HAL
open science

Adaptation à la volée de situations d'apprentissage modélisées conformément à un langage de modélisation pédagogique

Salim Ouari

To cite this version:

Salim Ouari. Adaptation à la volée de situations d'apprentissage modélisées conformément à un langage de modélisation pédagogique. Autre [cs.OH]. Université de Grenoble, 2011. Français. NNT : 2011GRENM063 . tel-00680028

HAL Id: tel-00680028

https://theses.hal.science/tel-00680028v1

Submitted on 17 Mar 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Informatique**

Arrêté ministériel : 7 août 2006

Présentée par

Salim OUARI

Thèse dirigée par **Jean-Pierre PEYRIN** et
coencadrée par **Anne LEJEUNE** et **Christine FERRARIS**

préparée, au sein du Laboratoire d'Informatique de Grenoble, et de l'équipe SysCom de l'Université de Savoie, dans l'Ecole Doctorale Mathématiques, Sciences et Technologie de l'Information, Informatique.

Adaptation à la volée de situations d'apprentissage modélisées conformément à un langage de modélisation pédagogique

Thèse soutenue publiquement le **25 novembre 2011**,
devant le jury composé de :

Bertrand David
Professeur à l'Ecole Centrale de Lyon (président)
Dominique Leclet
Maître de Conférences à l'Université de Picardie Jules Verne (rapporteur)
Philippe Vidal
Professeur à l'Université Paul Sabatier (rapporteur)
Anne Lejeune
Maître de Conférences à l'Université Pierre Mendès-France (examinateur)
Christine Ferraris
Maître de Conférences à l'Université de Savoie (examinateur)
Jean-Pierre Peyrin
Professeur à l'Université Joseph Fourier (examinateur)

Remerciements

Je tiens en premier lieu à remercier Monsieur Bertrand David, Professeur des Universités à l'Ecole centrale de Lyon, pour m'avoir fait l'honneur d'accepter de présider le jury de ma soutenance de thèse.
Je tiens aussi à remercier Monsieur Philippe Vidal, Professeur à l'Université de Toulouse, ainsi que Madame Dominique Leclet, Maître de conférences à l'Université d'Amiens, pour m'avoir fait l'honneur de rapporter et de juger mes travaux.
Je remercie également le cluster ISLE (Informatique, Signal, Logiciel Embarqué) d'avoir financé ce travail pendant trois ans.
Un grand merci à Monsieur Jean Pierre Peyrin, Professeur des Universités à l'Université Joseph Fourier de Grenoble, et Directeur de ma Thèse. Merci pour m'avoir soutenu, conseillé, et pour m'avoir permis d'être dirigé par quelqu'un que j'apprécie.
Un grand merci à Madame Christine Ferraris, Maître de Conférences à l'Université de Savoie, et Madame Anne Lejeune, Maître de conférences à l'Université Pierre Mendès France de Grenoble, pour avoir accepté de faire partie de mon jury. Merci de m'avoir encadré, conseillé et aidé au cours de cette thèse. Cette thèse n'aurait pas vu le jour sans vous, vraiment un grand merci.
Je tiens également à remercier tous les membres de l'équipe SCENARIO à l'Université de Savoie et tout particulièrement Monsieur Christian Martel et Madame Laurence Vignollet. Merci aux autres membres du laboratoire SysCom pour leur aide en diverses occasions.
Enfin, je tiens énormément à remercier ma fiancée, Kahina, ainsi que ma petite famille (Nassim, Karim et Lamia) pour leur soutien et leurs encouragements. Kahina, merci de m'avoir écouté, conseillé et parfois supporté. Un immense merci à ma mère et à mon défunt père (que dieu l'accueille en son vaste paradis), d'avoir cru en moi pendant toutes ces années, sans eux, sans leurs conseils, et sans leur amour, ce travail n'aurait pas abouti.

Résumé

Le travail présenté dans ce mémoire s'inscrit dans le domaine des Environnements Informatiques pour l'Apprentissage Humain (EIAH), plus précisément celui de l'ingénierie des EIAH dans le cadre d'une approche de type "Learning Design". Cette approche propose de construire des EIAH à partir de la description formelle d'une activité d'apprentissage. Elle suppose l'existence d'un langage de modélisation communément appelé EML (Educational Modelling Language) et d'un moteur capable d'interpréter ce langage. LDL est le langage sur lequel nous avons travaillé, en relation avec l'infrastructure LDI intégrant un moteur d'interprétation de LDL.

L'EML est utilisé pour produire un scénario, modèle formel d'une activité d'apprentissage. L'EIAH servant de support au déroulement de l'activité décrite dans le scénario est alors construit de manière semi-automatique au sein de l'infrastructure associée au langage selon le processus suivant : le scénario est créé lors d'une phase de conception ; il est instancié et déployé sur une plate-forme de services lors d'une phase d'opérationnalisation (choix des participants à l'activité, affectation des rôles, choix des ressources et services) ; le scénario instancié et déployé est pris en charge par le moteur qui va l'interpréter pour en assurer l'exécution.

Dans ce cadre, l'activité se déroule conformément à ce qui a été spécifié dans le scénario. Or il est impossible de prévoir par avance tout ce qui peut se produire dans une activité, les activités étant par nature imprévisibles. Des situations non prévues peuvent survenir et conduire à des perturbations dans l'activité, voire à des blocages. Il devient alors primordial de fournir les moyens de débloquer la situation. L'enseignant peut par ailleurs vouloir exploiter une situation ou une opportunité en modifiant l'activité en cours d'exécution.

C'est le problème qui est traité dans cette thèse : il s'agit de fournir les moyens d'adapter une activité "à la volée", c'est-à-dire pendant son exécution, de manière à pouvoir gérer une situation non prévue et poursuivre l'activité.

La proposition que nous formulons s'appuie sur la différentiation entre les données convoquées dans chacune des trois phases du processus de construction de l'EIAH : la conception,

l'opérationnalisation et l'exécution. Nous exhibons un modèle pour chacune de ces phases, qui organise ces données et les positionne les unes par rapport aux autres. Adapter une activité "à la volée" revient alors à modifier ces modèles en fonction des situations à traiter. Certaines nécessitent la modification d'un seul de ses modèles, d'autres conduisent à propager les modifications d'un modèle vers un autre.

Nous considérons l'adaptation "à la volée" comme une activité à part entière menée, en parallèle de l'activité d'apprentissage, par un superviseur humain qui dispose d'un environnement adéquat pour observer l'activité, détecter les éventuels problèmes et y remédier par intervention dans l'activité d'apprentissage en modifiant les modèles qui la spécifient. Pour développer les outils support à la modification et les intégrer dans l'infrastructure LDI, nous avons eu recours à des techniques de l'Ingénierie Dirigée par les Modèles. Les modèles manipulés dans ces outils en sont ainsi des données à part entière : les outils réalisés n'en offrent ainsi que plus de flexibilité et d'abstraction. Les modèles sont alors exploités comme des leviers pour atteindre et modifier les données ciblées par l'adaptation.

