fichiers XML servent au même besoin de réutilisabilité et d'interopérabilité que ceux de la spécification IMS-LD.

En comparaison à un processus général de conception-développement-diffusion, MISA est une méthode similaire à celles de l'ingénierie logicielle qui vise à décrire de nombreux éléments de conception

et pas uniquement le résultat de la spécification d'un scénario d'apprentissage. Par contre, IMS-LD s'intéresse davantage à la diffusion, au paquetage des contenus, et aux moyens permettant leur implémentation/exécution.

IMS-LD apporte aux niveaux B et C des concepts permettant d'améliorer les interactions et la dynamiques des scénarios construits. Ces concepts sont absents de MISA qui les considère comme trop proche de préoccupations d'exécution. MISA décrit les services comme des ressources particulières conformément à la définition IEEE de *learning object* tandis que la spécification IMS-LD sépare services et objets d'apprentissage (car les services doivent être initiés à l'exécution). Le tableau 4.2 établit une correspondance entre certains éléments des deux terminologies d'IMS-LD et de MISA.

IMS-LD	MISA
Unit of Learning	*Learning System*
Learning Design	*Instructional Model*
Learning Objective	*Competency*
Prerequisite	*Competency*
Method	*Instructional Scenario*
Activity	*Learning Event/Learning Unit*
Activity-Structure	*Learning Events Network*

TAB. 4.2 – Correspondances IMS-LD/MISA

Alors qu'un scénario d'IMS-LD décrit dans un *play* est toujours une séquence linéaire d'actes, un scénario pédagogique de MISA peut décrire un réseau d'activités et leurs ressources sous une forme également de séquence linéaire ou bien comme une structure d'arbre, un modèle de *workflow* avec des interactions, etc. [Paq04a].

La Figure 4.20 tirée de [Paq04a] illustre un exemple de scénario[74] décrit avec MISA selon l'approche séquentielle d'IMS-LD. Cet exemple montre le découpage en actes ainsi que la concurrence des *role-part* associant les rôles aux activités, ici représentés par une association *R link* dans le modèle MOT.

[74]Il s'agit de l'exemple du premier *play* de la Figure 4.14

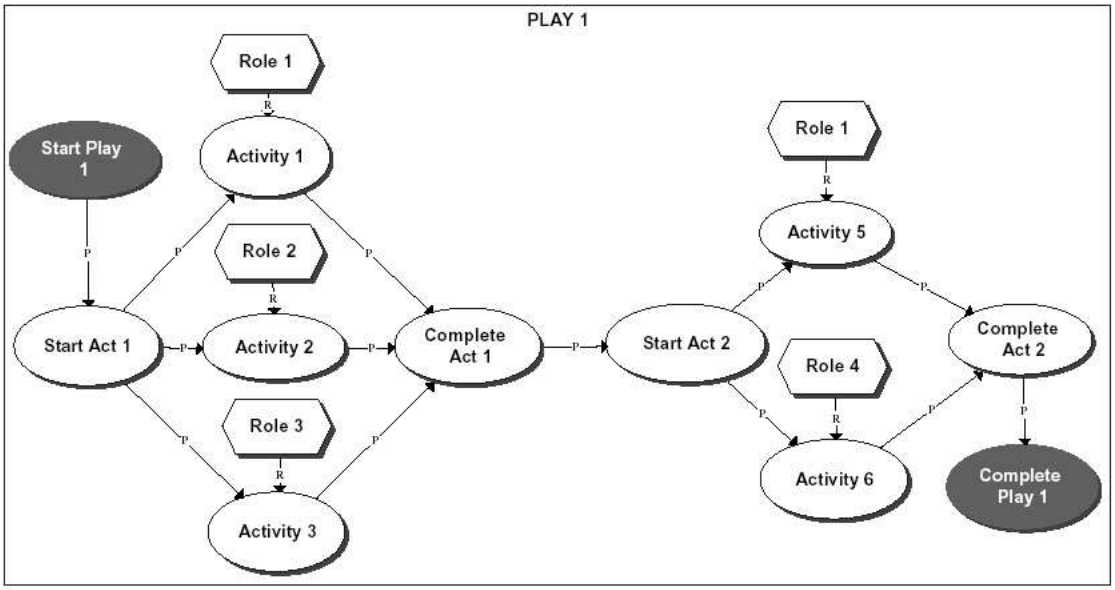

FIG. 4.20 – Exemple d'une *play* IMS-LD modélisée avec MISA [Paq04a].

4.4 Bilan

Les trois types de langage que nous avons étudiés (méta-données, ontologies et EML) ont pour objectif de contribuer à améliorer la compatibilité, la réutilisation et l'interopérabilité d'éléments pédagogiques. Les méta-données et les ontologies s'intéressent davantage à cette réutilisation/interopérabilité pour les ressources pédagogiques (description du contenu/des ressources par **séparation fond/forme**). Les descriptions de ces ressources recherchent alors à être pédagogiquement neutres afin d'améliorer la réutilisation. Il est donc nécessaire de pouvoir également décrire précisément des modèles de situations d'apprentissage qui manipulent ces ressources (ou objet d'apprentissage lorsqu'elles sont accompagnées de leurs méta-données).

Les ontologies, issues des travaux de l'ingénierie des connaissances, permettent de spécifier de tels scénarios mais il se pose alors des problèmes liés à la construction de l'ontologie, de son outillage, de son utilisation. En effet, alors que les méta-données s'adressent davantage au rôle de *fournisseur de ressources pédagogiques*, les ontologies nécessitent des experts et/ou des outils adaptés. De plus, les ontologies sont plus adaptées pour des phases de conception (et non pour les phases amont qui nous intéressent également) pour lesquelles les concepts et relations sont plus facilement identifiables et donc formalisables (l'un des atouts des ontologies étant en effet qu'elles sont interprétables facilement par la machine).

Le travail autour des EML, initié avec EML-OUNL puis standardisé dans la spécification d'IMS-LD, permet de spécifier formellement des unités d'apprentissage correspondants à la description des ressources et du scénario les manipulant (**séparation contenu/scénario** sous la forme d'une méthode spécifiant les rôles, les activités réalisées pour chaque rôle et les ressources manipulées au travers de la réalisation). La spécification d'IMS-LD n'est pas complètement neutre pédagogiquement : le modèle conceptuel qu'elle définit reflète une approche sémantique non neutre [All04]. Ces EML concernent des ingénieurs pédagogiques experts de ces langages : une formation à la spécification IMS-LD est nécessaire.

De plus, cette spécification n'est qu'une proposition de standard indépendante de tout environnement informatique permettant sa mise en œuvre. Bien qu'IMS-LD ait été finalisé récemment (peut-être « *choisi très (trop ?) rapidement comme standard* » comme l'évoque certains auteurs [Per03]), quelques initiatives ont abouti à des prototypes d'outils-auteurs compatibles avec IMS-LD : Edubox [75], Reload[76], le système LAMS [Dal03] et l'Open Source LD engine [Vog03]. Toutefois, aucun environnement auteur n'adresse l'équipe pluridisciplinaire chargée de la conception des unités d'apprentissage. Il n'existe encore pas, à l'heure actuelle, d'outils compatibles IMS-LD à 100% et s'adressant à des utilisateurs finaux non-experts.

L'élaboration de modèles avec des langages comme IMS-LD concerne la phase de conception avancée pour laquelle un pré-scénario a déjà été établi par les enseignants et autres concepteurs de l'équipe en charge de concevoir la formation. Des modèles basés sur le langage UML, et plus particulièrement les diagrammes d'activités, servent à illustrer les résultats d'analyse sur lesquels se basent les EML. La spécification d'IMS-LD encourage l'utilisation d'UML pour ces phases plus amont.

Nous avons également étudié le *pseudo*-langage de la méthode d'ingénierie pédagogique MISA. Il permet de décrire des unités d'apprentissage proches de celles des EML par le biais d'une approche issue de l'ingénierie des connaissances. Des modèles graphiques utilisant la notation MOT permettent alors de décrire également les rôles, les activités, les ressources, etc. Toutefois, bien que les plus-values d'un langage graphique sont justifiés dans ce cas, la notation MOT est trop restrictive pour permettre de décrire des modèles de scénario aussi complexes que ceux d'IMS-LD (interaction et dynamique des scénarios).

[75] http ://www.ou.nl/info-alg-edubox/
[76] http ://www.jisc.ac.uk/index.cfm ?name=project_reload

Chapitre 5

Modélisation et méta-modélisation UML

«La différence entre la théorie et la pratique
c'est qu'en théorie il n'y a pas de différence,
alors qu'en pratique il y en a.
Anonyme»

Sommaire

Ce chapitre a pour objectif d'étudier les possibilités offertes par le langage UML, l'orientation initiale de nos travaux (proposée en 1.3.2), afin qu'il serve de support à la mise en œuvre de notre langage dédié à la conception d'apprentissage par situations-problèmes coopératives médiatisées sur une plate-forme de formation à distance.

La première section de ce chapitre introduit le langage UML (5.1) : nous présentons brièvement ce qu'est UML, quels sont ses usages et quelle est sa structure. Nous évoquons également la sémantique de

ce langage, les outils existants et esquissons la notion de méthode pour UML. La section suivante (5.2) est consacrée à la présentation de la modélisation avec le langage UML. Les différents diagrammes et leurs usages proposés par la notation sont présentés ; ce travail est réalisé au regard des usages possibles pour les modèles de conception qui nous intéressent. La troisième et dernière section (5.3) concerne la méta-modélisation avec UML. Après avoir introduit ce nouveau concept en détail et ses différentes applications, nous présentons la notion de profil UML qui permet de spécialiser UML pour des domaines spécifiques.

5.1 Introduction à UML

UML (*Unified Modeling Language*) est un langage pour **visualiser**, **spécifier**, **construire** et **documenter** tous les aspects et artefacts d'un système logiciel [OMG03d]. Dans la pratique, UML concerne tout système, au sens large [77], construit en utilisant l'approche orienté objet (OO).

Comme son nom l'indique, UML est le résultat de la fusion d'un ensemble d'autres langages de modélisation graphique orientés objets (expliquée dans la section suivante). Il est devenu un standard *de facto*. Un des ingrédients de son succès est le fait d'être un langage générique. L'ensemble des concepts et vues composant la sémantique d'UML peut s'adapter à tous les domaines et problèmes de conception.

Toutefois, il ne faut pas oublier que la notation UML est un langage de modélisation objet et non pas une méthode objet. La notation UML est conçue pour servir de langage de modélisation objet indépendamment de la méthode suivie.

L'évolution du langage UML est contrôlée par l'OMG[78], un organisme international à but non lucratif, créé en 1989 à l'initiative de grandes entreprises (HP, Sun, Unisys, American Airlines, ou encore Philips). Aujourd'hui, l'OMG fédère plus de 860 acteurs du monde informatique et œuvre essentiellement à produire et distribuer des spécifications dans le but de fournir un cadre commun au développement d'applications.

5.1.1 Historique

Dans les années 90, de nombreuses méthodes ont été développées afin de satisfaire les besoins de documentation et de spécification du logiciel (OMT, OOSE, Booch, etc.). Ces méthodes étaient très similaires ; des même concepts de base pouvaient apparaître différemment dans la notation accompagnant ces méthodes, portant à confusion au niveau de l'interprétation faite par leurs utilisateurs. Cette diversité a engendré un problème d'interopérabilité entre outils qui a ralenti le déploiement des méthodes de développement à objets. L'ambition d'UML, dont la première version a été publiée en 1997 par l'OMG, est de rassembler en une seule notation les meilleurs caractéristiques des différents langages de modélisation à objets. Ainsi, la spécification UML définit les concepts fondamentaux de l'objet ainsi que d'autres concepts utiles lors de la modélisation logiciel (cas d'utilisation, etc.).

Depuis la première version, la spécification d'UML a évolué : v1.1 (standard officiel de l'OMG), v1.2 (révisions mineures), v1.3 (révisions majeures), v1.4 (des concepts autour des composants et profils sont

[77] Non restrictifs aux seuls systèmes logiciels.
[78] L'OMG a été rendue célèbre par le standard CORBA (*Common Object Request Broker Architecture*).

détaillés)[79], v1.5 (ajout des *action semantics*)[80]. La spécification UML 2.0 est en cours de finalisation [OMG03b, OMG03c]. Un historique d'UML plus détaillé est disponible dans la plupart des ouvrages du domaine ([Fow03] par exemple).

5.1.2 Différentes perspectives d'usage d'UML

Il existe différentes manières d'utiliser un langage de modélisation graphique. UML peut être utilisé de trois façons différentes [Fow03] :

UML comme illustration. Avec cet usage, UML permet de communiquer certains aspects du système. La base de l'illustration est la *sélectivité* : le but est d'utiliser les illustrations comme moyen pour communiquer des idées et alternatives à propos du travail à réaliser.

L'illustration est une activité plutôt informelle et dynamique. Les illustrations sont également utiles dans la documentation pour lesquelles la communication prime sur la complétude.

Les outils utilisés pour réaliser les illustrations peuvent être de simples outils de dessin, car souvent les utilisateurs ne cherchent pas à suivre strictement les règles d'UML.

UML comme plan (*blueprint*). En contraste avec les illustrations, les *blueprints* cherchent la complétude. L'idée centrale est qu'ils sont développés par un concepteur dont le travail consiste à construire une spécification détaillée de ce que le programmeur devra coder. La conception doit être suffisamment complète pour qu'aucune décision ne soit laissée au programmeur.

Les outils nécessaires à cet usage nécessitent davantage de sophistication afin de manipuler tous les détails requis pour décrire la tâche. Ces outils spécialisés sont appelés Atelier de Génie Logiciel (AGL) [81] pour UML.

La limite entre *blueprints* et illustrations n'est pas distincte. Les différences se rapportent au fait que les illustrations sont délibérément incomplètes, mettant en valeur les informations importantes, tandis que les *blueprints* ont pour but d'être complets de manière à réduire l'activité de programmation à une simple mécanique. Pour résumer, les illustrations sont exploratives tandis que les *blueprints* sont définitifs.

UML comme langage de programmation. Dans cette perspective, deux approches légèrement différentes sont recensées : l'approche MDA (*Model Driven Architecture*) et l'approche *Executable UML*. Le MDA est une approche, standardisée et controlée par l'OMG [OMG03a], divisant le travail de développement en deux grandes parties. Dans la première, les concepteurs représentent leurs applications en créant des *Platform Independant Models* (PIM) : des modèles UML indépendants de toute technologie. Ensuite, les outils peuvent transformer ces PIM en *Platform Specific Models* (PSM) : il s'agit d'un modèle prenant en compte un environnement d'exécution spécifique (ce modèle peut être un modèle UML). D'autres outils génèrent alors le code correspondant à ce PSM et à la plate-forme technologique choisie. Si le processus allant du PIM, puis au PSM, puis au code final est complètement automatisé, alors il est possible de considérer UML comme langage de programmation. Si l'une des étapes est manuelle, il s'agit de *blueprints*. La deuxième approche

[79] Il s'agit de la version avec laquelle nous avons commencé la thèse.

[80] La dernière spécification en date [OMG03d] proposée à travers le site officiel d'UML (http ://www.omg.org/uml/)

[81] On parle de *CASE-tool* en anglais (Computer-Aided Software Engineering).

concernant *Executable UML* [Mel02] est similaire en partie à celle du MDA : le concepteur commence par spécifier un modèle indépendant d'une plate-forme technologique (équivalent au PIM), puis, il utilise un compilateur de modèles pour transformer, en une seule étape, le modèle UML en code exécutable ; il n'y a donc pas de modèle PSM. Ces usages d'UML demandent des outils très sophistiqués.

Parmi ces trois usages d'UML, seuls les deux premiers nous semblent être utiles pour nos modèles de conception : UML comme illustration pour supporter l'expression initiale des besoins, l'analyse et la conception, les *blueprints* pour la conception avancée de nos modèles de PBL coopératives.

Une autre façon de décrire UML est de s'intéresser à la manière dont les utilisateurs le perçoivent.

Perspective logicielle. Dans cette perspective, les éléments d'UML correspondent directement à des éléments d'un système logiciel.

Perspective conceptuelle. Dans ce cas, UML représente une description des concepts d'un domaine d'étude. Les modèles ne décrivent pas des éléments logiciels mais construisent un vocabulaire pour discuter d'un domaine particulier.

Selon ce découpage, la perspective conceptuelle convient à l'usage de nos modèles de conception. En effet, ces modèles concernent davantage la description d'une situation apprentissage que la description d'un système logiciel support à cet apprentissage.

Pour la plupart des utilisateurs, l'essence même d'UML repose sur les diagrammes alors que pour d'autres, les diagrammes sont secondaires, le cœur d'UML étant son méta-modèle. En fait, les diagrammes ne sont qu'une simple présentation du méta-modèle (la notation). Dans le cas de nos travaux, nous nous sommes intéressés autant au méta-modèle d'UML, car nous cherchons à étendre sa sémantique à notre domaine des situations-problèmes, mais également aux diagrammes, car ce sont ceux qui permettront l'élaboration de la notation de notre langage.

5.1.3 Cœur d'UML : la notation et le méta-modèle

La spécification UML définit deux éléments : une **notation** et un **méta-modèle**. La notation correspond à tous les éléments graphiques que l'on peut voir dans les modèles ; elle correspond à la *syntaxe graphique* du langage de modélisation UML (on parle également de *syntaxe concrète*). Le méta-modèle, quant à lui, est défini dans le but de décrire avec rigueur les concepts d'UML : il correspond à la *syntaxe abstraite* d'UML. Le méta-modèle est lui-même défini comme un ensemble de diagrammes de classe UML. On retrouve les deux éléments, notation et méta-modèle, décrits en détail dans la spécification officielle d'UML [OMG03d]. Des règles de bonne écriture, écrites en OCL (*Object Constraint Language*) précisent la sémantique des concepts du méta-modèle ainsi que ceux de la notation.

La figure 5.1 montre un petit extrait du méta-modèle UML (cet extrait est présenté uniquement pour donner une idée de la présentation du méta-modèle ; notre but n'est pas de l'expliquer).

Le méta-modèle est présenté plus en détail dans la section de ce chapitre consacrée à la méta-modélisation UML (5.3). Les types de diagrammes et leurs usages proposés par UML sont décrits dans la section suivante dédiée à la modélisation UML (5.2).

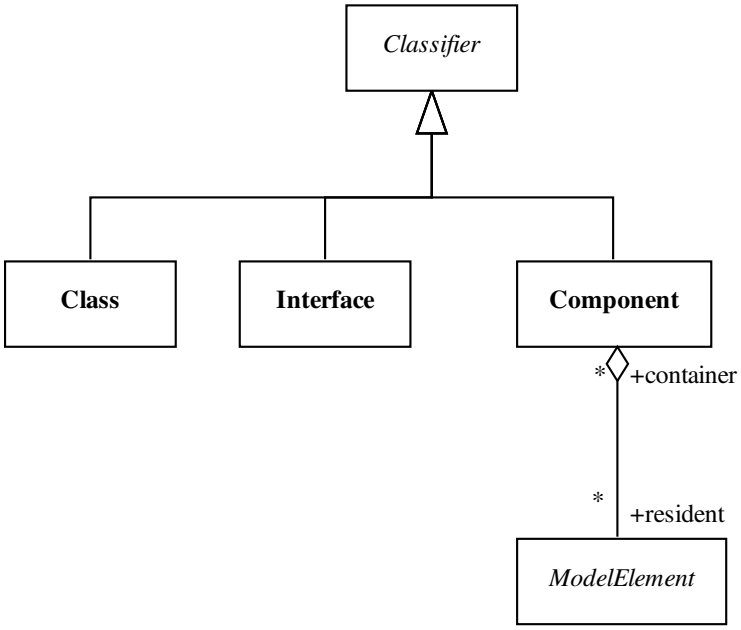

FIG. 5.1 – Un extrait du méta-modèle UML.

5.1.4 Méthodologie UML

Comme nous l'avons précédemment évoqué, UML est un langage de modélisation indépendant des méthodes de conception objets ou processus d'élaboration des modèles. Toutefois, la littérature sur UML prône, pour tirer avantage d'UML, de suivre un processus :

- dirigé par les cas d'utilisation : ils doivent être utilisés en premier pour établir le comportement désiré du système, pour vérifier et valider l'architecture du système, pour tester et pour communiquer entre les responsables du projet.
- centré architecture : l'architecture du système (un modèle d'architecture est présenté ci-après) est utilisée comme le premier artefact pour concevoir, construire, gérer et faire évoluer le système en développement.
- itératif et incrémental : le principe repose sur l'idée selon laquelle pour modéliser (comprendre et représenter) un système complexe, il vaut mieux s'y prendre en plusieurs fois, en affinant son analyse par étapes. Cette démarche devrait aussi s'appliquer au cycle de développement dans son ensemble, en favorisant le prototypage. Le but est de mieux maîtriser la part d'inconnu et d'incertitudes qui caractérisent les systèmes complexes. Les itérations du processus désignent les étapes, tandis que les incréments correspondent à des stades de développement du produit.

Kruchten propose différentes vues, indépendantes et complémentaires, permettant de définir un modèle d'architecture [Kru95] (chaque vue est une projection, selon un aspect particulier, dans l'organisation et la structure du système) :

- vue des cas d'utilisation : elle montre les fonctionnalités du système telles que perçues par les

acteurs externes ;
- vue logique : elle montre comment les fonctionnalités sont conçues dans le système en termes de structure statique et structure dynamique ;
- vue de réalisation : montre l'organisation du code et de son contexte d'exécution ;
- vue des processus : elle montre les principaux éléments du système relatifs à la performance des processus (décomposition du système en terme de processus (tâches) ; interactions entre les processus (communication) ; synchronisation et communication des activités parallèles (threads)).
- vue de déploiement : cette vue, très importante dans les environnements distribués, décrit les ressources matérielles et l'affectation des composants du logiciel à ces ressources.

Cette vue « 4+1 » a fortement inspiré UML (Figure 5.2).

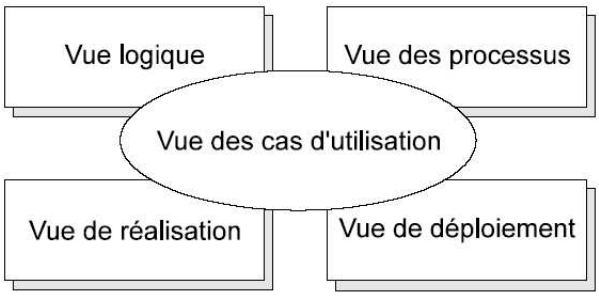

FIG. 5.2 – Le modèle d'architecture « 4+1 » [Kru95].

Des exemples de processus d'élaboration de modèles UML reprenant les caractéristiques précédentes sont le RUP[82] (*Rational Unified process* ou UP pour seulement *Unified process*) [83] d'IBM/Rationale, le cycle en Y (2TUP - « *2 Tracks Unified Process* ») de Valtech[84] [Roq00].

5.1.5 Outils UML et usages

L'utilisation d'un langage de modélisation aussi complexe qu'UML nécessite le support d'outils. Bien que les premières ébauches de modèles peuvent être réalisées manuellement ou avec des outils traditionnels de dessin, le travail de maintenance, de synchronisation et de cohérence, dans la plupart des diagrammes, est généralement impossible sans l'aide d'un outil adapté.

Il existe de nombreux outils supportant UML, allant de simples outils de dessin proposant la notation UML jusqu'aux ateliers de génie logiciel (AGL) pour UML[85]. Tous les outils ne proposent pas un support complet de la notation UML (certains diagrammes ne sont pas exploités). Ils peuvent également fournir des niveaux différents de support selon les fonctionnalités proposées (par exemple le *model-checking*). Des outils compatibles UML 2.0 sont actuellement proposés, toutefois, aucun n'est encore compatible à 100%. Les outils permettent également d'aider la définition d'UML : l'opérationnalisation relève les ambiguïtés

[82] Bien qu'il soit appelé processus il correspond plutôt à un *canevas* pour la spécification de processus.

[83] http ://www-136.ibm.com/developerworks/rational/products/rup

[84] http ://www.valtech.fr

[85] Une liste exhaustive des AGL UML les plus aboutis et les plus utilisés est disponible sur http ://www.objectsbydesign.com/tools/umltools_byCompany.html

de la spécification UML pour laquelle un grand effort de clarification de la sémantique est nécessaire [Har04].

Voici les fonctionnalités qu'un outil pour le support du développement de modèles UML doit fournir[86] :

- proposer un support de notation pour l'ensemble des diagrammes UML ainsi qu'un support pour assister et guider l'utilisateur dans l'élaboration de modèles corrects ;
- agir comme un *entrepôt* : les informations contenues dans le modèles doivent être conservées et synchronisées (par exemple, si le nom d'une classe est modifié alors le changement doit être répercuté dans tous les autres diagrammes dans lesquels la classe est utilisée). Certaines fonctionnalités sont facilitées grâce à cet entrepôt : vérifier la cohérence des modèles, critiquer les modèles (l'outil peut indiquer les erreurs comme les oublis ou pointer les solutions inappropriées en appliquant des heuristiques sur les modèles), générer des documentations, permettre la réutilisation d'éléments ou de diagrammes (de manière à ce que tout ou partie de solutions dans un projet puisse être facilement réutilisé dans d'autres).
- supporter et faciliter la navigation entre les différentes vues et diagrammes d'un modèle (et par extension faciliter la recherche d'éléments de modélisation) ;
- fournir un support pour la modélisation multi-utilisateurs (travail coopératif sur des modèles partagés, historique des versions, etc.) ;
- permettre la génération de code (par le biais de générateurs de code ou de compilateurs de modèle) pour faciliter le travail d'implémentation ;
- permettre l'échange ou l'exportation des modèles pour d'autres outils. Cette fonctionnalité nécessite un format standardisé pour le stockage et le partage des modèles. La spécification *XML Metadata Interchange* (XMI [OMG03e]) de l'OMG fournit cette fonctionnalité pour les modèles UML.

5.2 Modélisation UML

Cette section est consacrée à la présentation de la modélisation avec le langage UML. Les différents diagrammes et leurs usages proposés par la notation sont présentés ; ce travail est réalisé au regard des usages possibles pour les modèles de conception qui nous intéressent.

5.2.1 Terminologie - la structure des modèles

Un **système** est une collection de sous-systèmes organisés pour accomplir un objectif et décrits par un ensemble de modèles, selon des points de vue qui peuvent être différents. Un **sous-système** est un groupement d'éléments dont certains constituent une spécification du comportement offert par les autres éléments contenus. Un **modèle** est une abstraction d'un système : il représente une simplification de la réalité complète et consistante, créé dans le but de mieux comprendre le système. Dans le contexte d'une architecture, une **vue** est une projection dans l'organisation et la structure d'un modèle du système, centrée sur un aspect de ce système (voir par exemple les vues proposés par Kruchten Figure 5.2). Un **diagramme** est une représentation graphique d'un ensemble d'éléments de modélisation. Un diagramme décrit une partie du contenu d'une vue. Les concepts utilisés dans les diagrammes sont des **éléments de**

[86] D'autres fonctionnalités comme le *reverse-engineering* existent mais nous ne listons que celles ayant un intérêt dans notre contexte.

modélisation qui représentent des concepts orientés-objets (comme les classes, les objets, les messages) ainsi que leurs relations (comme les associations, les dépendances, les généralisations). Ce sont les éléments de base pour la modélisation orientée-objet de systèmes. L'un de ces éléments peut servir dans différents diagrammes mais il conserve alors le même sens.

En résumé, un système représente ce que l'on est en train de développer, vu au travers de différentes perspectives par des modèles différents, ces vues étant présentées sous la forme de diagrammes.

5.2.2 Diagrammes UML

Le langage UML propose 13 diagrammes dans sa version 2.0 (10 dans les versions précédentes)[87].

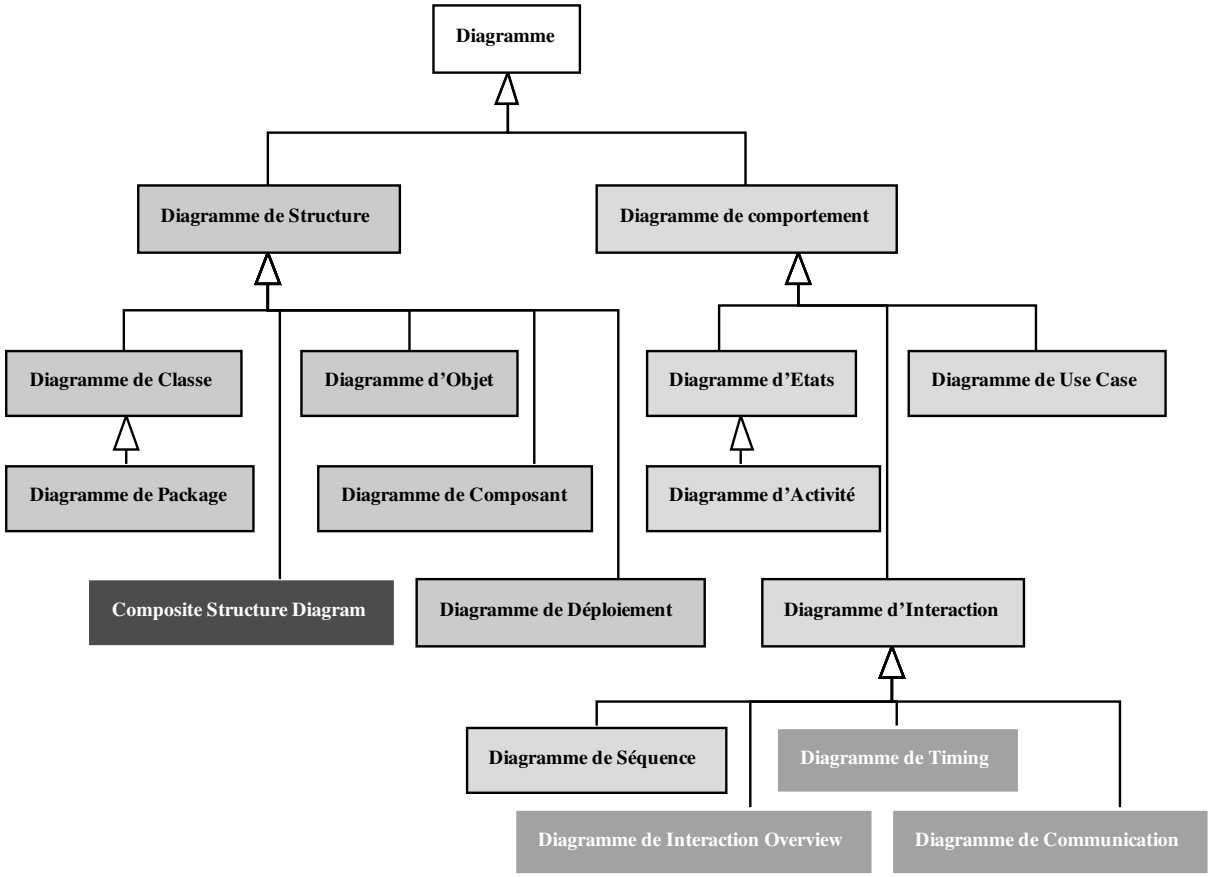

FIG. 5.3 – Classification des types de diagrammes UML.

La figure 5.3 illustre ces 13 diagrammes. Certains d'entre eux permettent de représenter la structure d'un système (les aspects statiques) ; d'autres représentent son comportement (aspects dynamiques). La notion d'héritage entre certains diagrammes (comme entre le diagramme d'activités et le diagramme

[87]Le diagramme de paquetages est souvent non compté car il s'appuie sur la notation des diagrammes de classe.

d'états) exprime le fait que la syntaxe (méta-modèle et notation) de ces diagrammes s'appuie sur celle d'un autre. Les diagrammes représentés par des rectangles foncés dans la figure 5.3 font référence aux nouveaux diagrammes proposés dans UML 2 (le diagramme de communication d'UML 2 fait référence au diagramme de collaboration d'UML 1.X). Les diagrammes de *timing*, de communication, de séquence et d'*interaction overview* sont regroupés sous l'appellation de diagrammes d'interaction.

Ces diagrammes permettent de décrire différentes perspectives d'un système mais ils peuvent également être utilisés pour modéliser une même vue à des niveaux d'abstraction variés. Ceci est possible de deux manières : en décrivant, dans un même modèle, un diagramme pour chaque niveau de détail ; ou en créant plusieurs modèles à des niveaux d'abstraction différents et en « traçant » les diagrammes d'un modèle à un autre.

Nous décrivons succinctement par la suite les diagrammes UML en donnant pour chacun d'eux leurs usages principaux ainsi que pour certains diagrammes une illustration annotée dans le but d'introduire les concepts élémentaires manipulés par le diagramme (cependant, nous n'explicitons pas le contenu et la sémantique de ces diagrammes). Ces diagrammes sont étudiés plus en profondeur dans de nombreux ouvrages [Mul98, Boo00, Fow03, Eri04]. Nous donnons aussi pour certains diagrammes des usages plus spécifiques pour le domaine éducatif que nous avons trouvé dans la littérature de ce domaine ; nous indiquons également, pour certains diagrammes, les usages pressentis spécifiques aux besoins en modèles de conception dédiés aux PBL coopératives médiatisées sur une plate-forme de formation à distance.

5.2.2.1 Diagramme de classe

Usage traditionnel. Un diagramme de classe exprime de manière générale la structure statique d'un système, en termes de classes et de relations entre ces classes. Il s'agit du diagramme le plus commun que l'on retrouve dans la modélisation orienté-objet. Généralement, il est utilisé pour modéliser le vocabulaire du système, modéliser les dépendances (au sens général), ou modéliser les schémas de base de données. Le diagramme de classe sert également de base pour d'autres diagrammes : diagrammes de paquetages, de composants et de déploiement. Les diagrammes de classe sont importants pour visualiser, spécifier et documenter les modèles de structure.

Usages et exemples en Génie Éducatif. Le diagramme de classe est généralement utilisé pour modéliser tout aspect statique d'un système éducatif. Il permet également de modéliser le système d'information de nombreux systèmes (comme ceux des plates-formes de formation à distance Ganesha ou OpenUSS). Dans de nombreux cas, le diagramme de classes sert à la description conceptuelle des méta-modèles (cf. travaux sur les EML du chapitre précédent) sous le nom de « modèle conceptuel ».

Intérêts pour nos modèles de conception. Dans notre contexte, le diagramme de classe pourrait permettre de conceptualiser notre langage autant que pour le décrire précisément. Du point de vue des modèles à produire avec ce langage, le diagramme de classe permettrait de représenter tous les aspects statiques d'un scénario (comme le modèle de ressources pédagogiques de [Hei04]). La modélisation des connaissances possédées (pré-requis) ou à acquérir pourrait être modélisée avec ce diagramme.

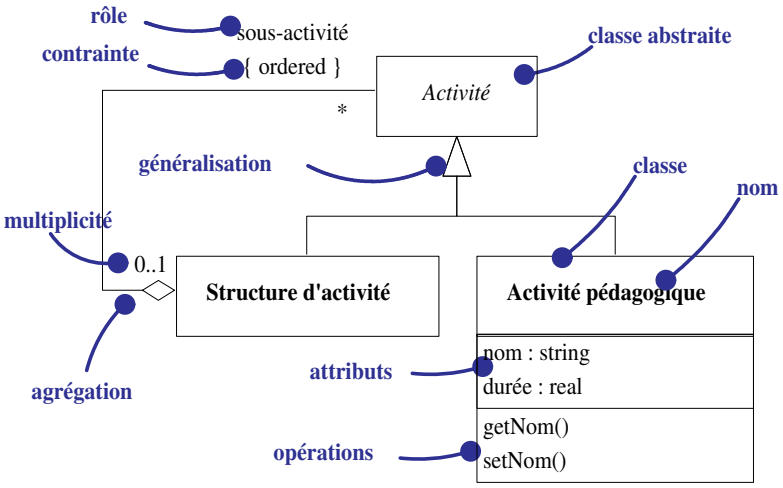

FIG. 5.4 – Un exemple simplifié et annoté de diagramme de classe.

5.2.2.2 Diagramme de cas d'utilisation

Usage traditionnel. Les diagrammes de cas d'utilisation décrivent, sous la forme d'actions et de réactions, les fonctionnalités du système ainsi que son comportement du point de vue d'un utilisateur. Ils permettent de définir les limites du système et les relations entre le système et l'environnement. Un cas d'utilisation est une manière spécifique d'utiliser un système. C'est l'image d'une fonctionnalité du système, déclenchée en réponse à la stimulation d'un acteur externe.

Usages et exemples en Génie Éducatif. Les diagrammes de cas d'utilisation sont généralement utilisés pour définir les rôles de différents utilisateurs des systèmes éducatifs (acteurs), les fonctionnalités du système (cas d'utilisation) offertes aux acteurs et pour chacune des fonctionnalités les acteurs ayant le droit de s'en servir [Lec03].

Intérêts pour nos modèles de conception. Ces diagrammes semblent convenir pour modéliser, en phase d'analyse de nos préoccupations, les différentes activités (encore définies à un haut niveau d'abstraction ; elle seront définies en détail dans la conception plus avancée). Les différents rôles (ceux présentés en 2.1.4.3) peuvent être définis dans ces diagrammes et associés aux activités dont ils auront la charge.

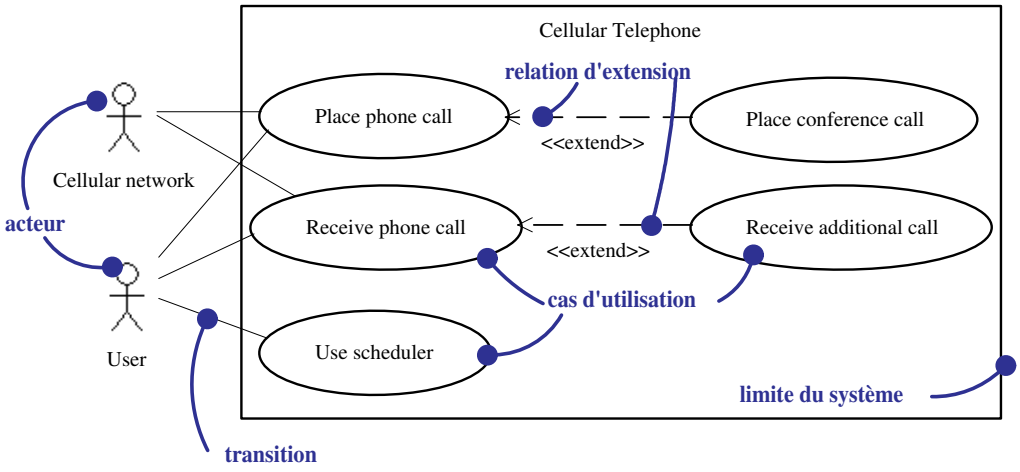

FIG. 5.5 – Exemple de diagramme de cas d'utilisation (tiré de [Fow03]).

5.2.2.3 Diagramme d'objets

Usage traditionnel. Les diagrammes d'objets ou d'instances montrent des objets et leurs liens. Ils permettent également de représenter la vue statique du système. Ils s'utilisent principalement pour représenter un contexte, avant ou après une interaction. Ils permettent de modéliser une « capture » du système à un moment donné au travers du rendu d'un ensemble d'objets, de leurs états et de leurs relations. Ils permettent ainsi, lorsqu'ils sont utilisés conjointement avec les diagrammes de classe, de valider et faciliter la compréhension des concepts spécifiés dans les diagrammes de classe.

Usages et exemples en Génie Éducatif. Le diagramme d'objets est souvent utilisé de pair avec le diagramme de classe dans de nombreux travaux.

Intérêts pour nos modèles de conception. Le diagramme d'objets pourrait permettre de préciser des exemples d'instanciation de nos modèles comme par exemple le nombre de personnes jouant les rôles définis au niveau des classes.

5.2.2.4 Diagramme de séquence

Usage traditionnel. Un diagramme de séquence permet de décrire les interactions entre objets en insistant sur l'ordonnancement temporel des messages. Il fait partie des diagrammes d'interaction dans lesquels on retrouve systématiquement des objets, des liens et des messages. Le diagramme de séquence est utilisé pour modéliser des flux de contrôle ordonnés selon le temps. Il ne décrit toutefois pas le contexte des objets.

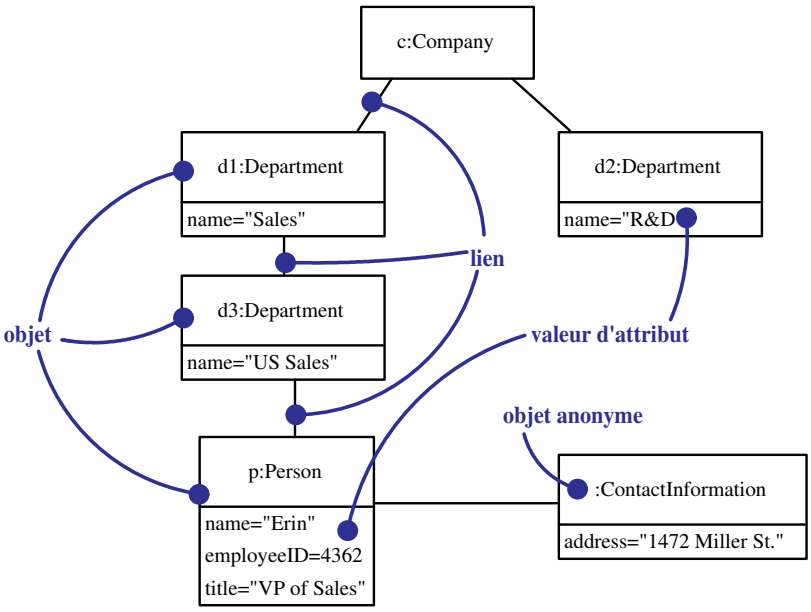

FIG. 5.6 – Exemple de diagramme d'objets (tiré de [Fow03]).

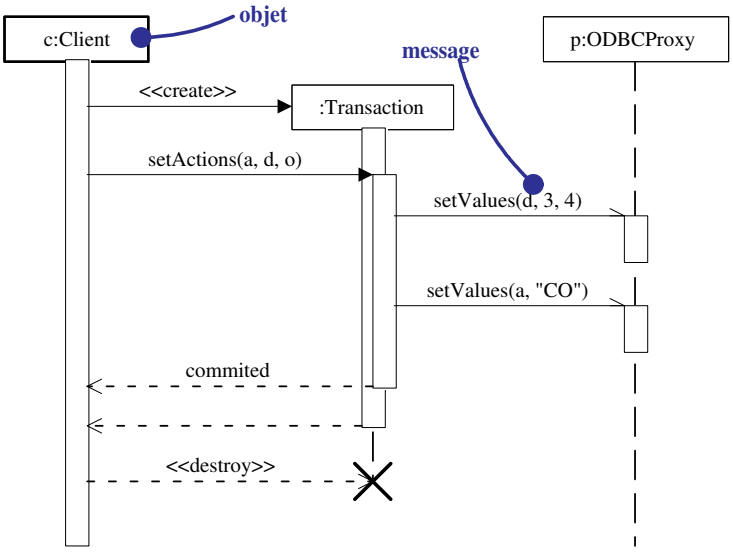

FIG. 5.7 – Exemple de diagramme de séquence (tiré de [Fow03]).

5.2.2.5 Diagramme de collaboration (UML 1.X) ou diagramme de communication (UML 2)

Usage traditionnel. Les diagrammes de collaboration font également partie des diagrammes d'interaction. Ils insistent plus particulièrement sur la structure spatiale statique permettant la mise en collaboration d'un groupe d'objets. Ils expriment à la fois le contexte d'un groupe d'objets (au travers des objets et des liens) et l'interaction entre ces objets (par la représentation des envois de messages). Les messages envoyés et reçus par les objets sont annotés sur les liens et précédés d'un numéro de séquence. Ces diagrammes sont parfois perçus comme des extensions des diagrammes d'objets.

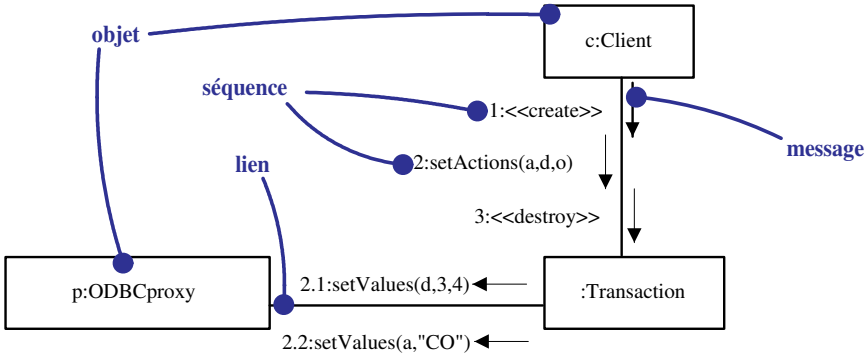

FIG. 5.8 – Exemple de diagramme de collaboration (tiré de [Fow03]).

5.2.2.6 Diagramme d'états-transitions

Usage traditionnel. Les diagrammes d'états-transitions visualisent des automates d'états finis du point de vue des états et des transitions. Ils permettent de décrire le comportement d'un système.

Usages et exemples en Génie Éducatif. Dans [Hei04], le diagramme d'états-transitions permet de décrire la scénarisation d'un module de formation de FOAD : les états sont des cours, les transitions indiquent l'enchaînement possible entre deux cours.

Intérêts pour nos modèles de conception. Le diagramme d'états-transitions pourrait permettre de préciser la dynamique des différents concepts manipulés dans une PBL et ayant la particularité d'évoluer entre différents états : les ressources à créer ou mises à disposition des apprenants par exemple, ou bien les représentations mentales des apprenants.

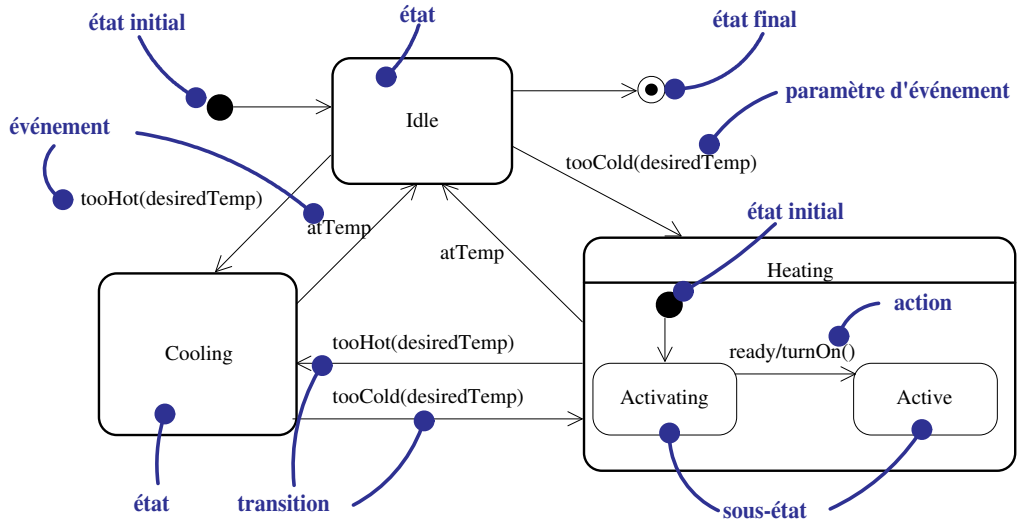

FIG. 5.9 – Exemple de diagramme d'états-transitions (tiré de [Fow03]).

5.2.2.7 Diagramme d'activités

Usage traditionnel. Le diagramme d'activités est une variante des diagrammes d'états-transitions organisée par rapport aux actions et principalement destinée à représenter le comportement interne d'une méthode (la réalisation d'une opération) ou d'un cas d'utilisation. Ils permettent également de décrire les flux d'informations (*workflows*) comme les flux de données (*dataflow*).

Usages et exemples en Génie Éducatif. Nous avons vu un usage particulièrement intéressant de ce diagramme (Figure 4.15) dans l'étude des EML. Ce diagramme sert de modèle d'analyse pour décrire les activités et les rôles les réalisant dans les situations d'apprentissage en général comme pour les PBL. Dans [Cos03], des diagrammes d'activités étendus (du point de vue de la notation et de la sémantique) sont utilisés pour spécifier des contenus didactiques, des activités d'évaluation et leurs relations.

Intérêts pour nos modèles de conception. Les diagrammes d'activités pourrait permettre de spécifier les scénarios de nos PBL coopératives à des niveaux d'abstraction variables selon la phase de conception (entre analyse et conception avancée). La structure de ces scénarios ou organisation des activités (décrite en termes de *act* et *scene* dans IMS-LD), indépendante de rôles particuliers, pourrait être décrite avec ces diagrammes ; de manière identique, le détail interne des activités en termes d'étapes ou d'actions attendues pourrait être précisé. Les diagrammes d'activités fournissent de nombreux éléments (les couloirs d'activités (*swimlanes* ou *partitions*), les barres de synchronisation (*fork* et *join*), les branchements conditionnels (*control branchs*) susceptibles de décrire l'apprentissage coopératif en termes d'activités individuelles et collaboratives réalisées séquentiellement ou en parallèle, ainsi que les choix de parcours d'un apprenant dans le scénario.

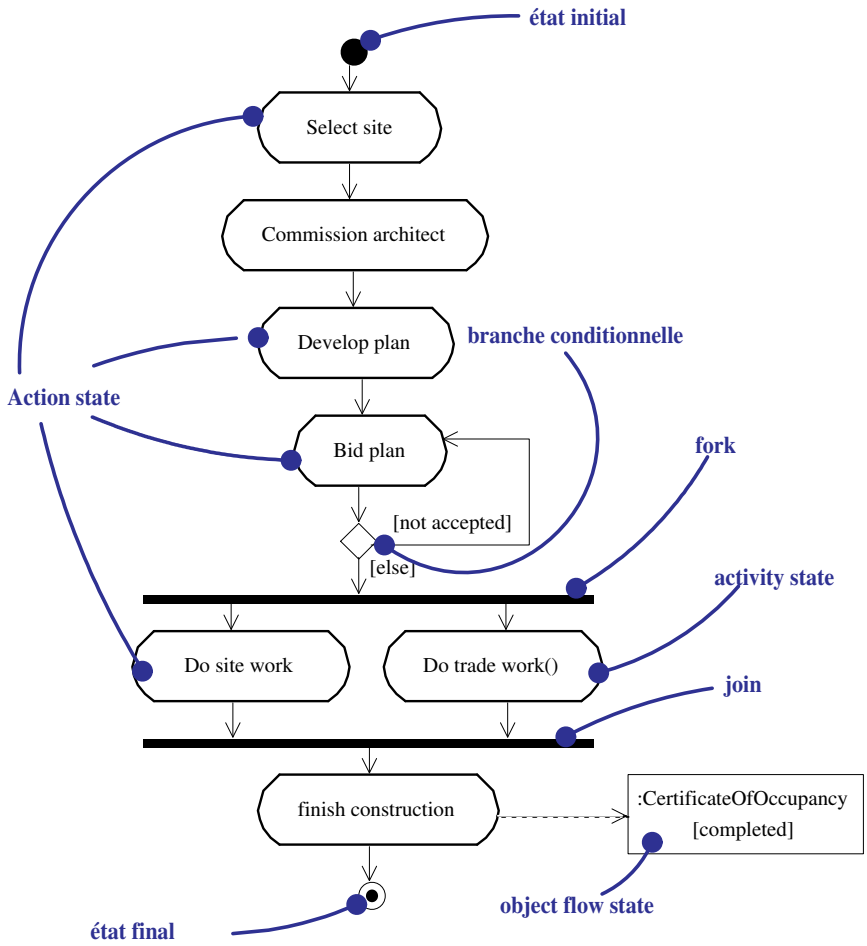

FIG. 5.10 – Exemple de diagramme d'activités (tiré de [Fow03]).

5.2.2.8 Diagramme de *composite structure* (UML 2)

Usage traditionnel. Ces nouveaux diagrammes proposés dans la spécification d'UML 2 permettent de décomposer hiérarchiquement un *classifier* (c'est-à-dire une classe, un composant, etc.) en décrivant sa structure interne. Malgré ses ressemblances méthodologiques avec les diagrammes de paquetage, le diagramme des structures composites montre des groupements d'exécution (*runtime*), tandis que les paquetages montrent des groupements de compilation (*compile-time*).

5.2.2.9 Diagramme de composants

Usage traditionnel. Les diagrammes de composants décrivent les éléments physiques et leurs relations dans l'environnement de réalisation. Ces diagrammes permettent de décrire les choix de réalisation. La sémantique et la notation des concepts de ces diagrammes ont beaucoup évolué avec la spécification UML 2. Les composants sont alors connectés au travers d'interfaces requises et fournies. Les composants peuvent également être décomposés grâce aux diagrammes de *composite structure*.

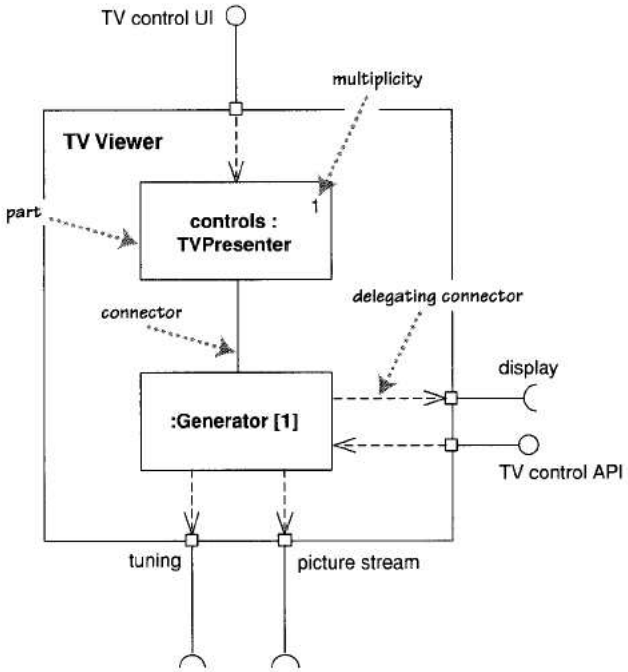

FIG. 5.11 – Exemple de diagramme de *composite structure* (tiré de [Fow03]).

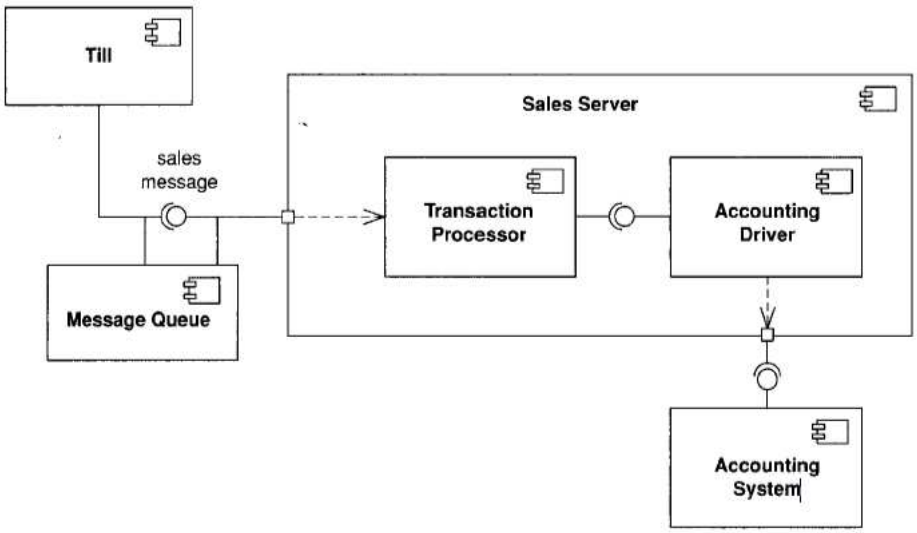

FIG. 5.12 – Exemple de diagramme de composants (tiré de [Fow03]).

5.2.2.10 Diagramme de déploiement

Usage traditionnel. Les diagrammes de déploiement permettent de décrire la disposition physique des différents matériels (les nœuds) qui entrent dans la composition d'un système et la répartition des programmes exécutables sur ces matériels.

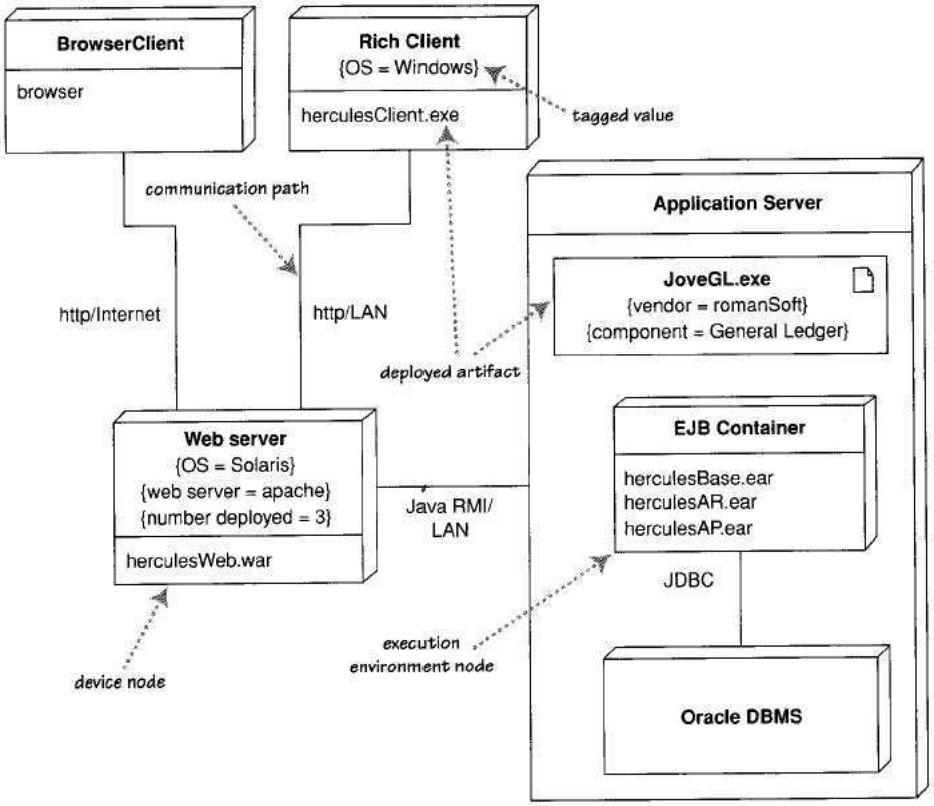

FIG. 5.13 – Exemple de diagramme de déploiement (tiré de [Fow03]).

5.2.2.11 Diagramme de paquetage

Usage traditionnel. Un paquetage est un élément de construction permettant de regrouper d'autres éléments dans des unités de plus haut niveau. En général, les paquetages permettent de regrouper les classes et de les structurer hiérarchiquement. Un paquetage contient donc des classes et des sous-paquetages. Chaque paquetage correspond en fait à un espace de nommage, ce qui signifie que les classes contenues doivent avoir un nom unique. Un diagramme de paquetage décrit alors les paquetages et leurs dépendances (non transitives). Il est alors utilisé pour décrire la structure des applications volumineuses.

Nous n'illustrons pas ce type de diagramme ni les prochains (voir [Fow03] pour des exemples d'illustration).

5.2.2.12 Diagramme de d'*interaction overview* (UML 2)

Usage traditionnel. Ces diagrammes sont un mélange des diagrammes d'activités et des diagrammes de séquence : ce sont des sortes de diagrammes d'activités dans lesquels les activités sont remplacées par des petits diagrammes de séquence. La nouveauté de ce type de diagramme ne permet pas de bien cerner leurs usages qui seront dévoilés par la pratique.

5.2.2.13 Diagramme de *timing* (UML 2)

Usage traditionnel. Les diagrammes de *timing* sont d'autres diagrammes d'interaction qui sont centrés sur la définition des contraintes de temps. Ils sont utiles pour montrer les contraintes de temps entre les changements d'états de différents objets.

5.3 Méta-modélisation avec UML

La section précédente s'intéressait à la modélisation avec UML. Cette section concerne davantage l'usage d'UML pour la méta-modélisation.

5.3.1 Introduction à la méta-modélisation

Nous avons vu précédemment en quoi la modélisation UML et, *a fortiori*, la modélisation en général sont des techniques permettant de représenter des systèmes. Or, un modèle n'est pas forcément une représentation exhaustive mais bien une abstraction du système, moins complexe qu'il ne l'est. Un modèle est donc une restriction à une description partielle d'un système pour des raisons de compréhension, lisibilité et manipulation. Ainsi, un modèle correspond à un point de vue sur un système : selon les objectifs du concepteur certains aspects sont pris en compte alors que d'autres sont omis. Le modèle est donc le résultat de l'application d'un filtre sur un système [Bre02]. Ce filtre est le méta-modèle[88] qui contraint la définition d'un modèle à partir d'un système. Un même système peut donc donner lieu à une multitude de modèles visant des aspects différents, donc basés sur des méta-modèles différents. Il est également possible et fréquent qu'un même méta-modèle permette d'extraire plusieurs modèles selon des choix arbitraires de représentation.

Tout méta-modèle est spécifique à un domaine. Il est ainsi composé d'une terminologie et d'assertions. La terminologie est l'ensemble des concepts, propriétés et de leurs relations alors que les assertions sont des règles supplémentaires permettant de contraindre les éléments de la terminologie. Les concepts définis dans la terminologie doivent être suffisamment génériques pour être réutilisés dans plusieurs modèles.

La possibilité de définir de multiples méta-modèles pour un même domaine a introduit le besoin d'un nouveau concept : le méta-méta-modèle. Son rôle est de fournir un langage unique pour la définition de tous les méta-modèles afin de faciliter leur interopérabilité. Ce méta-méta-modèle est le MOF (*Meta Object Facility* [OMG00]) adopté en 1997 par l'OMG.

[88] **Méta** vient du grec et signifie *à propos de*.

Une *architecture*[89] de modélisation basée sur quatre couches a été définie pour illustrer les différentes notions précédentes (Figure 5.14) : le niveau le plus haut (M3) correspond à l'unique méta-méta-modèle ; le niveau inférieur (M2) est celui des différents méta-modèles définissant chacun un formalisme propre à un domaine particulier ; au niveau encore inférieur (M1) se trouvent les modèles, chaque modèle étant basé sur un seul méta-modèle : au niveau le plus bas (M0) se trouvent « les instances »[90] des modèles.

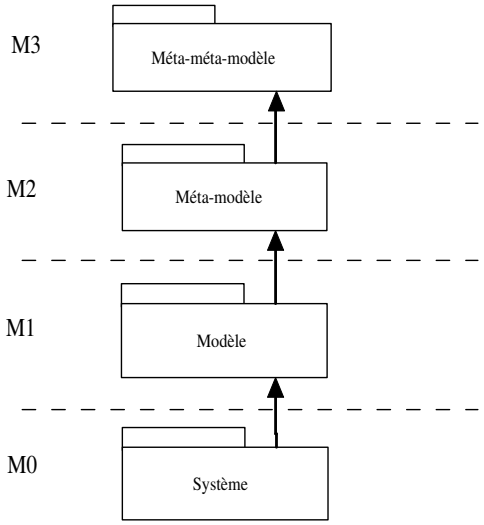

FIG. 5.14 – L'architecture de modélisation en couches [OMG03d].

5.3.2 Méta-modèles et UML

Initialement, UML était réflexif (UML était décrit en UML). La partie permettant cette réflexivité a été légèrement modifiée et a donné naissance au MOF. Une fois la standardisation du MOF effectuée, UML, qui était initialement exprimé dans son propre formalisme, est devenu, en théorie, un méta-modèle MOF - c'est à dire un méta-modèle exprimé avec le MOF. En théorie seulement car certains éléments de la version 1.3 du méta-modèle d'UML sont définis à partir de concepts non présents dans le MOF tels que les classes associatives. Un travail d'alignement du MOF et de UML est actuellement en cours.

Les concepts définis par le MOF étant très proches de ceux définis dans le coeur d'UML, la notation graphique d'UML a été retenue pour représenter également les méta-modèles MOF. Ainsi, l'apport principal d'UML dans le domaine de la méta-modélisation est sa notation graphique et ses concepts objets.

Une illustration de l'architecture quatre couches est montrée dans la figure 5.15. UML appartient donc à la couche M2, le MOF à la couche M3. Les modèles bâtis avec UML sont au niveau M1, les instances d'exécution du système sont au niveau M0.

[89] L'architecture d'un élément (ici la modélisation) est l'ensemble des règles définissant la structure de cet élément et les inter-relations entre ses parties.

[90] Ces instances ne représentent pas le système réel mais le système tel qu'il est perçu ; on parle alors de *runtime instances* ou de *snapshots*.

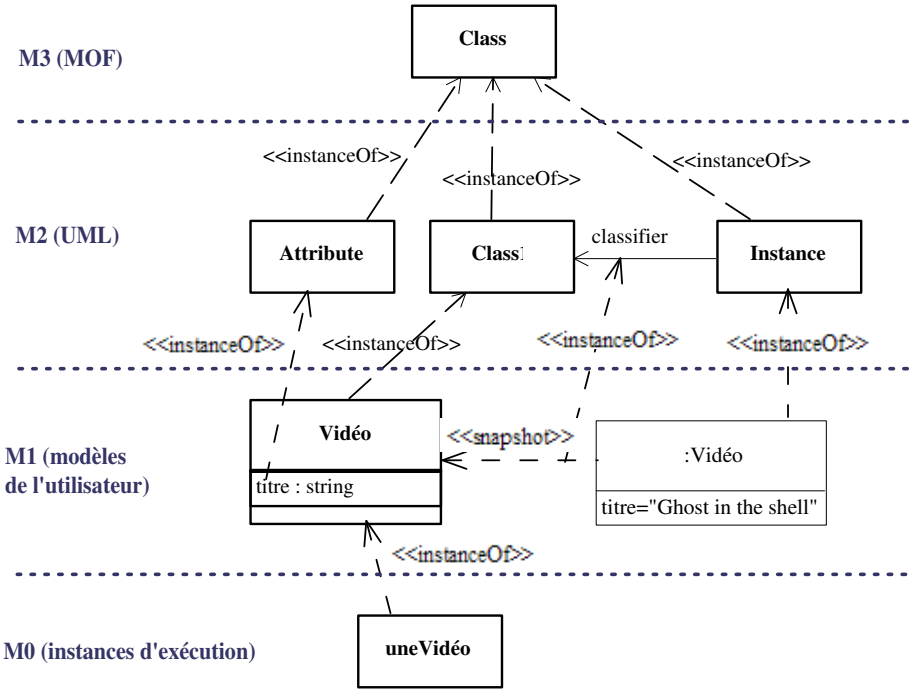

FIG. 5.15 – Exemple d'application des modèles en quatre couches (réadapté de [OMG03b]).

Basé sur le MOF, l'OMG a standardisé et continue de standardiser de nombreux méta-modèles tels que les méta-modèles d'UML, de *DataWareHouse* (CWM) [OMG01a], *Software Process Engineering Metamodel* (SPEM) [OMG01b], ou encore *Enterprise Distributed Object Computing* (EDOC) [OMG04].

5.3.3 Profil OMG et profil UML

Chaque méta-modèle standard couvre un domaine spécifique ayant un large champ d'applications. Chaque domaine est généralement décomposé en sous-domaines qui nécessitent un raffinement spécifique du méta-modèle standard.

Le méta-modèle standard UML fournit déjà des mécanismes d'extension appelés valeurs marquées, stéréotypes, contraintes (section 2.6 dans [OMG03d]). Ces éléments permettent d'adapter la sémantique d'UML sans changer le méta-modèle d'UML. C'est pour cela qu'ils sont souvent référencés comme *mécanismes d'extension légers*. Ils sont expliqués en détail dans la section suivante. A l'opposé, le standard MOF fournit des moyens pour étendre les méta-modèles, comme la définition de nouvelles méta-classes, qui sont référencés comme des *mécanismes d'extension lourds*.

La figure suivante (Figure 5.16) est un extrait du méta-modèle UML relatif aux mécanismes d'extension légers. On peut déduire des différentes associations que : un stéréotype peut être appliqué à plusieurs

éléments de modélisation et que tout élément de modélisation peut être stéréotypé plusieurs fois[91] ; de même pour les valeurs marquées (*tagged values*).

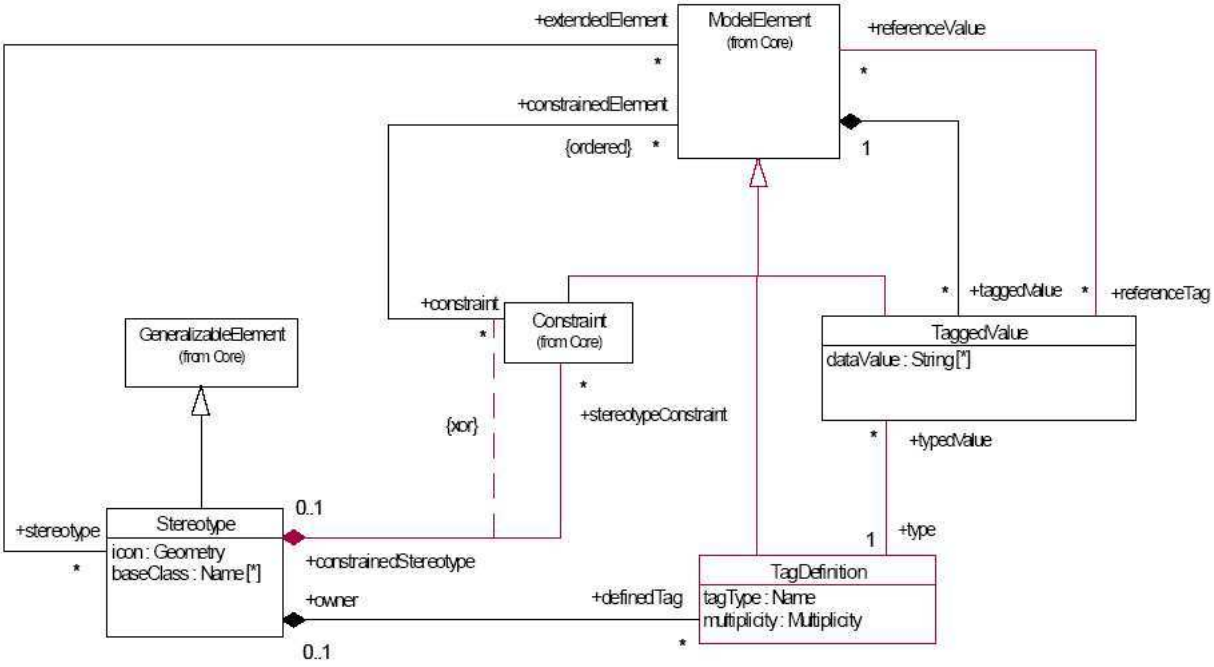

FIG. 5.16 – Les mécanismes d'extensions ([OMG03d]).

5.3.3.1 Définition d'un profil UML

La définition officielle d'un profil selon la dernière spécification d'UML (1.5) en vigueur est la suivante :

« *A profile is a stereotyped package that contains model elements that have been customized for a specific domain or purpose by extending the metamodel using stereotypes, tagged definitions, and constraints. A profile may specify model libraries on which it depends and the metamodel subset that it extends.* »

Un ensemble cohérent d'extensions est alors appelé *profil* [OMG99]. Ainsi, un profil est une spécification qui spécialise un ou plusieurs méta-modèles standards appelés les *méta-modèles de référence*. Le profil est alors dédié à un domaine spécifique de ces méta-modèles de référence (Figure 5.17).

Nous nous intéressons dans nos recherches au concept de *profil UML* [92]. Il regroupe ainsi des extensions d'UML qu'il organise afin de spécialiser ou raffiner UML dans un but spécifique.

Les différents éléments structurant un profil UML sont décrits ci-après :

[91] Dans la pratique, notre expérience d'utilisation de ces outils a montré qu'ils limitent l'application d'un seul stéréotype par élément de modélisation donné.

[92] Contrairement au profil UML, un profil (au sens large) peut également regrouper des mécanismes d'extensions *lourds* s'ils sont permis par le méta-modèle de référence. UML ne fournit pas ce type d'extensions.

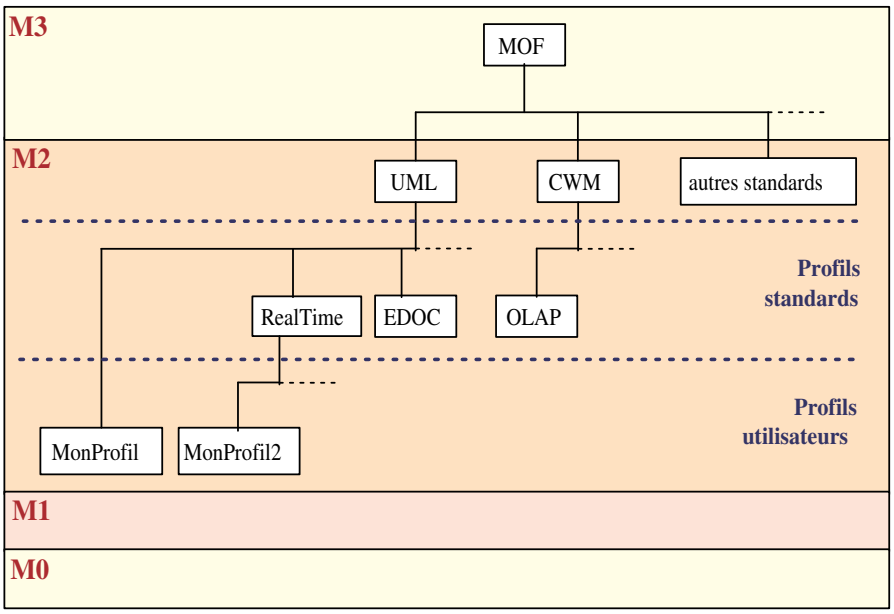

FIG. 5.17 – Les profils spécialisent des méta-modèles standards au niveau M2, et sont utilisés par les modèles au niveau M1 ([OMG99]).

Les éléments sélectionnés du méta-modèle de référence : un profil fournit la sélection du méta-modèle de référence qui constitue la focalisation particulière choisie. Cette sélection n'exclut pas les autres éléments du méta-modèle de référence, mais simplement spécifie ceux qui sont spécialisés. Par exemple, le profil EDOC [OMG04] sélectionne les éléments de modélisation statiques du méta-modèle d'UML mais pas les éléments de Use Case.

Les stéréotypes : un stéréotype est défini pour une méta-classe spécifique du méta-modèle de référence[93]. Dans un profil UML, le stéréotype crée une méta-classe UML virtuelle basée sur la méta-classe UML existante. Il fournit ainsi un moyen de classer les instances de cette méta-classe de base, et peut également aussi spécifier des contraintes additionnelles ou des valeurs marquées requises. Par exemple, le stéréotype « `TopLevelPackage` » étend la méta-classe UML `Package`. Lorsque ce stéréotype est appliqué à une instance de `Package`, il contraint l'instance à ne pas être comprise dans un autre paquetage.

Les définitions de valeurs marquées : la sémantique d'une valeur marquée est définie pour une méta-classe spécifique du méta-modèle de référence [94]. La définition d'une valeur marquée contient le nom des valeurs marquées correspondantes, le type des valeurs qu'elles peuvent prendre, la description de la sémantique et des contraintes s'appliquant à chaque valeur marquée correspondante. Comme mécanisme d'extension d'UML, la valeur marquée agit comme un attribut d'une méta-classe UML, permettant ainsi l'attachement arbitraire d'informations à une instance. Un ensemble de valeurs marquées peut être associé à un stéréotype afin d'être appliqué aux éléments de modélisation

[93] Plusieurs méta-classes peuvent avoir un stéréotype de même nom, mais il s'agit bien de stéréotypes différents.

[94] Comme pour les stéréotypes, deux définitions de valeur marquée ayant le même nom peuvent exister dans le même profil à condition de faire référence à des méta-classes différentes.

portant ce stéréotype. Par exemple, `EJBSessionType` est une valeur marquée associée au stéréotype `EJBSessionHomeInterface` de la méta-classe UML `Class`. Cette valeur marquée prend une des deux valeurs `Statefull` ou `Stateless` afin d'indiquer si le EJB Session Bean maintient ou non son état.

Les contraintes : les contraintes peuvent être définies au niveau d'une méta-classe particulière comme au niveau d'un stéréotype particulier. Elles permettent de spécialiser davantage la sémantique des éléments du méta-modèle de référence utilisés dans le profil. Cette spécification est écrite sous la forme d'une expression dans un langage de contrainte particulier. Le langage de contrainte formel utilisé par le méta-modèle UML est le Object Constraint Language (OCL). Mais les contraintes peuvent être spécifiées de manière informelle en langage naturel. Une ou plusieurs contraintes peuvent être appliquées à tout élément de modélisation pour spécifier l'utilisation de ses instances. De plus, les contraintes peuvent être associées à un stéréotype pour ainsi ne s'appliquer qu'à des éléments de modélisation classés par ce stéréotype.

Les descriptions : il est possible de préciser la sémantique d'un profil (comme des éléments qu'il contient) par des descriptions en langage naturel. Par exemple, les objectifs d'un profil ou sa compatibilité avec d'autres profils peuvent ainsi être décrits en détail.

La notation : la notation d'UML peut être personnalisée par le mécanisme des profils : définition d'*icônes* associés aux stéréotypes ; *disposition pour les diagrammes et capacités de filtrage* (décrits *via* les règles de présentation).

Les règles : les profils doivent être capables de définir des règles dédiées à leur domaine spécifique. Elles peuvent être de différents types :

- Règles de transformation : pour exprimer comment un modèle peut être transformé pour être modélisé ou implémenté vers un but spécifique. Par exemple, le profil CORBA exprime comment un modèle UML peut être transformé en une implémentation CORBA. Certaines règles de transformation permettent la définition de produits de développement ou bien assistent ou automatisent le développement de certains types d'activités (génération de code, *design-patterns*).
- Règles de validation : pour vérifier que le modèle possède les bonnes propriétés du domaine du profil. Ces règles vérifient les critères de cohérence sur le modèle. Elles peuvent être exprimées à l'aide de contraintes ou tout autre mécanisme : les *Well-formedness rules*.
- Règles de présentation : pour définir quel types d'éléments de modélisation doivent apparaître dans tel type de diagramme et indiquer aussi quelles informations doivent être cachées.

