COW, une plate-forme de support d'exécution de scénarios pédagogiques.

Thomas Vantroys, Yvan Peter

Sciences et Technologies de l´Information et de la Communication pour l´Éducation et la Formation

Volume 12, 2005
Article de recherche

COW, un service de support d'exécution de scénarios pédagogiques

Thomas Vantroys, Yvan Peter [TRIGONE, Lille]

■ RÉSUMÉ : Cet article présente un service de support d'exécution de scénarios pédagogiques conçu pour être intégré dans des plates-formes de formation existantes. Dans un premier temps nous prenons un point de vue ingénierie des plates-formes afin d'identifier les processus de conception et d'utilisation ainsi que les acteurs qui y sont associés. Dans un deuxième temps nous présentons les caractéristiques nécessaires que doit offrir notre service afin que les différents acteurs puissent gérer et utiliser des scénarios pédagogiques. La suite de l'article est consacrée à la présentation de la conception et de l'utilisation du service que nous avons développé. Nous présenterons plus en détail l'implémentation et la gestion des modèles de notre service qui est basé sur un système de workflows flexible.

■ MOTS CLÉS : Scénarios pédagogiques, moteur de workflows flexible, exécution de scénarios

■ ABSTRACT : This article deals with supporting the execution of pedagogical scenarios in Learning Management Systems by the means of a workflow service. It takes an engineering point of view to identify actors, design and use processes. Based on this point of view, we define the characteristics of a service to support the actors and the pedagogical scenarios. Then, we present the design and the implementation of Cooperative Open Workflows (COW), the flexible workflow based execution service we have developed.

■ KEYWORDS : Pedagogical scenarios, workflow engine, scenario execution

1 Introduction

L'e-Learning représente une solution aux besoins croissants de formation notamment de la part des entreprises. De nombreuses institutions et entreprises ont ainsi mis en place des plates-formes de gestion de formation (ou *Learning Management System* : LMS) et organisé leur travail et leur offre de formation autour de ces nouvelles technologies. Cependant, lorsque le nombre d'apprenants grandit, il devient difficile pour les tuteurs de leur offrir un support et une attention suffisante tout en étant attentif aux personnes qui nécessitent une attention plus importante car elles éprouvent des difficultés. Ainsi, les LMS doivent fournir des mécanismes pour la gestion et l'orchestration des activités à l'intérieur des unités

d'apprentissage tout en permettant aux tuteurs d'être conscients et avertis des étudiants ayant des difficultés et permettre une adaptation des activités pour ces cas particuliers. Une manière de définir les activités qui vont prendre place au sein d'une unité d'apprentissage est de les décrire dans un scénario pédagogique. Un tel scénario définit les activités qui vont être réalisées par les apprenants et par les tuteurs, l'ordonnancement de ces activités ainsi que les objets pédagogiques et les outils qui doivent être fournis aux différents acteurs. Par exemple, le standard émergent IMS-LD (IMS, 2003) utilise une métaphore théâtrale où les activités prennent place dans différents actes qui définissent l'ordonnancement. Les activités sont associées à des rôles correspondants aux utilisateurs et à un environnement composé des objets d'apprentissage et des outils. Le bénéfice de l'utilisation de scénario pédagogique, comme indiqué par (Hummel et al., 2004), réside dans le fait que l'attention principale est mise sur les activités d'apprentissage qui doivent être réalisées pour atteindre les objectifs pédagogiques plutôt que sur les ressources pédagogiques. Dans cette approche, les activités fournissent un contexte pour l'utilisation des objets pédagogiques plutôt que de mettre l'accent sur une approche "consumériste" passive de ces derniers.

1.1 Hypothèses et objectifs de notre approche

Cet article présente nos travaux portant sur la réalisation d'un service de support d'exécution aux scénarios pédagogiques aussi bien d'un point de vue organisationnel que technique. Les hypothèses que nous avons suivies sont les suivantes :

• Les scénarios pédagogiques seront au cœur des futurs LMS.

• Les scénarios pédagogiques pourront ne pas être définis complètement car les concepteurs ne peuvent pas prévoir toutes les éventualités. Les scénarios sont donc amenés à évoluer au cours du temps. Les changements seront réalisés directement par les utilisateurs, il n'y aura aucune intervention automatique du système.

• Il doit être possible d'individualiser et de personnaliser les scénarios en fonction des utilisateurs, tout en gérant des groupes d'apprenants.

• L'interprétation des scénarios, *i.e.* leurs exécutions, peut elle-même évoluer au cours du temps ou de circonstances particulières.

Pour valider nos hypothèses nous avons défini trois objectifs principaux :

• Le premier objectif est de concevoir un service **centré sur l'utilisateur** (Bourguin et al., 2001). Ce dernier intervient tout au long du cycle de vie du système. Il participe à la création des différents modèles (orientés pédagogie ou orientés workflow) et intervient dans la configuration initiale du service en paramétrant le comportement de ce dernier en fonction du contexte courant d'utilisation. L'utilisateur modifiera éventuellement ce comportement afin de refléter les changements de contextes. Il est co-constructeur de son environnement (Bourguin, 2000).

• Le deuxième objectif, qui est lié au premier, est de repositionner les **modèles au cœur du système**. Ils doivent être constamment disponibles et manipulables afin de conserver une vue orientée métier du service. Cela permettra aux utilisateurs d'adapter plus facilement le modèle à leurs besoins.

• Le troisième objectif, lié aux deux précédents, est d'ordre technique. Le service doit mettre à disposition un ensemble de mécanismes de **flexibilité** permettant la manipulation de tous les éléments constituants notre système et leur modification de manière dynamique au cours de l'exécution. Ces changements peuvent intervenir à différents niveaux d'abstraction. Le plus haut niveau d'abstraction concerne le langage de modélisation pédagogique. Il est nécessaire d'offrir la possibilité de redéfinir tout ou partie d'un scénario

pédagogique. Le plus bas niveau d'abstraction concerne l'adaptation du comportement des différents composants logiciels qui constituent notre service.

Pour bien positionner notre approche, nous avons défini le cycle de vie d'un scénario pédagogique et identifié les différents acteurs qui y prennent part. En effet, la production et l'utilisation de scénarios pédagogiques doivent s'adosser à une organisation adéquate comme c'est le cas pour la production et l'utilisation des ressources pédagogiques.

1.2 Plan de l'article

La première partie de cet article présente la conception et l'utilisation de scénarios pédagogiques en prenant un point de vue ingénierie des plates-formes. Dans la seconde partie, nous présentons les caractéristiques nécessaires à l'exécution de scénarios pédagogiques par l'intermédiaire de plates-formes. La troisième partie présente les bases que nous avons utilisées pour construire notre solution. La quatrième partie est dédiée au service Cooperative Open Workflow, la solution que nous avons développée, basée sur les technologies de système de workflows. Nous montrons notamment de quelles manières nous pouvons répondre aux différentes contraintes définies dans la seconde partie en nous appuyant sur un scénario d'usage simple. Cette solution est ensuite comparée à d'autres approches existantes. La conclusion résume le travail réalisé et présente nos perspectives.

2 Processus de conception des scénarios pédagogiques

L'architecture et l'utilisation des plates-formes de gestion d'apprentissage deviennent de plus en plus complexes. La conception et la fourniture d'un nouveau cours implique, durant le cycle de vie complet, de nombreuses tâches qui reposent sur des acteurs hautement spécialisés. Dans les prochaines sections, nous allons présenter le processus de conception, le processus d'utilisation ainsi que les acteurs impliqués dans la définition et les différentes opérations liées à un scénario pédagogique.

2.1 Processus de conception

Le processus de conception d'un module d'enseignement, visible à la figure 1, comprend 6 phases principales. Nous pouvons le rapprocher du *processus unifié de développement* (*Rational Unified Process*: RUP) (Jacobson et al., 2000). Chaque phase permet de passer à la suivante et également de faire un retour en arrière lorsque des problèmes inhérents à la phase précédente apparaissent.

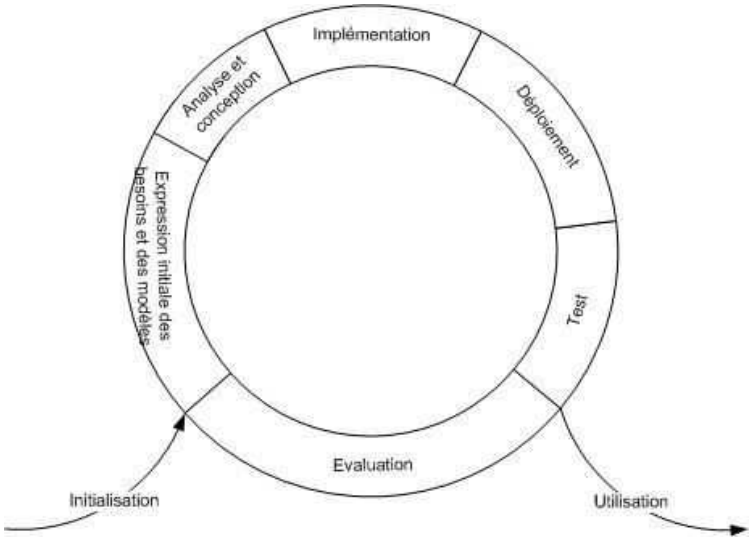

Figure 1 : Processus de développement d'un module de formation

Expression initiale des besoins. Cette phase correspond à l'expression initiale d'un module de formation. Un *enseignant* va décrire de manière informelle (non compréhensible directement par un système informatique) les différentes activités du module en fixant des objectifs pédagogiques globaux ou locaux et éventuellement des prérequis pour réaliser le module. Il précisera également les ressources pédagogiques qui seront nécessaires pour l'accomplissement du module. Dans le RUP, cette phase correspond à l'élaboration des cas d'utilisation (*use case*) définis en UML (Muller, 1999). Ces cas d'utilisation présentent l'avantage d'être compréhensibles facilement par l'ensemble des acteurs du processus de développement logiciel.

Analyse et conception. A la suite de la description informelle, un *ingénieur pédagogique* qui connaît 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 scénario sera réalisée en utilisant un langage de modélisation pédagogique. Au cours de cette phase, certaines activités pourront 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 fera également appel aux *fournisseurs de ressources pédagogiques* et aux *développeurs de composants*. Les premiers indiqueront les ressources disponibles ou à développer pour répondre aux besoins de l'enseignant et les deuxièmes pourront 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. Le rôle principal revient ici au développeur de composants. Il 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 dans les phases précédentes sont mis en place sur la plate-forme. Cette phase concerne l'assembleur de composants et *l'administrateur de plate-forme*. A la fin de cette phase, le module d'enseignement peut alors être référencé.

Test. Cette phase permet la vérification du comportement de la plate-forme, la cohérence des parcours et la bonne agrégation de tous les composants de la plate-forme. Ces tests seront réalisés par l'ingénieur pédagogique qui assurera de cette manière la validité d'un point de vue à la fois pédagogique et à la fois fonctionnel. Le développeur de composants et l'assembleur de composants contrôleront le bon fonctionnement de l'ensemble.

Evaluation. Au cours de l'utilisation des modules d'enseignement, il est possible d'évaluer le comportement des apprenants et des enseignants afin de vérifier que le parcours 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 afin de déterminer s'il existe des problèmes de surcharge des serveurs, afin d'y 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.

La phase de **constitution des parcours de formation** se trouve à la limite entre le processus de conception et celui d'utilisation. Dans cette phase, l'ingénieur pédagogique avec l'aide des enseignants de divers modules et du conseiller en formation, peut réaliser des parcours de formations, *i.e.* l'assemblage de différents modules pour constituer par exemple une année de formation complète. Cette création se décompose selon les mêmes phases que le processus de conception.

2.2 Processus d'utilisation

Le processus d'utilisation détermine les différentes phases relatives à l'exécution d'un module de formation.