Table des matières

Remerciements **1**

Résumé **3**

Introduction **9**

I Contexte scientifique **13**

I.1 Ingénierie des EIAH 13

I.2 L'approche EML 14

I.2.1 Emergence des EML 14

I.2.2 Langage de notation abstrait 16

I.2.2.1 Modèle *calculable* 16

I.2.2.2 Réutilisabilité 16

I.2.2.3 Interopérabilité 16

I.2.3 Le standard IMS-LD 17

I.2.4 Cycle de vie d'un scénario pédagogique décrit conformément à un EML . 20

I.2.5 Outillage de l'approche EML suivant les phases du cycle de vie 21

I.2.5.1 Editeurs pour la conception des scénarios 22

I.2.5.2 Le player 23

I.3 Classification des EML 24

I.4 L'EML LDL et l'infrastructure LDI associée 25

I.4.0.3 Le méta-modèle LDL 25

I.4.1 LDI : une infrastructure pour l'opérationnalisation et l'exécution de scénarios LDL 30

I.4.1.1 Architecture de l'infrastructure LDI 31

I.4.1.2 Gestion de l'exécution des scénarios dans LDI 33

I.4.1.3 L'observation dans LDI 33

I.4.1.4 Quelques possibilités d'adaptation dans LDI 34
I.5 Conclusion 35

II Définition et étude de la problématique 37

II.1 Imprévisibilité d'une activité 38
II.1.1 Du point de vue des sciences sociales 38
II.1.2 Du point de vue de la Théorie de l'Activité 38
II.1.3 Du point de vue des sciences de l'éducation 40
II.2 Leçons apprises : l'expérimentation du scénario *LearnElec* 41
II.2.1 Le scénario *LearnElec* 41
II.2.2 Situation bloquante 41
II.3 Exemple fil rouge 43
II.3.1 L'étude de cas proposée dans le workshop AAMCS 43
II.3.2 Situations imprévues proposées dans l'étude de cas 44
II.4 Définition de la problématique d'adaptation 45
II.4.1 Adaptation à la volée 45
II.4.2 Ce qui n'est pas de l'adaptation à la volée 46
II.4.2.1 Adaptation prédéfinie 46
II.4.2.2 Adaptation par conception itérative 48
II.5 Caractérisation de l'adaptation à la volée 49
II.5.1 Tensions ou opportunités 49
II.5.2 Caractère bloquant d'une situation imprévue 49
II.5.3 Temporalité d'une situation imprévue 49
II.5.4 Portée d'une situation imprévue 50
II.5.5 Nature de ce qui est adapté 51
II.6 Cadrage de la problématique 52
II.6.1 Questions pour l'adaptation à la volée 52
II.6.2 Quels moyens pour l'observation 53
II.7 Bilan et conclusion 54

III Etat de l'art : adaptation à la volée 57

III.1 De l'adaptation dans le domaine du Learning Design 58
III.1.1 Approches d'adaptation prédéfinie 58
III.1.1.1 Adaptation par modélisation avancée 59

III.1.1.2 Des EMLs supportant l'approche de modélisation avancée 60
III.1.1.3 Adaptation par modélisation différée 62
III.1.1.4 Conclusion 62
III.1.2 Approches d'adaptation à la volée dans le domaine du LD 62
III.1.2.1 Poches d'adaptation : proposition de Zarraonandia 63
III.1.2.2 Adaptation à la volée dans le cadre du projet *TENCompetence* . 64
III.1.2.3 L'adaptation à la volée et PoEML : proposition de Perez-Rodriguez 65
III.1.2.4 Adaptation à la volée dans LAMS 67
III.2 De l'adaptation dans le domaine du Workflow 68
III.2.1 Ce qu'est le workflow 68
III.2.2 Parallèle avec le Learning Design 69
III.2.2.1 Les similitudes entre Learning Design et Workflow 69
III.2.2.2 L'imprévisibilité dans le Workflow 70
III.2.2.3 Alignement du vocabulaire 70
III.2.3 Les exceptions dans le Workflow 71
III.2.3.1 Définition 71
III.2.3.2 Caractérisation des exceptions 72
III.2.3.3 Classification des exceptions 73
III.2.4 Gestion des exceptions dans le Workflow 74
III.2.4.1 Adaptation prédéfinie par configuration de modèles de workflow 74
III.2.4.2 Adaptation à la volée dans le workflow 76
III.3 Conclusion sur l'adaptation dans le LD et dans le Workflow 80

IV Première proposition : adaptation par propagation 83
IV.0.1 Situations imprévues proposées dans l'étude de cas 83
IV.1 Première proposition : adaptation par propagation 84
IV.1.1 Hypothèse : l'adaptation à la volée doit prendre en compte la nature de la situation imprévue 84
IV.1.2 Niveaux des informations liées à l'exécution du scénario 84
IV.1.3 Relations entre les niveaux d'une activité 85
IV.1.4 Niveau d'une adaptation à la volée 86
IV.1.5 Propagation d'une adaptation à la volée 87
IV.1.5.1 Processus d'une adaptation par propagation 88
IV.1.6 Adaptation par propagation complète 89

IV.2 Un premier prototype pour l'adaptation par propagation 90
IV.2.1 Architecture du prototype 90
IV.2.2 Exemple d'utilisation du prototype 92
IV.3 Conclusion 95

V Deuxième proposition : utilisation de l'IDM pour l'adaptation par propagation 97

V.1 Hypothèse : un méta-modèle pour chaque niveau 98
V.2 Rôle de l'Ingénierie Dirigée par les Modèles 100
V.3 Méta-modèle lié à la phase d'opérationnalisation 101
V.4 Méta-modèle lié à la phase d'exécution 102
V.5 Un méta-modèle pour spécifier des modifications 104
V.5.1 Avantages d'un modèle de spécification des modifications 105
V.5.2 Les actions de base du méta-modèle des modifications 106
V.5.3 Un exemple d'utilisation du méta-modèle des modifications 107
V.6 Mise en œuvre d'un *Framework* pour l'adaptation par propagation 109
V.6.1 Architecture du *Framework* 109
V.6.2 Processus d'adaptation par propagation dans le *Framework* 111
V.6.3 Déroulement d'un exemple d'adaptation par propagation dans le *Framework* 113
V.7 Conclusion 120

VI Conclusion et perspectives de nos travaux 121

Bibliographie 131

Introduction

Notre travail de thèse s'inscrit dans le domaine des Environnements Informatique pour l'Apprentissage Humain (EIAH). Notre problématique relève de l'Ingénierie des EIAH considérés en tant que logiciels conçus dans le but d'amener des apprenants à développer une activité favorable à l'atteinte des objectifs de la situation d'apprentissage visée [Tch09].