La figure 5.18 illustre un exemple de profil UML réduit. Un seul stéréotype **« Business_Object »** est créé par rapport à la méta-classe **Class** de référence ; la valeur marquée **{analysis}** est également définie. Des exemples de règles de validation, de présentation et de transformation sont donnés.

Un autre exemple est illustré par la figure 5.19 : il s'agit d'une version simplifiée pour la définition d'un profil pour les EJB ; le profil est défini ici sous la forme d'un diagramme de classe décrivant les stéréotypes et les définitions de valeurs marquées du profil EJB ainsi que les éléments du méta-modèle d'UML auxquels ils se rattachent[95].

[95] La notation utilisée dans l'exemple pour la notation des mécanismes d'extensions est celle définie dans la spécification UML 2.0

Eléments	Package, Classe, Use Case
Extensions	Stereotype: <<*Business_Object*>> (classe).
	Tagged Value: {analysis}
Règles de Validation	Métriques: 10 classes max par packages ; 10 opérations max par classes.
	Tous les acteurs doivent coopérer avec au moins un Use Case
	Détection des objets sans classes, des messages sans opérations
Règles de Présentation	Diagrammes de classes et de Use Case. Seules les opérations publiques sont affichées.
	Visualisation particulière des classes *Business_Object*.
Règles de Transformation	Plan type de documentation.
	Complétion automatique des diagrammes

FIG. 5.18 – Exemple de contenu d'un profil UML d'analyse ([Sof99]).

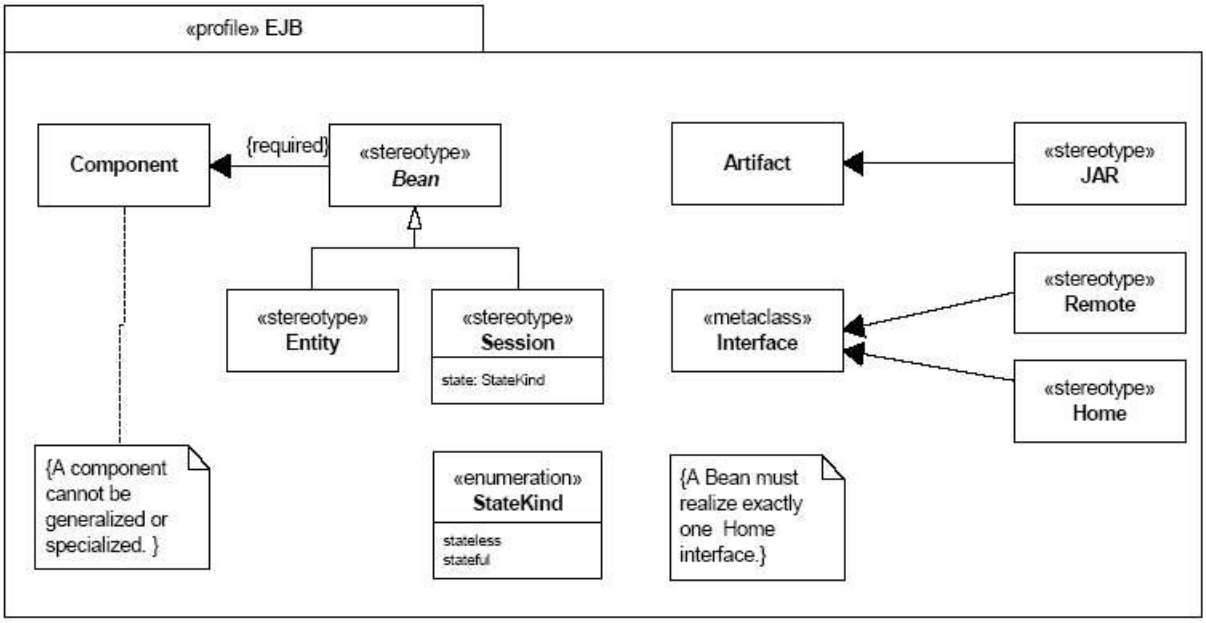

FIG. 5.19 – Exemple d'un profil simplifié pour les EJB (tiré de [OMG03b]).

La figure 5.20[96], basée sur la figure 5.15 précédente, illustre la conceptualisation d'un stéréotype comme une nouvelle méta-classe virtuelle basée sur une méta-classe d'UML existante, ainsi que la conceptualisation d'une définition de valeur marquée comme la définition d'un attribut sur une méta-classe. Dans cet exemple, le stéréotype <<**DVD**>> peut être perçu comme une nouvelle méta-classe d'UML. L'attribut *zone* qui lui est rattaché permet alors l'attachement arbitraire, au niveau des modèles produits, d'une information sur la zone du DVD. Un exemple d'utilisation de ces concepts au niveau M1 est donné : le stéréotype est appliqué sur la classe **Vidéo** *via* la notation caractéristique des stéréotypes (nom donné entre chevrons) ; une valeur marquée *zone* est définie sur cette classe et a pour valeur 2.

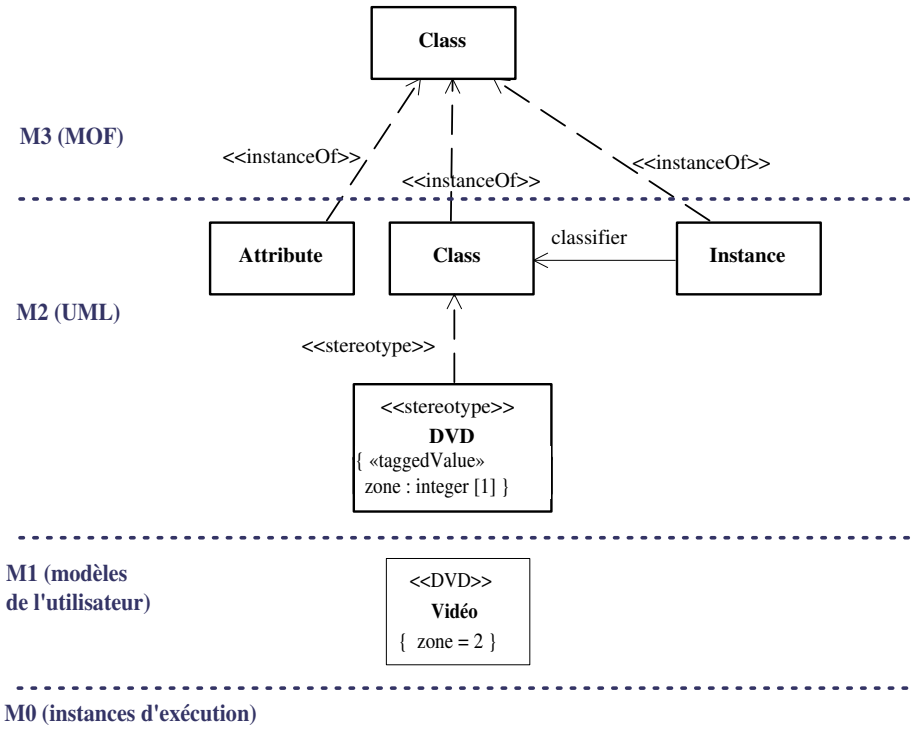

FIG. 5.20 – Exemple de création et d'utilisation des stéréotypes et définition de valeur marquée dans le cadre de l'architecture des modèles en quatre couches (figure de [OMG03b] étendue).

Il existe encore de nombreux mécanismes relatifs aux profils détaillés dans [OMG99]. Ces mécanismes concernent plus les relations entre différents profils que leur caractérisation. Ainsi nous préférons uniquement les citer sans les approfondir : généralisation/héritage entre profils, dépendance d'importation, compatibilité entre profils, sélection de profil, échange de profils, ou encore regroupement de profils.

[96]La notation utilisée dans cet exemple pour la notation des mécanismes d'extensions du niveau M2 est celle définie dans la spécification UML 1.X.

5.3.3.2 Objectifs détaillés d'un profil UML

Nous avons vu qu'un profil UML permet de spécialiser le méta-modèle d'UML pour un domaine d'utilisation particulier et qu'il regroupe de manière cohérente des extensions du méta-modèle UML.

Lors de la construction d'un profil UML, d'un côté certaines notations d'UML sont favorisées tandis que de l'autre côté certaines utilisations de diagrammes ou de notations dans certains diagrammes sont écartées. Ces choix réalisés vis-à-vis de besoins se rapprochent alors d'une méthode bien identifiée. Toutefois, ces choix ne remettent pas en question la sémantique et la notation du standard UML. Il s'avère donc toujours possible de réintégrer les vues ignorées dès que les besoins évoluent.

Un profil UML fournit, dans le cadre de son utilisation dans le processus de mise en oeuvre d'un système ou d'une application, une représentation ou une méthode partagée par tous les acteurs d'un domaine.

Le profil est parfois utilisé comme *outil* (graphique) *de communication et partage des modèles*. Comme l'indique [Cav02], le formalisme du profil est alors *choisi parce qu'il est indépendant de toute technologie et langage informatique et permet donc de choisir/substituer l'outil d'exploitation (par simulation des modèles)*.

Les profils UML apportent un mécanisme permettant de spécialiser UML pour chaque contexte de travail comme par exemple l'analyse, la conception technique, le codage, etc. Ils introduisent des notions plus adaptées au type de travail courant, des règles de modélisation spécifiques, des règles de production de délivrables, et des modes de présentation des modèles adaptés [Sof99]. Par exemple, un profil UML « C++ » va définir les extensions nécessaires pour réaliser une conception détaillée UML pour C++ ; il va également vérifier que le modèle UML spécialisé respecte des contraintes de modélisation spécifiques pour C++ ; il permet aussi de présenter des diagrammes UML à l'intention des programmeurs C++ ou des concepteurs UML ; pour terminer, le profil correspondant va permettre de produire un code C++ conforme aux règles de qualité de programmation en C++, en appliquant des règles de traduction modèle/code recommandées par les guides de programmation.

L'outillage d'un profil UML constitue un outil puissant pour spécifier le processus de développement mais aussi pour le **guider**. En effet, à chaque étape de développement, les profils permettent d'exprimer l'utilisation d'UML, les produits de développement attendus et les règles que le modèle doit respecter. Les règles de transformation accompagnant le profil participent à définir des produits de développements, des règles de génération de code et des *design patterns* qui permettent, à leur tour, d'assister et d'automatiser le développement spécifiquement pour chaque type d'activité.

5.3.3.3 Démarche d'élaboration d'un profil

Pour définir un profil UML à partir du méta-modèle d'UML, les règles suivantes doivent être généralement suivies :

1. Identifier le sous-ensemble des classes du méta-modèles d'UML qui seront incluses dans le profil.
2. Pour chaque classe du méta-modèle, identifier une *classe de base* dans le sous-ensemble du méta-modèle d'UML qui, une fois stéréotypée de manière appropriée, jouera le rôle de la classe du méta-modèle (cette technique est spécifiée dans la spécification d'UML [OMG03d] section 3.18).

3. Pour chaque attribut et association du méta-modèle, définir un moyen de les émuler. Les attributs, pour lesquels aucun méta-attribut du sous-ensemble du méta-modèle d'UML ne correspond, peuvent être émulés avec la création d'une valeur marquée. La plupart des associations ont une association analogue dans le méta-modèle d'UML.
4. Pour les éléments du sous-ensemble du méta-modèle UML qui ont un lien possible (*mapping*) avec des concepts du méta-modèle à transformer mais qui ne sont pas directement utilisés pour l'émulation, montrer comment ils sont reliés aux concepts du méta-modèle.
5. Donner des contraintes additionnelles aux éléments du méta-modèle UML qui sont impliqués par l'utilisation du profil.
6. Définir des notations (icônes) pour les concepts du méta-modèle représentés par des stéréotypes UML.

5.3.3.4 Exemples de profils UML

Il existe de nombreux profils ayant fait l'objet d'une standardisation de l'OMG [97] : le profil *Software Process Engineering Metamodel* (SPEM), le profil CORBA, le profil *CORBA Component Model* (CCM), le profil *Enterprise Application Integration* (EAI), le profil *Enterprise Distributed Object Computing* (EDOC)[98], un profil *QoS and Fault Tolerance*, ou encore un profil *Schedulability, Performance and Time*.

De nombreux autres profils non standardisés mais issus de travaux de recherche ont été produits :
- les profils TURTLE et TURTLE-P dédiés à la validation d'architectures distribuées [Apv04, Apv03],
- le profil DAMeSI pour l'évaluation des performances des systèmes d'automatisation distribués [Cav02],
- des profils dédiés à la modélisation des architectures d'applications Web [Con00, Hen01] ; ces profils ayant inspiré des travaux similaires pour la modélisation d'applications éducatives basées Web de type hypermédias adaptatifs [Ret02, Ret03].
- UML-G : un profil pour modéliser les collecticiels (*Groupware*)[99] [Rub03].

5.3.4 Bilan

Ce chapitre a permis d'étudier le langage UML, l'orientation choisie dans nos travaux pour élaborer notre propre langage (orientation définie en 1.3.2). Les objectifs de ce chapitre étaient doubles :

1. étudier la modélisation UML. Ceci a permis de mettre en exergue les diagrammes susceptibles, par leurs usages « traditionnels » et dans le domaine éducatif, de pouvoir représenter des vues des PBL coopératives. Nous avons relevé les diagrammes suivants : diagramme de classe, de cas d'utilisation, d'activités et d'états/transitions.
2. étudier la méta-modélisation UML. Le concept de profil UML, mécanisme d'extension léger défini et proposé par UML, convient comme support pour notre langage dédié au domaine des PBL.

[97] Ils sont disponibles sur le site Web de l'OMG : http ://www.omg.org/technology/documents/modeling_spec_catalog.htm
[98] EDOC est lui-même composé d'un ensemble de spécifications comme un profil UML pour Java et les EJB, un profil pour les patrons (*pattern*), un profil pour ECA, un profil pour le MOF et un profil sur les relations
[99] http ://www.uml-g.org/

Deuxième partie

Contribution

Chapitre 6

Proposition globale

Sommaire

Cette seconde partie du mémoire vise à présenter, expliquer et discuter notre contribution vis-à-vis des conclusions de l'étude réalisée en première partie. Ce chapitre a également pour objectif principal de présenter notre contribution qui sera détaillée dans les chapitres suivants.

Pour cela, nous synthétisons, dans un premier temps, les résultats de l'état de l'art conformément aux objectifs fixés dans l'introduction générale (section 6.1). Ensuite, nous présentons globalement notre proposition sous la forme de deux contributions complémentaires : le langage CPM (qui constitue la majorité de la contribution) et le modèle de composant CPL (section 6.2). Pour terminer, nous rappelons l'organisation des chapitres suivants composant cette seconde partie (section 6.3).

6.1 Bilan de l'état de l'art et révision des objectifs initiaux

Cette section synthétise les conclusions issues des différentes études menées dans la première partie, puis, révise et re-positionne nos objectifs vis-à-vis de l'existant.

6.1.1 Bilan de l'état de l'art

Nous avons étudié, dans un premier temps, le cadre de travail que nous nous étions fixés pour la conception d'EIAH. L'apprentissage par situations-problèmes coopératives correspond à l'intention didactique que nous souhaitons articuler avec les environnements informatiques que sont les plates-formes de formation à distance.

L'étude théorique des PBL a permis de mettre en exergue des concepts liés à ce « modèle pédagogique » de l'apprentissage et a également permis d'extraire les différents besoins ou questions auxquels l'équipe chargée d'élaborer la PBL se doit de répondre. Tous ces résultats ont une place importante en amont du processus de conception décrit en section 1.2.2.2 : l'expression initiale des besoins. Les concepts des PBL correspondent au vocabulaire que notre langage devra intégrer afin de permettre l'expression de modèles amont décrivant l'élaboration initiale de PBL. De même, les besoins que nous avons identifiés peuvent correspondre aux différents usages des modèles amont dont nous souhaitons faciliter la construction.

Nous avons également étudié de manière approfondie les plates-formes de formation à distance ou LMS (*Learning Management System*) : usages, fonctionnalités, structures, architectures, etc. Les objectifs étaient doubles : examiner les caractéristiques de ces environnements informatiques afin que les modèles de conception puissent y faire référence et sonder en quoi l'évolution actuelle des composants éducatifs pour les plates-formes est susceptible d'améliorer les apprentissages. Nous avons abouti aux conclusions que ces LMS s'enrichissent continuellement de nouvelles fonctionnalités visant à élargir leur champ d'usage pour des situations d'apprentissage de plus en plus complexes, comme les PBL par exemple. Cette évolution est réalisée dans la pratique par un changement technique aux niveaux architecturaux et structurels de ces plates-formes par le biais de l'utilisation des composants. Toutefois, malgré ces évolutions, les services ou fonctionnalités proposés par les plates-formes sont encore *transversaux* aux apprentissages (intégrés pour un contexte d'activité donné).

Nous avons ensuite étudié des langages de modélisation pour la conception de formations (langage à base de méta-données, ontologies éducatives et langages de modélisation éducatifs - EML), ce qui nous a conduit à remarquer que :

- ces langages interviennent tardivement dans la conception (on parle alors de *conception avancée*) ;
- ils cherchent à organiser et à structurer les contenus (ressources ou objets d'apprentissage) et les conteneurs (scénario pédagogique, unité d'apprentissage) dans le but de répondre à des besoins de réutilisation, assemblage, interopérabilité, etc. ;
- ils font l'objet de standards et de normes ;
- ce sont des langages formels dont les modèles produits sont concrètement spécifiés sous des formes dérivés du langage XML afin d'être facilement interprétables par la machine.

Les langages à base de méta-données s'adressent essentiellement aux fournisseurs de ressources pédagogiques. En revanche, les ontologies éducatives et les EML semblent mieux adaptés pour guider et aider l'ingénieur pédagogique. Toutefois, l'utilisation et la compréhension de ces langages posent le problème de l'expertise et de l'outillage nécessaires. Vis-à-vis de la description de situations d'apprentissage, les EML correspondent davantage à nos objectifs que les ontologies. Toutefois, ces langages ne peuvent décrire les situations-problèmes qu'à un niveau de conception très avancée dans lequel les modalités et les conditions d'exécution ont déjà été fixées. L'étude des EML a également mis en avant les particularités des environnements informatiques pris en compte dans les modèles de conception produits : il s'agit uniquement des objets d'apprentissages et des services (logiciels). De plus, les prérequis d'interopérabilité, de reproduction et de réutilisation (définis en 4.3.2.1) pour les descriptions d'unités d'apprentissage impliquent que les modèles de conception avec les EML ne doivent contenir aucune information liée à l'instanciation (les personnes jouant les rôles définis) comme à l'exécution (modalités d'exécution, types de médias, usage en présentiel, à distance ou en apprentissage « mixte », etc.).

Des modèles basés sur le langage UML servent à illustrer les résultats d'analyse sur lesquels se basent les EML. Par exemple, la spécification d'IMS-LD encourage l'utilisation d'UML pour ces phases plus amont. De plus, nous avons montré que des méthodes d'ingénierie pédagogique comme MISA utilisent des notations graphiques pour représenter les modèles amont de conception de situations d'apprentissage. Ces résultats confortent notre orientation de départ concernant l'utilisation d'UML pour la description de modèles de conception. Nous avons alors étudié la modélisation UML afin de mettre en valeur les notations (diagrammes UML) susceptibles de représenter des vues de situations d'apprentissage de type PBL. La spécialisation d'UML, sous la forme de profil UML, a également été examinée dans la perspective de l'élaboration de notre langage sous cette forme.

6.1.2 Positionnement vis-à-vis de l'existant

Tout d'abord nous positionnons les différents langages existants que nous avons étudiés. Pour cela nous décomposons, dans un premier temps, la phase de conception en trois étapes (en gardant toutefois la correspondance avec celles définies en section 1.2.2.2) ; puis, dans un second temps, nous positionnons l'existant pour chacune d'entre elles :

1. l'étape d'expression initiale des besoins : les modèles sont souvent informels et en langage naturel afin de garantir un dialogue entre les différents intervenants (enseignants, pédagogues, ou encore didacticiens) ;
2. l'étape d'analyse/conception : les modèles produits sont moins abstraits ; l'utilisation du langage UML pour l'analyse est préconisée par des spécifications comme IMS-LD ;
3. l'étape de conception avancée : les différents langages que nous avons étudiés (méta-données, ontologies et EML) interviennent à cette étape.

La figure 6.1 illustre ce constat.

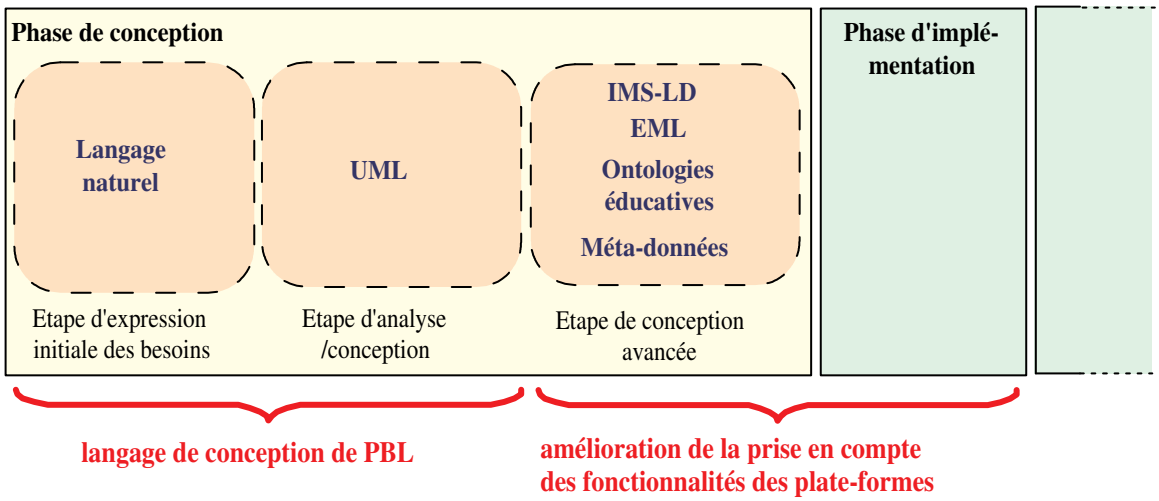

FIG. 6.1 – Positionnement des langages existants utilisés pour la conception de situations d'apprentissage et positionnement de notre contribution.

Vis-à-vis de nos objectifs initiaux (définis en 1.3.1) et du constat précédent, nous positionnons ainsi notre contribution :

- proposer un langage de modélisation graphique pour la conception de PBL en amont des langages de type IMS-LD ; il doit couvrir alors la phase de conception de l'étape d'expression initiale des besoins jusqu'à l'étape de conception ;
- proposer une meilleure intégration des fonctionnalités des plates-formes pour la conception avancée de PBL et leur implémentation.

6.2 Notre proposition

Nous proposons dans ce travail une contribution comportant deux aspects complémentaires :

- un langage facilitant l'élaboration de modèles pour la conception de PBL coopératives sans prise en compte de l'environnement informatique support : le langage CPM (*Cooperative Pbl Metamodel*). En effet, nous voyons cette contribution au niveau du manque actuel que nous avons mis en exergue précédemment.
- un modèle de composants éducatif dont l'objectif est de réduire l'écart entre les modèles de conception construits grâce au langage de la première proposition et les services actuellement rendus par les LMS basées composants : le modèle CPL (Composant Pédagogique Logiciel).

La première proposition constitue la majeure partie de notre contribution et répond à des attentes actuelles de la communauté française des EIAH concernant l'usage d'UML. Elle est expliquée plus en détail dans la prochaine section.

La deuxième proposition a fait l'objet de nombreuses actions de recherches dans nos travaux. Dans ce mémoire, nous présentons les résultats constituant un complément à la première proposition (ils traitent du lien entre la phase de conception et la phase d'implémentation).

6.2.1 Première proposition : le langage CPM

Cette section définit plus précisément notre première partie de proposition. Pour cela, nous nous interrogeons sur les usages du langage et les usages des modèles produits afin de mettre en évidence les besoins que le langage devra satisfaire.

6.2.1.1 Les usages du langage

Le langage CPM a pour principal usage de permettre l'élaboration de modèles pour la phase de conception de PBL coopératives qui seront exécutées sur des plates-formes de formation à distance.

Il s'adresse à l'équipe pluridisciplinaire de conception et plus particulièrement à l'ingénieur pédagogique dont le rôle est défini précisément en § 1.2.2.4.

D'autres types de modèles peuvent être nécessaires pour les autres phases du processus global de conception (voir 1.2.2.2), comme par exemple les phases d'implémentation, de test, de déploiement, d'évaluation, mais également pour les phases du processus d'utilisation (1.2.2.3) : modèles d'instanciation, modèles d'exécution. Ces modèles pourraient également s'exprimer à l'aide du même langage ou faire l'objet d'un autre langage plus approprié.

6.2.1.2 Les types de modèles de conception à spécifier et leurs usages

La conception de situations-problèmes est un processus d'ingénierie pédagogique. Au début du processus la situation d'apprentissage n'est pas clairement définie. Elle peut même être très abstraite et peut correspondre, à un certain niveau, à l'idée de l'enseignant/pédagogue.

L'étape de conception peut être considérée comme un processus dont l'objectif est d'aboutir à une description de plus en plus précise et formelle de l'apprentissage visé sous la forme d'un scénario pédagogique. Un scénario pédagogique est défini alors comme « *le résultat manipulable de la modélisation d'une situation d'apprentissage* » ou plus précisément comme « *la description du déroulement d'une situation d'apprentissage en termes de rôles, d'activités et d'environnement nécessaire à sa mise en œuvre, mais aussi en termes de connaissances manipulées* » [Per03].

Les modèles intermédiaires, produits tout au long du processus de conception, prennent toujours en compte des éléments de pédagogie mais font de moins en moins référence explicitement aux théories et modèles sous-jacents (dans notre contexte l'apprentissage par situation-problème) au fur et à mesure de la progression dans la phase de conception.

En revanche, tous les modèles partagent des caractéristiques générales que l'on retrouve dans la littérature sur la conception éducative. Nous les avons aussi étudiés dans la section consacrée aux langages existants pour la description d'apprentissage [Paq04a] :

- des activités, ou des phases au travers desquelles le processus d'apprentissage est exposé ;
- un système social, les rôles et les relations entre les apprenants et les autres acteurs impliqués dans l'apprentissage, leurs coopérations, collaborations ;
- le *feedback* du formateur à travers ses analyses des activités conduites par les apprenants ;
- un système « support » incluant les ressources et matériaux comme les services fournis par les formateurs ou tuteurs ;
- les résultats attendus, qui représentent l'objectif de l'apprentissage.

Nous élargissons toutefois l'usage des modèles de conception à la prise en compte dès la conception d'informations relevant de la régulation de l'apprentissage, donc liées à l'exécution *effective* de la situation d'apprentissage : par exemple, indiquer au niveau d'une ressource, qui sera produite par les apprenants, comment elle devra être évaluée. Ces informations, indications qualitatives ou quantitatives sur la régulation des apprentissages (tutorat humain ou automatique, suivi des activités, etc.), bien qu'intervenant à l'exécution, sont également issues des réflexions sur l'élaboration de la situation d'apprentissage ; nous souhaitons donc permettre leur description et leur rattachement aux modèles de conception.

Les modèles CPM[100] utilisent la notation UML, il est donc possible de faire les correspondances suivantes :

- la dimension horizontale correspond à l'ensemble des modèles CPM produits (des modèles différents selon l'étape dans la phase de conception) ;
- la dimension verticale (les modèles capturent différentes perspectives de la situation-problème) correspond, pour chaque modèle, à la description de plusieurs vues complémentaires :
  - la vue structurelle : il s'agit de la scénarisation des activités en phases (sur plusieurs niveaux hiérarchiques différents) ; la dynamique de l'enchaînement des activités (liée à des conditions d'exécution) est également définie dans cette vue ;

[100] Raccourci d'écriture pour indiquer les modèles produits avec le langage CPM.

- la vue pédagogique : les objectifs, les pré-requis, les pré-conditions, les post-conditions, les éléments d'aide pour la régulation ou le tutorat, etc. ;
- la vue sociale : les relations entre les activités, les ressources et les rôles.

Chaque vue d'un modèle est représentée par un ensemble de diagrammes.

6.2.1.3 Les utilisations pour nos modèles

Nous avons recensé les utilisations suivantes des modèles bâtis avec notre langage pour la conception de situations-problèmes :

Abstraction de la complexité : de manière générale, les modèles permettent de décrire les situations d'apprentissage afin d'en abstraire la complexité et donc d'en aider l'élaboration.

Communication : dans les phases préliminaires de la conception, l'enjeu majeur consiste à utiliser des représentations permettant un dialogue entre les différents participants à la conception (experts du domaine, pédagogue, didacticien, etc.).

Simulation : c'est en essayant de modéliser le scénario d'apprentissage pour la PBL que les concepteurs aboutiront à son élaboration ; le langage permet ainsi, à travers les modèles et vues qu'il produit, de *simuler* certains points de vue de l'apprentissage et d'identifier les manques pendant l'analyse.

Documentation : les modèles produits pendant la phase de conception permettent de documenter le travail réalisé afin de faciliter la reproduction du scénario pour différentes utilisations, ou pour faciliter la réutilisation de tout ou partie de la situation d'apprentissage modélisée.

L'emploi exagéré de techniques de modélisation comme UML peut constituer un obstacle à la compréhension pour certains intervenants de la phase de conception et aller ainsi à l'encontre de leur implication dans la modélisation [Des04]. Nous pensons alors qu'un langage comme CPM, construit comme une spécialisation d'UML pour le domaine des situations-problèmes, permettra d'augmenter la compréhension des modèles produits. Accompagné d'une méthode, notre langage permettra la collecte d'informations précises, cohérentes et exhaustives sur la situation d'apprentissage qu'il sera possible d'abstraire ou de détailler selon les niveaux de dialogue désirés.

Nous ne réduisons pas la phase de conception à la seule utilisation de la technique liée au langage que nous voulons produire. Il est possible de combiner plusieurs techniques afin d'adresser spécifiquement chaque type de problème, mais aussi pour fournir des représentations compréhensibles par les différentes personnes impliquées dans les phases amont de la conception.

6.2.1.4 Ce que le langage CPM propose

Nous avons vu quels types de modèles de situations-problèmes nous intéressaient et également quels en seraient leurs usages possibles. La difficulté de notre langage est de proposer une syntaxe (terminologie et notation) adaptée pour l'ensemble de ces modèles. En effet, les modèles pour l'étape d'expression initiale des besoins sont très proches du modèle pédagogique théorique des PBL tandis que les modèles de conception plus avancée sont proches des modèles d'unité d'apprentissage et de scénario pédagogique comme ceux de la spécification IMS-LD. La terminologie employée diffère donc selon l'étape associée au modèle ; il en sera de même pour la notation utilisée dans les modèles.

Voici la liste des pré-requis pour le langage CPM :

Reproductibilité : les modèles de situations-problèmes produits doivent pouvoir être décrits à un niveau d'abstraction suffisant de manière à permettre leur répétition pour d'autres configurations avec des personnes différentes.

Complétude : le langage CPM doit être capable de décrire entièrement une situation-problème (modèle de conception avancée proche de ceux de la spécification IMS-LD), ce qui inclut :
- l'intégration des activités des apprenants et des tuteurs,
- l'intégration des ressources utilisées pendant l'apprentissage,
- le support de l'apprentissage coopératif et des activités individuelles comme collaboratives,
- les références aux objets d'apprentissage (numériques ou non).

Personnalisation : le langage doit être « ouvert », c'est-à-dire qu'il doit permettre l'ajout de nouveaux concepts et relations afin de permettre l'élaboration de modèles adaptés aux besoins des différentes équipes de conception l'utilisant. Ceci concerne particulièrement les modèles de l'étape amont (expression initiale des besoins) pour laquelle le modèle théorique de l'apprentissage par situation-problème tient une place importante.

Indépendance du médium et du cadre d'utilisation : le langage doit permettre de décrire le contenu des ressources indépendamment du médium utilisé afin que la PBL modélisée puisse être réutilisée pour des formats différents de publication (exemples : Web, papier, e-books, mobile, etc.) et aussi pour des cadres d'utilisation différents (apprentissage à distance, apprentissage mixte, présentiel).

Indépendance des plates-formes de formation à distance : cette propriété est liée à la précédente.

Réutilisabilité : il doit être possible d'identifier, d'isoler, de dé-contextualiser et de réutiliser tout ou parties des modèles produits pour d'autres contextes.

6.2.1.5 Élaboration du langage

Le langage CPM est conçu comme une extension du langage UML 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 (décrit dans le chapitre 7) qui définit les concepts et leurs relations ;
- une syntaxe concrète : représentée par le profil CPM (chapitre 8) 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).

La construction d'un méta-modèle n'est pas obligatoire pour l'élaboration d'un profil UML ; toutefois, nous avons suivi cette démarche afin de clairement séparer la terminologie de la notation et également pour faciliter la démarche d'élaboration du profil CPM (démarche détaillée précédemment dans la section 5.3.3.3).

6.2.2 Deuxième proposition : le composant CPL

Le langage CPM permet de décrire de nombreux modèles pour les situations-problèmes coopératives en particulier des modèles très en *amont* du processus de conception de la situation-problème visée. Ces

modèles sont proches de ceux obtenus avec d'autres langages de modélisation éducative comme IMS-LD et ne prennent pas en compte l'environnement informatique d'exécution (conformément à nos exigences).

Notre intérêt pour les plates-formes de formation à distance nous amène à proposer un moyen pour réduire l'écart entre les modèles CPM de conception précédents et les fonctionnalités rendues par ces plates-formes. Actuellement, les plate-formes sont prises en compte, dans les modèles, uniquement comme des environnements informatiques manipulant des objets d'apprentissage (*Learning Object* - LO) et proposant des services logiciels standards (comme dans la spécification IMS-LD). Nous illustrons ce constat par la figure 6.2 (cadre supérieur).

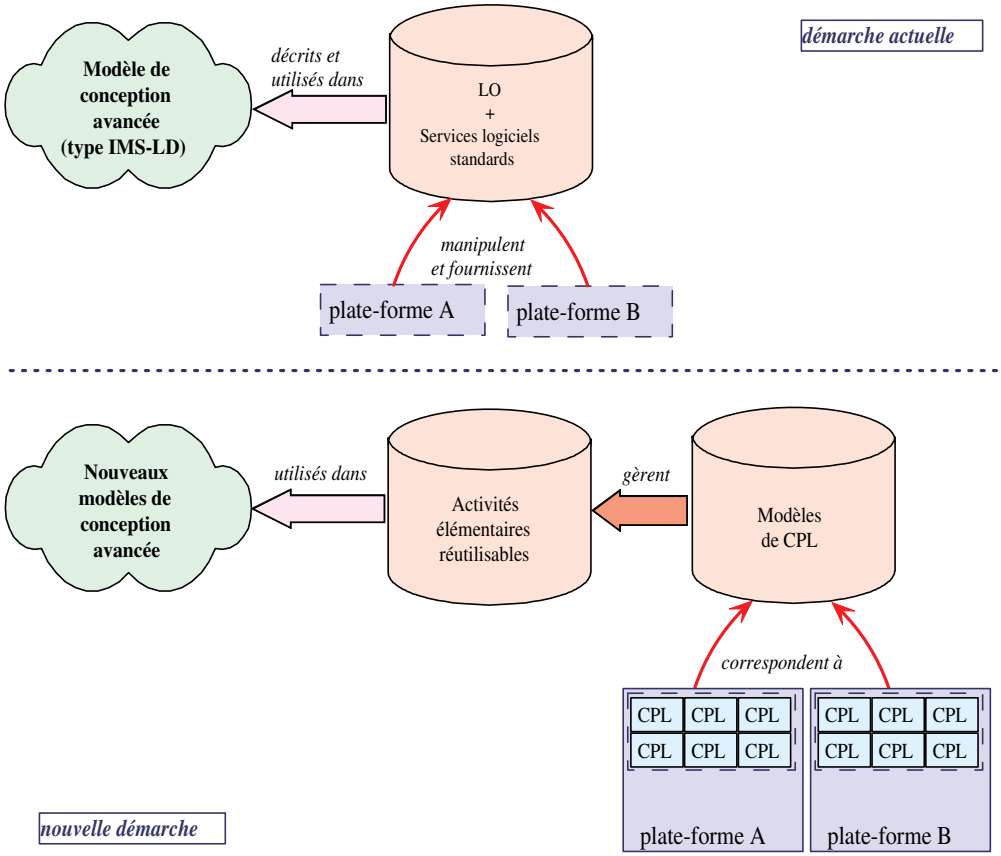

FIG. 6.2 – Vers une meilleure prise en compte des besoins de conception et des fonctionnalités des plates-formes.

Notre proposition est basée sur le constat que les multiples spécifications de modèles de conception très détaillés (avec le langage CPM comme avec d'autres langages) font apparaître des activités élémentaires récurrentes. L'idée de base de nos travaux est alors de « capturer » ces activités dans des composants appelés Composant Pédagogique Logiciel (CPL). Chaque composant CPL a pour rôle de gérer une activité pédagogique élémentaire et de réaliser le pont entre les services pédagogiques attendus pour réaliser cette activité avec les services logiciels traditionnels rendus par les plate-formes. Ainsi, de nouveaux modèles de conception avancée pour les PBL peuvent être spécifiés en réutilisant et contextualisant ces activités élémentaires. Notre proposition concerne alors principalement le modèle spécifique suivi par ces

composants CPL. En effet, quels que soient les différentes implémentations pour un même composant CPL (pour différentes plates-formes cibles), le modèle correspondant sera indépendant d'une plate-forme particulière. L'activité pédagogique correspondante sera à son tour indépendante des plates-formes, ce qui garantit l'interopérabilité des nouveaux modèles. La figure 6.2 (cadre inférieur) illustre nos propos.

Nous expliquons en détail cette seconde proposition ainsi que son intégration dans le langage CPM dans le chapitre 10.

6.3 Organisation de la contribution

Les trois prochains chapitres concernent la première partie de proposition : le chapitre 7 présente le méta-modèle CPM, le chapitre 8 concerne le profil CPM et le chapitre 9 propose un prototype d'environnement auteur pour notre langage ainsi qu'une expérimentation du langage sur le cas d'étude SMASH. Pour terminer, le chapitre 10 est consacré à la seconde partie de la proposition : le modèle de composant CPL.

Chapitre 7

Méta-modèle CPM

Sommaire

Le chapitre précédent a présenté globalement la proposition concernant le langage CPM dédié à la modélisation de PBL coopératives pour supporter la phase de conception. La mise en œuvre de ce langage est basée sur une spécialisation d'UML via le mécanisme d'extension des profils UML.

Ce chapitre expose la syntaxe abstraite du langage CPM sous la forme d'un méta-modèle. L'une des premières étapes pour la construction d'un profil est de définir le domaine d'application. Dans notre cas, **nous avons choisi de décrire l'ensemble des concepts et relations des situations-problèmes par un méta-modèle *indépendant* basé sur une sous-partie du méta-modèle *Core* d'UML**. Ce choix permet de faciliter l'étape de construction du profil qui nécessite l'identification d'un sous-ensemble du méta-modèle d'UML.

Le méta-modèle CPM est décrit dans la section 7.3 de ce chapitre. Dans le but de faciliter sa compréhension, nous présentons tout d'abord, en section 7.1, le modèle conceptuel minimal du langage CPM. De même, la section 7.2 présente et étudie d'autres extraits de méta-modèles partageant des points communs avec notre modèle conceptuel. Ce travail d'analyse a pour objectif de faciliter la construction du méta-modèle CPM.

7.1 Modèle conceptuel

La notion d'activité est au cœur des modèles que nous voulons spécifier avec notre langage : l'équipe de conception prévoit les activités des apprenants et des tuteurs impliqués dans la réalisation de la PBL coopérative. Nous avons vu que les PBL sont issues du courant constructiviste selon lequel l'apprentissage est le résultat de l'activité des apprenants. L'étude des travaux sur les EML a montré également la place importante de la description des activités et de leur organisation pour la mise en œuvre de scénarios d'apprentissage. Ceci nous conduit naturellement à étudier la théorie de l'activité (TA), référencée actuellement dans la plupart des travaux de CSCW ou CSCL (section 7.1.1. Ensuite, nous présentons le modèle conceptuel comme l'abstraction préliminaire à l'élaboration du méta-modèle CPM. Bien que l'analogie avec la TA appliquée aux PBL est implicite dans l'illustration de ce modèle conceptuel, l'ensemble des concepts de la TA sont explicitement définis dans le méta-modèle dérivé du modèle conceptuel.

7.1.1 Théorie de l'activité et PBL

Dans cette section, nous présentons tout d'abord brièvement la théorie de l'activité, puis nous l'appliquons aux situations-problèmes.

7.1.1.1 La théorie de l'activité

La théorie de l'activité correspond à « un corps de concepts dont le but est d'unifier la compréhension de l'activité humaine en fournissant des ponts vers les autres approches provenant des sciences humaines » [Bou00]. Cette théorie provient des différents travaux des psychologues russes Vygotsky, Leont'ev et Luria. Ces travaux ont été ensuite enrichis grâce aux différentes contributions de Engeström [Eng87], Kuutti [Kuu96], ou encore Bardram [Bar97]. Ils envisagent l'activité au travers des concepts de *sujet* et d'*objet* médiatisé par un *outil*. Cette théorie de l'activité, telle que présentée par Nardi [Nar96], est un des courants de large audience parmi les travaux actuels en CSCW et CSCL [Bou00, Geo01, Bet03]. Cette référence à la théorie de l'activité s'explique en raison par les outils conceptuels qu'elle propose pour décrire et comprendre une activité comme un processus social de développement [Bet03].

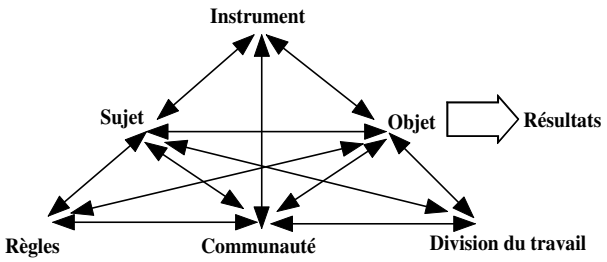

FIG. 7.1 – Structure de l'activité humaine (tirée de [Eng87]).

Engeström a proposé une structure de l'activité humaine (figure 7.1) [Eng87] : la relation entre le **sujet** (l'individu ou le groupe au centre de l'analyse) et l'**objet** (ce vers quoi l'activité est dirigée) est

médiatisée par des **instruments** (outils matériels ou outils *pour penser*[101] mis à disposition du sujet) ; la relation entre le sujet et la **communauté** (individus ou groupes partageant le même objet général) est médiatisée par des **règles** (explicites ou implicites) ; la relation entre la communauté et l'objet est médiatisée par la **division du travail** (division horizontale des tâches entre membres de la communauté et division verticale de statuts et de droits).

Leont'ev a également proposé une structure de l'activité à trois niveaux : activité, action et opération (tableau 7.1). L'activité est dirigée par un objet et est effectuée par la communauté. Au niveau inférieur, l'activité se décompose en actions, chaque action étant dirigée vers un but dont la réalisation dépend des individus ou des groupes d'individus. Enfin au niveau le plus bas, chaque action propose un ensemble d'opérations dirigés par des conditions ; la réalisation de ces opérations peut être effectuée par des personnes ou par la machine.

Niveau	Dirigé par	Effectué par
Activité	objet	individu/communauté
Action	but	individus ou groupe d'individus
Opération	conditions	humains ou machines

TAB. 7.1 – La structure hiérarchique de l'activité (repris et modifié de [Mia00])

7.1.1.2 Théorie de l'activité appliquée aux PBL

Y. Miao a analysé les PBL du point de vue de la théorie de l'activité [Mia00]. Le **sujet** d'une activité dans une PBL est le groupe d'apprenants impliqués dans la PBL. L'**objet** d'une activité de PBL correspond au problème qui leur a été présenté. Les **résultats** attendus d'une activité de PBL sont d'une part, les connaissances et compétences à acquérir qui pourront ensuite être décontextualisées pour résoudre des problèmes similaires à un niveau individuel, et d'autre part, les connaissances partagées et la compréhension mutuelle au niveau du groupe. Les **instruments** correspondent dans ce cas :

- aux outils : tableau noir ou encore outils spécifiques à un domaine (pour une PBL en classe) ; outils proposés par les plates-formes (pour une PBL médiatisée par ordinateur) ;
- aux lieux : salles de discussion, ou encore bibliothèque (pour une PBL non médiatisée) ; salons de discussion, forums, etc. (liés aux outils pour une PBL médiatisée) ;
- aux documents : matériaux d'apprentissage ou objets d'apprentissage.

La **communauté** d'une PBL rassemble toutes les personnes impliquées dans la réalisation des activités (apprenants, tuteurs, experts, etc.). Dans l'apprentissage par PBL coopératives, la **division du travail** permet d'assigner des tâches différentes aux apprenants pour un même but partagé. Les responsabilités des apprenants sont également définies. Les **règles** correspondent, quant à elles, aux moyens utilisés pour réguler les comportements dans les interactions entre apprenants (droits de collaboration), entre apprenants et tuteurs, mais également pour utiliser les instruments (droits d'utilisation) et pour mesurer les résultats (critères de réussite).

[101] Le langage est l'un de ces outils.

La structure hiérarchique de l'activité humaine est présente dans la mise en place de PBL coopératives. Les activités réalisées par les apprenants et tuteurs (la communauté) sont dirigées vers les objectifs d'apprentissage de la PBL. Leur organisation correspond à la spécification des actes ou encore des scènes employés dans certains langages (EML). Les activités d'une PBL peuvent être décomposées selon des sous-buts. Ceux-ci sont alors atteints en réalisant des actions internes à l'activité (les EML ne permettent pas de spécifier les actions). Ces actions font évoluer des objectifs d'apprentissage intermédiaires (par exemple : le problème est défini, les informations nécessaires ont été collectées, etc.) qui font partie de conditions pour de futurs actions. Ces conditions permettent alors aux acteurs d'exécuter les d'opérations par le biais d'outils.

7.1.2 Notre modèle conceptuel

Notre modèle conceptuel est une abstraction du cœur du méta-modèle CPM. Nous le représentons sous la forme d'un diagramme de classes (Figure 7.2). Ce modèle conceptuel résume l'idée selon laquelle une personne joue différents rôles en réalisant différentes activités pour résoudre le problème donné dans une PBL coopérative.

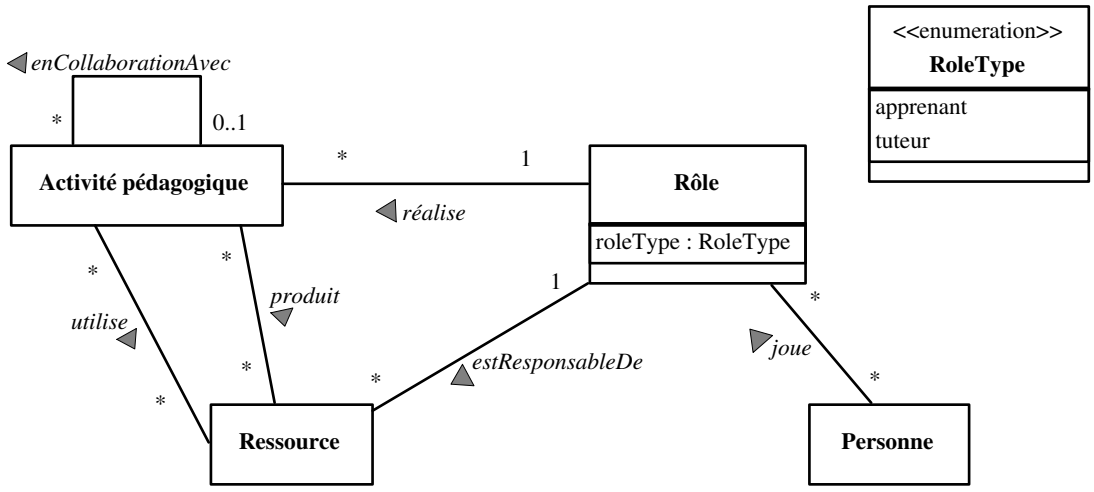

Fig. 7.2 – Le modèle conceptuel du langage CPM.

La « personne » représente l'acteur réel impliqué dans la PBL. Il est amené à jouer un ou plusieurs « rôles » car le concept de rôle est ici général et peut correspondre à des rôles situés comme authentiques (voir les différents concepts de rôles dans les PBL 2.1.4.3). Par contre, le concept de rôle global (apprenant ou tuteur) fait l'objet d'une précision sur le concept de « rôle » par le biais de l'attribut « roleType ». A chacun de ces rôles correspondent des « activités pédagogiques » à réaliser et des « ressources » qui sont attribuées. L'association « enCollaborationAvec » sous-entend que les activités pédagogiques seront parfois individuelles parfois en relation avec d'autres (activités collaboratives).

Les ressources représentent, au sens large, tout ce qui peut être utilisé, créé, modifié dans une activité. Une ressource peut donc être :

- un objet d'apprentissage ou un service (au sens de la spécification IMS-LD [IMS03c]) ;
- un outil, un document, etc. ;
- un savoir, un savoir-faire : dans ce cas, le modèle conceptuel permet d'exprimer l'idée qu'au travers des rôles qu'il joue, un apprenant réalise des activités pédagogiques pour lesquelles il va mobiliser et utiliser des connaissances et compétences qu'il possède déjà pour en acquérir des nouvelles ;
- une représentation mentale d'un apprenant, c'est-à-dire la conception qu'il a, à un moment donné, d'un objet ou d'un phénomène lié à la situation d'apprentissage ;
- un événement pédagogique : les actions d'un apprenant, pendant la réalisation d'une activité, peuvent produire des événements pédagogiques destinés à être captés par les tuteurs afin qu'ils puissent *réagir* (évaluer, corriger, intervenir, assister, etc.) si besoin : boucle action/rétroaction de l'apprentissage [Rod00].

Les différents concepts de la théorie de l'activité n'apparaissent pas explicitement dans ce modèle conceptuel (mis à part *ressource* et *rôle* faisant référence à *instrument* et *sujet* du modèle d'Engeström). Le méta-modèle CPM que nous allons présenter en détail dans la prochaine section présente une analogie explicite des différents termes du modèle de l'activité. Nous y reviendrons en conclusion de ce chapitre.

Notre modèle conceptuel exprime implicitement les particularités suivantes que nous retrouverons dans le méta-modèle CPM :

1. Une activité n'est réalisée que par un seul rôle. Ce choix privilégie la facilité de spécification de fiche de tâche individuelle. L'inconvénient réside dans la représentation des activités collaboratives qui nécessite alors de décomposer l'activité collaborative en d'autant d'activités qu'il y a de rôles impliqués dans la collaboration et de relier ces activités entre elles afin de spécifier leur lien de collaboration et aussi de les distinguer des activités individuelles.
2. Le rôle représente aussi bien le rôle global joué dans la PBL qu'un rôle plus particulier joué dans une partie du scénario. Les rôles peuvent alors être spécialisés (relation d'héritage) et composés (relation d'agrégation) d'autres rôles.
3. L'apprentissage coopératif des apprenants dans la résolution de la PBL s'effectue par le biais d'activités individuelles et collaboratives mais également par l'échange de ressources de production nécessaires à la PBL (associations « utilise » et « produit ») : une activité peut utiliser, voire modifier une ressource créée par une autre activité et donc un autre rôle.
4. La « responsabilité » des ressources répond au besoin du concepteur de PBL d'associer des ressources particulières à certains rôles.

7.2 Construction du méta-modèle

Dans cette section, nous présentons des modèles[102] issus de travaux variés de recherche mais ayant le point commun de décrire des structures d'organisation pour l'activité. Nous analysons les similitudes de

[102]Ici, *modèle* est employé au sens large ; que les diagrammes présentés soient en fait des méta-modèles, tels que défini par l'OMG, ou bien des modèles importe peu ; notre intérêt porte en fait sur les concepts et relations qu'ils décrivent.

ces modèles et dégageons les conceptualisations communes que notre méta-modèle CPM devra prendre en compte.

1. Le modèle conceptuel du support à l'activité de DARE [Bou00] (Figure 7.3) permet à son auteur d'identifier les concepts et mécanismes sous-jacents permettant de décrire un environnement support malléable pour l'activité coopérative.

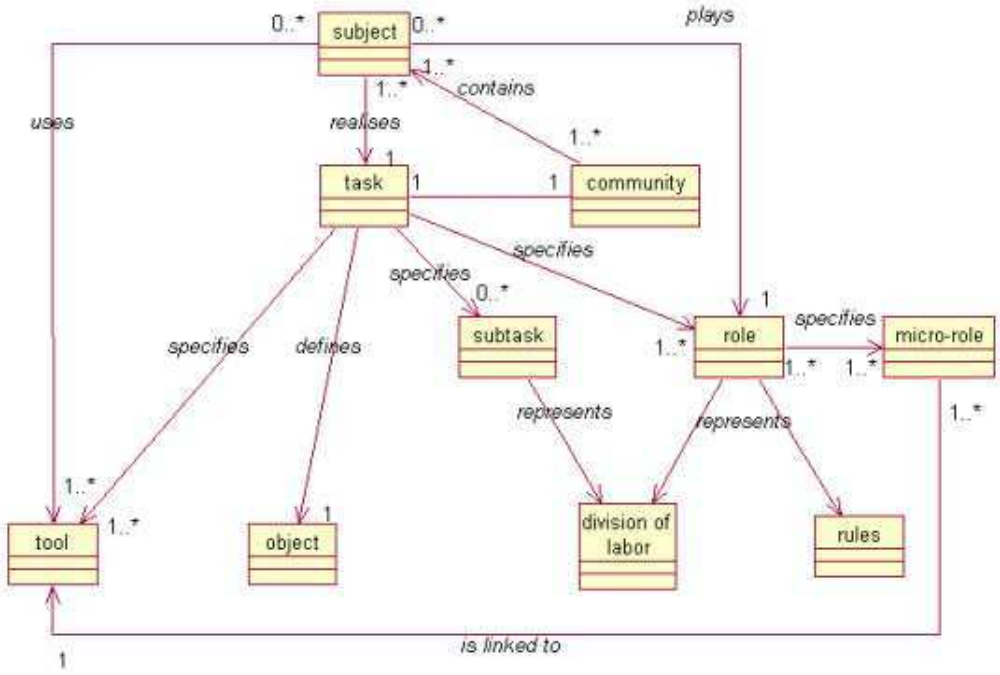

FIG. 7.3 – Les concepts de la structure de base d'une activité et ceux du modèle conceptuel de DARE [Bou00] (figure re-dessinée).

Le modèle conceptuel d'activité de SimuLigne (Figure 7.4) est également adapté de la structure de base de l'activité proposée par Engeström.

Les modèles conceptuels de DARE et SimuLigne sont tous deux dérivés du modèle de l'activité (on retrouve les concepts de *sujet*, *division du travail*, *objet*, *outil* et *règles*). Entre les deux modèles se retrouvent de nombreux concepts similaires : *communauté* dans DARE est appelé *groupe* dans SimuLigne ; *tâche* pour *activité* ; *sous-tâche* pour *tâche* ; *rôle* pour *statut*. Le découpage de la tâche en sous-tâche est similaire dans les deux modèles, SimuLigne ajoutant seulement la notion d'*étape*. Alors que le modèle de SimuLigne s'intéresse davantage à décrire le *temps* d'une activité, celui de Dare propose le concept de *micro-rôle* pour définir les droits et les devoirs envers un outil. Pour SimuLigne, l'outil est spécifié directement par l'activité et lié à une tâche particulière. Rien n'indique alors si la tâche sera réalisée par un seul sujet ou pas ; aucune restriction sur l'usage des outils n'est précisée. Pour DARE, l'outil est spécifié dans le cadre d'une tâche pouvant impliquer plusieurs sujets utilisant l'outil mais dont les droits d'usage sont définis grâce aux micro-rôles. Par contre, le découpage de tâches en sous-tâches n'implique aucun changement vis-à-vis des outils. Les relations entre sous-tâche et rôle (pour DARE) et entre tâche et statut (pour SimuLigne) ne sont

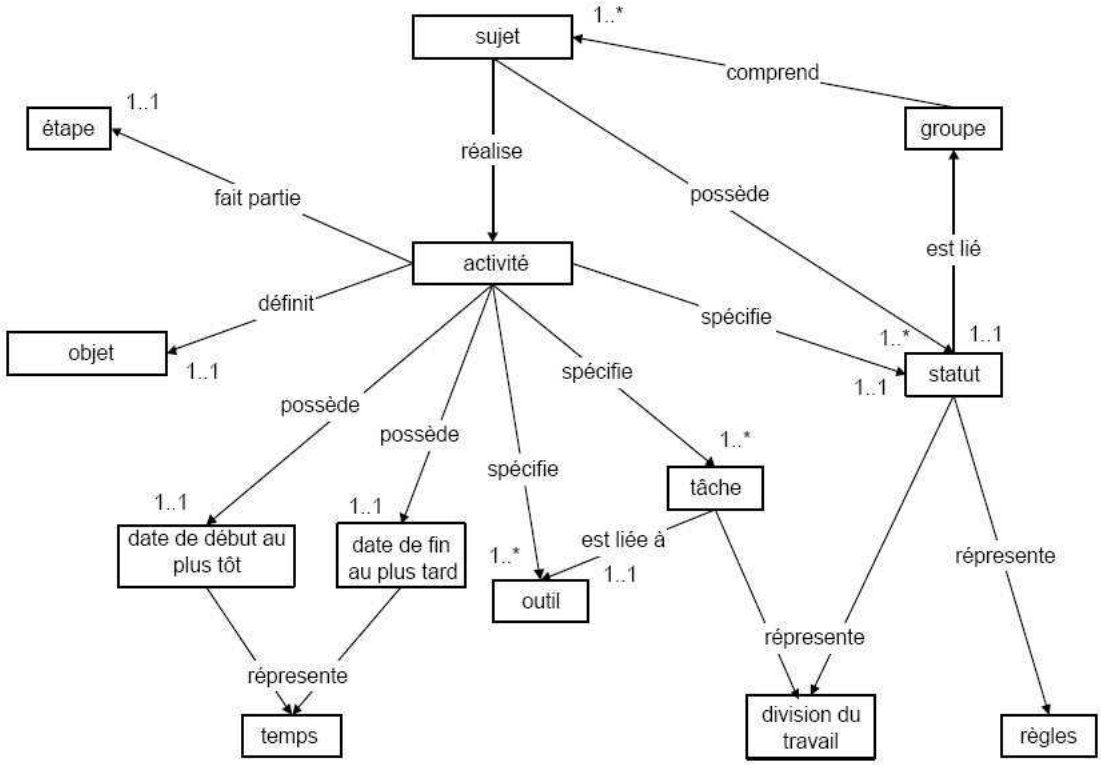

FIG. 7.4 – Modèle conceptuel d'activité de SimuLigne ([Mba03]).

pas explicitement définies dans les modèles alors que ces termes participent à définir la division du travail.

2. Le modèle coopératif proposé dans [Gua00a, Gua00b] (Figure 7.5) capture le vocabulaire nécessaire à la représentation de nombreuses applications *groupware*.

   Le modèle présenté pour le *groupware* ne fait pas référence à la théorie de l'activité explicitement. Comme les autres, le concept central est l'*activité*, structurée hiérarchiquement par le biais d'*actions* et *sous-activités*. Une activité peut être réalisée par plusieurs *acteurs* comme pour les modèles précédents (et contrairement à notre modèle conceptuel). En conséquence, les rôles et règles entre ces acteurs sont définis par la classe d'association *coordination*. Les acteurs ont également des *canaux de communication* (audio/vidéo, emails, etc.) à leur disposition pour que l'un d'entre eux puisse émettre vers d'autres. Les restrictions d'usage de ces canaux ne sont pas spécifiées explicitement. Une relation entre plusieurs activités est possible (séquentialité, synchronisation, etc.) et est alors typée par le biais de *Relationship*. Les outils n'apparaissent pas mais la notion de *service* est proposée. Ils sont associés directement aux activités. Les restrictions et droits d'usage sur ces services ne sont pas non plus définis explicitement. Les concepts de service et canal de communication se recoupent et peuvent porter à confusion. Le concept d'information représente tout ce qui est utilisé ou produit par les activités (messages, documents, etc.). Afin de permettre le partage d'informations ou de restrictions d'accès par exemple, le type de l'information peut être décrit par *Concurrency*. Ce modèle apporte davantage de conceptualisation pour le travail coopératif (et l'apprentissage coopératif par analogie) que les deux précédents modèles. Seule la notion d'objet n'est

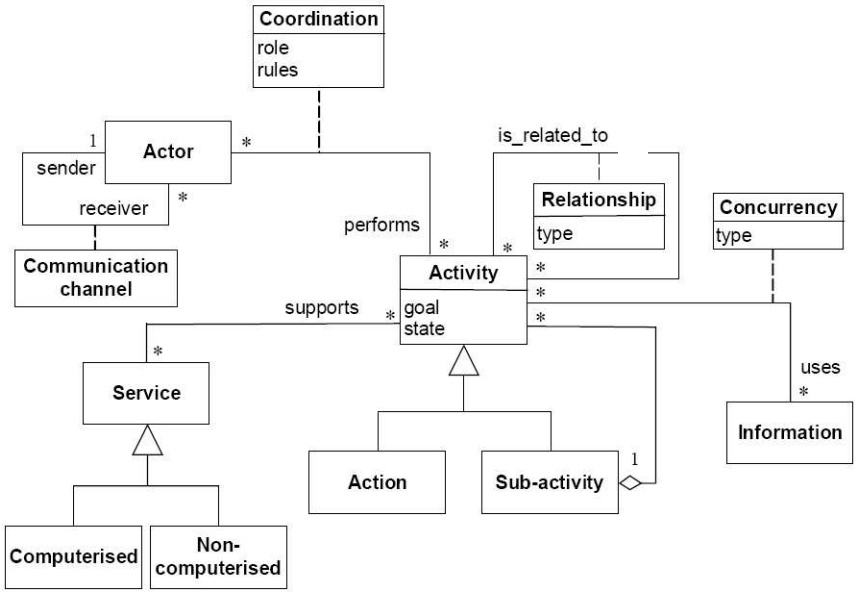

FIG. 7.5 – Modèle coopératif pour le développement de collecticiels basés composants [Gua00a].

pas exprimée. Les concepts et relations de notre modèle conceptuel se retrouvent indirectement au travers de ce modèle.

3. Le paquetage *ProcessStructure* du *Software Process Engineering Metamodel* (SPEM)

   Cet extrait de méta-modèle standardisé par l'OMG nous intéresse pour deux raisons. Premièrement, le méta-modèle CPM est construit de manière analogue au méta-modèle SPEM (voir la section suivante 7.3) ; deuxièmement, cet extrait a de nombreux points communs avec notre modèle conceptuel. En effet, un seul *ProcessPerformer* réalise une activité ou *WorkDefinition*. Chaque *ProcessPerformer* a des *Workproduct* dont il est responsable. Les *Activityparameter* permettent alors de relier les produits avec les activités tout en précisant si les produits sont créés, modifiés ou utilisés. Comme pour les autres modèles présentés précédemment, les activités peuvent être structurées hiérarchiquement (association réflexive sur *WorkDefinition*). Par contre, aucun concept ne permet de décrire des activités collaboratives et de définir des règles quant à la coordination.

   Nous avons remarqué que les travaux sur les processus ([OMG01b, Bre02]) ou les workflows en général (*cf.* travaux de Vantroys [Van03]) partagent de nombreux points communs avec une vision de conception avancée selon laquelle la définition d'un scénario pédagogique s'apparente à celle d'un *Learning Flow*.

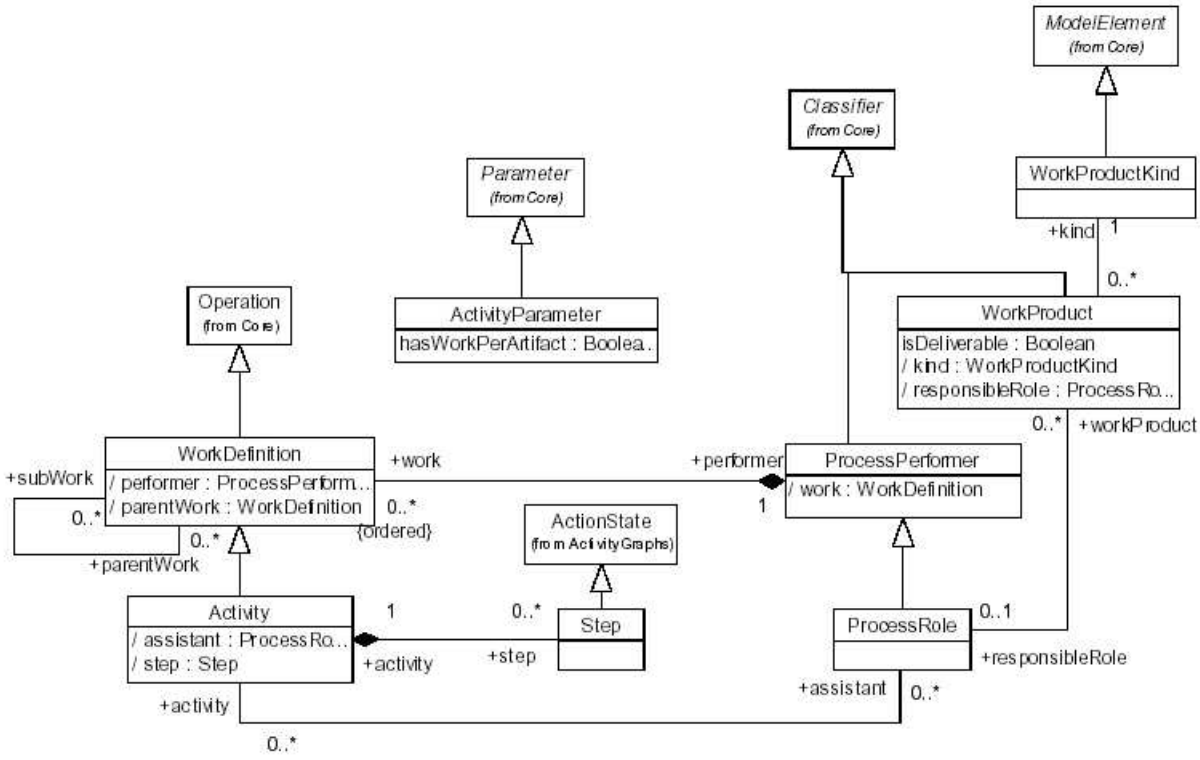

FIG. 7.6 – Paquetage présentant la structure des processus d'Ingénierie Logicielle (SPEM).

4. Le modèle conceptuel de la spécification IMS-LD

   Ce modèle a été présenté et étudié en détail en section 4.3.3.1 (voir la figure 4.11). De manière similaire aux autres modèles que nous venons d'étudier, il propose une organisation (*method*, *play*, *act*) et une structuration (*activity*, *activity-structure*) de l'activité. Les activités peuvent être réalisées par plusieurs rôles. Ceci permet de définir des activités individuelles réalisées de manière identique par différents rôles ou bien de définir des activités collaboratives. Dans ce dernier cas, *environment* permet alors de préciser les différentes modalités de collaboration en fournissant des *services* adaptés.

Nous retenons de l'analyse de ces différents modèles trois caractéristiques essentielles pour notre méta-modèle CPM :

1. proposer des éléments pour la *structuration* hiérarchique des activités d'une PBL coopérative ;
2. proposer des éléments pour décrire l'*objet* de la PBL, c'est-à-dire les objectifs pédagogiques mais également tous les autres aspects *pédagogiques* des PBL (obstacles, ressources, contraintes, prérequis, connaissances, etc.) ;
3. proposer des éléments permettant de décrire les aspects *sociaux* des PBL (rôles, activités collaboratives, restrictions, droits, etc.).

7.3 Méta-modèle CPM

Dans la section précédente nous avons abouti à un modèle conceptuel de base illustrant les concepts et les relations au cœur de notre langage. Ce modèle conceptuel est un socle de construction pour notre méta-modèle CPM[103] présenté dans cette section.

Le méta-modèle CPM est construit sous la forme de 2 paquetages : `CPM_Foundation` et `CPM_Extension` (Figure 7.7). Concrètement, les concepts et les relations pour le langage CPM sont définis dans le second paquetage `CPM_Extension`. En effet, le méta-modèle CPM est construit par extension d'un sous-ensemble du méta-modèle physique de UML 1.4. **Ce sous-ensemble d'UML correspond à notre paquetage** `CPM_Foundation` de la Figure 7.7. L'annexe A décrit en détail le contenu de ce paquetage. Nous avons réalisé ce découplage pour les raisons suivantes :

1. l'alignement d'UML sur le MOF n'a été réalisé que récemment par l'OMG avec la spécification d'UML 2.0 alors que nos travaux de recherche avaient déjà commencés sur la base des spécifications UML 1.X ;
2. de nombreux travaux analogues de spécification de méta-modèles puis de profils suivent la même démarche [OMG02, OMG04] ;
3. cette démarche permet d'identifier les éléments du méta-modèle d'UML qui serviront de base à la définition des stéréotypes dans le profil CPM ; ainsi, la spécification du profil sera facilitée ;
4. certains éléments de méta-modélisation nécessaires pour le point précédent n'existent pas dans le MOF (exemple des éléments pour les *ActivityGraphs*).

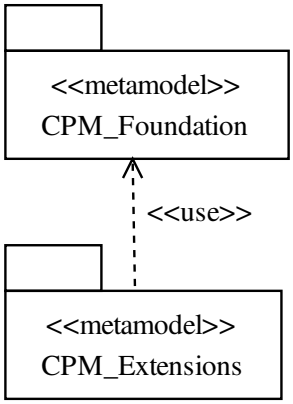

FIG. 7.7 – Les deux paquetages formant le méta-modèle CPM : le paquetage `CPM_Foundation` et le paquetage `CPM_Extensions`.

Ainsi, notre méta-modèle CPM peut être considéré comme *stand-alone*, c'est-à-dire qu'il est réflexif (comme UML) et qu'il n'est pas une instance du MOF. Nous aurions pu construire le méta-modèle CPM

[103] Le méta-modèle CPM est défini entièrement en anglais par souci de communication au niveau international.

comme une instance du MOF mais, dans ce cas, les points 3 et 4 précédents n'auraient pas été respectés et un travail supplémentaire au niveau de l'élaboration du profil aurait été nécessaire. De plus, l'alignement de CPM sur le MOF n'aurait été intéressant que si nous avions privilégié dès le départ une orientation de méta-modélisation pour notre langage indépendamment d'UML.

7.3.1 Paquetage `CPM_Extensions`

Le paquetage `CPM_Extensions` se décompose en plusieurs paquetages regroupant les concepts et leurs relations par centre d'intérêt (Figure 7.8) :

- le paquetage `CPM_BasicElements` décrit les éléments de base de notre langage ;
- le paquetage `CPM_PedagogicalPackage` concerne les concepts pédagogiques des situations-problèmes : objectif, critère de succès, représentation mentale, etc. ;
- le paquetage `CPM_StructuralPackage` décrit les différents concepts pour la structuration d'un scénario ;
- le paquetage `CPM_SocialPackage` décrit les relations sociales, la mise en œuvre de l'apprentissage coopératif, les activités collaboratives, etc.

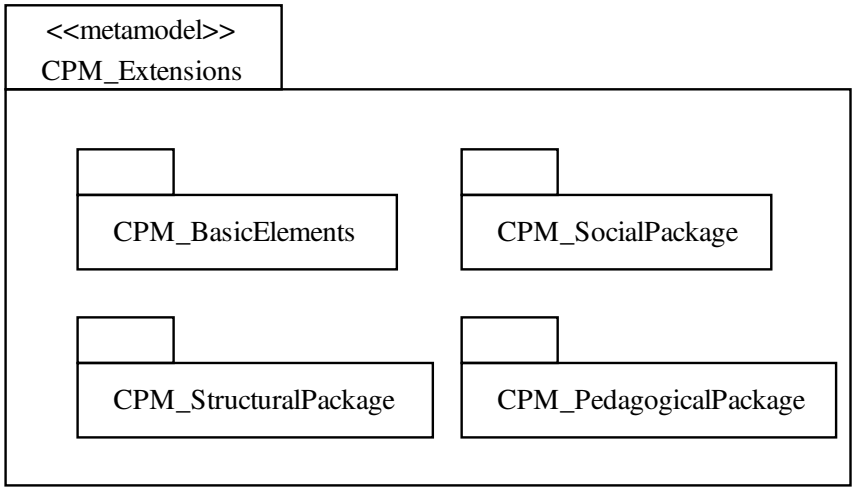

FIG. 7.8 – Décomposition en sous-paquetages du paquetage `CPM_Extensions`.

Tous les éléments du méta-modèle CPM sont dérivés directement ou indirectement de **ModelElement**[104]. La figure 7.9 illustre cette propriété en montrant l'ensemble des éléments du paquetage `CPM_Extensions` (boîtes claires) ainsi que les éléments du paquetage `CPM_Foundation` dont ils héritent (boîtes foncées). Certains spécialisent directement **ModelElement** tandis que d'autres le spécialisent indirectement à travers d'autres concepts du paquetage `CPM_Foundation`.

Les différents paquetages d'extension sont présentés dans les sous sections suivantes. Ensuite, pour chaque paquetage, nous définissons tous les concepts contenus : définition, sémantique du concept, liens

[104] **ModelElement** appartient également au méta-modèle d'UML.

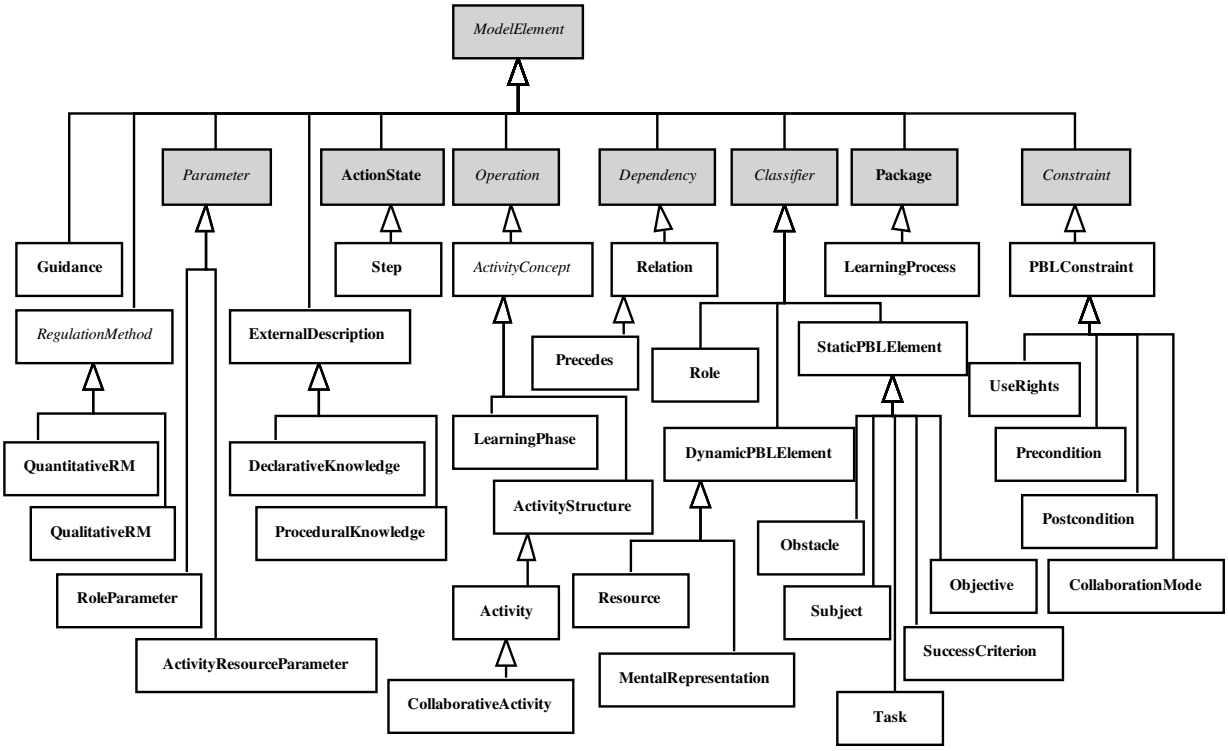

FIG. 7.9 – Diagramme de généralisation de tous les éléments composants le paquetage `CPM_Extensions`.

entre ce concept et d'autres concepts du paquetage d'extension s'ils existent, lien avec les éléments de modélisation du paquetage `CPM_Foundation`, précisions sur les contraintes liées à la sémantique si nécessaire, et exemple d'utilisation de ce concept.

Certains exemples font l'objet d'une illustration (exemple : figure 7.11). De manière à décrire visuellement au mieux les instances de nos méta-classes, nous avons choisi d'utiliser uniquement les diagrammes de classes. **Ces illustrations sont indépendantes de celles permises par le profil CPM** présenté dans le chapitre suivant. Toutefois, afin de décrire une instance de méta-classe, nous utilisons la notation stéréotypée des classes : le nom du stéréotype est celui de la méta-classe instanciée ; pour décrire l'instanciation des méta-attributs nous utilisons la notation des valeurs marquées : une valeur marquée représente un méta-attribut.

7.3.2 Paquetage des éléments de base : `CPM_BasicElements`

Ce paquetage définit des éléments de base utilisés dans notre langage (Figure 7.10). Ces éléments ne sont pas des concepts intrinsèques des situations-problèmes coopératives mais des éléments dont les relations sont valides quelque soit l'élément de modélisation de notre langage (en relation avec **ModelElement**).