Instanciation. La première phase consiste à déterminer les différentes personnes qui prendront part à la formation (nous utilisons ici le terme formation pour faire à la fois référence à un module d'enseignement ou à un parcours de formation) ainsi que leurs rôles (*apprenant, tuteur, référent*). 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. Nous sommes ici en présence de la phase visible de l'iceberg. C'est à partir de ce moment que la plate-forme va réellement être utilisée. L'exécution "normale" consistera en l'enchaînement des différents modules et de leurs activités respectives. Un tuteur assure le contrôle du bon déroulement d'un module. 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 (erreur du système d'exploitation, coupure réseau, défaillance matérielle). Les seconds sont les plus nombreux et les plus difficiles à résoudre. Le tuteur doit adapter le contenu du module, voire son organisation (comme par exemple redéfinir l'ordre des activités) afin de faire face au problème de compétence et de niveau des apprenants. Ce problème s'accentue lorsque les modifications ne s'adressent pas au groupe complet mais à une partie des membres. Le troisième type de problème relatif aux tracas administratifs (problème d'inscription, de paiement, *etc.*) pourront être réglés par le responsable administratif.

2.3 Rôles liés à une plate-forme de FOAD

Afin de préciser les différentes tâches, nous supposons qu'elles sont reliées à des rôles particuliers les réalisant. Notre division en rôles (figure 2) ne cherche pas à être exhaustive. Elle s'appuie sur notre expérience et nos recherches dans le domaine des EIAH (Environnements Informatiques pour l'Apprentissage Humain). Dans les sections suivantes de cet article, nous ferons souvent référence à ces rôles afin d'indiquer vers qui se sont portés nos efforts. Cette subdivision en rôles permet de cerner les besoins et les attentes des différentes personnes intervenants dans le cycle de vie d'une plate-forme d'enseignement. D’autres approches par rôle ont déjà été utilisées dans le cadre des plates-formes de formation (Paquette, 2001). Ce type de découpage est également utilisé dans le domaine du génie logiciel afin de déterminer les tâches à réaliser lors du processus de conception (Jacobson et al., 2000) et de développement d'applications (Matena et Stearns, 2001), (Schmidt-Wesche, 2003). Nous nous intéresserons ici aussi bien aux rôles des utilisateurs finaux qu'à ceux des concepteurs et développeurs de solutions pour la FOAD. Afin de mieux catégoriser les différents rôles, nous avons commencé par distinguer les différents domaines fonctionnels qui sont :

- les tâches administratives (gestion des inscriptions et des compétences, *etc.*);
- les tâches de modélisation des modules et des parcours de formation;
- les tâches techniques, c'est à dire le développement et la maintenance de la plate-forme;
- les tâches d'utilisation de la plate-forme.

Nous avons ainsi obtenu 11 rôles différents que nous avons introduits lors de la description des processus de conception et d'utilisation. Même si une personne peut tenir plusieurs rôles, la structure des plates-formes actuelles, notamment leur taille, comme par exemple (OpenUSS, 2006) empêche une même personne de tenir la totalité des rôles présentés.

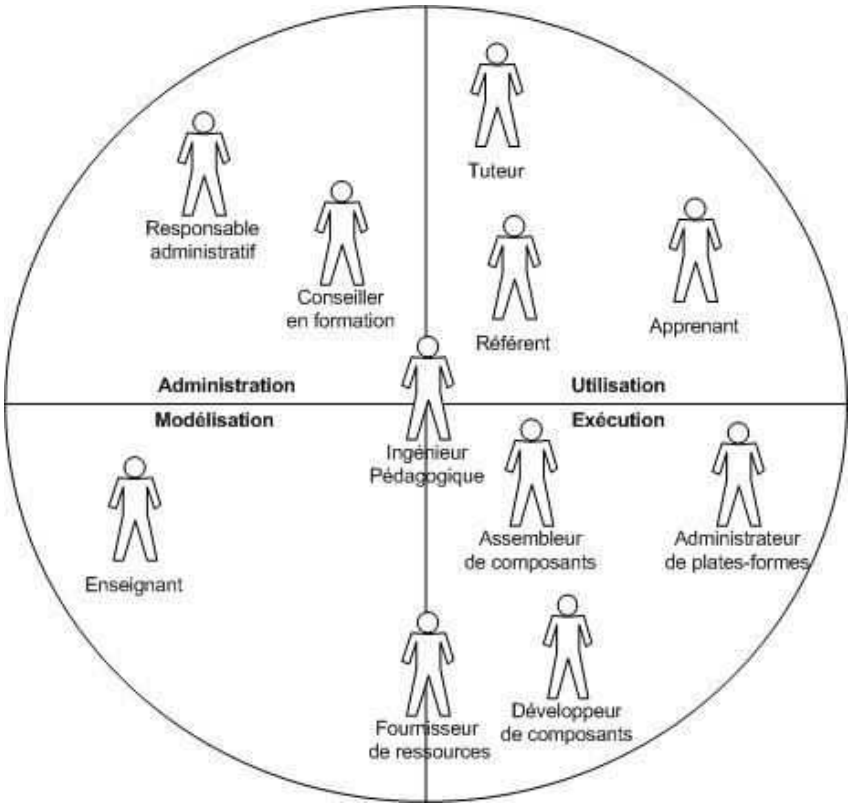

Figure 2 : Rôles et domaines fonctionnels liés à une plate-forme de FOAD

2.3.1 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 prérequis 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, *etc.*) 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, *etc.*) 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 scénarios et de la mise en œuvre des parcours. Il fait office de consultant auprès du conseiller en formation pour juger de l'aptitude d'un apprenant à suivre la formation.

2.3.2 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 (numériques ou non) pouvant être utilisées lors de l'exécution d'un module. Il représente une force de proposition lors de la formalisation des scénarios, 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.

2.3.3 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 scénarios 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 œuvre 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. Afin d'assurer la pérennité des modèles, l'ingénieur pédagogique désire une certaine indépendance de ces derniers vis-à-vis de la plate-forme d'exécution. En effet, la manière d'organiser un parcours de formation pour répondre à des objectifs pédagogiques est liée aux styles pédagogiques mis en œuvre et non aux contraintes technologiques de la plate-forme utilisée. Un même modèle peut être utilisé sur différentes plates-formes.

2.3.4 Apprenant

L'apprenant est le bénéficiaire principal d'une plate-forme de FOAD. Sans lui, la plate-forme n'a pas de raison d'exister. Il utilise le système pour acquérir de nouvelles compétences. Il peut travailler seul ou en groupe.

De nombreux services lui sont nécessaires. Il a besoin d'un catalogue des enseignements notamment en fonction de ses compétences acquises ou de celles qu'il souhaite acquérir. Il peut avoir besoin d'aide pour la création et la modification de son parcours. Cette aide peut provenir du tuteur, de l'ingénieur pédagogique ou du conseiller en formation. Afin de s'intégrer et d'échanger avec les autres utilisateurs de la plate-forme, il lui faut des outils de communications asynchrones (mail, forum) ou synchrones (messagerie instantanée) ainsi que des outils de travail en groupe (éditeur partagé, tableau blanc, *etc.*).

L'apprenant a besoin de connaître le travail à faire, déjà effectué et celui qui arrivera dans le futur. Cela lui permet d'avoir une vision globale du module/parcours et ainsi de mieux se situer dans le parcours global ainsi que dans les actions qu'il effectue.

D'un point de vue plus technique, il attend que la plate-forme offre un support de la mobilité (accès de n'importe quel endroit avec n'importe quel périphérique et à n'importe quel moment à la plate-forme). Il souhaite qu'on lui fournisse les outils adéquats et éventuellement qu'il puisse les personnaliser en fonction de ses besoins et de ses compétences.

2.3.5 Tuteur

Le tuteur a un rôle primordial pour le bon déroulement d'un module de formation dont il réalise l'animation. Il assure le suivi des différents étudiants en les guidant et en les aidant à atteindre les divers objectifs pédagogiques. Il informe le référent du déroulement et le prévient si un étudiant présente des difficultés particulières. Le tuteur peut, en collaboration avec l'ingénieur pédagogique et le référent, apporter des modifications temporaires (*i.e.* liées à une instance particulière du module) pour un étudiant ou pour tous les étudiants ainsi qu'une modification définitive avec l'accord de l'enseignant. En effet, le tuteur n'est pas obligatoirement un expert du domaine. Il peut ne disposer que des connaissances

suffisantes pour répondre aux attentes des étudiants dans un certain nombre de modules.

Afin de réaliser ces tâches, ce rôle nécessite des outils de communication classiques (mail, forum) pour échanger avec les apprenants. Pour faciliter le suivi, un ensemble d'outils de gestion de l'avancement ainsi que de modification du module doivent être disponibles.

2.3.6 Référent

Le référent gère un ou plusieurs groupes de formation. Il permet aux apprenants d'avoir un interlocuteur unique tout au long de leurs parcours de formation. Il peut quand cela devient nécessaire jouer le rôle de médiateur avec les différents tuteurs de modules. Il peut demander des modifications de parcours aux tuteurs.

Afin de remplir pleinement son rôle, le référent dispose d'outils semblables à ceux du tuteur comme les outils de communication et de suivi des apprenants. Il dispose en plus de systèmes permettant d'assurer le suivi sur l'ensemble du parcours des étudiants et notamment des outils de suivi de groupes. Ceci, afin d'être capable de déterminer rapidement si les étudiants avancent au même rythme ou s'il y a des écarts très importants entre les plus rapides et les plus lents, et ainsi prendre les bonnes décisions pour harmoniser le travail des apprenants.

2.3.7 Responsable administratif

Ses tâches principales sont la gestion des inscriptions ainsi que le suivi administratif (vérification de présence, ...) des différents participants. Il est le garant du bon déroulement des formations face aux différentes autorités (ministères, entreprises, ...). Pour réaliser ces différentes tâches il doit disposer d'outils de suivi des utilisateurs.

Le responsable administratif doit pouvoir vérifier qu'une personne inscrite dans la formation réalise bien le travail demandé afin, par exemple, de pouvoir fournir des certificats de scolarité et/ou des justificatifs pour les employeurs. La notion temporelle est également importante pour le monde industriel afin de valider le temps effectif passé par l'employé sur la plate-forme.

2.3.8 Conseiller en formation

Le conseiller en formation accueille les apprenants et les dirige vers les cursus en fonction de leurs compétences et de leurs souhaits. Il peut, avec l'aide de l'ingénieur pédagogique, créer de nouveaux parcours à partir des modules existants. Le conseiller en formation doit avoir accès aux différentes formations et modules disponibles. Cela implique pour la plate-forme la nécessité de disposer d'un service d'indexation.

2.3.9 Assembleur de composants

Le rôle d'assembleur de composants est un des trois rôles techniques. Son objectif principal est d'assembler les différents composants. Nous parlons ici de composants au sens large. Il peut s'agir de composants techniques ou de composants métiers. Une définition plus fine des rôles diviserait l'assemblage en deux sous-rôles, une partie dédiée aux aspects techniques et une partie dédiée aux ressources pédagogiques. L'assembleur de composants n'a pas de besoin spécifique vis-à-vis de la plate-forme d'enseignement à distance. Elle n'est pour lui qu'un domaine d'application parmi d'autres.

2.3.10 Développeur de composants

Il conçoit et développe des composants logiciels en fonction des besoins exprimés par l'ingénieur pédagogique, le fournisseur de ressources et l'assembleur de composants. Il crée de nouvelles fonctionnalités pour la plate-forme. Il définit et réalise les composants "techniques". Il intervient principalement dans les phases de conception et d'implémentation. Le développeur n'a pas de besoin

spécifique vis-à-vis de la plate-forme d'enseignement à distance. Elle n'est pour lui qu'un domaine d'application parmi d'autres.

2.3.11 Administrateur de la plate-forme

Le rôle d'administrateur de la plate-forme intervient principalement au cours de l'utilisation du système par les différents utilisateurs. Il peut également intervenir lors de la phase de test afin de valider le comportement de la plate-forme. Il assure le bon fonctionnement de la plate-forme d'un point de vue technique. Afin de réaliser ses différentes tâches, il doit disposer d'outils d'administration et de surveillance aussi bien de la plate-forme elle-même que du système logiciel et matériel sur lesquels elle s'appuie (système d'exploitation, réseau, matériel, ...).

Il possède une vision globale du système. Il peut intervenir pour régler des problèmes liés à l'accès aux différents services, il réalise les configurations par défaut des environnements, il a un pouvoir d'administration des différents comptes des utilisateurs (création, destruction, gestion des droits d'accès) en accord avec le responsable administratif (apprenants à jour de leurs inscriptions, mise à jour de la liste des tuteurs, gestion des enseignants), l'ingénieur pédagogique et le référent (pour la constitution des groupes par exemple). Il assure également les différentes sauvegardes et la remise en état après incident.