Plus précisément, notre recherche porte sur l'ingénierie d'EIAH dans lesquels la situation d'apprentissage informatisée a été modélisée conformément à un EML (*Educational Modelling Language*). Selon la définition proposée par le Comité Européen de Normalisation (CEN ISS WS LT) un EML est un méta-modèle conceptuel dédié à la conception de modèles calculables [1] de situations d'apprentissage dans une perspective de réutilisation et d'interopérabilté : "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" [RVRK+02].

Le plus souvent, le modèle de la situation modélisée avec un EML est appelé "scénario pédagogique". Dans la suite de ce mémoire, il faudra prendre sous ce sens l'expression scénario pédagogique. La définition donnée par [PL04] concrétise cette dénomination : "Un scénario d'apprentissage représente la description, effectuée a priori ou a posteriori, du déroulement d'une situation d'apprentissage ou unité d'apprentissage visant l'appropriation d'un ensemble précis de connaissances, en précisant les rôles, les activités ainsi que les ressources de manipulation de connaissances, outils et services nécessaires à la mise en œuvre des activités". Durant cette mise en œuvre, le scénario pédagogique passe par trois phases, la conception (ou création), la contextualisation (ou opérationnalisation) et enfin la phase d'exploitation (ou exécution) [PL04]. Ces trois phases constituent le Cycle De Vie (CDV) du scénario pédagogique. Durant ce cycle, le scénario pédagogique est transformé de manière automatisée ou semi-automatisée (outillée)

1. Calculable est à prendre au sens "compréhensible par une machine"

afin de générer l'EIAH, support à l'activité d'apprentissage effective.

Le choix d'utiliser un EML - considéré selon la définition du CEN ISSS WS LT - en phase de conception d'un EIAH, garantit *a priori* l'indépendance du modèle produit (le scénario pédagogique) relativement aux technologies supportées par les environnements cibles, environnements dans lesquels la situation d'apprentissage se déroulera effectivement [RVRK+02, KT05, Lej04, FLVD05]. Par ailleurs, grâce à sa nature de "modèle", le scénario pédagogique n'est pas intrinsèquement lié à des ressources d'apprentissage particulières ni à des services particuliers mais fait référence à des types de ressources ou services dont l'utilisation par les acteurs de la situation décrite doit permettre d'atteindre les objectifs visés. De fait, les modèles produits peuvent être réutilisés et éventuellement adaptés en fonction des besoins.

Le fait de scénariser formellement une situation d'apprentissage a pour conséquence immédiate d'introduire un fort degré de prescription. Notre recherche s'intéresse aux effets induits par l'imprévisibilité de l'activité effective en regard à l'activité prescrite. En effet, dans certains cas, l'activité produite par l'instanciation du scénario pédagogique peut être confrontée à des situations dites imprévues. Nous appelons *situation imprévue*, un événement qui perturbe le déroulement prévu du scénario pédagogique. Par exemple, un problème technique (une ressource inaccessible), un participant en difficulté ou des groupes d'apprenants non équilibrés sont des situations imprévues. Certaines situations imprévues peuvent entrainer l'interruption de l'activité d'apprentissage et par conséquent son échec.

Dans notre domaine d'étude, plusieurs recherches s'intéressent à prendre en compte lors de la conception de ce type d'EIAH des considérations liées à la grande variabilité des profils des acteurs cibles - on parlera alors de personnalisation ou d'adaptation prédéfinie en fonction des profils des utilisateurs [BS06, SB06, BSB+04]. La personnalisation permet en effet de bâtir des scénarios pédagogiques supportant une certaine flexibilité, mais ne répond pas au problème posé par l'imprévisibilité des activités d'apprentissage et au besoin de les adapter en temps réel.

Notre objectif dans cette thèse est d'étudier la rigidité des EIAH résultant de l'utilisation d'un EML et de proposer une approche permettant de les adapter à la volée [2]. Cela nous conduit à étudier les caractéristiques des situations imprévues et à voir dans quelle mesure il est possible

2. durant le déroulement même de l'activité d'apprentissage

de mettre en place des moyens permettant de les adapter.

Plan de la thèse

Ce mémoire est organisé en deux parties. La première partie présente le contexte scientifique et la problématique. Elle rassemble les chapitres I, II, III.

Dans le chapitre I, nous présentons les différents domaines liés à notre recherche. Nous nous intéressons ainsi aux EIAH, à l'ingénierie des EIAH et au Learning Design. Nous y présentons également le langage LDL et l'infrastructure LDI que nous avons exploités dans nos travaux pour développer les solutions proposées dans nos contributions.

Dans le chapitre II, nous développons la problématique traitée dans cette thèse. Nous commençons par la définition d'un exemple fil rouge qui nous servira tout au long du mémoire. Cet exemple nous permettra ainsi de dégager des études de cas que nous utiliserons pour étudier et caractériser la problématique. Nous présentons aussi un cadrage pour circonscrire les éléments auxquels nous nous restreindrons dans le traitement de la problématique. Nous terminons ce chapitre par un bilan qui donne un résumé des éléments dégagés de l'étude de la problématique.

Dans le chapitre III, nous présentons ce qu'est l'adaptation et les divers travaux de recherche qui s'y intéressent dans le domaine du Learning Design (LD) et du Workflow. La conclusion de ce chapitre consiste à identifier dans la littérature les moyens manquants pour répondre à la problématique d'adaptation à la volée, l'objectif étant de permettre de flexibiliser davantage les activités qui s'exécutent conformément à des modèles formels et notamment les activités d'apprentissage dans un contexte LD.

La seconde partie présente les propositions que nous formulons dans le cadre de cette thèse. Elle est composée des chapitres IV, V, VI.

Dans le chapitre IV, nous présentons notre première proposition pour répondre aux besoins dégagés dans les chapitres II et III. Nous y présentons ainsi les concepts et mécanismes fondamentaux de notre approche d'adaptation à la volée. Nous montrons aussi dans quelle mesure il est nécessaire de mettre en place une ingénierie pour guider l'adaptation à la volée dans un contexte LD et nous présentons un prototype d'expérimentation de notre approche.

Dans le chapitre V, nous complétons la proposition faite dans le chapitre IV. Nous y présentons la démarche que nous avons suivie pour implanter et mettre en œuvre notre approche d'adaptation à la volée. Nous y décrivons notamment les méta-modèles et les modèles que nous avons implémentés dans le *Framework* proposé.

Dans le chapitre VI, nous présentons le bilan des travaux réalisés. Enfin, nous concluons ce

chapitre et ce mémoire par les perspectives de nos travaux futurs.