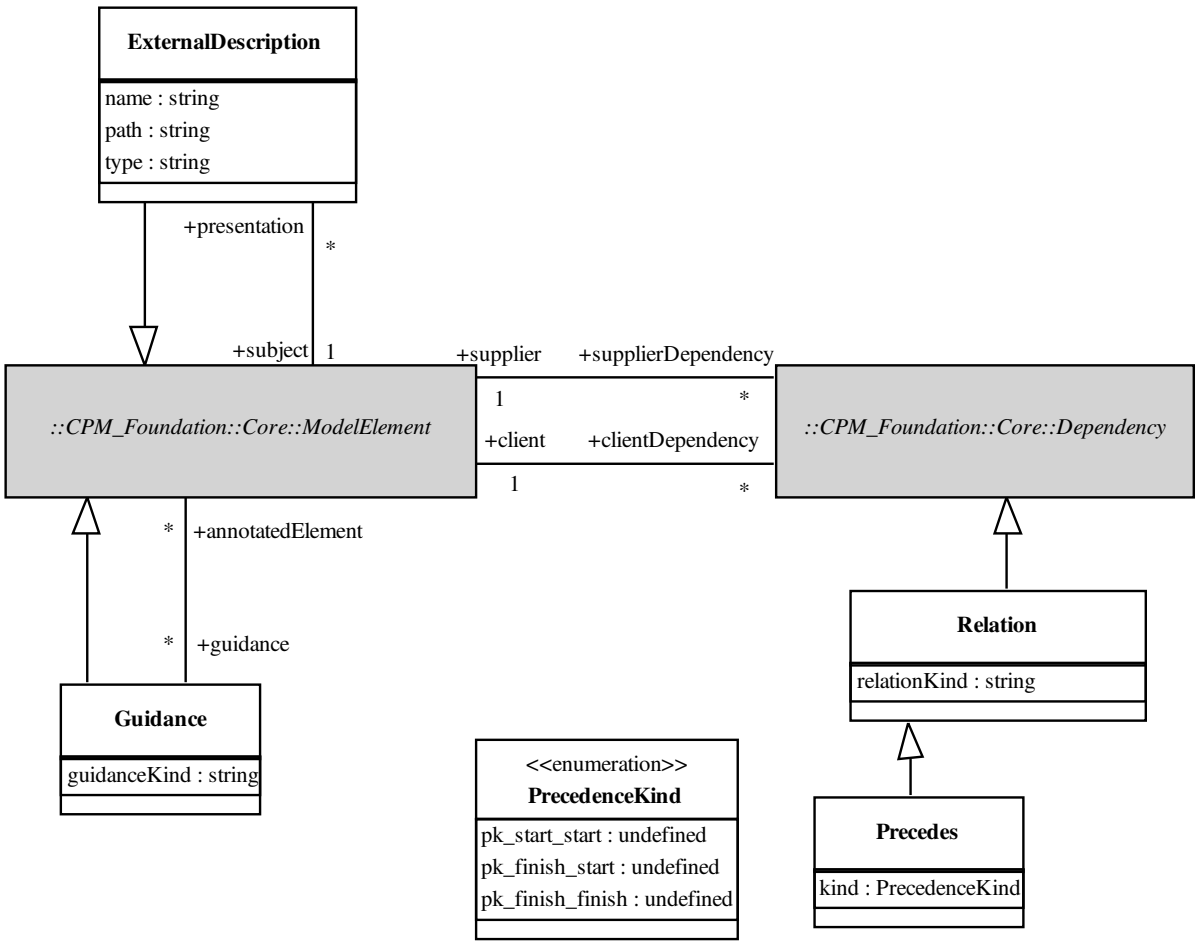

FIG. 7.10 – Le paquetage des éléments de base : `CPM_BasicElements`.

7.3.2.1 ExternalDescription

Définition

Une *description externe* permet de lier tout élément de modélisation manipulé dans les futurs modèles produits à des éléments physiques leur correspondant Il s'agit d'une correspondance entre l'élément modélisé et l'objet d'apprentissage concret qu'il modélise. Cette notion correspond au concept de *binding* des travaux de IMS-LD [IMS03c] et garantit la séparation concept/objet qui permet la réutilisabilité des modèles.

Les attributs de cette méta-classe sont :

- **name** : le nom de l'élément physique,
- **path** : le chemin d'accès à cet élément (optionnel),
- **type** : le type MIME de l'élément (URL, fichier doc, etc.).

Relations avec les autres concepts

Tout élément de notre langage peut avoir plusieurs *descriptions externes*. Par contre, une *description externe* ne correspond qu'à un seul élément de modélisation.

Lien avec le paquetage `CPM_Foundation`

ExternalDescription hérite directement de **ModelElement**.

Exemple

La figure suivante (Figure 7.11) illustre un exemple d'utilisation d'**ExternalDescription** : la *ressource* témoignage numéro 1 est associée à une *description externe* décrivant le nom, chemin et type de l'*objet d'apprentissage* correspondant.

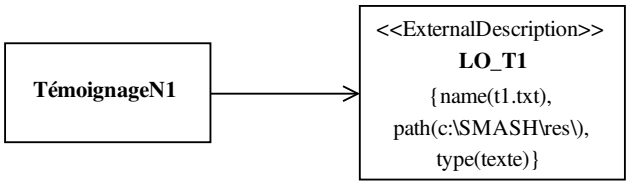

FIG. 7.11 – Exemple d'utilisation des *descriptions externes*.

7.3.2.2 Guidance

Définition

Guidance représente une aide, un conseil qui peut être associé à tout élément de modélisation. Cette aide permet de fournir plus de détails et d'informations sur l'élément associé.

Différents types de **Guidance** sont possibles à travers l'instanciation du méta-attribut **guidanceKind** : exemples, *guidelines*, techniques, *checklists*, modèles (*templates*)...

Relations avec les autres concepts

Tout élément de notre langage peut avoir plusieurs *aides* et une *aide* peut correspondre à plusieurs éléments de modélisation.

Un **Guidance** peut avoir une association vers un **ExternalDescription** décrivant l'élément physique apportant l'aide.

Lien avec le paquetage `CPM_Foundation`

Guidance hérite directement de **ModelElement**.

Exemple

Dans la figure suivante (Figure 7.12) une aide de type *guidelines* est définie pour deux *ressources* témoignages.

7.3.2.3 Relation

Définition

Relation permet d'indiquer qu'un ou plusieurs éléments sont en relation, utilisent, font abstraction,

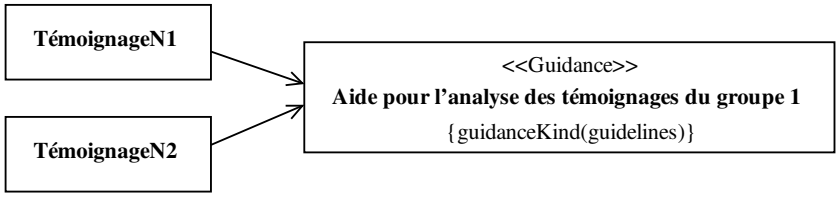

FIG. 7.12 – Exemple d'utilisation des *guides*.

ou requièrent la présence d'un ou plusieurs autres éléments.

Le méta-attribut **relationKind** permet d'indiquer le type de la relation.

Relations avec les autres concepts

Une **Relation** s'établit entre deux éléments de modélisation qui peuvent être des instances d'un même élément CPM.

Lien avec le paquetage `CPM_Foundation`

Relation hérite de **Dependency** car il suit la même définition [OMG03d] :
« *A dependency states that the implementation or functioning of one or more elements requires the presence of one or more other elements. In the metamodel, a Dependency is a directed relationship from a client (or clients) to a supplier (or suppliers) stating that the client is dependent on the supplier (i.e., the client element requires the presence and knowledge of the supplier element).[. . .]* »

Nous avons choisi de ne pas ajouter notre méta-attribut **relationKind** à la méta-classe **Dependency** déjà existante mais plutôt de le rattacher à une nouvelle méta-classe. Ce choix permet de séparer les fondations des extensions et ainsi d'améliorer la compréhension et l'utilisation de notre méta-modèle.

Bien que **Dependency** hérite de **ModelElement**, il ne peut pas utiliser le méta-attribut **name**[105]. Ceci explique pourquoi nous utilisons notre propre méta-attribut pour nommer le type de la relation.

Exemples

De nombreux types de dépendance peuvent être utilisés pour décrire les situations-problèmes.

Notre méta-modèle propose une *relation* nommée **Precedes** (définie également dans le méta-modèle de SPEM [OMG02]). Cette relation relie une **Activity** à une autre (ce concept est défini dans la section 7.3.4) pour indiquer une dépendance de type *start_ start*, *finish_ start* ou *finish_ finish* selon la valeur du méta-attribut **kind** :

- Si l'activité B a une dépendance de type *finish_ start* avec l'activité A, alors B peut commencer que lorsque A a terminé (séquence stricte, pas de parallélisme).
- Si l'activité B a une dépendance de type *finish_ finish* avec l'activité A, alors B peut se terminer uniquement si A a est également terminée (parallélisme possible avec synchronisation à la fin).

[105] En effet, dans la spécification d'UML 1.X l'usage de *name* pour les dépendances est réservée (voir [OMG03d] section 2.5.2.1).

– Si l'activité B a une dépendance de type *start_ start* avec l'activité A, alors B peut commencer seulement après que A ait commencé (parallélisme possible avec synchronisation au début).

Dans ce paquetage nous définissons aussi la relation de dépendance nommée **Trace**. Elle a la même sémantique que le stéréotype de Dependency *Trace* du méta-modèle d'UML : *Trace* permet de relier deux éléments de modélisation représentant le même concept dans des modèles différents ([OMG03d] section 2.5.2.1).

D'autres *relations* particulières peuvent être définies par l'usager (l'ingénieur pédagogique) de notre méta-modèle en instanciant directement *Relation* et en stipulant le type de la relation *via* le méta-attribut **relationKind**. Par exemple, si **Precedes** n'avait pas été défini dans notre méta-modèle, il aurait fallu le créer au niveau des modèles comme illustré dans la partie haute de la figure 7.13. Par contre, il n'y a pas possibilité d'ajouter une information supplémentaire comme avec le méta-attribut **relationKind** (partie basse de la figure 7.13).

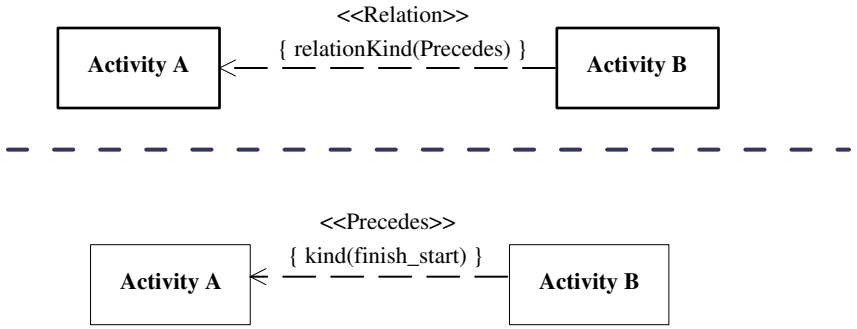

FIG. 7.13 – Exemples a/ de définition d'une **Relation** au niveau des modèles M1 et b/ de l'utilisation de la relation prédéfinie **Precedes**.

Contraintes [106]

La *relation* **Precedes** suit la contrainte précisant que le *supplier* et le *client* (*associationEnd* hérités du `CPM_Foundation`, voir A.1.1.3) sont tous deux de type **Activity**.

```
context Precedes inv:
self.supplier.oclIsKindOf(Activity) and
self.client.oclIsKindOf(Activity)
```

7.3.3 Paquetage pédagogique : `CPM_PedagogicalPackage`

Ce paquetage concerne les différents concepts et relations centrés sur l'aspect pédagogique des situations-problèmes que l'on désire modéliser. Il contient des concepts de deux types :

[106] Les contraintes sont exprimées en OCL (cf la spécification d'OCL dans [OMG03d] section 6).

– des concepts liés à l'expression initiale des besoins des situations-problèmes ; ils permettent la spécification de *modèles d'expression initiale des besoins* et des *modèles d'analyse* ;
– des concepts liés à la conception des situations-problèmes ; les *modèles de conception* produits suivent des critères semblables aux modèles de scénario de situations d'apprentissage que nous avons étudiés dans la section 4.3 sur les EML.

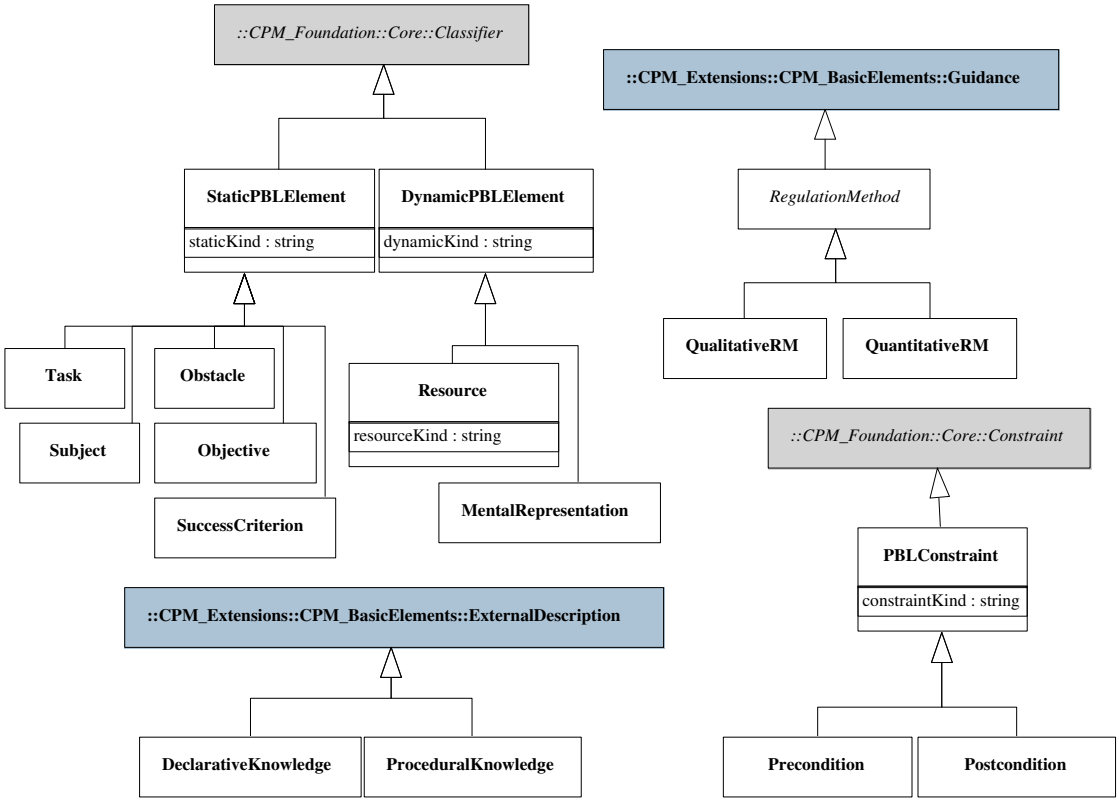

FIG. 7.14 – Le paquetage pédagogique détaillé : `CPM_PedagogicalPackage`.

7.3.3.1 StaticPBLElement

Définition

StaticPBLElement représente une abstraction de tous les concepts d'une situation-problème utilisables pour l'*analyse*. Ces concepts sont considérés ici comme *statiques* en opposition à leurs pendants de type **DynamicPBLElement**.

Le méta-attribut **staticKind** permet d'indiquer le type de l'élément de la situation-problème considérée.

Lien avec le paquetage `CPM_Foundation`

StaticPBLElement est une spécialisation de **Classifier** et peut alors participer dans des associations avec d'autres éléments d'analyse statiques comme dynamiques ; il peut aussi contenir des dé-

finitions imbriquées, c'est-à-dire d'autres **Classifier** peuvent être définis dans l'espace de nommage d'un **Classifier** (voir [OMG03d] section 2.28 ou l'annexe A sur le paquetage `CPM_Foundation`).

Exemples

Notre langage CPM est conçu pour être flexible quant à la définition de ces éléments pour les PBL (propriété de **personnalisation** présentée en section 6.2.1.4). Cette flexibilité permet dans n'importe quel modèle de PBL de définir de nouveaux éléments statiques de PBL en instanciant directement la méta-classe **StaticPBLElement** et en précisant le nom du nouvel élément *via* le méta-attribut **staticKind**.

Toutefois, nous proposons une liste d'éléments statiques pour l'analyse de PBL. Cette liste n'est qu'un répertoire de base ; les concepteurs modélisant des situations-problèmes avec le langage CPM peuvent ajouter de nouveaux éléments si nécessaire.

Task : représente le concept de *tâche* tel qu'il est souvent employé en *expression initiale des besoins* pour désigner l'ensemble des activités que doit réaliser un *sujet* où un *rôle* (par exemple, pour la situation-problème SMASH, la tâche globale d'analyse quels que soient les rôles joués est d'« enquêter sur l'accident et faire une reconstitution »).

Subject : permet de faire référence aux personnes auxquelles s'adresse la situation-problème que l'on décrit. Ce concept ne sert que dans les modèles d'*expression initiale des besoins* puisque dans les modèles de *conception* l'individu ou le groupe réel, jouant les rôles décrits, ne sont pas modélisés car ils sont en dehors des limites du système d'apprentissage (propriété de **reproductibilité** présentée en section 6.2.1.4) : ils sont en effet liés à l'instanciation (*binding*) du modèle et à l'exécution (*runtime*) de la situation d'apprentissage.

Obstacle : cet élément permet de préciser dans les modèles d'*expression initiale des besoins* le concept d'*obstacle* tel que nous l'avons étudié au chapitre sur l'étude des PBL (chapitre 2).

Objective : un *objectif* représente l'abstraction de l'ensemble des objectifs d'une situation problème et par extension ceux des situations d'apprentissage : objectifs d'apprentissage, objectifs disciplinaires, ou encore objectifs transversaux. Les modèles d'*analyse* et de *conception* emploient ce terme. L'équipe chargée de la description de l'apprentissage pourra préciser dans le modèle explicite pour ces situations-problèmes à quoi se rattachent les objectifs. En effet, tel que défini dans notre méta-modèle, un *objectif* n'est relié à aucun autre élément mais par héritage de **Classifier** il peut être associé à d'autres **Classifier** ; de même **Relation** peut participer à créer des dépendances particulières entre *objectifs* ; il est même possible de définir des *sous-objectifs* en utilisant la relation de *generalization* entre **Classifier**.

SuccessCriterion : les *critères de succès* sont utiles dans les modèles d'*analyse* pour décrire par exemple l'*analyse critériologique* (voir 2.3.2) dont le but est de définir correctement les différents objectifs d'apprentissage, et ainsi, d'en déduire correctement la tâche et les critères garantissant la réussite de l'apprentissage. Les *critères de succès* sont également utilisés dans les modèles de *conception* pour indiquer quelles informations seront données aux acteurs (apprenants comme tuteurs) pour faciliter leur auto-évaluation, c'est-à-dire les aider à savoir s'ils ont oui ou non atteints l'*objectif*. Dans tous les cas d'utilisation, les *critères de succès* sont donc rattachés à un élément auquel ils font référence. Cette relation peut selon les modèles et le besoin des concepteurs se réaliser par le biais d'une *association* comme avec une *relation*.

Les instances des différents exemples de **StaticPBLElement** que nous proposons, ainsi que ceux

que peut ajouter le concepteur par instanciation directe de **StaticPBLElement**, peuvent être reliés entre eux dans les modèles produits par les concepteurs de multiples façons :

– par des *associations* puisqu'ils héritent tous de **Classifier** ;

– par des instances de **Relation**.

De nombreux éléments d'analyse comme ceux que nous venons de citer peuvent être reliés à d'autres éléments issus de modèles de conception afin de spécifier explicitement les changements ou les évolutions entre l'expression initiale des besoins, l'analyse et la conception. La relation de dépendance nommée **Trace** (définie en 7.3.2.3) permet de suivre ces changements.

7.3.3.2 DynamicPBLElement

Définition

DynamicPBLElement représente, comme **StaticPBLElement**, une abstraction des concepts d'une situation-problème utilisables pour l'*expression initiale des besoins* et l'*analyse*. La différence avec **StaticPBLElement** réside dans le changement d'états de l'élément considéré.

Le méta-attribut **dynamicKind** permet d'indiquer le type de l'élément de la situation-problème considérée.

Lien avec le paquetage `CPM_Foundation`

DynamicPBLElement est une spécialisation de **Classifier** et peut alors participer dans des associations avec d'autres éléments d'analyse statiques comme dynamiques ; il peut aussi contenir des définitions imbriquées, c'est-à-dire d'autres **Classifier** peuvent être définis dans son espace de nommage.

En tant que **Classifier**, **DynamicPBLElement** hérite aussi de la possibilité de décrire son comportement interne à l'aide de **StateMachine**. Cette propriété distingue la différence essentielle entre **DynamicPBLElement** et **StaticPBLElement**. La dynamique est ainsi modélisée au travers de la description des différents états que peut prendre l'élément considéré.

Exemples

La flexibilité de notre méta-modèle permet, dans n'importe quel modèle de PBL, de définir de nouveaux éléments dynamiques en instanciant directement la méta-classe **DynamicPBLElement** et en précisant le nom du nouvel élément en instanciant le méta-attribut **dynamicKind**.

Toutefois, nous proposons quelques éléments dynamiques pour l'analyse de PBL qui nous ont paru utiles dans nos travaux de modélisation. Cette liste n'est qu'un répertoire de base. Les concepteurs peuvent ajouter de nouveaux types si nécessaire pour d'autres situations-problèmes modélisées avec notre langage.

Resource : une *ressource* représente dans notre méta-modèle tout élément qui est produit, consommé/utilisé ou modifié par une *activité* (des apprenants comme des tuteurs). Ainsi, il peut s'agir de documents, de pièces d'information, de services ou d'outils utilisés dans l'activité. Le méta-attribut **resourceKind** permet d'indiquer le type de la ressource.

Cet élément regroupe, par exemple, les concepts de *Learning Object* et *Service* des travaux d'EML-OUNL et IMS-LD [Kop02, IMS03c]. En effet, l'objectif de nos travaux est d'aider et de supporter l'analyse et la conception de PBL indépendamment d'un contexte d'exécution. Ainsi, les ressources décrites peuvent, par exemple, ne pas exister encore sur support

numérique et donc ne pas répondre à la définition générale de *Learning Object* [Wil00]. De même pour les services, l'analyse d'une PBL peut prévoir l'utilisation d'un service générique (par exemple communication asynchrone) pour une activité sans être contrainte de préciser davantage la nature précise du service réellement utilisé à l'exécution. En effet, le choix d'un service plus précis de type *courrier électronique* ou bien *forum*, par exemple, n'intervient que dans la phase d'*implémentation*.

Resource est un concept important des modèles d'*analyse* mais aussi de *conception*.

MentalRepresentation : les représentations mentales des apprenants sont utiles pour l'*analyse* d'une situation-problème. En effet, dans l'apprentissage, les apprenants sont amenés à faire évoluer leurs connaissances[107] modélisées au travers des représentations mentales[108]. Les différentes évolutions possibles de ces représentions mentales peuvent être représentées à travers des diagrammes *statemachine* modélisant le comportement dynamique de la *représentation mentale* associée. Les différents *états* ainsi décrits correspondent à ce que l'on retrouve parfois dans la littérature sous les noms de *conception* ou *misconception*. En effet, il ne s'agit pas seulement de modéliser uniquement les connaissances qui vont être acquises, mais aussi d'autres déjà existantes qui seront renforcées, voire supprimées (dans le cas des *misconception* que l'apprentissage doit surmonter). Dans les apprentissages par situations-problèmes les *misconceptions* ont une place importante car elles font partie intégrante de l'analyse de la situation d'apprentissage que l'on veut mettre en œuvre : certaines activités peuvent *volontairement*[109] amener les apprenants à une mauvaise conception du problème qui sera corrigée par la suite (par des confrontations/discussions entre apprenants par exemple).

Une **MentalRepresentation** peut être alors reliée à un apprenant particulier comme à un élément de modélisation représentant un ensemble d'apprenants.

Les différents états composants une *représentation mentale* peuvent servir à préciser des *pré-requis*, ou *post-conditions* sur les *activités*.

Contrainte

Tout *StateMachine* exprime le comportement dynamique d'un **DynamicPBLElement** :

```
context CPM_Foundation::StateMachine inv:
self.oclIsTypeOf(StateMachine) implies
    self.context->notEmpty() and
    self.context.oclIsKindOf(DynamicPBLElement)
```

7.3.3.3 RegulationMethod

Définition

Une *méthode de régulation* permet d'ajouter aux modèles spécifiés avec le langage CPM des informations sur la manière de réguler certains éléments, c'est-à-dire des informations destinées au

[107] *Connaissance* au sens le plus large : c'est-à-dire savoirs, savoir-faire.

[108] Nous limitons cette réflexion des domaines de la psychologie et de la philosophie à l'hypothèse de Piaget selon laquelle l'apprentissage consiste à passer d'une représentation métaphorique à une représentation de plus en plus conceptualisée (tirée de [Mei02b] p.191).

[109] Du point de vue de la conception.

tuteur (pour de la régulation de type *assistance*, *guidage*) ou bien des informations destinées à une régulation *automatique* par l'environnement informatique d'exécution (régulation de type *suivi* des activités, des productions, etc.). Nous considérons que la précision d'une méthode de régulation est un type particulier de **Guidance** tel que nous l'avons défini dans le paquetage **BasicElement**. Ainsi, chaque *méthode de régulation* peut être rattachée à un ou plusieurs éléments de nos modèles quels qu'ils soient (**ModelElement**).

Exemples

Deux types de méthodes de régulation sont possibles : celle reposant sur des valeurs quantitatives (*i.e.* le temps passé dans une activité, le niveau de progression dans la réalisation d'une ressource, etc.) représentée à travers le concept de **QuantitativeRM** ; et celle basée sur des valeurs qualitatives [Bla96] (*i.e.* fin d'une étape d'une activité, information renvoyée par l'apprenant indiquant qu'il a terminé de lire, écrire un document, etc.) représentées avec le concept **QualitativeRM**.

Les informations de régulation décrites *via* ces concepts ne sont qu'informatives et non formelles.

7.3.3.4 DeclarativeKnowledge et ProceduralKnowledge

Définition

DeclarativeKnowledge et **ProceduralKnowledge** permettent de définir une description interprétable par l'outil informatique. Le choix du langage de programmation ou de tout autre dialecte interprétable dépend du concepteur qui utilisera cet élément de modélisation dans ses spécifications. Alors que **DeclarativeKnowledge** aura un contenu interprétable de type déclaratif (*XML* par exemple), **ProceduralKnowledge** contiendra des informations procédurales (*Java* par exemple). Ces deux éléments héritent d'**ExternalDescription** ; ils héritent donc des propriétés (attributs) indiquant diverses informations sur la nature *physique* du fichier. Comme la description externe de base, ces éléments peuvent être rattachés à un ou plusieurs éléments de modélisation (n'importe quel concept de notre méta-modèle).

Lien avec le paquetage `CPM_Foundation`

DeclarativeKnowledge et **ProceduralKnowledge** héritent de **ExternalDescription** et donc de **ModelElement**

Exemple : les *méthodes de régulation* (présentées dans ce même paquetage) peuvent être associées à ces éléments externes interprétables afin de formaliser leur contenu.

7.3.3.5 PBLConstraint

Définition

PBLConstraint permet de définir les différentes contraintes ou restrictions pouvant s'appliquer à tous les éléments de notre méta-modèle.

Des exemples de contraintes pédagogiques sont : pré-conditions et post-conditions à une activité, ou encore pré-conditions d'utilisation d'une ressource, contrainte de temps pour les activités.

Une contrainte peut au delà d'une simple assertion en langage naturel être spécifiée plus formellement sous forme d'expression booléenne ; elle peut également faire référence aux différents états possibles de certains éléments de modélisation héritant de **Classifier** dont le comportement peut être précisé sous forme de diagramme d'états.

Le méta-attribut **constraintKind** permet d'indiquer le type de la contrainte. Comme pour **Relation**, nous avons choisi de ne pas ajouter notre méta-attribut **constraintKind** à la méta-classe **Constraint** déjà existante mais plutôt de le rattacher à une nouvelle méta-classe. Ce choix permet de séparer les fondations des extensions et ainsi d'améliorer la compréhension et l'utilisation de notre méta-modèle.

Relation avec les autres concepts

Tout élément de modélisation, c'est-à-dire ici tous nos concepts liés aux situations d'apprentissage, peuvent être *contraints* ou associés à des contraintes. Comme pour la sémantique UML, une **Constraint** est une assertion, et non un mécanisme exécutable, qui indique une restriction. Une contrainte peut être reliée à plusieurs éléments et ce dans un ordre qui est conservé. Ainsi, **Constraint** peut permettre de contextualiser certaines contraintes : par exemple décrire une contrainte d'utilisation d'un document uniquement pour une activité donnée.

Lien avec le paquetage `CPM_Foundation`

PBLConstraint hérite directement de **Constraint** du paquetage `CPM_Foundation`.

Exemples

La flexibilité de notre méta-modèle permet dans n'importe quel modèle de PBL de définir de nouveaux éléments dynamiques en instanciant directement la méta-classe **PBLConstraint** et en précisant le nom du nouvel élément *via* le méta-attribut **constraintKind**.

Toutefois, nous proposons quelques contraintes pour la conception de PBL qui nous ont paru utiles.

Precondition et Postcondition : à chaque concept d'*activité* (défini dans la sous-section suivante 7.3.4.4) peut être associé une *pré-condition* et une *post-condition*. Ces contraintes sont exprimées sous la forme d'une expression booléenne (qui est une chaîne de caractères) suivant la syntaxe similaire de celle d'une condition (*guard*) UML ([OMG03d] section 2.12.2.6). La condition est exprimée en termes des états pris par les *ressources* (**Resource** dans notre méta-modèle) qui sont des paramètres de l'*activité*.

Exemple : une activité de notre cas d'étude SMASH, *lire le premier témoignage et répondre au QCM associé*, a les paramètres *Témoignage_1* et *QCM_1* en entrée et sortie. La pré-condition associée peut avoir la forme[110] :

```
(Témoignage_1 in state NotRead) and (QCM_1 in state NotAnswered)
```

De même, la post-condition peut être :

```
(Témoignage_1 in state Read) and (QCM_1 in state Answered)
```

Il faut alors que les différents états (par exemple *NotRead* et *Read* pour la ressource *Témoignage_1*) soient définis préalablement.

Contraintes : une *activité* ne peut pas avoir plus d'une pré-condition, ni plus d'une post-condition (dans le cas où l'on désire en spécifier plusieurs il suffit de faire usage de la conjonction *and* comme dans l'exemple précédent).

```
context Activity inv:
```

[110] L'expression n'est pas écrite en OCL.

```
        (constraint->select(c |
            c.oclIsKindOf(Precondition)))->size() < 2

    context Activity inv:
        (constraint->select(c |
            c.oclIsKindOf(Postcondition)))->size() < 2
```

CollaborativeMode : cette contrainte relie différentes activités pour indiquer que leurs rôles associés sont en collaboration. Cette contrainte est détaillée en 7.3.5.3.

7.3.4 Paquetage structurel : `CPM_StructuralPackage`

Ce paquetage traite de la définition des différents concepts et relations liés à la structure des situations-problèmes coopératives et capturés dans le *scénario*. Le concept d'*activité* est présenté et décomposé en différents éléments permettant la définition complète et précise de scénarios.

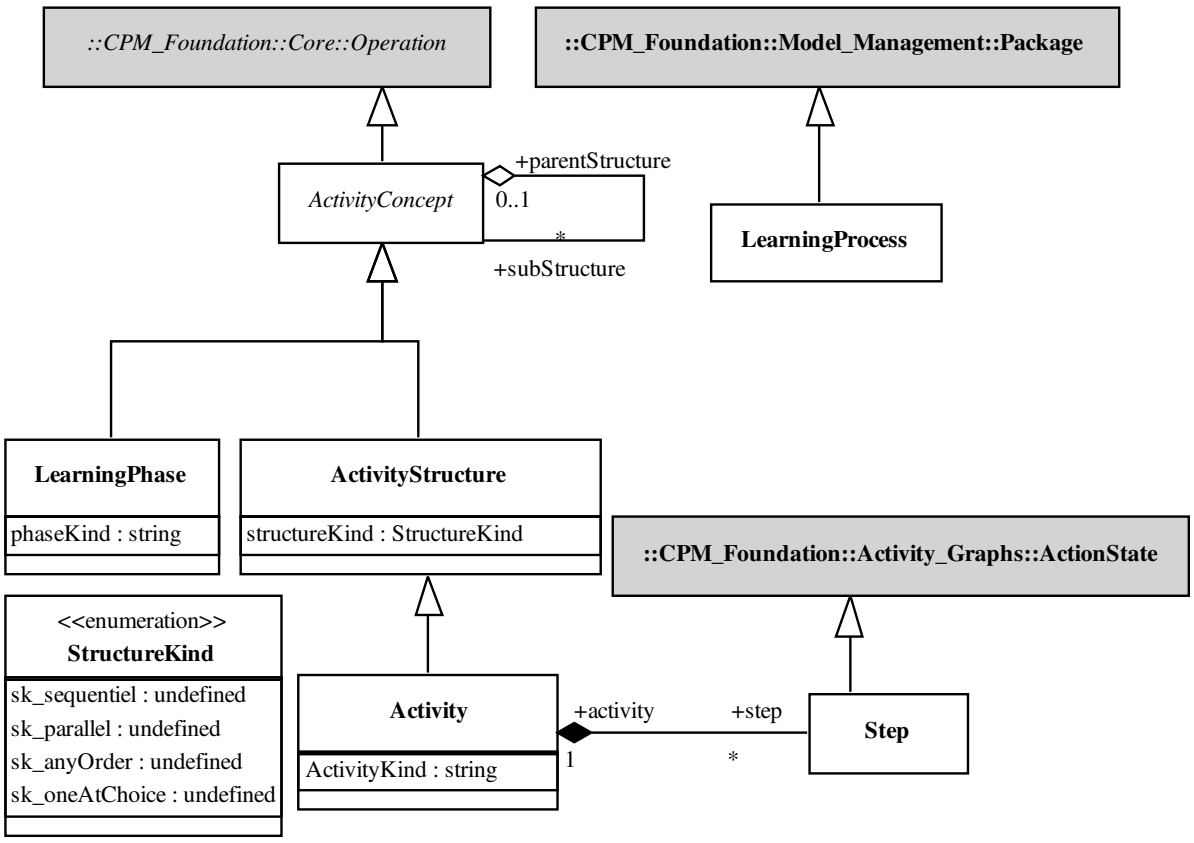

FIG. 7.15 – Le paquetage structurel détaillé : `CPM_StructuralPackage`.

7.3.4.1 ActivityConcept

Définition

ActivityConcept représente une abstraction de tous les concepts d'activités utilisables (**LearningPhase**, **ActivityStructure**, **Activity** et **Step**) pour la conception de situations-problèmes. Dans notre méta-modèle, **ActivityConcept** est une méta-classe abstraite afin de modéliser explicitement le fait que les modèles devront instancier les autres sous méta-classes (au sens de l'héritage).

Lien avec le paquetage `CPM_Foundation`

StaticPBLElement est une spécialisation de **Operation**. Ce choix est justifié dans la section suivante consacrée au paquetage social.

Un *ActivityConcept* peut être décomposé en d'autres *ActivityConcept* par le biais de l'association appelée **subStructure**. La décomposition peut être aussi modélisée en utilisant un graphe d'activités dans le cas où l'association **subStructure** est dérivée de la structure en graphe d'activités (voir la contrainte suivante). Un *ActivityConcept* peut être alors référencé au travers d'un **ActionState** dans un graphe d'activité[111] (voir annexe A).

Le concept de décomposition (*Work-Breakdown Structure* [OMG01b]) peut être décrit en utilisant différentes constructions du langage CPM :

- la décomposition en utilisant l'association **subStructure** permet de spécifier qu'un *ActivityConcept* est décomposé en d'autres *ActivityConcept*.
- la décomposition d'un *ActivityConcept* peut être représentée en détail par des graphes d'activités limités à un seul niveau de décomposition.
- la dépendance **Precedes** (décrite en 7.3.2.3) fournit la possibilité de créer une séquence d'*ActivityConcept* appartenant au même niveau de décomposition.

Contraintes

La décomposition d'activités par le biais de graphes d'activités est limitée à un niveau (cette règle s'applique également à la description des comportements des **DynamicPBLElement** par des **StateMachine** sur un niveau) :

```
context StateMachine inv:
    self.top.subvertex->forall( s |
        not s.oclIsKindOf(CompositeState) )
```

L'opération reliée à un **ActionState** doit être de type **ActivityConcept** :

```
context ActionState inv:
    self.entry.operation.oclIsKindOf(ActivityConcept)
```

S'il y a un graphe d'activité utilisé pour décomposer un **ActivityConcept** alors **subStructure** doit être dérivé :

```
context ActivityConcept inv:
    self.behavior->notEmpty() implies
        self.behavior.top.subvertex->select( a |
```

[111] On parle de graphe d'activités dans le contexte du méta-modèle CPM indépendant d'UML ; au niveau du profil CPM il s'agira alors de diagramme d'activités UML.

```
        a.oclIsKindOf(ActionState))->collect(b |
            b.entry.operation) = self.subStructure
```

7.3.4.2 LearningPhase

Définition

Une **LearningPhase** est une spécialisation d'**ActivityConcept** qui permet de définir la décomposition *externe* du scénario de la situation d'apprentissage. Nous appelons décomposition *externe* le découpage en séquences du scénario indépendamment des activités jouées par des rôles particuliers (*externe* vis-à-vis des activités dont nous verrons plus loin le découpage *interne*) .

Cette décomposition *externe* correspond aux concepts de *Phase*, *Cycle de vie* et *Iteration* employés pour structurer les processus dans les méthodes d'ingénierie logicielle [OMG01b].

En comparaison avec la spécification IMS-LD [IMS03c] (section 4.3.3), **LearningPhase** correspond au concept d'*acte*. Toutefois, alors que pour IMS-LD il n'y a qu'un seul niveau de décomposition du scénario[112], **LearningPhase** permet de définir autant de niveaux que nécessaire ; il hérite en effet de l'association **subStructure** d'**ActivityConcept**.

Le méta-attribut **phaseKind** permet d'indiquer le type de la phase. Ceci permet de catégoriser les différentes phases proposées dans les modèles.

Lien avec le paquetage *CPM_Foundation*

LearningPhase hérite d'**ActivityConcept** et est donc une spécialisation d'**Operation**.

Exemple

Dans nos travaux, un scénario de situations-problème se décompose tout d'abord en actes, puis en scènes. L'exemple suivant montre un extrait d'un scénario suivant ce découpage :

```
Acte 1 : Présentation de la situation-problème
    Scène 1 : présentation du contexte
    Scène 2 : présentation des objectifs
    Scène 3 : présentation des différents rôles
    Scène 4 : présentation des différentes ressources attribuées aux rôles
Acte 2 : Analyse des ressources et production de résultats
    Scène 1 : recherche de ressources supplémentaires
    Scène 2 : analyse des ressources récoltées
    Scène 3 : production d'un document de synthèse présentant la solution
Acte 3 : Présentation des solutions et discussions
    ...
```

Ces deux niveaux de scénario (acte et scène) sont modélisés *via* **LearningPhase**. Nous illustrons une modélisation semblable à ce scénario dans le chapitre consacré à l'expérimentation du langage CPM (chapitre 9).

[112]Nous rappelons en effet que le concept de *play* ne permet pas d'ajouter un niveau supplémentaire étant donné qu'il a pour sémantique de définir les multiples réalisations pédagogiques d'une même scénario.

7.3.4.3 ActivityStructure

Définition

Une **ActivityStructure** est également une spécialisation d'**ActivityConcept**. Ce concept définit un ensemble d'activités pour un rôle particulier seulement, contrairement donc à **LearningPhase** qui concerne tous les acteurs.

En comparaison avec la spécification IMS-LD [IMS03c] (section 4.3.3), **ActivityStructure** correspond au concept d'*activity-structure*.

ActivityStructure hérite de l'association **subStructure** d'**ActivityConcept** avec toutefois la contrainte qu'une **ActivityStructure** ne peut se décomposer qu'en **ActivityStructure** ou **Activity** (les deux concepts dépendants d'un rôle particulier).

Le méta-attribut **structureKind** permet d'indiquer le type d'ordonnancement des sous-activités la composant :

- **sk_sequentiel** : le rôle associé devra réaliser les activités de la structure dans l'ordre fourni (cet ordre devra être modélisé entre les composantes par le biais d'un graphe d'activités ou grâce à l'utilisation de la dépendance **Precedes**).
- **sk_selection** : le rôle peut choisir un certain nombre d'activités à réaliser parmi celles fournies dans la structure (une contrainte sur cet attribut permet dans les modèles de conception de préciser le nombre minimal d'activité à réaliser pour considérer la structure comme terminée).
- **sk_oneAtChoice** : il ne s'agit que d'un cas particulier de **sk_selection** pour une valeur fixée à 1.
- **sk_anyOrder** : toutes les activités de la structure devront être réalisées par le rôle mais dans n'importe quel ordre. Il s'agit à nouveau d'un cas particulier de **sk_selection** avec le nombre d'activités à réaliser fixé à la cardinalité de l'ensemble fourni.

Lien avec le paquetage `CPM_Foundation`

ActivityStructure hérite d'**ActivityConcept** et est donc une spécialisation d'**Operation**.

Contrainte

La décomposition d'une **ActivityStructure** ne peut se réaliser qu'en termes d'**ActivityStructure** ou d'**Activity** :

```
context ActivityStructure inv:
    self.subStructure->forall( v |
        v.oclIsKindOf(ActivityStructure) or
        v.oclIsKindOf(Activity))
```

7.3.4.4 Activity

Définition

Une **Activity** est une spécialisation d'**ActivityStructure** car les deux concepts partagent une même relation avec le concept de **Rôle** (voir paquetage social).

Activity est le même concept que son homonyme dans la spécification IMS-LD [IMS03c] (section 4.3.3). Toutefois, dans notre méta-modèle nous ne distinguons pas les activités réalisées par les apprenants (*learning activity*) de celles réalisées par les tuteurs (*support activity*).

Bien qu'**Activity** hérite de l'association **subStructure** d'**ActivityConcept**, celle-ci ne doit pas être employée. Une nouvelle association entre **Activity** et **Step** définit qu'une activité ne peut se décomposer qu'en éléments *atomiques* (d'où la redéfinition de l'association **subStructure**) appelés **Step**.

Le méta-attribut **activityKind** permet de préciser le type d'activité modélisée, par exemple s'il s'agit d'une activité d'apprentissage ou encore de tutorat (pour reprendre les termes de la spécification IMS-LD). Toutefois, d'autres types peuvent être utilisés selon les concepteurs.

Lien avec le paquetage *CPM_Foundation*

Activity hérite d'**ActivityStructure** et est donc une spécialisation d'**Operation**.

Contrainte

Une **Activity** se décompose en **Step** par le biais de l'association *step* alors que l'association *subStructure* n'est plus employée :

```
context Activity inv:
    self.subStructure->isEmpty()

context Activity inv:
    self.step->forall( v | v.oclIsKindOf(Step))
```

7.3.4.5 Step

Définition

Un **Step** permet de décrire la composition *interne* d'une activité. Il s'agit d'un élément *atomique* : un *step* n'est pas décomposable.

Step est défini dans le contexte précis de l'activité le contenant : il partage donc le même rôle et les mêmes ressources (voir 7.3.5.2).

Lien avec le paquetage *CPM_Foundation*

Step hérite d'**ActionState** de manière à ce que les différentes *étapes* dans une activité puisse être représentées par un graphe d'activités.

Contrainte

Contrairement aux autres concepts d'*activités* de ce paquetage, un **Step** n'est pas associé à un concept dérivant d'**Operation** (puisque les *steps* ne sont pas décomposables à leur tour) :

```
context Step inv:
    self.entry->isEmpty()
```

7.3.4.6 LearningProcess

Définition

LearningProcess est le concept auquel se rattachent tous les autres concepts pour la description de scénario pédagogique.

Il correspond au concept de *method* de la spécification IMS-LD. Son usage est alors limité aux modèles de conception avancée.

Lien avec le paquetage *CPM_Foundation*

LearningProcess hérite de **Package** de manière à pouvoir contenir d'autres éléments.

7.3.5 Paquetage social : CPM_SocialPackage

Ce paquetage étend le modèle conceptuel présenté en section 7.1.2. Il définit la structure *sociale* des situations d'apprentissages qui nous intéressent. Le terme *social* désigne ici les relations entre les différents rôles, activités et ressources.

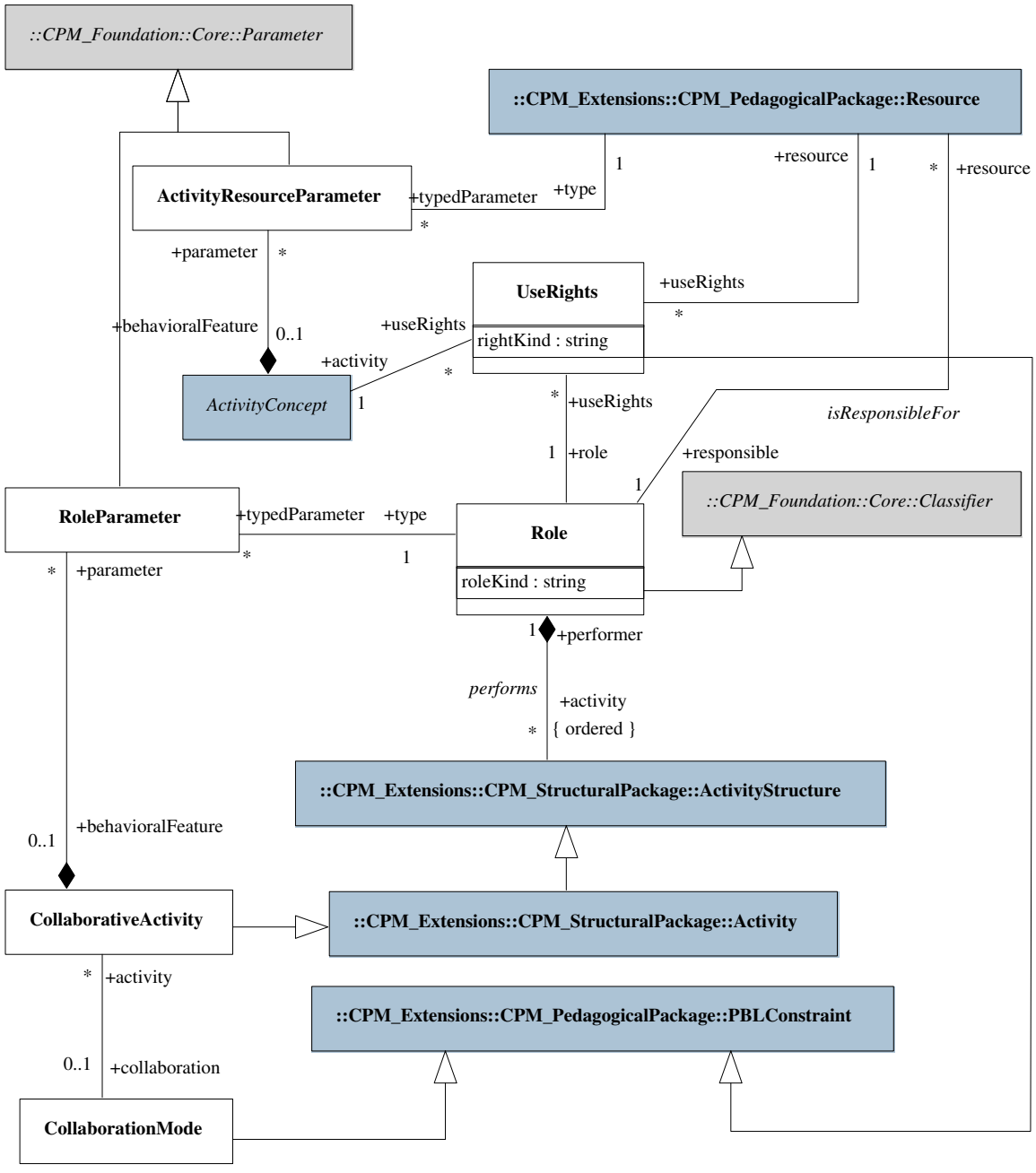

FIG. 7.16 – Le paquetage social détaillé : `CPM_SocialPackage`.

7.3.5.1 Role

Définition

Un **Role** permet de spécifier les *rôles situés* joués sur la globalité de la situation-problème et liés à la situation *authentique* qu'elle met en scène (voir la section 7.1.2 concernant la gestion des différents types de rôle).

Toutefois, deux catégories principales de participation sont recensées : les apprenants et les tuteurs. Comme leur intitulé varie selon le modèle de PBL suivi (pour certains on parle par exemple de guide plutôt que de tuteur), il est possible de spécifier le type de tout rôle grâce au méta-attribut **roleKind** : apprenant, participant, élève, tuteur, assistant, entraîneur, facilitateur...

Les rôles peuvent être définis par spécialisation (ou héritage) d'autres rôles afin de permettre de manipuler dans les modèles un ensemble de rôles plus facilement. Par exemple, le rôle *Enquêteur* (typé *roleKind=apprenant*) peut être spécialisé en rôles *Enquêteur 1* et *Enquêteur 2* pour désigner deux rôles distincts dans la situation-problème SMASH travaillant sur des ressources bien différentes.

Des rôles peuvent également être créés afin d'agir en tant que *proxy* à d'autres rôles dans un contexte particulier du scénario. En exemple, la scène 2 de l'acte 3 de SMASH nécessite de manipuler les rôles locaux de *présentateur* (joué par un seul rôle d'enquêteur parmi les rôles *Enquêteur 1* à *Enquêteur 4*) et *questionneur* (joué par les autres rôles). Afin de spécifier concrètement ce nouveau rôle local, une dépendance entre rôles peut être créée *via* **Relation** (elle sera alors typée *relationKind=SubRole*) dont l'usage permettra de relier chaque rôle à son *proxy* local.

Un **Role** n'est pas une personne réelle ou un acteur participant à la situation-problème. Ainsi, une personne donnée peut jouer plusieurs rôles et plusieurs personnes peuvent jouer un même rôle (notion de groupe).

Relations avec les autres concepts

- Un **Role** est responsable d'un ensemble de ressources (association *isResponsibleFor* avec **Resource**).
- Un **Role** réalise des activités (association *performs* avec **ActivityStructure** et tout autre concept le spécialisant comme **Activity** ou encore **CollaborativeActivity**). Les activités réalisées sont ordonnées (contrainte *{ordered}*) ; l'ordre sera spécifié par le biais des transitions du graphe d'activité.

Lien avec le paquetage `CPM_Foundation`

Role hérite de **Classifier** et ainsi peut participer dans des relations d'héritage et d'associations avec d'autres concepts héritant de **Classifier**.

Contraintes

Chaque **ActivityStructure** (et aussi **Activity** par héritage) est réalisée par un seul **Role** :

```
context ActivityStructure inv:
    self.performer.oclIsKindOf(Role)
```

Un **Role** ne réalise que des **ActivityStructure** (et **Activity** par héritage) :

```
context Role inv:
```

```
self.work->forall(a | a.oclIsKindOf(ActivityStructure)
```

7.3.5.2 ActivityResourceParameter

Définition

La méta-classe **ActivityResourceParameter** permet de spécifier que tout **ActivityConcept** est relatif aux **Resource** qu'il utilise. **ActivityResourceParameter** indique pour chaque ressource si elle est utilisée en *entrée* ou en *sortie*. Le travail décrit dans l'activité utilise alors les ressources en *entrée*, et crée ou modifie celles en *sortie*.

L'attribut *kind* hérité de **Parameter** (voir le paquetage `CPM_Foundation` : `:Data Types` en annexe A) est utilisé pour indiquer si la ressource associée à une activité est une entrée (*pdk_ in*), une sortie (*pdk_ out*), une entrée modifiable (*pdk_ inout*) ; la valeur *pdk_ return* n'est pas utilisée.

ActivityResourceParameter indique les ressources utilisées quelque soit le *type* d'activité :

- avec **LearningPhase** : il permet, dans les modèles d'analyse où la situation-problème est plus abstraite, de spécifier quelles ressources seront utilisées dans chaque partie du scénario. Comme chaque **LearningPhase** (acte, scène par exemple) est indépendante des rôles joués, les ressources paramétrées le sont également.
- avec **Activity** ou **ActivityStructure** ou **CollaborativeActivity** : dans ce cas, les ressources paramétrées dans le contexte d'une activité sont liées à un rôle particulier (association *performs* entre **ActivityStructure** et **Role**). Ce paramétrage des ressources est utilisé dans les modèles de conception.

Lien avec le paquetage `CPM_Foundation`

ActivityResourceParameter hérite de **Parameter** et ainsi hérite des associations (*typedParameter<->type*) avec un **Classifier** et (*parameter<->behavioralFeature*) avec une **Operation**.

Dans notre méta-modèle, le **Classifier** est restreint à **Resource**, tandis que l'**Operation** concerne seulement **ActivityConcept**.

Comme tous les concepts d'activités de notre méta-modèle peuvent être référencés par un **ActionState** dans les graphes d'activités, les **Resource** pourront également être référencés dans les mêmes graphes d'activités par des **ObjectFlowState** (ceci sera expliqué plus en détail au niveau de la notation du langage CPM dans le chapitre suivant).

Contraintes

Les deux contraintes suivantes précisent les restrictions sur les associations héritées de **Parameter** :

```
context ActivityResourceParameter inv:
    self.type.oclIsKindOf(Resource)
```

```
context ActivityResourceParameter inv:
    self.behavioralFeature->notEmpty() implies
    self.behavioralFeature.oclIsKindOf(ActivityConcept)
```

Le type d'un **ObjectFlowState** doit être de type **Resource** :

```
context ObjectFlowState inv:
    self.type.oclIsKindOf(Resource)
```

7.3.5.3 CollaborativeActivity, CollaborationMode et RoleParameter

Définition

Une **CollaborativeActivity** est une spécialisation d'**Activity** pour désigner les activités collaboratives de la situation-problème. Ces activités, bien que collaboratives, sont toutefois jouées par un seul rôle (à cause de la relation *performs*). Ainsi, le concept de **CollaborativeActivity** ne sert qu'à préciser le fait que l'activité se réalise en collaboration avec d'autres activités.

Afin de préciser ce lien de collaboration entre plusieurs activités, celles-ci devront être associées à un même **CollaborationMode** (association *collaboration<->activity*).

De plus, il est possible de préciser les sous-rôles joués par les différents participants à l'activité collaborative grâce à **RoleParameter**.

La prise en charge des activités collaboratives n'est pas explicite dans la spécification IMS-LD [IMS03c] (section 4.3.3). Toutefois, la gestion des *environnements* et des *services* semble y répondre en partie : des rôles différents peuvent réaliser des activités différentes mais partageant un environnement commun fournissant le même service (de communication par exemple). Les modèles de conception avancée d'IMS-LD ne précise rien de plus sur la collaboration. Toutefois, lorsque l'unité d'apprentissage ainsi conçue est instancié le service doit être préalablement configuré pour reconnaître les différents rôles des participants dans l'utilisation du service. Dans le cas du langage CPM, nous pensons que des modèles de conception respectant la règle de **reproductibilité** (définie en 6.2.1.4) peuvent décrire des contraintes sur les activités collaboratives à un certain niveau d'abstraction.

Lien avec le paquetage `CPM_Foundation`

ActivityCollaboration hérite d'**Activity** et est donc une spécialisation d'**Operation**.

CollaborationMode hérite de **PBLConstraint** ce qui lui permet de relier les différents concepts d'activités concernés.

RoleParameter hérite de **Parameter** au même titre que le concept **ActivityResourceParameter** présenté précédemment.

Contraintes

RoleParameter ne concerne que les rôles et les activités collaboratives :

```
context RoleParameter inv:
    self.type.oclIsKindOf(Role)
```

```
context RoleParameter inv:
    self.behavioralFeature->notEmpty() implies
    self.behavioralFeature.oclIsKindOf(CollaborativeActivity)
```

Toute instance de **CollaborationMode** relie au minimum deux activités collaboratives :

```
context CollaborationMode inv:
    self.constrainedElement->size() > 1
```

7.3.5.4 UseRights

Définition

Le concept de **UseRights** permet de spécifier les différents droits d'utilisation que peuvent avoir les différents rôles sur des ressources qu'ils partagent dans le contexte d'une activité particulière. Ceci est utile dans le cadre des activités collaboratives mais pas seulement : l'apprentissage coopératif peut, par exemple, impliquer des activités individuelles partageant des ressources identiques sur lesquelles des droits peuvent alors être spécifiés (voir les différents modes de coopération dans [Tar00, Dav01]).

La spécification de ces *droits d'utilisation* convient davantage aux modèles de conception qu'aux modèles plus amont.

Le contexte de l'activité dans laquelle sont précisés les droits d'utilisation d'une ressource par un rôle peut être celui d'une **Activity** comme celui d'une **LearningPhase** (pour fixer par exemple les droits entre un rôle et une ressource donnée pour l'ensemble d'un acte ou d'une scène).

Un **UseRights** est alors défini dans le contexte
- d'une **Resource** : association *useRights*<->*resource*,
- d'un **ActivityConcept** : association *useRights*<->*activity*,
- d'un **Role** : association *useRights*<->*role*.

Lien avec le paquetage `CPM_Foundation`

UseRights hérite de **PBLConstraint** ce qui lui permet de réaliser les associations avec un rôle, une activité et une ressource.

Contraintes

L'association *constraint*<->*constrainedElement* héritée de **PBLConstraint** doit permettre de relier un rôle, une activité et une ressource ensemble :

```
context UseRights inv:
    self.constrainedElement->size() = 3 and
    self.constrainedElement->select( a |
        a.oclIsKindOf(Role))->size() = 1 and
    self.constrainedElement->select( b |
        b.oclIsKindOf(ActivityConcept))->size() = 1 and
    self.constrainedElement->select( c |
        c.oclIsKindOf(Resource))->size() = 1
```

7.4 Bilan

Ce chapitre visait à présenter la syntaxe abstraite et une partie de la sémantique de notre langage CPM sous la forme d'un méta-modèle. Pour cela, nous avons dans un premier temps analysé les PBL au regard de la théorie de l'activité, puis présenté le modèle conceptuel à la base de notre langage. Dans un second temps, nous avons présenté et étudié des modèles d'informations provenant de divers travaux également liés directement ou indirectement à la théorie de l'activité. Ces modèles nous ont aidés à élaborer notre méta-modèle CPM (en particulier le SPEM [OMG02]) ou à valider par analogie les concepts et relations que nous proposons dans CPM.

Le méta-modèle CPM repose sur deux principaux paquetages : le paquetage de fondation, qui représente un sous-ensemble du méta-modèle d'UML (on peut considérer le *Core* du méta-modèle d'UML comme l'origine du MOF), et le paquetage des extensions qui regroupe tous les concepts, relations et contraintes de notre langage. Le choix de construction d'un méta-modèle CPM pour faciliter ensuite l'élaboration du profil CPM a été justifié dans le chapitre 6 consacré à la présentation de notre contribution.

Le paquetage des extensions est également décomposé à son tour en quatre sous-paquetages interdépendants afin de faciliter la présentation de la terminologie : paquetage des éléments de base, paquetage pédagogique, paquetage structurel, paquetage social. Les trois derniers paquetages font référence aux trois vues (pédagogique, sociale et structurelle) définies en 6.2.1.2 pour décrire les modèles de conception que nous voulons spécifier avec le langage CPM.

7.4.1 Analyse de la terminologie CPM vis-à-vis de la théorie de l'activité

Le tableau 7.2 présente les concepts de CPM faisant référence aux concepts du modèle de l'activité étudié en début de ce chapitre (7.1.1). Le sujet de l'activité dans la PBL correspond à la définition des différents rôles d'apprenants impliqués *via* le concept de **Role**. Les instruments ou outils pour la réalisation de l'activité correspondent aux **Resources**. L'objet de l'activité de la PBL est décrit au travers du concept d'**Objective**. La communauté rassemble l'ensemble des rôles d'apprenants décrits avec **Role** ; en effet, tous les apprenants partagent le même objet d'activité. La division du travail est décrite au travers de la scénarisation des activités donc *via* les différents concepts dérivés d'**ActivityConcept**. Finalement, les règles régissant les activités et les relations communautés-sujet se traduisent grâce aux différents concepts d'**ActivityResourceParameter**, de **Precondition** et **Postcondition**, de **RoleParameter**, de **PBLConstraint**, de **SuccessCriterion** et de **UseRights**.

Terminologie du triangle d'Engeström	Terminologie dans CPM
Sujet	**Role**
Outil	**Resource**
Objet	**Objective**
Communauté	ensemble des **Role** de type apprenant
Division du travail	**LearningPhase**, **ActivityStructure**, **Activity**, **CollaborativeActivity**, **Role**
Règles	**ActivityResourceParameter** **Precondition**, **Postcondition**, **RoleParameter** **PBLConstraint**, **SuccessCriterion**, **UseRights**

Tab. 7.2 – Correspondances entre la terminologie du triangle d'Engeström et celle de CPM

7.4.2 Analyse de la terminologie CPM vis-à-vis des modèles à produire

Il est possible de distinguer ou classer les différents concepts du langage CPM selon leurs usages dans les différents modèles produits dans la phase de conception. Tout d'abord, certains concepts représentent le vocabulaire nécessaire à l'étape d'*expression initiale des besoins*, celle où l'équipe de conception, et surtout l'enseignant, ont besoin de définir la PBL. D'autres concepts, à l'opposé, conviennent davantage à un usage de conception plus avancée pour lequel le vocabulaire doit pouvoir correspondre avec la terminologie d'autres langages existants comme les EML (voir le positionnement de la première proposition en section 6.1.2).

Pour cela, nous proposons les deux tableaux suivants exprimant pour l'un, les correspondances entre les termes de PBL issus de l'étude du chapitre 2 et ceux de CPM (tableau 7.3), et pour l'autre, les correspondances entre les concepts de la spécification IMS-LD (représentative de la standardisation actuelle des langages EML) et ceux de CPM (tableau 7.4).

Concepts issus de l'étude des PBL	Terminologie dans CPM
Sujet	**Subject**
Objectifs d'apprentissage	**Objective**
Tâche globale	**Task**
Rôles authentiques	**Role**
Ressources et matériels	**Resource**
Obstacle	**Obstacle**
Systèmes de contraintes	**PBLConstraint**
Critères de succès	**SuccessCriterion**

Tab. 7.3 – Correspondances entre la terminologie des PBL et celle de CPM

Nous ne commentons pas plus précisément le tableau 7.3 car la plupart des concepts correspondent directement. Par contre, nous donnons quelques commentaires pour le tableau 7.4 :

- Le concept **LearningPhase** permet de décrire l'ensemble du découpage des activités du scénario ; la hiérarchie des niveaux est réalisée en donnant des valeurs différentes par niveau à *phaseKind*. Il correspond alors aux concepts de *play* et *act* de la spécification d'IMS-LD, bien qu'il permette la définition de scénarios plus complexes.
- Le concept d'**Activity** de CPM permet de décrire les activités des apprenants (*learning activity*), comme les activités des tuteurs (*support activity*).
- **Role** permet de définir l'ensemble des rôles des PBL (*roleKind* permettant de spécifier pour chacun d'entre eux s'ils sont de type « apprenant » ou « tuteur »).
- Le concept *role-part* d'IMS-LD permet de relier les rôles avec les activités qu'ils réalisent (rôles et activités sont vus comme des composants ; voir figure 4.12). Dans CPM, une activité ne peut être réalisée que par un seul rôle ; ils sont reliés directement par une association dans le méta-modèle CPM.
- Pour IMS-LD, chaque activité peut faire référence à un *environment* agrégé en *services* et *learning objects*. Pour CPM, les activités manipulent directement des **Resource**.

Terminologie d'IMS-LD	Terminologie dans CPM
method	**LearningProcess**
play, *act*	**LearningPhase**
activity, *learning activity*, *support activity*	**Activity**
activity-structure	**ActivityStructure**
role, *learner*, *staff*	**Role**
role-part	$\emptyset$
environment, *learning object*, *service*	**Resource**
learning objective	**Objective**, **Postcondition**
prerequisite	**StaticPBLElement**, **Precondition**
condition et *property*	**PBLConstraint**, **StaticPBLElement**
notification	**Resource**, **PBLConstraint**

Tab. 7.4 – Correspondances entre la terminologie d'IMS-LD et celle de CPM

- Les objectifs d'apprentissage sont définis au niveau global dans IMS-LD (ils peuvent aussi être précisés au niveau de chaque activité mais cela n'apparaît pas dans le modèle conceptuel, figure 4.11). Dans CPM, **Objective** permet de décrire les objectifs aux niveaux global et local. De plus, **Postcondition** permet, au niveau de chaque activité, de fixer des « objectifs » pour la réalisation de l'activité (la sémantique n'est pas la même que celle des objectifs d'apprentissage).
- le concept de *Prerequisite* d'IMS-LD n'a pas de correspondance dans CPM en ce qui concerne les pré-requis à l'apprentissage (niveau global). Toutefois, il est possible d'ajouter ce concept par la suite en spécialisant **StaticPBLElement** (voir la section suivante 7.4.3). Sinon, au niveau local, il est possible de décrire les **Preconditions** aux activités, ce qui correspond à une forme locale de pré-requis.
- Les concepts *condition* et *property* du niveau B et C de la spécification d'IMS-LD permettent conjointement de raffiner et d'ajouter une personnalisation aux scénarios d'apprentissage[113]. Dans CPM, il est possible de créer des contraintes *via* **PBLConstraint** afin de décrire ces règles. La notion de *property* n'est pas définie dans CPM mais peut faire l'objet d'une spécialisation de **StaticPBLElement**. Nous verrons dans le chapitre suivant dédié à la notation CPM, une notation UML (branche conditionnelle) permettant de spécifier autrement des règles dans les modèles.
- Le concept *notification* d'IMS-LD (niveau C) permet d'envoyer des messages à des rôles ou de leur proposer de nouvelles activités sur la base d'événements pré-déterminés (fin d'une activité par exemple). Ceci n'a pas été intégré explicitement dans CPM. Toutefois, le concept d'événement peut être ajouté en spécialisant **Resource** (sémantiquement, un événement peut être considéré comme quelque chose de produit par l'activité). Par le biais de contraintes spécifiques (en spécialisant **PBLConstraint**) ou bien avec **Precondition**, il est possible de spécifier les pré-requis de certaines activités selon l'existence de certains événements.

[113]Par le biais de conditions de type « IF [*expression*] THEN [*show*, *hide*, ou *change something* ou *notify someone*] » dont les expressions sont définies sur des propriétés globales ou locales à l'unité d'apprentissage.

A l'inverse, de nombreux concepts de CPM n'ont pas d'équivalent dans la spécification IMS-LD (**Guidance**, **QuantitativeRM**, **QualitativeRM**, **RoleParameter**, **ActivityResourceparameter**, **CollaborativeActivity**, **ExternalDescription**, **DeclarativeKnowledge**, **ProceduralKnowledge**, **Step**, **Precedes**, **MentalRepresentation**, etc.). Il est possible de considérer alors le langage CPM comme plus riche d'un point de vue syntaxique que la spécification d'IMS-LD ; ceci étant principalement dû aux objectifs différents et aux usages plus larges fixés pour CPM (voir section 6.2.1).

7.4.3 Propriété de personnalisation du langage CPM

L'un des pré-requis pour la spécification du langage CPM concerne sa personnalisation (définie en section 6.2.1.4), c'est-à-dire qu'il soit « ouvert » au niveau de la terminologie (concepts et relations) afin de pouvoir être enrichi selon les besoins des différents concepteurs.

Le langage CPM répond à cette attente en proposant aux concepteurs de spécialiser certains concepts du méta-modèle CPM en utilisant le méta-attribut associé. La liste suivante rassemble ces différents concepts :

- **Relation**, pour créer une nouvelle relation dont le nom est précisé avec *relationKind*. Dans [Nod03a], nous donnons quelques exemples d'utilisation de relation : *ConsistsOf* / *IsEquivalent* pour relier une **Resource** à une autre **Resource**, *RefersTo* / *ConsistantWith* entre deux **LearningPhase**, *Implies* entre une **ActivityStructure** ou **Activity** et une autre, *Governs* / *Promotes* entre un **LearningPhase** et une **ActivityStructure**, etc.
- **Activity**, pour définir de nouvelles catégories d'activités (précisées avec *activityKind*). De telles catégories d'activités peuvent être, par exemple, celles identifiées dans [Etc02] pour spécifier la coordination : supervision directe, ajustement mutuel, etc.
- **StaticPBLElement**, pour ajouter de nouveaux concepts statiques (le nom est précisé avec *staticKind*) ;
- **DynamicPBLElement**, *idem* pour les concepts dynamiques (le nom est précisé avec *dynamicKind*) ;
- **Resource**, pour définir des catégories de ressources comme par exemples : outils, documents, services, etc. (la catégorie est précisée *via resourceKind*) ;
- **PBLConstraint**, pour définir de nouvelles contraintes sur les autres concepts et relations, par exemple pour préciser la durée des activités (le nouveau type de contrainte est précisé avec *constraintKind*).

Toutefois, ces ajouts sont à réaliser au niveau des modèles produits avec le langage CPM et non directement au niveau du méta-modèle CPM. Cette personnalisation est donc dépendante du langage CPM au niveau de sa syntaxe concrète, c'est-à-dire au niveau du profil CPM que nous présentons ci-après.

Chapitre 8

Profil CPM

« You can model 80 percent of most problems
by using about 20 percent of the UML»
The Unified Modeling Language User Guide,
Los Amigos : Booch, Jacobson, Rumbaugh

Sommaire

Le chapitre précédent a présenté en détail le méta-modèle CPM, ainsi que tous les concepts, relations et contraintes pour modéliser des situations-problèmes coopératives pendant la phase de conception. Le méta-modèle CPM correspond à la syntaxe abstraite du langage CPM ; il a permis aussi de définir la sémantique de l'ensemble de la terminologie. Le méta-modèle CPM a été également construit de manière à faciliter la démarche d'élaboration du profil CPM que nous allons présenter dans ce chapitre.

Les différentes étapes pour l'élaboration d'un profil ont été présentées dans le chapitre consacré à l'étude de la modélisation et de la méta-modélisation UML (5.3.3.3). Ces différentes étapes sont appliquées dans les prochaines sections de ce chapitre.

8.1 Identification du sous-ensemble du méta-modèle UML

La première étape consiste à identifier le sous-ensemble du méta-modèle d'UML qui sera inclus dans l'élaboration du profil.

Le profil CPM utilise les paquetages suivants du méta-modèle UML (UML dans sa version 1.5) :

- Core
- ExtensionMechanisms
- DataTypes
- CommonBehavior
- Collaboration
- UseCases
- StateMachines
- ActivityGraphs
- ModelManagement

Tous les éléments de chaque paquetage ne sont pas forcément utilisés (exemple : *Node* et *Artifact* du sous-paquetage `Classifiers` du paquetage `Core`).

Les classes, attributs et associations du paquetage `CPM_Foundation` présenté en annexe A sont directement représentés par les classes, attributs et associations équivalentes du méta-modèle UML. Ainsi, l'identification du sous-ensemble du méta-modèle d'UML a pu être facilitée par la décomposition du méta-modèle CPM en deux paquetages distincts : les fondations et les extensions. Toutefois, nous avons identifié l'usage du paquetage supplémentaire **UsesCases** (non présent dans le méta-modèle CPM) en vue d'adapter l'utilisation des diagrammes UML pour la notation CPM.

8.2 Correspondance entre le méta-modèle d'UML et le méta-modèle CPM

Maintenant que le sous-ensemble du méta-modèle d'UML a été défini, il s'agit d'identifier les classes de ce sous-ensemble qui serviront de *base* à la création de stéréotypes. Ces stéréotypes agiront en lieu et place des éléments du méta-modèle CPM. Le choix de ces classes de *base* est facilité par le méta-modèle CPM. La plupart des classes de base sont les classes du paquetage `CPM_Foundation` ayant fait l'objet d'une spécialisation par nos concepts CPM définis dans le paquetage `CPM_Extensions`.

La plupart des correspondances entre les concepts du paquetage `CPM_Extensions` et les classes du sous-ensemble du méta-modèle d'UML suivent le *patron* illustré dans la figure suivante (Figure 8.1)[114] : toute classe du `CPM_Extensions` va faire l'objet d'un stéréotype de la méta-classe UML correspondant à celle dont il hérite directement ou indirectement dans le méta-modèle CPM.

[114]La notation utilisée pour la relation d'extension est celle définie dans UML 1.X (elle a été modifiée dans UML 2.0).

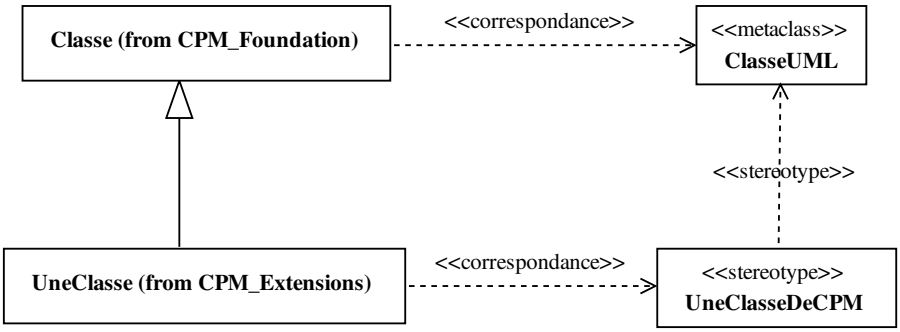

FIG. 8.1 – Patron de correspondance entre la plupart des classes du méta-modèle CPM vers les classes de base du méta-modèle d'UML.

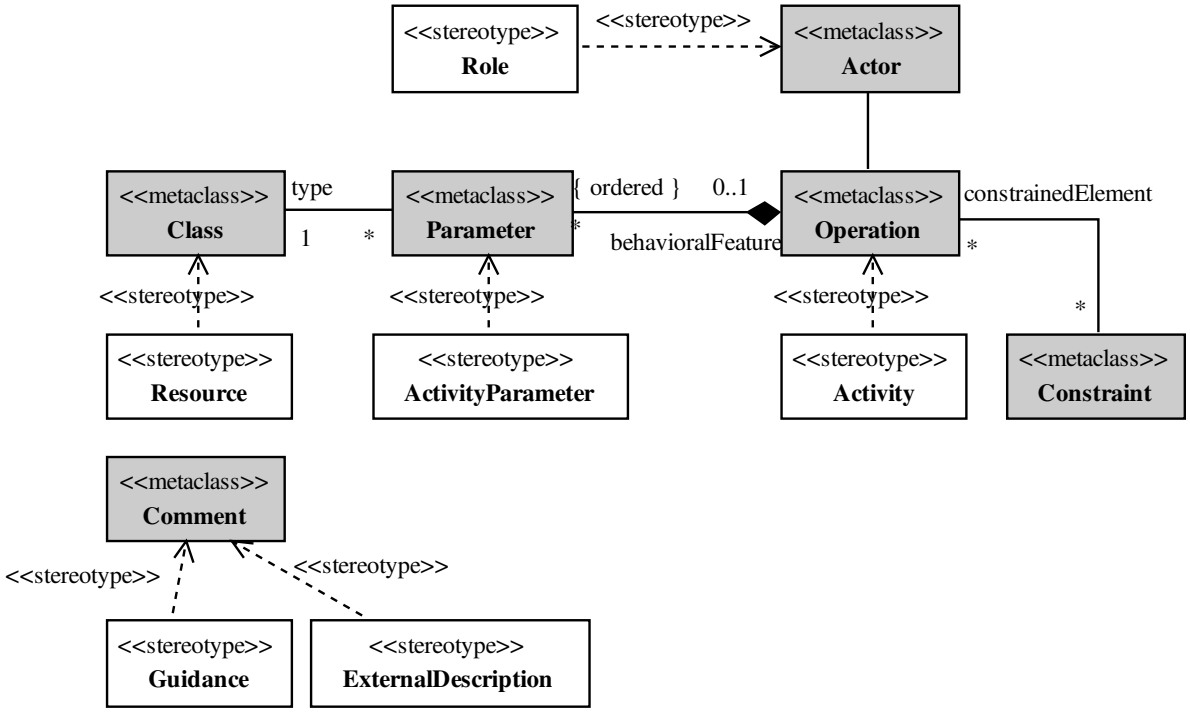

FIG. 8.2 – Quelques exemples de correspondance entre classes du méta-modèle CPM et classes du méta-modèle d'UML.

Un exemple de correspondance est montré dans la Figure 8.2. Dans cet exemple, les éléments grisés sont des classes du méta-modèle d'UML, tandis que les autres classes sont les éléments d'extension du méta-modèle CPM. Les différentes associations entre les classes grisées sont spécifiées directement ou indirectement (obtenues par héritage) entre les classes concernées dans le méta-modèle d'UML. Les classes du langage CPM étendent les classes du méta-modèle UML en définissant un stéréotype. Ces stéréotypes, considérés comme de nouvelles méta-classes, héritent donc des associations de leurs parents.

On peut donc remarquer, par exemple, que le stéréotype <<**ActivityParameter**>>[115] est défini par rapport à la méta-classe **Parameter** et donc qu'il hérite des associations vers **Class** et **Operation** ; ces associations sont redéfinies, conformément à la sémantique d'**ActivityParameter** définie dans le chapitre précédent, de manière à ce qu'**ActivityParameter** relie une **resource** à une **Activity** (définis également comme des stéréotypes pour **Class** et **Operation**).