3 Plate-forme de support à l'exécution de scénarios pédagogiques

Lorsque l'on arrive au déploiement d'un scénario pédagogique sur une plate-forme spécifique, la plupart du temps cela revient à organiser l'accès aux ressources pédagogiques au sein de la plate-forme et ainsi la majeure partie du scénario est perdue. Par exemple, si un ordre a été défini pour accéder aux ressources, ceci ne peut pas toujours être imposé par la plate-forme. C'est notamment le cas avec la plate-forme *OpenUSS* (Grob et al., 2004), (OpenUSS, 2006) où il est possible d'organiser l'accès au contenu pédagogique par semestre et par cours mais nous devons fournir le contenu de manière manuelle si nous voulons un meilleur contrôle sur l'ordre des accès. Cela est principalement dû au fait que la focalisation principale de ces plates-formes est dirigée vers l'accès aux ressources plutôt que la scénarisation. Dans cette partie, nous allons décrire les propriétés nécessaires pour supporter complètement les scénarios pédagogiques et indiquer quels sont les liens avec les rôles impliqués dans les opérations liées à une plate-forme d'enseignement.

3.1 Modèles pour les scénarios pédagogiques

Pour profiter des avantages de l'utilisation des plates-formes pour supporter des scénarios pédagogiques, il doit y avoir obligatoirement une description formelle qui peut être manipulée par la plate-forme. Les langages de modélisation pédagogique (*Educational Modelling Languages*) comme par exemple IMS-LD (IMS, 2003) peuvent être de bons candidats pour cela. Ces langages formels seront utilisés par l'ingénieur pédagogique pour transcrire les idées des enseignants et des conseillers d'orientation en prenant en compte les contraintes des responsables administratifs. Pour cela, un langage de modélisation générique devrait supporter les éléments suivants :

• **L'ordonnancement d'activités individuelles et de groupe**. Les apprenants peuvent s'engager dans un module ou un chemin d'apprentissage de manière individuelle ou à l'intérieur d'un groupe. Dans le dernier cas, il est raisonnable de tenir compte de la progression individuelle entre les activités de sorte que chacun puisse progresser à son rythme sur certaines portions du module. La synchronisation du groupe pourra être imposée pour des activités collaboratives ou pour respecter des contraintes de temps et ainsi éviter un trop grand écart entre les apprenants. La définition des différentes portions individuelles est réalisée par l'enseignant et l'ingénieur pédagogique. Pour assurer un

ordonnancement dynamique, ils pourront également ajouter des conditions sur les différentes branches du parcours.

• **La gestion du temps**. Il est important de pouvoir imposer des contraintes de temps sur une unité d'étude ou un chemin d'apprentissage. Ces contraintes peuvent provenir premièrement de raisons pédagogiques (*e.g.* pour limiter la durée d'un examen en ligne) ou elles peuvent être édictées par des raisons administratives (*e.g.* pour respecter les semestres académiques).

• **Le support de la collaboration**. L'apprentissage collaboratif est un moyen d'améliorer l'expérience d'apprentissage et une manière de soutenir la motivation des apprenants (Eales et al., 2002). Il devra donc exister des moyens de spécifier la collaboration.

3.2 Support à l'exécution des modèles

Les supports à l'exécution des modèles sont étroitement reliés au pouvoir d'expression des langages utilisés. De manière générale, la plate-forme est responsable de l'automatisation de la gestion de l'avancement des apprenants d'activité en activité à l'intérieur d'un module d'apprentissage en fournissant les bons outils et les bonnes ressources pédagogiques au bon moment. Ces possibilités sont d'une importance primordiale pour soulager les tuteurs des tâches fastidieuses et ainsi leur permettre de suivre plus finement les apprenants notamment dans le cas de grands groupes d'apprenants.

3.3 Suivi de l'exécution du modèle

L'automatisation du processus d'apprentissage est d'une grande aide pour l'éducation à distance. Cependant, elle doit être accompagnée de services d'observations afin d'aider les tuteurs et les référents pour le suivi des apprenants. La plate-forme doit fournir des moyens d'augmenter le retour d'informations sur les problèmes éventuels que peuvent rencontrer les apprenants (*e.g.* une progression lente au sein d'un groupe, la violation d'une contrainte temporelle, *etc.*).

3.4 Amélioration de la gestion des modèles

Ayant identifié les problèmes rencontrés par un étudiant ou un groupe grâce au service de surveillance, le tuteur et l'ingénieur pédagogique doivent être capable de modifier le modèle en cours d'exécution afin d'atteindre les objectifs pédagogiques de départ. De plus, si le modèle rectifié présente de manière globale des avantages par rapport à la version initiale, il doit être disponible pour les futures exécutions. De cette façon, cela permet une amélioration en continu des scénarios pédagogiques. Il est également plus facile de créer de nouveaux modules d'enseignement en s'appuyant sur des activités et des scénarios existants.

4 Construction d'un support flexible pour les scénarios pédagogiques

Dans notre travail de recherche sur le support à l'exécution de scénarios pédagogiques, nous avons décidé de ne pas nous concentrer sur une plate-forme spécifique mais de développer plutôt un service dédié qui pourra être inclus dans les systèmes de gestion d'apprentissage existant. Ceci a soulevé quelques contraintes techniques que nous avons gardées à l'esprit tout au long du développement de notre composant :

• **Basé sur des standards** : Comme nous concevons un composant technique qui sera inclus dans différentes plates-formes, la première condition à laquelle nous devons veiller consiste à respecter les différents standards existants ;

• **Support d'intégration** : L'intégration à l'intérieur d'une plate-forme doit être la plus simple, la moins contraignante et la plus interopérable possible. Pour cela, une approche

basée sur les standards des services web et sur une communication événementielle semble une bonne solution ;

• **Persistance** : Les scénarios pédagogiques correspondent le plus souvent à des processus dont le temps d'exécution est long. Pour cela, nous devons prendre en compte les problèmes liés à la persistance et l'intégrité des données manipulées.

Pour résoudre les deux derniers points, nous avons choisi de faire reposer notre architecture sur les standards *J2EE* (*Java 2 Enterprise Edition*) (J2EE, 2006). Le service de support à l'exécution des scénarios pédagogiques a été construit en utilisant des composants logiciels de type *Enterprise JavaBeans* (Matena et Stearns, 2001) qui gèrent nativement les questions relatives à la persistance des données. Ces composants logiciels sont accessibles via RMI/IIOP ou les services web, nous permettant ainsi une intégration dans des environnements variés supportant les protocoles RMI, CORBA ou SOAP.

Concernant notre première contrainte portant sur les standards, nous nous sommes dirigés vers une solution générale plutôt que d'employer un langage pédagogique spécifique. L'exécution d'un scénario pédagogique peut être comparée à l'exécution d'un processus métier traditionnel (*e.g.* gestion et suivi de documents administratifs). La solution naturelle pour répondre aux besoins d'exécution de processus réside dans les moteurs de workflows. Cependant, les lacunes des systèmes de workflows classiques comme le manque de flexibilité dans l'exécution des modèles ou la mauvaise gestion des exceptions les empêchent de répondre complètement à tous les critères que nous avons présentés précédemment. Pour ces raisons, nous avons entrepris la conception et le développement de notre moteur de workflows tout en respectant les standards existants dans le domaine, à savoir, ceux du Workflow Management Coalition et ceux de l'Object Management Group. Dans la section suivante nous allons décrire ces standards avant de présenter de quelle manière nous avons répondu aux différents critères précédents.

4.1 Les standards des systèmes de workflows

4.1.1 Le modèle de référence workflow et le XML Process Definition Language

Le modèle de référence workflow (WfMC, 1995) est un standard provenant du consortium WfMC (Workflow Management Coalition) (WfMC, 2006). Le but de cette organisation consiste à promouvoir les systèmes de workflows notamment à travers la standardisation et l'interopérabilité. Le modèle de référence ne définit pas le moteur de workflows lui-même mais ses 5 interfaces de connexion avec le monde extérieur. Dans cet article, nous sommes principalement intéressés par l'interface numéro 1 qui définit un langage XML, nommé *XPDL* (*XML Process Definition Language*) (WfMC, 2002), pour exprimer des modèles de processus de manière indépendante des plates-formes d'exécution. Nous nous sommes également intéressés à l'interface numéro 5 car elle définit la manière dont les outils de surveillance et d'administration vont interagir avec le moteur de workflows.

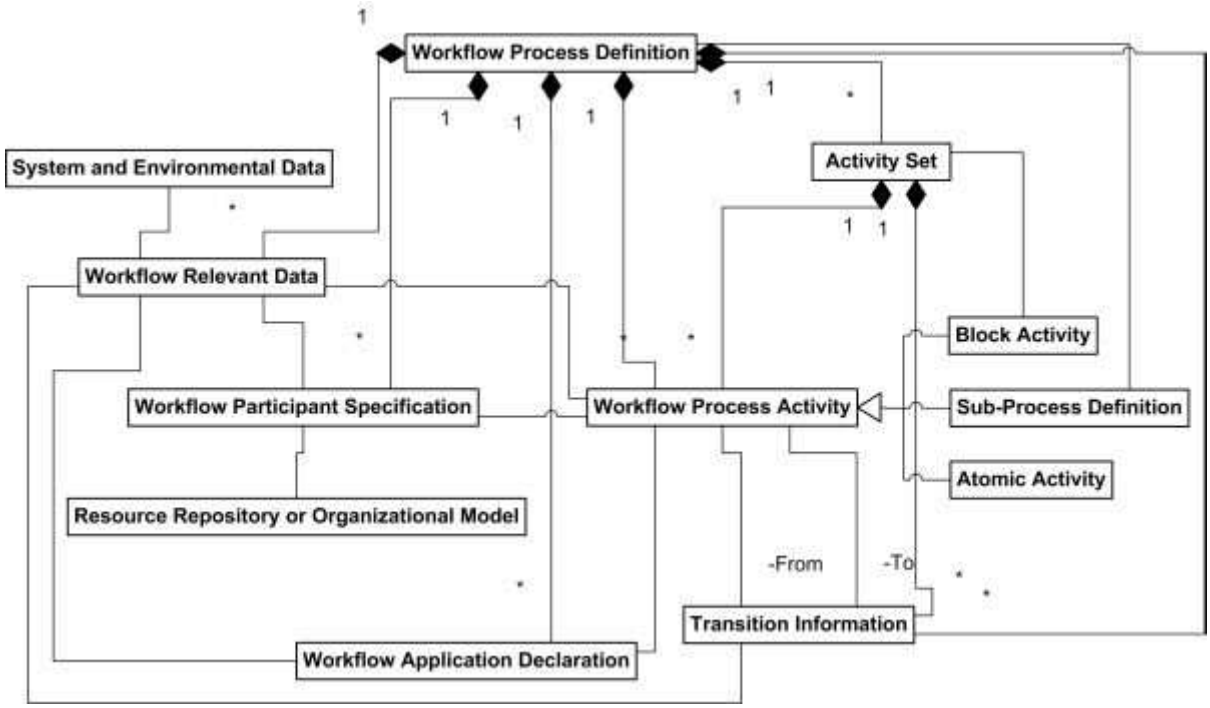

Figure 3 : Méta-modèle de XPDL tiré de (WfMC, 2002)

La figure 3 montre le méta-modèle du langage XPDL avec lequel nous pouvons définir un scénario pédagogique qui s'exécutera dans un moteur de workflows. Il est composé de 6 entités principales qui sont *Process*, *Activity*, *Transition*, *Participant*, *Application* et *Relevant Data*.

• **Workflow Process Definition**. L'entité de définition de processus contient des informations contextuelles qui s'appliquent aux autres entités du processus. Il définit l'objectif global du travail à réaliser. Il fournit des informations d'ordre administratif (date de création, auteur, *etc.*) ou utilisées au cours de l'exécution (paramètres d'initialisation, limites temporelles, *etc.*). Il se compose d'une ou plusieurs activités.