Chapitre I

Contexte scientifique

Nous avons présenté le contexte de notre recherche dans l'introduction de cette thèse. Le rôle de ce chapitre est de poser les éléments nécessaires pour la compréhension de notre problématique de recherche, de nos contributions et des travaux réalisés. Pour apporter ces éléments, nous parcourrons les différents domaines dans lesquels se situent nos travaux. Nous nous intéresserons ainsi à l'Ingénierie des EIAH (IEIAH) et aux langages de modélisation pédagogique (EML). Nous terminerons ce chapitre par la présentation de notre terrain d'expérimentation : l'EML LDL et son infrastructure.

L'EIAH est un domaine de recherche pluridisciplinaire : pédagogie, didactique, psychologie cognitive, sciences de l'éducation et informatique, etc. Selon [Tch09], les travaux sur les EIAH peuvent être abordés de différents points de vue. Du point de vue des Sciences Humaines et Sociales, où sont abordées les problématiques liées à l'enseignement et l'apprentissage, et du point de vue de l'Informatique où on aborde les problématiques liées à la conception et à la réalisation des EIAH. Dans notre recherche, c'est ce dernier point de vue qui nous intéresse et que nous prendrons en considération dans le reste de ce mémoire.

I.1 Ingénierie des EIAH

La construction des EIAH nécessite le plus souvent un processus complexe, car elle fait appel à des expertises et des connaissances pouvant provenir de diverses disciplines. De plus, ces processus doivent s'adapter continuellement aux technologies actuelles. C'est ce que propose l'Ingénierie dans le domaine des EIAH, à travers la définition de concepts, de méthodes et de techniques reproductibles et/ou réutilisables facilitant la mise en place (conception - réalisation

- expérimentation - évaluation - diffusion) des EIAH [Tch02].

Dans le contexte de notre recherche, nous nous intéressons à des suites logicielles permettant la conception conformément à un EML de modèles de situations d'apprentissage et l'instanciation, le déploiement et l'exécution dans un environnement cible de ces modèles. Par extension nous pouvons considérer que chaque instance d'un modèle de situation d'apprentissage est assimilable à un EIAH considéré en tant que logiciel d'apprentissage. Dans la suite de ce mémoire, nous ferons référence à cette approche en utilisant les termes *approche EML*. Nous donnons plus de détails (origines, mise en œuvre, concepts, outils) concernant cette approche dans la section qui suit.

I.2 L'approche EML

L'approche *EML* repose sur des langages de modélisation dont le but est de faciliter la modélisation d'une unité d'apprentissage (cours, leçon, séquence d'activités [KO04]). Chaque modèle produit est indépendant de l'environnement d'exécution cible, tout en étant calculable, c'est-à-dire, moyennant l'existence d'outils logiciels appropriés, instanciables, déployables dans un environnement cible et interprétables [BAK05].

I.2.1 Emergence des EML

Le début des années 2000 a été marqué par une nouvelle approche dans le développement des solutions e-learning, celle de la conception d'unités d'apprentissage [1] conformément à des langages de modélisation pédagogique. Ces langages ont pour objectif de permettre l'expression d'approches pédagogiques variées et de garantir l'interopérabilité des solutions e-learning qui leur sont conformes.

C'est ainsi que fut publié en décembre 2000 par l'OUNL (Open University of Netherlands) le langage Educational Modelling Language (EML [2]) comme résultat d'un projet de recherche et développement initié par la nécessité pour la Hollande de renouveler son système universitaire en implémentant de nouveaux modèles éducatifs intégrés dans des environnements informatiques.

1. Une unité d'apprentissage (UA) est définie comme une section d'apprentissage délimitée (cours, module, leçon) qui ne peut être dissociée en sous-unités sans perdre son sens ou sa capacité à satisfaire ses objectifs pédagogiques [Koper 2003].

2. EML a un double sens qui peut prêter à confusion : ici, il désigne le langage proposé par l'OUNL, plus généralement dans ce mémoire, il représente l'acronyme de tout langage de modélisation pédagogique.

EML est défini par Koper comme un modèle relationnel sémantiquement riche, décrivant les contenus et les parcours au sein d'unités d'apprentissage, selon un point de vue pédagogique et dans un objectif de réutilisabilité et d'interopérabilité [Kop01]. Ce langage a servi de base au développement de la spécification *Learning Design* (LD) du consortium IMS [3] [IMS11], publiée en février 2003 [Lej04].

Comme le rappellent Koper et Tattersal [KT05], un groupe de 33 experts du e-learning s'est réuni en mars 2002 dans le but d'améliorer la qualité pédagogique des cours en ligne, leur portabilité et leur outillage. Ce groupe, nommé "Valkenburg Group", reconnaissant la pauvreté des cours en ligne existants, partageait le principe suivant : "Pour remplir sa mission, le e-learning doit offrir aux apprenants des cours et des programmes attractifs tout en fournissant un environnement agréable et efficace pour les équipes pédagogiques en charge du développement des matériels pédagogiques, de la planification de l'enseignement, du tutorat et de l'évaluation." [ibid]. A l'issue de ses travaux, le Valkengburg group s'est consensuellement entendu sur le fait qu'EML et LD fournissaient un bon point de départ pour atteindre cet objectif. Bien que différents par leur structure, ces deux langages de modélisation pédagogique sont fonctionnellement plus ou moins équivalents et rendent possible la conception et l'opérationnalisation de cours e-learning avancés et interopérables reflétant un nombre illimité d'approches pédagogiques par abstraction de celles décrites dans la littérature (jeux de rôles, jeux éducatifs, apprentissage par résolution de problème, approches socio-constructivistes, apprentissage adapté au profil de l'apprenant, etc.).

En parallèle ou dans le sillage du développement des spécifications EML et LD, d'autres langages de modélisation pédagogique ont été proposés. Nous exposerons rapidement les concepts de LD (qui reprend les principaux concepts d'EML) puis, plus en détail, ceux du langage de modélisation pédagogique que nous avons utilisé dans nos travaux, Learning Design Language (LDL), développé en collaboration entre les équipes SYSCOM et MeTAH qui co-encadrent notre travail de thèse [MVF+06]. Auparavant, nous retraçons dans le paragraphe suivant les principes phares de l'approche EML et ses apports pour l'ingénierie des EIAH.

3. IMS représente un regroupement de 250 établissements éducatifs dont le MIT (Massachusetts Institute of Technology) et UCM (Université Carnegie Mellon), d'entreprises telles que Apple et IBM, d'agences gouvernementales telles qu'Industrie Canada et des sociétés de développement telles que Canvas Learning et Blackboard.

I.2.2 Langage de notation abstrait

Une unité d'apprentissage est formellement décrite selon des concepts pédagogiquement neutres et indépendants des contenus d'apprentissage utilisés (donc des connaissances manipulées, des disciplines auxquelles elles se rapportent, des contextes organisationnels ou de la population cible) ou des services manipulés. La flexibilité pédagogique est donc intrinsèque à ces modèles et dans le cas de LD elle a été éprouvée au travers de plusieurs implémentations et avec des situations d'apprentissage variées et dans différents contextes [KM04, KT05]. Néanmoins, la modélisation de certaines approches pédagogiques peut être particulièrement difficile à mettre en œuvre et nécessite une grande expertise des différents concepts du langage [PG06].