L'objectif des correspondances est d'aider la mise en œuvre des éléments d'extensions du profil CPM. En effet, certains stéréotypes sont identifiés facilement par cette méthode. Toutefois, de nouveaux stéréotypes peuvent être définis et des correspondances pour tous les méta-attributs et associations du méta-modèle CPM doivent encore être spécifiées. La correspondance pour les méta-attributs du méta-modèle CPM est directe : tout méta-attribut CPM est défini comme un type de valeur marquée. Ce dernier est associé au stéréotype correspondant à la méta-classe CPM. La sous-section (8.3.3) présente l'ensemble de ces types de valeurs marquées.

Les associations entre les classes du méta-modèle CPM doivent également trouver une correspondance avec les différentes possibilités de relation proposées par le méta-modèle d'UML. Le tableau 8.1 résume ces correspondances : pour chaque ligne nous donnons 1/ l'association du méta-modèle CPM (au format **NomClasse : :nomFinAssociation**), 2/ le nom de l'association UML correspondante et 3/ le nom du stéréotype créé, s'il existe, spécialisant l'association UML.

[115] Les stéréotypes de CPM sont notés entre chevrons pour les distinguer de la méta-classe CPM homonyme.

Association CPM	Correspondance UML	Stéréotype s'il existe
`ExternalDescription : :subject`	`Comment : :annotatedElement`	
`Guidance : :annotatedElement`	`Comment : :annotatedElement`	
`ActivityConcept : :subStructure`	représentation indirecte au travers des diagrammes d'activités	
`Activity : :step`	*idem*	
`Role : :resource`	représentée par une association de niveau M1 entre **Role** et **Resource**	*isResponsibleFor*
`Role : :activity`	`Classifier : :feature`	
`ActivityStructure : :role`	`Feature : :owner`	
`ActivityResourceParameter : :type`	`Parameter : :type`	
`ActivityResourceParameter : : behavioralFeature`	`Parameter : :behavioralFeature`	
`RoleParameter : :type`	`Parameter : :type`	
`RoleParameter : :behavioralFeature`	`Parameter : :behavioralFeature`	
`CollaborativeActivity : : collaboration`	`ModelElement : :constraint`	
`CollaborativeMode : :activity`	`Constraint : :constrainedElement`	
`UseRights : :role`	`Constraint : :constrainedElement`	
`UseRights : :activity`	`Constraint : :constrainedElement`	
`UseRights : :resource`	`Constraint : :constrainedElement`	

TAB. 8.1 – *Correspondances des associations du méta-modèle CPM avec le méta-modèle d'UML*

8.3 Éléments d'extension

Les différents éléments d'extension composant le profil CPM (stéréotypes, types de valeurs marquées et contraintes) sont spécifiés dans les sous-sections suivantes.

La plupart des stéréotypes sont identifiés par correspondance directe des classes parentes des concepts CPM dans le méta-modèle CPM avec leur classe homologue dans le sous-ensemble du méta-modèle d'UML. D'autres classes du sous-ensemble du méta-modèle d'UML sont proposées toutefois comme alternatives à la classe de base afin de pouvoir utiliser les concepts du langage CPM dans des diagrammes UML différents et ainsi augmenter les possibilités de description visuelle des modèles pour la conception de situations-problèmes coopératives.

8.3.1 Notion de *proxy* : utilisation des diagrammes d'activités et des diagrammes de cas d'utilisation

Les concepts du langage CPM sont définis comme stéréotypes de classes de base et apparaissent donc uniformément dans tous les diagrammes UML dans lesquels des références à ces classes de base sont possibles.

Toutefois, de manière à pouvoir faire référence aux concepts CPM d'activités (**LearningPhase**, **Activity**, **ActivityStructure** et **CollaborativeActivity**) dans des diagrammes d'activités où la classe de base **Operation** n'est pas utilisable directement, nous utilisons les instances d'**ActionState** comme notation *proxy* des concepts CPM d'activités. De manière similaire, les instances **ObjectFlowState** sont utilisées comme *proxy* pour les instances correspondantes de **Resource**.

Ainsi, le profil CPM permet à **ActionState** et **SubactivityState** d'apparaître comme une classe de base alternative pour les différents stéréotypes d'*activités pédagogiques*. Cette idée, déjà utilisée dans [OMG01b], est que l'élément de notation **ActionState** est un *proxy* pour l'**Operation** stéréotypée qui est associée avec le *CallAction* de l'**ActionState**. De même, **ObjectFlowState** est définie comme classe de base alternative pour **Resource** avec l'interprétation que cet élément de notation est un *proxy* pour le **Classifier** stéréotypé associé à l'**ObjectFlowState**.

De manière identique, l'utilisation des diagrammes de cas d'utilisation permet d'illustrer les relations entre rôles et activités. Pour cela, **UseCase** est utilisée comme classe de base alternative pour les concepts d'activités du langage CPM. La classe de base pour **Role** est alors **Actor** pour pouvoir être référencée dans les diagrammes de classes et les diagrammes de cas d'utilisation (le concept de **Role** est une spécialisation de **Classifier** dans le méta-modèle CPM ; mais comme **Actor** hérite de **Classifier** dans le méta-modèle d'UML il n'y pas contradiction vis-à-vis de la sémantique du concept).

Pour compléter cette interprétation, la dépendance UML *trace* (définie dans le méta-modèle CPM en 7.3.2.3) doit être alors créée entre l'activité et le **UseCase** qui le représente. Si deux activités sont représentées en tant que **UseCase** et que l'une de ces deux activités est une sous-activité de l'autre (association *subWork*) alors une relation d'*include* (définie dans le méta-modèle d'UML 1.X) peut être utilisée à partir de l'activité *conteneur* vers celle contenue.

8.3.2 Stéréotypes

Le tableau 8.2 donne un résumé complet de tous les stéréotypes du profil CPM basés sur les discussions précédentes.

Pour chaque stéréotype sont donnés les noms des méta-classes du sous-ensemble d'UML qui seront spécialisées par le stéréotype et le nom du stéréotype parent s'il existe. L'existence d'une contrainte s'appliquant au stéréotype est également précisée. Puis, une icône est fournie pour tous les stéréotypes dont l'usage dans les diagrammes UML pourra être géré par une notation particulière.

Lorsque plusieurs méta-classes font l'objet d'une spécialisation par le même stéréotype, cela signifie qu'elles permettent de modéliser le même concept dans des diagrammes différents (notation de *proxy* expliquée dans la sous-section précédente 8.3.1).

Stéréotype	**Méta-classe**	**Stéréotype parent**	**Contraintes**	**Icône**
ExternalDescription	`Core : :Comment`			
Guidance	`Core : :Comment`			
Relation	`Core : :Usage`			

Precedes	Core : :Usage	Relation	oui	
StaticPBLElement	Core : :Classifier			
Task	Core : :Classifier	Static-PBLElement		
Subject	Core : :Classifier	Static-PBLElement		
Objective	Core : :Classifier	Static-PBLElement		
Obstacle	Core : :Classifier	Static-PBLElement		
SuccessCriterion	Core : :Classifier	Static-PBLElement		
DynamicPBLElement	Core : :Classifier			
Resource	Core : :Classifier	Dynamic-PBLElement		
MentalRepresentation	Core : :Classifier	Dynamic-PBLElement		
DeclarativeKnowledge	Core : :Comment	External-Description		
ProceduralKnowledge	Core : :Comment	External-Description		
QualitativeRegulationMethod	Core : :Comment	Guidance		
QuantitativeRegulationMethod	Core : :Comment	Guidance		
PBLConstraint	Core : :Constraint			
Precondition	Core : :Constraint	PBLConstraint	oui	
Postcondition	Core : :Constraint	PBLConstraint	oui	
CollaborationMode	Core : :Constraint	PBLConstraint	oui	

LearningPhase	`Core : :Operation`
	`ActivityGraphs : :ActionState`
	`ActivityGraphs : :SubactivityState`
	`UseCases : :UseCase`
	`Core : :Classifier`
				

ActivityStructure	`Core : :Operation`
	`ActivityGraphs : :ActionState`
	`ActivityGraphs : :SubactivityState`
	`UseCases : :UseCase`
	`Core : :Classifier`
			oui	

Activity	`Core : :Operation`
	`ActivityGraphs : :ActionState`
	`ActivityGraphs : :SubactivityState`
	`UseCases : :UseCase`
	`Core : :Classifier`
		Activity-Structure	oui	

Step	`ActivityGraphs : :ActionState`		oui	
LearningProcess	`ModelManagement : :Package`			
LearningPackage	`ModelManagement : :Package`			
Role	`UseCases : :Actor` `ActivityGraphs : :Partition`		oui	
Activity-ResourceParameter	`Core : :Parameter`		oui	

CollaborativeActivity	`Core : :Operation`
	`ActivityGraphs : :ActionState`
	`ActivityGraphs : :SubactivityState`
	`UseCases : :UseCase`
	`Core : :Classifier`
		Activity		

RoleParameter	`Core : :Parameter` `Core : :Constraint`		oui	
UseRights	`Core : :Constraint` `StateMachines : :Transition`	PBLConstraint	oui	
performs	`Core : :Relationship`			
assists	`Core : :Relationship`			
trace	`Core : :Abstraction`			
isResponsibleFor	`Core : :Association`			

TAB. 8.2: *Les stéréotypes du profil* CPM

Nous donnons quelques commentaires sur la table des stéréotypes précédents :

- Certains stéréotypes n'ont pas d'icône : cela signifie que la spécification d'UML [OMG03d] ne permet pas de décrire les méta-classes de référence du stéréotype dans les diagrammes par le biais d'une représentation avec une icône. C'est le cas, par exemple, de la méta-classe **Association**.
- Les concepts du méta-modèle CPM directement rattachés à **ModelElement** (**ExternalDescription**, **Guidance**, **Declarativeknowledge**, **Proceduralknowledge**, **QualitativeRegulationMethod**, **QuantitativeRegulationMethod**) deviennent des stéréotypes de **Comment** (défini dans le paquetage `CPM_Foundation`). En effet, il s'agit du seul élément du méta-modèle d'UML dont la sémantique[116] convient.
- Différentes méta-classes sont spécialisées par le même stéréotype dans le but de permettre aux concepteurs de modéliser un même concept dans des contextes (diagrammes UML) différents.
- La plupart des concepts d'activités comme <<**LearningPhase**>> (sauf <<**Step**>>) peuvent être utilisés dans les diagrammes d'activités, les diagrammes de classes et les diagrammes de cas d'utilisation. La figure 8.3 montre les différentes possibilités de description.
- Le stéréotype <<**Role**>> spécialise également la méta-classe **Partition** afin de pouvoir regrouper, dans les diagrammes d'activités, les activités pédagogiques selon le rôle qui les réalise (voir figure 8.4).
- Le stéréotype <<**RoleParameter**>> spécialise également la méta-classe **Constraint** de manière à permettre aux concepteurs de décrire le rôle joué dans une collaboration dans les diagrammes d'activités.
- La notation proxy d'<<**ActivityResourceParameter**>> (stéréotype de **Parameter**) dans un diagramme d'activité consiste à faire pointer les ressources vers les activités pour indiquer « IN », pointer l'activité vers la ressource pour « OUT » et dans les deux sens pour « INOUT » (voir Figure 8.4).
- Le stéréotype <<**UseRights**>> spécialise également la méta-classe **Transition** afin de pouvoir décrire directement les droits d'utilisation sur les transitions reliant les ressources et les activités dans les diagrammes d'activités. **UseRights** relie une ressource, un rôle et une activité ; or dans certains diagrammes d'activités, les activités sont regroupées dans des partitions représentant les rôles les réalisant ; la transition entre une ressource et une activité est donc définie dans le contexte d'un rôle particulier : la sémantique est vérifiée (voir figure 10.2).
- Nous avons ajouté de nouveaux stéréotypes dont le concept associé n'a pas été présenté dans le méta-modèle CPM : il s'agit de <<**performs**>> et <<**assists**>> qui permettent d'exprimer la relation de réalisation active (*performs*) et passive (*assists*) entre un rôle et ses activités dans un diagramme de cas d'utilisation (voir un exemple en figure 9.16).
- Le stéréotype <<**LearningPackage**>> n'est pas présent dans le méta-modèle CPM ; il s'agit d'une notation supplémentaire pour désigner les paquetages des modèles construits en utilisant le langage CPM.

[116] « *A comment is an annotation attached to a model element or a set of model elements. It has no semantic force but may contain information useful to the modeler* » [OMG03d].

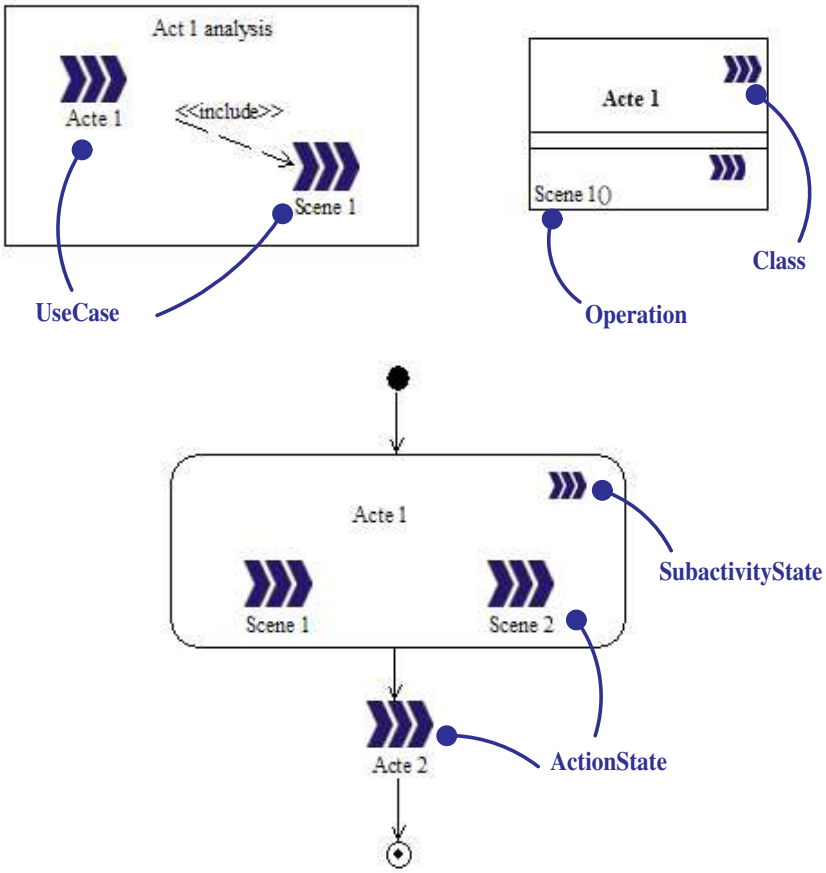

FIG. 8.3 – Exemples d'applications du même stéréotype <<**LearningPhase**>> sur des méta-classes UML différentes.

La figure 8.3 suivante illustre différentes représentations et méta-classes portant le même stéréotype <<**LearningPhase**>>. En haut à gauche se trouve un extrait de diagramme de cas d'utilisation dans lequel le stéréotype <<**LearningPhase**>> est associé à la méta-classe **UseCase** ; de plus, la relation <<include>> permet d'indiquer que la *Scene 1* est au niveau hiérarchique inférieur à celui de l'*acte 1*. À droite il s'agit d'un diagramme de classe dans lequel *Acte 1* est une **Class** stéréotypée avec <<**LearningPhase**>>, tandis que *Scene 1* est une **Operation** stéréotypée avec <<**LearningPhase**>>. Pour finir, en bas se trouve un diagramme d'activité dans lequel l'*acte 1* est un **SubActivityState** afin de pouvoir contenir *Scene 1* et *Scene 2* comme **ActionState** au même titre que l'*Acte 2*.

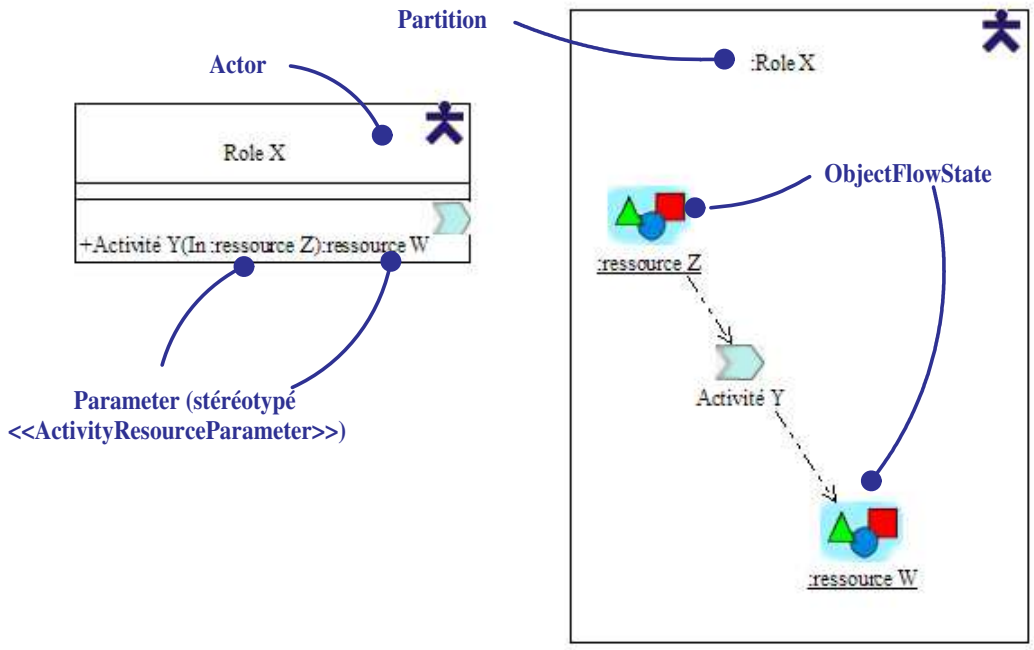

FIG. 8.4 – Exemples d'applications des mêmes stéréotypes <<**Resource**>> et <<**Role**>> sur des méta-classes UML différentes.

La figure 8.4 présente également différentes utilisations possibles d'un même stéréotype avec des méta-classe UML différentes. Dans la partie gauche de la figure est présenté un extrait de diagramme de classe et à droite un extrait de diagramme d'activité. Le stéréotype <<**Role**>> de *Role X* est défini à gauche sur la méta-classe **Actor** tandis qu'à droite il est défini sur une **Partition**. L'*Activité Y* est également stéréotypé <<**Activity**>> sur une **Operation** à gauche et sur **ActionState** à droite mais il n'y a pas d'annotation dans la figure car ces éléments ont déjà été illustrés dans la figure précédente. *ressource Z* est défini comme un **Classifier** stéréotypé <<**Resource**>> mais il n'est pas visible dans le diagramme de classe de gauche. Par contre, *ressource Z* est passé en paramètre à *Activity Y* en mode IN. De même, *ressource W* est un **Classifier** stéréotypé <<**Resource**>> qui est spécifié comme paramètre de retour de *Activité Y* (mode OUT). Dans le diagramme d'activité, l'utilisation *proxy* des ressources par les activités est plus intuitive : *ressource Z* est un **ObjectFlowState** stéréotypé <<**Resource**>> faisant référence au **Classifier** *ressource Z* défini dans un autre diagramme. La transition entre *ressource Z* et *Activité Y* pointant vers l'activité indique que la ressource est utilisée. De la même façon *ressource W* apparaît dans sa version *proxy* mais, cette fois-ci, est pointé par la transition : cela signifie que la ressource est produite par l'activité. Si une même ressource est utilisée en entrée et en sortie d'une activité (IN/OUT), cela signifie qu'elle est modifiée.

8.3.3 Types de valeurs marquées

Les attributs dans le paquetage `CPM_Extensions` sont représentés par des valeurs marquées. Toutes nos définitions de valeurs marquées ont la multiplicité 1 (elles ne s'appliquent qu'une seule fois par élément de modélisation concerné).

Le tableau 8.3 donne un résumé complet de toutes les définitions de valeurs marquées du profil CPM : définition de la marque, type de la valeur, nom du stéréotype sur laquelle elle s'applique. La sémantique de ces éléments est celle définie dans le chapitre précédent sur le méta-modèle CPM.

Définition de valeur marquée	**Type de la valeur**	**Sur stéréotype**
name	String	ExternalDescription
path	String	ExternalDescription
type	String	ExternalDescription
guidanceKind	String	Guidance
relationKind	String	Relation
kind	PrecedendenceKind	Precedes
staticKind	String	StaticPBLElement
dynamicKind	String	DynamicPBLElement
resourceKind	String	Resource
constraintKind	String	PBLConstraint
phaseKind	String	LearningPhase
structureKind	StructureKind	ActivityStructure
activityKind	String	Activity
roleKind	String	Role
rightKind	String	UseRights

TAB. 8.3 – *Les valeurs marquées du profil* CPM

8.3.4 Contraintes

L'ensemble des contraintes présentées dans le méta-modèle CPM s'applique également aux éléments d'extension du profil avec toutefois quelques changements dans l'écriture avec le langage OCL. En effet, il n'est pas possible d'utiliser le nom d'un stéréotype dans la partie *contexte* d'une contrainte. Toutefois, nous considérons cette notation valide comme raccourci d'écriture. Voici un exemple :

```
context ActivityConcept inv:
    X
```

Cette contrainte est un raccourci d'écriture pour :

```
context Operation inv:
    (self.stereotype.name= "LearningPhase" or
     self.stereotype.name= "ActivityStructure" or
     self.stereotype.name= "Activity" or
     self.stereotype.name= "CollaborativeActivity") implies X
```

Les associations du méta-modèle CPM ne devraient plus également apparaître dans la spécification des contraintes mais devraient être remplacées par les associations UML correspondantes (voir 8.1).

8.4 Notation

Alors que la précédente section était consacrée aux éléments d'extensions utilisés dans le profil CPM nous présentons dans cette nouvelle section l'ensemble des notations UML utilisées pour représenter la notation du profil CPM.

8.4.1 Suggestion d'icônes

La colonne *Icône* dans le tableau des stéréotypes (Tableau 8.2) propose des représentations alternatives pour la plupart des classes du méta-modèle d'UML pour lesquels nous avons proposé des stéréotypes. Ces différentes notations peuvent être utilisées pendant la modélisation de PBL coopératives pour représenter des activités, des rôles, ressources, etc. Dans la suite de ce document, nous utiliserons ces représentations graphiques dans nos diagrammes.

8.4.2 Diagrammes

Le tableau 8.4 présente l'ensemble des diagrammes UML 1.X et note pour chacun d'eux s'il est :

- *non utilisé* : certains éléments sémantiques d'UML associés à l'utilisation de ce diagramme ont été exclus dans le profil CPM, ce qui rend les notations inutilisables,
- *utilisé* : certaines notations associées à ce diagramme sont particulièrement utiles pour la modélisation de nos situations-problèmes coopératives,
- *utilisable* : les éléments de notation du diagramme ne sont pas exclus mais aucune correspondance ou sémantique n'a été définie en relation avec les concepts CPM.

Dans les sous-sections suivantes nous présentons pour chaque diagramme UML utilisé ou utilisable des exemples d'usages adaptés à nos besoins de modélisation de situations-problèmes coopératives. Nous faisons référence à de nombreuses figures du chapitre suivant, illustrant l'application du langage CPM au cas SMASH. Ces exemples de notation ne sont que des illustrations d'usage et ne sont en aucun cas des obligations. UML est, à l'origine, un langage avec de multiples possibilités de notation. Toutefois, les exemples que nous proposons garantissent, selon notre point de vue, la meilleure utilisation des concepts du langage CPM.

Diagrammes de structure	Diagramme de classe	utilisé
	Diagramme d'objets	utilisable
	Diagramme de composants	non utilisé
	Diagramme de déploiement	non utilisé
Diagrammes de comportement	Diagramme de cas d'utilisation	utilisé
	Diagramme de séquence	utilisable
	Diagramme de collaboration	utilisable
	Diagramme d'états	utilisé
	Diagramme d'activités	utilisé

TAB. 8.4 – *Diagrammes UML dans la notation du profil* CPM

8.4.2.1 Les diagrammes de classe

Les diagrammes de classe permettent la représentation des aspects suivants de la conception de PBL coopératives :

Définition de rôles :

- décomposition des rôles en sous-rôles (associations d'héritage/spécialisation) : voir figure 9.15 page 216 ;
- précision du type de chaque rôle, apprenant ou encore tuteur (avec la valeur marquée *roleKind*) (Figure 9.26 page 229) ;
- définition de regroupement de rôles pour manipuler plusieurs rôles comme un seul élément dans les modèles (associations d'héritage/spécialisation) : Figure 9.26 page 229 ;
- définition de nouveaux rôles joués localement dans une scène par exemple : on peut utiliser le concept **Relation** pour relier les rôles globaux et les rôles locaux (voir figure 9.33 page 236).

Définition et attribution de ressources aux différents rôles :

- de manière détaillée (Figure 9.27 page 230),
- en décrivant le contenu d'un paquetage : (Figure 9.19 page 220).

Description de l'organisation des activités ou des phases : la décomposition des activités peut être décrite sur un niveau ; la relation **Precedes** permet d'indiquer le type de précédance entre les deux activités reliées (Figure 9.14 page 215).

Analyse et conception individuelle des activités : chaque activité peut faire l'objet d'une description (*via* un diagramme de classe) individuelle en termes d'objectifs, de critères de réussite, de ressources utilisées, modifiées, créées, etc. (Figure 9.24 page 226). Cette représentation est empruntée à celle des diagrammes de processus [Eri00, deC02] : les ressources utilisées sont placées à gauche (*input objects*), les ressources produites à droite (*output objects*), les ressources/concepts participant à l'activité en dessous (*supplying objects*), les ressources/concepts contrôlant l'activité au dessus (*controlling objects*).

Spécification de modèles d'expression initiale des besoins : le diagramme de classe convient pour la description des modèles amont de la conception pour lesquels l'objectif concerne davantage l'élaboration de la PBL (c'est-à-dire des aspects statiques de la PBL : objectifs d'apprentissage, obstacle,

ressources initiales, etc.) que la mise au point d'un scénario pédagogique détaillé (relevant davantage d'aspects dynamiques : enchaînement des activités pédagogiques et des activités de tutorat, échanges de ressources, etc.). Des exemples de tels diagrammes sont présentés dans les figures 9.10 page 211, 9.11 page 212, 9.12 page 213, 9.20 page 222. Pour ces diagrammes, les différents concepts peuvent être reliés en utilisant l'élément **Relation** défini dans le langage CPM ou bien en utilisant les relations d'UML (associations, aggrégations, dépendances, composition) ; ce choix dépend de l'équipe de conception et du niveau de formalisation désiré (selon l'usage futur que l'on désire faire du modèle).

8.4.2.2 Les diagrammes de cas d'utilisation

Définition de la structure du scénario ou d'une sous-partie : les activités peuvent être décrites dans les diagrammes de cas d'utilisation (Figure 9.16 page 216). A l'exception de la relation d'inclusion entre deux activités (Figure 9.17 page 217), il n'est pas possible de décrire l'enchaînement dynamique des activités.

Description des relations entre les rôles et les activités qu'ils réalisent : la réalisation des activités par certains rôles peut être décrite dans les diagrammes de cas d'utilisation par le biais des associations stéréotypées <<performs>> pour signifier la participation active à la réalisation de l'activité ou bien <<assists>> pour une participation passive (Figure 9.17 page 217).

8.4.2.3 Les diagrammes d'états

Les diagrammes d'états permettent d'illustrer le comportement des éléments de modélisation CPM. La sémantique du méta-modèle CPM précise que seuls les éléments **DynamicPBLElement** peuvent avoir leur comportement précisé sous la forme d'un diagramme décrivant les états qu'ils traversent pendant leur cycle de vie.

Ceci peut permettre de décrire, entre autres, les différents états d'une ressource (Figure 9.28 page 231) ; ces états pouvant alors être réutilisés dans d'autres modèles afin de détailler plus précisément l'état des ressources au moment de leur utilisation, création ou modification par les activités. Ces différents états peuvent également faire l'objet d'une utilisation au niveau des contraintes (**PBLConstraint**, **Precondition**, **Postcondition**).

8.4.2.4 Les diagrammes d'activités

Définition de la structure du scénario sur un niveau : un diagramme d'activités permet de décrire l'organisation des activités pédagogiques sur un seul niveau uniquement ; par exemple le niveau des actes (Figure 9.13 page 214 ou Figure 9.29 page 232). Les transitions entre les phases ou les activités indiquent le sens de leur réalisation dans le temps. Ces phases concernent tous les rôles impliqués dans la PBL. Ainsi, aucun rôle n'apparaît dans le diagramme. Pour définir un niveau hiérarchique inférieur (par exemple les scènes composant un acte), un nouveau diagramme d'activités doit être utilisé (Figure 9.30 page 233)

Définition d'activités et attribution à un rôle : un diagramme d'activités permet également de décrire les activités et les structures d'activités (voir la sémantique donnée en 7.3.4.3) dans le contexte du rôle qui les réalise. Il suffit alors de décrire ces activités dans une *partition*[117] référençant le rôle (Figure 9.21 page 223).

Décrire le contenu d'une structure d'activité : voir Figure 9.22 page 224.

Décrire l'utilisation des ressources par les activités : les ressources utilisées, crées ou modifiées par les activités peuvent être décrites dans les diagrammes d'activités sous la forme *proxy* d'un **ObjectFlowState** (Figure 9.23 page 225, Figure 9.22 page 224). Une ressource pointant vers une activité doit être interprétée comme l'utilisation d'une ressource ; à l'inverse, une activité pointant vers une ressource signifie la création de la ressource ; la même ressource apparaissant en entrée et en sortie d'une activité signifie qu'elle a été modifiée par l'activité. Il est possible de décrire les différents états de la ressource lors de sa description (Figure 9.34 page 237).

Décrire le parallélisme entre des activités : il est possible de décrire le parallélisme entre plusieurs activités réalisées par le même rôle (simultanéité, voir Figure 9.22 page 224) comme la réalisation en parallèle d'activités réalisées par différents rôles (Figure 9.31 page 234) par le biais des barres horizontales *fork* (début de parallélisme) et *join* (fin de parallélisme).

Décrire différentes possibilités dans la réalisation des activités ou de phases : il est possible d'utiliser les branches conditionnelles des diagrammes d'activités pour décrire des choix ou conditions portant sur l'exécution dynamique des activités (Figure 9.31 page 234).

8.4.2.5 Les diagrammes d'objets, de séquence, de collaboration

Les concepts liés aux diagrammes d'objets, de séquence et de collaboration permettent de décrire de nombreux aspects de tout système en manipulant des instances ou objets.

Les concepts concernés par ces diagrammes n'ont pas fait l'objet de spécialisation précise dans le profil CPM, ni de modification de leur sémantique. Leurs usages restent donc similaires à ceux décrits dans la spécification d'UML 1.5.

Les diagrammes d'objets permettent de spécifier les instances particulières des concepts utilisés dans les modèles héritant du type **Classifier**. Il est donc possible de décrire au niveau M0 des instances de **Role**[118], de **Resource** ainsi que de tous les **StaticPBLElement**, et les **DynamicPBLElement**.

Les diagrammes de séquence peuvent être utilisés pour illustrer des *patrons* d'interactions entre des instances des éléments de modélisation CPM.

[117] Encore appelée *swimlane* ou *couloir*.
[118] Car la méta-classe **Actor** spécialise **Classifier**.

Chapitre 9

Vérification et mise à l'essai du langage CPM

Sommaire

Dans ce chapitre nous présentons deux travaux d'expérimentation pour le langage CPM. La première section (9.1) vise à fournir un environnement auteur, sous la forme d'un outil logiciel, supportant le langage CPM et permettant l'élaboration de modèles CPM. La deuxième section (9.2) concerne une mise à l'essai pratique de l'élaboration de modèles CPM expérimentée dans le cadre de la situation-problème SMASH (présentée en 2.4).

9.1 Outil pour le langage CPM

Il existe une pléthore d'Ateliers de Génie Logiciel (AGL) compatibles avec le langage UML.[119]

Le cahier des charges relatif au choix d'un AGL UML pour valider nos travaux se restreint à ces deux pré-requis :

1. proposer la possibilité de modéliser les différents diagrammes privilégiés dans la notation de notre profil CPM : diagrammes de classes, d'états, d'activités, de cas d'utilisation ;

[119]Une liste exhaustive des AGL UML les plus aboutis et les plus utilisés est disponible sur http ://www.objectsbydesign.com/tools/umltools_byCompany.html

2. permettre d'étendre le méta-modèle d'UML (au minimum les stéréotypes et valeurs marquées) et permettre d'utiliser ces extensions dans les modèles.

L'outil `Objecteering©`[120] de Softeam est l'AGL qui convenait au mieux à nos exigences. En effet, Softeam[121] a dirigé le groupe de travail ayant pour but de consolider la notion de profil UML dans UML 1.4. Leur outil a donc pu supporter toutes les propriétés et caractéristiques des profils UML : définition de stéréotypes, de valeurs marquées, d'icônes, d'héritage entre profils, etc.

9.1.1 Présentation d'Objecteering

9.1.1.1 Présentation générale de la suite Objecteering

Objecteering/UML est un atelier objet proposant le support de l'approche MDA[122]. Il est structuré autour de deux outils principaux :

- Objecteering/UML Modeler : l'atelier de construction de modèles, paramétrable par les profils UML,
- Objecteering/UML Profile Builder : l'atelier de construction de profils UML.

Objecteering/UML fournit ainsi une séparation claire entre, d'une part, la construction d'un modèle métier indépendant des plates-formes technologiques et, d'autre part, l'application sur ce modèle d'un savoir-faire technique spécifique d'une plate-forme cible. L'association de ces deux points de vue complémentaires se fait à l'aide des profils UML.

La figure 9.1 est une capture d'écran de l'interface proposée par l'outil Objecteering Modeler. L'interface se décompose en quatre grande parties :

1. L'explorateur du modèle permet de visualiser l'organisation et la structuration hiérarchique de tous les éléments de modélisation créés dans le modèle.
2. L'éditeur de propriétés détaille les propriétés de l'élément de modélisation sélectionné dans l'explorateur du modèle ou dans l'éditeur de diagramme. Deux onglets importants sont proposés : *diagrams* (où sont recensés les diagrammes créés et associés à l'élément sélectionné), et *items* (où apparaissent les relations, les contraintes, et les commentaires sur l'élément sélectionné).
3. La console fournit une trace de tous les messages de type *feedback* à l'utilisateur sur le fonctionnement de l'outil : informations sur la création de diagrammes, avertissements, messages d'erreur, etc.
4. L'éditeur graphique propose, selon le type de diagramme UML créé, une interface adaptée (boutons) pour faciliter la création des diagrammes.

[120] http ://www.objecteering.com/
[121] http ://www.softeam.fr/
[122] Une présentation *commerciale* de la suite logicielle est disponible à l'adresse http ://www.objecteering.com/pdf/datasheets/fr/objecteering_uml.pdf

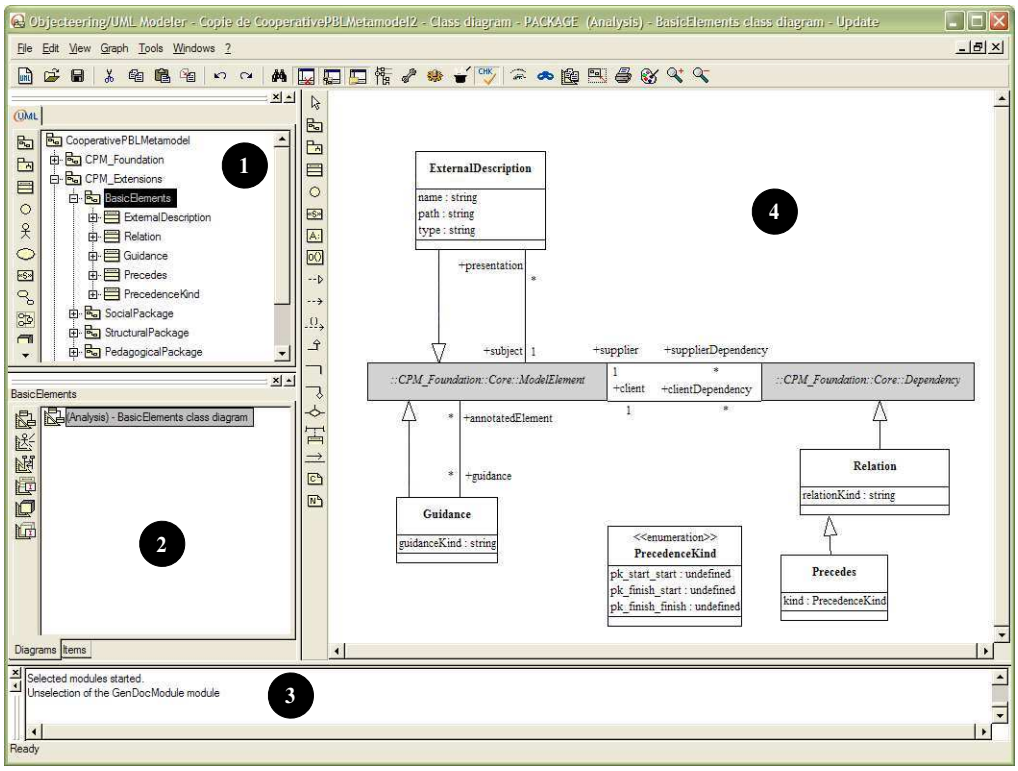

FIG. 9.1 – L'interface d'*Objecteering/UML Modeler*.

9.1.1.2 L'outil profile Builder

Objecteering/UML Profile Builder[123] permet de construire les profils UML destinés à implémenter les outils de transformation automatique de modèles, de génération de code et d'automatisation du processus de développement.

Objecteering/UML Profile Builder fournit un explorateur permettant un accès direct au méta-modèle d'UML (tel qu'il est implémenté dans l'outil). Ceci permet de faciliter la création des profils en structurant les stéréotypes et les valeurs marquées par rapport aux méta-classes UML auxquelles il font référence. L'outil propose également le langage propriétaire J, à la syntaxe similaire de Java, qui permet la construction de requêtes sur le méta-modèle ainsi que l'écriture de règles de transformation de modèles. De nouvelles extensions (valeurs marquées, stéréotypes) et de nouvelles rubriques de textes destinées à particulariser les modèles et à guider les transformations, peuvent être ajoutées au niveau du méta-modèle.

Le langage J intégré est un langage objet, dynamique et interprété, dédié à la navigation et à la construction de requêtes sur le méta-modèle UML. J apporte les notions de contexte et de diffusion de messages sur un ensemble d'objets, mais aussi l'héritage entre profils UML. L'environnement de développement J est doté d'un interpréteur qui permet l'insertion dynamique de règles de transformation et leur

[123]Une présentation *commerciale* du Profile Builder est disponible à l'adresse http ://www.objecteering.com/pdf/datasheets/fr/objecteering_uml_profile_builder.pdf

mise au point sur un projet de test, avant leur diffusion sur un projet de développement.

Une librairie de composants J pré-existants permet d'implémenter des fonctions standards de l'atelier :
- création et placement automatique de diagrammes,
- création de boites de dialogue, de menus,
- adaptation des boites de propriétés,
- maintien dynamique de la cohérence modèle/code,
- lancement d'éditeurs spécialisés,
- appel d'environnements externes, etc.

9.1.1.3 Exemples de profils créés avec le Profile Builder

L'outil Objecteering Profile Builder a déjà permis d'implémenter et valider divers profils UML :
- le profil SPEM dédié à la modélisation de processus d'Ingénierie Logicielle (mentionné dans [OMG02]) ;
- les profils EJB et CCM du projet RNTL Accord [ACC] dédié à l'Assemblage de Composants par Contrats en environnement Ouvert et Réparti ;
- le profil SIG dédié à l'aide à la conception de Systèmes d'Informations Géographique [Mir04].

9.1.2 Implémentation du profil CPM

Cette section est consacrée à l'implémentation du profil CPM théorique du chapitre précédent avec l'outil Objecteering Profile Builder.

9.1.2.1 Adaptation du profil CPM au méta-modèle UML d'Objecteering

Le méta-modèle d'UML intégré dans l'atelier Objecteering diffère sur certains points du méta-modèle d'UML décrit par l'OMG. Des tables de correspondances entre ces deux méta-modèles sont disponibles dans la documentation d'Objecteering.

La figure 9.2 illustre un extrait du méta-modèle d'UML tel qu'il a été implanté sous Objecteering. En comparant cet extrait à celui du méta-modèle d'UML décrit par l'OMG (Figure 5.16), nous remarquons les différences suivantes :
- un seul stéréotype peut être défini par *ModelElement* (plusieurs dans le cas de l'OMG) ;
- une valeur marquée peut avoir des paramètres et doit correspondre à un *TagType* (pour l'OMG les valeurs d'un *Tag* sont définis *via* le méta-attribut *dataValue* sur la méta-classe *TaggedValue*) ;
- *Comment* a été remplacé par l'élément *Note* qui est toujours relative à un *NoteType*.
- *Dependency* est implémentée comme une classe abstraite spécialisée par divers éléments selon le contexte : *UseCaseDependency* dans les diagrammes de cas d'utilisation et *Use* pour les autres diagrammes.

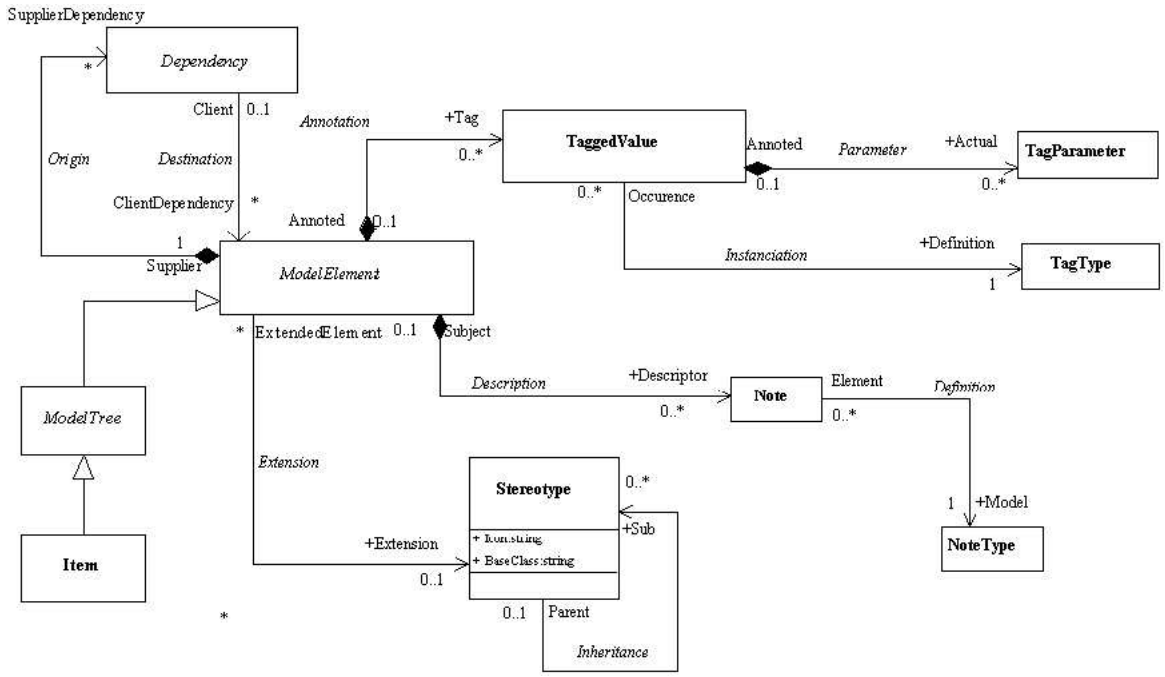

FIG. 9.2 – Extrait du méta-modèle d'UML implanté dans Objecteering concernant les mécanismes d'extension.

Les différentes extensions définies dans notre profil CPM sous Objecteering sont conformes à la manière dont l'outil demande leur spécification :

```
Méta-classe de référence
    Définition de stéréotype
        Définition de type de valeur marquée
        Définition de type de note
    Définition de type de valeur marquée
    Définition de type de note
```

Il faut, dans un premier temps, faire référence à une méta-classe du méta-modèle d'UML puis, dans le contexte de cette référence, définir les nouveaux stéréotypes, types de valeurs marquées et types de notes ; dans le contexte d'un stéréotype il est également possible de créer des types de valeurs marquées et des types de notes qui seront uniquement associés à l'usage du stéréotype. Un extrait de notre implémentation est illustrée dans la figure 9.3. On peut remarquer, par exemple, la référence à la méta-classe **Actor**, puis la définition du stéréotype « **Role** » et, ensuite, la définition du type de valeur marquée *roleKind*.

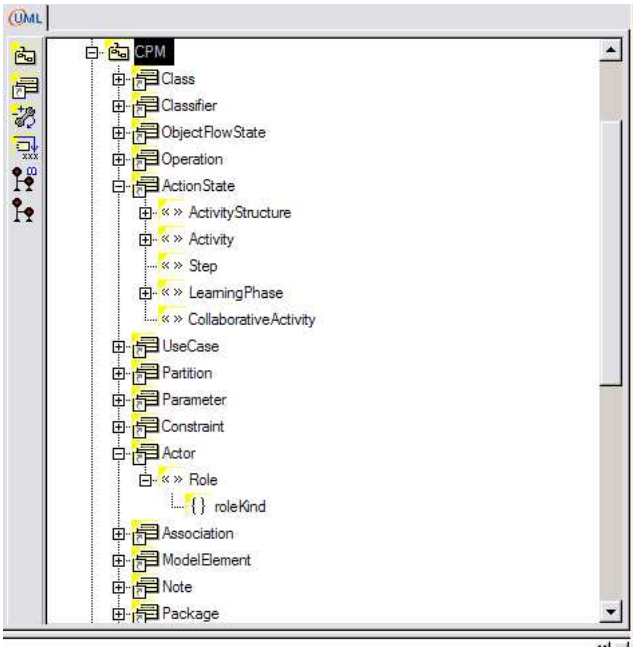

FIG. 9.3 – Extrait du profil CPM implanté dans Objecteering Profile Builder.

9.1.2.2 Implémentation du profil CPM en module Objecteering

Le profil CPM est implémenté sous la forme d'un module Objecteering (fichier : *CPM.prof*). Un *module* regroupe différents profils UML, propose des commandes (qui seront accessibles dans le *Modeler*), ainsi que des valeurs par défaut pour les paramètres (s'ils existent). Le module correspond à un paquetage fonctionnel et cohérent vis-à-vis de l'utilisateur final. Notre module CPM contient deux profils :

- **CPM** : le profil contenant nos mécanismes d'extensions d'UML constituant le langage CPM ;
- **CPM_Code** : le profil contenant l'ensemble des méthodes en langage J.

Le module CPM est utilisable avec la version gratuite d'Objecteering Modeler.

9.1.2.3 Exploitation des possibilités du langage J

Nous avons défini dans le profil **CPM_Code** un ensemble de méthodes J (code en annexe B) permettant d'aider les utilisateurs à utiliser l'AGL Objecteering Modeler avec notre langage CPM.

L'amélioration de l'interface des AGL permet d'aider le travail de modélisation. Les différents travaux de recherche autour de l'AGL UML *ArgoUML*[124] revendiquent l'importance des fonctionnalités d'aide à la décision : « *design tools that support designers in decision-making are a promising way to increase designer productivity and the quality of the resulting designs* »[Rob00]).

Il existe deux possibilités pour proposer un environnement de modélisation adéquat avec notre profil CPM :

[124]http ://argouml.tigris.org/

1. Créer un nouvel environnement dédié ; par exemple, le profil SPEM a fait l'objet du développement d'AGL dédiés : Enterprise Architect [125], Iris Suite [126].
2. Personnaliser/adapter un Atelier de Génie Logiciel UML existant.

La deuxième orientation est la nôtre car l'AGL Objecteering propose de nombreuses fonctionnalités (accessibles *via* le langage propriétaire J) pour personnaliser l'AGL. Nous avons alors développé de nombreuses méthodes pour lesquelles des exemples significatifs de la syntaxe du langage et de l'usage que nous en avons fait sont décrits en annexe B. Pendant la modélisation de situations-problèmes, il est alors possible, par le biais de *commandes* accessibles à l'utilisateur par une nouvelle entrée « CPM » dans le menu contextuel), de :

Appliquer un concept CPM : l'ajout de stéréotype lié à notre langage CPM est ainsi plus accessible et plus rapide (Figure 9.4). L'utilisateur choisit un stéréotype dans la liste et celui-ci est appliqué à l'élément de modélisation sélectionné dans le modèle.

FIG. 9.4 – Application d'un concept CPM.

Rechercher des éléments CPM : ceci permet de retrouver facilement et rapidement tous les éléments de modélisation partageant les mêmes critères de recherche (stéréotype, valeur marquée) (voir Figure 9.5). Voici un exemple de l'utilité de cette commande : si vous désirez rechercher tous les éléments de modélisation représentant des *scènes* vous sélectionnez alors dans la première liste le stéréotype **LearningPhase**, puis une seconde fenêtre vous propose une liste de types de valeurs marquées associées à ce stéréotype dans laquelle vous pouvez choisir un élément ou pas (dans notre cas vous choisissez *phaseKind*), puis une troisième fenêtre vous propose d'entrer ou non le nom du paramètre pour la valeur marquée (vous indiquez *scene*), la fenêtre des résultats propose alors tous les éléments de modélisation répondant aux critères de recherche. Il suffit de sélectionner l'un des résultats pour que l'élément de modélisation concerné soit également sélectionné dans le modèle.

[125] http ://www.sparxsystems.com.au/ea.htm
[126] http ://www.osellus.com/products/irispas.html

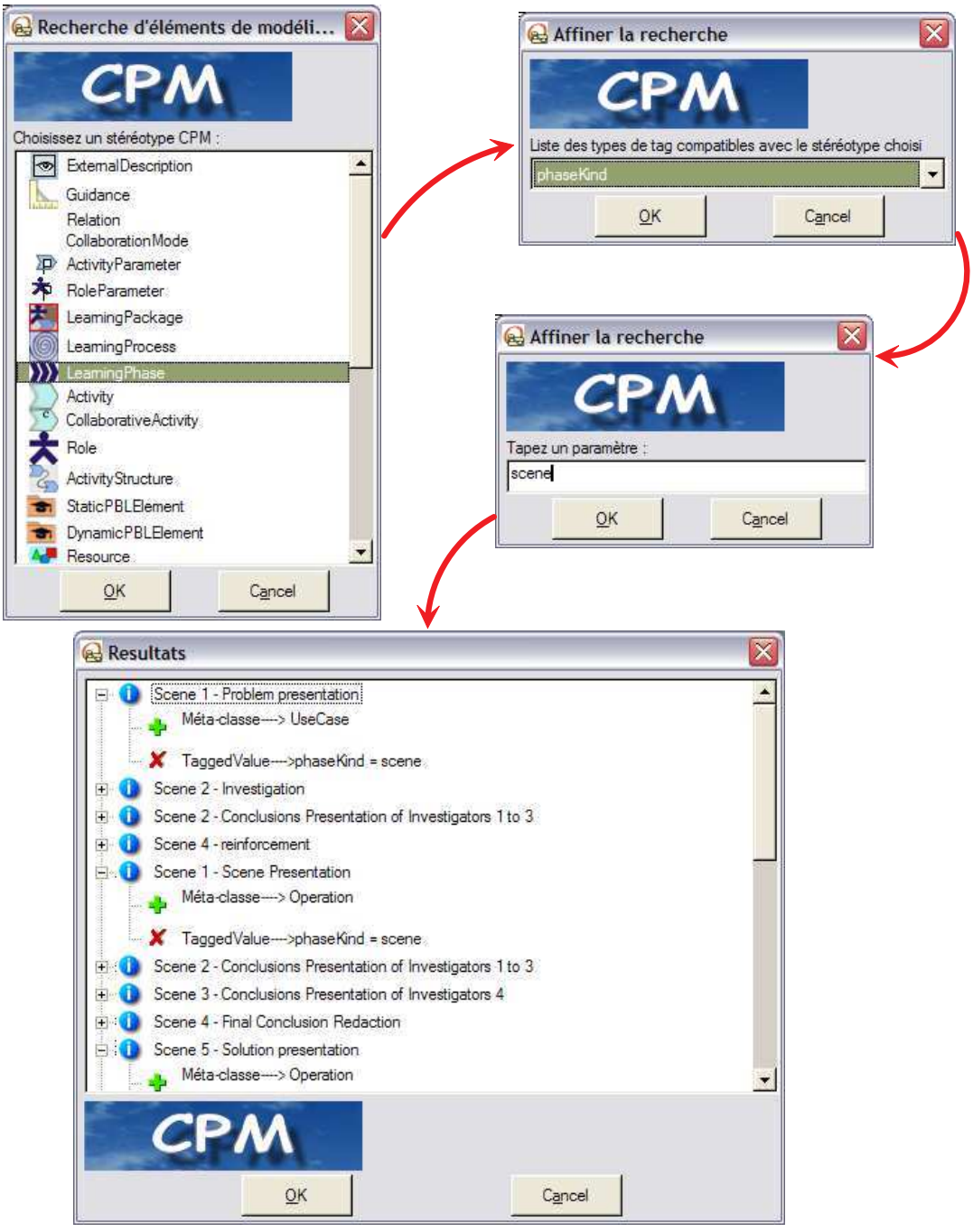

FIG. 9.5 – Recherche d'éléments CPM.

Ajouter des éléments CPM : cette fonction permet d'ajouter des éléments de modélisation correspondant à des concepts du langage CPM sans se préoccuper du type de la méta-classe nécessaire dans le contexte de modélisation (voir Figure 9.6). Par exemple, sous un graphe d'activités une **LearningPhase** sera définie comme un **ActionState** stéréotypé <<**LearningPhase**>> tandis que sous le contexte d'un cas d'utilisation ou d'une classe le même élément sera défini comme une **Operation** stéréotypée <<**LearningPhase**>>.

Cette fonctionnalité est très importante pour les raisons suivantes :

- elle permet d'aider la modélisation des concepts CPM : selon le contexte particulier (l'élément de modélisation sélectionné dans le modèle) seuls les concepts CPM valides conformément au méta-modèle CPM sont proposés ;
- elle permet de simplifier la création des concepts : des boites de dialogue personnalisées sont définies pour chaque concept CPM ;
- elle évite certaines erreurs de construction et ainsi elle permet de vérifier les contraintes du profil CPM ;
- les concepts CPM ajoutés peuvent facilement être utilisés dans les diagrammes par *glisser-déposer*.

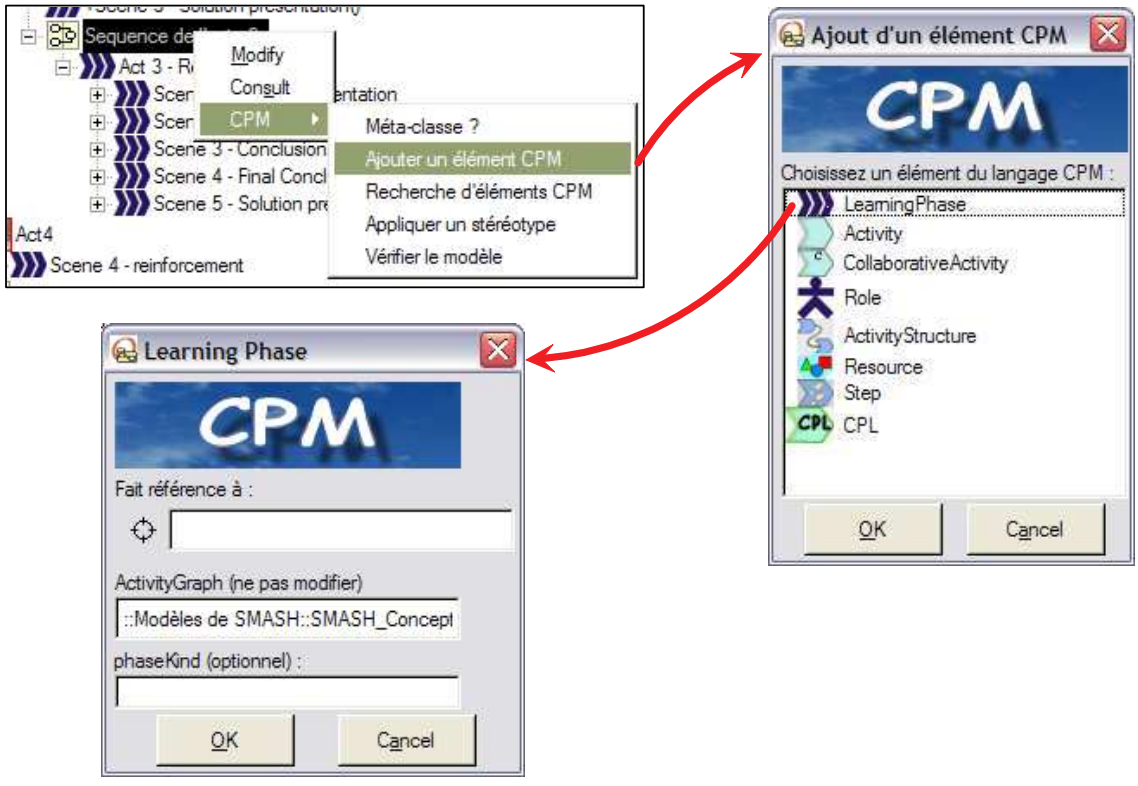

FIG. 9.6 – Ajouts d'éléments CPM.

Modifier directement les propriétés des éléments CPM : nous avons personnalisé l'éditeur de propriétés pour qu'un nouvel onglet « *CPM* » apparaisse (Figure 9.7) et affiche pour chaque élément de modélisation CPM son nom, son stéréotype et ses valeurs marquées. Ses propriétés sont modifiables directement et l'élément concerné est automatiquement mis à jour dans le modèle.

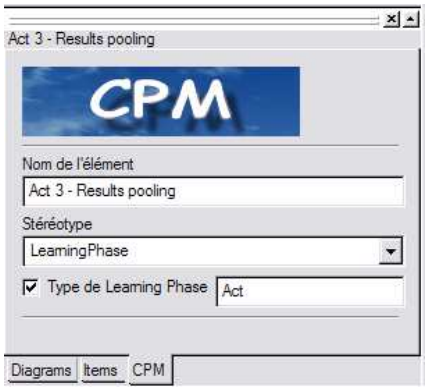

FIG. 9.7 – Propriétés modifiables des éléments CPM.

Vérifier les contraintes de CPM : un appel à cette nouvelle commande permet d'afficher un rapport de vérification des contraintes du langage CPM. Nous avons implémenté des règles en J afin de pouvoir démontrer la faisabilité technique de ces vérifications.

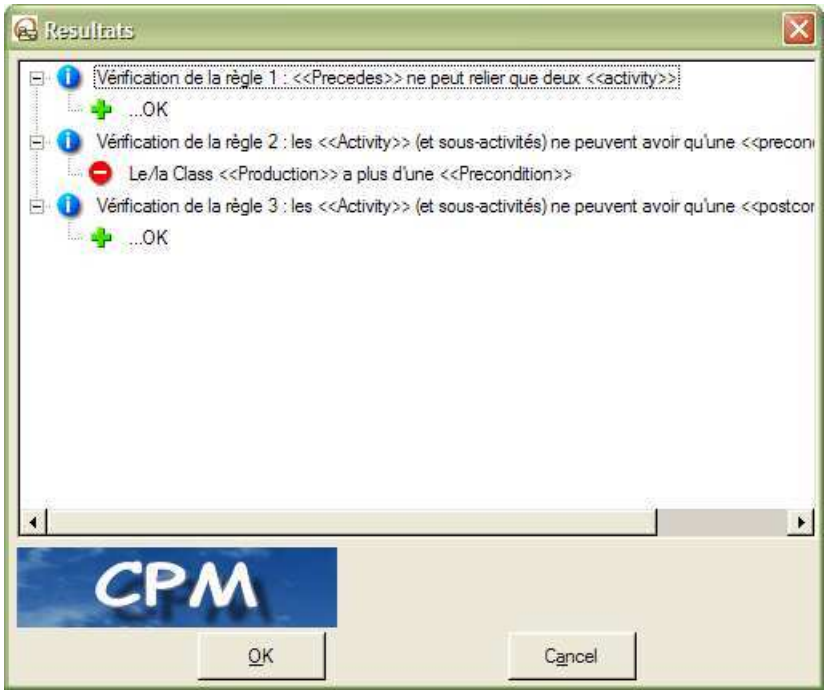

FIG. 9.8 – Vérification sur demande explicite de la validité des modèles produits conformément à la sémantique du langage CPM.

Génération de modèles XML conformes à la spécification IMS-LD [IMS03c] : nous avons initié un travail de transformation de modèles CPM vers des modèles IMS-LD. L'objectif visé est de démontrer un nouvel usage possible des modèles CPM faciliter par l'utilisation des profils UML : celui de modèles amont pour faciliter la spécification de modèles IMS-LD. Toutefois, nous rappelons que l'objectif principal de notre contribution est de fournir un langage de modélisation graphique

pour les concepteurs de PBL coopérative. La projection du langage CPM vers d'autres langages, comme par exemple IMS-LD, n'est pas au cœur de nos objectifs.

Nous avons limité pour le moment cette perspective aux seuls diagrammes d'activités comme source d'information sur nos modèles. L'utilisation d'autres diagrammes, comme par exemple des diagrammes de classes décrivant les objectifs et les pré-requis de la PBL, devront être également utilisés par la suite. Du point de vue du langage-cible IMS-LD, nous nous sommes limités à la génération du niveau A ; toutefois, les niveaux B et C sont envisageables étant donné les correspondances entre CPM et IMS-LD (voir tableau 7.4). En revanche, de nombreuses informations capturées dans les modèles CPM n'ont pas de correspondance avec IMS-LD ; cette perte d'information prouve la richesse sémantique du langage CPM, due principalement à son positionnement plus amont dans le processus de conception.

La figure 9.9 illustre différentes actions permettant d'aboutir à cette projection vers IMS-LD : tout d'abord un diagramme d'activités doit être construit ; puis, un *workproduct*[127] IMS-LD est créé ; ensuite, la génération du modèle IMS-LD peut être exécutée ; le fichier XML généré est alors accessible.

Nous donnons en annexe B le code J des différentes méthodes permettant cette génération.

[127] Un *workproduct* est un élément d'Objecteering rassemblant un ensemble de fonctionnalités de génération ; il permet, entre autres, de gérer des fichiers externes.

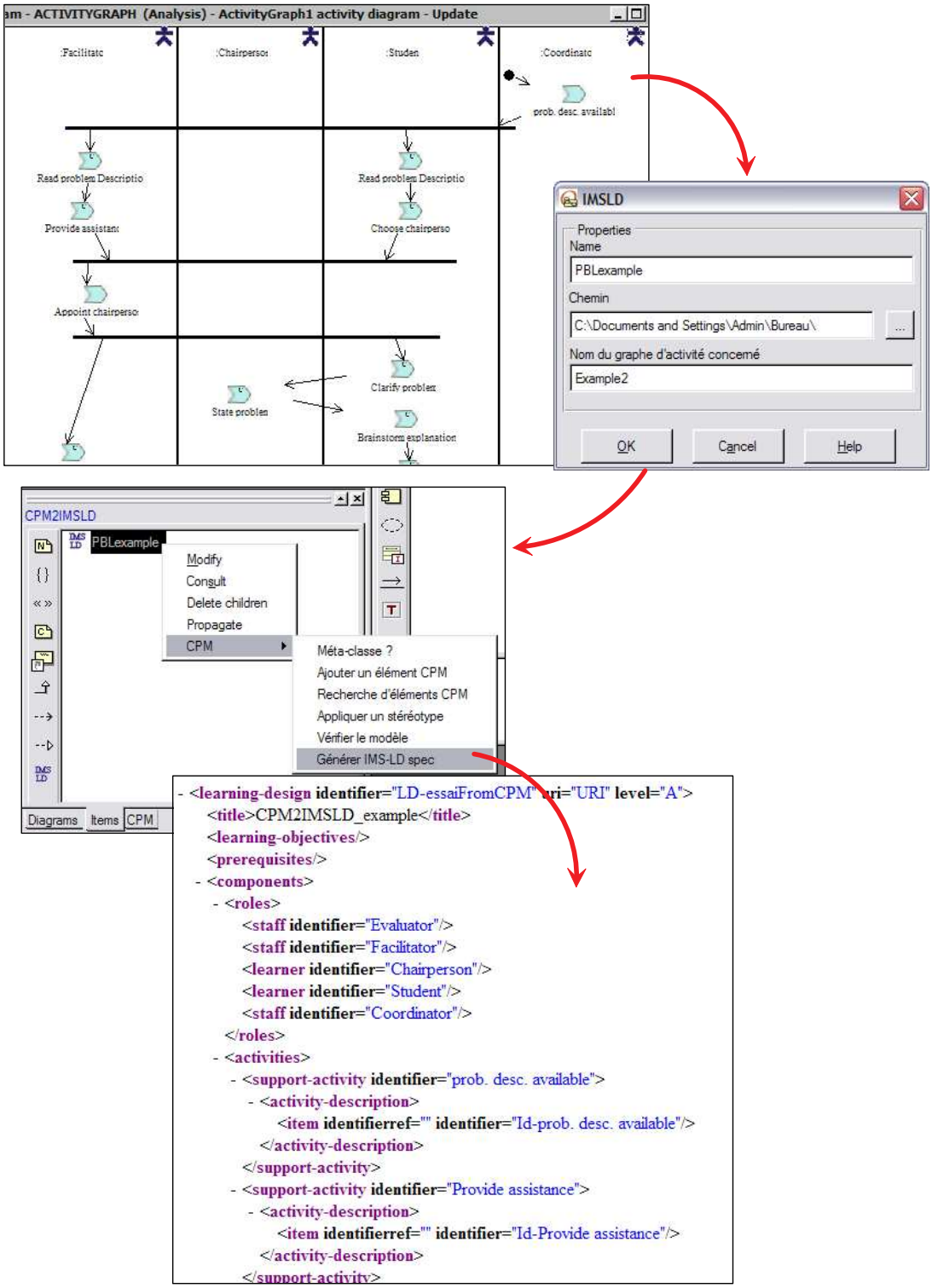

FIG. 9.9 – Génération automatique d'un modèle XML conforme à la spécification d'IMS-LD sur la base d'un diagramme d'activité.

9.2 Application au cas d'étude SMASH

Cette section est consacrée à la présentation de l'application de notre langage CPM au cas d'étude SMASH présenté en § 2.4. Pour des raisons de place, nous avons volontairement sélectionné un sous-ensemble représentatif de diagrammes de différents modèles de SMASH. Nous avons choisi de montrer l'expressivité et la richesse graphique du langage CPM sur l'ensemble du spectre de conception au travers de trois ensembles d'extraits correspondant à :

- un modèle pour l'*expression initiale des besoins* de SMASH (section 9.2.1),
- un modèle pour l'*analyse* de SMASH (section 9.2.2),
- et un modèle pour la *conception* de SMASH (section 9.2.3).

Les diagrammes présentés sont construits à l'aide de l'AGL Objecteering Modeler dans lequel nous avons importé notre module contenant le profil CPM (le langage) et le profil CPM_Code (les commandes, menus, interfaces personnalisées pour aider la conception).

9.2.1 Expression initiale des besoins de SMASH

Les sous-sections suivantes font référence à l'élaboration d'une PBL telle que définie en section 2.3. Elles font également référence à la présentation de SMASH (section 2.4).

9.2.1.1 Définition des objectifs

Énoncé. Les objectifs d'apprentissage de SMASH concernent la sécurité routière (Figure 9.10) et plus précisément la sécurité des piétons et des deux roues dont les enfants du cours moyen à qui s'adressent SMASH sont représentatifs. Ces objectifs d'apprentissage sont issus d'un document officiel que possèdent les professeurs des écoles et dans lequel leur sont listés les matières et sujets à aborder avec les enfants pour un niveau donné.

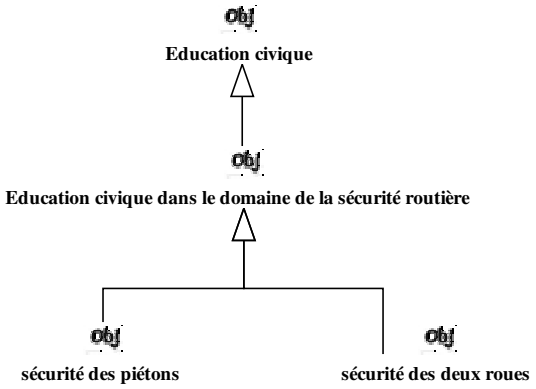

FIG. 9.10 – Les objectifs de SMASH.

Utilisation du langage CPM. Le stéréotype <<**Objective**>> est utilisé. Nous utilisons également l'héritage UML pour raffiner les différents objectifs.

9.2.1.2 Définition de la tâche

Énoncé. SMASH s'adresse aux enfants de 8-10 ans du cycle 3 (ou cours moyen). La tâche qu'il leur est demandé d'accomplir est d'enquêter sur un accident de vélo et de trouver le coupable (Figure 9.11).

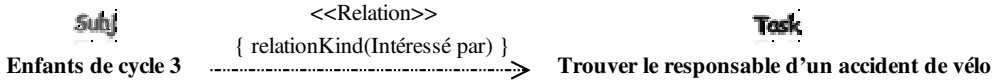

FIG. 9.11 – La tâche globale et les sujets de SMASH.

Utilisation du langage CPM. Dans ce diagramme, nous utilisons le concept de `Relation` pour associer le sujet (stéréotype <<**Subject**>>) et la tâche (stéréotype <<**Task**>>). Dans le prochain diagramme nous utilisons l'association classique proposée par UML. Le choix de l'un ou de l'autre dépend des concepteurs. Les avantages du premier choix sont de formaliser précisément un type de relation et de permettre de le réutiliser plusieurs fois mais aussi de permettre une future automatisation (transformation de modèle par exemple) ; nous abordons ce point dans les perspectives de nos travaux.

9.2.1.3 Définition de l'obstacle

Énoncé. La figure 9.12 présente une définition de l'obstacle de SMASH, ainsi que de son contexte de résolution : les ressources nécessaires et les contraintes. Le sujet précédent (*enfants de cycle 3*) est affiné en *4 groupes de 3 à 4 élèves* qui joueront le rôle d'*enquêteur de police*. Leur tâche consiste à *mener une enquête sur l'accident de vélo* qui s'est déroulé dans un certain contexte (précisé par un autre diagramme du modèle). L'obstacle implicite pour SMASH concerne les *responsabilités partagées entre différentes personnes*. Les apprenants devront surmonter cet obstacle en s'aidant d'un ensemble de ressources (*témoignages récoltés*, *carte du lieu de l'accident*, *fiche bilan du code de la route* et *feuilles de notes* pour écrire leurs analyses). Un système de contraintes assure également le franchissement de l'obstacle : *tous les enquêteurs n'ont pas les mêmes témoignages à disposition* et ces témoignages sont en *nombre limité*.

Utilisation du langage CPM. Nous utilisons les stéréotypes <<**Subject**>>, <<**Task**>>, <<**Obstacle**>>, <<**Resource**>> et <<**PBLStaticElement**>>. Afin de décrire les relations entre les différents éléments de modélisation (*poursuit*, *se réalise dans*, *doit surmonter*, *en s'aidant de*), nous utilisons l'association standard d'UML.

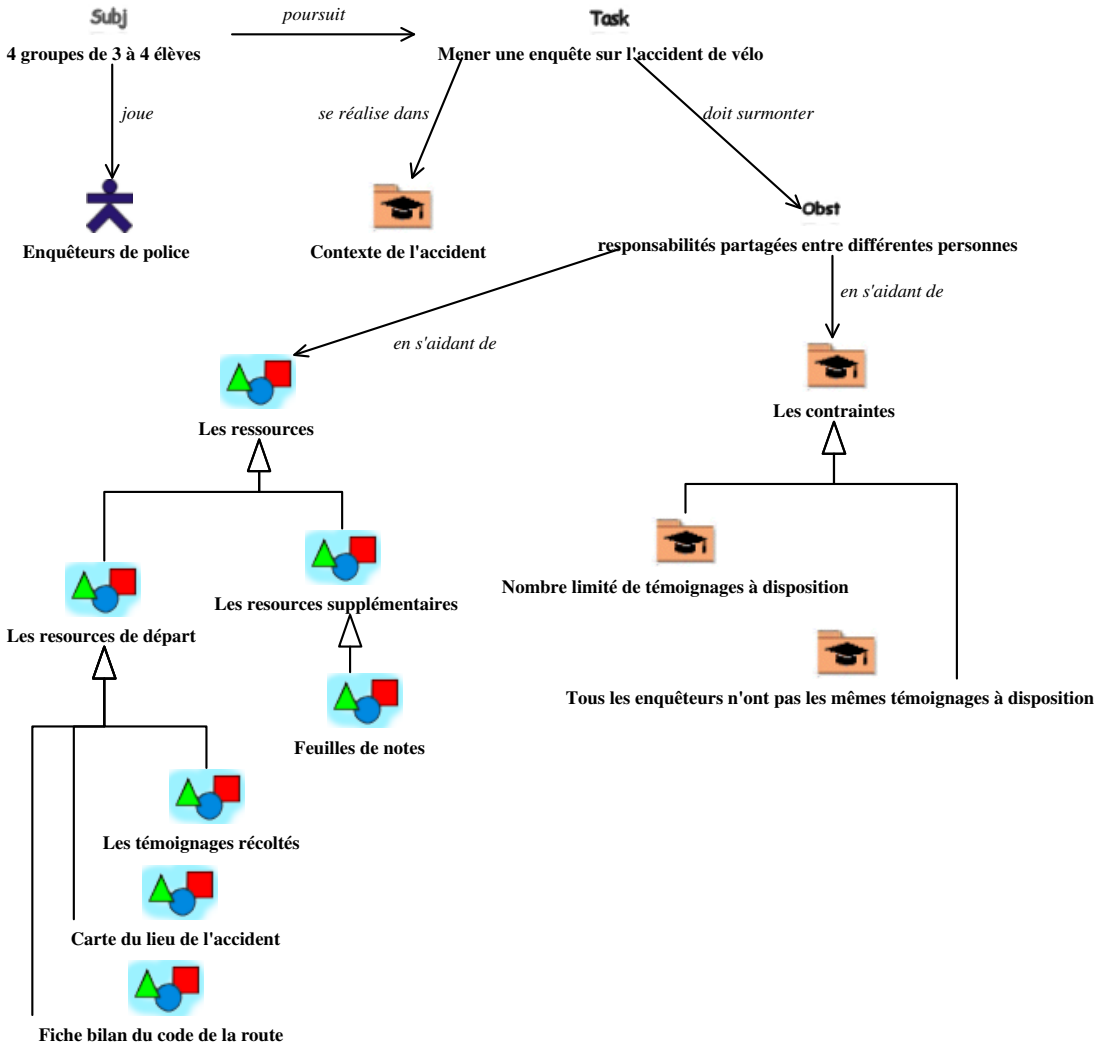

FIG. 9.12 – L'obstacle, les ressources et les contraintes de SMASH.

9.2.1.4 Définition de la tâche globale/déroulement de SMASH

Énoncé. La figure 9.13 présente la tâche globale commune à tous les apprenants pour la résolution de SMASH. Elle est décomposée en quatre phases successives : *présentation du problème*, *production*, *présentation individuelle et production d'une solution commune*, puis *correction*. Ces phases sont décrites en détail au chapitre 2 (section 2.4.2.3).

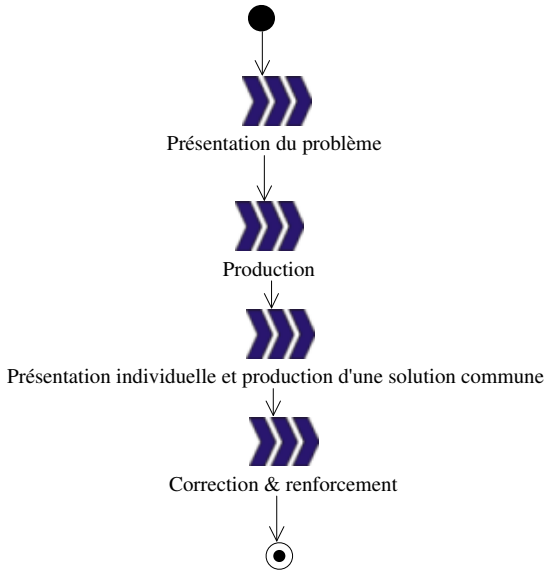

FIG. 9.13 – Déroulement général de SMASH.

Utilisation du langage CPM. Le stéréotype <<**LearningPhase**>> est utilisé pour les quatre instances d'**ActionState** de ce diagramme d'activités ; ils sont également marqués par la valeur marquée *phaseKind* avec la valeur "Act" (les valeurs marquées ne sont pas visibles dans la figure). Les transitions entre **ActionState** indiquent la séquence des actes.

9.2.1.5 Définition des critères de succès vis-à-vis de la tâche globale

Énoncé. La figure 9.14 reprend la tâche globale (cette fois-ci présentée par un diagramme de classe) et met en relation les phases avec les critères de succès identifiés qui permettront à l'apprenant de s'engager dans la réalisation des activités. Par exemple, pour la phase de *production*, les critères de succès qui seront donnés aux apprenants seront *Feuilles de notes correctement remplies*, *QCM remplis avec succès* et *témoins, voitures, vélo et panneaux bien positionnés sur la carte*.