• **Workflow Process Activity**. L'entité activité définit le travail qui sera réalisé par un ensemble de ressources (participants, applications informatiques, *etc.*). Certaines informations peuvent être associées comme par exemple la priorité, le type de démarrage et d'arrêt (automatique ou manuel), *etc*. Il existe trois catégories d'activités. Une activité peut être un sous-processus (*Sub-process definition*), *i.e.* elle est un conteneur pour l'exécution d'un processus spécifié séparément. Ce processus peut s'exécuter localement ou à distance. Une activité peut également être une activité de bloc (*bloc activity*), *i.e.* l'exécution d'un regroupement d'activités et de transitions (*activity set*). Enfin, une activité peut être atomique (*atomic activity*). Il peut s'agir soit d'une activité de routage qui ne réalise aucun travail mais qui est utilisée pour synchroniser un ensemble de transitions, soit d'une activité "normale". Dans ce cas lors de l'exécution, cette activité est transformée en bons de travail (*work item*). Les activités sont reliées entre elles via des transitions.

• **Transition Information**. Ces entités représentent le flot de contrôle, *i.e.* l'enchaînement des différentes activités. Chaque transition possède trois caractéristiques principales, l'identifiant de l'activité initiale (*from-activity*), l'identifiant de l'activité d'arrivée (*to-activity*) et les conditions de passage d'une activité à l'autre.

• **Workflow Participant Specification**. Cette entité fournit la description des ressources qui vont réaliser les activités. Il existe plusieurs types de participants (*human*, *role*, *system*,

organizational unit).

• **Workflow Application Declaration**. Cette entité fournit des informations sur les applications informatiques qui peuvent être invoquées par le moteur de workflows afin de réaliser tout ou partie du traitement associé à une activité.

• **Workflow Relevant Data**. Cette entité représente les données créées et utilisées dans chaque instance de processus. Ces données sont disponibles pour les différentes activités, transitions (pour l'évaluation de conditions par exemple) ou applications (paramètres d'entrée et de sortie).

Nous avons utilisé le langage XPDL ainsi que les possibilités qu'il offre pour son extension. Nous avons en effet ajouté des éléments pour notamment supporter la définition d'activités de groupe et les contraintes temporelles. Nous avons également séparé les modèles de processus et d'activité afin d'augmenter la réutilisation en offrant la possibilité de réemployer les modèles d'activités pour la construction de nouveaux modèles de processus.

4.1.2 Le Workflow Management Facility

L'OMG (*Object Management Group)* (OMG, 2006) s'est basé sur le travail du WfMC et son modèle de référence pour proposer le WMF *(Workflow Management Facility*) (OMG, 2000), un standard pour représenter, avec une conception orientée objet, les différents constituants d'un moteur de workflows. Ce standard propose les interfaces d'objets telles que *WfProcess*, *WfActivity*, *WfResource*, qui correspondent aux descriptions des éléments XPDL dans les modèles de processus. Il existe également une hiérarchie d'interfaces dédiées à la gestion des différents événements qui se produisent au sein du moteur, cela nous aidera pour la construction d'outils de suivi. Nous avons réalisé une implémentation du WMF avec des extensions pour supporter les activités collaboratives, la gestion des groupes et pour améliorer la réutilisation des modèles de processus et d'activités.

5 Cooperative Open Workflow

Dans cette section, nous allons décrire le moteur de workflows, nommé *Cooperative Open Workflow* (Vantroys et Peter, 2003) que nous avons développé en nous basant sur les standards présentés précédemment. Nous allons dans un premier temps indiquer de quelle manière nous pouvons modéliser un scénario pédagogique simple. Nous nous concentrons ensuite sur les activités collaboratives et sur la gestion des contraintes temporelles. Nous verrons de quelle manière le modèle est géré par le moteur de workflows lors de l'exécution. Finalement, nous donnerons quelques détails sur l'implémentation et la gestion des modèles.

5.1 Modélisation d'une unité d'apprentissage

La principale fonction du système de workflows est de gérer l'ordonnancement des différentes activités qui composent un module de formation. Un tel module est suivi par un groupe d'apprenants (un apprenant individuel étant un cas particulier d'un "groupe" réduit à un seul apprenant). Dans la suite, nous prendrons un enseignement en physique comme exemple pour modéliser un cours pour illustrer la gestion des modèles et des instances par le moteur de workflows ainsi que la gestion d'activités individuelles et de groupe. Le scénario correspondant au cours est composé de quatre activités :

• Une *activité d'apprentissage de documents de cours* associée au rôle *apprenant* qui donnera un accès à un ensemble d'objets pédagogiques relatifs au sujet du cours comme par l'exemple l'étude des composants électroniques. Nous pouvons indiquer une *date limite* pour réaliser cette activité afin que les apprenants ne restent pas éternellement à l'étude des ressources.

• Une *activité d'autoévaluation* associée au rôle *apprenant*. Ce dernier devra répondre à un questionnaire à choix multiples afin de valider ou non les connaissances acquises lors de la phase précédente. Comme il s'agit d'une évaluation, cette activité devra être réalisée avec une *durée limitée*.

• Une *activité de correction* associée au rôle *tuteur* qui vérifiera les réponses de l'étudiant et proposera éventuellement une modification du parcours pour ajouter des activités permettant à l'apprenant de combler les parties qu'il ne maîtrise pas.

• Une *activité de discussion* associée aux rôles *apprenant* et *tuteur*. Cette activité a pour objectif de faire le point avec les étudiants et de répondre à leurs questions. Elle permet également de préparer le module d'enseignement suivant qui consiste en la réalisation d'un TP à l'aide d'un environnement virtuel de collaboration. Une date limite de début sera positionnée afin de synchroniser les différents participants de l'activité de discussion.

Comme certaines parties du module peuvent être réalisées par les étudiants à leurs propres rythmes, nous devons prendre en compte cet aspect afin d'offrir une grande flexibilité dans l'ordonnancement des différentes activités. Il y a deux moyens de gérer l'enchaînement des activités pour un groupe d'apprenants :

• Dans le premier mode, une activité se termine uniquement lorsque tous les apprenants l'ont terminée. De cette manière, toutes les activités du groupe sont synchronisées. Bien que ce mode de fonctionnement soit très proche de celui en face-à-face traditionnel, il ne tire pas profit des avantages offerts par l'apprentissage à distance.

• Le second mode consiste à identifier les parties du module qui peuvent être réalisées de manière autonome. De cette manière, les étudiants pourront progresser à leurs propres rythmes à l'intérieur d'un groupe. Certaines activités seront utilisées pour offrir des points de synchronisation au groupe.

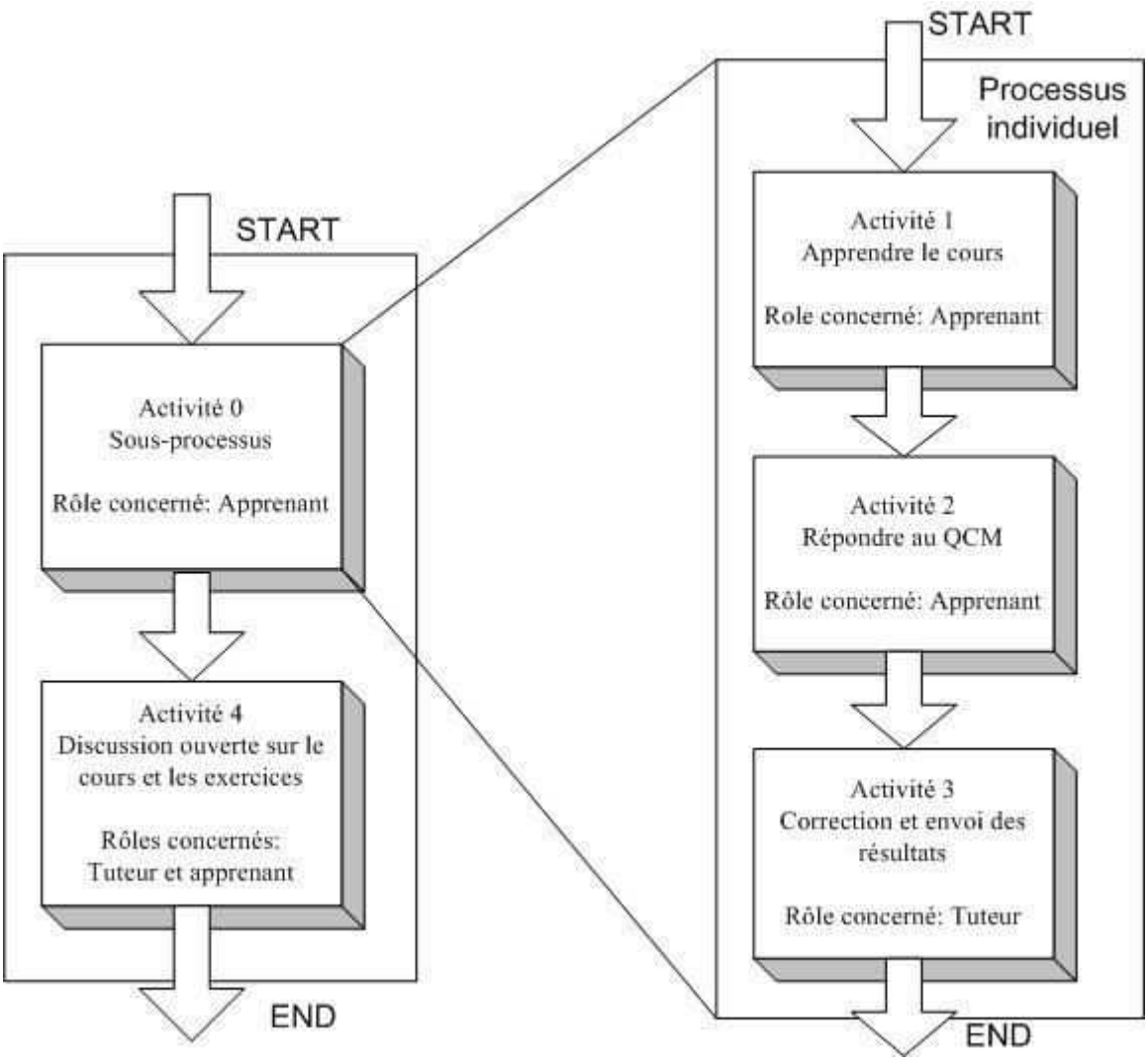

Figure 4 : Exemple de scénario pédagogique

Ces deux modes sont supportés dans COW par l'utilisation de sous-processus. Dans notre scénario, l'enseignant décide que les trois premières activités peuvent être réalisées de manière individuelle par chaque apprenant. Ces activités sont ainsi modélisées par l'intermédiaire d'un processus. Le processus global du cours est composé d'une séquence de deux activités (partie gauche de la figure 4), la première fait référence au modèle de processus individuel (partie droite de la figure 4) et la seconde correspond à une discussion synchrone entre les différents membres du groupe.

La figure 5 montre une partie du modèle global présenté précédemment, en utilisant le format XML dérivé de XPDL utilisé par le moteur de workflows. Nous avons notamment séparé les modèles de processus et les modèles d'activités. De cette manière, nous pouvons facilement changer et remplacer une activité par une autre ou modifier une activité du processus sans remettre en cause l'ordonnancement.

```
<WorkflowProcess Id="physics"
                 name="physique">
  ...
  <Activities>
    <ActivityLink Id="START" URL="start" />
    <ActivityLink Id="A1"    URL="lecture_et_auto_evaluation"/>
    <ActivityLink Id="A2"    URL="discussion"/>
    <ActivityLink Id="END"   URL="end"/>
  </Activities>
  <Transitions>
    <Transition Id="T0" From="START" To="A1" />
    <Transition Id="T1" From="A1" To="A2" />
    <Transition Id="T2" From="A2" To="END" />
</WorkflowProcess>
```

Figure 5 : Modèle XML global du cours

Nous pouvons également construire de nouveaux processus en assemblant des modèles d'activités existants. Un modèle de processus, tel que visible à la figure 5, définit les identifiants des modèles d'activités utilisés et les transitions qui les ordonnent. La figure 6 montre un modèle d'activité de discussion référencé dans le processus global.