I.2.2.1 Modèle *calculable*

La description d'une unité d'apprentissage conformément à un EML conduit à l'obtention d'un document dit *calculable* dans la mesure où il peut alors être compris par des programmes qui en assureront l'opérationnalisation et le déploiement dans un environnement informatique. Le plus souvent, le document produit est exprimé au format XML.

I.2.2.2 Réutilisabilité

A partir du moment où les unités d'apprentissage décrites avec un EML distinguent formellement les activités des apprenants ou de l'équipe pédagogique des ressources et services qu'elles utilisent, il devient plus facile d'en réutiliser tout ou partie moyennant d'éventuelles modifications liées à la prise en compte d'un contexte particulier ou de l'utilisation de ressources (ex : une vidéo à la place d'une image fixe) ou services spécifiques (ex : un wiki à la place d'un forum par exemple) [PL04]. Des patterns d'orchestration d'activités reflétant une approche pédagogique particulière peuvent également être partagés [LPD04]. En ce sens, et sous réserve d'une bonne documentation des modèles produits (et notamment de l'explicitation des objectifs pédagogiques de l'unité d'apprentissage), la réutilisabilité pour les concepteurs de situations d'apprentissage en ligne dépasse le seul partage de contenus ou de services.

I.2.2.3 Interopérabilité

Comme nous l'avons rappelé précédemment, l'émergence de l'approche EML a également été motivée par le besoin d'interopérabilité nécessaire à la mise en place de situations d'apprentissage en ligne. Sur ce point, beaucoup de travail reste à faire pour qu'un scénario pédagogique modélisé avec LD puisse échanger des informations avec un scénario pédagogique modélisé conformément

à un autre EML ou encore avec une simulation dédiée à l'apprentissage. Néanmoins le fait de disposer d'un modèle d'information et d'agrégation sémantique calculable ouvre de réelles perspectives dans cette voie et certains travaux en font état, comme ceux réalisés dans le cadre du projet SVL du réseau d'excellence européen Kaléidoscope où il a en particulier été démontré l'interopérabilité d'un scénario pédagogique modélisé conformément à LDL avec des scénarios de plus petit grain modélisés conformément à un méta-modèle de scénarios de travaux pratiques basés sur des exercices de manipulation d'une simulation ou d'un micro-monde [MAL07].

I.2.3 Le standard IMS-LD

La spécification IMS Learning Design (IMS-LD) fournit un cadre conceptuel de modélisation d'une UA instrumentée par les technologies d'information et de communication. Ce cadre conceptuel inclut un méta-modèle (cf. Figure I.1) permettant de concevoir des situations d'apprentissage en se basant sur la métaphore de la scène théâtrale.

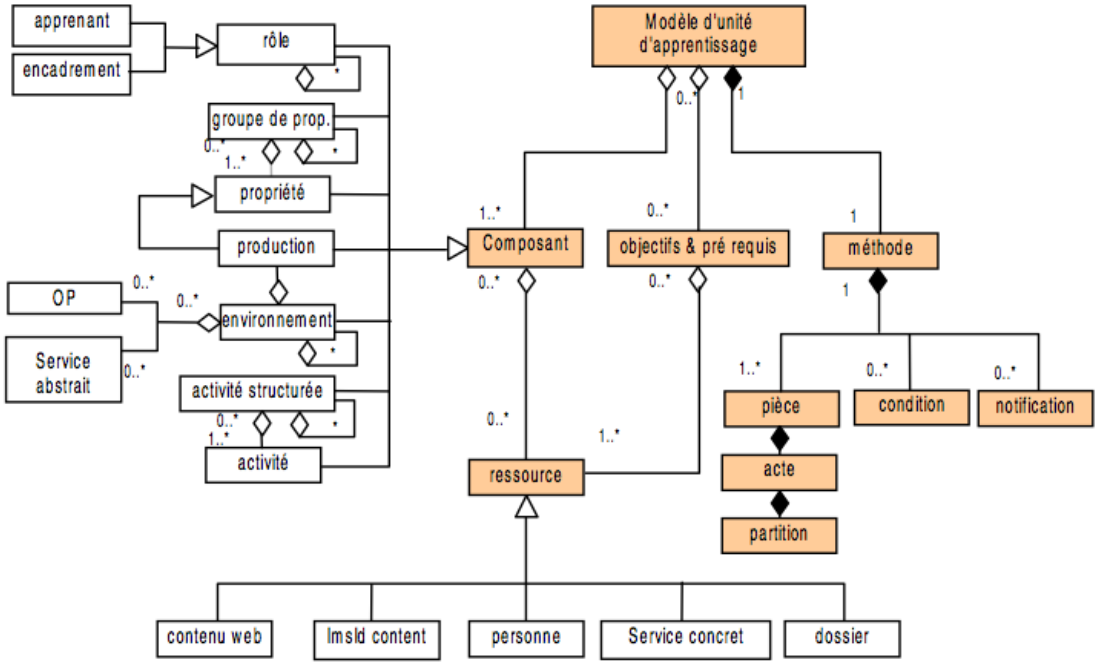

Figure I.1 : Méta-modèle de la spécification IMS-LD. Source [Lej04]

Les tâches que les différents utilisateurs de l'UA ont à accomplir sont décrites et organisées comme des scènes d'une pièce de théâtre où des acteurs doivent jouer des rôles en respectant les indications du metteur en scène dans un décor [Lej04]. La conception d'une UA peut se faire à travers les étapes suivantes :

- déterminer les pré-requis ainsi que les objectifs pédagogiques nécessaires pour le cours ;
- définir les activités d'apprentissage du cours ;
- désigner les ressources, les outils et les services que requièrent ces activités ;
- déterminer l'ordonnancement et la synchronisation des activités.