Utilisation du langage CPM. Le stéréotype <<**LearningPhase**>> est utilisé cette fois-ci sur des instances de **Class** ; ces instances sont également marquées par *phaseKind* avec la valeur "*Act*" (les valeurs marquées ne sont pas visibles dans la figure). La relation <<**Precedes**>> relie les actes deux par deux et indique à chaque fois, par le biais d'une valeur marquée, que l'acte suivant ne commence que lorsque le précédent est terminé : *kind=finish_ start* (une seule des va-

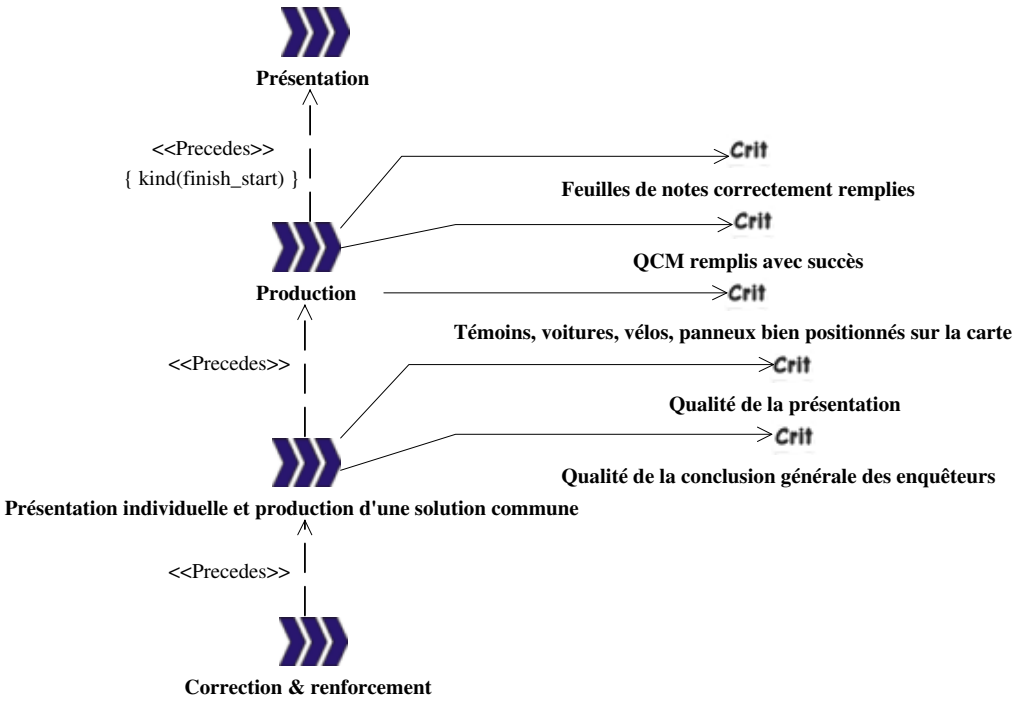

FIG. 9.14 – Déroulement général et critères de succès associés.

leurs marquées définies est montrée dans le diagramme). Les critères de succès sont définis *via* <<**SuccessCriterion**>> et associés aux actes par de simples associations UML.

9.2.1.6 Apprentissage coopératif : vers une division du travail

Énoncé pour la figure 9.15. Les différents rôles d'inspecteurs de police joués par les apprenants se décomposent en 4 groupes. Cette distribution particulière des rôles participe à la division du travail. Ces quatre rôles ont été définis dans la section 2.4.2.1. Le rôle de *chef de la police* joué par le tuteur est également défini.

Utilisation du langage CPM. Le stéréotype <<**Role**>> est utilisé pour définir les différents rôles d'enquêteurs ainsi que celui de *chef de la police*. Le rôle d'*Enquêteurs de police* est également défini afin de représenter l'ensemble des enquêteurs et permettre ainsi leur référencement global dans les autres diagrammes. Tous ces concepts sont marqués avec *roleKind* avec pour certains la valeur "*learner*" et pour le chef de la police la valeur "*tutor*".

La division du travail est également définie pour chaque étape de la tâche globale précédente (Figure 9.13). Nous donnons en exemples l'étape de *présentation* (Figure 9.16) et l'étape de *production* (Figure 9.17). Au travers des ces diagrammes, nous remarquons qu'en phase amont les activités ne sont pas entièrement individuelles : l'utilisation de <<**performs**>> et <<**assists**>> permet de faire intervenir le tuteur et les apprenants dans l'activité (activement avec <<**performs**>>, passivement avec l'autre).

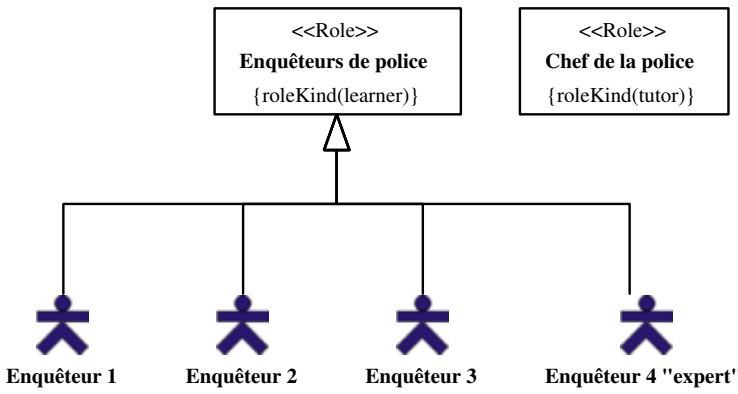

FIG. 9.15 – Les différents type de rôles d'inspecteur de police.

Par la suite, en analyse et en conception, ces activités seront décomposées afin que chaque activité définie ne puisse être réalisée que par un unique rôle.

Énoncé pour la figure 9.16. Le *chef de la police* (le tuteur) réalise le *lancement de la phase de présentation* et *présente le contexte de l'accident, les rôles joués et l'organisation à suivre* ; les apprenants sont impliqués *passivement* dans la réalisation de ces activités mais *activement* dans l'activité consistant à situer cinq lieux susceptibles d'être dangereux sur la carte du village.

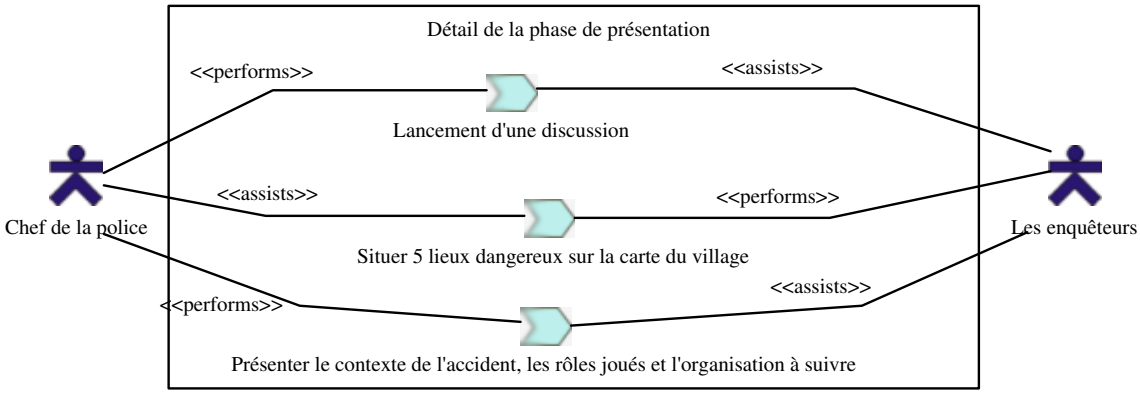

FIG. 9.16 – Découpage en activités de la phase de présentation.

Utilisation du langage CPM. Il s'agit d'un diagramme de cas d'utilisation. *Chef de la police* et *Les enquêteurs* font référence aux rôles définis précédemment. Les activités sont définis comme des **Use-Case** stéréotypé <<**Activity**>>. Les rôles sont reliés aux activités *via* les relations stéréotypées <<**assists**>> ou <<**performs**>>.

Énoncé pour la figure 9.17. Elle décrit l'activité principale réalisée par les enquêteurs 1 à 3 pour la phase de production : l'*analyse de témoignages* ; cette dernière inclut la réalisation de deux autres activités (*Prise de notes* et *réponse à un QCM*). De manière identique, les activités principales d'*étude des cartes* et de *participation à l'analyse des témoignages* sont décrites et rattachées à l'*enquêteur 4*. La *préparation des conclusions* est une activité concernant l'ensemble des enquêteurs : le rôle *les enquêteurs* (défini dans la figure 9.15) est alors utilisé. Pour l'ensemble de ces activités, le *chef de la police* n'a qu'un rôle d'assistant.

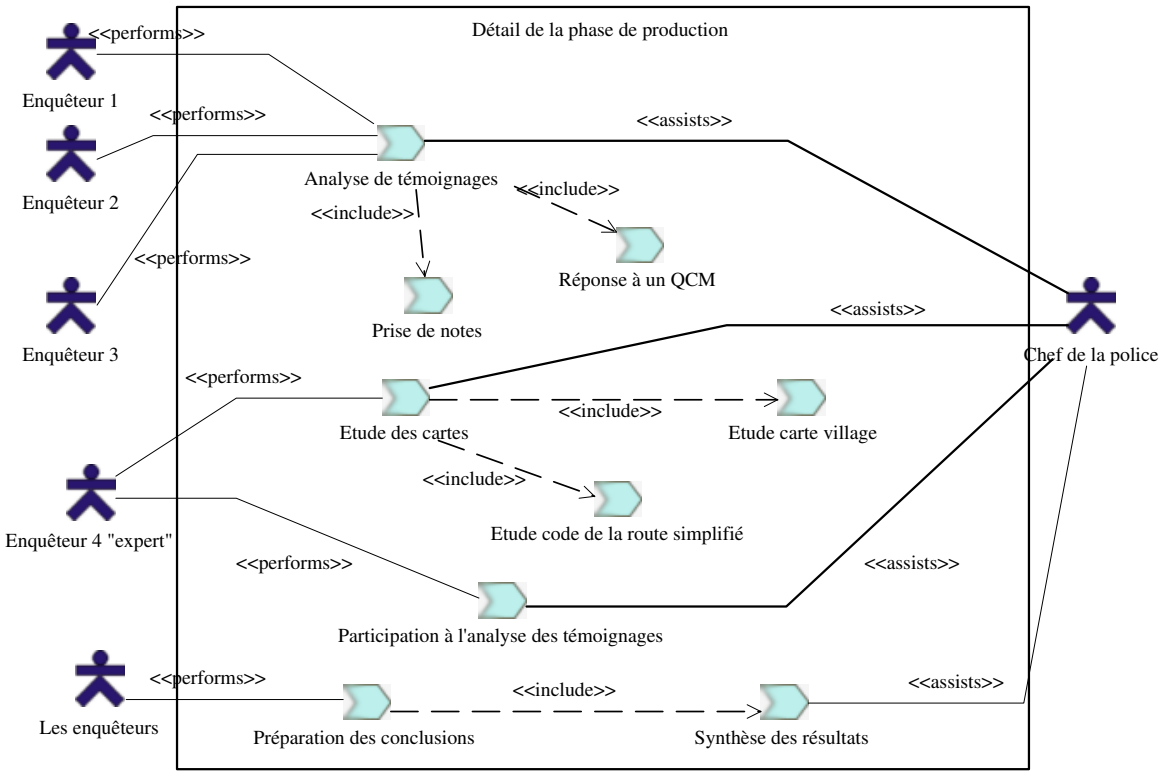

FIG. 9.17 – Découpage en activités de la phase de production.

Utilisation du langage CPM. Identique à celle de la figure précédente à l'exception de l'utilisation de la relation <<*include*>> (déjà définie dans UML 1.X).

Nous pouvons remarquer le manque d'indications sur la séquentialité/parallélisme des activités dans ce type de diagramme. Le diagramme de cas d'utilisation permet uniquement d'aider à la décomposition des phases de la PBL en activités pour les différents rôles. Par le biais des dépendances <<*include*>>, il est possible de décrire les sous-activités.

9.2.2 Analyse de SMASH

Nous montrons dans cette sous-section quelques exemples extraits d'un modèle d'analyse possible pour le cas d'étude SMASH.

9.2.2.1 Analyse détaillée des objectifs

Énoncé. Les objectifs d'apprentissage sont détaillés en analyse (Figure 9.18). Nous définissons les objectifs d'apprentissages en trois sous-catégories *objectifs disciplinaires*, *objectifs pédagogiques* et *objectifs transversaux* (travail réalisé dans [Oud03]).

Utilisation du langage CPM. Nous utilisons la relation de spécialisation d'UML pour définir différentes catégories d'objectifs. Nous avons fait le choix de décrire précisément les différents objectifs au niveau des classes (et non comme des objets) afin qu'ils soient réutilisables dans d'autres diagrammes de classes décrivant, par exemple, pour chaque activité les objectifs visés.

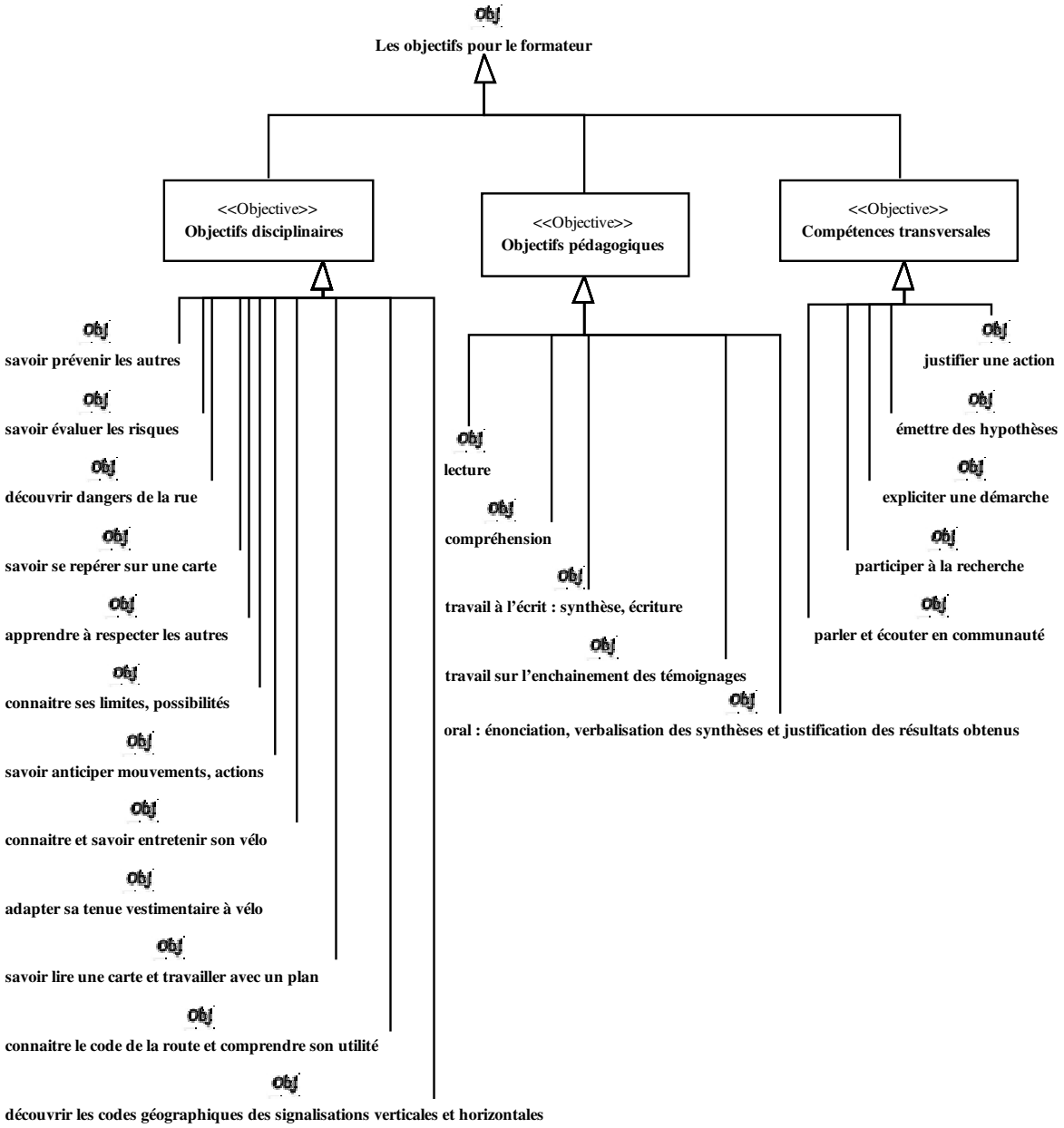

FIG. 9.18 – Les objectifs détaillés de SMASH (repris de [Oud03]).

9.2.2.2 Analyse des ressources

Énoncé. La figure 9.19 montre les différentes ressources distribuées au groupe jouant le rôle d'enquêteur 1 : il a les *témoignages 1, 2 et 3*, ainsi que les *QCM 1, 2 et 3* associés et une *feuille de note*. Un diagramme de paquetage identique peut être décrit pour les ressources des autres enquêteurs.

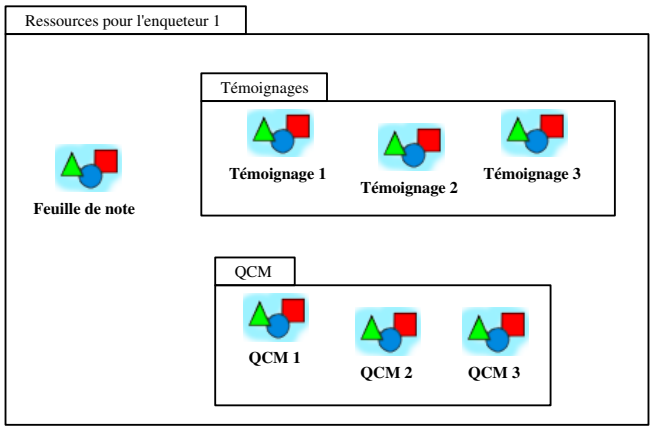

FIG. 9.19 – Les ressources distribuées à l'enquêteur 1.

Utilisation du langage CPM. Les différentes ressources sont modélisées par des **Class** stéréotypées <<**Resource**>>.

Les ressources peuvent également être élaborées lors de la phase d'analyse. Par exemple, un témoignage est conçu pour qu'à sa lecture l'enquêteur concerné aboutisse à des conclusions plus ou moins erronées (volontairement) de l'accident. De même, les QCM associés à chaque témoignage proposent des questions permettant au tuteur de déceler les (mauvaises) conceptions des apprenants et de les guider davantage si besoin. Les feuilles de notes permettent également d'aider le tuteur à suivre la progression des apprenants.

Énoncé de la figure 9.20. Il s'agit d'une analyse des indices et déductions que devront faire les apprenants jouant le rôle d'enquêteur 1 à leur lecture du témoignage 1.

Ce témoignage concerne le témoin appelé *Simon Dupond*. La lecture du témoignage doit amener le lecteur (ici le rôle d'enquêteur 1) à relever les informations suivantes :

- le témoin était devant le passage piéton du magasin,
- une femme se trouvait en face (elle fera l'objet d'un autre témoignage),
- une voiture roulait vite en direction de la route des Marais ; elle n'a pas vu le passage piéton mais a vu le témoin s'engager au dernier moment,
- la voiture s'est brusquement arrêtée dans un grand bruit de crissement de freins,
- une voiture blanche passait en direction du village au même moment sans ralentir,
- cette voiture blanche n'a pu éviter le cycliste.

A partir de ces informations, les apprenants jouant le rôle de l'enquêteur 1 devront déduire que la première voiture, malgré son freinage tardif, s'est arrêtée devant le passage piéton tandis que l'autre

voiture (blanche) n'a pas pu s'arrêter. Afin de vérifier que les informations les plus importantes pour la suite du déroulement de SMASH ont été perçues, les enquêteurs doivent répondre au QCM associé au témoignage (le témoignage 1 et le QCM 1 sont donnés en annexe C).

Utilisation du langage CPM. Nous utilisons le concept de <<**MentalRepresentation**>> pour représenter les conceptions du contexte de l'accident que se feront les apprenants en lisant le témoignage 1. Dans la figure, le QCM associé au témoignage est présenté ainsi que les connaissances qu'il devra vérifier en posant les questions à choix multiples. Dans ce diagramme de classe, la spécification des dépendances est formelle : nous définissons des relations en utilisant conjointement le stéréotype <<**Relation**>> de **Dependency** et la marque *relationKind* dont la valeur correspond au nom de la nouvelle relation. Trois types de relations sont ainsi définies et réutilisées :

- *indice* (pour associer le témoignage aux conceptions qu'il contient),
- *déduction* (pour associer des conceptions explicites à la lecture du témoignage à d'autres qui doivent être déduites),
- et *vérifie* (pour associer le QCM aux conceptions qu'il devra vérifier).

Remarque. La modélisation des connaissances du domaine d'apprentissage et des connaissances des apprenants (comme c'est le cas dans cet exemple) relève également du domaine de l'ingénierie des connaissances (IC). Dans l'exemple de la figure 9.20, nous avons choisi de modéliser les connaissances du domaine et des apprenants par des représentations mentales dans un diagramme de classes. Toutefois, une autre approche de modélisation possible, inspirée de l'IC, serait de décrire dans un diagramme de classes les concepts génériques du domaine (par exemple pour le témoignage 1 : témoin concerné, lieu, véhicule, direction, etc.) et de spécifier dans un diagramme d'objets des instances de ces concepts.

La méthode d'ingénierie pédagogique MISA (présentée en § 4.3.4), différencie le modèle pédagogique (le scénario) du modèle des connaissances. Par contre, les deux modèles sont reliés au travers des objectifs et pré-requis d'apprentissage du scénario qui correspondent à des éléments décrits dans le modèle des connaissances. La modélisation du domaine de l'apprentissage et des connaissances des apprenants est à la limite des usages du langage CPM. Un travail d'expérimentation doit permettre de définir si ce type de modélisation doit être intégrée ou non. Nous discutons de cet aspect dans les perspective de nos travaux.

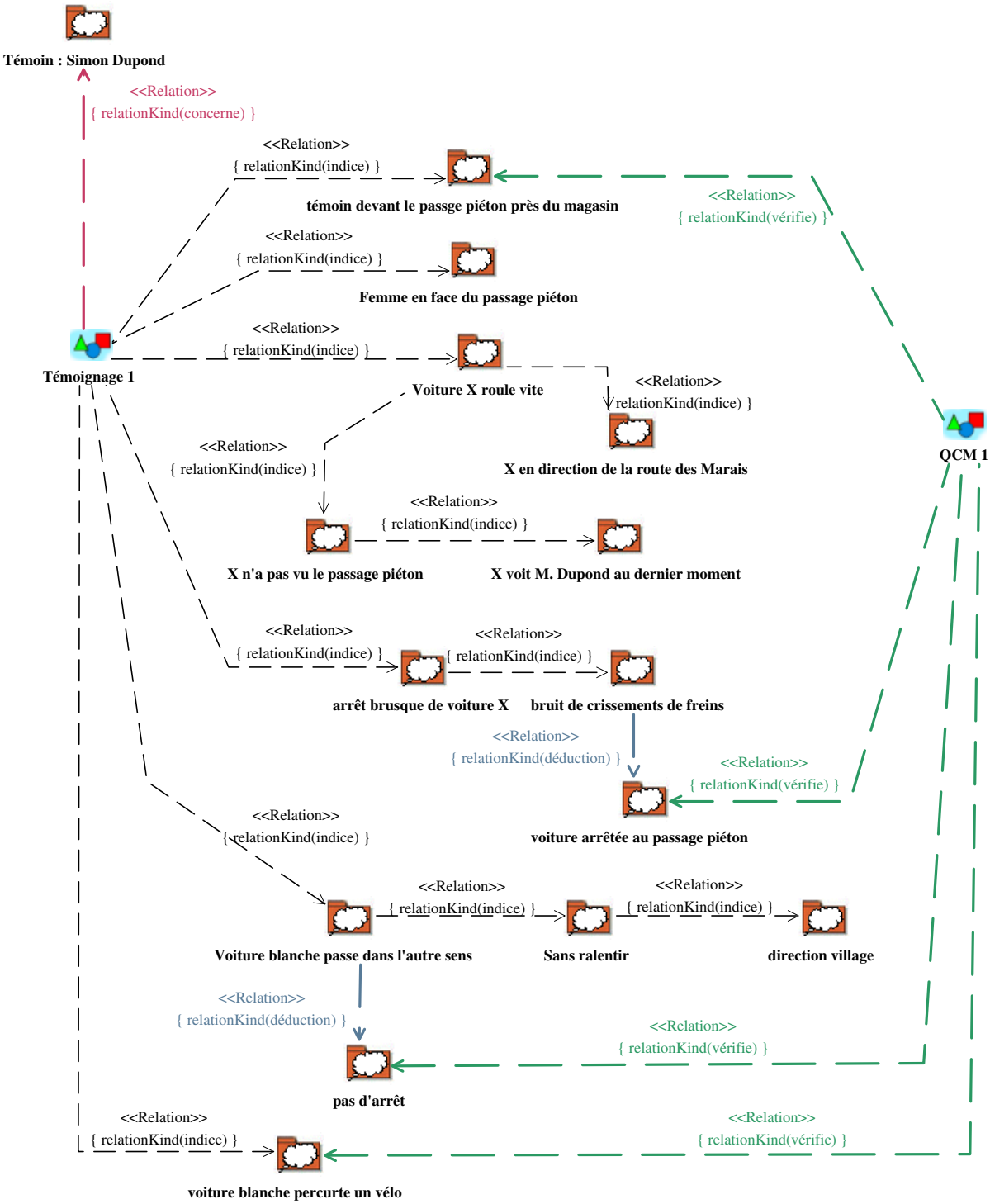

FIG. 9.20 – Analyse du contenu du témoignage 1 (indices, déductions).

9.2.2.3 Fiches de tâches individuelles ou collectives

Une fiche de tâche individuelle peut être définie pour chaque rôle joué. En exemple nous décrivons celle concernant l'enquêteur 1 pour la phase de production.

Énoncé de la figure 9.21 : l'enquêteur 1 doit réaliser en parallèle l'analyse des témoignages et l'étude de la carte et du positionnement des témoins, personnages, voitures, etc. Ensuite, lorsque ces deux activités seront terminées, il devra préparer ses conclusions d'enquête.

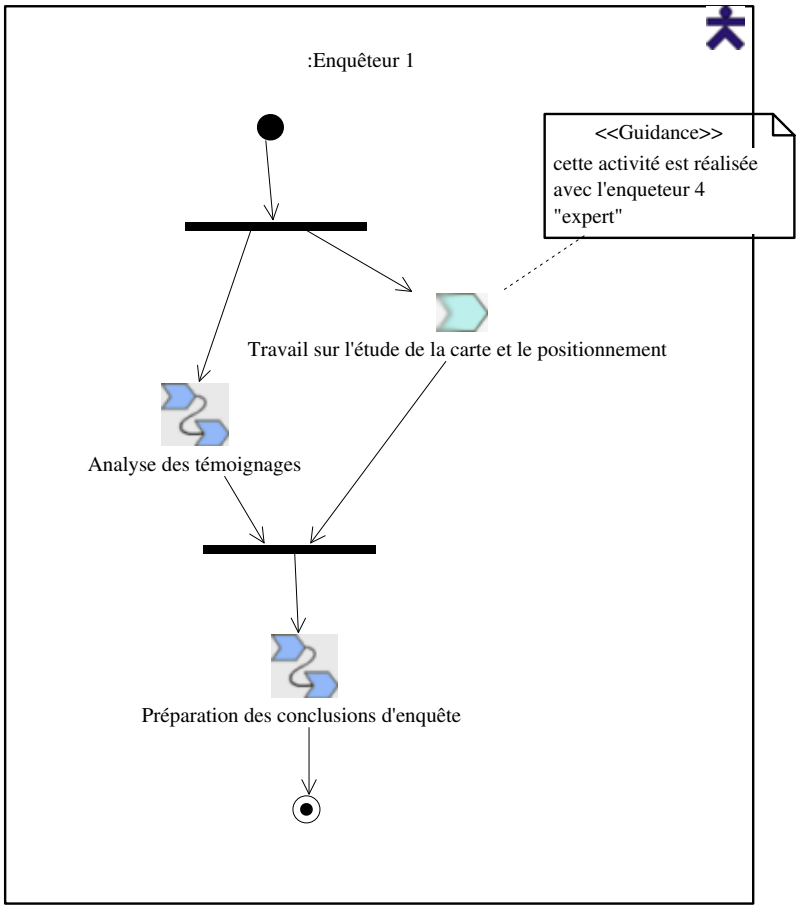

FIG. 9.21 – Fiche de tâche pour le rôle d'enquêteur 1 dans la phase de production.

Utilisation du langage CPM. Dans ce diagramme d'activité, nous avons défini une **Partition** stéréotypée <<**Role**>> qui fait référence à l'*Enquêteur 1* défini dans le diagramme de classe vu précédemment. Les **ActionState** sont stéréotypés <<**Activity**>> (pour *travail sur l'étude de la carte et le positionnement*) et <<**ActivityStructure**>> (pour les autres). Les barres de synchronisation permettent d'encadrer les activités se déroulant en parallèle. Nous avons également attaché à l'une des activités une aide sous la forme d'un **Comment** stéréotypé <<**Guidance**>>.

Nous donnons maintenant un exemple de description d'une structure d'activité.

Énoncé de la figure 9.22. Dans cet exemple, l'analyse des témoignages est une structure d'activité composée d'activités concernant l'analyse de chacun des trois témoignages mis à disposition pour ce rôle. Ces activités doivent toutes êtres réalisées mais dans n'importe quel ordre (*sk_ anyOrder*).

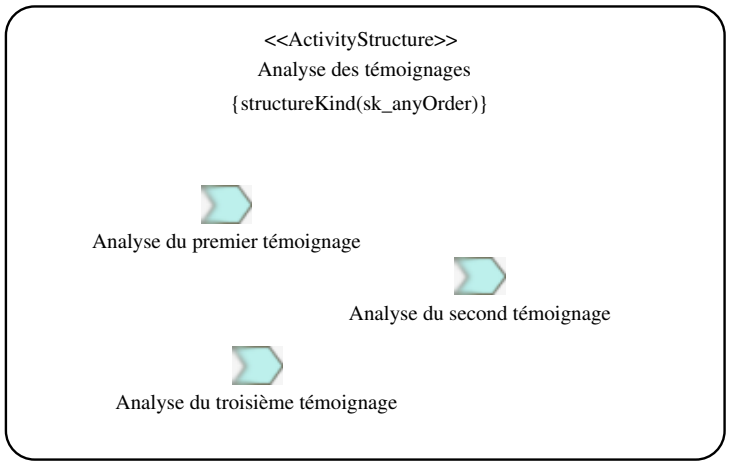

FIG. 9.22 – Détail de la structure d'activité d'analyse des témoignages.

Utilisation du langage CPM. Dans ce diagramme d'activité, *analyse du premier témoignage* est défini comme un **SubActivityState** stéréotypé <<**ActivityStructure**>> ; il contient des **ActionState** stéréotypés <<**Activity**>> représentant les différentes analyses de témoignage. La marque *structureKind* est attachée à la structure d'activité et a pour valeur "*sk_ anyOrder*".

9.2.2.4 Analyse indépendante de chaque activité

Chaque activité peut être décrite indépendamment des autres, du point de vue interne (décomposition) ou externe. Par exemple, l'activité d'*analyse du premier témoignage* pour le rôle d'*enquêteur 1* est décrite en figure 9.23.

Énoncé de la figure 9.23. L'activité d'*analyse du premier témoignage* se décompose en trois étapes : *étudier le premier témoignage*, qui utilise le *témoignage 1*, et *prendre des notes*, qui utilise une *feuille de note*, exécutées en parallèle puis suivies de *répondre au QCM* qui utilise le *QCM 1*.

Utilisation du langage CPM. L'activité d'*analyse du premier témoignage* est décrite comme un **SubActivityState** stéréotypé <<**Activity**>> qui se décompose alors en **ActivityState** stéréotypés <<*Step*>>. Des références aux ressources préalablement décrites (Figure 9.19 par exemple) sont utilisées dans le diagramme afin de représenter leur utilisation ; ainsi le *témoignage 1* est un **ObjectFlowState** stéréotypé <<**Resource**>>.

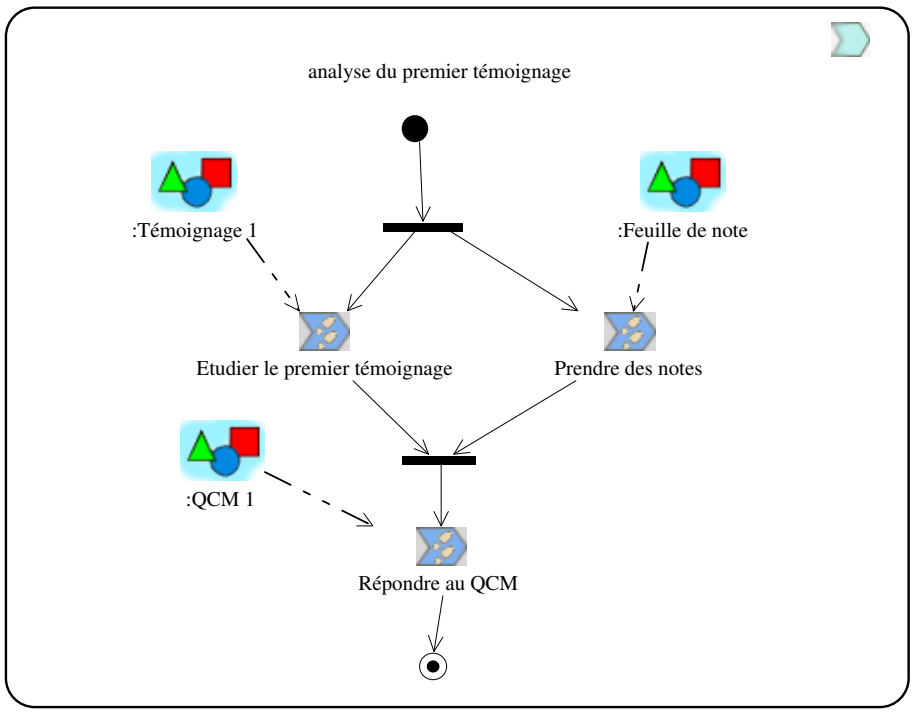

FIG. 9.23 – Détail de l'activité *analyse du premier témoignage*.

Il est également possible d'analyser chaque activité en terme d'entrées/sorties.

Énoncé de la figure 9.24. Elle décrit la structure d'activité *Préparation des conclusions d'enquête* concernant le rôle d'*enquêteur 1* :

- à gauche se situent les *entrants* : les *témoignages 1, 2 et 3*, les *QCM 1, 2 et 3* et la *feuille de notes remplie* après l'analyse des témoignages,
- à droite les *sortants* : la *présentation écrite*,
- en haut les informations relatives au contrôle de l'activité : l'*enquêteur 1* qui réalise l'activité, l'objectif d'apprentissage concerné par l'activité (*travail à l'écrit, synthèse, écriture*),
- en bas les autres informations : ici les critères de succès qui seront connus des apprenants jouant le rôle de l'enquêteur 1 (*où sont les témoins et qu'ont-ils vu ?*, etc.).

Utilisation du langage CPM. Dans ce diagramme de classe *Préparation des conclusions d'enquête* est une **Class** stéréotypée <<**ActivityStructure**>>. Les autres éléments sont des références aux ressources, rôle, objectif déjà définis précédemment. Les critères de succès sont créés pour ce diagramme. Les relations entre ces différents concepts sont réalisées *via* des **Dependency** stéréotypée <<**Relation**>> et marquée de *relationKind* avec pour valeur le nom de la relation.

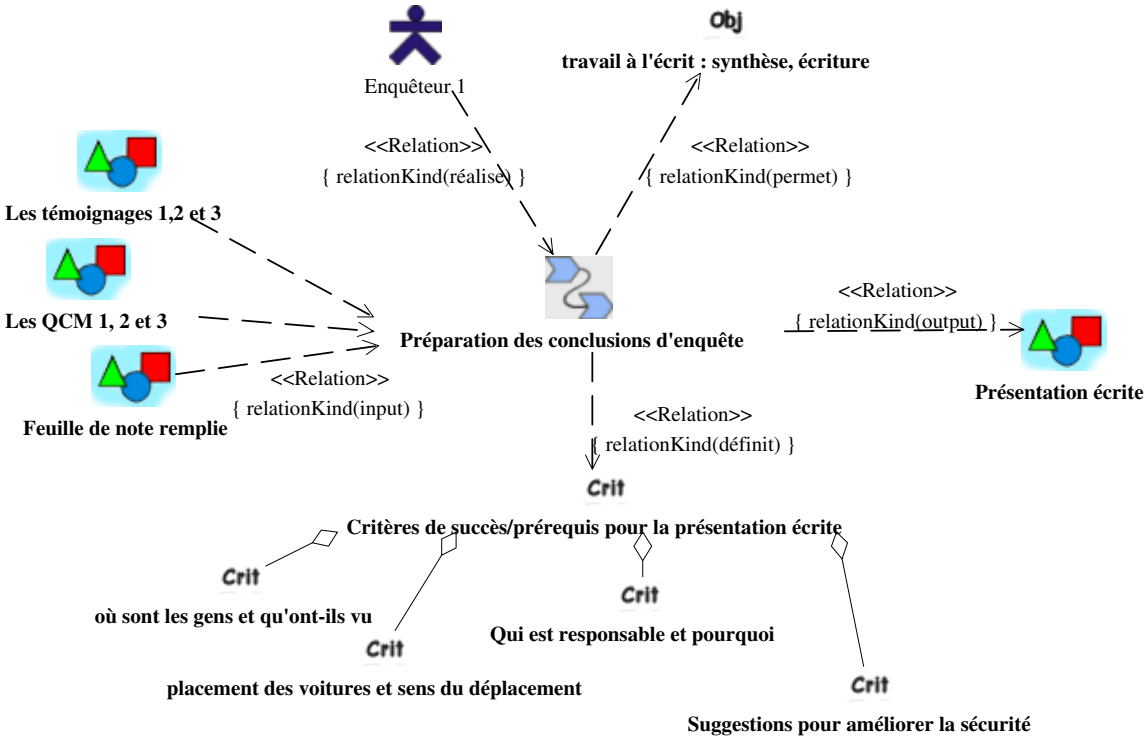

FIG. 9.24 – Analyse externe de l'activité *préparation des conclusions d'enquête* pour le rôle *enquêteur 1*.

9.2.3 Conception de SMASH

La conception de SMASH concerne la mise au point d'un scénario pédagogique. Celui-ci se base sur les analyses précédentes. Il propose une scénarisation plus précise des activités, des rôles et des ressources.

Nous avons choisi d'organiser notre modèle de conception avancée de SMASH d'une manière analogue à l'organisation proposée par la spécification d'IMS-LD (voir 4.3.3.1) Toutefois, notre modèle propose davantage de précisions quant à la description de l'apprentissage :

- les activités des tuteurs sont prises en compte (comme avec IMS-LD) ;
- le découpage du scénario s'effectue en deux niveaux : actes et scènes (toutefois le nombre de niveaux hiérarchiques n'est pas seulement limité à deux niveaux comme dans IMS-LD) ;
- les différents états par lesquels passe une ressource pendant l'ensemble de son cycle de vie peuvent être décrits et réutilisés dans les diagrammes d'activités afin de montrer l'évolution des ressources avant et après les activités les utilisant ; ces états sont également utilisables dans les contraintes *precondition*, *postcondition* des activités (pas possible avec IMS-LD) ;
- les activités collaboratives sont précisées ainsi que les droits sur les ressources de chaque rôle impliqué dans la collaboration (meilleure gestion des activités collaboratives que dans IMS-LD) ;
- d'autres contraintes peuvent être précisées comme la gestion du temps ou les conditions d'exécution

des activités par exemple (traité en partie par IMS-LD).

9.2.3.1 Organisation du modèle de conception avancé

La figure 9.25 illustre l'organisation du modèle de conception. Il s'agit d'une vue de l'organisation interne des éléments de modélisation utilisés dans les diagrammes. La figure 9.25 n'est qu'un exemple indicatif de structuration que nous avons choisi ; l'organisation des éléments de modélisation peut suivre d'autres organisations. De plus, cette vue est liée à l'outil que nous avons choisi, mais tous les AGL UML existants proposent un *explorateur* de ce type.

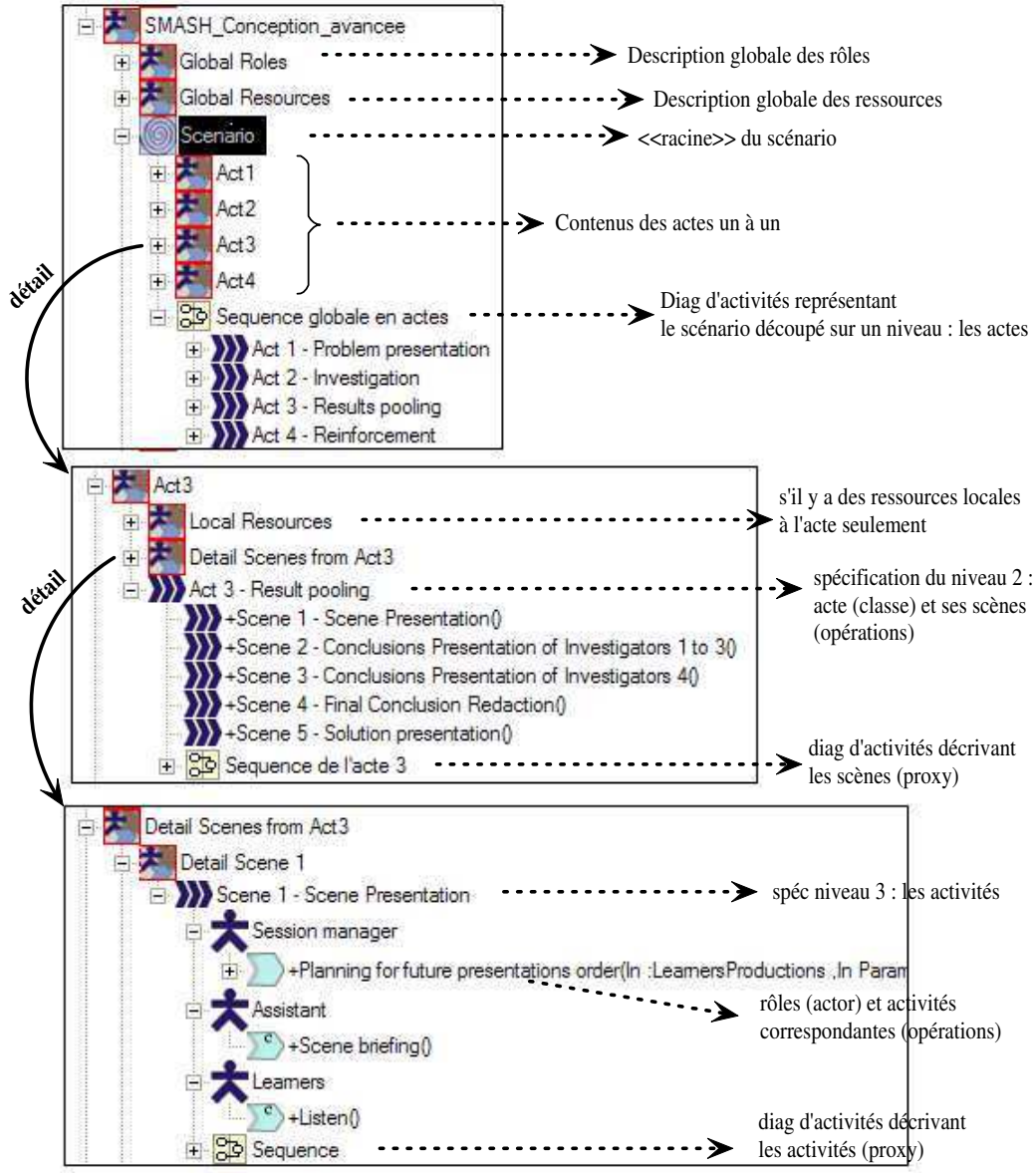

FIG. 9.25 – L'organisation des éléments de modélisation dans le modèle (vue par l'outil).

Nous donnons quelques commentaires afin de mieux décrire la figure 9.25 :

- Le cadre du haut montre les trois paquetages principaux du modèle de conception que nous avons construit pour SMASH :
  - le paquetage *Global Roles* qui décrit les rôles utilisés pour l'ensemble du scénario ;
  - le paquetage *Global Resources* qui décrit les ressources utilisées pour l'ensemble du scénario ;
  - le paquetage *Scenario* qui correspond à la racine de l'organisation du scénario.

  Un diagramme d'activité modélisant le premier niveau des *actes* est rattaché au paquetage *Scenario*. Le détail de chaque acte est décrit dans un paquetage spécifique sous le paquetage *Scenario*.
- Le cadre du milieu correspond à une vue détaillée du paquetage *Act3* du cadre précédent. Il est composé de :
  - un paquetage *Local Resources* qui décrit des ressources utilisés uniquement pour l'acte 3 ;
  - un paquetage *Detail Scenes from Act3* qui décrit en détail le niveau inférieur des scènes ;
  - la **Class** *Act 3* à laquelle sont rattachés :
    - les scènes sous la forme d'**Operation** ;
    - un diagramme d'activité correspondant à la notation *proxy* du découpage en scènes de l'acte3.
- Le cadre du bas correspond à une vue détaillée du paquetage *Detail Scene 1* du cadre précédent. Il se compose d'une **Class** *Scene 1* à laquelle sont rattachés :
  - les différents rôles intervenant dans la scène, sous la forme d'**Actor** ; à chacun de ces rôles est rattachés également les activités qu'il réalise sous la forme d'**Operation**.
  - un diagramme d'activité correspondant à la notation *proxy* du découpage en activités de la scène 1.

L'objectif d'une telle organisation précise et méthodique est de modéliser le scénario de manière formelle et cohérente, c'est-à-dire :

- les éléments de modélisation sont définis une seule fois, à l'exception des notations *proxy* qui nécessitent de créer un élément de modélisation portant le même nom que l'élément dont il fait référence ;
- les éléments de modélisation sont définis dans des paquetages structurés de manière à garantir la visibilité des éléments là où il est possible de les référencer.

Cette formalisation dépasse le cadre des usages que nous avions fixés pour le langage CPM. Toutefois, nous nous y sommes intéressés dans une perspective de future opérationnalisation du scénario et du modèle le spécifiant. En effet, toute opérationnalisation (transformation de modèles, exportation vers d'autres langages, etc.) nécessite une connaissance précise sur la construction des modèles UML. Ce travail est nécessaire si l'on veut réaliser une projection d'un modèle de scénario décrit avec CPM vers un scénario spécifié avec IMS-LD.

9.2.3.2 Description détaillée des rôles

La figure 9.26 décrit en détail les différents rôles intervenant dans la spécification de conception avancée de SMASH. Ils s'agit des mêmes rôles que ceux définis en analyse. Seul le rôle de *gestionnaire de séance* a été rajouté. Les éléments de modélisation constituant ce diagramme sont décrits dans le paquetage *Global Roles* de la figure 9.25.

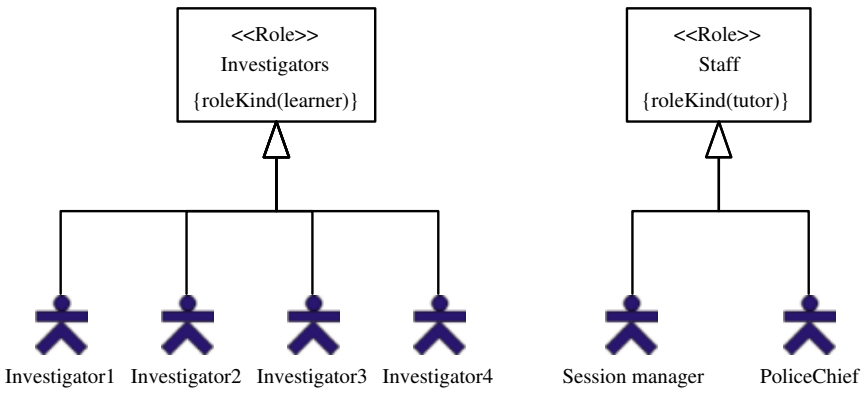

FIG. 9.26 – Exemple de spécification des rôles pour un modèle de conception avancé de SMASH.

9.2.3.3 Description détaillée des ressources

Énoncé. La figure 9.27 décrit en détail les différentes ressources attribuées au rôle d'enquêteur 1 : les témoignages, les QCM, feuille de notes, etc. De nouvelles ressources sont décrites pour regrouper les ressources élémentaires et faciliter leurs manipulations dans les autres vues : *QCMs* qui regroupe tous les QCM, *Witnesses* qui regroupe tous les témoignages, *Resources1* pour représenter l'ensemble des ressources et *Productions1* qui rassemble l'ensemble des ressources qui seront créées ou modifiées par le rôle associé (c'est-à-dire les QCM, la feuille de notes et la feuille sur laquelle les apprenants écriront leur conclusions).

Utilisation du langage CPM. Dans ce diagramme de classe, nous utilisons le stéréotype <<**isResponsibleFor**>> défini sur la méta-classe **Association** ainsi que le stéréotype <<**Resource**>> que nous avons déjà utilisé dans des figures précédentes. Nous utilisons la relation de composition UML pour indiquer qu'une ressource en rassemble d'autres. Par exemple, *Resource1* regroupe toutes les ressources pour le rôle d'enquêteur 1 ; de la même manière, *Productions1* regroupe toutes les ressources qui seront créées ou modifiées par les apprenants jouant ce rôle. La définition de ces ressources composées permet de faire référence plus facilement dans les modèles à plusieurs ressources élémentaires.

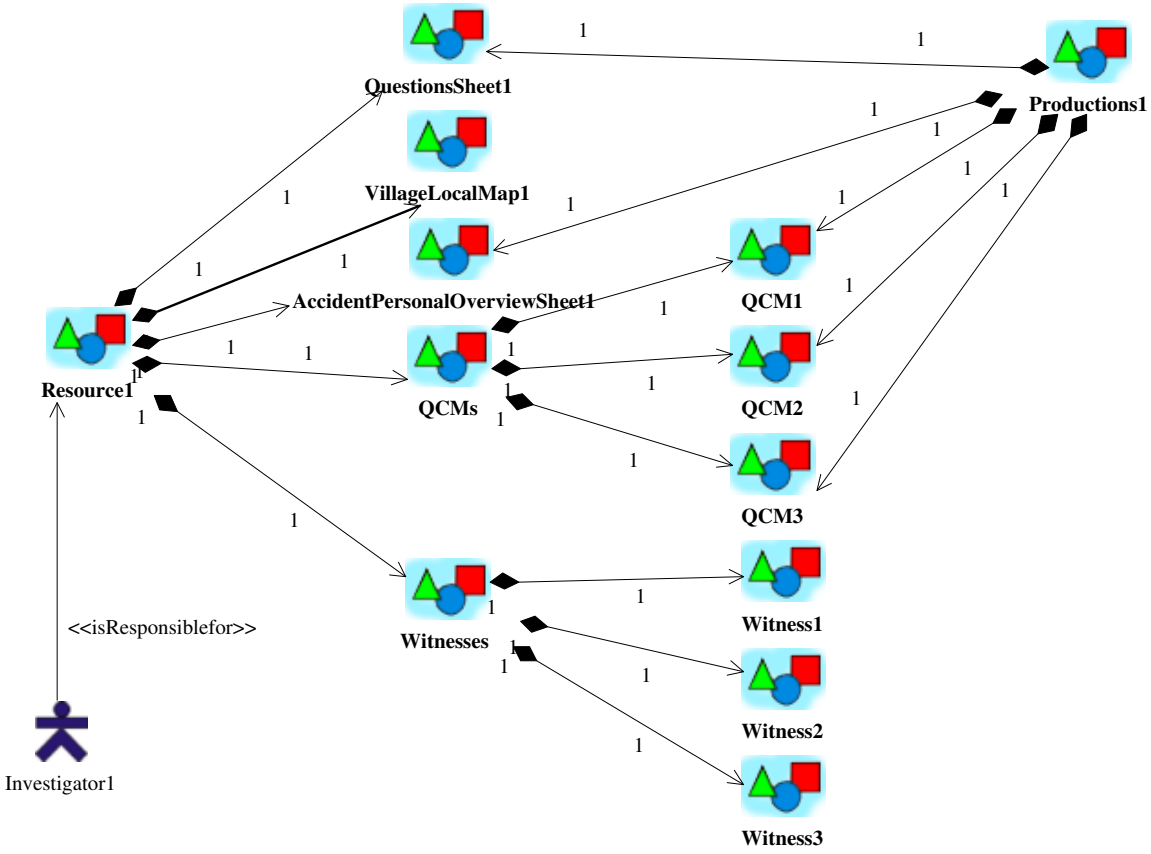

FIG. 9.27 – Exemple de spécification des ressources pour un modèle de conception avancé de SMASH.

9.2.3.4 Description des différents états d'une ressource

Les différents états d'une ressource peuvent être décrits (exemple en Figure 9.28) afin d'être réutilisés dans les modèles pour préciser l'état des ressources en entrée (pré-condition) et en sortie (postcondition) des activités pédagogiques. Dans l'exemple que nous donnons, sont représentés les différents états du QCM 1 (tous les QCM ont ces mêmes états). L'état initial est *not completed* ; lorsque les apprenants jouant le rôle de l'enquêteur 1 auront répondu au QCM, il sera dans l'état suivant *completed* ; les deux autres états indiquent que les réponses au QCM sont correctes (*well completed*) ou incorrectes (*bad completed*).

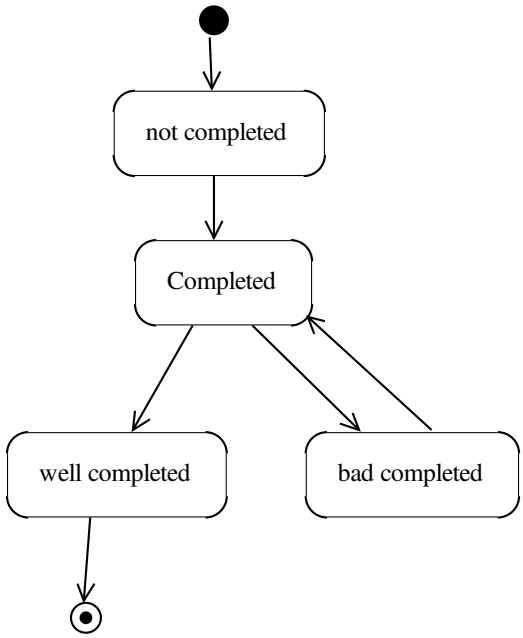

FIG. 9.28 – Exemple de spécification des états de la ressource *QCM 1*.

9.2.3.5 Spécification du scénario (premier niveau) : les actes

La vue globale du découpage du scénario peut être représentée sur un niveau (les actes) par un diagramme d'activités semblable à celui du modèle d'analyse précédent (Figure 9.29). Le terme d'*acte* apparaît dans le nom de la phase mais il est surtout précisé au niveau de la valeur marquée *phaseKind* (la valeur marquée est spécifiée pour tous les actes du diagramme mais elle est seulement affichée pour le premier[128]).

[128]Concrètement, il s'agit d'une contrainte de l'outil utilisé : il n'est pas permis d'utiliser la représentation sous forme d'icône pour un élément de modélisation stéréotypé pour lequel on affiche ses valeurs marquées.

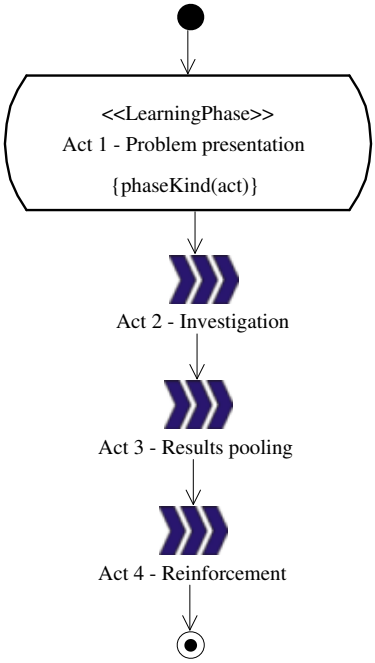

FIG. 9.29 – Exemple de spécification de scénario : les actes (niveau 1).

9.2.3.6 Spécification du scénario (deuxième niveau) : les scènes

Chaque acte est décomposé en scènes. La figure suivante (9.30) décrit ce découpage pour l'acte 3 en montrant d'une part un diagramme de classe décrivant la structure de l'acte (à gauche de la figure) et un diagramme d'activité décrivant la dynamique des enchaînements des scènes (à droite). Ce dernier fait référence aux scènes du diagramme de classes (décrites comme des **Operation** stéréotypées <<**LearningPhase**>>) sous leur forme *proxy*[129] (décrites comme des **ActionState**).

Nous pouvons remarquer que lorsque les scènes sont décrites sous formes d'opérations elles sont alors rattachées à l'acte correspondant.

[129]Nous avons expérimenté dans l'outil des interfaces aidant le concepteur à créer des références *proxy* par glisser/déposer des éléments de modélisation de référence.

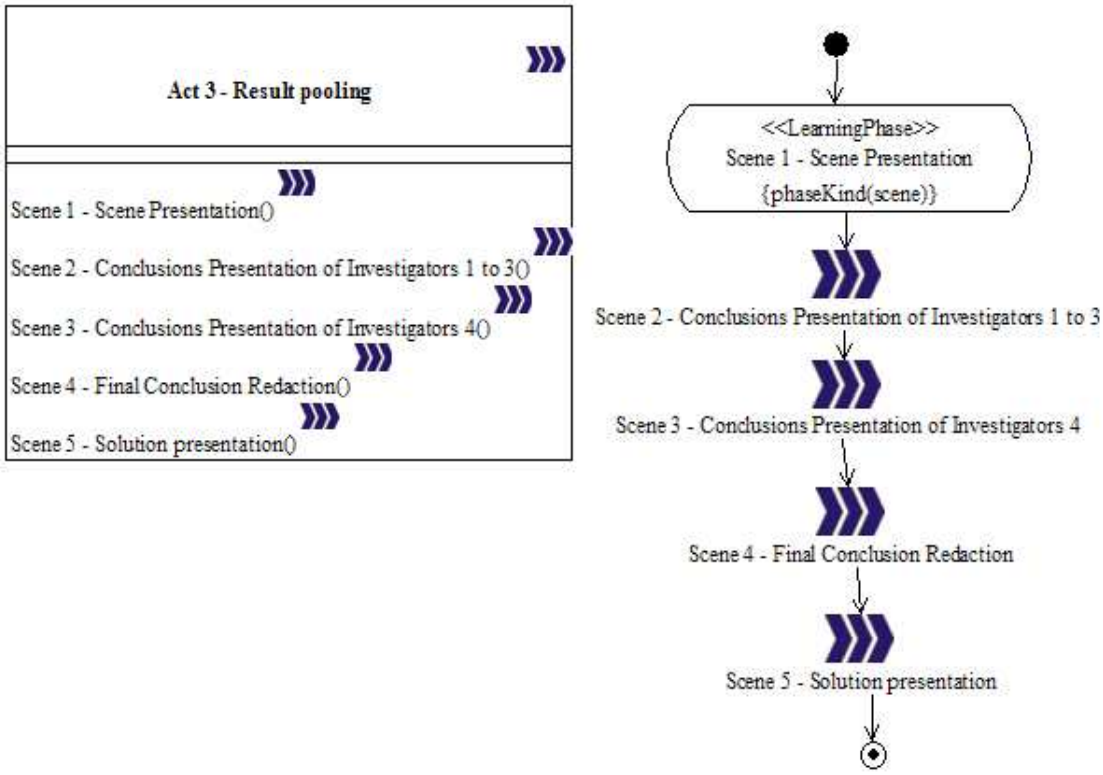

FIG. 9.30 – Spécification des scènes de l'acte 3 (niveau 2).

9.2.3.7 Spécification du scénario (troisième niveau) : les activités de la scène 1

Les scènes se décomposent à leur tour mais en activités. Il n'y a donc que deux niveaux d'organisation (actes et scènes) ; davantage de niveaux peuvent être définis pour d'autres PBL. Les figures suivantes illustrent le détail de la scène 1 de l'acte 3.

Énoncé de la figure 9.31. La scène 1 de *Présentation* de l'acte 3 est constituée de 3 activités se déroulant en parallèle : le *PoliceChief* présente le déroulement de la séquence qui débute (*Scene presentation*) tandis que les apprenants (*Investigators*) écoutent (activité *Listen*). Dans le même temps, le gestionnaire de séance étudie les productions des enquêteurs issues de l'acte précédent et décide d'un ordre pour les présentations des conclusions qui seront réalisées par les différents rôles d'enquêteur dans la scène suivante.

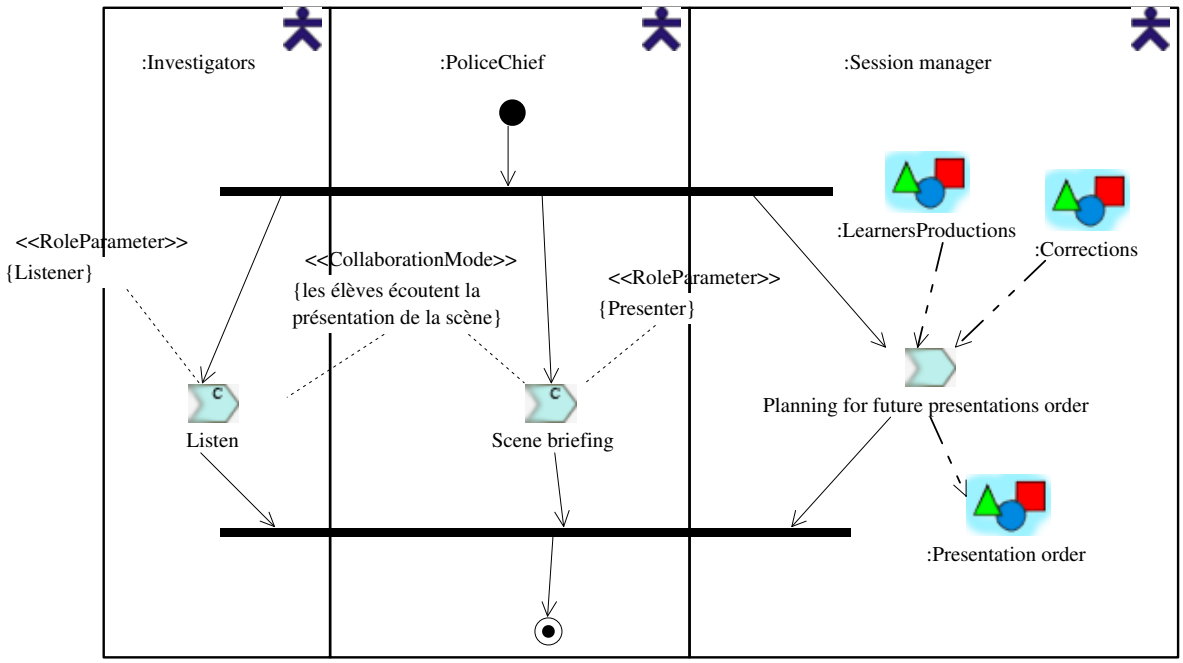

FIG. 9.31 – Spécification des activités de la scène 1 de l'acte 3.

Utilisation du langage CPM. Les activités *Listen* et *Scene briefing* sont des activités collaboratives (**ActionState** stéréotypé par <<**CollaborativeActivity**>>) tandis que la troisième activité, bien que se déroulant en parallèle, n'est qu'une activité individuelle (stéréotype <<**Activity**>>). Les deux activités collaboratives sont reliées *via* la **Constraint** stéréotypée <<**CollaborationMode**>>. Afin d'indiquer un rôle dans la collaboration nous utilisons une contrainte stéréotypée <<**RoleParameter**>>.

Nous donnons dans la figure suivante (9.32) la scène 1 telle qu'elle est capturée statiquement en termes de **Class**, d'**Operation** et de **Parameter** pour le passage des ressources. Nous faisons remarquer qu'une ressource utilisée correspond à un **Parameter** stéréotypé <<**ActivityParameter**>> avec

l'attribut de direction indiquant *IN*, tandis que les ressources créées ont l'attribut de direction à *OUT* (malheureusement il n'est pas possible de faire apparaître les noms des stéréotypes sur les **Parameter** dans la figure).

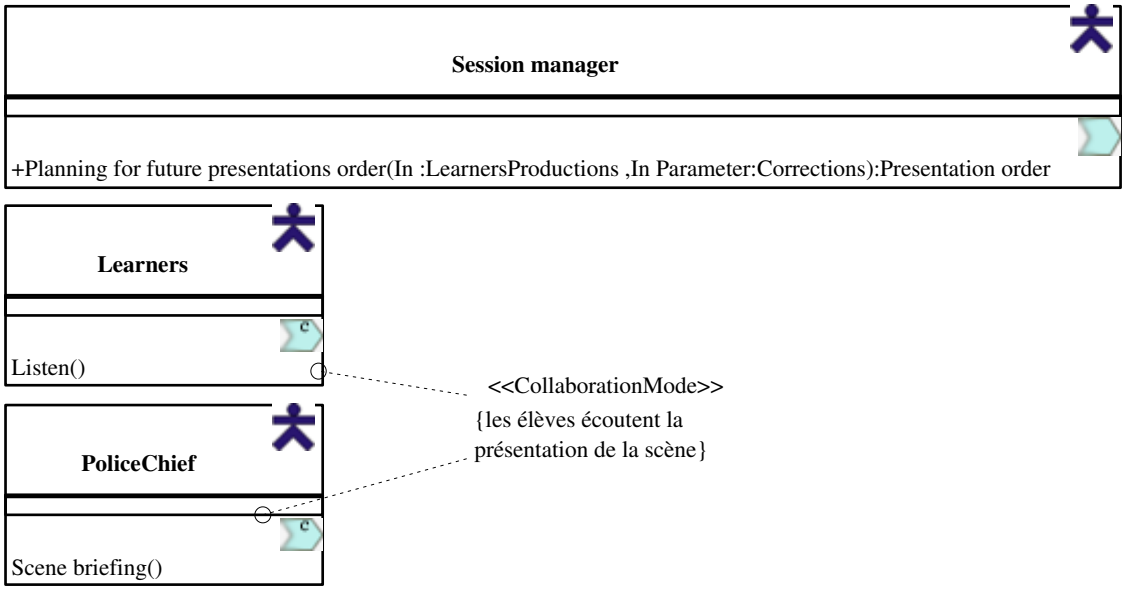

FIG. 9.32 – Spécification statique des activités de la scène 1 de l'acte 3.

9.2.3.8 Spécification du scénario (troisième niveau) : les activités de la scène 2

Les figures suivantes illustrent le détail de la scène 2 de l'acte 3. Tout d'abord nous présentons de nouveaux rôles créés localement pour cette scène : ils sont illustrés par la figure 9.33.

Énoncé de la figure 9.33. Deux nouveaux rôles sont spécifiés (*Speaker* et *Auditor*) uniquement pour la scène 2. En effet, pendant cette scène les enquêteurs 1 à 3 présentent chacun leur tour leurs conclusions (ils sont donc momentanément *Speaker*) tandis que les autres écoutent (*Auditor*).

Utilisation du langage CPM. Grâce au stéréotype <<**Relation**>> et à la valeur marquée *relationKind*, nous définissons la relation *subRole*, afin d'associer les rôles globaux aux nouveaux rôles locaux.

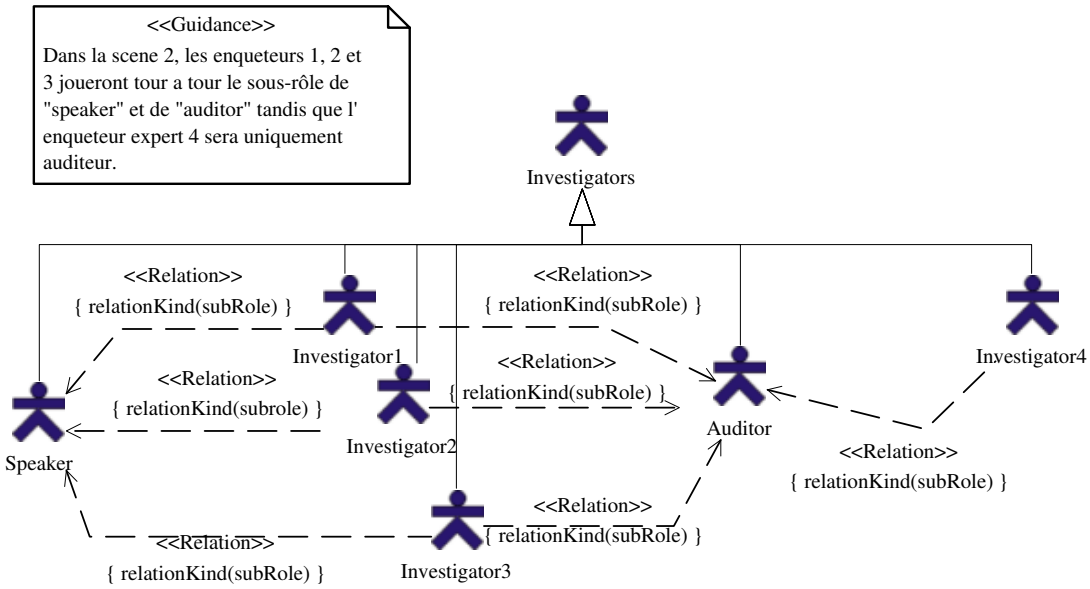

FIG. 9.33 – Spécification de nouveaux rôles locaux pour la scène 2 de l'acte 3.

Le détail de la scène 2 est présenté cette fois-ci uniquement dans sa réalisation *dynamique* (nous n'illustrons pas le diagramme de classe comme pour la scène 1 précédente).

Énoncé de la figure 9.34. Le gestionnaire de séance donne la main au premier enquêteur pour qu'il présente ses conclusions sur l'accident (selon l'ordre de présentation produit dans la scène précédente). Ensuite, la présentation démarre : les apprenants jouant le rôle de l'enquêteur *Speaker* présentent leurs conclusions, assistés du tuteur (*PoliceChief*) et écoutés des autres enquêteurs. Ceux qui écoutent peuvent noter des questions au fur et à mesure *via* la ressource *QuestionSheet* mais ils n'interviennent pas. Le temps imparti est géré par le gestionnaire de séance : 8 minutes par présentation. Ensuite, c'est le passage des questions/réponses entre le groupe qui présentait et le tuteur seulement. Les interventions du tuteur sont liées à la présentation qui a précédé mais sont également dirigées selon les différentes erreurs qu'il peut relever en comparant les résultats des productions du groupe (les réponses aux QCM par exemple) avec les réponses qui étaient attendues. Lorsqu'il n'y pas plus de question ou que le temps est dépassé, un nouveau groupe d'enquêteurs devient à son tour le *speaker*, etc.

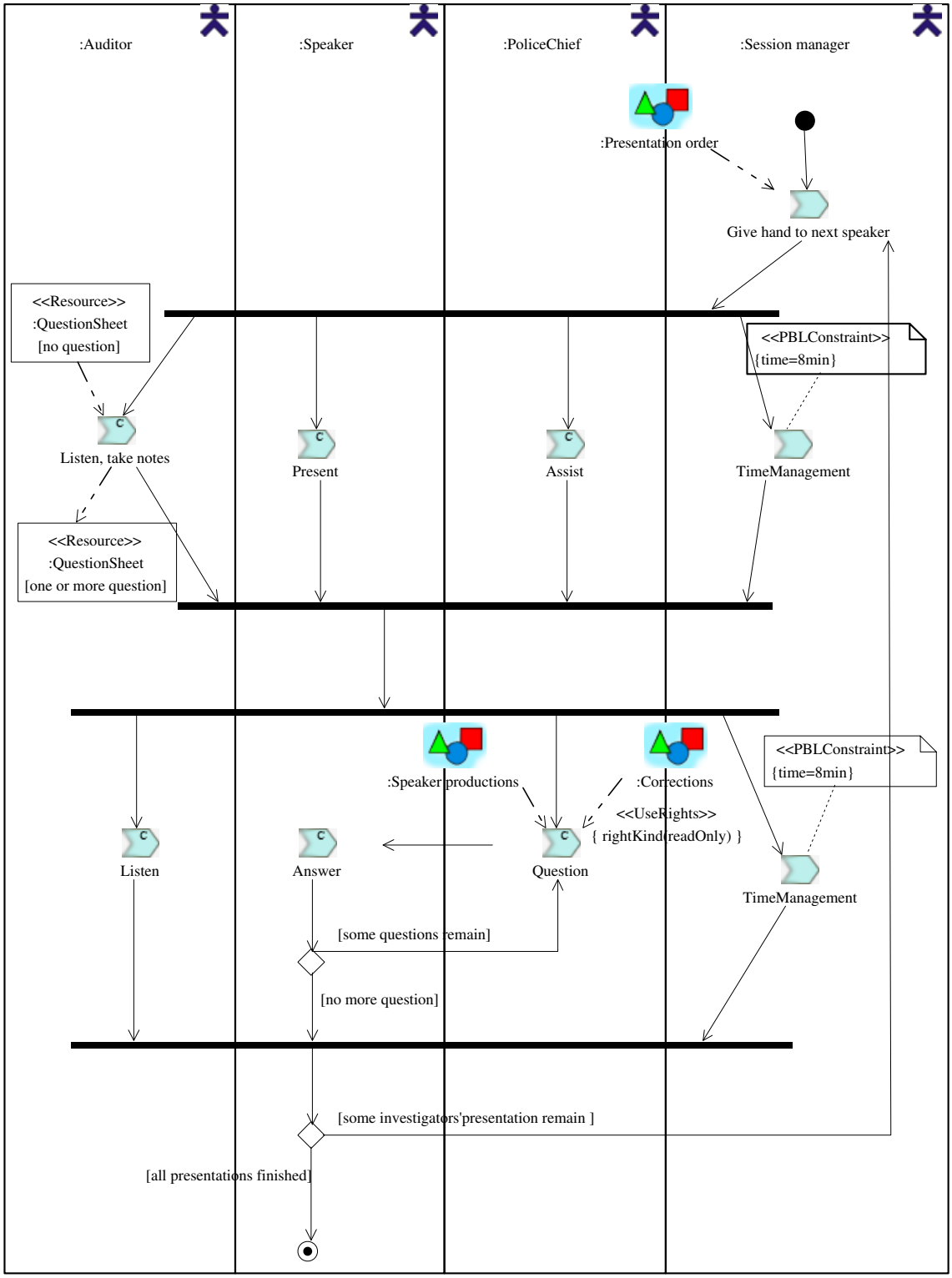

FIG. 9.34 – Spécification des activités de la scène 2 de l'acte 3.

Utilisation du langage CPM. Certaines activités sont stéréotypées à l'aide de <<**Collaborative-Activity**>> : elles sont également reliées entre elles par une contrainte <<**CollaborationMode**>> (mais ces contraintes ont été masquées pour ne pas surcharger le diagramme). La ressource *QuestionSheet* fait référence à la feuille de questions propre au rôle d'enquêteur concerné ; elle est définie dans le paquetage des ressources locales à la scène 2 (non représenté par une figure). Les états de la ressource en entrée de l'activité (*[no question]*) et en sortie (*[one or more question]*) sont précisés. Ces états attendus permettent de spécifier des <<**Precondition**>> et des <<**Postcondition**>> sur cette activité *via* des contraintes convenablement stéréotypées (ces contraintes ont été masquées sur la figure pour ne pas la rendre illisible). La gestion du temps est modélisée ici par une activité dédiée (*TimeManagement*) sur laquelle nous avons défini une contrainte (*via* <<**PBLConstraint**>>). Au niveau de la deuxième partie de la scène, nous utilisons les branches conditionnelles d'UML pour décrire l'enchaînement dynamique des activités selon des événements. Nous avons défini également un exemple de l'utilisation de <<**UseRights**>> sur la **Transition** allant de la ressource *Corrections* vers l'activité *Question*.

9.3 Bilan

Ce chapitre a montré deux travaux expérimentaux pour la vérification et la mise à l'essai du langage CPM :

1. Un outillage avec un AGL existant : *Objecteering*. Ceci a permis de :
   - vérifier la faisabilité d'élaboration de modèles en utilisant le profil CPM ;
   - vérifier la cohérence des concepts et relations du profil et du méta-modèle CPM ;
   - vérifier la réutilisation de modèles, parties de modèles ;
   - vérifier la réutilisation de concepts définis dans une partie du modèle et réutilisé dans d'autres parties ;
   - préparer l'expérimentation sur le cas d'étude SMASH ;
   - vérifier la faisabilité de personnaliser et adapter l'interface de l'AGL (menus, commandes, onglets de propriétés, etc.) pour faciliter la création et le suivi de modèles pour des non-spécialistes d'UML.
   - illustrer la possibilité de transformer les modèles produits vers d'autres langages (nouvel usage possible pour les modèles CPM).
2. Une application au cas d'étude SMASH pour :
   - élaborer des modèles de conception concernant différents étapes horizontales (expression initiale des besoins, analyse et conception) ;
   - élaborer différentes vues verticales pour chaque modèle (vue structurelle, vue sociale, vue pédagogique).

Nous avons pu remarquer également que l'utilisation d'un AGL UML existant offre de nombreux avantages (rapidité d'élaboration de modèles avec notre langage, possibilité d'adaptation de l'outil) mais également des inconvénients liés à l'implémentation du méta-modèle et de la notation d'UML spécifique à chaque outil.

La mise à l'essai applicative du langage CPM sur le cas d'étude SMASH a permis d'illustrer et d'*explorer* la richesse d'expression héritée d'UML. Cette grande variété de diagrammes, pour des niveaux

d'abstraction variables, pour des vues différentes (sociales, structurelles, pédagogiques, etc.) et des usages différents (analyse de la tâche en amont ou encore spécification précise du scénario pédagogique en aval) valide notre principal objectif : fournir un langage de modélisation visuel riche permettant de décrire un large champ de modèles pour la conception de PBL. L'adjonction d'une méthode adaptée au langage CPM permettra de mieux guider les concepteurs et l'ingénieur pédagogique dans l'utilisation de notre proposition.