Les liens d'ordonnancement entre activités sont réalisés par les transitions. Les transitions peuvent contenir des conditions permettant de réaliser des scénarios "dynamiques", *i.e.* liés à l'évaluation des conditions. Le scénario pédagogique précédent pourrait par exemple être amélioré en comportant des chemins différents en fonction des résultats obtenus à l'évaluation, *e.g.* si la note est inférieure à 10 alors retourner à la première activité. COW permet d'associer ce type de condition aux transitions. Il peut y avoir plusieurs transitions sortantes et/ou entrantes pour une activité donnée. Chaque activité peut contenir des pré- et des post- conditions précisant la manière de gérer les différentes transitions entrantes et sortantes.

```
<Activity Id="chat" Name="discussion" />
...
  <Implementation>
    <Tool Id="outil_discussion_synchrone" />
  </Implementation>
  <Group><YES/></Group>
  <Performer>
    <Participant Id="R1" Name="Apprenant" ParticipantType="ROLE" />
    <Participant Id="R2" Name="tuteur"    ParticipantType="ROLE" />
  </Performer>
  ...
</Activity>
```

Figure 6 : Modèle XML d'une activité collaborative

5.2 Activités collaboratives

Afin de supporter des activités collaboratives, nous avons été amenés à réaliser quelques ajouts dans le langage XPDL ainsi que quelques modifications dans le WMF afin d'introduire de manière explicite la notion de *tâche* (*workitem*). Une tâche représente un élément atomique de travail à réaliser, une activité se compose de tâches et définit un contexte d'exécution pour les tâches qu'elle contient. Dans le cas le plus simple, il n'y a qu'une tâche par activité. Cependant, à l'intérieur d'une activité collaborative, on trouvera plusieurs tâches. Les tâches sont attribuées à des rôles. Ainsi, si plusieurs acteurs disposent du même rôle, le système créera une instance de tâche pour chaque acteur dans l'activité concernée. Les ressources sont

allouées aux tâches plutôt qu'à l'activité.

Dans notre exemple (figure 6), l'activité dédiée à la discussion pourra être réalisée par l'intermédiaire d'un outil de discussion synchrone. Au cours de cette activité, chaque personne ayant le rôle d'apprenant ou de tuteur possédera une tâche correspondant au travail à réaliser. Ici, tous les participants ont la même tâche, donc aucune tâche spécifique n'a été définie. Cependant, les apprenants et le tuteur n'auront pas les mêmes droits sur l'outil car ils ont des rôles différents. Il n'y a qu'un seul outil déclaré pour cette activité et nous pouvons remarquer qu'il n'est représenté que par un identifiant. En effet, afin d'augmenter la réutilisation des modèles, les outils réellement utilisés sont définis dans des *patrons d'instances* propres à chaque nouvelle instance d'activité ou de processus. Nous réalisons ainsi une liaison tardive entre les outils véritables et ceux spécifiés dans les modèles, nous permettant de remplacer au dernier moment un outil par un autre, augmentant ainsi la flexibilité lors de l'exécution.

5.3 Contraintes temporelles

La gestion du temps est un élément important de la modélisation et de l'exécution d'un processus de Workflow. Dans le cas de l'éducation à distance, la prise en compte du temps permet de limiter l'accès aux ressources et aux tâches et évite dans le cadre d'une formation de groupe, une dérive trop importante entre les apprenants les plus rapides et les plus lents. L'avancement du groupe est ainsi plus homogène. Cette connaissance des contraintes temporelles permet également au tuteur et au référent de vérifier la charge de travail de l'apprenant et évite les excès (trop ou trop peu de travail). Chaque élément du modèle représentant un travail (processus, activité, tâche) dispose de deux notions temporelles. La première, appelée durée, représente le temps minimal ou maximal pendant lequel l'élément devra se trouver dans l'état "en cours d'exécution". Le temps minimal permet de ralentir les apprenants qui auraient tendance à passer trop vite à la suite du processus. Le temps maximal, au contraire, évite que l'apprenant ne s'attarde trop. La deuxième notion, appelée date limite, spécifie les périodes durant lesquelles le travail doit se situer.

Un exemple de modélisation de dates limites est visible à la figure 7. Dans le cas présenté, la date limite est positionnée 4 jours après le début de l'activité en n'incluant pas les week-ends. En cas de violation de cette date, la stratégie employée consiste à envoyer un courriel à l'apprenant pour l'avertir que l'activité est terminée et qu'il doit passer à la suivante.

```
<Deadline>
  <EndDeadLine>
    <AdjustedTime>
      <Date day="4"/>
      <strategy id=WithoutWeekEnd/>
    </AdjustedTime>
    <Strategy id="sendMail">
  </EndDeadLine>
</Deadline>
```

Figure 7 : Modèle XML des contraintes de dates limites

La seconde contrainte temporelle est définie dans l'activité d'autoévaluation, la durée maximale sera de 3 heures comme illustré à la figure 8 et lorsque cette durée sera atteinte la stratégie sera de terminer automatiquement l'activité.

```
<Limit>
  <MaxTime>
    <AdjustedTime>
      <Date hour="3"/>
    </AdjustedTime>
    <Strategy id=automaticCompletion/>
  </MaxTime>
</Limit>
```

Figure 8 : Modèle XML des contraintes de durée

Ces contraintes temporelles peuvent être relâchées dans le cadre d'un enseignement de type formation tout au long de la vie, où l'apprenant, qui travaille généralement seul, peut avancer complètement à son rythme. Il est possible de modifier, retirer et ajouter des contraintes dynamiquement en utilisant les mécanismes de flexibilités mis en œuvre par COW.

5.4 Exécution du modèle

La création d'une instance de processus requiert un patron d'instance qui décrit les liens entre les rôles et les véritables acteurs ainsi que les liens entre les identifiants des ressources, les objets pédagogiques et les outils. La projection peut être globale pour le patron ou être définie activité par activité. La séparation entre le modèle de processus et les données d'instance autorise une meilleure réutilisation des modèles. La figure 9 représente les différentes opérations du moteur de workflows pour l'exécution de notre scénario avec un groupe de trois apprenants et un tuteur.

A partir des modèles de processus et d'activités ainsi que du patron d'instance, le moteur va commencer par créer un sous-processus pour chaque apprenant. Ces sous-processus contiennent chacun les trois activités qui constituent le processus individuel (apprendre le cours, réaliser l'exercice et correction) et chaque activité ne contient qu'une seule tâche. La troisième activité est réalisée par la personne associée au rôle de tuteur, il aura donc à réaliser trois tâches assignées par des processus différents. Lorsque tous les sous-processus seront terminés, le moteur créera une activité collaborative comportant une tâche pour chaque apprenant et une tâche pour le tuteur.

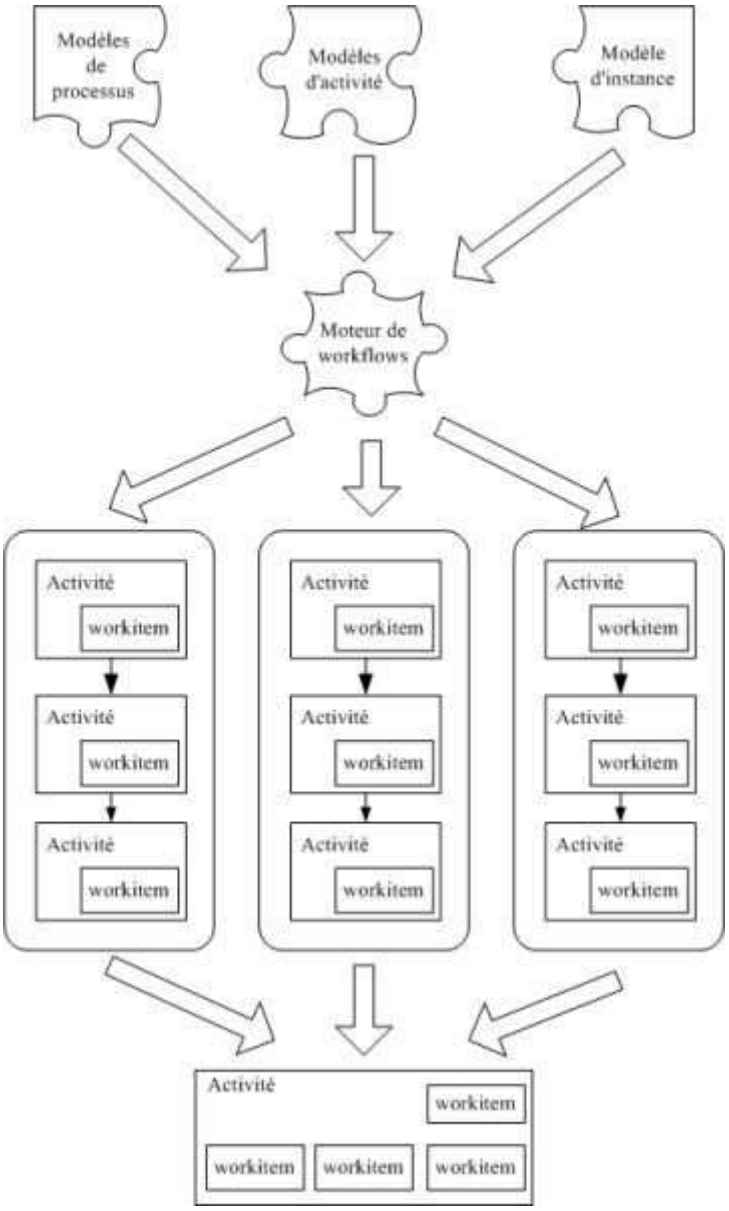

Figure 9 : Instanciation du modèle de scénario pédagogique

5.5 Implémentation du moteur de workflows

Suite à la présentation du fonctionnement de notre service, nous allons maintenant détailler dans cette section l'implémentation de notre service d'exécution de scénarios pédagogiques.

5.5.1 Architecture globale

Afin d'atteindre nos objectifs d'extensibilité et de réutilisation, nous avons choisi de modulariser au maximum la réalisation de COW. Cette conception nous offre de plus une plus grande facilité de maintenance et d'évolution des fonctionnalités. La figure 10a représente l'architecture actuelle de notre moteur.