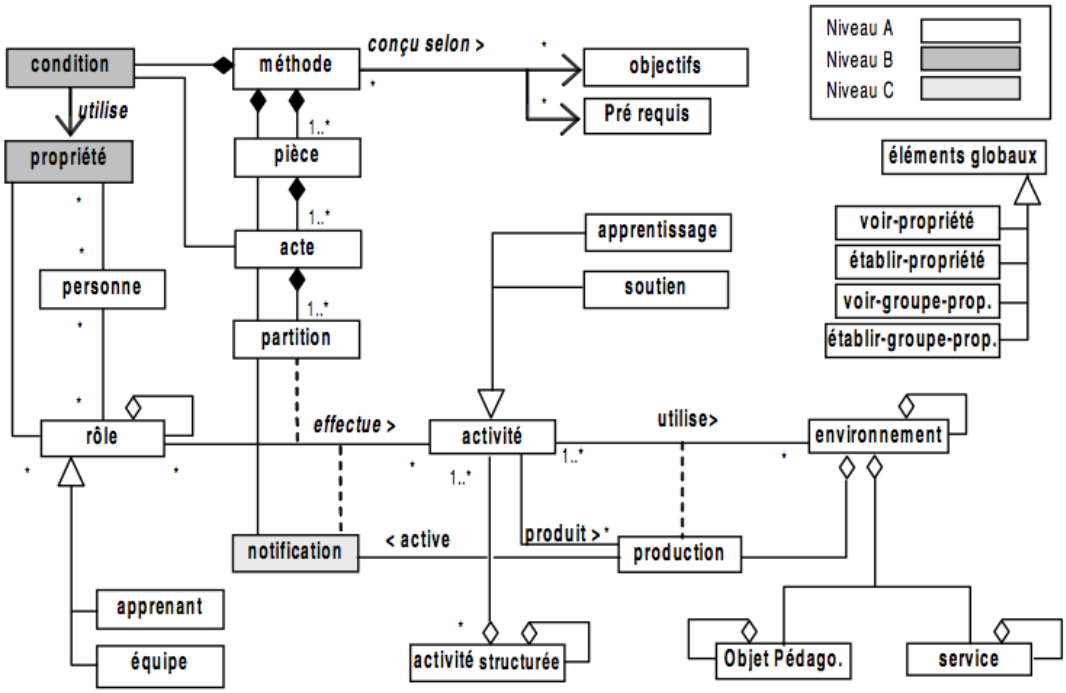

Figure I.2 : Modèle relationnel d'IMS LD. Source [Lej04]

Les concepts du méta-modèle d'IMS-LD sont organisés en trois niveaux A, B et C (cf. Figure I.2).

Le niveau *A* : Le niveau *A* regroupe tous les concepts du méta-modèle d'IMS-LD à l'exception des propriétés, des conditions et des notifications. Dans ce niveau, la conception de l'UA s'appuie sur les éléments suivants :

- **Les objectifs et pré requis :** Les activités sont caractérisées par des *objectifs* et des *pré-requis* qui fixent le cadre d'utilisation en termes de connaissances ou de compétences pour l'activité modélisée par l'UA [Lej04] ;
- **La méthode :** cet élément décrit l'organisation de l'unité d'apprentissage. Les *pièces* qui composent la méthode représentent les alternatives de scénarios de déroulement de l'UA. Chaque pièce est structurée en *actes* qui sont à ce niveau exécutés séquentiellement. Les actes sont formés de *partitions* qui associent les rôles aux activités effectuées dans un

environnement (décor) [Lej04, FLVD05] ; les partitions se déroulent en parallèle au sein d'un acte ;

- **Les composants :** L'élément *Composant* rassemble toutes les entités nécessaires à la mise en place de la situation d'apprentissage : les rôles impliqués, les différentes activités, la description des propriétés propres à un rôle ou à une personne, les objets d'apprentissage et les services manipulés ;
- **Les ressources :** Les ressources spécifient les éléments concrets de l'environnement d'exécution : ressources physiques au format XHTML, les personnes physiques affectées à un rôle lors de l'exécution d'une UA, ressources physiques associées à un service abstrait (service de messagerie, chat, forum, etc.), ou encore, ressources physiques de type document HTML.

Les UA décrites à ce niveau peuvent correspondre à des approches pédagogiques différentes, mais leur exécution sera similaire d'une session à l'autre. La personnalisation (dans le sens adaptation) de l'UA est réalisée en utilisant les éléments du niveau B.

Le niveau B :

Ce niveau permet de contrôler et de personnaliser l'exécution de l'UA à travers l'utilisation de *conditions* et de *propriétés* :

- Les **conditions** permettent de contraindre l'exécution des activités dans l'UA par l'évaluation d'expressions conditionnelles. Les conditions sont de la forme $si < expression > alors < action1 > sinon < action2 >$. Elles permettent de prendre en compte les propriétés caractérisant les personnes et les utilisateurs associés à l'unité d'apprentissage ;
- Les **propriétés** sont des variables locales ou globales, relatives à un rôle ou à une personne. Le test de leur valeur par des expressions conditionnelles permet, entre autres, la personnalisation du déroulement d'une unité d'apprentissage.

Le niveau C :

L'élément **notification** disponible à ce niveau permet de dynamiser l'exécution d'une unité d'apprentissage par le déclenchement d'actions spécifiques suite à l'observation d'événements (fin d'un acte, évaluation d'une condition, mise à jour d'une propriété, etc.)

I.2.4 Cycle de vie d'un scénario pédagogique décrit conformément à un EML

Le cycle de vie du scénario est un processus qui couvre les différentes phases à travers lesquelles se met en place la situation d'apprentissage. Il consiste à rendre le scénario exécutable. Après une phase de conception dont le résultat est un modèle de scénario, ou scénario abstrait, conforme à l'EML choisi, suit une phase de contextualisation dans laquelle les rôles décrits dans le scénario abstrait sont distribués à des personnes physiques, les ressources et les services abstraits sont instanciés sur des ressources et des services concrets et la date de mise en exploitation est déterminée. La dernière phase consiste en l'exécution sur la plateforme cible du scénario préalablement contextualisé (ou opérationnalisé). Le cycle de vie d'un scénario pédagogique est représenté dans la figure I.3.

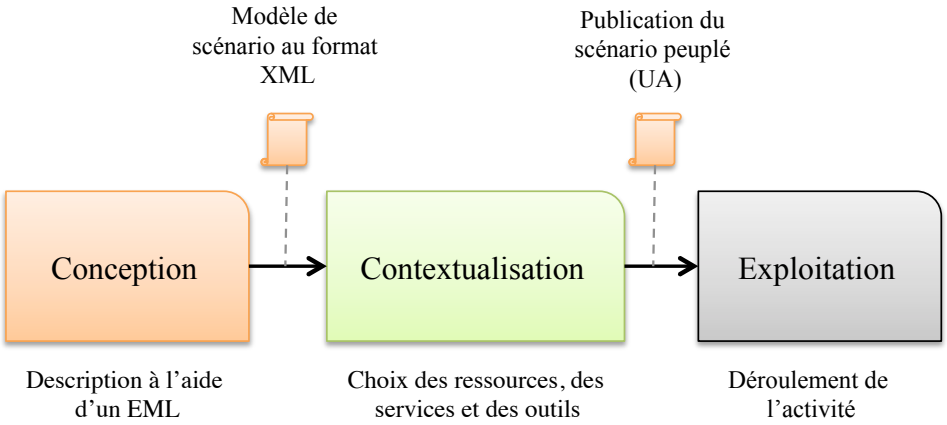

Figure I.3 : Cycle de vie d'un scénario pédagogique

Il est constitué d'au moins trois phases qui sont :

- **la conception :** c'est la description du modèle du scénario abstrait ;
- **la contextualisation :** choix des personnes physiques, des ressources et des services requis pour le déroulement de l'activité ;
- **l'exploitation** : durant cette phase, le scénario est interprété par un moteur d'exécution pour permettre aux participants d'accéder aux ressources et services, dans la limite de leurs rôles.