Une seconde démarche de validation concerne, non plus le langage CPM, mais les modèles construits avec celui-ci. Dans le cas de SMASH, les modèles produits, dont certains extraits ont été présentés dans ce chapitre, ont été bâtis sur la base de documents traditionnels d'analyse et de conception de SMASH élaborés par des professeurs des écoles avec lesquels nous avons collaborés. De plus, le scénario de SMASH a été expérimenté en présentiel dans une classe de cours moyen. Toutefois, un travail expérimental de validation doit être mené sur d'autres cas d'études pédagogiques et par des concepteurs pédagogiques. Pour cela, nous partageons dans les perspectives de nos travaux des pistes pour initier ce travail.

Chapitre 10

Composant CPL

Sommaire

Ce chapitre est le dernier de la partie consacrée à notre contribution. Il est dédié à la deuxième proposition telle que nous l'avons présentée dans le chapitre 6.

Ce chapitre se décompose en trois sections. Tout d'abord nous introduisons le cadre de notre proposition (10.1), puis nous détaillons notre modèle de composant éducatif CPL ainsi que les démarches d'utilisation (10.2), et pour finir, nous expérimentons la proposition (10.3).

10.1 Introduction

Le langage CPM permet aux concepteurs de prendre en compte leurs besoins pédagogiques liés à la situation-problème coopérative qu'ils désirent mettre en place. Ces besoins sont capturés par différents modèles de conception *indépendants* du dispositif informatique mettant en œuvre à distance cette situation d'apprentissage.

Notre étude des plates-formes pour l'apprentissage à distance a mis en évidence leurs fonctionnalités et outils transversaux à l'apprentissage. Ces outils sont plus ou moins génériques et peuvent être considérés comme des outils « standards » proposés par les LMS actuels.

La problématique à laquelle nous nous intéressons s'exprime alors ainsi : comment prendre en compte les fonctionnalités rendues par les plates-formes dans les modèles de conception décrits avec le langage CPM ? (Figure 10.1)

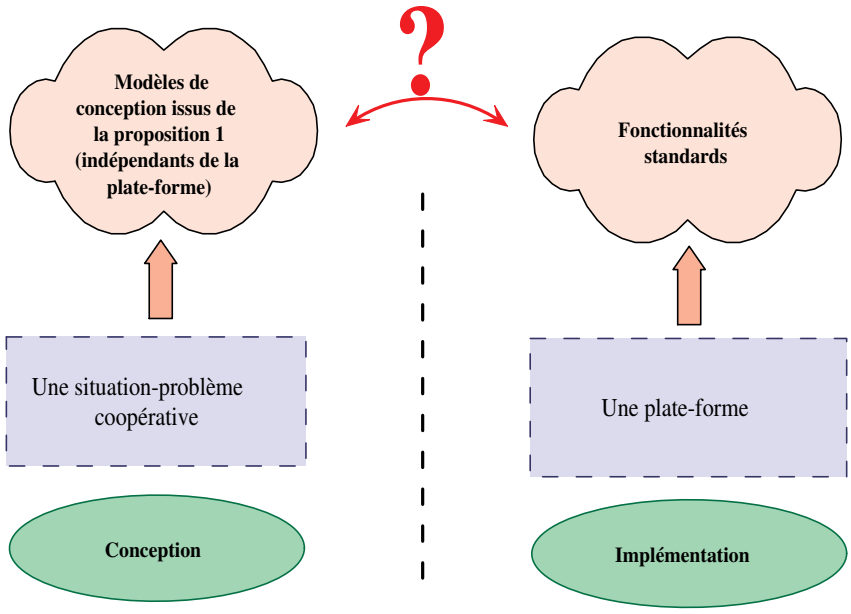

FIG. 10.1 – Illustration de l'écart entre les besoins issus de la conception pédagogique indépendante du dispositif informatique support et les fonctionnalités standards fournies par les plates-formes.

10.1.1 Démarche intuitive

Pour la spécification IMS-LD [IMS03c] (le standard actuel pour la conception avancée et formelle de situations d'apprentissage), la prise en compte des fonctionnalités rendues par le LMS s'effectue uniquement par le biais des services. Afin que la situation d'apprentissage conçue soit réutilisable, ces services ne sont définis que de manière abstraite dans les modèles de conception (*design models*). Ils seront ensuite *instanciés* par des services locaux d'exécution (appelés *service facilities*) lors de l'utilisation de l'unité d'apprentissage pour des personnes et configurations précises (il s'agit de la phase d'*instanciation* du processus d'utilisation que nous avons présentée dans la section 1.2.2.3). La spécification IMS-LD rappelle qu'il est nécessaire d'avoir un ensemble précis de types de services locaux tel que le *chat*, les forums de discussion, etc. Par contre, IMS-LD ne propose actuellement que quatre types de services abstraits pour les modèles d'unités d'apprentissage : le courrier électronique, la conférence, le suivi de propriétés et la recherche indexée (ces services sont détaillés dans la spécification). Par exemple, le service de conférence spécifie quatre rôles d'utilisation : participant, observateur, conférence-manager, et modérateur.

Ces rôles doivent être référencés pendant la conception aux endroits où le service est utilisé. Par contre, les permissions et autres droits liés à ces rôles sont laissés à la charge de l'implémenteur. Ce service de conférence peut être divisé en trois sous-types : conférences synchrones (exemple : chat et conférences audio/vidéo), conférences asynchrones (newsgroups, forums), et *announcements* (conférences asynchrones *one to many*). Comme indiqué dans [IMS03c], la sélection des services à proposer a besoin d'être dirigée par la communauté ; c'est pour cela que la spécification ne propose, pour commencer, que les services les plus largement implémentés et utilisés dans les environnements d'apprentissage distants.

Une démarche intuitive consiste alors à utiliser le langage CPM de manière similaire à la spécification IMS-LD. Pour cela, le concept **Resource** convient pour la définition de service : la valeur marquée *resourceKind* permet alors de typer les ressources comme étant des *services*. La figure 10.2 (partie gauche) illustre la définition de tels services dans nos modèles. L'un des services est montré sous sa forme détaillée afin de caractériser l'usage de la valeur marquée.

Les services peuvent donc être réutilisés ensuite dans les modèles là où les **Resource** sont autorisées par le langage CPM. Un exemple simple d'utilisation est illustré (Figure 10.2 - partie droite). Dans cet exemple, nous utilisons également le stéréotype **UseRights** et la valeur marquée *rightKind* pour donner des précisions quant à l'utilisation d'une ressource pour un rôle et une activité donnée. Dans le diagramme d'activités, ils permettent de préciser les rôles joués (participant et observateur) dans l'utilisation du service.

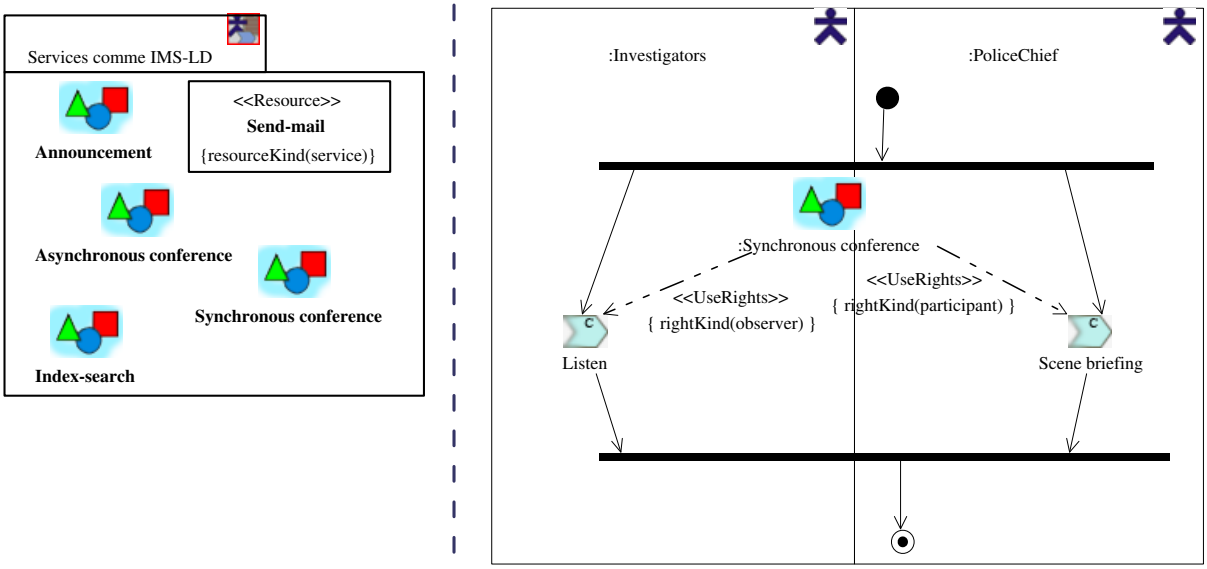

FIG. 10.2 – Une démarche analogue à la spécification d'IMS-LD pour décrire des services avec le langage CPM.

10.1.2 Vers une nouvelle démarche de mise en œuvre des situations d'apprentissages

L'étude des plates-formes et des composants pour les plates-formes du chapitre 3 a abouti aux conclusions suivantes :

- les outils (ou services rendus) par les plates-formes sont transversaux aux apprentissages mis en œuvre sur le dispositif : ils sont choisis dans le cadre d'une activité [130] et donc pour sa globalité (ils sont liés au cycle de vie de l'activité) ;
- les services sont accessibles *via* l'interface (IHM) de la plate-forme ; les événements ou actions de l'utilisateur (hormis la terminaison de l'activité) n'ont pas d'influence sur les services proposés ;
- les plates-formes proposent des outils/services génériques dédiés à la conversation (chat, forum, visio-conférence, etc.), la communication (échange de fichiers, espace de travail partagé, etc.), la production (outils de bureautique intégrés, outils spécifiques à un domaine, partage d'applications, etc.), la coordination (agenda, suivi de la progression, etc.) ;
- les architectures des LMS à base de composants technologiques (EJB par exemple) se démocratisent et représentent le futur palier d'évolution de ces systèmes du point de vue technologique ;
- les couches (verticales) structurant ces architectures proposent actuellement sur la couche externe des composants/outils fournissant des services génériques (voir figure 3.4).

Plutôt que d'essayer d'intégrer les fonctionnalités des plates-formes dans les modèles de conception, nous avons basé notre proposition sur une démarche consistant à ajouter de nouvelles fonctionnalités aux plates-formes qui répondent davantage aux besoins des concepteurs : c'est-à-dire des fonctionnalités mieux intégrées aux activités pédagogiques qu'ils conçoivent. Nous comptons ainsi réduire l'écart entre les besoins nés de la conception et les fonctionnalités des plates-formes en ajoutant un nouvel élément intermédiaire : le Composant Pédagogique Logiciel (CPL).

10.2 Notre proposition

Nous proposons un modèle spécifique pour le composant éducatif baptisé Composant Pédagogique Logiciel (CPL). Ce modèle a pour objectifs de :

- définir et spécifier des composants CPL pour les plates-formes de formation à distance ;
- spécifier des modèles de conception avancée, avec une version étendue du langage CPM, intégrant ces composants CPL comme des activités pédagogiques réutilisables de bas niveau.

Notre modèle de CPL s'adresse ainsi :

- aux *développeurs* et *assembleurs de composants* (définis en 1.2.2.2) des LMS qui spécifieront et implémenteront les composants CPL sur les plates-formes LMS (phase d'implémentation) selon notre modèle ;
- à l'équipe de conception qui réutilisera une partie des représentations des nouveaux composants CPL spécifiés selon le modèle (étape de conception avancée de la phase de conception globale, voir figure 6.1).

La prochaine sous-section présente cette contribution : le modèle des composants CPL (10.2.1). Puis, nous présentons une démarche pour aider les développeurs de composants et l'équipe de conception à

[130] La spécification IMS-LD appelle ce cadre *environnement*.

collaborer pour définir, spécifier et implémenter des composants CPL en utilisant notre modèle (10.2.2). Enfin, nous présentons la démarche concernant l'utilisation de ces modèles de composants CPL pour la spécification de modèles de conception avancée pour les PBL (10.2.3).

10.2.1 Notre modèle de Composant Éducatif : le composant CPL

Nous commençons tout d'abord par définir les composants CPL (10.2.1.1), puis nous proposons le modèle de CPL, basé sur la modélisation UML, qui servira à les spécifier (10.2.1.2). Ensuite, nous discutons un exemple de composant CPL décrit selon le modèle précédent (10.2.1.3). Pour terminer, nous détaillons le cœur de la proposition basé sur une utilisation spécifique des diagrammes d'états UML (10.2.1.4).

10.2.1.1 Définition et description d'un composant CPL

Le composant CPL concerne à la fois les concepteurs de PBL et les développeurs de composants pour les plates-formes ; il est toutefois perçu différemment selon les acteurs concernés.

Perception pédagogique du composant CPL. Le concept d'activité est central dans la conception de situations-problèmes. La plupart des modèles détaillés (généralement des modèles en *aval* de la phase de conception comme nous en avons présenté des extraits dans le chapitre précédent) mettent en évidence, pour des situations-problèmes différentes, des activités identiques pour les apprenants comme pour les tuteurs (par exemple nous avons rencontré dans le chapitre précédent *Listen*, *Present*, *Question* ou encore *Answer*). Notre idée consiste alors à « capturer » ces activités afin de pouvoir les ré-utiliser.

Le rôle du composant CPL est alors d'embarquer les fonctionnalités nécessaires à la réalisation d'une activité pédagogique élémentaire.

Perception technique du composant CPL. Le composant CPL est perçu comme un composant logiciel « métier » ayant les caractéristiques suivantes :

- il fournit des services « pédagogiques » pour le tutorat, la régulation, ou encore la production (ces services sont désignés comme *pédagogiques* car ils répondent à des attentes des concepteurs des situations d'apprentissage) ; les services fournis représentent les services attendus pour la réalisation de l'activité pédagogique gérée par le composant CPL ;
- il nécessite des services logiciels (qui supporteront la réalisation des services pédagogiques) ;
- il décrit une ou plusieurs vues : une vue correspond à la description précise du processus interne de réalisation de l'activité de base. L'activité pédagogique capturée dans le composant peut faire intervenir différents acteurs aux comportements différents dans la réalisation de l'activité de base ; ainsi, chacun de ces comportements est décrit au travers d'une vue spécifique ;
- il est configurable au moment de son utilisation : certains paramètres génériques doivent être précisés/instanciés, ceci afin d'assurer une meilleure intégration de l'activité générique dans son environnement d'usage. Cette configuration permet aux composants éducatifs de s'adapter à différentes situations d'apprentissage ;
- il est personnalisable : plusieurs choix peuvent être possibles pour la réalisation de l'activité de base ; il est alors possible de préciser ces choix lors de l'utilisation du composant éducatif. Par

exemple, le mode de communication synchrone ou asynchrone peut être un choix proposé par un composant éducatif de conversation ; selon le mode choisi, différents services logiciels comme la messagerie instantanée (synchrone) ou comme la messagerie classique (asynchrone) seront choisis.

La configuration (réglages obligatoires) et la personnalisation (choix supplémentaires) peuvent être décrits également par une vue des modèles de conception ; ils peuvent également ne pas être précisés, reportant alors ces choix à la phase d'implémentation.

Pour résumer, du point de vue pédagogique, un composant CPL gère une activité pédagogique réutilisable, tandis que du point de vue technique, un composant CPL est un composant logiciel « métier » s'appuyant sur les autres composants logiciels de la plate-forme (Table 10.1).

Point de vue	**Perception du composant CPL**
pédagogique	un support à une activité élémentaire réutilisable
technique	un composant logiciel métier

TAB. 10.1 – Les différents points de vue définissant un composant CPL.

Vis-à-vis de ces différentes caractéristiques, le composant CPL correspond à la catégorie des composants éducatifs tels que nous les avons étudiés dans la section 3.3.3. L'IHM n'est pas considéré comme l'élément central de nos préoccupations. Ainsi, les composants CPL correspondent davantage aux composants *plugables* évoqués dans le cadre de l'architecture de la plate-forme OpenUSS [Dew00] qu'aux composants éducatifs logiciels tels que présentés par J. Roschelle dans le cadre des projets E-SLATE [Kou99] ou ESCOT [ESC].

10.2.1.2 Modélisation UML du composant CPL

Nous nous attachons maintenant à fournir une modélisation précise et concrète des composants CPL. Cette modélisation a pour objectif de préciser notre vision du composant CPL mais également d'expliquer en quoi ce nouveau type de composant va permettre de réduire l'écart entre les modèles pédagogiques de conception avancée indépendants de l'environnement de mise en œuvre et les services *traditionnels* fournis par les plates-formes.

La modélisation d'un composant CPL s'appuie sur le modèle de composant défini dans la spécification UML 2.0 ([OMG03c] section 8.3.1). Le modèle de composant d'UML 2.0 est justifié par le désir de l'OMG d'intégrer et de proposer dans le standard UML une amélioration significative du concept de composant en adéquation avec les nombreux travaux actuels issus de la communauté du génie logiciel et plus précisément de la communauté de recherche CBSE (*Component-Based Software Engineering*) [Szy02]. L'ensemble de nos travaux étant basés sur une application des techniques de modélisation et de méta-modélisation UML, nous avons donc choisi d'appliquer le modèle de composant décrit dans la spécification UML 2.0 pour décrire nos composants CPL. Toutefois, le modèle de composant d'UML 2.0 présente de nombreux manques dans la sémantique des nouveaux concepts et des incohérences entre les différentes notations proposées. Notre objectif n'est pas de rendre compte précisément de ces lacunes et d'y répondre mais de nous appuyer sur les concepts de composants les plus reconnus, comme les notions d'*interfaces fournies*

et *requises*, et d'occulter les concepts encore ambigus ou sujets à discussion comme la notion de *port* ou de *connecteur*.

Un composant CPL se modélise selon plusieurs vues :

- la vue externe « boîte noire » : cette vue décrit le composant en termes d'opérations rendues et nécessaires. Le modèle de composant d'UML 2.0 permet la définition de deux types d'interfaces : les interfaces fournies (illustrées par le symbole rond déjà présent dans les spécifications précédentes d'UML 1.X) et les interfaces requises (illustrées par le demi-cercle). L'application de ce modèle à nos composants CPL est illustrée par la Figure 10.3.
  Un composant CPL est alors un composant pouvant posséder :
  - une ou plusieurs interfaces requises de type *services logiciels* ;
  - zéro ou plusieurs interfaces fournies de type *services de configuration* ;
  - une ou plusieurs interfaces fournies de type *services pédagogiques*.

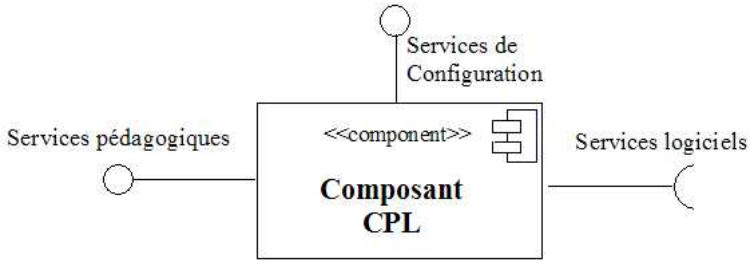

FIG. 10.3 – Vue externe « boite noire » d'un composant CPL [Laf03a] (modélisation UML 2.0).

- la vue comportementale : A chaque interface fournie de type « services pédagogiques » est rattaché son comportement décrit sous la forme d'un diagramme d'états-transitions. La vue comportementale spécifie alors explicitement comment les services « pédagogiques » sont réalisés par les services logiciels (sous la forme de contraintes dynamiques liées aux séquences d'appels d'opérations de ces interfaces).

10.2.1.3 Un exemple de modélisation d'un composant CPL

Un exemple simplifié de modélisation d'un composant CPL est illustré par la figure 10.4. Cet exemple concerne un composant CPL gérant une activité pédagogique élémentaire pour le monitorat : l'aide sur demande[131] (appelé « *Q/A* » pour *Question/Answer* dans l'illustration).

L'activité consiste à donner la possibilité à un apprenant de demander de l'aide au tuteur chargé d'assister l'activité principale. Le dialogue est à l'*initiative* de l'apprenant. Lorsqu'une demande, sous la forme d'un message, est reçue par le tuteur, alors ce dernier peut répondre à l'apprenant concerné. Les services logiciels requis peuvent correspondre à ceux fournis dans un *chat* traditionnel.

Le composant CPL (marqué dans l'exemple par un stéréotype « CPL » à la place de « component ») propose deux comportements pour la réalisation de cette activité[132] (un pour chaque acteur impliqué

[131] Le fonctionnement simplifié de cet exemple est volontaire.
[132] Cette activité n'est pas nécessairement collaborative.

dans l'activité). Chaque comportement est rattaché à une interface regroupant les services nécessaires à la réalisation de ce comportement :

- l'interface *QServices* regroupe les services pédagogiques dédiés à l'apprenant (l'initiateur de l'aide),
- et l'interface *AServices* rassemble les services dédiés au tuteur.

Les *statecharts* associés à chacune de ces interfaces sont respectivement *LearnerSide* pour l'interface *QServices* et *TutorSide* pour l'interface *AServices*.

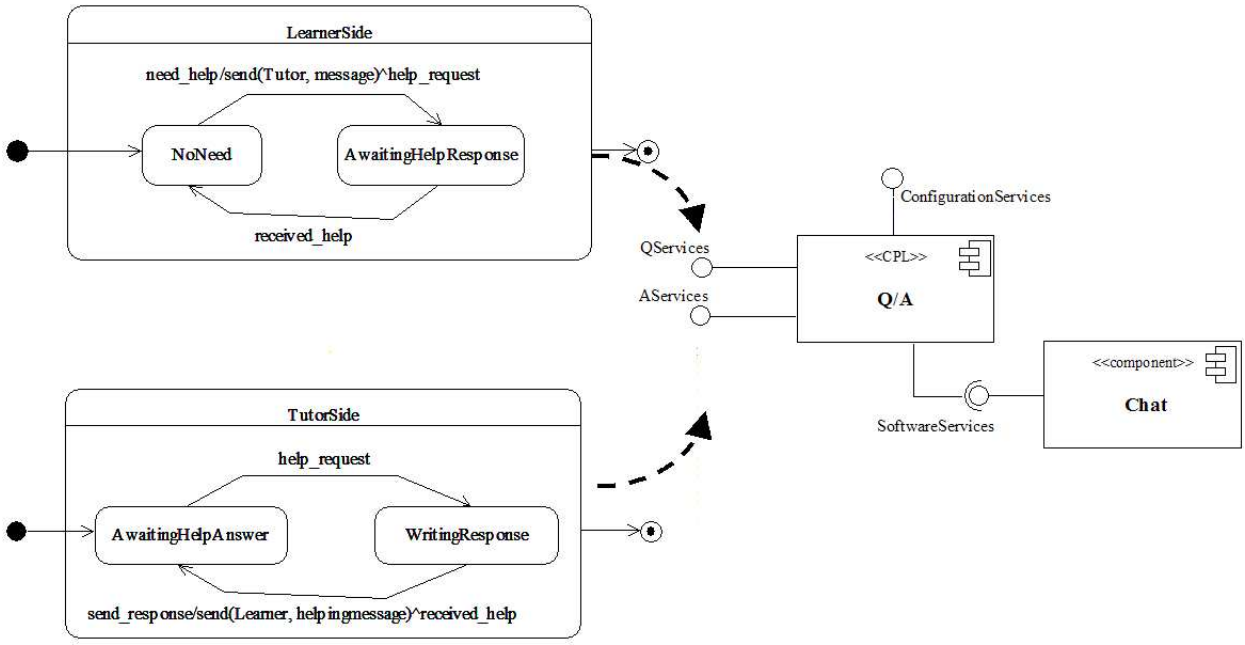

FIG. 10.4 – Un exemple de modélisation complet d'un composant éducatif ([Laf04].

Nous décrivons plus en détail l'utilisation des *statecharts* dans la sous-section suivante.

10.2.1.4 Utilisation des *State Machine* pour la description des activités réutilisables

D. Harel est à l'origine du formalisme des *statecharts* [Har87]. Ce formalisme visuel étend le formalisme conventionnel des diagrammes à états-transitions en leur ajoutant les trois notions d'emboîtement, de concurrence et de communication. Le langage obtenu permet alors de décrire la complexité des comportements de tout système réactif. Ce formalisme a été intégré dans UML à travers les diagrammes d'états ou *state machines*. Dans le domaine éducatif ce formalisme a été souvent utilisé pour, par exemple, modéliser des applications de type hypermédia [Pau99, deO01].

L'originalité de notre proposition réside dans l'utilisation de ce formalisme pour

- lier les services pédagogiques avec les services logiciels rendus par les LMS ;
- modéliser indirectement l'un des comportements impliqués dans l'activité pédagogique élémentaire.

Sémantique du *state machine* pour notre usage. Le *state machine*, rattaché aux interfaces de services pédagogiques, permet de définir l'ordre (*i.e.* la séquence) dans lequel les opérations de l'interface peuvent être invoquées (le comportement de chaque opération est défini par une méthode associée, c'est-à-dire l'un des services pédagogiques).

Une transition dans un *state machine* a comme déclencheur un événement d'appel (appelé *call event*) qui fait référence à l'une de ces opérations. Une transition permet donc d'indiquer que si l'événement d'appel survient quand une instance du composant CPL est dans l'état *source* de la transition et que la *garde* sur la transition est *vraie*, alors la méthode associée à l'opération de l'événement d'appel est exécutée (si elle existe), et l'instance entre dans l'état cible.

D'un point de vue sémantique, l'invocation de la méthode ne peut pas amener à générer un nouvel événement d'appel pour le même *state machine*. Si un événement d'appel survient alors que le *state machine* n'est pas dans l'état approprié alors l'événement est ignoré.

Voici de manière plus détaillée, la sémantique des éléments de modélisation composant le *state machine* dans leur usage avec les composants CPL :

- un *état* représente un *contexte* dans la réalisation de l'activité pédagogique ;
- une *transition* représente un *changement de contexte*. Chaque transition entre deux états (ou contextes) est de la forme :

  ```
  signature_d'événement [ condition-garde ] / expression_action
  ```

  La signature d'un événement est alors indiquée comme un nom et une liste d'arguments :

  ```
  nom-événement ( liste-de-paramètres )
  ```

  Le changement de contexte décrit par l'événement d'appel peut être à l'initiative :
  - d'une action de l'utilisateur (l'un des rôles impliqués dans l'activité pédagogique) dont le *state machine* décrit l'activité ; l'événement correspond dans ce cas à l'appel d'un service pédagogique décrit dans l'interface ;
  - de l'action d'un autre utilisateur (uniquement si l'activité élémentaire gérée dans le composant CPL implique plusieurs rôles) : on parle alors dans ce cas d'événement interne.

  La *garde* représente une condition requise pour valider le changement de contexte. La garde est évaluée uniquement après la réception de l'événement.

  L'*action* réalisée au passage dans la transition peut être composée de :
  - un événement d'appel correspondant à l'une des méthodes définies dans l'interface requise du composant CPL (l'un des services logiciels) ;
  - un événement interne à destination d'un autre *state machine* rattaché au même composant CPL
- les concepts de généralisation et d'agrégation d'états ainsi que de concurrence s'appliquent à la description de l'activité. Ils permettent de structurer et de hiérarchiser la conception de l'activité pédagogique de base.

Notation : retour à l'exemple précédent. Dans l'exemple précédent concernant le composant CPL *Q/A*, il y a deux *State Machine* rattachés aux deux interfaces fournies de services pédagogiques. Dans celui concernant l'activité de l'apprenant (*LearnerSide*), deux contextes sont définis : *NoNeed* et *AwaitinghelpResponse* ; le premier étant l'état initial. La seule transition possible, de cet état vers le second, attend l'événement d'appel *need_ help* correspondant à l'un des services pédagogiques de l'interface *QServices*. Lorsque cet événement survient, l'action de la transition est effectuée (il n'y a pas de garde) : la méthode *send(Tutor, message)* (l'un des services logiciels requis par l'interface *SoftwareServices*) est exécutée et l'événement interne *help_ request* est émis. L'état (le contexte dans l'activité) passe alors à *AwaitingHelpResponse*.

Dans ce contexte, il n'est permis aucune action à l'initiative de l'utilisateur pour changer de contexte et donc évoluer dans la réalisation de l'activité élémentaire. La seule transition possible (de *AwaitingHelpResponse* vers *NoNeed*) attend de recevoir l'événement interne *received_help*. Lorsque cet événement survient, la transition est effectuée et aucune action n'est réalisée ; l'état initial redevient l'état actif : une nouvelle question de l'apprenant peut être posée au tuteur.

Concrètement, le composant CPL, une fois implémenté et utilisé, doit fournir pour chaque interface de services pédagogiques une IHM adaptée, c'est-à-dire reflétant le contexte dans l'activité (les états actifs du *state machine* correspondant). Cette IHM doit proposer à l'utilisateur (apprenant comme tuteur) des éléments d'interface (boutons, menus, champ de texte, etc.) correspondant uniquement aux services pédagogiques pour lesquels un appel d'événement est attendu sur les transitions valides (liées aux états actifs). Par exemple, dans le cas du composant *Q/A* et pour l'IHM de l'apprenant, une première interface peut lui proposer un champ de texte pour taper le message de demande d'aide et un bouton pour valider l'appel. Le deuxième contexte peut proposer une seconde IHM sans le champ de texte et le bouton mais avec un texte indiquant l'envoi de la demande. Le retour à la première IHM sera automatiquement réalisé lorsque le tuteur aura répondu, sa réponse pouvant apparaître sous diverses formes.

Ainsi, les services pédagogiques proposés par ces nouveaux composants de LMS ne sont plus uniquement contextualisés pour une activité donnée (planification *prédictive*) mais intégrés à la réalisation *effective* de l'activité : les fonctionnalités sont dynamiques et dépendent des actions des utilisateurs.

10.2.2 Démarche de spécification des Composants CPL

Nous présentons maintenant une démarche pour aider la spécification des composants CPL. Cette démarche s'adresse autant aux concepteurs de PBL qu'aux développeurs de composants.

La démarche s'effectue en plusieurs étapes :

1. Expression des besoins en termes d'activités pédagogiques élémentaires réutilisables. Cette étape concerne l'équipe de conception de PBL et plus particulièrement l'ingénieur pédagogique dont le rôle est de formaliser la situation d'apprentissage.

1'. Abstraction des fonctionnalités rendues en grande majorité par les plates-formes LMS : cette étape aboutit à une description de « composants logiciels » encapsulant les fonctionnalités sous la forme de services (diagramme de composants UML « boîte noire »). Cette étape peut se dérouler dans le même temps que la première étant donné qu'elle concerne les développeurs de composants qui ont l'expertise nécessaire concernant le fonctionnement et l'architecture des plate-formes.

2. Définition et spécification des composants CPL, *via* notre modèle, sur la base des résultats des deux premières étapes :
   (a) description externe des composants CPL (diagramme de composants UML « boîte noire ») ;
   (b) description des vues comportementales des composants CPL (pour chaque interface de services pédagogiques).

   Cette étape concerne l'ingénieur pédagogique comme le développeur de composants.

La figure 10.5 résume cette démarche. Nous appelons *modèle « boîte grise »* la spécification d'un composant CPL regroupant la description externe et les vues comportementales.

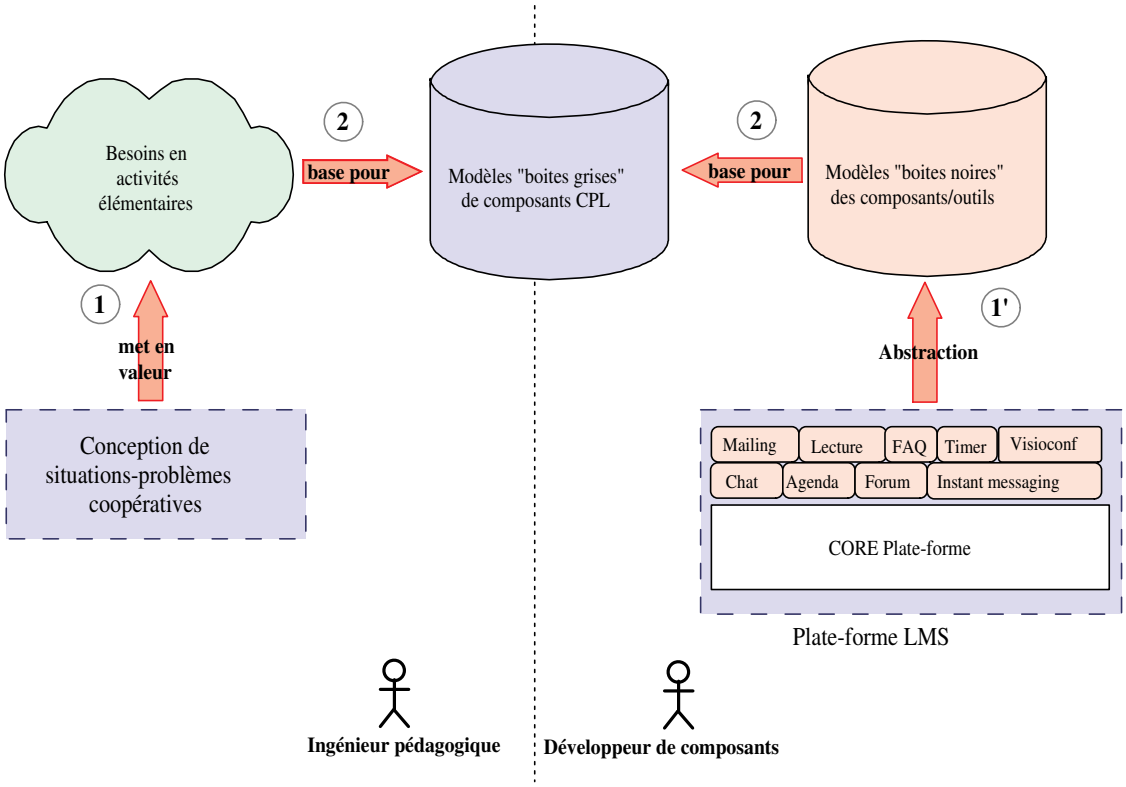

FIG. 10.5 – Notre démarche pour le développement de modèles pour les composants CPL.

L'abstraction des outils/services proposés par les LMS peut concerner de nombreuses catégories fonctionnelles des composants : composants de communication, composants de tutorat, ou encore composants d'administration (exemples tirés des composants proposés par la plate-forme Cybeosphere II [133]).

Nous proposons, dans un premier temps, de baser cette étape d'abstraction uniquement pour les composants de type chat, visio-conférence, messagerie électronique, etc., c'est-à-dire les outils de communication car :

1. ce sont des outils que les LMS proposent ;
2. ces outils sont les premiers à être implémentés sous la forme de composants logiciels.

Notre contribution concerne la proposition *théorique* d'un modèle pour les composants CPL. La démarche que nous proposons conjointement à notre modèle nécessite une identification des besoins pédagogiques et donc s'adresse également à des experts pédagogues.

10.2.3 Démarche d'utilisation : vers des nouveaux modèles de conception avancée

Nous nous intéressons maintenant à la démarche pour réutiliser les activités élémentaires décrites dans les modèles de composants CPL. L'hypothèse de départ est donc que l'ingénieur pédagogique dis-

[133] *cf.* la documentation disponible à l'URL : http ://www.cybeosphere.fr

pose d'une *bibliothèque* d'activités pédagogiques élémentaires (à chacune de ces activités correspond un modèle de CPL construit selon la démarche précédente décrite en § 10.2.2). Il s'agit alors, pour l'ingénieur pédagogique et les autres concepteurs de PBL coopératives, de choisir parmi les activités de base réutilisables celles qui conviennent pour décrire les activités de la situation d'apprentissage (Figure 10.6). Ainsi, de nouveaux modèles de conception avancée seront spécifiés.

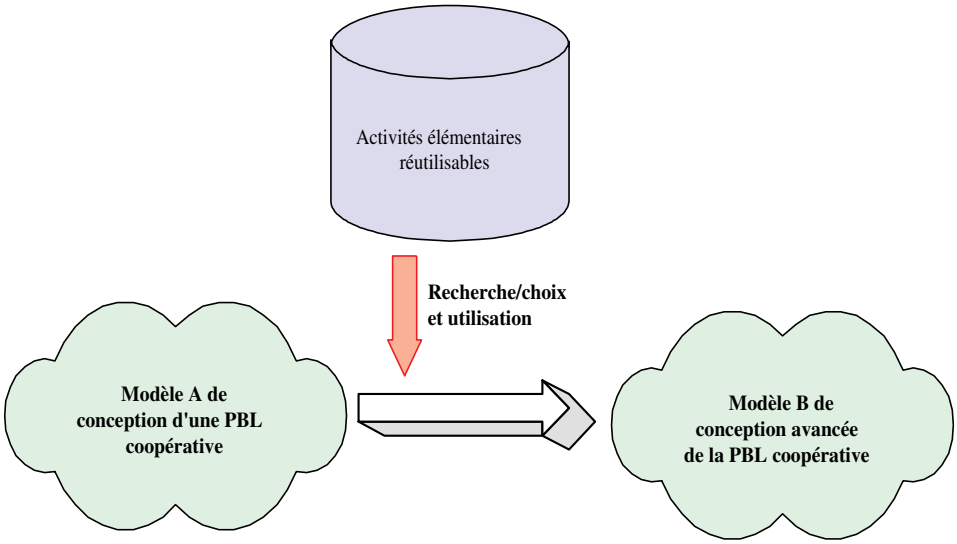

FIG. 10.6 – Démarche pour l'utilisation des modèles de CPL pour décrire de nouveaux scénarios d'apprentissage.

Du point de vue de l'équipe de conception, seule la vue externe des composants CPL est intéressante. En effet, cette vue renseigne les concepteurs sur le type de l'activité de base et sur les services pédagogiques rendus. Il s'agit alors de proposer une extension au langage CPM afin de prendre en compte ces composants CPL comme activités de base. Nous avons également choisi d'enrichir CPM afin qu'il permette de modéliser les vues externes et comportementales des composants CPL. L'utilisation du langage CPM pour modéliser les composants CPL s'adresse alors aux *développeurs de composants* (voir rôles définis en 1.2.2.2).

Pour les mêmes raisons de distinction sémantique/notation que pour l'élaboration initiale du langage CPM, nous proposons de présenter dans un premier temps les extensions du méta-modèle CPM, puis, dans un second temps, de présenter les extensions du profil CPM.

10.2.3.1 Extension du méta-modèle CPM

L'extension se réalise sur deux plans :

Ajout du paquetage CPL : ce paquetage définit les concepts liés à la description des composants CPL (Figure 10.7).

Les méta-classes **Component** et **Interface** proviennent du paquetage **CPM_Foundation : :Core** et correspondent aux éléments définis dans UML 1.5. Les méta-classes **ProvidedInterface** et **RequiredInterface** sont des éléments que nous avons rajoutés afin d'adapter le méta-modèle d'UML

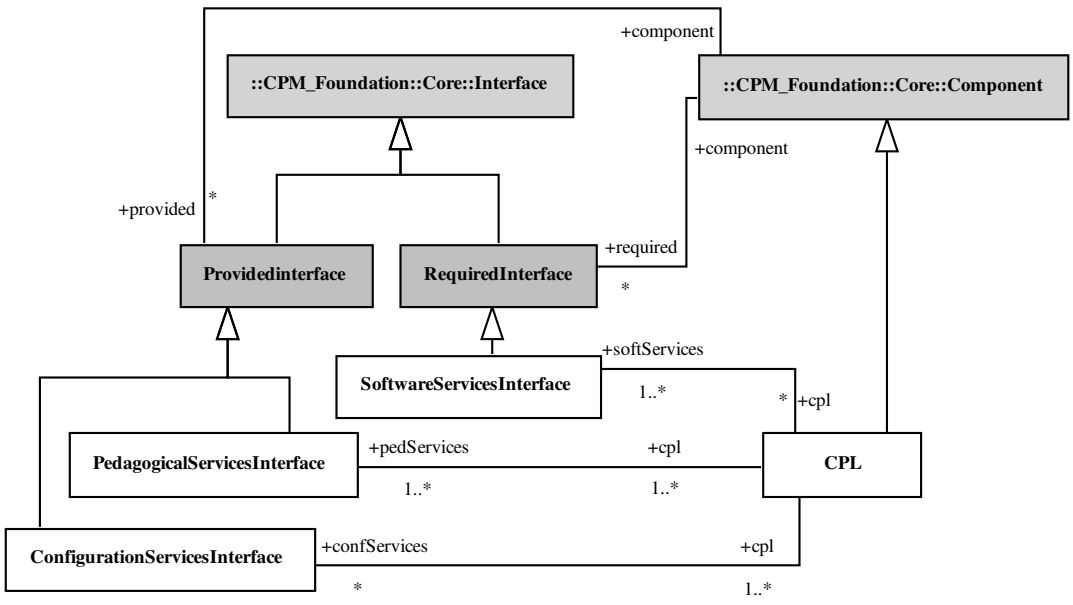

FIG. 10.7 – Le paquetage des composants CPL.

1.5 à ces concepts apparus dans la spécification d'UML 2.0[134]. Ensuite, nous proposons les concepts d'interfaces de services pédagogiques, d'interfaces de services de configuration et d'interfaces de services logiciels conformément à la spécification décrite en 10.2.1.2. Un composant **CPL** est alors défini comme une spécialisation de **Component** ; il peut posséder zéro ou plusieurs **ConfigurationServicesInterface**, et de une à plusieurs **PedagogicalServicesInterface** et **SoftwareServicesInterface**.

Lien entre les activités pédagogiques et l'activité de base représentée par le composant CPL : nous introduisons le concept **CPLactivity** qui spécialise **Activity** (telle que définie en 7.3.4.4) et qui correspond à l'activité proposée par un composant CPL (Figure 10.8).

[134]Ceci nous permettra également, dans le profil CPM amélioré, d'ajouter les notations d'UML 2 correspondantes à ces interfaces.

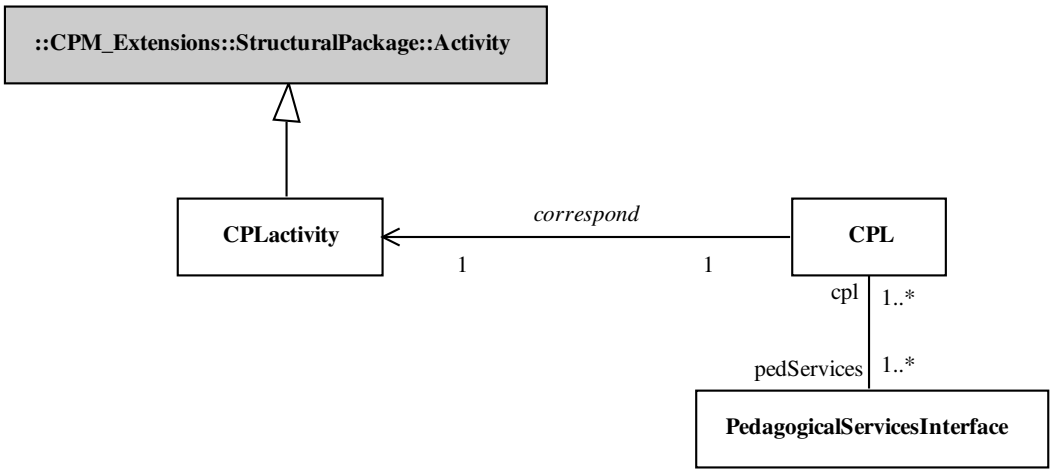

FIG. 10.8 – Le paquetage des composants CPL : liaison avec le concept d'activité.

10.2.3.2 Extension du profil CPM

Les nouveaux stéréotypes et définitions de valeurs marquées sont donnés dans les tableaux 10.2 et 10.3. La démarche pour les obtenir est similaire à celle que nous avons présentée en section 5.3.3.3 et appliquée pour le profil CPM (chapitre 8).

Stéréotype	**Méta-classe**	**Classe-parente**	**Icône**
ProvidedInterface	`Core : :Interface`		
RequiredInterface	`Core : :Interface`		
CPL	`Core : :Component`		
PedagogicalServices-Interface	`Core : :Interface`	Provided-Interface	
ConfigurationServices-Interface	`Core : :Interface`	Provided-Interface	
SoftwareServices-Interface	`Core : :Interface`	Required-Interface	
CPLactivity	`Core : :Operation` `ActivityGraphs : :ActionState`	Activity	CPL

TAB. 10.2 – Les nouveaux stéréotypes du langage CPM.

Valeur marquée	Type	Sur stéréotype	Description
pedInterface	String	CPLactivity	permet de décrire quelle interface de services pédagogiques (rendue par le composant CPL correspondant) est concernée

TAB. 10.3 – La nouvelle valeur marquée du langage CPM.

Le rôle de la valeur marquée *pedInterface* est de permettre aux concepteurs de préciser quelle interface de services pédagogiques du composant CPL est concernée : c'est-à-dire quel est le comportement (dans le cas où il y en a plusieurs) attendu dans la réalisation de l'activité élémentaire.

Concernant les nouvelles notations, le diagramme de classes est utilisé pour représenter la vue externe des composants CPL ainsi que les composants/outils de la plate-forme ; le diagramme d'états-transitions permet de spécifier les vues comportementales des interfaces de services pédagogiques des composants CPL ; ensuite, **CPLactivity** suit la même notation qu'**Activity**. Des exemples de notation sont donnés dans la section suivante.

10.2.3.3 Remarques sur la démarche

La démarche d'utilisation est basée sur l'hypothèse de l'existence d'une *bibliothèque* d'activités élémentaires réutilisables pour la conception avancée de situations-problèmes coopératives. Ces activités sont gérées par les composants CPL.

Bien que les modèles de composants CPL dépendent des plates-formes (puisqu'ils font référence à des services logiciels précis), ils sont indépendants de l'implémentation concrète du composant CPL. En effet, l'implémentation est basée sur le modèle CPL mais dépend des spécificités techniques et technologiques de la plate-forme cible.

Ainsi, les modèles de conception avancée spécifiés pour les PBL sont indépendants de toute plate-forme particulière : un changement de plate-forme n'invalide pas le modèle de conception avancée d'une PBL (propriété d'interopérabilité).

10.3 Exemples

Dans un premier temps, nous donnons un exemple complet de spécification d'un composant CPL (10.3.1). Cette spécification s'appuie sur UML et le langage CPM. Les diagrammes de cette section ont été réalisés avec l'outil *Objecteering UML Modeler* dans lequel nous avons importé une version étendue de notre module **CPM** (avec de nouvelles fonctionnalités pour faciliter la mise en œuvre des nouveaux modèles). Ces exemples s'adressent aux développeurs de composants.

Puis, nous présentons de manière synthétique les différents travaux d'expérimentation que nous avons menés quant à l'implémentation concrète de composants CPL (10.3.3). Cette section s'adresse également aux développeurs de composants.

Finalement, nous présentons des exemples de conception avancée de SMASH pour lesquels nous utilisons les activités CPL élémentaires (10.3.2). Ces exemples s'adressent à l'équipe de conception.

10.3.1 Spécification/modélisation d'un composant CPL

L'exemple de spécification de composant CPL que nous proposons est celui d'une autre activité élémentaire : la gestion de conflit. Cette activité est basée sur l'utilisation d'un composant traditionnel de communication (le *chat*). La gestion de conflit a pour but d'établir une conversation synchrone entre deux rôles d'apprenants de la PBL afin qu'ils puissent s'expliquer sur une divergence d'opinion ou bien pour essayer de résoudre un problème particulier. Cette conversation peut être à l'initiative d'un rôle de tuteur ; celui-ci supervise également la gestion du conflit en ayant la possibilité d'intervenir dans le débat afin de diriger ou recadrer la conversation vers son objectif.

Dans les prochaines sous-sections, nous suivons la démarche proposée en 10.2.2 : tout d'abord nous modélisons à un haut niveau d'abstraction un composant de *chat* par un diagramme UML de composants (10.3.1.1), puis nous modélisons le composant CPL *Gestion de conflit* en nous servant de la modélisation précédente (10.3.1.2).

10.3.1.1 Modélisation du composant *chat*

Cette abstraction des fonctionnalités traditionnelles d'un chat est inspirée des travaux de Geir Melby [Mel03, Mel04] sur UML 2.0 et *UML executable* dans lesquels il illustre ses propos sur l'exemple d'une application *chat*.

Le service de *chat* tel que modélisé ici suit les règles suivantes :
- il peut y avoir plusieurs *salles* de discussion (*chat rooms*) ;
- un client peut seulement être connecté à une seule salle, à un moment donné ;
- les salles sont créées dynamiquement sur la demande d'un utilisateur ;
- quand il n'y a plus de client connecté à une salle, celle-ci est détruite.

La figure 10.9 illustre cette modélisation : à gauche se trouve une modélisation *boîte noire* du composant *chat* et à droite sont détaillées les deux interfaces fournies par ce composant (*IManager* et *IChat*).

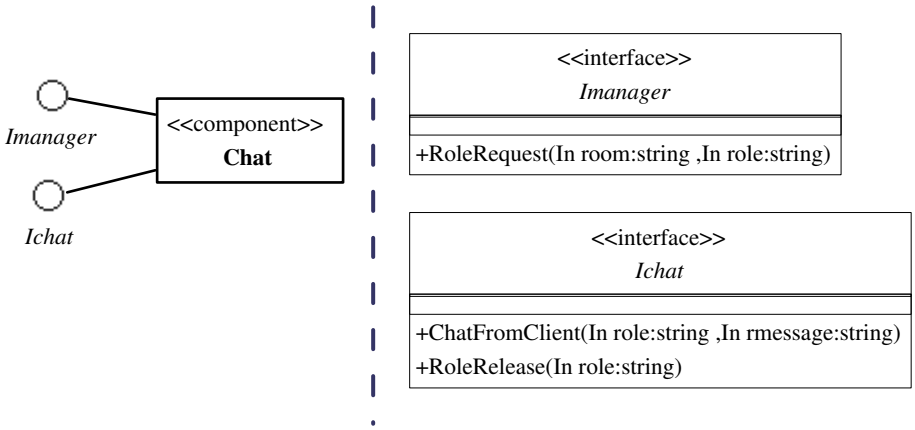

FIG. 10.9 – Abstraction des fonctionnalités d'un *chat* et modélisation UML.

IManager : cette interface propose une unique méthode *RoleRequest* qui prend en paramètre le nom de la salle, puis le pseudonyme pour la discussion. Si la salle n'existe pas alors elle est créée et le pseudonyme est rattaché à cette salle, sinon le pseudonyme est rattaché à la salle existante.

IChat : cette interface rassemble les méthodes utilisées lorsque l'on est rattaché à une salle.

- la méthode *ChatFromClient* permet d'envoyer un message : elle prend en paramètre le pseudonyme et le message ;
- la méthode *RoleRelease* permet de quitter une salle : elle prend en paramètre le pseudonyme. Si la salle est vide alors elle est détruite.

10.3.1.2 Modélisation du composant CPL *Gestion de conflit*

Description externe. Deux interfaces de services pédagogiques sont proposées par le composant CPL de *Gestion de conflit* : l'interface *Ituteur* et l'interface *Iapprenant* (Figure 10.10).

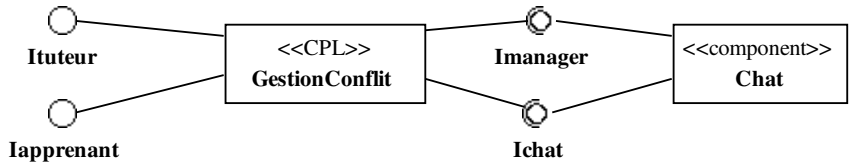

FIG. 10.10 – Modélisation « boîte noire » du composant CPL de *Gestion de conflit* sur la base du composant logiciel *Chat*.

L'interface *Ituteur* rassemble trois méthodes pour le tuteur (Figure 10.11) :

- *CreerDiscussion()* : elle permet de créer une salle de *chat* pour la conversation entre deux rôles d'apprenants ; elle prend en paramètre le nom des rôles des apprenants impliqués et le nom d'une salle ;
- *IntervenirTuteur()* : elle permet au tuteur d'intervenir dans la conversation entre les deux rôles d'apprenants ; elle prend en paramètre le message ;
- *Terminer* : cette méthode ferme la salle de conversation donnée en paramètre.

L'interface *Iapprenant* rassemble deux méthodes pour les apprenants (Figure 10.11) :

- *Intervenir()* : elle permet à l'apprenant d'ajouter un message dans la conversation ; elle prend en paramètre le nom du rôle joué et le message ;
- *Conclure()* : cette méthode permet à l'apprenant de terminer la conversation (son nom de rôle est passé en paramètre).

Nous définissons également deux signaux *Début* et *Fin* comme événements internes servant à la description comportementale.

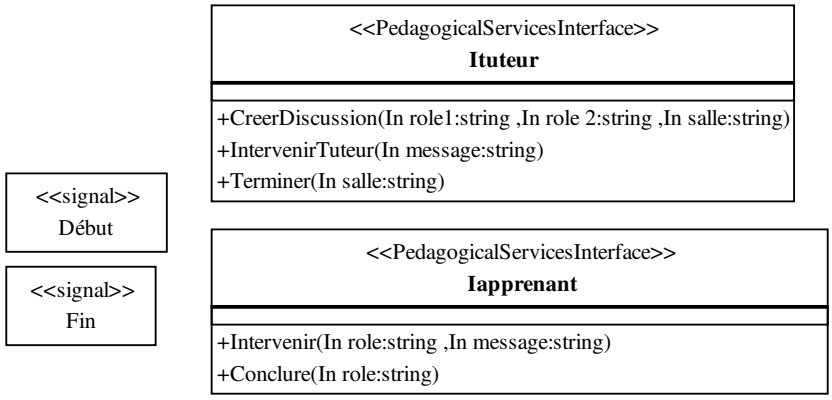

FIG. 10.11 – Détail des méthodes des interfaces du composant CPL de *Gestion de conflit.*

Description comportementale. Les *State Machine* décrivant chacun des comportements pour les deux interfaces précédentes sont décrits dans les figures 10.12 et 10.13.

Le *State Machine* destiné au tuteur comporte trois contextes d'activité. Le contexte (ou état initial) est intitulé *Pas de discussion.* Dans ce contexte, une seule transition est possible vers l'état *Discussion créée* ; elle est initiée par le tuteur *via* la méthode *CréerDiscussion()* pour laquelle il désigne les deux rôles participant à la gestion du conflit et le nom de la salle (pour le *chat*) à laquelle ils participeront. L'action réalisée au passage de cette transition consiste alors en une séquence de trois appels successifs à la méthode *RoleRequest()* fournie par le composant logiciel de *chat.* Le débat entre les deux apprenants ne peut commencer qu'après une intervention du tuteur expliquant par un message le sujet du débat (appel à la méthode *IntervenirTuteur()*) ; l'action correspondante consiste alors à faire un appel à la méthode *ChatfromClient()* pour envoyer le message du tuteur dans la salle correspondante ; l'événement interne *Début* (correspondant au signal défini précédemment) est également émis pour l'autre *State Machine.* L'état actif devient alors *Débat lancé.* Deux transitions sont possibles : une intervention du tuteur pour envoyer un message (transition vers le

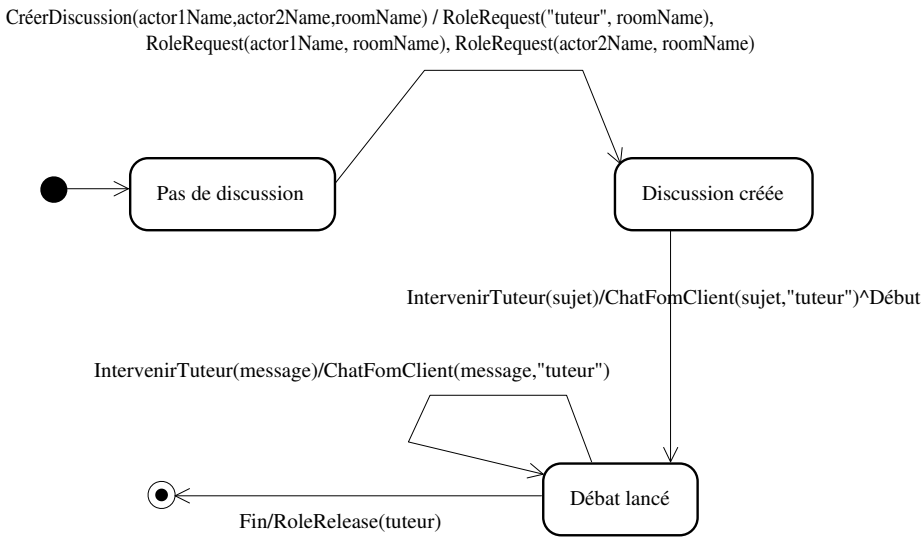

FIG. 10.12 – Le *State Machine* de l'interface *Ituteur*.

même état) ou alors l'événement interne *Fin* est reçu et une transition vers l'état final est réalisée (l'action correspondante consiste à libérer la salle par un appel à la méthode *RoleRelease()*).

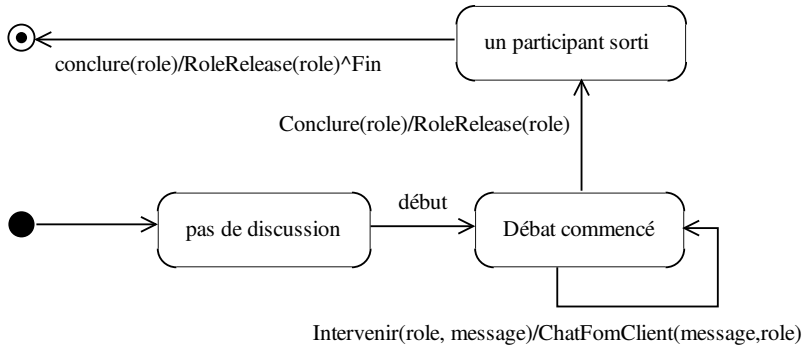

FIG. 10.13 – Le *State Machine* de l'interface *Iapprenant*.

Le second *State Machine*, destiné aux apprenants, comporte trois contextes d'activité. Le contexte (ou état initial) est intitulé *Pas de discussion*. Dans ce contexte, une seule transition est possible vers l'état *Débat commencé* ; elle est initiée lors de la réception de l'événement interne *Début*. Ensuite, les interventions des apprenants consistent à l'appel de la méthode *Intervenir()* dont l'action correspondante consiste alors à faire un appel à la méthode *ChatfromClient()*. Lorsque l'un des apprenants fait appel à la méthode *Conclure()* l'état actif devient alors *un participant sorti*. Un deuxième événement identique (indiquant la sortie du deuxième participant) a pour effet d'émettre

le signal *Fin* afin que le tuteur puisse terminer le débat.

10.3.2 Modèle de conception avancée réutilisant les composants CPL

Nous proposons dans la figure 10.14 un exemple illustrant l'utilisation des composants CPL pour décrire de nouveaux modèles de conception avancée. Il s'agit encore d'un exemple lié au cas SMASH. Dans la figure, nous décrivons un diagramme d'activités. Toutefois, le concept de **CPLactivity** n'est pas limité à ce diagramme mais peut être utilisé dans tous les diagrammes dans lesquels le concept d'**Activity** est permis.

Dans l'exemple, *enquêteur 1* et *enquêteur 2* doivent participer à une activité collaborative consistant à positionner des représentations des témoins de l'accident de vélo sur un fond de carte (*cf.* annexe C). Ils utilisent alors un tableau blanc pour réaliser cette activité. Le tuteur doit pendant ce temps les assister dans leur travail ; il utilise donc lui aussi le même tableau blanc (nous masquons sur le diagramme les droits d'utilisation sur cette même ressource ainsi que la contrainte **CollaborationMode** reliant ces trois activités). En parallèle de leur activité, nous proposons aux trois rôles l'activité élémentaire de *Gestion de conflit* telle que décrite dans la section précédente. La valeur marquée *pedInterface* est appliquée aux trois **ActionState** stéréotypés <<**CPLactivity**>> : l'*enquêteur 2* comme l'*enquêteur 1* utilisent l'interface *Iapprenant* tandis que le *chef de la police* utilise l'interface *Ituteur* (seule la valeur marquée pour l'*enquêteur 2* est visible[135]).

10.3.3 Implémentation des composants CPL

Nous avons réalisé divers travaux afin d'expérimenter l'implémentation des composants CPL. Tous ces travaux sont basés sur l'utilisation d'une librairie Java permettant l'implémentation des *statecharts* (un exemple de codage d'un *State Machine* est donné dans la figure 10.15 : il correspond au *LearnerSide* pour l'exemple précédent du composant *Question/Answer*).

Les différentes expérimentations nous ont amenés à :
- valider l'utilisation des *State Machine* pour proposer à l'utilisateur, pendant la phase d'exécution, une IHM dynamique offrant des services selon le contexte dans l'activité ;
- implémenter le composant CPL *Question/Answer* sous la forme d'une application Web (*Servlet*) ;
- commencer un travail d'ajout du composant *Q/A* à la couche des composants intégrables de la plate-forme de formation à distance OpenUSS.

La figure 10.16 présente différentes captures d'écran successives liées à l'utilisation du composant *Question/Answer* par deux rôles différents (côté gauche de la figure, l'initiateur des questions et côté droit, celui qui répond). Nous pouvons remarquer que le *répondeur* doit attendre l'arrivée d'une question afin que son IHM lui permette d'envoyer un message. De même, l'initiateur doit attendre de recevoir une réponse avant d'envoyer un nouveau message.

[135] L'outil Objecteering ne permet pas d'afficher l'icône d'un stéréotype si des valeurs marquées sont rendues visibles ; c'est pour cette raison que nous laissons la représentation sous forme d'icône aux autres **CPLactivity** et que nous ne détaillons la visibilité que d'une seule.

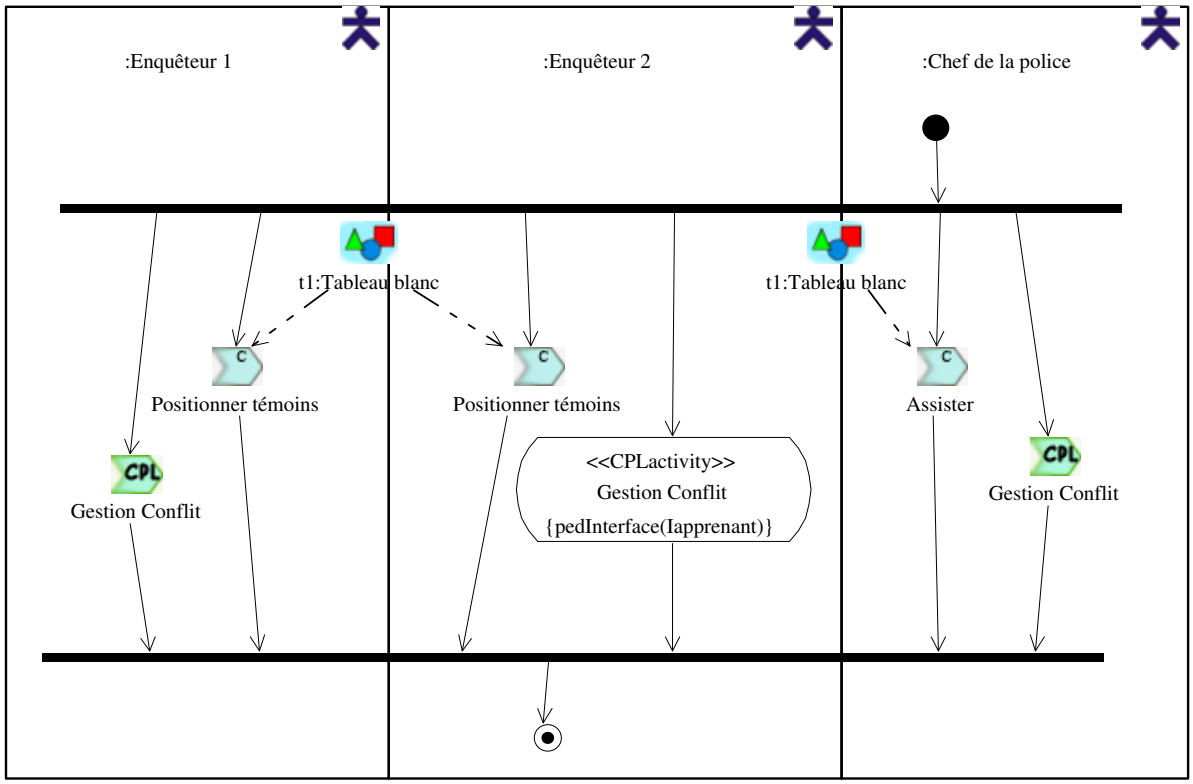

FIG. 10.14 – Exemple de diagramme d'activité montrant l'utilisation du composant CPL de *Gestion de conflit*.

```
public Statechart_monitor build_statechart(){
  // Statechart_monitor extends Statechart class with new transition features
  Statechart_monitor s=null;
  Statechart s1,s2,s3=null;
  try {
     s1=new Statechart("No need");
     s2=new Statechart("Awaiting help response");
     // "xor" composition of the two previous states into a new state
     s3=s1.xor(s2);
     s3.set_name("grp1");
     s=new Statechart_monitor(s3,"CET");
     Object[] o=new Object[1];
     o[0]=(Object)new String("Answer reception");
     String[] internal_events={};
     s.set_transition(s2,s1,"receipt_response",false,internal_events,
                      System.out, "println",o);
     Object[] o2=new Object[1];
     o2[0]=(Object)new String("sending of the question");
     String[] internal_events2={"receipt_question"};
     s.set_transition(s1,s2,"send_question",true,internal_events2,
                      System.out,"println",o2);
     try {
        s1.inputState(); //s1 is the entry state
     } catch(_Statecharts.StatechartException se){};
  } catch (_Statecharts.StatechartConfigurationException ss){};
  return s;
}
```

FIG. 10.15 – Exemple de code Java pour la création de *statecharts*

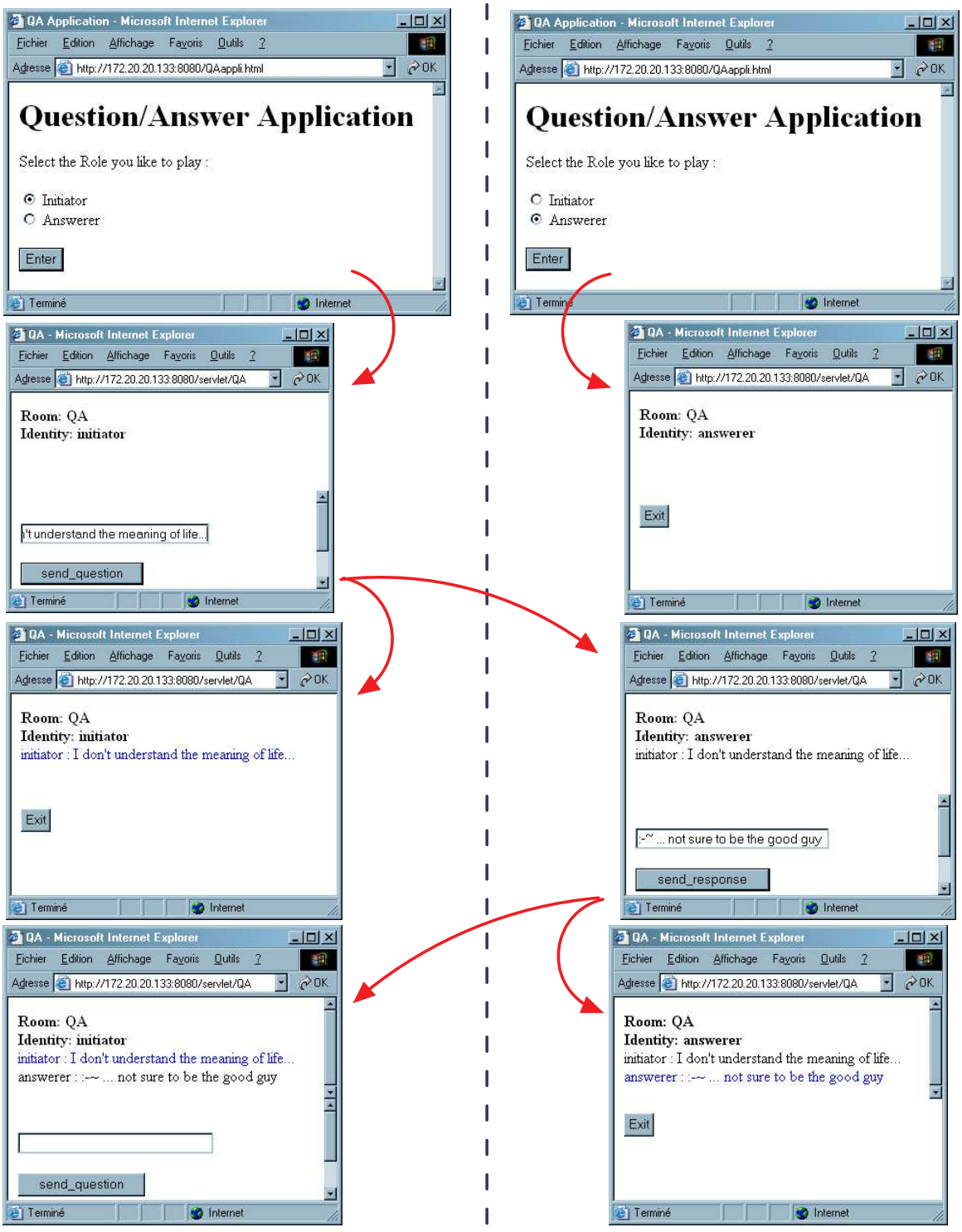

FIG. 10.16 – Captures d'écran montrant les différentes IHM pendant l'exécution dynamique du composant CPL *Question/Answer*.

Chapitre 11

Conclusion générale

Sommaire

Dans cette thèse, nous nous sommes intéressés principalement à l'élaboration d'un langage permettant la mise en œuvre de modèles pour la phase de conception de formations à distance de type situations-problèmes coopératives. Nous concluons, maintenant, en présentant un bilan des travaux réalisés, puis, nous mettons en évidence les apports de notre recherche. Enfin, nous terminons en exprimant et en discutant des perspectives pour ce travail.

11.1 Bilan des travaux réalisés

Notre contribution se présente sous la forme de deux propositions complémentaires : un langage de modélisation CPM (*Cooperative PBL Metamodel*) et un modèle de composant éducatif CPL (Composant Pédagogique Logiciel).

11.1.1 Langage CPM

Nous avons proposé un langage facilitant l'élaboration de modèles pour la conception de situations-problèmes (PBL) sur des plates-formes de formation à distance. L'étude des différents langages correspondant en partie à cet objectif a permis de mettre en évidence i) le manque actuel de modèle et de langage pour l'étape d'expression initiale des besoins, ainsi que ii) la nécessité d'utiliser UML pour décrire des

modèles abstraits (étape d'analyse) servant de base à la spécification formelle réalisée par les EML comme IMS-LD (étape de conception avancée). Ce constat nous a alors amené à positionner notre proposition comme un langage de modélisation graphique spécialisant UML pour la conception de PBL en amont des langages de type EML : il couvre alors la phase de conception pour les étapes d'expression initiale des besoins, d'analyse et de conception.

Le langage CPM proposé permet de décrire les différents modèles de la PBL pour les phases amont de conception. Ces modèles aident l'équipe pluridisciplinaire de conception à décrire, spécifier et documenter la PBL qu'ils conçoivent. De plus, le caractère visuel des modèles produits permet d'abstraire la complexité de la PBL mais constitue également une représentation adéquate pour favoriser le dialogue, la compréhension et l'implication des différents intervenants. CPM répond à des pré-requis identifiés de personnalisation, de reproductibilité, de réutilisation, d'indépendance du médium/configuration, et d'indépendance vis-à-vis des plates-formes de formation à distance.

Le langage CPM propose une syntaxe (terminologie et notation) et une sémantique adaptées afin de satisfaire l'ensemble des différents modèles possibles pour la phase de conception. La syntaxe *abstraite* est capturée dans le méta-modèle CPM (chapitre 7), tandis que la syntaxe *concrète* est décrite au travers du profil CPM (chapitre 8). La sémantique des différents concepts et relations proposés est donnée *via* des contraintes OCL et des explications en langage naturel.

Un prototypage d'environnement-auteur pour le langage CPM a été proposé sur la base d'une implantation du profil CPM dans l'outil *Objecteering Profile Builder* et d'une personnalisation/adaptation de l'interface de l'AGL pour faciliter la création et le suivi de modèles CPM. Ce type d'environnement concerne plus particulièrement l'ingénieur pédagogique pour lequel un niveau de connaissance suffisant d'UML est nécessaire. Concrètement, le langage CPM est actuellement disponible sous la forme d'un module intégrable à l'AGL UML *Objecteering Modeler*. Nous avons également initié un travail de transformation de modèle afin d'expérimenter et de garantir de nouveaux usages qui permettront de ne pas restreindre les modèles élaborés avec le langage CPM à un usage contemplatif.

Finalement, nous avons mis à l'essai le langage CPM sur le cas d'étude SMASH dans lequel des enfants du cycle 3 jouent le rôle d'inspecteurs de police enquêtant sur la reconstitution d'un accident de vélo. Cette expérimentation a abouti à la mise en œuvre de modèles pour les différentes étapes de la phase de conception. Ce travail a permis de valider la richesse graphique et la grande variété de modèles possibles pour la description des différentes perspectives d'une situation-problème coopérative.

Cette première proposition a fait l'objet de nombreuses publications présentant l'évolution itérative et incrémentale de nos travaux sur le méta-modèle CPM [Nod03a, Nod03b] ainsi que sur le profil CPM [Laf03b] et son usage [Nod04, Sal05].

Le langage CPM, permet la modélisation de scénarios pédagogiques indépendamment des fonctionnalités des plates-formes. La seconde proposition étend notre contribution en prenant en compte ces plates-formes *via* les composants CPL.

11.1.2 Modèle de composant CPL

Le deuxième objectif de la thèse consiste à prendre en compte l'évolution actuelle des plates-formes (application du *savoir-faire composant*) et à identifier en quoi les composants peuvent améliorer la conception des formations.

L'étude que nous avons réalisée nous a permis de constater que les objectifs visés par cette tendance sont, dans un premier temps, de suivre les avancées technologiques en appliquant les savoirs-faire technologiques (J2EE, EJB, Web Services par exemple) et fonctionnels (structuration en couches) en terme de composants logiciels traditionnels ; ceci afin d'étendre, dans un deuxième temps, les fonctionnalités et usages des plates-formes (LMS).

Actuellement, les LMS basés composants n'en sont qu'à la première étape : ils fournissent les mêmes fonctionnalités qu'auparavant (outils de communication, production, coordination, conversation) mais sous forme de composants logiciels. Du point de vue des standards proposant la spécification d'unités d'apprentissage interopérables (quels que soient les LMS les exécutant), le LMS n'est perçu qu'en termes d'objets pédagogiques et de services logiciels fournis. Nous pensons que la prochaine étape visant à étendre les fonctionnalités et les usages de ces plates-formes doit prendre en compte les besoins des concepteurs et ainsi doit participer à diminuer l'écart existant actuellement entre les besoins liés à des attentes d'ordre pédagogique et les services logiciels standards fournis par les LMS.

Nous avons donc proposé un modèle pour la spécification d'un *composant éducatif* appelé CPL (Composant Pédagogique Logiciel). Ce composant CPL vise à réduire l'écart en répondant aux exigences des concepteurs comme aux développeurs de composants cherchant à étendre les fonctionnalités des LMS.

Du point de vue des concepteurs d'unités d'apprentissage, le composant CPL représente une activité pédagogique élémentaire et non un outil. Cette activité est une brique de base réutilisable pour la conception d'activités pédagogiques plus complexes pour les apprenants comme pour les tuteurs. Les services pédagogiques fournis par ces composants sont dynamiques et dépendent du comportement de l'utilisateur réalisant l'activité. En ce sens, les services ne sont plus transversaux aux activités d'apprentissage, comme ils le sont actuellement dans les LMS, mais intégrés à l'activité. Du point de vue du développeur de composant, le composant CPL représente un nouveau composant logiciel « métier » construit sur la base des composants fonctionnels traditionnels des LMS ; le principe étant de contraindre les outils existants pour en bâtir de nouveaux plus adaptés aux besoins des concepteurs.

Le modèle adapté aux composants CPL que nous proposons est basé sur le concept de composant d'UML 2.0. Ce modèle permet d'aider les développeurs de composants à décrire, spécifier, puis développer ces composants CPL mais il permet surtout aux concepteurs de situations-problèmes de spécifier de nouveaux modèles de conception avancée pour leurs scénarios pédagogiques.

Le langage CPM a également été étendu afin de prendre en compte le concept d'activité élémentaire réutilisable et de permettre ainsi l'élaboration des nouveaux modèles de conception avancée. Nous l'avons également étendu afin qu'il puisse servir aux développeurs de composants et aux concepteurs pour supporter leur spécification des composants CPL.

Divers exemples, illustrations, expérimentations et prototypes logiciels nous ont permis de vérifier et valider cette seconde proposition. Bien que nous ayons publié ces différents résultats [Laf03a, Laf04], nos travaux doivent encore être approfondis.

11.2 Apports de nos travaux de thèse

Nos travaux répondent en partie à de nombreuses problématiques actuellement étudiées dans la communauté française des EIAH.

Concernant le cadre de recherche général de l'ingénierie des EIAH, nous nous sommes focalisés dans ce travail de thèse sur la phase de conception d'apprentissage pour la formation à distance. Il s'agit d'un contexte dans lequel les travaux « orientés pédagogie » [Tch02b] à forte connotation d'ingénierie sont actuellement les plus nombreux. Plus précisément, notre travail a permis d'identifier et de décrire l'apport de la méta-modélisation UML, *via* la technique des profils UML, pour spécifier des scénarios d'apprentissage. Nous avons montré que l'utilisation d'un profil UML amène à une conceptualisation explicite des situations d'apprentissage : les stéréotypes et les valeurs marquées décrivent les concepts et les relations sous-jacents. Ceci permet une meilleure capitalisation des connaissances et améliore la communication entre les différents intervenants de l'équipe pluridisciplinaire de conception. De plus, les modèles bâtis avec le langage CPM peuvent exploiter les possibilités d'UML en termes de transformation de modèles : exportation en XML *via* XMI ou encore génération de code, ce qui confère aux modèles une ouverture pour de plus larges usages : passerelle vers IMS-LD, mise en œuvre d'un système d'information pour exploiter les modèles comme outils de régulation de l'apprentissage, etc.