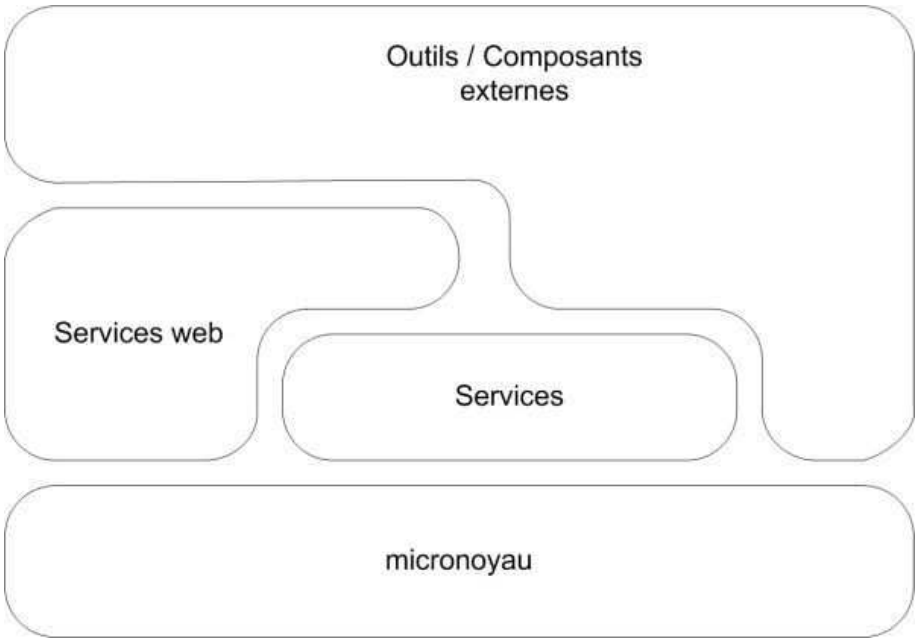

Figure 10a : Architecture globale de COW

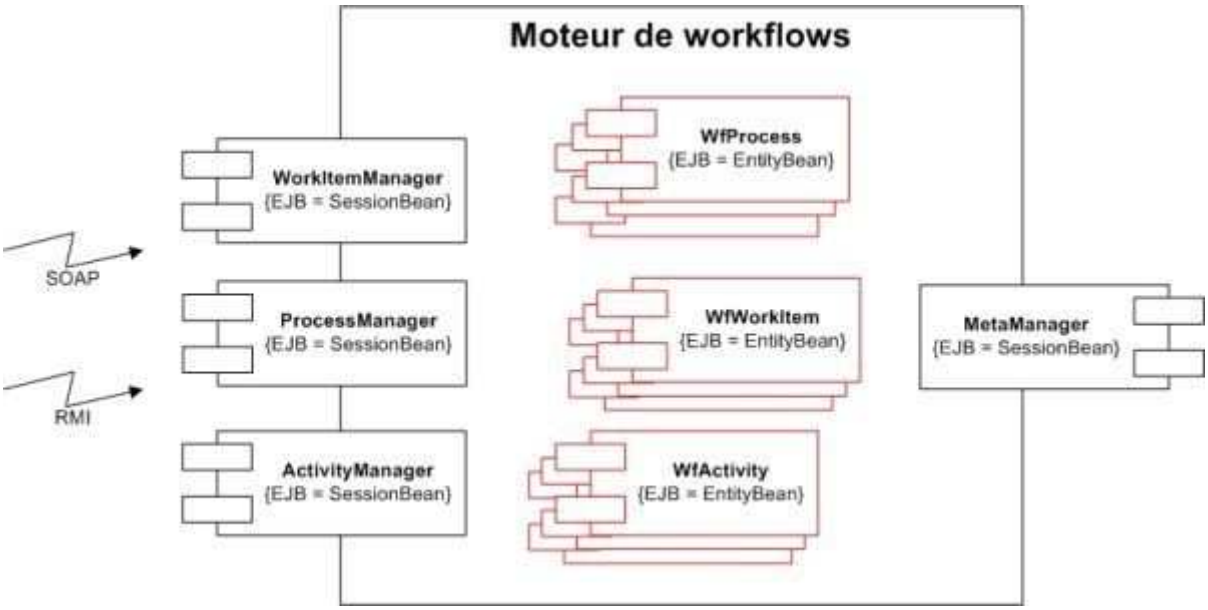

Figure 10b : Constitution du moteur de workflows

L'architecture est décomposée en quatre domaines distincts :

• Le **micronoyau** représente le cœur de notre système. Nous avons suivi une démarche proche de celle réalisée dans le domaine des systèmes d'exploitation à micronoyau (Tanenbaum, 1994). La philosophie de ce type d'architecture est d'offrir un ensemble minimal de fonctions spécifiques. Ce minimalisme rend cette architecture très souple car il implique peu de contraintes pour ajouter des fonctionnalités dédiées de plus haut niveau. Dans le cadre d'un système de workflows les fonctionnalités de base concernent la gestion rudimentaire des processus et des activités, la circulation des documents et l'envoi/réception d'événements permettant au monde extérieur de connaître les opérations en cours du système. Cette approche minimaliste a également été employée pour la conception de *micro-workflow* (Manolescu, 2000). Dans notre cas, cela correspond à

l'implémentation de la spécification WMF (figure 10b).

• Les **services** offrent un niveau d'abstraction supplémentaire par rapport au noyau en offrant des fonctionnalités de plus haut niveau avec une valeur ajoutée pouvant varier. Certains services n'offrent qu'une vue unifiée et simplifiée du micronoyau (figure 10b). Ils sont une mise en œuvre du patron de conception *façade* (Gamma et al., 1995). D'autres plus complets permettent par exemple de gérer la liste des travaux d'un utilisateur ou l'état d'avancement d'un étudiant dans un processus en cours d'exécution.

• Les **services web** offrent une couche d'abstraction et de transformation permettant l'intégration du micronoyau et des services dans un environnement hétérogène. Ils réalisent notamment des conversions de types comme par exemple la transformation d'objets Java en une représentation XML. Cela nous permet de n'utiliser que des types de données simples pour la communication SOAP. Les types de données complexes sont beaucoup plus difficiles à faire circuler car ils nécessitent souvent une sérialisation puis une désérialisation qui ne sont pas toujours réalisables de manière simple entre deux mondes technologiques différents comme par exemple pour passer du monde Java au monde Visual Basic. Une partie de ces services est basée sur le patron de conception *adaptateur*.

• Les **outils et composants externes** accèdent et manipulent les différents services offerts par la plate-forme. Dans ce niveau nous allons retrouver les outils de surveillance, de manipulation des modèles et de leurs instances, ainsi que toute la gestion des interactions des utilisateurs avec notre système.

5.5.2 Gestion du modèle et de la flexibilité

Comme nous l'avons indiqué dans la section 5.1, les modèles des scénarios pédagogiques sont décrits dans un langage XML dérivé de XPDL et nommé COWL pour *COW Language*. Pour manipuler ces modèles, nous avons développé un outil de modélisation et de gestion, nommé COW-GIRL (*COW-Graphical Interface for ReaL user*) (figure 11). Bien que l'interface ne soit pas encore adaptée complètement à un ingénieur pédagogique, elle permet de créer graphiquement un nouveau scénario qui sera sauvegardé automatiquement dans COW au format COWL. De même, il est possible de voir graphiquement l'état d'avancement de toutes les instances de scénarios qui ont créées.

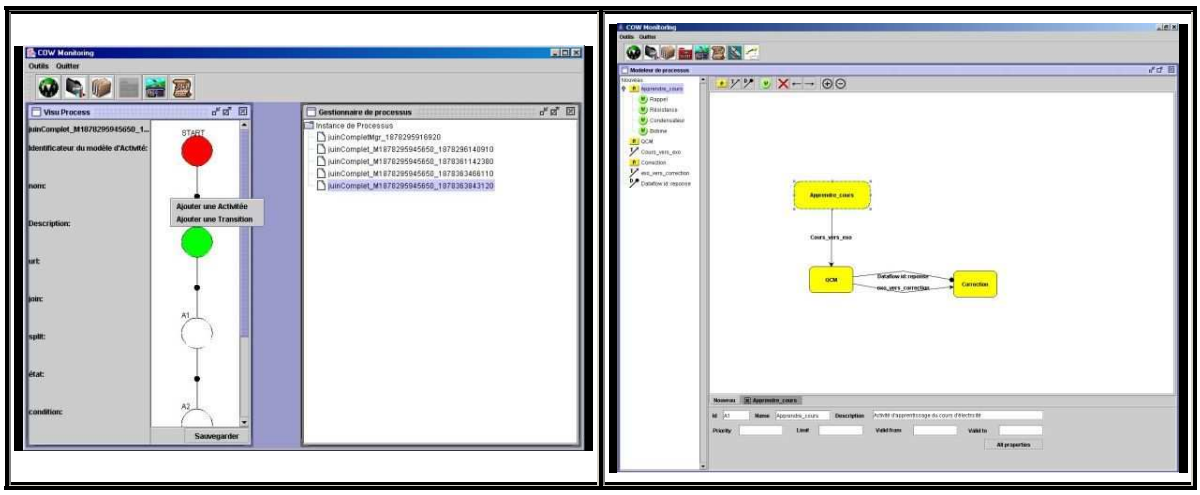

Figure 11 : Outils de modélisation et de gestion de COW

Lors de l'instanciation d'un scénario, COW crée à partir des fichiers XML un ensemble d'objets java correspondant aux différents éléments du méta-modèle (processus, activité, transition, ...) (figure 12). Lorsque les processus et les activités passeront dans l'état "en cours d'exécution", des composants EJB entités seront alors instanciés. Ils permettent la sauvegarde des éléments persistants (date de démarrage,

état, ...). Lorsqu'un utilisateur accède à COW via son API, c'est le niveau des EJB qui est appelé. L'instanciation tardive des EJB facilite la flexibilité du modèle.

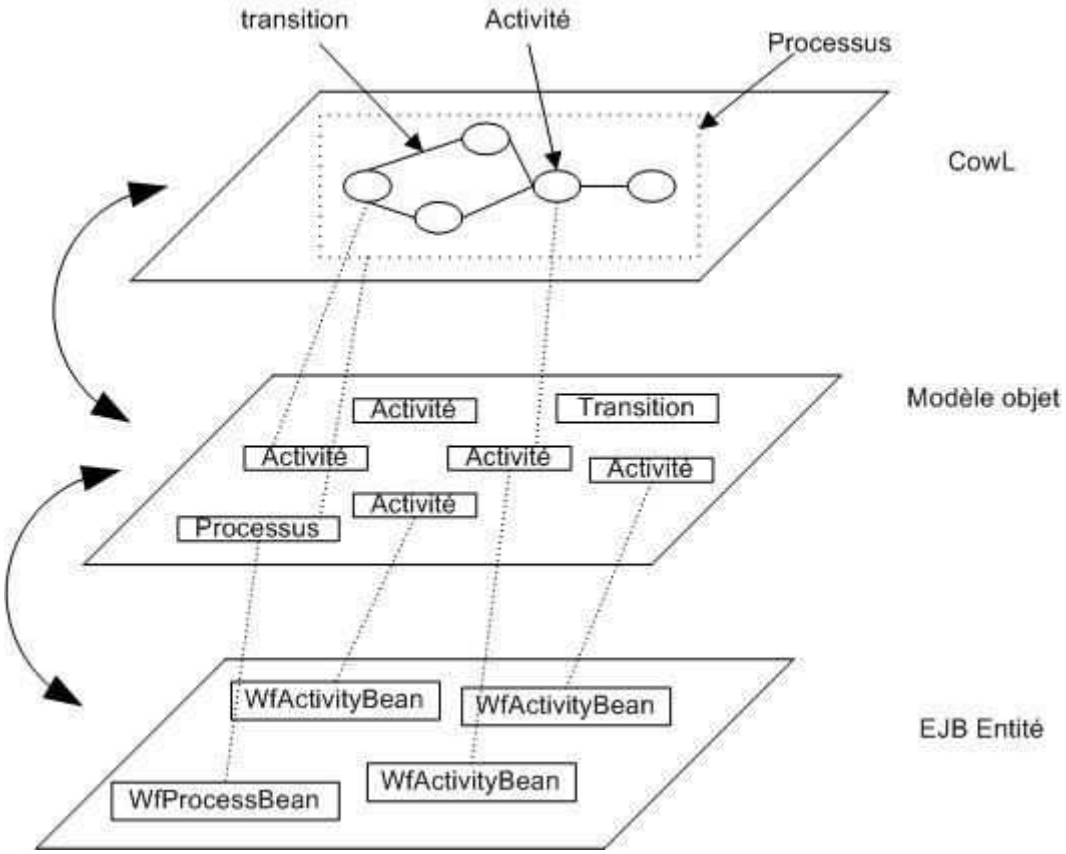

Figure 12 : Niveaux de modélisation de COW

La gestion du flux de contrôle est réalisée par un objet *ModelManager*. Cet objet manipule les objets correspondants au modèle. A chaque changement d'état d'une activité ou d'une tâche, il indique les nouvelles activités à instancier. Ces activités sont déterminées en fonction de l'évaluation des différentes transitions ainsi que des pré- et post- conditions des activités. La gestion du flux de données est organisée de la même manière.

La flexibilité est offerte en suivant une approche de type *Meta-Object Protocol* (Kiczales et al., 1991). Cela signifie que certaines façades sont dédiées à la consultation et à la modification des modèles en cours d'exécution et aux changements de comportement du moteur, notamment la façade *MetaManager* (figure 10b). Tous les changements sont effectués manuellement par les tuteurs ou les ingénieurs pédagogiques notamment en utilisant l'outil présenté à la figure 11 :

• La modification des processus consiste à ajouter/retirer des activités et des transitions ou modifier les modèles d'activité (*e.g.* changer les outils, ajouter des ressources pédagogiques, modifier l'assignement d'une tâche). Ces changements peuvent être réalisés par le tuteur pour un apprenant unique ou pour un groupe d'apprenant. Ils sont exprimés par un langage XML. Il est également possible d'exécuter les modifications graphiquement en utilisant l'outil d'administration (figure 11). Les changements seront répercutés au niveau des objets Java représentant le modèle via l'interface *MetaManager*, puis le nouveau modèle sera sauvegardé au format COWL. Il sera donc possible de recréer des instances du scénario modifié. Avec cette approche, nous ne pouvons pas défaire des activités terminées ayant déjà eu lieu. Le système n'exerce aucune vérification de conformité du nouveau modèle, *i.e.* si des activités sont ajoutées mais qu'aucune transition n'y mène le modèle sera tout de même considéré valide. Nous postulons que la

personne réalisant les modifications fera elle-même les vérifications.