[PL04] introduit deux autres phases qui sont : l'*évaluation* du déroulement de la situation d'apprentissage et la *réutilisation* du scénario après une nouvelle phase de contextualisation.

Ce cycle de vie peut être différent en fonction de l'EML utilisé. Prenons comme exemple,

l'EML LDL que nous avons exploité dans nos travaux et que nous présentons plus loin dans ce chapitre. Le cycle de vie d'un scénario LDL est un peu différent, car constitué d'une phase d'opérationnalisation comme le montre la figure I.4.

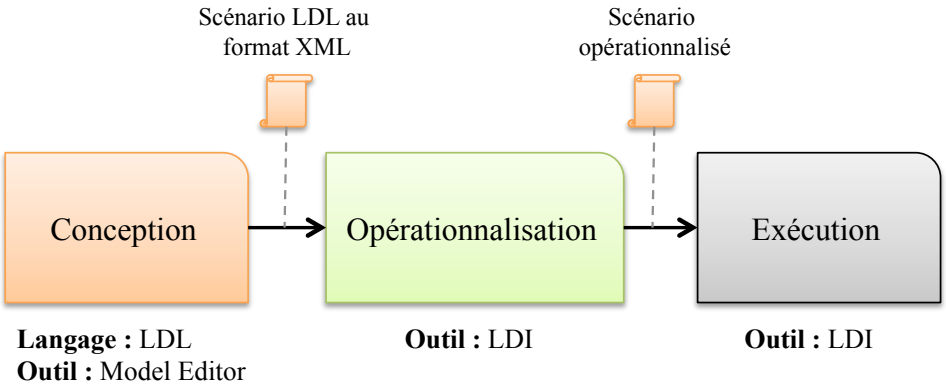

Figure I.4 : Cycle de vie d'un scénario *LDL*

L'opérationnalisation du scénario LDL se fait en utilisant un outil appelé Learning Design Infrastrcuture (LDI). L'opérationnalisation dans LDI consiste alors à :

- déployer le scénario LDL dans un environnement doté d'un moteur d'exécution capable de l'interpréter ;
- instancier et transformer le scénario XML en du code pouvant être interprété par un moteur d'exécution ;
- choisir les participants de l'activité en fonction des critères établis par le scénario ;
- attribuer à ces participants leurs rôles dans le scénario ;
- sélectionner les services et les ressources dont l'usage est prévu par le scénario.

I.2.5 Outillage de l'approche EML suivant les phases du cycle de vie

La mise en œuvre de l'approche EML permet de mettre en place des activités d'apprentissage permettant de dérouler la situation d'apprentissage décrite par le scénario pédagogique, au sein d'un Environnement Numérique de Travail (ENT). Le succès de cette mise œuvre dépend de l'ingénierie existante et donc des outils et de la démarche qu'elle propose (cf. Section I.2.4). Les outils permettent d'instrumenter le cycle de vie du scénario pédagogique (cf. Figure I.5).

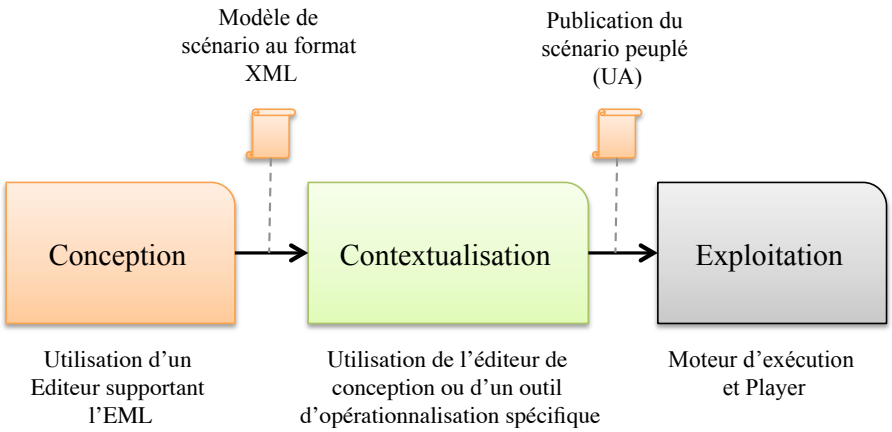

Figure I.5 : Outils utilisés dans le cycle de vie d'un scénario pédagogique

I.2.5.1 Editeurs pour la conception des scénarios

L'*éditeur* fournit une interface permettant de couvrir les taches nécessaires à la modélisation du scénario pédagogique. Il cadre l'activité de conception pour garantir la conformité du modèle produit par rapport à l'EML utilisé. La modélisation ne se limite pas à la création (instanciation d'éléments à partir du méta-modèle de l'EML utilisé), elle s'étend aussi à la suppression et à la modification d'éléments dans le scénario.

Certains éditeurs proposent en plus la possibilité de publier le modèle de scénario pour son opérationnalisation. C'est le cas de *Model Editor* qui est un éditeur de scénarios LDL. Le scénario produit est abstrait. Il est destiné à être opérationnalisé dans une infrastructure cible, supportant le langage LDL.

D'autres éditeurs comme *Reload Learning Design Editor* ou *Reload LDE* [MBS05] qui permet de modéliser des unités d'apprentissage conformes à l'EML IMS-LD, proposent des fonctionnalités allant jusqu'à produire un scénario peuplé (ou unité d'apprentissage) pouvant être publiées directement dans l'environnement d'exécution *CopperCore LD*. L'implémentation de *Reload LDE* se base sur la spécification proposée dans [KT05]. La figure I.6 présente le diagramme des cas d'utilisation d'un tel éditeur.

Comme on peut le constater sur le diagramme de cas d'utilisation illustré par la figure I.6, l'éditeur devrait permettre de créer le scénario (à travers le rôle *spécialiste de l'éducation*), de le peupler (à travers le rôle *enseignant*), de gérer et d'affecter des rôles, de gérer des activités, etc.