Nous avons réalisé un travail en *profondeur* dans le sens où nous avons focalisé nos travaux pour un type particulier d'apprentissage (les situations-problèmes) en articulation à un dispositif informatique particulier (les plates-formes de formation à distance). Toutefois, les résultats obtenus (le langage de conception CPM) comme les démarches pour les obtenir (élaboration d'un méta-modèle, transformation en profil UML, élaboration de modèles en utilisant un profil) sont généralisables et reproductibles pour d'autres types de formations ainsi que pour d'autres supports informatiques.

Nous avons positionné dans un premier temps notre proposition du langage CPM en amont des langages de type EML comme la spécification IMS-LD ; dans un second temps, nous avons proposé un type de composant éducatif (le composant CPL) afin de réduire l'écart entre les besoins des concepteurs et les fonctionnalités proposées par les plates-formes mais également afin de proposer de nouveaux modèles de conception avancée. Le positionnement de nos travaux est un travail préliminaire d'ingénierie qui doit être complété ensuite par une recherche en *largeur* visant à considérer les dimensions d'abstraction, de généralisation, de capitalisation et de réutilisation de nos travaux pour un cadre plus large.

Les travaux de cette thèse s'inscrivent directement dans les actions de recherche engagées par l'équipe IDÉE au sein de la communauté EIAH : participation à l'Action Spécifique « Conception d'une Plate-forme Technologique pour la Recherche en EIAH »[136] (action souhaitée par le RTP39 « Apprentissage, Éducation, Formation »). Dans le cadre de cette AS *plate-forme*, le cas d'étude SMASH va constituer l'un des scénarios pédagogiques de référence pour la description de situations types d'usages variés des EIAH. Ces scénarios vont servir à la validation de nouveaux formalismes et modèles pour les EIAH. Dans ce contexte, notre travail de thèse participe en partie à l'étude faite sur le potentiel et les limites de langages de description de scénarios pédagogiques tels que la spécification IMS-LD. Dans cette thèse, nous proposons une comparaison de cette spécification standard avec le langage CPM, basé sur une approche issue du génie logiciel : la méta-modélisation par les profils UML. Notre approche a permis de mettre en valeur l'exploitation et la spécification de la notation UML pour décrire et spécifier des modèles de situations d'apprentissage plus riches. Une autre approche de méta-modélisation directement basée sur le MOF a été également proposée (travaux sur l'environnement RAM3 [LeP03, Car03]). L'AS a mis en valeur la nécessité de comparer ces différentes approches et les standards actuels en développant des scénarios pédagogiques de référence et en les modélisant selon les diverses approches. L'objectif à

[136] http ://www-rtp39.imag.fr/

moyen terme est de créer une banque de scénarios pédagogiques de référence illustrant une grande variété de types et d'usages d'EIAH, pouvant conduire à une meilleure comparaison des futures propositions relatives à la modélisation en EIAH.

Le profil CPM a également fait l'objet d'une présentation à l'atelier « plate-forme pour la recherche en EIAH » dans le cadre du groupe de travail EIAH du GDR I3 (SA 4.1)[137].

11.3 Perspectives

Les différents travaux de cette thèse participent à un processus itératif de conception ouvrant de nombreuses perspectives à moyen et long termes. Toutefois, notre contribution doit être suivie dans l'immédiat de deux travaux à courts termes : la validation du langage CPM sur un second plan, la validation des modèles et de la proposition d'une méthode d'ingénierie pédagogique adaptée à CPM. Des pistes de travail sont d'ores et déjà envisagées ; elles sont présentées ci-après.

11.3.1 Vers une validation expérimentale des modèles produits avec le langage CPM

Le travail d'expérimentation que nous avons mené suite à la proposition du langage CPM concerne la vérification et la validation du langage. L'outillage et la mise à l'essai que nous avons réalisés participent à ce travail de vérification/validation.

Il s'agit maintenant de réaliser cette vérification/validation au niveau des modèles construits avec le langage CPM. En effet, la vérification s'intéresse alors dans ce cas à répondre à la question : les modèles produits avec le langage CPM sont ils corrects et cohérents vis-à-vis d'UML et de CPM ? En réponse, nous avons outillé notre proposition dans un AGL UML existant en implémentant le profil CPM sous la forme d'un module indépendant, utilisable par tous. Dans un second temps, nous avons personnalisé l'AGL en l'adaptant à notre langage et au public visé : l'ingénieur pédagogique de l'équipe pluridisciplinaire de conception des PBL. Nous avons ainsi proposé un prototype d'environnement-auteur facilitant l'élaboration des modèles et garantissant la cohérence de la sémantique de CPM. Nous avons également produit des modèles pour le cas d'étude SMASH (mise à l'essai). Nous pouvons considérer que SMASH a été validé *a priori*, c'est-à-dire en amont de nos travaux (il a été élaboré en collaboration avec des professeurs des écoles puis il a été expérimenté en présentiel). La validation de SMASH *a posteriori* de nos travaux pose la question : les modèles CPM produits pour SMASH répondent-ils aux besoins de l'équipe de conception d'unités pédagogiques de type PBL ? Pour répondre à cette question le langage CPM doit être utilisé et expérimenté sur d'autres cas d'utilisation par des équipes pluridisciplinaires, c'est-à-dire ceux à qui s'adressent le langage CPM.

Pour cela, nous devons mettre à disposition notre contribution. Le partage et la publication de notre langage CPM peut se réaliser sur deux plans :

– Au niveau de la communauté UML Objecteering[138] : ceci permettra de rassembler des retours d'expérience correspondant davantage à un public du génie logiciel mais garantissant la validation

[137] http ://sis.univ-tln.fr/gdri3/

[138] La communauté est accessible *via* http ://www.umlopenedition.com/index.htm

de CPM du point de vue du langage UML dont il est dérivé. Dans ce cas, le langage CPM sera mis à disposition sous la forme du module **CPM**, utilisable avec *Objecteering Modeler*, accompagné d'une documentation d'utilisation.

– Au niveau de la communauté des EIAH et de l'ingénierie des EIAH en proposant le langage CPM sous sa forme de proposition théorique (le méta-modèle et le profil CPM) ainsi que sous sa forme outillée (le module **CPM**). Les modèles CPM pour SMASH ainsi que les documents narratifs ayant servi de base de construction doivent également être partagés. Nous espérons ainsi obtenir des retours riches portant à la fois sur l'utilisation outillée du langage mais également sur le modèle théorique sous-jacent au langage. Ceci nous amènera donc à adapter, modifier et mettre à jour le langage CPM aux niveaux de la syntaxe (concepts, relations et notation), de la sémantique et de l'outillage.

  Le contexte de l'AS-Plate-forme semble un contexte propice à la mise à disposition de nos travaux pour la communauté. De même, le travail sur la transformation de modèles, que nous avons initié en opérant une projection de nos modèles CPM vers la spécification IS-LD, est un élément stratégique important qui permettra de mieux positionner notre contribution vis-à-vis des autres langages connus dans la communauté EIAH s'intéressant au design pédagogique. Un positionnement de CPM vis-à-vis du langage MOT, utilisé dans le cadre de la méthode d'ingénierie pédagogique MISA, constitue un autre élément pour la validation.

Ces différents retours d'expérimentations permettront alors de :

– cadrer le langage CPM en élargissant ou en limitant ses usages ;
– évaluer ses capacités à couvrir la spécification pédagogique ;
– examiner sa puissance d'expression ;
– déterminer la plus value de disposer de modèles CPM par rapport à la difficulté de modélisation ;
– comparer les avantages et inconvénients de ce langage par rapport à d'autres.

Ce travail de validation permettra également d'alimenter la mise en œuvre d'une méthode d'ingénierie pédagogique adaptée à notre langage.

11.3.2 Vers une méthode adaptée au langage CPM

Comme nous l'avions indiqué dès la présentation du cadre de recherche de nos travaux, la proposition d'une méthode de conception pédagogique (on parle également de *design* pédagogique) est considérée comme *postérieure* à l'élaboration du langage CPM. Le contenu des recherches présenté dans ce mémoire est donc indépendant de toute réflexion méthodologique. Pourtant, de nombreux éléments méthodologiques ont été abordés indirectement :

– la méthode d'ingénierie pédagogique MISA pour laquelle nous nous sommes intéressés à la notation MOT et au méta-modèle sous-jacent proche de celui de la spécification IMS-LD ;
– la phase de conception d'unités d'apprentissage aboutissant à la modélisation d'un scénario pédagogique et décomposée en différentes étapes identifiées ;
– ou encore le choix de diagrammes UML privilégiés pour la notation du langage CPM.

Concrètement, la mise au point d'un langage est implicitement liée à la prise en compte de certains éléments et à la mise à l'écart d'autres ; ces choix sont la base de la méthode qui accompagnera le langage.

Nous donnons ci-après quelques pistes que nous considérons comme fondamentales pour la mise au point d'une méthode adaptée au langage CPM. De par la nature intrinsèque de CPM, qui est une spé-

cialisation d'UML au domaine de la conception de PBL, **deux pistes doivent être étudiées conjointement** : les techniques et les outils autour d'UML et ceux du *design* pédagogique.

11.3.2.1 Une méthode inspirée des travaux de méthodologie UML

Comme UML sur lequel il est basé, le langage CPM est un langage de modélisation orienté objet mais indépendant d'une quelconque méthode objet. Toutefois, comme pour UML, il convient, pour tirer le meilleur avantage de CPM, de suivre un processus/méthode (section 5.1.4) :

- dirigé par les cas d'utilisations : nous proposons en effet les diagrammes de cas d'utilisation pour capturer, dès l'expression initiale des besoins, les activités des apprenants et tuteurs ainsi que leurs relations sociales dans la réalisation des activités ;
- centré architecture : il convient alors de définir un modèle d'architecture pour les situations d'apprentissage que nous modélisons afin de faciliter la construction, la gestion et l'évolution de la PBL en conception ; ce modèle peut être défini sur la base des vues complémentaires (vue sociale, vue pédagogique et vue structurelle) que nous proposons dans le méta-modèle CPM ;
- itératif et incrémental : la conception d'unité pédagogique est déjà initialement un processus itératif (voir section 1.2.2.2).

Nous proposons alors comme première piste d'étudier les méthodes actuelles pour UML (RUP [Kru01], 2TUP [Roq00], etc.) comme base initiale à l'élaboration d'une méthode pour CPM.

Le langage CPM est également une spécialisation d'UML sous la forme d'un profil UML. De par la démarche de construction du profil CPM, nous avons privilégié l'utilisation de certains diagrammes et notations au détriment d'autres (chapitre 8) : ces *choix* réalisés identifient implicitement des bases pour la méthode. Toutefois, ces choix peuvent encore être remis en cause par le travail de validation que nous proposons de mener dans un premier temps.

Dans ce travail de recherche, nous avons mis en évidence l'analogie entre les processus logiciels en général et le processus de conception de situations d'apprentissage, qui implique de nombreux acteurs (enseignant, ingénieur pédagogique, etc.) et de nombreuses activités (expression initiale des besoins, analyse et conception), et dont l'*extrant* consiste en une unité pédagogique décrivant un scénario d'apprentissage formalisé. Nous avons également identifié une seconde analogie entre les processus logiciels et ces scénarios pédagogiques : un scénario pédagogique est la description d'un processus impliquant des acteurs (apprenants et tuteurs) qui réalisent individuellement ou collectivement des activités dans un même but commun (*explicitement* les objectifs de la PBL et *implicitement* les objectifs d'apprentissage fixés par l'enseignant) ; les différentes activités utilisent, modifient et produisent des ressources intermédiaires (une ressource correspondant autant à des objets d'apprentissage concrets, qu'à des représentations mentales liées aux connaissances des apprenants, ou qu'à des événements qui seront générés par le système d'exécution). Nous avons d'ailleurs bâti le langage CPM sur la base des travaux, entre autres, du SPEM (*Software Process Engineering Metamodel*) [OMG01b]. Une piste supplémentaire consiste à étudier les travaux du génie logiciel sur la description et la modélisation UML de ces processus logiciels.

11.3.2.2 Une méthode inspirée des travaux du *design* pédagogique

Une méthode adaptée au langage CPM est une méthode d'ingénierie pédagogique au sens large. Dans ce sens, une première piste consiste à étudier, discuter et synthétiser les similitudes de ces différentes

méthodes afin d'extraire les éléments nécessaires à l'élaboration de notre méthode : les acteurs concernés, les étapes dans le processus de conception guidé par la méthode, ou encore les usages des modèles.

La méthode d'ingénierie des systèmes d'apprentissage (MISA) [Paq97] a déjà été introduite dans ce mémoire pour sa particularité qui consiste à appliquer l'usage de techniques de modélisation cognitive à la fois pour la représentation des connaissances, des traitements pédagogiques et des traitements médiatiques (section 4.3.4). Cette méthode permet ainsi « *[...] de produire le devis d'un système d'apprentissage, de guider la réalisation des matériels pédagogiques et de planifier la mise en place de l'infrastructure de support technologique et organisationnel du système d'apprentissage* ». Le *devis* du système d'apprentissage est composé d'un modèle des connaissances, d'un modèle pédagogique et d'un modèle médiatique ; ces modèles servent alors de base pour la construction des matériels pédagogiques et la mise en œuvre des infrastructures.

Ainsi, la mise en place d'une méthode adaptée au langage CPM est une démarche d'ingénierie portant, en particulier, sur la planification de séquences d'apprentissages (on parle plus précisément de séquences *prescriptives*) pour laquelle de nombreuses études et propositions théoriques existent [Cre97, Des00, Bra03].

L'ingénierie des connaissances[139] (IC) est également un domaine représentant une piste pour aider à cadrer les usages de la méthode. En effet, le point de convergence entre nos travaux et l'IC se situe au niveau de la modélisation des connaissances du domaine d'apprentissage ou encore des connaissances des apprenants. Ceci nous amène à nous poser la question : comment les connaissances à acquérir sont-elles associées aux activités ? Il existe plusieurs pistes possibles : simples descripteurs de pré-requis et d'objectifs (exemple : langage CPM, IMS-LD), représentation explicite des connaissances (MISA), modélisation didactique [Per03]. Dans le chapitre 9 de mise à l'essai de notre langage, nous avons proposé des extraits de modèles pour SMASH dont certains étaient très proches de la représentation de connaissances (voir figures 9.18, 9.20). Cette étude de l'IC permettrait soit d'étendre les usages du langage CPM de manière à recouvrir une partie de la modélisation au sens représentation de connaissances, soit de cadrer l'usage du langage CPM en restreignant son usage afin de ne pas couvrir ces types de modèles. La première perspective de validation du langage permettra de répondre, en partie probablement, à cette nouvelle problématique.

11.3.2.3 Autres pistes

La mise au point d'une méthode permet également d'identifier et de proposer des techniques et outils qui aideront et guideront les concepteurs dans le suivi de la méthode. Le langage CPM et son outillage en font partie. Une autre technique consiste à aider les utilisateurs de la méthode à réutiliser des parties de modèles répondant à un problème générique dans la modélisation des situations d'apprentissage. Nous pensons alors que les patrons[140] sont une piste intéressante à ce sujet. La recherche sur les patrons doit

[139] Voici une définition citée dans [Tch02b] comme extraite de l'appel à communication de la conférence IC'2001 : « l'ingénierie des connaissances propose des concepts, méthodes et techniques permettant de modéliser, de formaliser, d'acquérir des connaissances dans les organisations dans un but d'opérationalisation, de structuration ou de gestion au sens large. Ces mêmes connaissances sont des informations destinées à être, *in fine*, interprétées par un humain, dans son interaction avec l'artefact, *i.e.* le système à base de connaissances (SBC) construit ».

[140] Un patron est une solution à un problème récurrent dans un contexte donné.

se réaliser sur plusieurs plans : au niveau de l'identification des problèmes récurrents dans le domaine de la modélisation de situations d'apprentissage, au niveau de la définition de patrons pour l'analyse [Fow96, Joh97] et la conception [Gam95] et au niveau du formalisme UML pour les décrire [Evi00, Lar01].

Une autre problématique concerne également le rôle donné aux modèles dans la méthode. Dans le cadre des travaux de cette thèse, les rôles des modèles bâtis avec le langage CPM concernent le support de la phase de conception. Il est possible dans une approche d'ingénierie des modèles de donner une plus grande importance aux modèles : les modèles supportent et **guident** la conception. Cette approche, inspirée du cadre théorique du MDA (*Model Driven Architecture*) [OMG03a], confère aux modèles UML une importance capitale dans la traçabilité des différents modèles produits. Les profils UML sont au cœur de l'approche : ils représentent la technique utilisée pour automatiser la transformation des modèles. Des travaux portant sur cette nouvelle problématique de l'ingénierie des EIAH ont débuté dans d'autres laboratoires [Car03, LeP03].

Les différents travaux de validation et de mise en œuvre d'une méthode adaptée permettront de répondre à la problématique de l'usage des modèles (critère d'exhaustivité défini par J.P. Pernin) : en effet, une situation d'apprentissage présente plusieurs facettes du point de vue du tuteur : prescription (ce que nous avons traité), observation, régulation (traité en partie), évaluation, capitalisation, personnalisation (traité) [Per03].

11.4 Conclusion

Cette thèse s'inscrit dans le cadre de recherche pluridisciplinaire des EIAH et plus particulièrement de l'ingénierie des EIAH ; dès lors, nos recherche nous ont permis de nous intéresser à plusieurs domaines allant des sciences humaines et sociales jusqu'au génie logiciel. Cette pluridisciplinarité explique en partie l'effort d'expérimentation que nous avons suivi pour appliquer et vérifier nos propositions sur le cas d'étude pédagogique SMASH. Cette thèse est une contribution très orientée génie logiciel pour les EIAH ; elle concerne directement cette communauté.

Cependant, ce travail de thèse concerne également la communauté du génie logiciel pour laquelle notre contribution représente une application de ses méthodes et de ses techniques, tout particulièrement celles sur la méta-modélisation UML *via* les profils.

Annexes

Annexe A

Méta-modèle CPM

Sommaire

Cette annexe présente l'ensemble des différents paquetages composant notre méta-modèle CPM[141].

[141] Tous les diagrammes de classes décrivant notre méta-modèle ont été réalisés avec l'outil *Objecteering Modeler*.

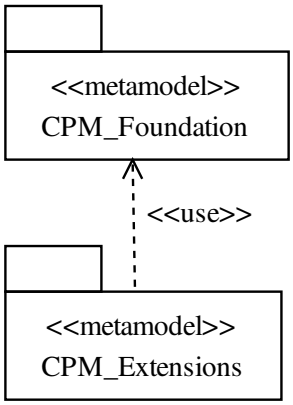

FIG. A.1 – Les deux paquetages formant le méta-modèle CPM : le paquetage `CPM_Foundation` et le paquetage `CPM_Extensions`.

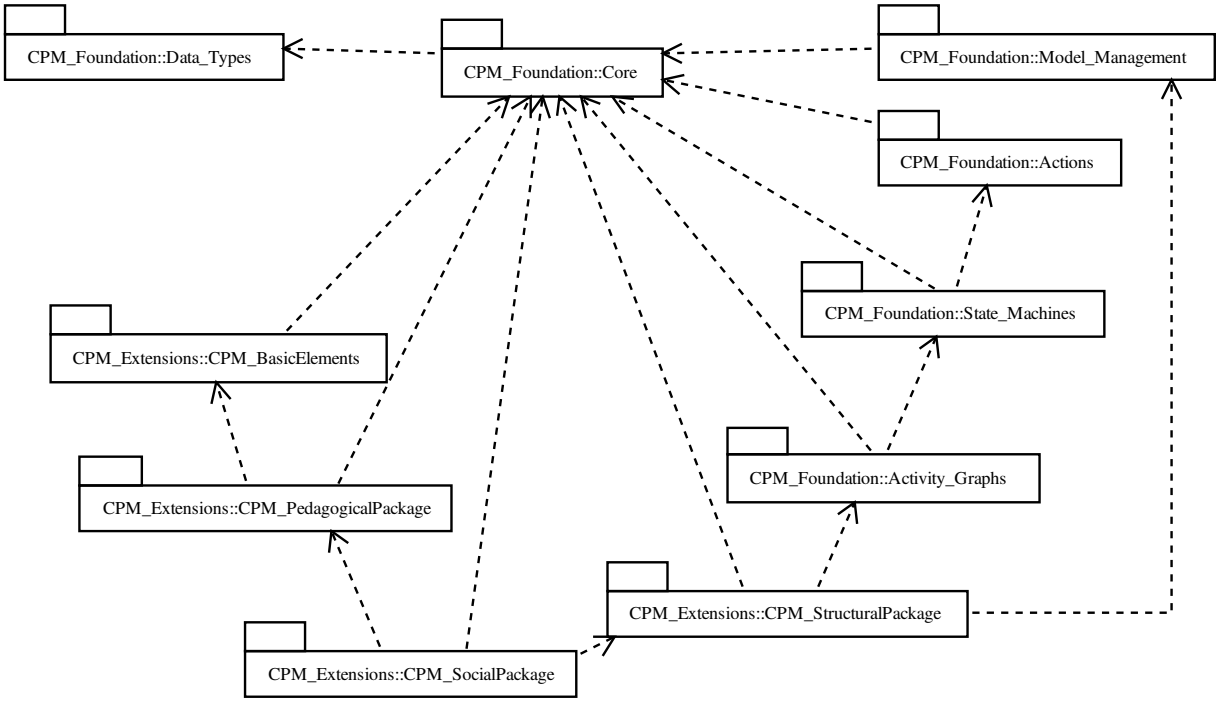

FIG. A.2 – Tous les paquetages et leurs dépendances.

A.1 Paquetage CPM_Foundation

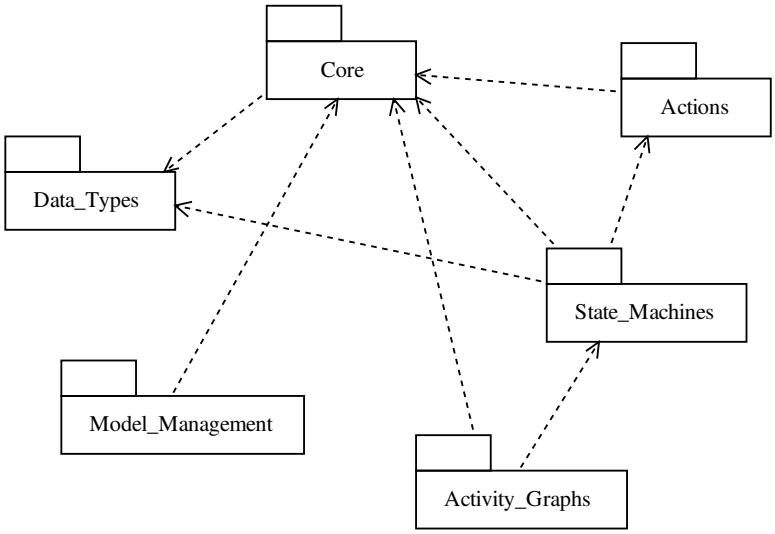

FIG. A.3 – Les dépendances entre sous-paquetages composant le paquetage `Foundation`.

A.1.1 Core

Le paquetage `CPM_Foundation` : :`Core` est structuré de la même manière que le paquetage `Core` d'UML 1.4.

Pour chacun des diagrammes présentés, des classes, attributs et associations peuvent avoir été omis du méta-modèle UML 1.4 afin de ne définir que les éléments nécessaires à la définition des modèles CPM.

Quelques variations ont été également apportées :

- Dans le diagramme *Relationships* la multiplicité de l'association entre `Association` et `AssociationEnd` vaut «2», au lieu de «2..*» tel que spécifié dans le méta-modèle d'UML 1.4. Ceci permet de limiter les associations aux seules associations binaires.
- de manière similaire, dans le diagramme `Dependencies` les associations *supplier* et *client* entre `Dependency` et `ModelElement` ont la multiplicité «1», au lieu de «1..*» tel que spécifié dans UML 1.4.

A.1.1.1 `Backbone`

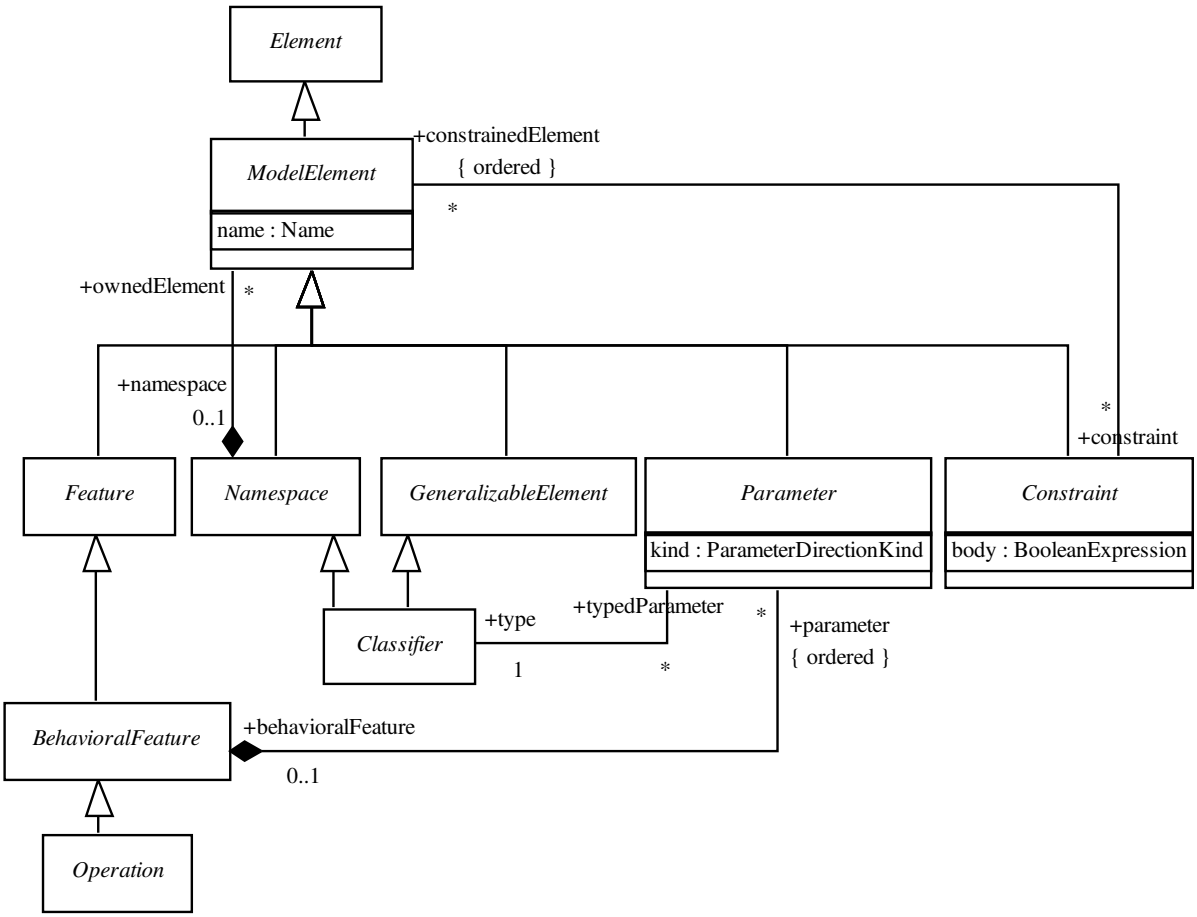

FIG. A.4 – Le sous-paquetage `Backbone`.

A.1.1.2 Relationships

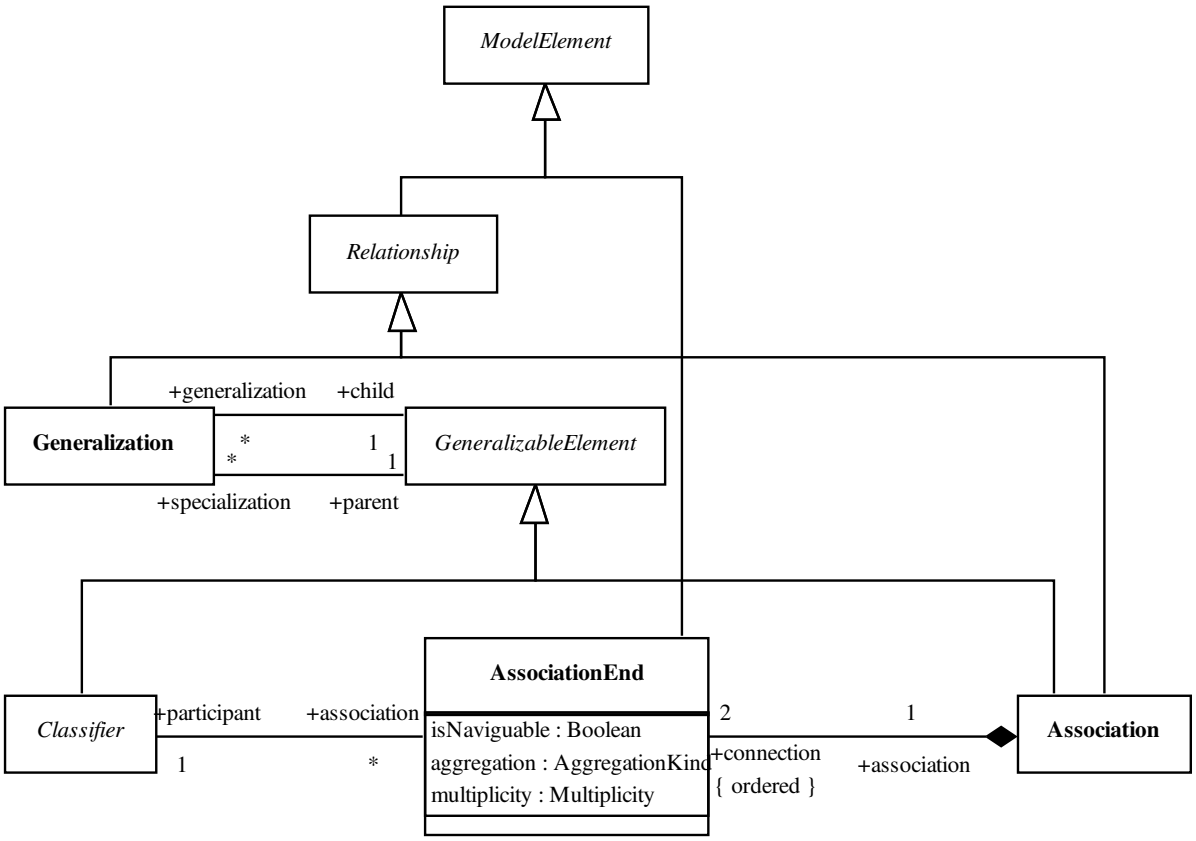

FIG. A.5 – Le sous-paquetage Relationships.

A.1.1.3 `Dependencies`

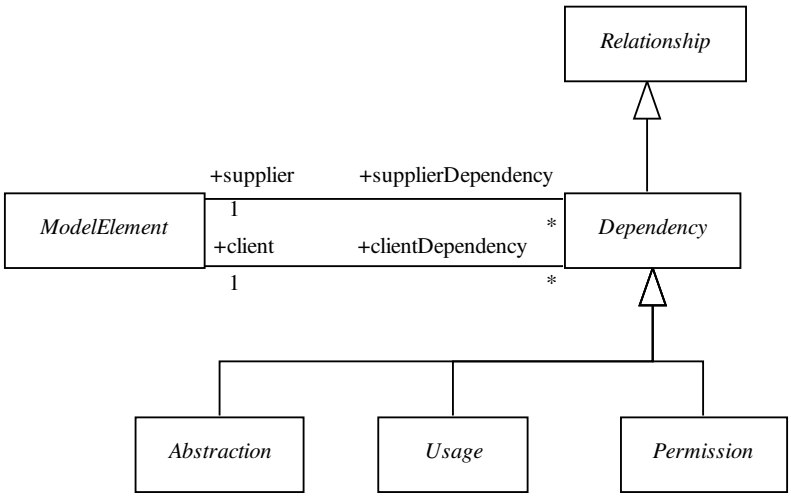

FIG. A.6 – Le sous-paquetage `Dependencies`.

A.1.1.4 Classifiers

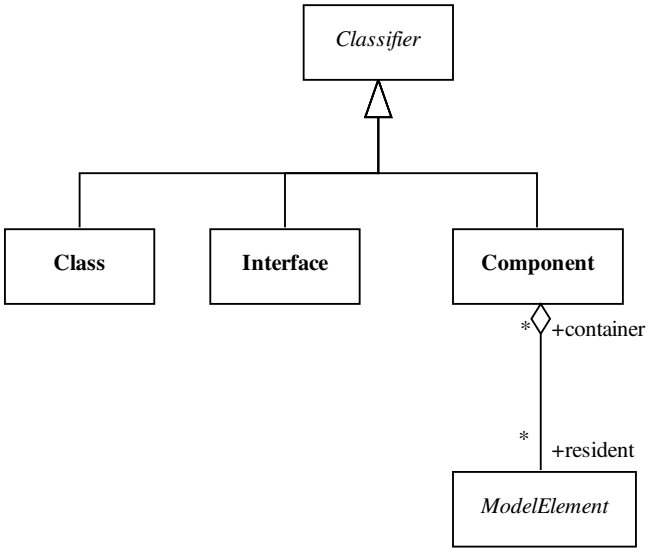

FIG. A.7 – Le sous-paquetage Classifiers.

A.1.1.5 `Auxiliary Elements`

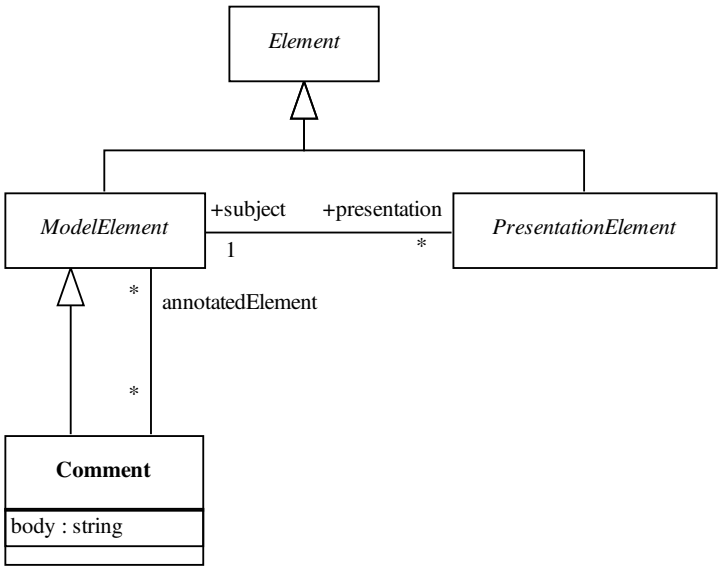

FIG. A.8 – Le sous-paquetage `Auxiliary Elements`.

A.1.2 `Data_types`

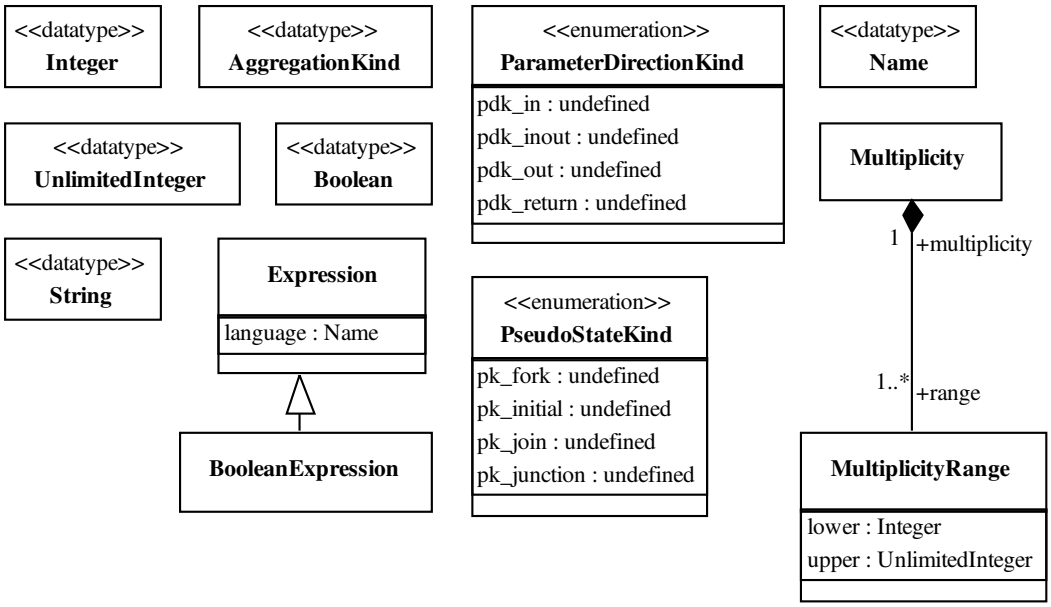

FIG. A.9 – Le sous-paquetage `Data_types`.

A.1.3 `State_Machines`

Le paquetage `CPM_Foundation : :State_Machines` est un sous-ensemble du paquetage `State_Machines` d'UML 1.4.

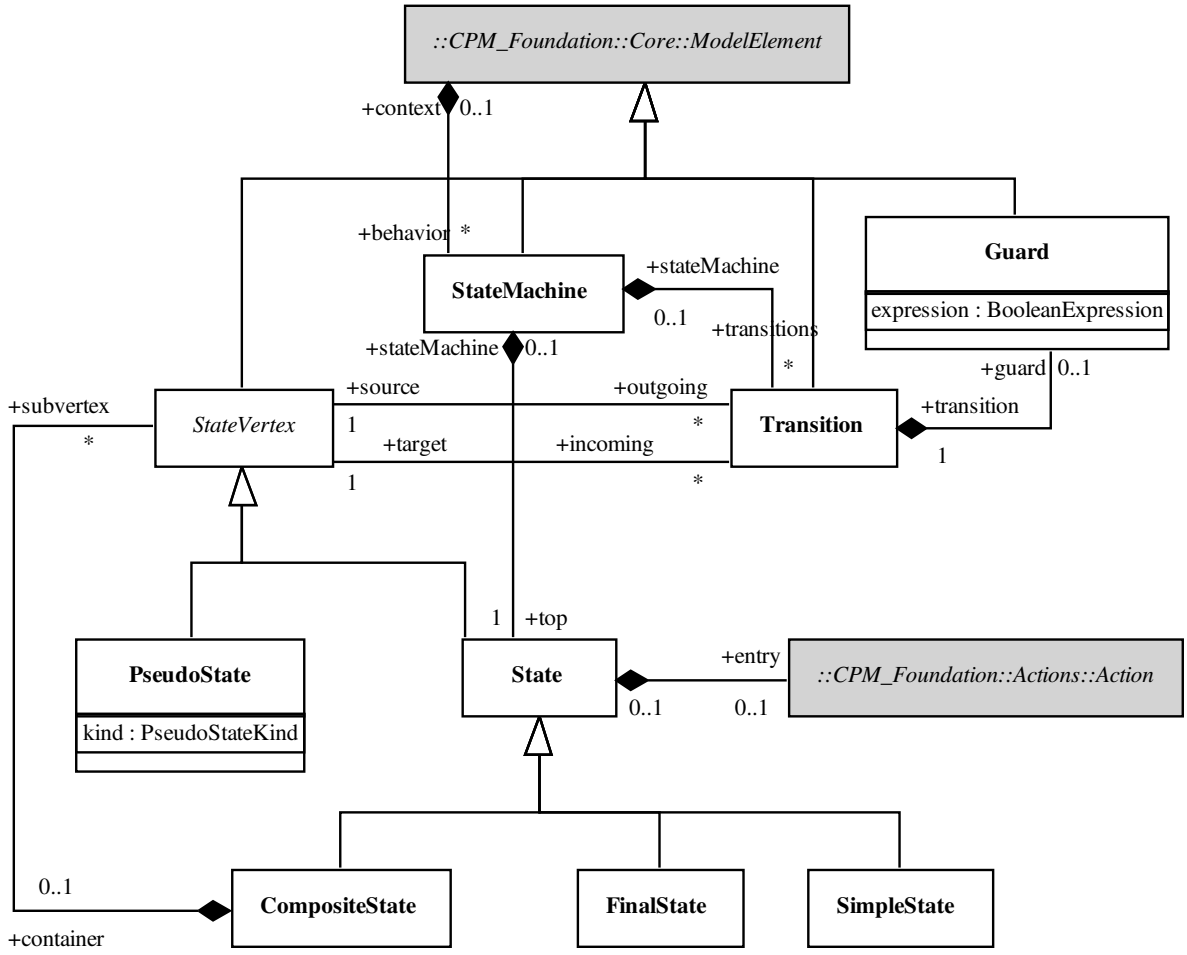

FIG. A.10 – Le sous-paquetage `State_Machines`.

A.1.4 `Actions`

Le paquetage `CPM_Foundation : :Actions` est un sous-ensemble du paquetage `Common_Behavior` d'UML 1.4.

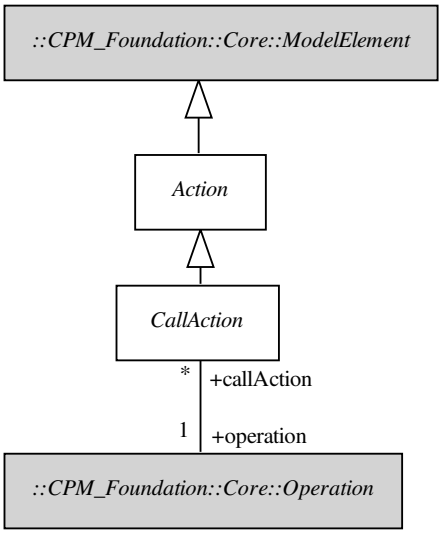

FIG. A.11 – Le sous-paquetage `Actions`.

A.1.5 `Activity_Graphs`

Le paquetage `CPM_Foundation : :Activity_Graphs` est un sous-ensemble du paquetage `Activity_Graphs` d'UML 1.4.

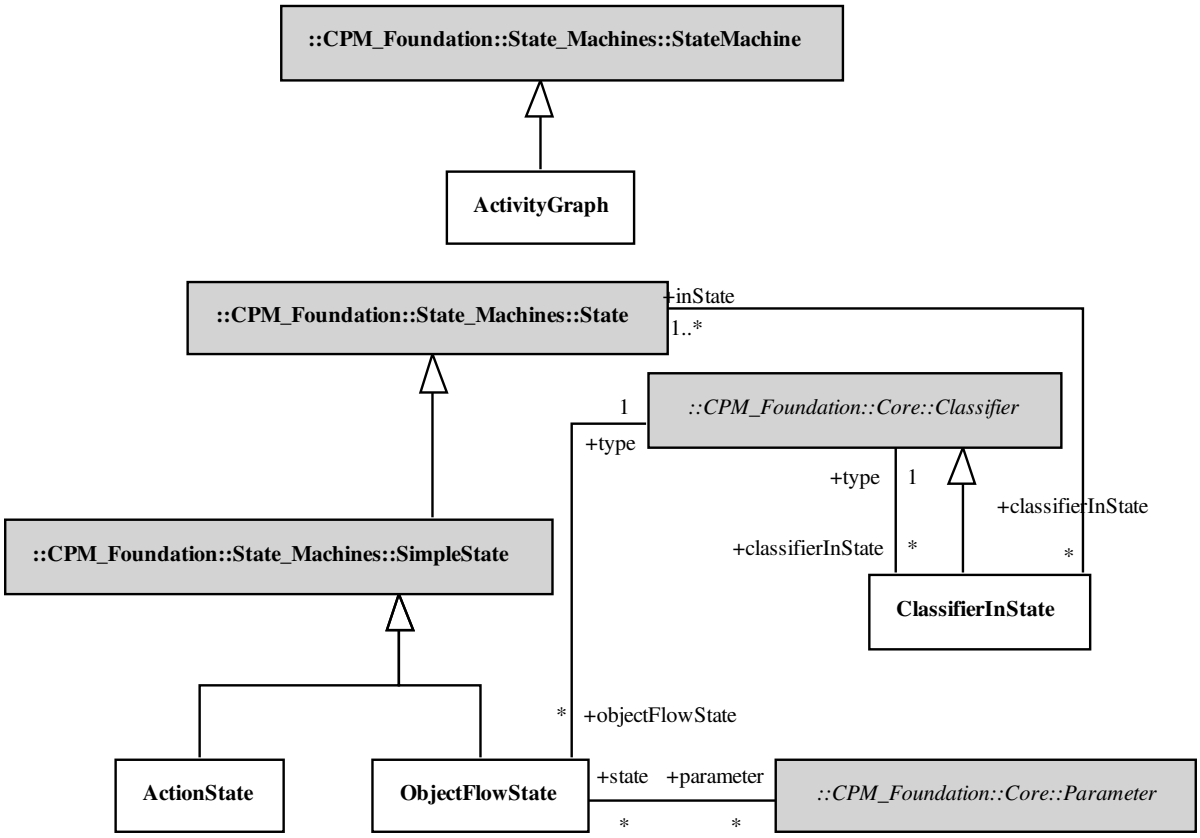

FIG. A.12 – Le sous-paquetage `Activity_Graphs`.

A.1.6 `Model_Management`

Le paquetage `CPM_Foundation : :Model_Management` est un sous-ensemble du paquetage `Model_Management` d'UML 1.4.

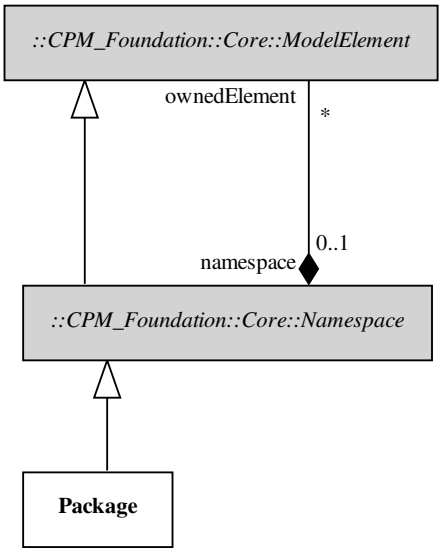

FIG. A.13 – Le sous-paquetage `Model_Management`.

A.2 Paquetage `CPM_Extensions`

Les différents paquetages composants les *extensions* au paquetage de fondations ont été décrits en détail dans le chapitre 7.

A.2.1 `CPM_BasicElements`

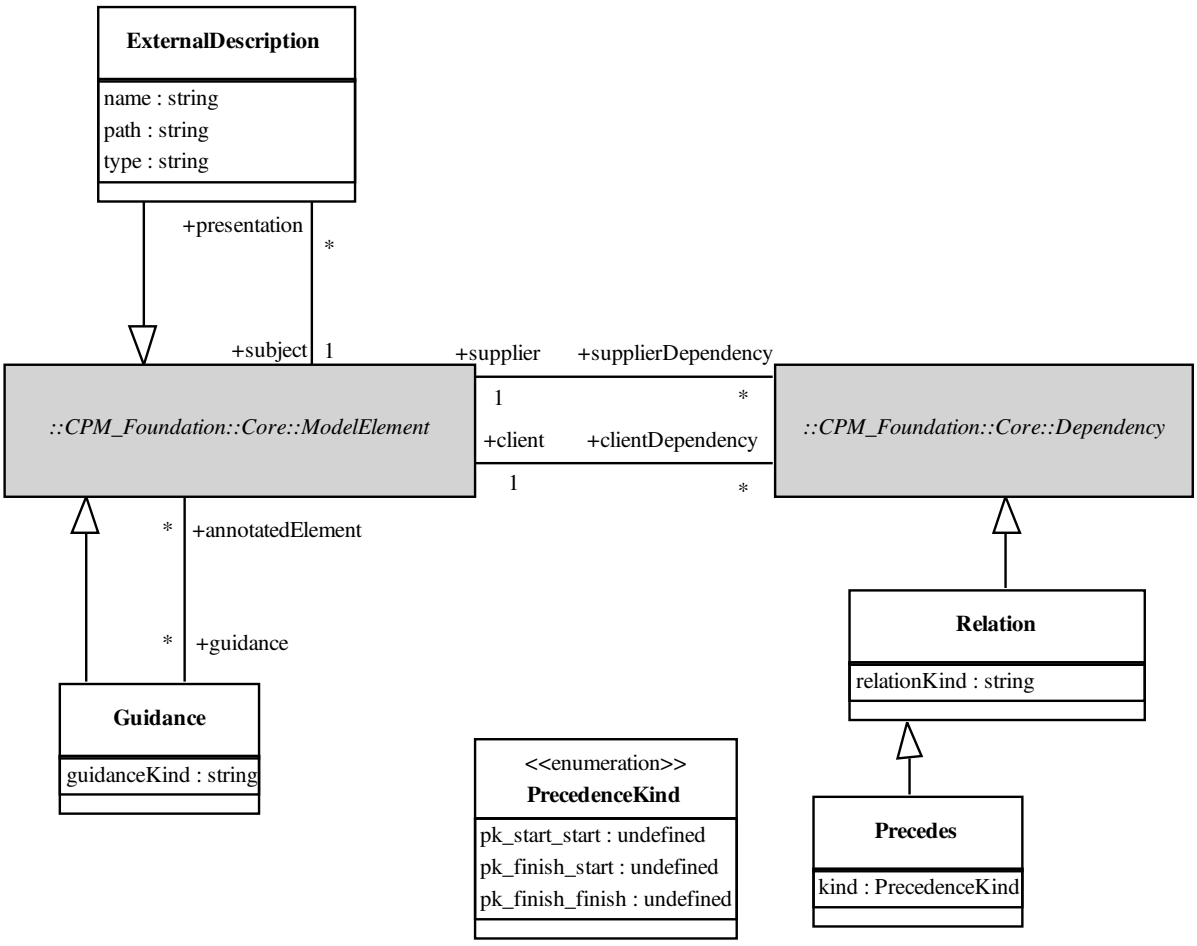

FIG. A.14 – Le paquetage des éléments de base : `CPM_BasicElements`.

A.2.2 CPM_PedagogicalPackage

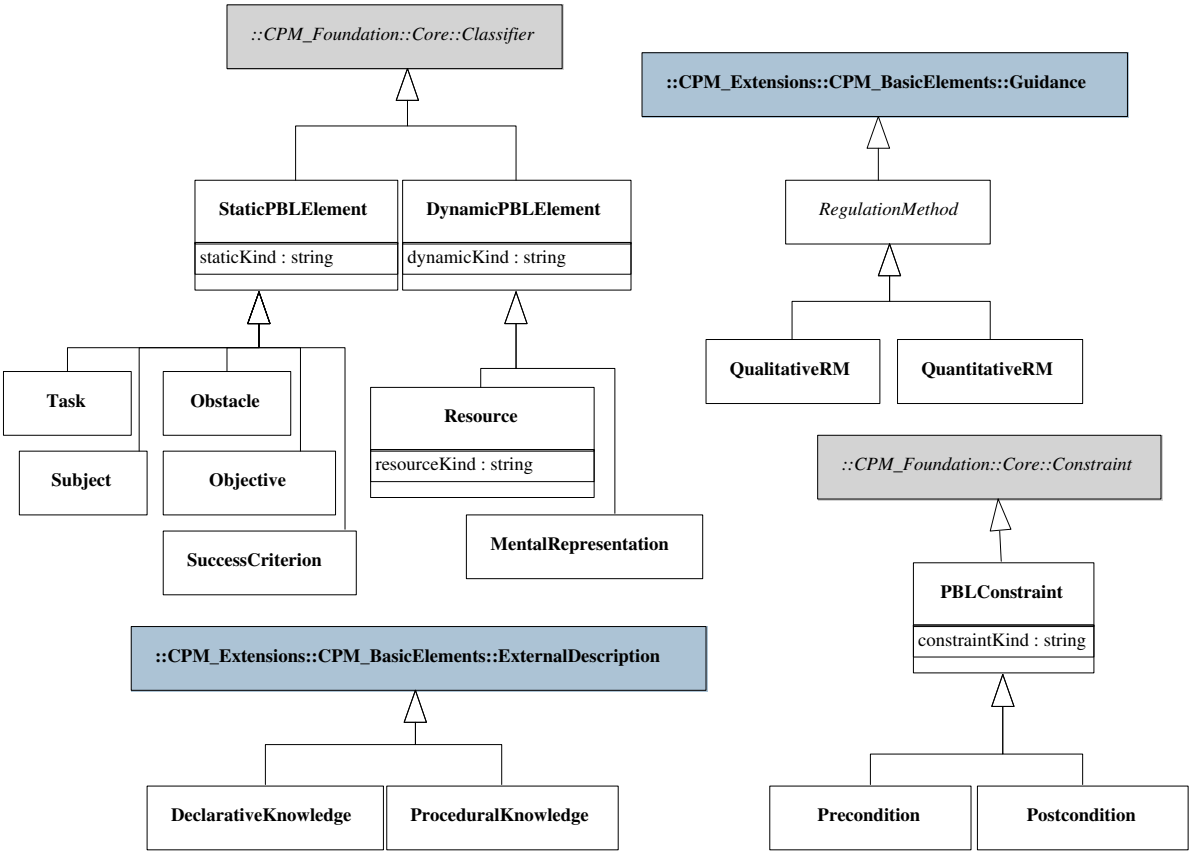

FIG. A.15 – Le paquetage pédagogique : CPM_PedagogicalPackage.

A.2.3 `CPM_StructuralPackage`

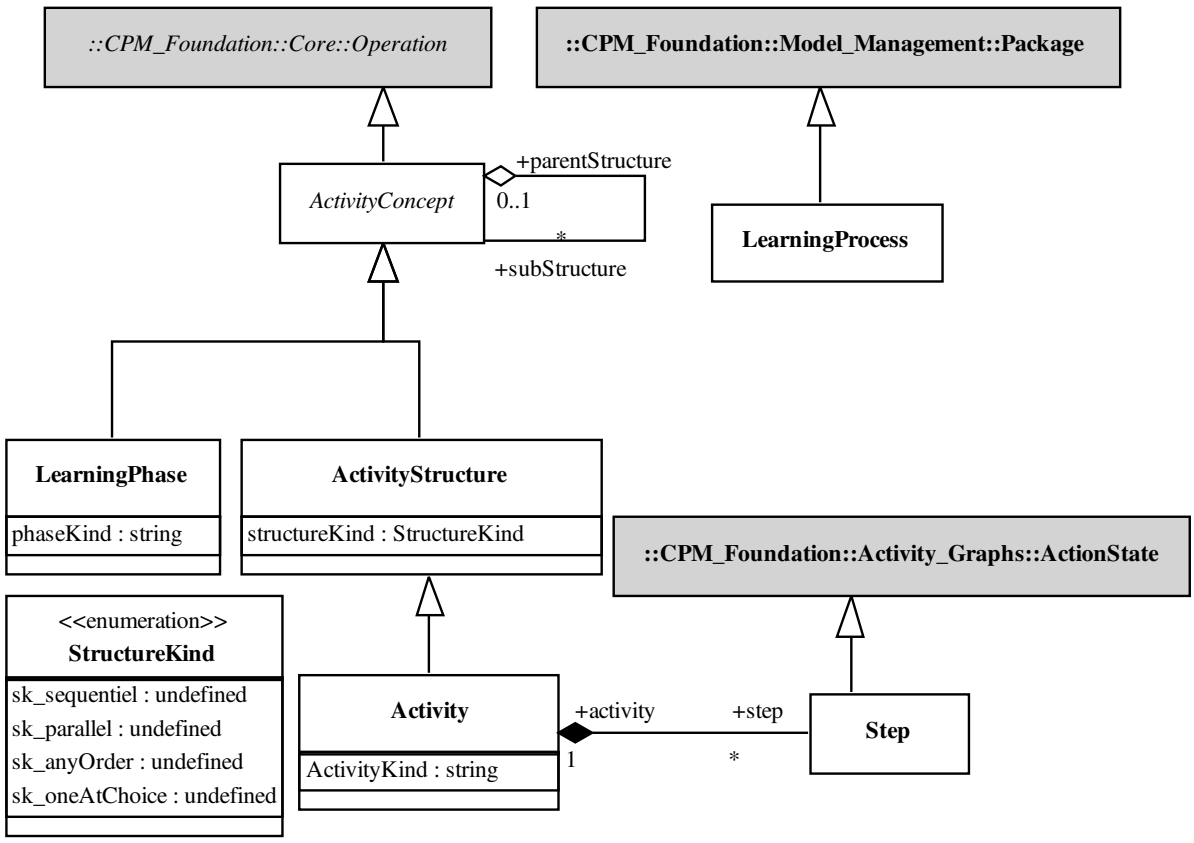

FIG. A.16 – Le paquetage structurel : `CPM_StructuralPackage`.

A.2.4 CPM_SocialPackage

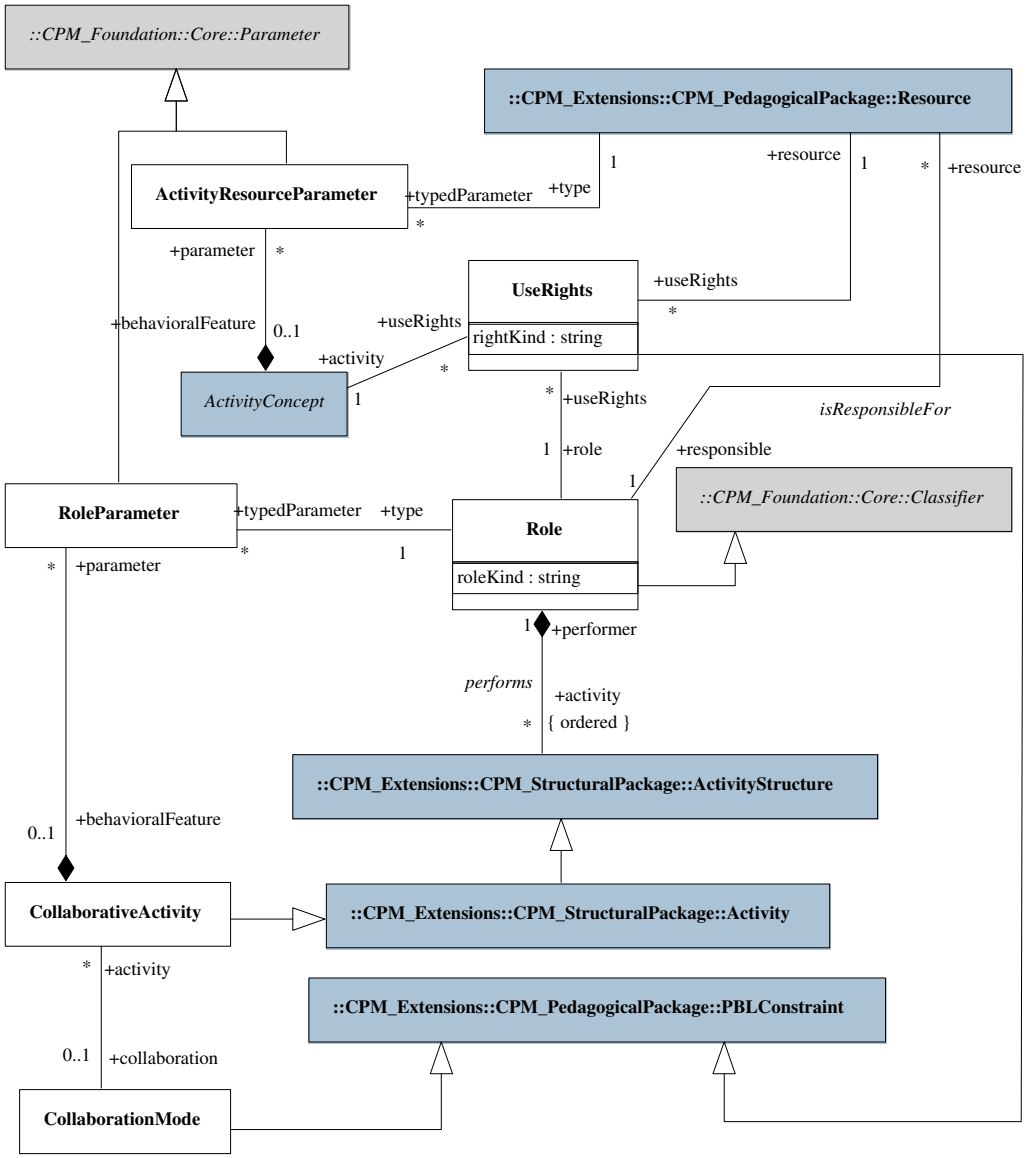

FIG. A.17 – Le paquetage social : CPM_SocialPackage.

A.2.5 CPM_CPL

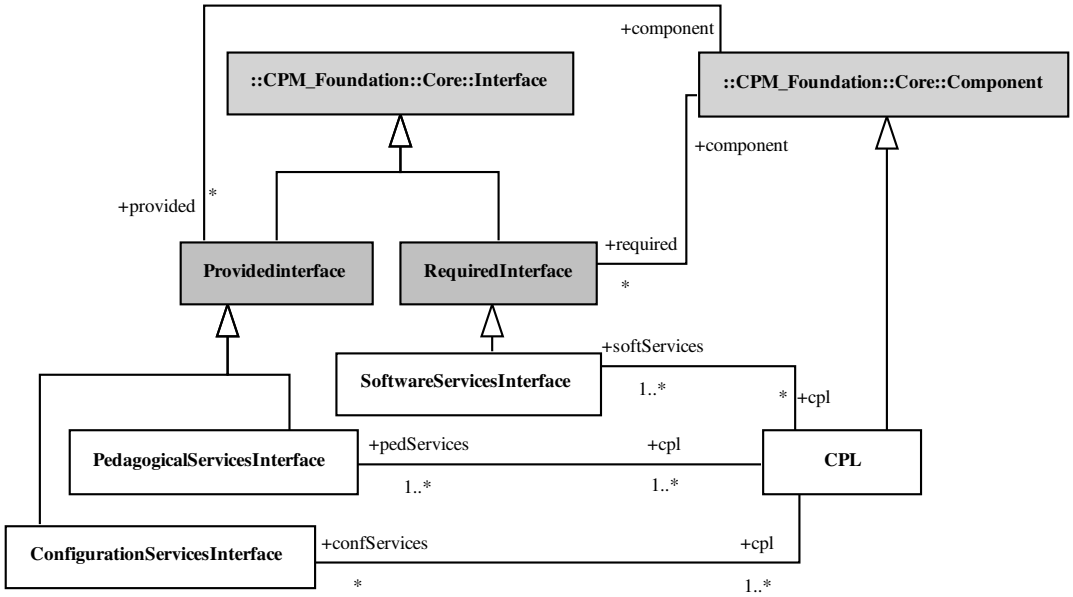

FIG. A.18 – Le paquetage des composants CPL.

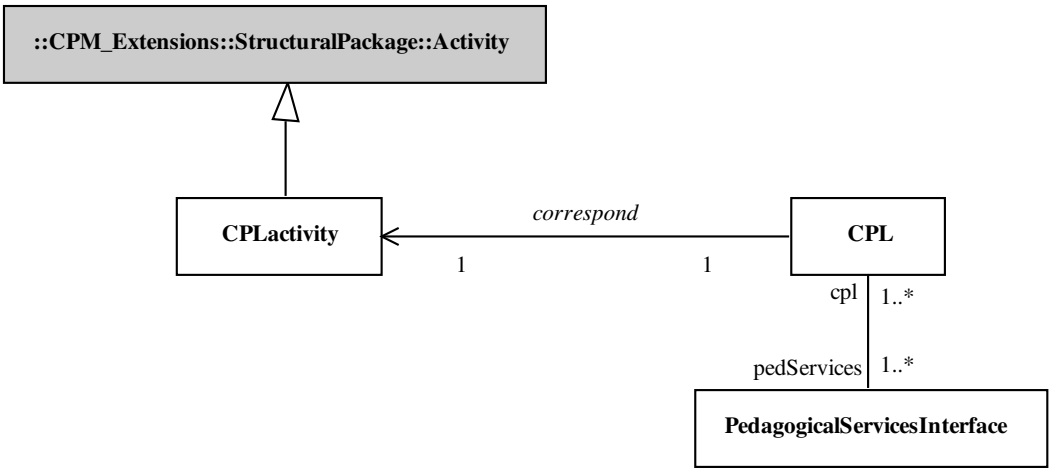

FIG. A.19 – Le paquetage des composants CPL : liaison avec le concept d'activité.

Annexe B

Programmation des commandes du module CPM_code

Sommaire

Cette annexe présente des extraits de code écrits dans le langage J.

B.1 Vérification d'une contrainte CPM

Le code suivant correspond à la procédure chargée de vérifier la contrainte : la relation stéréotypée <<**Precedes**>> ne peut relier que deux éléments de modélisation héritant de **ActivityConcept**.

```
void Object::verifRule1 (in JNoModalBox box) {

Stereotype s;
boolean trouve;
JTreeItem item=box.#Report#addReport("info","1",this);

box.#Report#linkToReportUnder("1","info", "Verification de la regle 1 :
    <<Precedes>> ne peut relier que deux <<activity>>");

trouve=false;
s=findStereotypeInProject ("Precedes","Use");

s.ExtendedElementModelElement {
    UsedNameSpace
    {
```

```
        if (isStereotyped("Activity")==false and isStereotyped("CPL")==false
        and isStereotyped("LearningPhase")==false and
        isStereotyped("ActivityStructure")==false
        and isStereotyped("CollaborativeActivity")==false)
        {
            box.#Report#linkToReportUnder("1","delete","La "+ ClassOf.Name +
                " <<"+Name+">> est l'origine d'une édpendance <<Precedes>>
                sans etre éstreotypee par un (sous-)concept d'<<Activity>>");
            trouve=true;
        }
    }
    UserNameSpace
        {
            if (isStereotyped("Activity")==false and
                isStereotyped("CPL")==false and
                isStereotyped("LearningPhase")==false and
                isStereotyped("ActivityStructure")==false and
                isStereotyped("CollaborativeActivity")==false)
            {
                box.#Report#linkToReportUnder("1","delete","La "+ ClassOf.Name +
                    " <<"+Name+">> est la destination d'une dependance <<Precedes>>
                    sans etre stereotypee par un (sous-)concept d'<<Activity>>");
                trouve=true;
            }
    }
}
if (trouve)
    return;

box.#Report#linkToReportUnder("1","add", "...OK");
}
```

B.2 Recherche d'éléments CPM dans l'ensemble du modèle

Cette procédure est rattachée à tout **ModelElement**. Elle permet ainsi d'effectuer une recherche sur la globalité du modèle à partir de n'importe quel élément de modélisation sélectionné. Cette procédure affiche une première boite de dialogue invitant l'utilisateur a sélectionné le stéréotype CPM pour lequel il recherche des instances. Ensuite, une deuxième boite de dialogue lui permet d'affiner les critères de recherche en sélectionnant une définition de valeur marquée pour le stéréotype précédemment choisi (ceci indique que la recherche devra retrouver tous les éléments de modélisation ayant le stéréotype et la valeur marquée choisis). Puis, une dernière boite de dialogue permet à l'utilisateur de donner une valeur à cette définition de valeur marquée. Les éléments correspondants du modèle sont alors rassemblés dans une fenêtre de résultat. Lorsque l'un des éléments résultat est sélectionné dans la fenêtre, l'élément de modélisation concerné dans le modèle est automatiquement sélectionné. Ceci permet de retrouver facilement des informations dans des modèles complexes.

```
void ModelElement::SeekStereotypes () {

String stereo; int width=260;
int height=300;

ModelElement[] res;
String type;
TagType[] tt;
JBox box;

int nbParam;
String param;

JBox box2;
JBox box3;

JNoModalBox resultsBox=#Report#createReportBox("Resultats");
resultsBox.addBitmap("c:\img\CPM.bmp");

box=createJBox("simpleBoxIdent","Recherche d'elements de modelisation",
    JLayoutVertical, false);

box.addBitmap("c:\img\CPM.bmp");

box.addList("list","Choisissez un stereotype CPM:",false,width,height);

box.addListItem("list","ExternalDescription","c:\img\ExternalDescription.bmp",
     true); // on selectionne le premier element de la liste par defaut
box.addListItem("list","Guidance","c:\img\smallGuidance.bmp",false);
box.addListItem("list","Relation","",false);
box.addListItem("list","CollaborationMode","",false);
box.addListItem("list","ActivityParameter","c:\img\smallActivityParameter.bmp",
    false);
box.addListItem("list","RoleParameter","c:\img\smallRoleParameter.bmp",false);
box.addListItem("list","LearningPackage","c:\img\smallLearningPackage.bmp",false);
box.addListItem("list","LearningProcess","c:\img\smallLearningProcess.bmp",false);
box.addListItem("list","LearningPhase","c:\img\smallLearningPhase.bmp",false);
box.addListItem("list","Activity","c:\img\smallActivity.bmp",false);
box.addListItem("list","CollaborativeActivity","c:\img\smallCollaborativeActivity
    .bmp",false);
box.addListItem("list","Role","c:\img\smallRole.bmp",false);
box.addListItem("list","ActivityStructure","c:\img\smallActivityStructure.bmp",
    false);
box.addListItem("list","StaticPBLElement","c:\img\smallPBLElement.bmp",false);
box.addListItem("list","DynamicPBLElement","c:\img\smallPBLElement.bmp",false);
box.addListItem("list","Resource","c:\img\smallResource.bmp",false);
```

```
box.addListItem("list","SuccessCriterion","c:\img\smallSuccessCriterion.bmp",
    false);
box.addListItem("list","MentalRepresentation","c:\img\smallMentalRepresentation
    .bmp", false);
box.addListItem("list","Step","c:\img\smallStep.bmp",false);
box.addListItem("list","DeclarativeKnowledge","c:\img\smallKnowledge.bmp",false);
box.addListItem("list","ProceduralKnowledge","c:\img\smallKnowledge.bmp",false);
box.addListItem("list","Constraint","",false);
box.addListItem("list","Objective","c:\img\smallObjective.bmp",false);
box.addListItem("list","Subject","c:\img\smallSubject.bmp",false);
box.addListItem("list","Task","c:\img\smallTask.bmp",false);
box.addListItem("list","Obstacle","c:\img\smallObstacle.bmp",false);
box.addListItem("list","RegulationMethod","c:\img\smallRegM.bmp",false);
box.addListItem("list","Precondition","",false);
box.addListItem("list","Postcondition","",false);

if (box.show()==true)

{

   // recuperer l'element selectionne dans la liste
   box.getValue("list",stereo);

   //faire les recherches du stereotype pour toutes les metaclasses auxquelles
   //il se refere
   if (stereo=="LearningPackage")
   {
     res.add(seekME(stereo,"Package",tt));
   }
   else if (stereo=="LearningProcess")
   {
     res.add(seekME(stereo,"Package",tt));
   }
   else if (stereo=="ExternalDescription")
   {
     res.add(seekME(stereo,"ModelElement",tt));
   }
   else if (stereo=="Resource")
   {
     res.add(seekME(stereo,"Class",tt));
   }
   else if (stereo=="Tool")
   {
     res.add(seekME(stereo,"Class",tt));
   }
   else if (stereo=="Role")
   {
```

```
  res.add(seekME(stereo,"Actor",tt));
}
else if ((stereo=="LearningPhase")or(stereo=="PedagogicalActivity")or
        (stereo=="PedagogicalStructureActivity"))
{
  res.add(seekME(stereo,"UseCase",tt));
  res.add(seekME(stereo,"Operation",tt));
  res.add(seekME(stereo,"ActionState",tt));
}
//on a recupere dans "res" tous les ModelElement stereotypes par le stereotype
recherche
//affiner la recherche en prososant un type de tag
if (tt.size()!=0)
{
   box2 = createJBox ("simpleBoxIdent2", "Affiner la recherche",
                      JLayoutVertical, false);
   box2.addBitmap("c:\img\CPM.bmp");
   box2.addCombo ("List", "Liste des types de tag compatibles avec le
                  stereotype choisi", 300);
   tt
   {
      box2.addListItem ("List", Name, "", false);
   }
   if (box2.show()==true)
   {
      box2.getValue("List",type);
      if (type!="")
         res=res.<select(selectIfTaggType(type)==true);
      tt.<select(Name==type)
      {
         if(ParamNumber.toInt() > 0)
         {
             nbParam=ParamNumber.toInt();
             while (nbParam>0)
             {
                box3 = createJBox ("simpleBoxIdent3", "Affiner la recherche",
                       JLayoutVertical, false);
                box3.addBitmap("c:\img\CPM.bmp");
                box3.addText("param","Tapez un parametre :","",width,25);
                if (box3.show()==true)
                {
                   box3.getValue("param",param);
                   if (param!="")
                      res=res.<select(selectIfTaggParam(param)==true);
                }
                nbParam=nbParam-1;
             }
```

```
            }
         }
      }
   }
   add2Report(resultsBox,res);

   if (res.size==0)
   {
      resultsBox.#Report#addReport("info","Aucune element de votre modele
      correspond a votre recherche...",this);
   }
   resultsBox.show(true);
}
else
{
   StdOut.write ("Annuler", NL);
}

}   // method SeekStereotypes
```

La méthode **seekME()** permet de renvoyer une liste d'élément de modélisation utilisant le stéréotype «*stereo*» sur la méta-classe UML «*baseclass*» donnés en paramètres. De plus, cette méthode renvoie dans «*listTagType*» la liste des définitions de valeur marquée associées au stéréotype.

```
ModelElement[] Object::seekME (
    in String stereo,
    in String baseclass,
    inout TagType[] listTagType)
{
   String nom;

   findStereotypeInProject(stereo,baseclass)
   {
       RequiredTagTagType
       {
           nom=Name;
           if (!notVoid(listTagType.<select(Name==nom)))
               listTagType.addElement(this);
       }
       return ExtendedElementModelElement;
   }
}
```

B.3 Génération d'un modèle XML conforme à la spécification IMS-LD

La méthode *generate()* suivante permet de générer un fichier externe à Objecteering dont le contenu est conforme à la spécification d'IMS-LD. L'appel de cette méthode est réalisée par une commande accessible *via* le *workproduct* «IMS-LD».