• La modification du comportement permet le changement de la manière dont le modèle est interprété et des réactions du moteur face à certains événements comme par exemple la violation d'une contrainte temporelle. Si nous prenons le cas d'une durée maximale associée à une activité, il existe plusieurs méthodes permettant de l'évaluer comme par exemple en la considérant comme absolue ou en tenant compte des jours ouvrables. De même lorsqu'elle arrive à échéance, nous pouvons définir différentes manières de réagir comme la validation automatique, l'envoi d'un courrier électronique au tuteur, *etc*. Nous utilisons pour cela le patron de conception *stratégie* (Gamma et al., 1995).

5.6 Intégration de COW dans une plate-forme existante

Comme nous l'avons indiqué précédemment, notre service d'exécution de scénarios pédagogiques a pour vocation d'être intégré dans des plates-formes de formation existantes. Nous allons donc présenter dans cette section les moyens pour réaliser l'intégration et nous l'illustrerons par un cas concret, l'intégration de COW dans la plate-forme "Campus Virtuel" de la société Archimed (Archimed, 2006).

5.6.1 Moyens de communication avec COW

COW fonctionne isolément de tout LMS. Si nous souhaitons l'intégrer, cela sera au LMS d'initier les communications avec COW. Pour cela, les constituants du LMS utiliseront les interfaces externes de COW. Ces interfaces sont accessibles au moyen du protocole SOAP (*Simple Object Access Protocol*) (Chauvet, 2002), ce qui évite les problèmes d'interopérabilité, le protocole SOAP n'étant lié à aucune technologie propriétaire. Cela s'illustre bien avec le Campus Virtuel car celui-ci est construit autour de la technologie à composant logiciel COM/DCOM de Microsoft alors que notre service s'articule autour des composants logiciels EJB. Dans un souci d'interopérabilité, seulement des types simples (chaînes de caractères, entiers, ...) sont transmis dans notre cas. Cependant, nous structurons les informations échangées au moyen de documents XML qui transitent en tant que chaînes de caractères.

Afin d'augmenter la réactivité et permettre à COW d'avertir le LMS de changements, nous avons également mis en place une communication événementielle utilisant les événements du WMF et architecturée autour d'un bus à message JMS (*Java Message Service*). Il est ainsi possible, après la création de connecteurs adaptés, de faire appel à l'API du LMS.

Maintenant que nous connaissons les méthodes permettant de communiquer avec COW, nous allons déterminer de quelle manière connecter COW avec les composants du LMS.

5.6.2 Assemblage de COW avec un LMS

Pour assembler COW avec un LMS, nous devons dans un premier temps déterminer les constituants du LMS qui devront échanger de l'information avec COW. Le premier sera celui permettant la création et la modification des scénarios pédagogiques. Dans la majorité des cas ce composant sera absent du LMS et sera donc à développer entièrement. Dans le cadre du Campus Virtuel, il existait déjà. Il a donc été modifié pour faire appel aux méthodes de COW au lieu de l'ancien module. Toutefois, nous n'avons pas changé l'interface homme-machine qui convenait déjà parfaitement aux différents utilisateurs.

Les seconds composants à mettre en relation avec COW seront ceux gérant les comptes utilisateurs et les droits et outils associés comme par exemple une liste de tâches à réaliser ou un agenda. COW ne gère pas directement les utilisateurs, seuls des identifiants correspondant le plus souvent aux logins sont utilisés dans les fichiers d'instances lors de la création d'une nouvelle instance de scénario. La synchronisation entre les tâches créées par COW et la liste de tâches d'un utilisateur peut être faite en utilisant les deux méthodes de communication et cela en fonction du mode de fonctionnement du composant du LMS. La synchronisation peut se faire chaque fois que l'utilisateur cherche à connaître ses tâches, *i.e.* que le composant gérant la liste va interroger sa propre base d'informations puis ira demander à COW les tâches

associées à l'utilisateur. L'autre solution consiste à avertir le composant du LMS à chaque fois qu'une tâche est créée dans COW. Dans ce cas là, il sauvegardera lui-même l'information dans sa propre base. Pour l'intégration dans le Campus Virtuel, la première méthode a été utilisée.

Les troisièmes composants à mettre en relation avec COW sont les outils qui seront utilisés dans les scénarios pédagogiques. Les deux méthodes de communication seront mises en œuvre. Lorsqu'un outil associé à une tâche démarre, il ira demander à COW ses paramètres d'initialisation. Les résultats produits seront envoyés à COW afin d'être réutilisés dans les activités suivantes. Il est en effet possible de décrire explicitement le flux de données entre les activités.

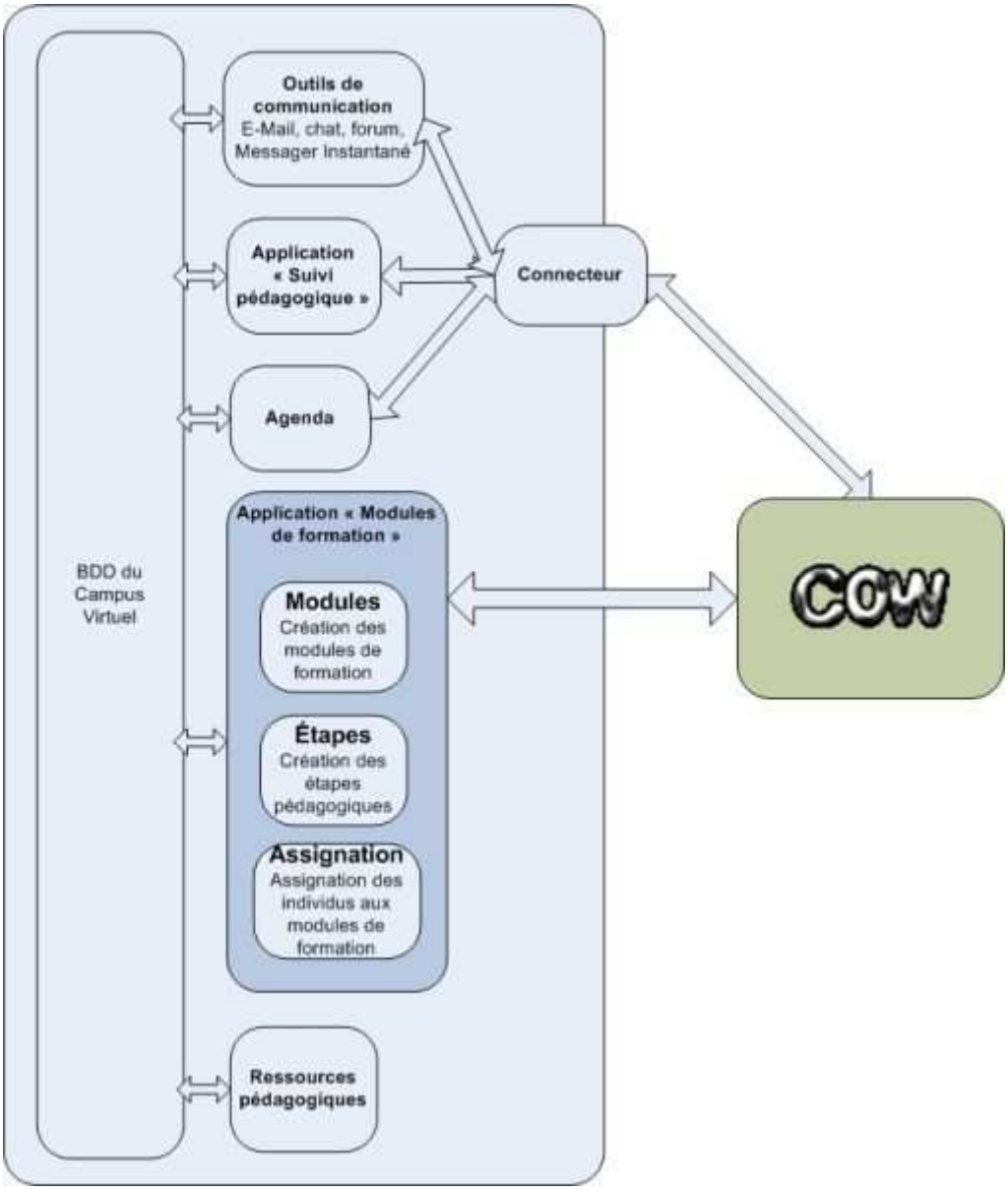

Figure 13 : Intégration de COW dans une plate-forme existante

La figure 13 représente le résultat de l'intégration de COW et du Campus Virtuel. Pour réaliser ce prototype, nous avons réalisé un connecteur permettant l'échange d'informations entre les deux parties, chacune utilisant sa propre base de données. Nous avons également modifié quelques unes des pages web du Campus Virtuel afin d'ajouter et présenter de nouvelles informations aux utilisateurs comme par exemple leur état d'avancement dans les scénarios en cours.

L'approche que nous venons de présenter peut être utilisé avec d'autres plates-formes. Nous l'avons notamment utilisée avec la plate-forme OpenUSS.

6 Travaux similaires

Un début d'ordonnancement automatique a été ajouté récemment dans des plates-formes comme Blackboard ou WebCT par l'utilisation de contraintes temporelles sur l'utilisation des ressources. Cependant, cela ne peut pas être comparé à l'utilisation de scénarios pédagogiques car la définition des dates d'utilisation est liée aux ressources pédagogiques et ne prend pas en compte les différentes activités. D'ailleurs, le temps est défini à l'avance et ne tient pas compte de l'avancement réel des activités alors que COW fournit les activités et les ressources quand elles sont nécessaires au regard du scénario pédagogique et peut gérer les contraintes de temps relatives au début du processus ou des activités.

Il existe peu de plates-formes qui fournissent un support à l'exécution de scénarios pédagogiques, nous pouvons notamment citer le système Flex-eL *(Flexible e-Learnin*g) (Lin et al., 2001), du DSTC (Distributed Systems Technology Centre) en collaboration avec l'university of Queensland en Australie, le projet *Virtual Campus* développé par le Politecnico di Milano (Cesarini et al., 2004), (VirtualCampus, 2006) et la plate-forme *CopperCore* de l'Open Universiteit Nederland (CopperCore, 2006).

Flex-eL consiste en une plate-forme d'éducation à distance basée sur un moteur de workflows flexible. Il existe une instance de processus d'apprentissage pour chaque étudiant. Cela permet une grande souplesse pour adapter les chemins d'apprentissage aux réels besoins de l'apprenant. Les groupes d'étudiants sont construits dynamiquement en rapprochant ceux réalisant les mêmes modules de formation au même moment. Les processus sont principalement séquentiels et les contraintes temporelles peuvent être relâchées.

Le projet Virtual Campus repose sur le moteur d'exécution de processus BizTalk (Biztalk, 2006) de Microsoft. Les concepteurs de scénarios pédagogiques peuvent ainsi tirer profit des outils de modélisation et des possibilités offertes par le langage XLANG (Thatte, 2001) pour exprimer l'organisation des activités pédagogiques.

CopperCore est une implémentation d'un moteur d'interprétation du langage IMS-LD. Il est basé sur une approche technologique similaire à COW, car il repose sur les composants EJB. COW est à mi-chemin entre CopperCore et les autres projets car il est basé sur les technologies de workflows plutôt que sur un langage de modélisation pédagogique particulier tout en offrant des mécanismes de projection permettant un futur support de IMS-LD.

7 Positionnement de nos travaux par rapport à IMS-LD

Depuis le début de nos travaux, le langage IMS-LD a émergé comme standard pour décrire les scénarios pédagogiques. Récemment un premier moteur d'exécution, CopperCore, est devenu disponible et quelques plates-formes comme par exemple EduPlone (EduPlone, 2006) ont commencé à supporter ce langage. Ce dernier semble en effet relativement compréhensible par un ingénieur pédagogique pour formaliser les scénarios. Nous avons développé un premier transformateur permettant de passer d'une description IMS-LD niveau A vers une description XPDL (Vantroys et Peter, 2003). Cependant, COW, basé sur un système de workflows, dispose des éléments permettant l'exécution de descriptions IMS-LD de niveau B (gestion des conditions et des propriétés) et C (notification). La gestion des conditions, de propriétés et la notification sont des éléments classiques des systèmes de workflows et sont inclus dans COW. IMS-LD souffre de quelques inconvénients comme par exemple l'expression de l'ordonnancement qui est principalement séquentiel (Caeiro et al., 2003) ou le manque de gestion des flux de données (notamment la réutilisation des savoirs produits).

8 Conclusion

Les scénarios pédagogiques provoquent un grand intérêt dans le domaine des Environnements Informatiques pour l'Apprentissage Humain. Ils peuvent ainsi être au centre de l'ingénierie de formation et sont des moyens pour définir l'utilisation des outils et des objets pédagogiques au cours d'un module ou d'une tâche dans lesquels les utilisateurs sont impliqués. Introduire les scénarios pédagogiques et les activités dans les plates-formes de gestion d'apprentissage doit se faire aussi bien au niveau pédagogique qu'au niveau technique. De nombreux acteurs et de nombreuses phases sont impliqués dans la définition et les différentes opérations des scénarios. L'ingénieur pédagogique est un acteur clé car il possède à la fois des compétences pédagogiques et techniques et réalise le lien entre les deux mondes. En effet, la production de scénarios pédagogiques n'est pas une tâche facile car il n'existe pas de méthodologie claire à l'heure actuelle. Pour ces raisons, des initiatives comme le projet européen UNFOLD (UNFOLD, 2006) sont créées pour supporter des communautés d'intérêts et de partages de connaissances Pour diminuer les coûts de production, les scénarios doivent avoir un haut niveau de réutilisation et doivent être facilement adaptés. Pour ces raisons, les scénarios ne doivent pas être trop liés aux objets pédagogiques et aux outils actuels qui doivent être choisis au déploiement et/ou à l'instanciation selon la plate-forme et les apprenants. L'adaptation continue des scénarios pédagogiques (*i.e.* durant l'exécution du scénario) est également importante pour fournir le meilleur résultat pour des groupes spécifiques d'apprenants.

Partant de l'idée que l'exécution d'un scénario pédagogique s'approche de l'exécution d'un processus par un moteur de workflows, nous avons développé un moteur de workflows flexible prenant en compte les caractéristiques propres à l'exécution de scénarios pédagogiques. En nous basant sur un cours de physique, nous avons montré de quelle manière pouvait être utilisé une version étendue du *XML Process Definition Language* pour modéliser un scénario pédagogique. La version étendue de ce langage permet la modélisation de travaux individuels ou de groupe à l'intérieur d'un processus et autorise la définition de contraintes temporelles. Le moteur permet la modification du modèle en cours d'exécution pour une adaptation en continu des processus et conserve les modèles modifiés pour une réutilisation facile de ces modèles augmentés qui pourront devenir à terme les modèles de référence.

Avoir une définition d'un scénario pédagogique avec les différentes activités à réaliser peut être utilisé pour appuyer les étudiants dans la gestion de leurs plannings et pour mieux appréhender de quelle manière ils vont atteindre les objectifs pédagogiques de l'unité d'apprentissage. Pour cela, nous avons développé un *guide d'apprentissage numérique, i.e.* une interface utilisateur qui aide les tuteurs et les apprenants à planifier leur travail à l'aide d'indicateurs d'avancement à l'intérieur du module et par rapport à l'avancement global du groupe.

Nous allons également renforcer la vision "service" que l'on peut avoir de notre système afin de pouvoir l'utiliser dans les futures plates-formes qui seront basées sur une architecture orientée service (ou *Service Oriented Architecture* : SOA).

Bibliographie

BOURGUIN G. (2000). *Un support informatique à l'activité cooperative fondé sur la théorie de l'activité : le projet DARE*. Thèse de Doctorat en Informatique, Université des Sciences et Technologies de Lille.

BOURGUIN G., DERYCKE A., TARBY J.-C. (2001). *Beyond the Interface : Co-evolution Inside Interactive Systems – A Proposal Founded on Activity Theory*. In J. VANDERDONCKT, A. BLANDFORD and P. GRAY (Eds) People and Computers XV – Interaction without Frontiers, Proceedings of IHM-HCI 2001, ISBN 1-85233-515-7, 297-310.

CAEIRO, M., ANIDO L., LLAMAS M. (2003). *A Critical Analysis of IMS Learning Design*. In B. WASSON, S. LUDVIGSEN and U. HOPPE (Eds) Designing for Change in Networked Learning

Environments, Proceedings of the International Conference on Computer Support for Collaborative Learning *2003*, ISBN 1-4020-1383-3, Kluwer Academic Publishers, 363-367.

CESARINI M., MONGA M., TEDESCO R. (2004). *Carrying on the elearning process with a workflow management engine*. In H. HADDAD, A. OMICINI, R. L. WAINWRIGHT, L. M. LIEBROCK (Eds.) *Proceedings of the 2004 ACM Symposium on Applied Computing (SAC)*, ISBN 1-58113-812-1, ACM, 940-945.

CHAUVET J.-M. (2002). *Services Web avec SOAP, WSDL, UDDI, ebXML*, Eyrolles ISBN 2-212-11047-2.

EALES, R. T. J., HALL T., BANNON L. J. (2002). *The Motivation is the Message: Comparing CSCL in different Settings*. In: G. Stahl (ed.) Computer Supported Collaborative Learning: Foundations for a CSCL Community (Proceedings of CSCL 2002), Lawrence Erlbaum Associates, 310-317.

GAMMA E., HELM R., JOHNSON R., VLISSIDES J. (1995). *Design Patterns: Elements of reusable Object-Oriented Software*, Addison-Wesley, ISBN 0-201-63361-2.

GROB H. L., BENSBERG F., DEWANTO B. L. (2004). *Developing, Deploying, Using and Evaluating an Open Source Learning Management System*. Proceedings of the 26th Int. Conf. Information Technology Interfaces ITI, Sagreb (Croatie), 387-393.

HUMMEL H., MANDERVELD J., TATTERSALL C., KOPER R. (2004). *Educational modelling language and learning design: new opportunities for instructional reusability and personalised learning*. International Journal on Learning Technology, Vol 1, No 1, 111-126.

IMS (2003). IMS Learning Design Information Model – version 1.0. IMS Global Learning Consortium Inc., disponible à l'adresse http://www.imsglobal.org/learningdesign/index.html (consulté en janvier 2006).

JACOBSON I., BOOCH G., RUMBAUGH J. (2000). *Le processus unifié de développement logiciel*, Eyrolles, ISBN 2-212-09142-7.

KICZALES G., DES RIVIèRES J., BOBROW G. (1991). *The Art of the Metaobject Protocol*, the MIT Press, ISBN 0-262-61074-4.

LIN J., HO C., SADIQ W, ORLOWSKA M. E. (2001). *On Workflow Enabled e-Learning Services*. In T. Okamoto, R. Hartley, Kinshuk, J. P. Klus (Eds.) Proceedings of the IEEE International Conference on Advanced Learning Technologies: Issues, Achievements and Challenges, ISBN 0-7695-1013-2, IEEE Computer Society, 349-352.

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

MATENA V., STEARNS B. (2001). Applying *Enterprise JavaBeans : Component-Based Development for the J2EE platform*. Addison-Wesley, ISBN 0201702673.

MULLER P.-A. (1999). *Modélisation objet avec UML*. Eyrolles, ISBN 2-212-08966-X.

OMG (2000). *Workflow Management Facility Specification, version 1.2*. Object Management Group, disponible à l'adresse http://www.omg.org/docs/formal/00-05-02.pdf (consulté en janvier 2006).

PAQUETTE G. (2001). *Designing Virtual Learning Centers*. In H. Adelsberger, B. Collis, J. Pawlowski (Eds) Handbook on Information Technologies for Education & Training within the Springer-Verlag series "International Handbook on Information Systems", pp. 249-272.

SCHMIDT-WESCHE B. (2003). *IBM WebSphere Platform User Roles*. IBM, disponible à l'adresse

http://www-106.ibm.com/developerworks/websphere/library/techarticles/0303_schmidt/schmidt.html (consulté en mars 2005).

ANDREW TANENBAUM. *Systèmes d'exploitation :Systèmes centralisés, systèmes distribués*. Informatique Intelligence Artificielle, InterEdition, Paris. ISBN :2-7296-0706-4.

THATTE S. (2001). *XLANG – Web Services for Business Process Design*. Microsoft Corporation, disponible à l'adresse http://godotnet.com/team/xml_wsspecs/xlang-c/default.htm. (consulté en mars 2005).

VANTROYS, T., PETER Y. (2003). *COW, a Flexible Platform for the Enactment of Learning Scenarios*. In: J. FAVELA and D. DECOUCHANT (eds.): Groupware: Design, Implementation, and Use - 9th International Workshop, CRIWG 2003. Vol. LNCS 2806, Springer-Verlag, 168-182.

WFMC (1995). The Workflow Reference Model, version 1.1. Technical Report WfMC-TC00-1003, Workflow Management Coalition, disponible à l'adresse http://www.wfmc.org/standards/docs/tc003v11.pdf (consulté en janvier 2006).

WfMC (2002). *Workflow Process Definition Inteface – XML Process Definition Language, version 1.0*. Workflow Management Coalition. WfMC-TC-1025, disponible à l'adresse http://www.wfmc.org/standards/docs/TC-1025_10_xpdl_102502.pdf (consulté en mars 2005).

Références à des sites Internet

Site officiel de la société Archimed :

http://www.archimed.fr (consulté en janvier 2006).

Site officiel de Microsoft BizTalk :

http://www.microsoft.com/biztalk/ (consulté en janvier2006).

Site officiel du moteur CopperCore :

http://www.coppercore.org (consulté en janvier 2006).

Site officiel du projet EduPlone :

http://www.eduplone.net (consulté en janvier 2006).

Site officiel de Java Enterprise Edition :

http://java.sun.com/j2ee/ (consulté en janvier 2006).

Site officiel de l'Object Management Group (OMG) :

http://www.omg.org (consulté en janvier 2006).

Site officiel de la plate-forme OpenUSS :

http://www.openuss.org (consulté en janvier 2006).

Site officiel du projet Virtual Campus du Politecnico di Milano :

http://www.elet.polimi.it/res/vcampus/ (consulté en janvier 2006).

Site officiel du projet UNFOLD :

http://www.unfold-project.net/UNFOLD (consulté en janvier 2006).

Site Officiel du Workflow Management Coalition (WfMC) :

http://www.wfmc.org (consulté en janvier 2006).

Références complémentaires non citées dans l'article :

Références bibliographiques

DILLENBOURG, P. (2002). Over-scripting CSCL: the risks of blending collaborative learning with instructional design. In P. A. Kirschner (Ed) *Three worlds of CSCL. Can we support CSCL*, Heerlen, Open Universiteit Nederland, 61-91.

A propos des auteurs

Thomas Vantroys est Maître de Conférences en informatique à l'Université des Sciences et Technologies de Lille (Lille 1) et membre du laboratoire TRIGONE. Il est concepteur du service COW. Sa recherche porte sur les systèmes de gestion d'activités et plus particulièrement sur l'utilisation des technologies de workflows dans les EIAH.

Courriel : Thomas.Vantroys@univ-lille1.fr

Toile : http://www.polytech-lille.fr/~tvantroy/

Yvan Peter est Maître de Conférences en informatique à l'Université des Sciences et Technologies de Lille (Lille 1) et membre du laboratoire TRIGONE. Sa recherche porte sur les infrastructures logicielles pour les EIAH et plus particulièrement l'exécution des scénarios pédagogiques. Il est membre du comité de direction de l'action Shared Virtual Laboratory du réseau d'excellence Kaleidoscope.

Courriel : Yvan.Peter@univ-lille1.fr

Toile : http://noce.univ-lille1.fr/peter/

Référence de l´article :

Thomas Vantroys, Yvan Peter, COW, un service de support d'exécution de scénarios pédagogiques, *Revue STICEF*, Volume 12, 2005, ISSN : 1764-7223, mis en ligne le 20/02/2006, http://sticef.org