Seul le niveau A est pour le moment généré. Les informations sources sont déduites d'un diagramme d'activités (précisé par un paramètre du *workproduct*). Le diagramme d'activités est parcouru au niveau M2 afin d'en retirer l'ensemble des informations permettant une projection vers IMS-LD. Ainsi, les stéréotypes <<**Role**>>, <<**Activity**>>, <<**CollaborativeActivity**>> et <<**CollaborativeMode**>> sont utilisés, ainsi que leurs définitions de valeurs marquées associées.

```
//-------------------------------------------------------------------
// Operation IMS-LD spec::default#external#Code#CPM_code#generate()
//-------------------------------------------------------------------

void IMS-LD spec::generate () {

String content;
String fileName;
String AGname;
String tab;
String typeActivity;
boolean activityAG;
State[] listeState;
String[] lStates;
State[] liste2;
boolean fin;
StateVertex[] listeDejaTraites;

tab="    ";
activityAG=true;
fin=true;

//en-tete de la spec
content="<learning-design identifier=~"LD-FromCPM~" uri=~"URI~" level=~"A~">";
content.strcat(NL); content.strcat(tab + "<title>CPM2IMSLD_example</title>" + NL);
content.strcat(tab + "<learning-objectives></learning-objectives>" + NL);
content.strcat(tab + "<prerequisites></prerequisites>" + NL+NL);

content.strcat(tab + "<components>" + NL+NL);

//on se place sur le graphe d'activite concerne
getActivityGraph() {

    //*************************************************
```

```
//**************  gestion des roles ******************
//****************************************************

// une partition --> un role
content.strcat(tab + tab +"<roles>" + NL+NL);

SwimlanePartition
{
    if (isStereotyped("Role"))
    {
        if (getTagValue("roleKind")=="learner")
            content.strcat(tab+tab+tab+"<learner identifier=~""
                           +RepresentedNameSpace.Name+"~"/>"+NL+NL);
        else
            content.strcat(tab+tab+tab+"<staff identifier=~""
                           +RepresentedNameSpace.Name+"~"/>"+NL+NL);
    }
    // au moins une partition => pas un AG pour specifier la methode
    activityAG=false;
}
content.strcat(tab + tab +"</roles>" + NL+NL);

//****************************************************
//**************  gestion des activites **************
//****************************************************

// un etat --> une activite
content.strcat(tab + tab +"<activities>" + NL+NL);

// ne pas prendre l'etat "Top"
listeState=getStates().<select(Name!="ActivityGraph");

// on enleve les doublons dans les activites
listeState
{
    if (!lStates.contains(Name))
    {
        liste2.addElement(this);
        lStates.addElement(Name);
    }
}

liste2
{
    //StdOut.write ("Etats = ", Name, NL);
    if (isStereotyped("Activity") or isStereotyped("CollaborativeActivity")
       or isStereotyped("CPLActivity"))
```

```
        {
            // selon le type du role realisant l'activite on peut savoir
            // s'il s'agit d'une activite de support ou d'apprentissage
            if (getTypeRole()=="learner")
                typeActivity="learning-activity";
            else
                typeActivity="support-activity";

            content.strcat(tab+tab+tab+"<"+typeActivity+" identifier=~""
                           +Name+"~">"+NL);
            content.strcat(tab+tab+tab+tab+"<activity-description>"+NL);
            content.strcat(tab+tab+tab+tab+tab+"<item identifierref=~"~"
                           identifier=~"Id-"+Name+"~"/>"+NL);
            content.strcat(tab+tab+tab+tab+"</activity-description>"+NL);
            content.strcat(tab+tab+tab+"</"+typeActivity+">"+NL+NL);
        }
        else if (isStereotyped("ActivityStructure"))
        {
            content.strcat(tab+tab+tab+"<activity-structure identifier=~""
                           +Name+"~" ");
            content.strcat("number-to-select=~""+ SubState.size+ "~" ");
            content.strcat("structure-type=~""+ getTagValue("structureKind")
                           +"~">"+NL);
            content.strcat(tab+tab+tab+tab+"<title/>"+NL);
            // pour toutes les activites de la structure
            SubState
            {
                content.strcat(tab+tab+tab+tab+"<learning-activity-ref
                               ref=~"Id-"+Name+"~"/>"+NL);
            }
            content.strcat(tab+tab+tab+"</activity-structure>"+NL+NL);
        }
    }
    content.strcat(tab + tab +"</activities>" + NL+NL);

    // pas de gestion des environnements car bases sur la narration
    // et non deductibles du diagramme UML
    content.strcat(tab + tab +"<environments>" + NL+tab + tab +"</environments>"
                   +NL+NL);
    content.strcat(tab + "</components>" + NL+NL);
    content.strcat(tab + "<method>" + NL+NL);

    //*************************************************
    //**************  gestion de la methode ***********
    //*************************************************

    // si le AG ne concernent que des activites (et dc des roles)
```

```
    if (!activityAG)
    {
        content.strcat(tab +tab + "<play identifier=~"PLAY-"
                        +Name+"~" isvisible=~"true~">" + NL+NL);

        // retrouver l'etat initial (pseudostate)
        getInitialState()
        {
            rec_traite(content, listeDejaTraites, 0);
        }
        // fin du modele
        content.strcat(tab +tab +tab + "<complete-play>" + NL);
        content.strcat(tab +tab +tab +tab + "<when-last-act-completed />" + NL);
        content.strcat(tab +tab +tab + "</complete-play>" + NL+NL);
        content.strcat(tab +tab + "</play>" + NL+NL);
    }
    content.strcat(tab + "</method>" + NL+NL);
    content.strcat(tab + "</learning-design>" + NL+NL+NL);
}
// creation du fichier ou maj
fileName.strcat (getAttributeVal("Chemin"),"/",Name,".xml");
mngFile(fileName, content);

updateAllEditors();

}   // method generate
```

La méthode *generate()* précédente fait appel à de nombreuses autres méthodes. Nous donnons ci-après le détail de la méthode la plus importante : *rec_traite()*. Cette dernière permet de traiter un état du diagramme d'activité et de générer les différents actes. En réalité, cette méthode est récursive : elle s'appelle sur les états suivants selon le flot d'activité déduit du parcours du graphe d'activités.

```
// Operation StateVertex:default#external#Code#CPM_code#rec_traite()
//-------------------------------------------------------------------

void StateVertex::rec_traite (
    inout String text,
    inout StateVertex[] dejaTraites,
    in int nbfork)
{
String tab;
int nb;
tab="    ";

getEtatsSuivants()
{
    StdOut.write ("Etat : ",Name, NL);
```

```
if (!dejaTraites.contains(this))
{
    StdOut.write ("On traite", NL);
    dejaTraites.add(this);
    StdOut.write ("ajout a la liste des traites", NL);
    if (ClassOf.Name=="PseudoState")
    {
        StdOut.write ("pseudoState", NL);

        if (Name=="Fork")
        {
            StdOut.write ("----------------------------------> fork", NL);
            StdOut.write (getEtatsSuivants().size(), NL);
            rec_traite(text,dejaTraites,getEtatsSuivants().size());
        }
        else if (Name=="Join")
        {
            StdOut.write ("----------------------------------> join", NL);
            StdOut.write (nbfork, NL);
            //nbfork=nbfork-1;
            if (nbfork<=1)
            {
                StdOut.write ("==================================>>>> fork a 0", NL);
                rec_traite(text,dejaTraites,nbfork-1);
            }
        }
        else if (Name!="final")
            rec_traite(text,dejaTraites,nbfork);
    }
    else if (ClassOf.Name=="SubActivityState" or ClassOf.Name=="ActionState")
    {
        StdOut.write ("autre", NL);
        text.strcat(tab+tab+tab + "<act identifier=~"ACT-"+Name+"~">" + NL);
        traiteRP(text);

        //ajout d'autres role-part si contrainte de collaborationMode !!
        if (isStereotyped("CollaborativeActivity"))
        {
            StdOut.write ("Activite collaborative !", NL);
            if (contrainte("CollaborationMode"))
            {
                StdOut.write ("contrainte CollaborativeMode trouvee !!", NL);
                // on recherche les activites reliees
                nb=recupActivitesReliees(text,this,dejaTraites);
                if (nb!=0)
                {
                    StdOut.write ("----------->  modif de nbfork", NL);
```

```
                        nbfork=nbfork-nb;
                    }
                }
                else
                    StdOut.write ("WARNING : pas de contrainte CollaborativeMode
                                   trouvee...", NL);
            }

            text.strcat(tab +tab +tab + "</act>" + NL+NL);
            rec_traite(text, dejaTraites,nbfork);
        }
    }
    else
    {
        StdOut.write ("deja traite", NL);
    }
}
}   // method rec_traite
```

Annexe C

Documents sur la situation-problème SMASH

Sommaire

Cette annexe rassemble différents documents liés à la situation-problème SMASH.

C.1 Exemple de ressources

Nous donnons en exemple l'énoncé du témoignage 1 (Figure C.1) et du QCM correspondant (Figure C.2).

C.2 Fond de carte

Les figures suivantes illustrent les cartes du village qui sont distribuées aux apprenants :
- la carte globale (partie gauche) (Figure C.3),
- la carte globale (partie droite) (Figure C.4),
- le gros plan sur la zone de l'accident (Figure C.5).

C.3 Documents intermédiaires de travail

Nous présentons ci-après quelques exemples de documents de travail, élaborés en collaboration avec des professeurs des écoles, qui nous ont permis pour affiner l'élaboration du cas SMASH mais également comme base pour la spécification de nos modèles :

- un tableau récapitulant les différentes séances prévues pour SMASH (Figure C.6),
- un énoncé détaillant le contenu de la séance 3 (Figure C.7),
- un énoncé détaillant le contenu de la phase 1 de la séance 3 (Figure C.8),
- un énoncé détaillant le contenu de la phase 2 de la séance 3 (Figure C.9),
- un énoncé détaillant le contenu de la phase 3 de la séance 3 (Figure C.10).

Témoignage n°1, groupe 1

Transcription de l'interview

Témoin interrogé sur les lieux de l'accident par l'inspecteur Geoffrey Morante et son assistante Carine Gregor.

Nom: Simon Dupond
Adresse: 16 route du Brasier
Age: 22 ans

Simon Dupond : Je m'appelle Simon Dupond. J'habite au 16 route du Brasier. J'ai 22 ans.
Je venais juste de sortir du magasin et je me tenais prêt à traverser au passage piéton qui se situe sur mon chemin pour aller route du Brasier. Je me souviens qu'une femme attendait comme moi de l'autre côté du passage piéton.
Une voiture sortait du village des Mille-fleurs à toute vitesse en direction de la route des Marais. Je pense que le conducteur n'avait pas vu qu'un passage piéton était signalé. Il ne m'a vu qu'à la toute dernière minute car il s'est arrêté brusquement – j'ai d'ailleurs entendu les crissements de freins. Au même moment, une voiture blanche a traversé le passage piéton dans l'autre sens, sans ralentir !

Inspecteur Morante: Dans quelle direction allait la voiture blanche?

Simon Dupond : Vers le village des Mille-fleurs.

Inspecteur Morante: Que s'est-il passé ensuite?

Simon Dupond : Et bien, il y a eu un autre crissement de freins, je n'ai pas pu voir exactement, mais on aurait dit que la voiture blanche avait percuté un vélo. J'ai couru pour aller aider. Quelqu'un d'autre est parti téléphoner.
Pauvre enfant, il était inconscient. Il y avait du sang partout. Son vélo était une véritable épave.

Fig. C.1 – Exemple du témoignage 1 de SMASH.

QCM n°1, groupe 1

1) Où se situe Simon Dupond au moment de l'accident ?
   - o Sur la route, il traverse le passage piéton
   - o Il se prépare à traverser le passage piéton
   - o Il marche à côté du passage piéton

2) La voiture qui sort du village des Mille-fleurs :
   - o S'est arrêtée au passage piéton
   - o Ne s'est pas arrêtée au passage piéton
   - o On ne sait pas

3) La voiture blanche
   - o S'est arrêtée au passage piéton
   - o Ne s'est pas arrêtée au passage piéton
   - o On ne sait pas

4) Qui a percuté le vélo ?
   - o La voiture qui sort du village des Mille-fleurs
   - o La voiture blanche
   - o Il est tombé tout seul

FIG. C.2 – Exemple du QCM 1 associé au premier témoignage.

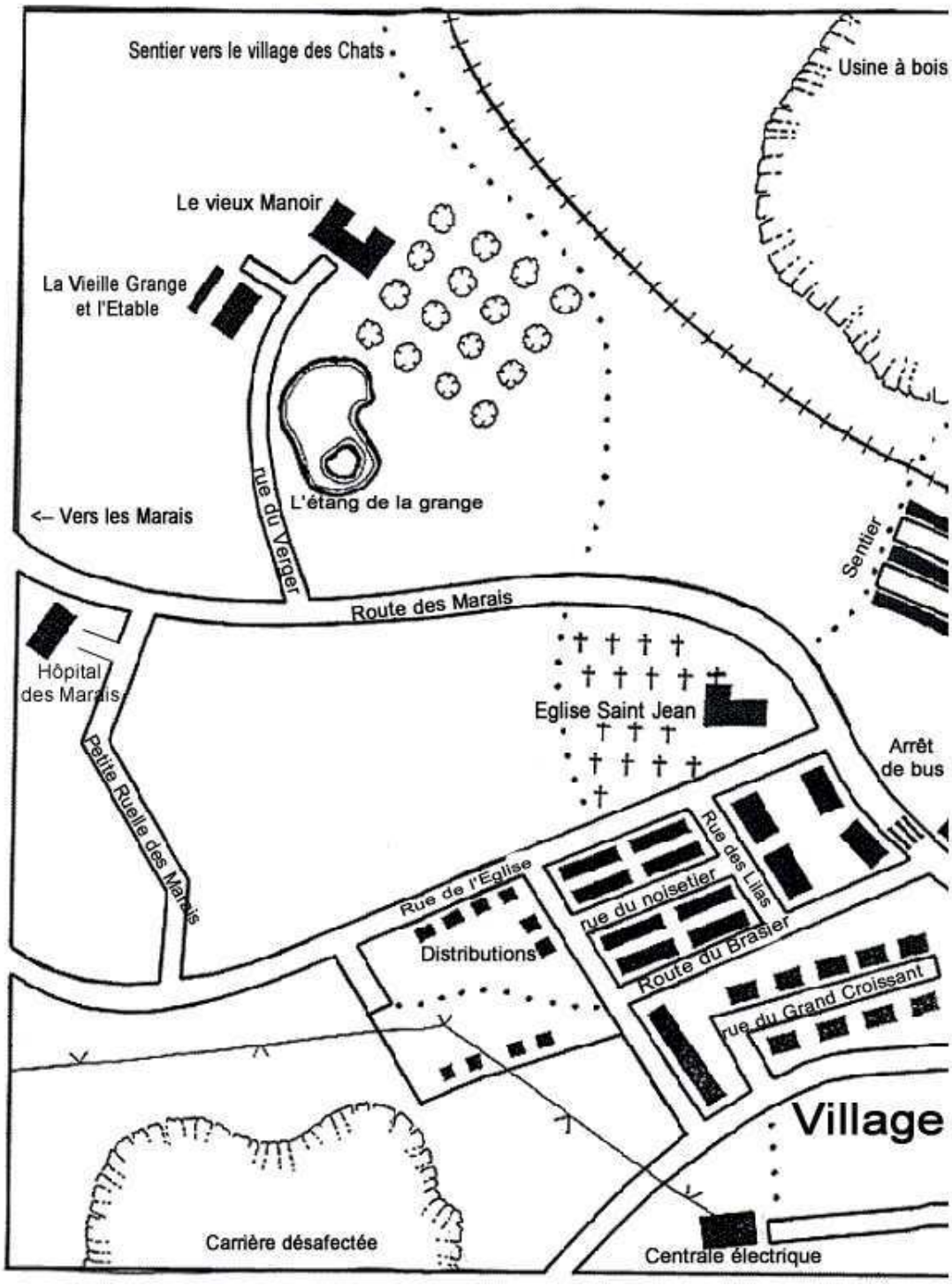

FIG. C.3 – Carte du village des MilleFleurs où s'est déroulé l'accident (partie gauche).

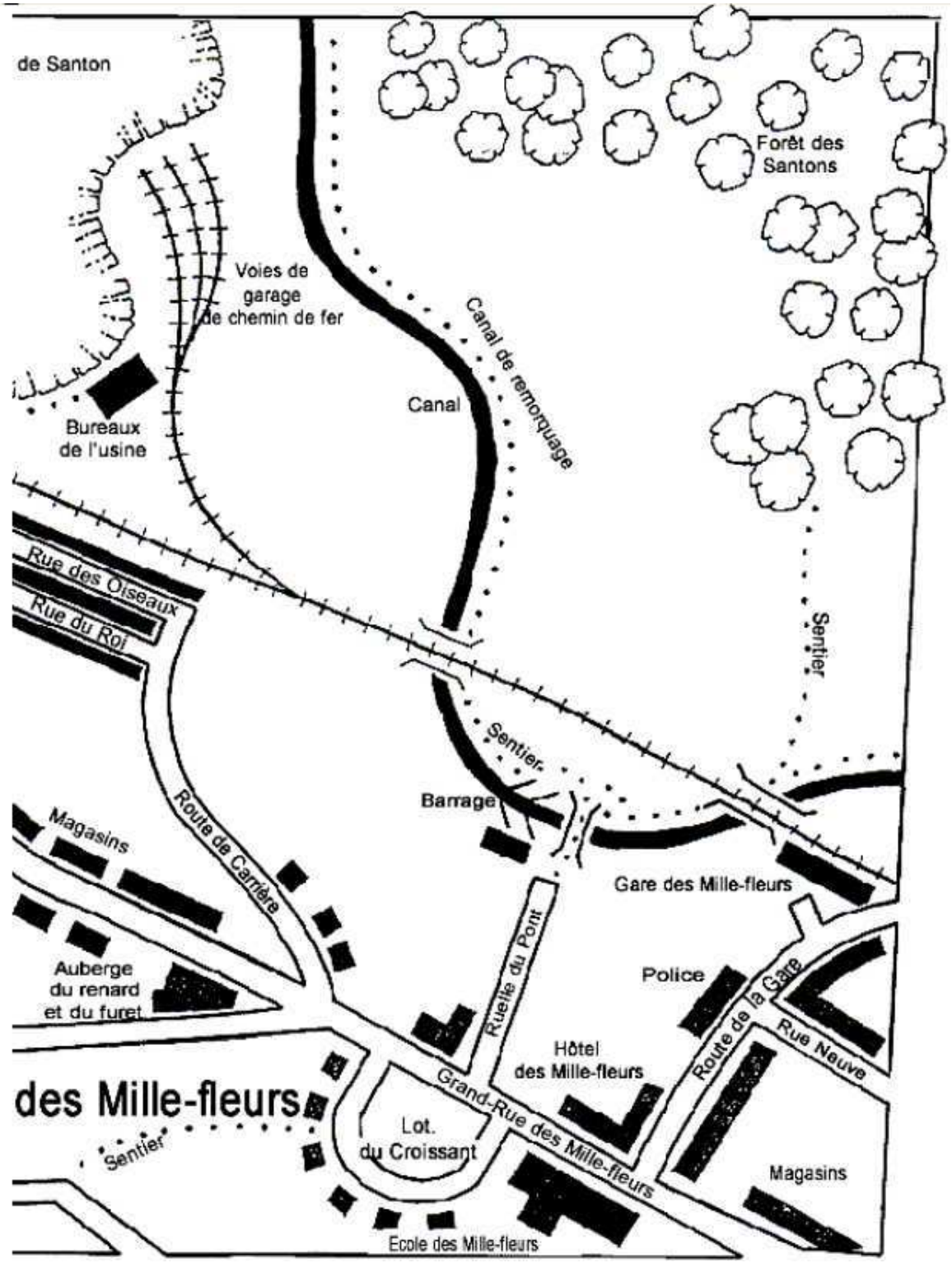

FIG. C.4 – Carte du village des MilleFleurs où s'est déroulé l'accident (partie droite).

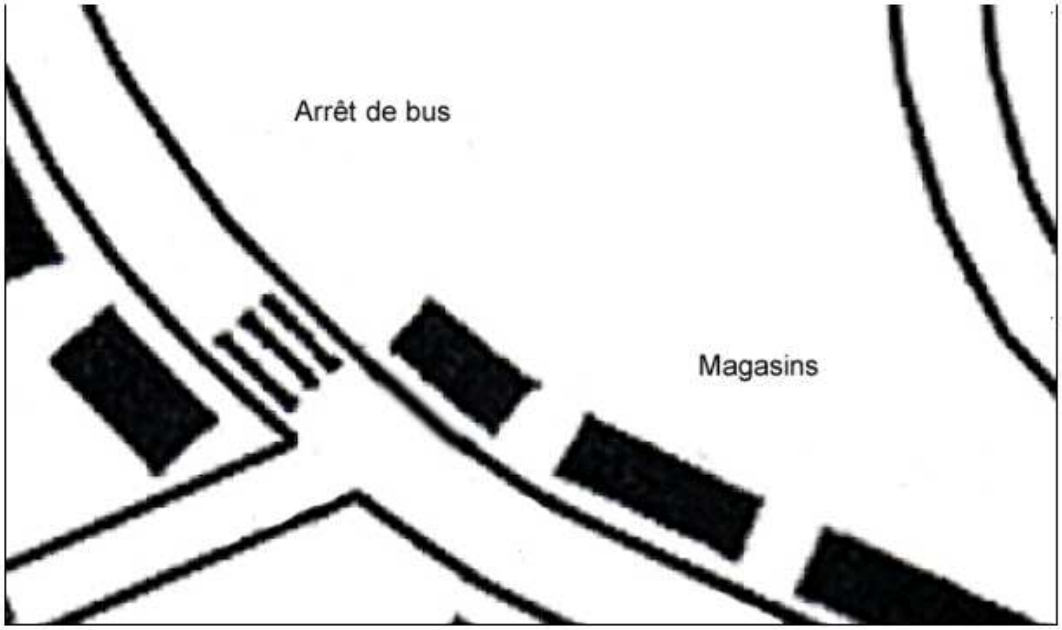

FIG. C.5 – Gros plan sur la zone de l'accident.

		Question de départ	Activités conduites par les élèves	Activités langagières	Organisation de la classe	Connaissances, savoirs et savoir-faire en jeu
Séance 1	Présentation de la situation-problème	Un accident de vélo a eu lieu. Comment trouver le coupable ?	Organisation en groupes de travail. Réflexion sur les tâches à accomplir pour réussir	Oral	classe entière	Communiquer, proposer des modes d'organisation
Séance 2	Phase de réalisation des enquêtes	Un accident de vélo a eu lieu. Qui est le coupable ?	Recherche d'information, positionnement d'objets sur un plan géographique, rédaction d'une synthèse	Oral et écrit à l'intérieur de chaque groupe	4 groupes	S'organiser à l'intérieur de chaque groupe. Communiquer par schémas et textes
Séance 3	Phase de mise en commun des résultats		Faire la synthèse du travail mené en séance 2, évaluer et corriger	Oral (compte-rendu de l'enquête) et écrit (modification de la carte)	4 groupes, classe entière	Argumenter, questionner
Séance 4	Phase de renforcement	Un accident de vélo a eu lieu. Quels sont les niveaux de responsabilité ?	positionnement d'objets sur un plan géographique, rédaction d'une synthèse argumentée	écrit	individuel	Réinvestir les connaissances acquises lors des séances précédentes

FIG. C.6 – Les différentes séances pour SMASH.

Séquence Smash
Séance 3 : mise en commun des résultats

Acteurs/Rôles :

- Enseignant-tuteur,
- Groupes G1 à G3 (qui sont successivement auditeurs et speakers),
- Groupe G4 (qui est successivement auditeur et speaker).

Contexte :

L'analyse des témoignages est terminée. Les différents groupes d'apprenants ont produit un schéma présentant leur vision de cet accident. Ils ont donc produit :

- Un schéma complété dans lequel ils ont positionné les témoins et acteurs,
- Un ensemble de QCM complétés (un QCM par témoignage),
- Leur synthèse : un ensemble de questions (zones d'ombre) + leur vision de l'accident (culpabilités, raisons, …).

Objectifs de la séance pour le prof :

- Vérifier que chaque groupe a bien prélevé les indices qui étaient dans leurs témoignages,
- Vérifier que, sur la carte, les éléments ont été placés comme il faut,
- Vérifier que chaque groupe a conscience de la fragilité de ses conclusions,
- Amener à une solution collective.

Motivation des groupes d'apprenants pour rentrer dans la séance :

- Exprimer sa vision, montrer qu'il a résolu l'énigme,
- Justifier ses choix, ses décisions (argumenter),
- Confronter sa vision à celle des autres groupes,
- Trouver des réponses aux zones d'ombre de la solution proposée par le groupe considéré.

Déroulement général de la séance :

1- Présentation de la séance et précisions sur l'ordonnancement pour le tour de parole + durée des exposés
2- Présentation par les groupes G1 à G3 de leurs productions (plan + conclusions) et correction de leur plan, compte tenu des témoignages incomplets dont ils disposaient.
3- Présentation par le groupe G4 du plan de l'accident tel qu'il l'a déduit de la totalité des témoignages dont il disposait. Critique collective (G1 à G4 + tuteur) de ce plan. Discussion orale sur les culpabilités, les responsabilités …
4- Production écrite par G4 / (G5=G1+G2+G3) de leurs conclusions d'enquête suite aux discussions
5- Présentation par le tuteur du film de l'accident et commentaires sur les niveaux de responsabilités.

FIG. C.7 – Les différentes phases pour la séance 3.

Séquence Smash

Séance 3 : mise en commun des résultats

Phase 1

Objet :

Présentation de la séance et précisions sur l'ordonnancement pour le tour de parole + durée des exposés

Rôle des acteurs :

- Le tuteur choisit l'ordre de tour de parole des groupes 1 à 3, en choisissant comme premier speaker un groupe dont les plans et synthèse sont corrects (ou proche de la solution),
- Le tuteur donnera la parole en dernier lieu au groupe 4 qui a eu tous les témoignages à disposition.

Objectifs d'apprentissage :

Ressources :

- (En entrée) Pour ordonnancer les groupes, le tuteur a accès aux différentes productions (carte et QCMs des groupes 1 à 4.
- (En entrée) Le tuteur dispose des moyens pour évaluer la justesse des productions
- (En sortie) La feuille de route prévue par le tuteur (écrit / oral) est lancée

Evénements pédagogiques significatifs :

- Feuille de route transmise aux groupes G1 -> G4,
- Parole est donnée à un des groupes G1 à G3 (phase 1 des exposés),
- Parole donnée au groupe G4 pour son exposé,
- Début exposé(i),
- Fin exposé(i) (décidé par groupe, par tuteur ou horloge),
- Fin du temps imparti au groupe(i),
- Fin de tous les exposés.

Déroulement de la phase 1 :

a) Le tuteur fait une planification concernant l'ordonnancement des présentations en tenant compte de la qualité des plans produits par les groupes G1 à G3. Pour cela, il dispose des plans types corrects pour chaque groupe compte tenu des témoignages dont disposaient les groupes considérés.

b) Le tuteur présente la séance aux 4 groupes d'apprenants. Il donne aussi l'ordre des interventions à respecter.

c) Le tuteur passe la parole successivement aux groupes 1 à 4 et fait respecter la feuille de route de chaque groupe (moment de l'intervention, durée de l'intervention, ...). Quand un groupe n'est pas speaker (mais auditeur), il écoute la présentation effectuée par le groupe speaker en cours.

d) Contrairement aux groupes 1 à 3, l'intervention du groupe 4 porte autant sur le plan de la situation d'accident que sur les conclusions. Pour les groupes 1 à 3, la phase d'échanges et de correction porte seulement sur le plan (positionnement des acteurs de l'accident).

FIG. C.8 – Détails pour la phase 1 de la séance 3.

Séquence Smash
Séance 3 : mise en commun des résultats
Phase 2

Objet :
Présentation par les groupes G1 à G3 de leurs productions (plan + conclusions) et correction de leur plan, compte tenu des témoignages incomplets dont ils disposaient.

Rôle des acteurs :
- Le tuteur est la pour gérer le déroulement de la séance et les événements pédagogiques qui se produisent,
- Les groupes 1 à 3 sont auditeurs et speakers selon qu'ils ont ou pas la parole,
- Les groupes speaker font leur présentation en montrant leur plan et en le commentant. Ils répondent à l'issue de leur exposé aux questions qui leur sont posées par le tuteur,
- Les groupes auditeur (dont fait partie le groupe 4) prennent des notes, lisent et commentent les documents qui leur sont remis.

Objectifs d'apprentissage :

Ressources :
- (En entrée) Les témoignages utilisés par chaque groupe (remis au fur et à mesure),
- (En entrée) Les plans et feuilles de notes produits par les groupes 1 à 3 durant la séance,
- (En sortie) Les plans corrigés des groupes 1 à 3,
- (En sortie) Des notes et questions que se posent les différents groupes.

Evénements pédagogiques significatifs :
- Parole est donnée à un des groupes G1 à G3 (phase 1 des exposés),
- Début exposé(i),
- Fin exposé(i) (décidé par groupe, par tuteur ou horloge),
- Fin du temps imparti au groupe(i)

Déroulement de la phase 2 :
a) La parole est donnée successivement aux groupes 1 à 3 dans l'ordre défini précédemment par le tuteur.
b) Les groupes 1 à 3 font leur présentation les uns après les autres en mettant à disposition des autres groupes les témoignages sur lesquels ils ont travaillé.
c) Les groupes auditeurs écoutent et prennent des notes mais n'interviennent pas.
d) A la fin de chaque présentation, le tuteur intervient pour faire corriger toutes les erreurs de compréhension (liées à l 'analyse des QCM) et de positionnement sur la carte.
e) A la fin des 3 exposés, le tuteur met à disposition des groupes 1 à 3 les trois plans corrigés

FIG. C.9 – Détails pour la phase 2 de la séance 3.

Séquence Smash

Séance 3 : mise en commun des résultats

Phase 3

Objet :

Présentation par le groupe G4 du plan de l'accident tel qu'il l'a déduit de la totalité des témoignages dont il disposait. Critique collective (G1 à G4 + tuteur) de ce plan. Discussion orale sur les culpabilités, les responsabilités …

Rôle des acteurs :

- Le tuteur est la pour gérer le déroulement de la séance et les événements pédagogiques qui se produisent,
- Les groupes 1 à 3 sont auditeurs durant la présentation du groupe 4,
- Durant la présentation du groupe 4, les groupes 1 à 3 notent leurs éventuelles questions,
- A l'issue de la présentation du groupe 4, le tuteur choisit parmi les questions écrites des groupes 1 à 3 celle qui va être posée (plusieurs seront posées en fonction du temps restant). Il passe la parole à un groupe qui va reformuler la question pour la dire oralement.
- Le groupe 4 répond à chacune des questions.
- Le tuteur fait des compléments s'il le souhaite entre chaque question.

Objectifs d'apprentissage :

Ressources :

- (En entrée) Les plans et feuilles de notes produits par le groupe 4 durant la séance,
- (En entrée) Les notes de chaque groupe 1 à 3 prises durant les présentations,
- (En entrée) Le plan correct,
- (En sortie) Le plan correct mis à dispo de chaque groupe pour les étapes suivantes,
- (En sortie) Des termes comme culpabilité, niveau de responsabilité, imprudence, …

Evénements pédagogiques significatifs :

- Parole est donnée au groupe G4 (phase 2 des exposés),
- Début exposé(4),
- Fin exposé(4) (décidé par groupe, par tuteur ou horloge),
- Question à poser (question q, groupe i),
- Commentaire,
- Fin du temps imparti au groupe(i)

Déroulement de la phase 3 :

a) La parole est donnée au groupe 4.
b) Le groupe 4 fait la présentation de sa carte et de ses conclusions
c) Le tuteur choisit parmi les questions des groupes 1 à 3 celles qui vont permettre de corriger la carte. Et les fait poser oralement. Il complète ou fait un complément si nécessaire
d) Le tuteur choisit ensuite les questions qui portent sur les problèmes de responsabilité, d'imprudence. Et les fait poser en commentant par introduction du vocabulaire adéquat
e) Le jeu des questions/réponses se termine quand les questions pendantes des groupes 1 à 3 sont évacuées.

FIG. C.10 – Détails pour la phase 3 de la séance 3.

Liste des acronymes

AGL	Atelier de Génie Logiciel
AIED	*Artificial Intelligence and EDucation research*
API	*Application Program Interface*. Interface de programme d'application
CCM	*Corba Component Model*
CMS	*Content Management System*
CPL	Composant Pédagogique Logiciel
CPM	Co-operative PBL Metamodel
CSCL	*Computer-Supported Collaborative Learning*
CSCW	*Computer-Supported Cooperative Work*
EIAH	Environnement Informatique pour l'Apprentissage Humain
EJB	*Enterprise Java Bean*
EML	*Educational Modeling Language*
EML-OUNL	*Educational Modeling Language of the Open University of NetherLands*
ESC	*Educational Software Component*
FAD	Formation A Distance
FOAD	Formation Ouverte et A Distance
IHM	Interface Homme-Machine
J2EE	*Java 2 Platform, Enterprise Edition*, ensemble d'API dédiées au développement d'applications d'entreprise
LCMS	*Learning Content Management System*
LMS	*Learning Management System*
LO	*Learning Object*
LOM	*Learning Object Medata*
MDA	*Model Driven Architecture*
NTIC/TIC	(Nouvelles) Technologies de l'Information et de la Communication
OMG	*Object Management Group*
OO	*Object-Oriented*
PBL	*Problem Based Learning*
SCORM	*Sharable Content Object Reference Model*
TCAO	Travail Coopératif Assisté par Ordinateur
TICE	Technologies de l'Information et de la Communication appliquées à l'Education
UML	*Unified Modeling Language*
XMI	*XML Metadata Interchange*
XML	*eXtensible Markup Language*

Table des figures

Liste des tableaux

Bibliographie

[ACC] ACCORD (projet). Assemblage de Composants par Contrats en environnement Ouvert et Réparti. http ://www.infres.enst.fr/projets/accord/.

[ACT01] ACTIS. Smash Scenario, 2001. http ://www.projectboxes.co.uk/index.htm.

[ADL] ADL. Advanced distributed learning initiative. http ://www.adlnet.org/.

[AFN] AFNOR. Afnor/cn tiefa technologies de l'information pour l'education, la formation et l'apprentissage. http ://www.afnor.fr/portail.asp.

[Age] AgentSheets. Website. http ://www.agentsheets.com/.

[AIC00] AICC. Aviation industry cbt committee, 2000. http : ://www.aicc.org/.

[Alb00] Albion, Peter. Evaluating the Implementation of Problem-based Learning in Interactive Multimedia. In *ASCILITE 2000)*, Coffs Harbour, AUSTRALIA, 2000.

[ALI] ALIC. Advanced learning infrastructure consortium. http ://www.alic.gr.jp.

[All02] Allert, H. and Dhraief, H. and Nejdl, W. Meta-Level Category "Role" in Metadata Standards for Learning : Instructional Roles and Instructional Qualities of Learning Objects. In *The second conference on Computational Semiotics for Games and New Media (COSIGN'02)*, Augsburg, Germany, September 2 - 4 2002.

[All04] Allert, H. Coherent Social Systems for Learning : An Approach for Contextualized and Community-Centred Metadata. *Journal of Interactive Media in Education. Special Issue on the Educational Semantic Web*, 2(ISSN :1365-893X), 2004. www-jime.open.ac.uk/2004/2.

[Ani01] Anido, L. and Llamas, M. and Fernandez, M.J. and Caeiro, M. and Santos, J. and Rodriguez, J. A Component Model for Standardized Web-based Education. In *WWW'10, Hong Kong*, 2001.

[Ani02] Anido, L. and and Santos-Gago, J. and Rodriguez-Estévez, J. and Caeiro-Rodriguez, M. and Fernandez-Iglesias, M. and Llamas-Nistal, M. A Step ahead in E-learning Standardization : Building Learning Systems from Reusable and Interoperable Software Components. In *WWW'2002*, Honolulu, Hawaii, USA, 2002.

[Apv03] Apvrille, L. and de Saqui-Sannes, P. and Khendek, F. TURTLE-P : un profil UML pour la validation d'architectures distribuées. In *CFIP'2003 (Colloque Francophone sur l'Ingénierie des Protocoles)*, Paris, France, octobre 2003.

[Apv04] Apvrille, Ludovic and Courtiat, Jean-Pierre and Lohr, Christophe and De Saqui-Sannes, Pierre. TURTLE : A Real-Time UML Profile Supported by a Formal Validation Framework. *accepted for publication, IEEE Transactions on Software Engineering*, July 2004.

[ARI] ARIADNE. Alliance of remote instructional and distribution networks for europe alliance of remote instructional and distribution networks for europe. http ://www.ariadne-eu.org/.

[Arm02] Armarego, Jocelyn. Advanced Software Design : a Case in Problem-based Learning. In *15th Conference on Software Engineering Education and Training (CSEET'02)*, Covington, Kentucky, 2002.

[Bak00] Baker, M. The roles of models in artificial intelligence and education research : a prospective view. *International Journal of Artificial Intelligence and Education*, 2000(11) :122–143, 2000.

[Bal91] Balacheff, N. Contribution de la didactique et de l'épistémologie aux recherches en eiao. In *Actes des XIIIemes Journées Francophones sur l'informatique*, IMAG et Université de Genève, Grenoble, 1991.

[Bar80] Barrows, H.S. and Tamblyn, R.M. *Problem-Based Learning. An Approach to Medical Education.* Springer Publishing Company : New York, 1980.

[Bar97] Bardram, J. E. Plans as situated action : An activity theory approach to workflow systems. *Proceedings of ECSCW '97*, pages p. 17–32, 1997.

[Bar02] Barros, B. and Verdejo, M. F. and Read, T. and Mizoguchi, R. Applications of a Collaborative Learning Ontology. In *MICAI'2002 Mexican International Conference on Artificial Inteligence*, pages pp. 301–310. LNAI, Springer-Verlag, 2002.

[Bar04] Barbier, F. and Cauvet, C. and Oussalah, M. and Rieu, D. and Bennasri, S. and Souveyet, C. Concepts clés et techniques de réutilisation dans l'ingénierie des systèmes d'information. *L'objet*, 10, issue 1 :11–35, 2004.

[Bel01] Bellier, Sandra. *Le E-learning.* Number ISBN : 2878803671. Editions liaisons edition, 2001.

[Bes02] Bessagnet, Marie-Noëlle and Marquesuzaà, Christophe and Nodenot, Thierry and Sallaberry, Christian and Laforcade, Pierre. Information systems and educational engineering : Bridging two concepts through meta modelling. In *Proceedings of the 17th Congrès international des technologies de l'information (WCC'2002)*, Montréal, Québec, August 25-30 2002. IFIP WCC (World Computer Congress).

[Bet03] Betbeder-Matibet, Marie-Laure. *Symba : un environnement malléable support d'activités collectives en contexte d'apprentissage.* PhD thesis, Thèse de doctorat de l'Université du Maine, 2003.

[Bir00] Birbilis, G. and Koutlis, M. and Kyrimis, K. and Tsironis, G. and Vasiliou, G. E-slate : A software architectural style for end-user programming. In *22nd International Conference on Software Engineering (ICSE 2000)*, Limerick, Ireland, 2000.

[Bla96] Blanc, Frédéric. *Des mécanismes pour la modélisation d'environnements interactifs d'apprentissage avec ordinateur, basés sur la simulation qualitative.* PhD thesis, Thèse de Doctorat de l'Université Paul Sabatier. N°2398, 11 juillet 1996.

[Boo00] Booch, G. and Rumbaugh, J and Jacobson, I. *The Unified Modeling Language User Guide.* Number ISBN : 0-201-57168-4 in The Addison-Wesley Object Technology Series. Addison Wesley Professional, 2000.

[Bou91] Boud, D., and Feletti, G. *The Challenge of Problem-Based Learning.* London : Kogan Page, 1991.

[Bou00] Bourguin, Grégory. *Un support informatique à l'activité coopérative fondé sur la Théorie de l'Activité : le projet DARE.* PhD thesis, Thèse de doctorat de l'Université des sciences et technologies de Lille. N°2753, 2000.

[Bou03] Bouzeghoub, Amel and Carpentier, Claire and Defude, Bruno and Duitama, Freddy. A model of reusable educational components for the generation of adaptative courses. In *CAiSE Workshops*, Klagenfurt, Austria, 2003.

[Bra03] Brassard, C. and Daele, A. Un outil réflexif pour concevoir un scénario pédagogique intégrant les TIC. *Communication au colloque EIAH*, 2003.

[Bre02] Breton, Erwan. *Contribution à la representation de processus par des techniques de méta-modélisation.* PhD thesis, Thèse de Doctorat de l'Université de Nantes. N°0366-066, 24 juin 2002.

[Bro89] Brown, J.S., Collins, A. and Duguid, S. Situated cognition and the culture of learning. In *Educational Researcher*, volume Vol. 18, No. 1, pages pp.32–42. 1989. accessible à l'URL : http ://www.ilt.columbia.edu/ilt/papers/JohnBrown.html.

[Bru03] P. Brusilovsky. Developing adaptive educational hypermedia systems : From design models to authoring tools. In S. Blessing T. Murray and S. Ainsworth (eds.), editors, *Authoring Tools for Advanced Technology Learning Environment.* Dordrecht : Kluwer Academic Publishers, 2003.

[Bru04] Brusilovsky, P. A distributed architecture for adaptive and intelligent leaning management systems. In M. Driscoll (eds.) and T. C. Reeves, editors, *Proceedings of Workshop on Intelligent Learning Management Systems at AI-ED'2004*, pages pp. 106–114, Sydney, Australia, University of Sydney, 2004. AACE.

[Bus03] Business Interactif. E-learning : Présentation générale et solutions logicielles. Technical report, 2003.

[Cam96] Camp, Gwendie. A Paradigm Shift or a Passing Fad ? In *Medical Education Online*, volume vol.3, no.2. 1996.

[Car03] Caron, Pierre-André and Le Pallec, Xavier and Derycke Alain. Exploitation de l'approche par modèles pour un développement accéléré et intuitif d'applications de eLearning. Présentation au gdri3 eiah, Paris, 2003.

[Cav02] Cavaliere, Domenico. *DAMeSI : un profil UML pour l'évaluation des performances des systèmes d'automatisation distribués.* PhD thesis, Thèse de Doctorat de l'Institut National Polytechnique de Lorraine., 14 juin 2002.

[CEN] CEN. Comité européen de normalisation.

[Cen03] MASIE Center. Making sense of learning specifications and standards : A decision maker's guide to their adoption, 2^e edition. Technical report, november, 2003 2003.

[Cis01] Cisco Systems. Reusable Learning Object Strategy v4.0, 2001.

[Col89] Collins, A. and Brown, J.S. and Newman, S.E. Cognitive apprenticeship : Teaching the crafts of reading, writing, and mathematics. In In L. B. Resnick (Ed.), editor, *Knowing, Learning and Instruction : Essays in Honor of Robert Glaser*, pages (pp.453–494). Hillsdale, NJ : Lawrence Erlbaum Associates, 1989.

[Col00] Collectif de Chasseneuil. Formations Ouvertes et à Distance - L'accompagnement pédagogique et organisationnel, 2000. http ://ressources.algora.org/reperes/tel/ccfod.pdf.

[Col03] Collectif Autrans'03. Cours 1 : modélisation des connaissances enjeu d'apprentissage, 2003.

[Con00] Conallen, Jim. *Building Web Applications with UML*. Number ISBN : 0201730383 in The Addison-Wesley Object Technology Series. Addison Wesley Professional, 2000.

[Cor] Dublin Core. The dublin core metadata initiative. http ://dublincore.org/.

[Cos02] Costagliola, G. and De Lucia, A. and Orefice, S. and Polese, G. "a classification framework for the design of visual languages". *Journal of Visual Languages and Computing*, 13 :573–600, 2002.

[Cos03] Costagliola, G. and Ferrucci, F. and Polese, G. and Scanniello, G. A Visual Language for Designing and Presenting e-learning Activities. In *Proceedings of the IEEE International Conference on Information Technology : Research and Education (ITRE'2003)*, Newark, New Jersey, USA, August 10-13 2003.

[CRE] CREATE. A Component Repository and Environment for Assembly of Teaching Environments project. http ://ir.chem.cmu.edu/create/.

[Cre97] Crevier, F. *Conception et validation d'une méthode d'ingénierie didactique*. PhD thesis, Thèse de Doctorat de l'Université de Montréal, 1997.

[Cre02] Crepuq. Les normes et standards de la formation en ligne - État des lieux et enjeux, 2002. http ://www.crepuq.qc.ca/IMG/pdf/norm-0210-d-rapport.pdf.

[CSC03] CSC. Les learning management systems (lms) - solutions logicielles de gestion de la formation - étude annuelle. Technical report, 2003.

[Dae02] Daele, A. and Lusalusa, S. *Quels nouveaux rôles pour les formateur d'enseignants ?* Bruxelles : DeBoeck, b. charlier and d. peraya (eds.) edition, 2002.

[Dae03] Daele, A. and Brassard, C. and Esnault, L. and O'Donoghue, M. and Uyttebrouck, E. and Zeileger, R. Wp2. conception, mise en oeuvre, analyse et évaluation de scénarios pédagogiques recourant à l'usage des technologies de l'information et de la communication. Rapport recre@sup, 2003. http ://tecfa.unige.ch/proj/recreasup/rapport/WP2.pdf.

[Dal03] Dalziel, J. Discussion Paper for Learning Activities and Meta-data. Technical report, 2003. Heerlen : Open University of the Netherlands.

[Dav01] David, Bertrand. IHM pour les collecticiels. In Hermès, Editions, editor, *Les télé-applications*, volume volume 13. 2001.

[DEA00] DEAT-AFPA, Direction des Etudes et de l'Appui Techniques. Audit sur les potentialités «Tutor Shop 2000» Plate-forme de Téléformation de l'AFPA, 2000.

[deC02] deCesare, S. and Lycett, M. and Patel, D. Business modelling with uml : Distilling directions for future research. In *4th International Conference on Enterprise Information Systems*, Ciudad Real, Spain, 2002.

[deO01] deOliveira, M.C.F. and Turine, M.S. and Masiero, P.C. A statechart-based model for modeling hypermedia applications. *ACM Transactions on Information Systems*, 2001.

[Des96] Deschênes, A.J. and Bilodeau, H. and Bourdages, L. and Dionne, M. and Gagné, P. and Lebel, C. and Rada-Donath, A. Constructivisme et formation à distance. *Revue DistanceS*, Vol 1, num 1, 1996.

[Des00] Dessus, P. La planification de séquences d'enseignement, objet de description ou de prescription ? *Revue Française de pédagogie* , Vol. 133 :pp 101–116, 2000.

[Des01] Després, Christophe. *Modélisation et Conception d'un Environnement de Suivi Pédagogique Synchrone d'Activités d'Apprentissage à distance*. PhD thesis, Université du Maine, 2001.

[Des02] Desmoulins, C and Grandbastien, M. Des ontologies pour la conception de manuels de formation à partir de documents techniques. *Sciences et Techniques Educatives (STE)*, Vol.9(num 3) :pp. 37–86, 2002.

[Des04] Desfray, Philippe. Réussir la modélisation uml des phases amont - techniques de « pré-modélisation » : un pont vers le modèle. Softeam document, 2004.

[Dew00] Dewanto, Lofi. OpenUSS Developer's Manual. Report, 2000.

[diS] diSessa, Andrea A. Boxer project. http ://www.soe.berkeley.edu/ boxer/materials.html.

[dR02] Académie de Rennes. Glossaire autour des plates-formes de travail collaboratif et de mutualisation ou formation à distance, 2002. http ://www.ac-rennes.fr/tic/glossaire/Lexique.htm.

[E-S] E-Slate. Website. http ://e-slate.cti.gr/.

[Eco01] Ecoutin, Eric. Fiche pratique no.1 : les utilisations d'une plate-forme. Technical report, 2001.

[EdN] EdNA. Education network australia. http ://www.edna.edu.au http ://standards.edna.edu.au.

[edu] eduSource. A pan-canadian collaborative project. http ://www.edna.edu.au http ://www.netera.ca http ://www.careo.org http ://www.canarie.ca.

[Ell94] Ellis, C. and Wainer, J. A conceptual model of groupware. In *Proceedings of CSCW'94*, pages 79–88. ACM Press, 1994.

[Eng87] Engeström, Y. Learning by expanding. an activity-theoretical approach to development research. *Helsinki : Orienta-konsultit*, 1987.

[Eri00] Eriksson, Hans-Erik and Penker, Magnus. *Business Modeling with UML : Business Patterns at Work*. Number ISBN : 0471295515. Robert Ipsen, 2000.

[Eri04] Eriksson, Hans-Erik and Penker, Magnus and Lyons, Brian and Fado, David. *UML 2 Toolkit*. Number ISBN : 0-471-46361-2. Wiley Publishing, Inc., 2004.

[ESC] ESCOT. Educational Software Components of Tomorrow Project. http ://www.escot.org/.

[Etc02] Etcheverry, Patrick. *SAPIC : Modélisation & Spécification de la Coordination Interne aux Processus d'Activités*. PhD thesis, Thèse de doctorat de l'Université de Pau et des pays de l'Adour, 2002.

[Eur02] European Institute For E-Learning. Le guide des solutions de e-formation. étude européenne des outils de e-formation, 2002.

[Evi00] Evitts, Paul. *A UML Pattern Language*. Number ISBN : 1-57870-118-X, 288 pages. New Riders Publishing, 2000.

[EXP] EXPLORATORIES Project. http ://cs.brown.edu/exploratories/.

[Fae02] Faerber, Richard. *La place des TICE en formation initiale et continue à l'enseignement : bilan et perspectives*, chapter Le groupe d'apprentissage en formation à distance : ses caractéristiques dans un environnement virtuel, pages 99–128. Université de Sherbrooke, Larose F and Karsenti T. (Ed.). Sherbrooke : Editions du CRP edition, 2002.

[Fae03] Faerber, Richard. Groupements, processus pédagogiques et quelques contraintes liés à un environnement virtuel d'apprentissage. In *Proceedings of EIAH'03*, 2003.

[Fer00] Ferraris C., and Martel C. Regulation in groupware : the example of a collaborative drawing tool for young children. *Proceedings of CRIWG'2000, 6th international workshop on groupware*, pages p. 119–127, 2000.

[Fer02] Ferrucci, F. and Tortora, G. and Vitello, G. Exploiting Visual Languages In Software Engineering. In Singapore World Scientific Publishing Co., editor, *Handbook of Software Engineering and Knowledge Engineering*. 2002.

[FFF02] FFFOD. Le B.A.BA de la FOAD, 2002. http ://www.fffod.org.

[Fin95] Finkle, S.L., and Torp, L.L. Introductory Documents. (available from the center for problem-based learning, illinois math and science academy, 1500 west sullivan road, aurora, il 60506-1000), 1995.

[For02] Forsythe, Franck. Problem-Based Learning. *Handbook for Economics Lecturers*, 2002.

[Fow96] Martin Fowler. *Analysis Patterns : Reusable Object Models*. Object Technology Series. Addison Wesley Professional, October 1996.

[Fow03] Fowler, Martin. *UML Distilled - Third Edition - A Brief Guide to the Standard Object Modeling Language*. Number ISBN : 0-321-19368-7 in The Addison-Wesley Object Technology Series. Addison Wesley Professional, 2003.

[Gal94] Gallagher, J.J. and Gallagher, S.A. *Teaching the gifted child*. Boston : Allyn and Bacon, 1994.

[Gam95] Gamma, E. and Helm, R. and Johnson, R. and Vlissides, J. *Design Patterns : Elements of Reusable Object Oriented Software*. Number 395 pages. Addison-Wesley, 1995.

[Geo01] Sébastien George. *Apprentissage collectif à distance. SPLASH : un environnement informatique support d'une pédagogie de projet*. PhD thesis, Thèse de doctorat de l'Université du Maine, 2001.

[GES] GESTALT. Getting educational systems talking across leading edge technologies. http ://www.fdgroup.co.uk/gestalt/about.html.

[Gru93] Gruber, T.R. A translation approach to portable ontologies. *Knowledge Acquisition*, Vol.5(num 2) :pp.199–220, 1993.

[Gua95] Guarino, N. and Giaretta, P. Ontology and Knowledge Bases, Towards a Terminological Clarification. In *KB&KS '95*. IOS Press, Amsterdam, 1995.

[Gua98] Guarino, N. Formal Ontology in Information Systems. In *Proceedings of FOIS'98*, pages pp. 3–15. IOS Press, Amsterdam, 6-8 June 1998.

[Gua00a] Guareis de Farias, C. R., Ferreira Pires, L., and van Sinderen, M. A Component-Based Groupware Development Methodology. In *Fourth International Enterprise Distributed Object Computing Conference (EDOC'00)*, Makuhari, Japan, September 25 - 28 2000.

[Gua00b] Guareis de Farias, C. R., Ferreira Pires, L., and van Sinderen, M. A conceptual model for the development of CSCW systems. In *Fifth International Conference on the Design of Cooperative Systems (COOP 2000)*, pages pp. 189–204, Sophia Antipolis, France, 2000.

[Har87] Harel, David. Statecharts : A visual formalism for complex systems. *Science of Computer Programming*, 8, issue 3(0167-642) :231–274, 1987.

[Har04] Harel, David and Rumpe, Bernhard. Meaningful modeling : What's the semantics of «semantics» ? *IEEE Computer*, October (Vol. 37, No. 10) :p. 64–72, 2004.

[Hei04] Heiwy, Véronique. Un modèle de ressources pédagogiques générique pour la FOAD. In *Colloque Miage et e-Mi@ge (Méthodes Informatiques Appliquées à la Gestion des Entreprises)*, Marrakech, 2004. http ://e-miage.ups-tlse.fr/colloque/papiers/V.HEIWY.pdf.

[Hen01] Hennicker, Rolf and Koch, Nora. Systematic design of web applications with uml. In *UML : systems analysis, design and development issues*, volume chapter 1, pages 1–20. Idea Group Publishing, 2001.

[Hon93] Honebein, P. and Duffy, T.M. and Fishman, B. Constructivism and the design of learning environments : Context and authentic activities for learning. In Thomas M. Duffy, Joost Lowyck, and David Jonassen (Eds.), editor, *Designing environments for constructivist learning*. CHeidelberg : Springer-Verlag, 1993.

[IMS] IMS. Ims global learning consortium, inc. http ://www.imsproject.org/.

[IMS01] IMS. IMS Meta-data Specification Version 1.2.1 Final Release. Technical report, 2001-October-01 2001.

[IMS03a] IMS. IMS Content Packaging Version 1.1.3 Final Specification . Technical report, 2003-June-12 2003.

[IMS03b] IMS. IMS Learning Design Best Practice and Implementation Guide. Technical report, 2003-January-2003 2003.

[IMS03c] IMS. IMS Learning Design Version 1.0 Final Specification . Technical report, 2003-February-13 2003.

[IMS03d] IMS. IMS Simple Sequencing Version 1.0 Final Specification . Technical report, 2003-March-20 2003.

[Ina00] Inaba, Akiko and Supnithi, Thepchai and Ikeda, Mitsuru and Mizoguchi, Riichiro and Toyoda, Jun'ichi. How Can We Form Effective Collaborative Learning Groups Theoretical jistification of "Opportunistic Group Formation" with ontological engineering. In *Proc. of ITS'2000*, pages pp.282–291, Montreal Canada, 2000.

[Ina04] Inaba, Akiko and Mizoguchi, Riichiro. Learning Design Palette : An Ontology-aware Authoring System for Learning Design. In *Proc. of International Conference on Computers in Education (ICCE2004)*, Melbourne, Australia, 2004.

[ITD] NATO ITD. North atlantic treaty organization, integrated technical data.

[Joh97] Ralph E. Johnson. Frameworks = Components + Patterns. *Communications of the ACM*, 40(10) :39–42, 1997.

[Jon94] Jones, B. and Valdez, G. and Norakowski, J. and Rasmussen, C. Designing Learning and Technology for Educational Reform, 1994.

[Kee96] Keegan, D. *Foundations of Distance Education*. Third edition. routledge studies in distance education edition, 1996.

[Kop00] Koper, Rob. From change to renewal : Educational technology foundations of electronic environments. Technical report, Educational Expertise Technology Centre, Open University of the Netherlands, 2000.

[Kop01] Koper, Rob. Modeling units of study from a pedagogical perspective : the pedagogical metamodel behind eml. first draft, version 2. Technical report, Educational Expertise Technology Centre, Open University of the Netherlands, june 2001.

[Kop02] Koper, Rob. Educational modelling language : adding instructional design to existing specifications. Technical report, Educational Expertise Technology Centre, Open University of the Netherlands, 2002.

[Kos96] Koschmann, T. *Computer Supported Collaborative Learning : theory and practice of an emerging paradigm*. Laurence Erlbaum Associates, Mahwah, New Jersey, 1996.

[Kou97] Koutlis, M. and Kynigos, Ch. and Oikonomou, A. and Tsironis, G. Empowering logo through a component-oriented approach. In *7th Eurologo Conference*, Boudapest, Hungary, 1997.

[Kou99] Koutlis, Manolis and Birbilis, George and Kynigos, Chronis and Tsironis, George and Dekoli, Margarita and Kyrimis, Kriton and Vasiliou, George and Hadzilacos, Thanasis and Siouti, Xenia. E-slate : a kit of educational components. 1999.

[Kru95] Kruchten, Philippe. Architectural blueprints - the "4+1" view model of software architecture. *IEEE Software 12(6)*, pages 42–50, 1995.

[Kru01] Kruchten, Philippe. What Is the Rational Unified Process ?, 2001. http ://www-106.ibm.com/developerworks/rational/library/content/RationalEdge/jan01/WhatIsthe-RationalUnifiedProcessJan01.pdf.

[Kuu96] Kuutti, K. Activity theory as a potential framework for human-computer interaction research. *Context and Consciousness*, pages 17–44, 1996.

[Kyn01] Kynigos, C. E-slate logo as a basis for constructing microworlds with mathematics teachers. In *Proceedings of the Ninth Eurologo Conference*, pages 65–74, Lintz, Austria, 2001.

[Kyn02] Kynigos, C., Koutlis, M. E-slate : A 'black-and-white box' approach to component computing. In *Paper presented at the Annual Meeting of the American Educational Research Association*, New Orleans, USA, 2002.

[Lac01] Lacek, Kateherine Ann. *A problem-based learning curriculum unit for high-school chemistry*. PhD thesis, California University of Pennsylvania, 2001.

[Laf03a] Laforcade, Pierre and Barbier, Franck. UML Modeling for Cooperative Problem-Based Learning Situations : Towards Educational Components. In *Proceedings of the 14th IRMA International Conference (IRMA'2003)*, Radisson Warwick Hotel Philadelphia Pennsylvania, USA, May 18-21 2003. Idea Group Publishing.

[Laf03b] Laforcade, Pierre and Barbier, Franck and Nodenot, Thierry and Sallaberry, Christian. Profiling Co-operative Problem-Based Learning Situations. In *Proceedings of the 2nd IEEE International Conference on Cognitive Informatics (ICCI'2003)*, South Bank University, London, UK, August 18-20 2003. IEEE Computer Society Press.

[Laf04] Laforcade, Pierre and Barbier, Franck. *Instructional Technologies : Cognitive Aspects of Online Programs*, chapter UML-based Modeling of Educational Components for Cooperative Problem-based Learning Situation Design, pages 165–191. Idea group Inc., 2004.

[Lah04] Lahaye, Philippe. *Les systèmes de gestion de contenus : description, classification et évaluation.* PhD thesis, Mémoire de fin d'études d'Ingénieur Informatique du CNAM, mai 2004.

[Lar01] Larman, Craig. *Applying UML and Patterns : An Introduction to Object-Oriented Analysis and Design and the Unified Process (2nd Edition).* Number ISBN : 0130925691. Prentice Hall PTR ; 2 edition (July 13, 2001), 2001.

[Lav91a] Lave, J. Situated learning in communities of practice. In L.B. Resnick, J.M. Levine, and S.D. Teasley (Eds), editor, *Perspectives on socially shared cognition*, pages pp. 63–82. 1991.

[Lav91b] Lave, J. and Wenger, E. *Situated Learning : Legitimate Peripheral Participation.* Cambridge, UK : Cambridge University Press, 1991.

[Lec03] Leclercq, Eric and Savonnet, Marinette and Terrasse, Marie-Noëlle. Adaptation d'une plateforme d'e-learning à un modèle pédagogique. In *Proceedings of 3rd Annual Ariadne Conference*, Katholieke Universiteit Leuven, Belgium, 2003.

[Leg01] Legros, Denis and Pudelko, Béatrice and Crinon, Jacques. Les nouveaux environnements technologiques et l'apprentissage collaboratif. *Apprendre avec le multimédia et Internet*, march/april :pp 203–214, 2001.

[LeP03] LePallec, Xavier and Derycke, Alain. RAM3 : towards a more intuitive MOF meta-modelling process. In *Proceedings of the 7th World Multiconference on Systemics, Cybernetics and Informatics (SCI'2003)*, Orlando, Florida, USA, 2003.

[LOM02] LOM. LOM (Learning Object Metadata) working draft v4.1, 2002. http ://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf.

[LTS] LTSC. Institute of electrical and electronics engineers (ieee) learning technology standards comittee (ltsc). http ://ltsc.ieee.org/.

[LTS01] LTSC'IEEE. IEEE P1484.1/D9, 2001-11-30 Draft Standard for Learning Technology - Learning Technology Systems Architecture(LTSA), 2001. http ://edutool.com/ltsa/09/IEEE_1484_01_D09_LTSA.pdf.

[Mar98] Marquesuzaa, Christophe. *OMAGE : Outils et méthodes pour la spécification des connaissances au sein d'un atelier de génie éducatif.* PhD thesis, Thèse de doctorat de l'Université de Pau et des pays de l'Adour, 1998.

[Mba03] Mbala Hikolo, Aloys. *Analyse, conception, spécification et développement d'un système multi-agents pour le soutien des activités en formation à distance.* PhD thesis, Thèse de doctorat de l'Université de Franche-Comté. N°977, 2003.

[Mei02a] Meirieu, Philippe. *Apprendre... Oui mais comment ?* Collection Pédagogies. ESF Edition, 18^e edition, 2002.

[Mei02b] Meirieu, Philippe. Guide méthodologique pour l'élaboration d'une situation-problème. In ESF Edition, 18^e, editor, *annexe to : Apprendre... oui, mais comment ?* Collection Pédagogies, 2002.

[Mel02] Mellor, Stephen J. and Balcer, Marc J. *Executable UML - A Foundation for Model-Driven Architecture.* Number ISBN : 0201748045. Addison-Wesley Pub Co ; 1st edition (May 15, 2002), 2002.

[Mel03] Melby, Geir. Using J2EE Technologies for Implementation of ActorFrame Based UML2.0 Models. Masters thesis in information and communication technology, 2003.

[Mel04] Melby, Geir. Open distributed systems - Executable UML. Report, 2004.

[Mia00] Miao, Yongwu. *Design and Implementation of a Collaborative Virtual Problem-Based Learning Environment.* PhD thesis, Die Dissertationsschrift Vorgelegt am Fachbereich Informatik Der Technischen Universität Darmstadt von, Darmstadt 2000.

[Mir04] Miralles, André and Libourel, Thérèse and Maurel, Pierre. Méthodologie d'aide à la conception de Systèmes d'Information Géographique. In *Colloque ALCAA (Agents Logiciels - Coopération Apprentissage - Activité humaine)*, Montpellier, France, June 17 - 18 2004.

[Miz00] Mizoguchi, Riichiro and Bourdeau, Jacqueline. Using ontological engineering to overcome ai-ed problems. *International Journal of Artificial Intelligence in Education*, Vol.11(num 2) :pp.107–121, 2000.

[Moo00] Moore, Graham. Problem Based Learning for the Design Process. In *(ASCILITE'2000)*, 2000.

[Mot03] Mottais, Frédéric. Web Sémantique pour le E-learning dans le cadre d'environnements d'apprentissage coopératif à base de situation-problème, 2003.

[Mul98] Muller, Pierre-Alain. *Modélisation objet avec UML.* Eyrolles edition, 1998.

[Nai96] Naidu, Som and Oliver, Mary. Computer-supported Collaborative Problem-based Learning : An Instructional Design Architecture for Virtual Learning in Nursing Education. In *Journal of Distance Education/Revue de l'enseignement à distance* , number ISSN : 0830-0445. 1996.

[Nar96] Nardi, B.A. *Context and consciousness : activity theory and Human-Computer Interaction.* Number ISBN : 0262140586. MIT Press, 1996.

[Nod01] Nodenot, Thierry. A case for an adult educational technology. In D.Watson Edition and J.Andersen, editors, *IFIP 7th World Conference on Computers in Education*, pages pp. 115–124, Copenhagen (Denmark), 2001. Kluwer Academic Publishers.

[Nod03a] Nodenot, Thierry and Laforcade, Pierre and Marquesuzaa, Christophe and Sallaberry, Christian. Knowledge Modelling of Co-operative Learning Situations : Towards a UML profile. In *Proceedings of the 11th International Conference on Artificial Intelligence in Education (AIED'2003)*, Sydney, Australia, July 20-24 2003. International AI-ED Society.

[Nod03b] Nodenot, Thierry and Laforcade, Pierre and Sallaberry, Christian and Marquesuzaa, Christophe. A UML Profile incorporating separate viewpoints when modeling Co-operative Learning Situations. In *Proceedings of the IEEE International Conference on Information Technology : Research and Education (ITRE'2003)*, Newark, New Jersey, USA, August 10-13 2003.

[Nod04] Nodenot, Thierry and Marquesuzaa, Christophe and Laforcade, Pierre and Sallaberry, Christian. Model based Engineering of Learning Situations for Adaptive Web Based Educational Systems. In *Proceedings of the ACM 13th International World Wide Web Conference (WWW'2004)*, New York, USA, May 17-22 2004.

[Nor03] Northwood, M.D. and Northwood, M.G. and Northwood, D.O. Problem-Based Learning (PBL) : from the health sciences to engineering to value-added in the workplace. *Global Journal of Engineering Education*, Vol. 7(No. 2), 2003.

[NSF] NSF NSDL. National Science Digital Library program. http ://www.ehr.nsf.gov/ehr/due/programs/nsdl/.

[OMG99] OMG. White Paper on the Profile mechanism v.1.0, Analysis and Design Platform Task Force. Report ad/99-04-07, Avril 1999.

[OMG00] OMG. Complete MOF 1.3 specification. OMG Final Adopted Specification formal/00-04-03, 2000.

[OMG01a] OMG. Common Warehouse Metamodel specification. OMG Final Adopted Specification ad/01-02-01, 2001.

[OMG01b] OMG. The Software Process Engineering Management Metamodel (SPEM). Technical Report ad/2001-02-08, February 14 2001.

[OMG02] OMG. Software Process Engineering Metamodel (SPEM). Technical Report formal/2002-11-14, November 2002.

[OMG03a] OMG. MDA Guide Version 1.0. OMG Final Adopted Specification omg/2003-05-01, 2003.

[OMG03b] OMG. UML 2.0 Infrastructure Specification. OMG Adopted Specification ptc/03-09-15, 2003.

[OMG03c] OMG. UML 2.0 Supertructure Specification. OMG Final Adopted Specification ptc/03-08-02, 2003.

[OMG03d] OMG. Unified Modeling Language v1.5 Specification. Report formal/03-03-01, March 2003.

[OMG03e] OMG. XML Metadata Interchange (XMI), v2.0. Report formal/03-05-02, May 2003.

[OMG04] OMG. UML Profile for Enterprise Distributed Object Computing (EDOC). Omg final adopted specification version 1.0, 2004. http ://www.omg.org/technology/documents/formal/edoc.htm.

[Ope04] OpenUniversiteitNederland. Learning forever ! a profile of open universiteit nederland in 2004. Présentation, 2004.

[ORA00] ORAVEP. Étude comparative technique et pédagogique des plates-formes pour la formation ouverte et à distance. Technical report, novembre 2000.

[Oud03] Oudot, Cécile. Modélisation UML d'une situation d'apprentissage coopérative, 2003.

[Paq97] Paquette, G. and Crevier, F. and Aubin, C. Méthode d'ingénierie d'un système d'apprentissage (MISA). *Revue Informations In Cognito*, (8), 1997.

[Paq99] Paquette, G. Meta-knowledge Representation for Learning Scenarios Engineering. In *Proceedings of AI-Ed'99, AI and Education, open learning environments, S. Lajoie et M. Vivet (Eds), IOS Press*, 1999.

[Paq04a] Paquette, Gilbert. Educational Modeling Language, From an Instructional Engineering Perspective. In *An article to be published in a forthcoming handbook*. 2004.

[Paq04b] Paquette, Gilbert. Instructional engineering for learning objects repositories networks. In *International Conference on Computer Aided Learning in Engineering Education (CALIE'04)*, 2004.

[Pau99] Paulo, F.B. and Masiero, P.C. and de Oliveira, M.C.F. . Hypercharts, extended statecharts to support hypermedia specifications. *IEEE Transactions on Software Engineering, USA, January/February*, 25(1) :p. 33–49, 1999.

[Per03] Pernin, Jean-Philippe. Critères pour une typologie des langages de modélisation pédagogique. Présentation au gdri3 eiah, Paris, 2003.

[Pia78] Piaget, Jean. *La notion de structure*. Unesco, Paris - La Haye - New York, mouton editeur edition, 1978.

[PRC97] PRC-IA 97. Conception d'environnements interactifs d'apprentissage avec ordinateur. tendances et perspectives. *contribution du groupe EIAO coordonnée par Balacheff N., Baron M., Desmoulins C., Grandbastien M., Vivet M., Actes des journées nationales du PRC IA*, 2000(11) :315–338, 1997.

[PRO] PROMETEUS. Promoting multimedia access to education and training in european society. http ://prometeus.org/.

[Qui03] Quignard, Matthieu and Baker, Michael and Lund, Kristine and Séjourné, Arnauld. Conception d'une situation d'apprentissage médiatisée par ordinateur pour le développement de la compréhension de l'espace du débat. In *Proceedings of EIAH'03*, 2003.

[Ran00] Ranwez-Chabert, S. *Composition Automatique de Documents Hypermédia Adaptatifs à partir d'Ontologies et de Requêtes Intentionnelles de l'Utilisateur*. Thèse de doctorat, Thèse de doctorat de l'Université Montpellier II, 2000.

[Ran02] Ranwez, Sylvie and Crampes, Michel. Instanciation d'ontologies pondérées et calcul de rôles pédagogiques. *Sciences et Techniques Educatives (STE)*, Vol.9(num 3-4), 2002.

[Raw02] Rawlings, A. and van Rosmalen, P. and Koper, R. and Rodriguez-Artacho, M. and Lefrere, P. Survey of educational modelling languages (emls). Technical report, September 19st 2002 2002.

[Rep01] Repenning, Alexander and Loannidou, Andri and Payton, Michele and Ye, Wenming and Roschelle, Jeremy. Using components for rapid distributed software-development. *IEEE Software*, march/april :pp 38–45, 2001.

[Ret02] Retalis, Simeon and Papasalouros, Andreas and Skordalakis, Manolis. Towards a generic conceptual design metamodel for web-based educational applications. In *2nd International Workshop on Web Oriented Software Technology (IWWOST'2002)*, Málaga, Spain, June 10, 2002.

[Ret03] Retalis, Simeon and Papasalouros, Andreas and Skordalakis, Manolis. Formalising the Design Process of Web-based Adaptive Educational Applications using an Object Oriented Design Model. In *Third International Workshop on Web-Oriented Software Technologies (IWWOST'2003)*, Oviedo, Asurias, July 15, 2003.

[Rhe98] Rhem, James. Problem-Based Learning : An Introduction. *The National Teaching and Learning Forum* , Vol. 8(No. 1), December December 1998.

[Rob00] Robbins, Jason E. and Redmiles, David F. Cognitive support, uml adherence, and xmi interchange in argo/uml. *Journal of Information and Software Technology*, Special issue : The Best of COSET'99, vol. 42, no. 2 :pp. 79–89, 2000.

[Rod00] Rodet, Jacques. La rétroaction, support d'apprentissage ? *Revue DistanceS*, Vol 4, num 2, 2000.

[Roq00] Roques, Pascal and Vallée, Franck. *UML en action*. Eyrolles edition, 2000. orienté process.

[Ros99] Roschelle, Jeremy and DiGiano, Chris and Koutlis, Manolis and Repenning, Alexander and Phillips, Jonathan and Jackiw, Nicolas and Suthers, Dan. Developing educational software components. *IEEE Computer*, 32(9) :2–10, 1999.

[Ros00] Roschelle, Jeremy and Pea, Roy D. and Hoadley, Christopher M. and Gordin, Douglas N. and Means Barbara M. Changing How and What Children Learn in School with Computer-Based Technologies. In *The Future of Children, Children and Computer Technology*, volume Vol. 10, No. 2. Fall/Winter 2000.

[Ros03] Rosselle, Marilyne. Etude des objectifs d'une plate-forme de coopération pour les eiah. In *Proceedings of EIAH'03*, 2003.

[Rub03] Rubart, Jessica and Dawabi, Peter. Towards UML-G : a UML Profile for Modeling Groupware. In J.A. Pino (Eds.) J.M. Haake, editor, *Groupware : Design, Implementation and Use : 8th International Workshop, CRIWG 2002, La Serena, Chile, September 1-4, 2002. Proceedings*, number ISSN : 0302-9743. Springer-Verlag Heidelberg, 2003.

[Sal02a] Sallaberry, Christian and Nodenot, Thierry and Marquesuzaà, Christophe and Bessagnet, Marie-Noëlle and Laforcade, Pierre. An attempt to design an Information System supporting Collaborative Problem-Based Learning Situations. In *Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI'2002)*, Orlando, Florida, USA, August 14-18 2002.

[Sal02b] Sallaberry, Christian and Nodenot, Thierry and Marquesuzaà, Christophe and Bessagnet, Marie-Noëlle and Laforcade, Pierre. Information modelling within a Net-Learning Environment. In *Proceedings of the 12th Conference On Information Modelling and Knowledge Bases (EJC'2002)*, Frontiers in Artificial Intelligence and Applications, Krippen, Swiss Saxony, Germany, May 27-30 2002. IOS, Amsterdam.

[Sal05] Sallaberry, Christian and Nodenot, Thierry and Laforcade, Pierre and Marquesuzaà, Christophe. Model driven development of cooperative Problem-Based Learning Situations. Implementing tools for teachers and learners from pedagogical models. In *Proceedings of the Hawaii International Conference on System Sciences (HICSS'2005)*, Hilton Waikoloa Village Big Island, Hawaii USA, January 3-6 2005. IEEE Computer Society Press.

[Sav95] Savery, John R. and Duffy, Thomas M. Problem Based Learning : An instructional model and its constructivist framework. *Educational Technology*, 35 :31–38, 1995.

[SC3] ISO/IEC JTC1 SC36. http ://jtc1sc36.org/.

[Sán01] Sánchez, Ismael and Berríos, Aida. Using the problem based learning (PBL) process in the design and construction of electronic circuits. In *ICEE 2001 (the International Conference on Engineering Education)*, Oslo/Bergen, Norway, 2001.

[Sof99] Softeam. Profiles UML et langage J : contrôlez totalement le développement d'applications avec UML. White paper, 1999.

[Spa02] Spalter, Anne Morgan. Problems with using components in educational software. In *ACM SIGGRAPH 2002, Proceedings, Conference Abstracts and Applications*, 2002.

[Ste02a] Steinkuehler, C.A. and Derry, S.J. and Hmelo-Silver, C.E. and Delmarcelle, M. Cracking the resource nut with distributed problem-based learning in secondary teacher education. *Distance education*, Vol. 23(No. 1) :pp. 23–39, 2002.

[Ste02b] Steinkuehler, C.A., Derry, S.J., Hmelo-Silver, C.E., Delmarcelle, M. Cracking the resource nut with distributed problem-based learning in secondary teacher education. In *Distance education*, volume vol.23, no.1, , pp.23-39. May 2002.

[Szy02] Szyperski, Clemens. *Component Software : beyond Object-Oriented Programming.* The Addison-Wesley Componennt Software Series. ACM Press, New York, addison-wesley edition, 2002.

[Tar97] Tarpin-Bernard, Franck. *Travail coopératif synchrone assisté par ordinateur : Approche AMF-C.* PhD thesis, Doctorat de l'Ecole Centrale de Lyon spécialité Ingénierie Informatique, 1997.

[Tar00] Tarpin-Bernard, Franck. La flexibilité dans les collecticiels. *Le temps, l'espace et l'évolutif*, 2 :449–458, 2000.

[Tch02a] Tchounikine, P. Conception des environnements informatiques d'apprentissage : mieux articuler informatique et sciences humaines et sociales. In Baron G.L., Bruillard E.(ed.), Paris : INRP - MSH - IUFM de Basse Normandie, editor, *Les technologies en éducation : Perspectives de recherche et questions vives.* 203-210, 2002.

[Tch02b] Tchounikine, P. Pour une ingénierie des environnements informatiques pour l'apprentissage humain. *Revue Information Interaction Intelligence (www.revue-i3.org)*, volume 2(n°1) :pages 59–93, 2002.

[Tho00] Thomas, John W. A Review of Research on Project-Based Learning. Executive Summary, March 2000.

[Van03] Vantroys, Thomas. *Du langage métier au langage technique, une plate-forme flexible d'exécution de scénarios pédagogiques.* PhD thesis, Thèse de Doctorat de l'Université des Sciences et Technologies de Lille. N°XXXX, 16 décembre 2003.

[Vog03] Vogten, H., and Martens, H. Open Source Learning Design Engine (software). Technical report, 2003. http ://mdlet.jtc1sc36.org/doc/SC36_WG4_N0043.pdf.

[WEB] WEB/COMP. The WEB/COMP project website. http ://dewey.soe.berkeley.edu/boxer/webcomp/index.html.

[Wen01] Wenger, Etienne. Supporting communities of practice - a survey of community-oriented technologies - version 1.3. Technical report, 2001.

[Wil00] Wiley, David A. *Connecting learning objects to instructional design theory : A definition, a metaphor, and a taxonomy.* The Instructional Use of Learning Objects. Bloomington : Association for Educational Communications and Technology., in d. wiley edition, 2000.

[Woo96] Woods, D. R. *Problem Based Learning : How to Get the Most from PBL.* McMaster University, 3rd edition edition, March 1996.

Résumé

Le sujet abordé par ce projet de thèse relève du domaine des Environnements Informatiques pour l'Apprentissage Humain (EIAH) et plus précisément de l'ingénierie des EIAH. Nous avons dans un premier temps restreint notre champ d'étude aux situations d'apprentissage de type situations-problèmes coopératives. Dans un deuxième temps, nous avons choisi d'étudier plus particulièrement comme dispositif informatique les plates-formes de formation à distance. Nous nous intéressons alors pour ce contexte précis aux modèles de conception de type *design pédagogique* dont les objectifs sont de faciliter la spécification des formations et d'agir comme supports de réflexion et de communication pour l'équipe multidisciplinaire chargée de la conception. Nous avons positionné nos objectifs comme la proposition d'un langage de modélisation pour la conception et la mise en œuvre de situations-problèmes coopératives avec la prise en compte de l'évolution actuelle des plates-formes vers des architectures *composants*. L'orientation initiale donnée à notre travail est de proposer le langage de modélisation sur la base d'une spécialisation du langage visuel orienté-objet UML.

Notre approche méthodologique, basée sur une double démarche de recherche en génie logiciel et en génie éducatif, nous a permis de constater le manque actuel en terme de langages et modèles en amont du processus de conception de scénarios d'apprentissage ainsi que le besoin en réutilisation des composants des plates-formes. Nous proposons alors de positionner notre langage pour l'expression initiale des besoins, l'analyse et la conception, en amont des travaux actuels formels et standardisés pour la conception avancée. Nous proposons d'étendre ce langage à la conception avancée de formations en intégrant des activités pédagogiques réutilisables gérées par des nouveaux composants éducatifs pour les plates-formes.

Notre travail propose ainsi une approche par méta-modélisation UML : nous avons construit le langage CPM (*Cooperative PBL Metamodel*) sous la forme du méta-modèle CPM (syntaxe abstraite), du profil UML CPM (syntaxe concrète) et d'une sémantique adaptée. L'utilisation de la richesse graphique de la notation UML fournit au langage CPM la représentation adéquate pour décrire des modèles favorisant le dialogue, la compréhension et l'implication des différents intervenants dans la conception de situations d'apprentissage. Nous avons proposé un prototype d'environnement-auteur pour le langage CPM, basé sur une adaptation d'un atelier de génie logiciel UML existant, ainsi qu'une mise à l'essai sur un cas d'étude concret : la situation-problème coopérative SMASH. Notre travail a abouti également à la proposition d'un modèle de composant éducatif CPL (Composant Pédagogique Logiciel) basé sur une modélisation UML. Ces composants logiciels métiers gèrent des activités élémentaires réutilisables dans les modèles de conception en proposant de relier les besoins pédagogiques de conception et les fonctionnalités standards proposées par les plates-formes. Le modèle que nous proposons permet de décrire, spécifier ces composants CPL. Diverses expérimentations ont permis de vérifier et valider cette proposition.

Mots-clés: EIAH, Situations-problèmes coopératives, Plates-formes de formation à distance, Modèles de conception, Méta-modélisation UML, Profil UML, Composants éducatifs