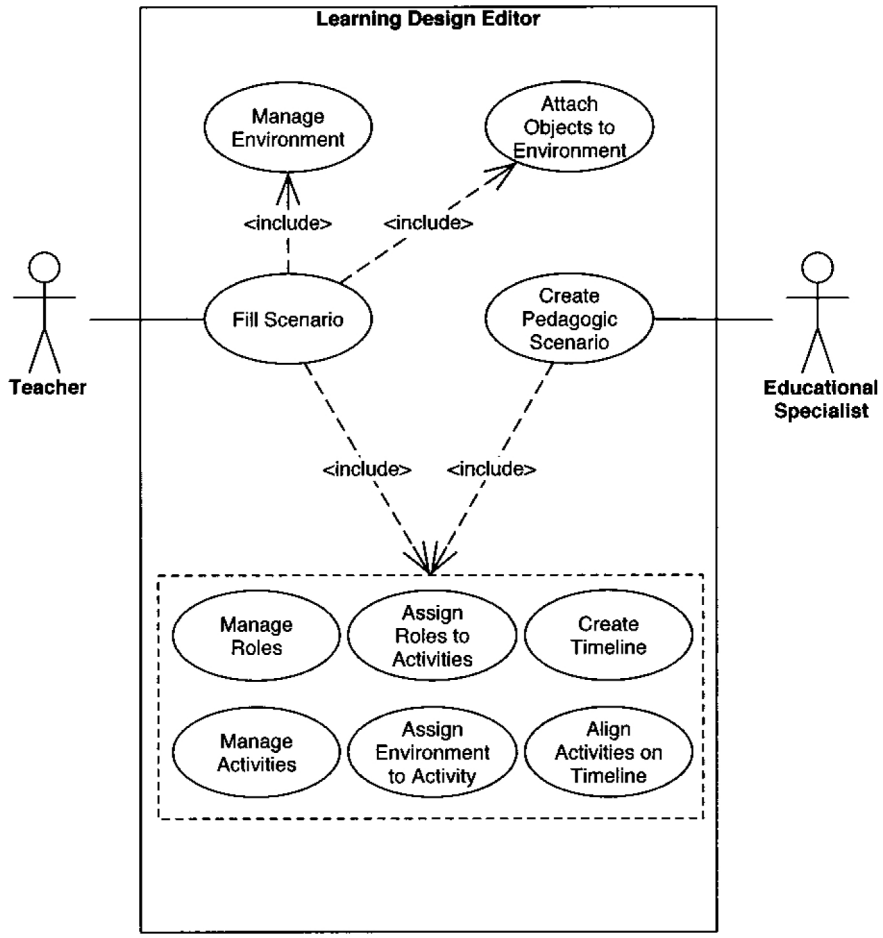

Figure I.6 : Diagramme des cas d'utilisation d'un éditeur LD. Source [KT05]

La figure I.7 positionne les éditeurs *Model Editor* et *Reload LDE* dans le cycle de vie du scénario. On peut par conséquent constater la différence en termes de phases couvertes par les deux éditeurs.

I.2.5.2 Le player

Le Player est le programme qui interprète l'UA. Il présente les différentes activités et ressources pour les rôles impliqués et contrôle leurs interactions [ZDFPD07]. Il permet donc de distribuer et de présenter les activités, les ressources et les services spécifiés dans l'UA, aux participants de l'activité en fonction des rôles qui leur ont été attribués. L'implémentation du Player dépend de l'EML et du méta-modèle sous-jacent. *CopperCore* [VM06] est un exemple de Player développé par l'Educational Technology Expertise Center de l'OUNL. Il a été conçu et implémenté pour supporter LD. Les fonctionnalités principales de *CopperCore* sont les suivantes :

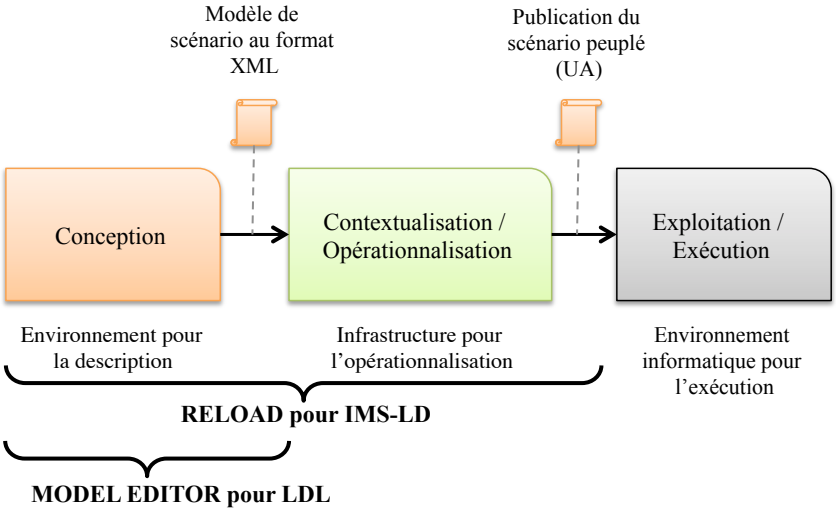

Figure I.7 : Positionnement de *Model Editor* et de *Relaod LDE* dans le cycle de vie du scénario

- validation technique et sémantique de l'UA pour s'assurer de sa conformité par rapport à la spécification de l'EML et notamment de son méta-modèle ;
- gestion des accès et des privilèges relatifs aux rôles impliqués dans l'UA avant l'utilisation des services et des ressources qu'il propose pour l'activité ;
- interprétation de l'UA et mise à disposition personnalisée des contenus en respectant les règles définies dans le scénario.

I.3 Classification des EML

[BDBF06] propose une classification et un cadre de comparaison des langages de modélisation pédagogiques. La classification proposée se base sur les caractéristiques suivantes :

- la *stratification* : la richesse et expressivité du langage
- la *formalisation* : caractère formel du langage, règles, rigueur d'utilisation et composition des concepts du langage pour la modélisation ;
- l'*élaboration* : niveaux de détails de description dans le langage ;
- la *perspective* : un seul langage en couche, proposant plusieurs sous langages pour décrire les mêmes entités, mais en partant de points de vue différents ;
- la *notation* : présence ou pas d'un système de notation.

Ces caractéristiques permettent de démarquer ces langages et de montrer que chacun d'entre eux possède une approche différente dans la formalisation d'une situation d'apprentissage. Voici

quelques exemples :

- E^2EML [Bot06] : un simple langage de modélisation proposant un système de notation visuelle qui consiste en un ensemble de diagrammes interdépendants. Il a surtout été conçu pour permettre aux spécialistes de l'éducation et aux concepteurs de scénarios pédagogiques de communiquer plus efficacement entre eux dans les projets *E-Learning* importants ;
- PCeL pattern initiative [Der05] : définit un profil UML pour modéliser des scénarios pédagogiques. Son utilisation réside principalement dans le contexte de l'apprentissage mixte, où une vue claire sur les éléments en face à face dans la conception du scénario est essentielle dans l'utilisation du support Web ;
- PoEML (Perspective-oriented Educational Modeling Language) [CRLNAR06] : peut-être utilisé pour modéliser des scénarios pédagogiques avec différents niveaux d'agrégation, il offre aussi des patterns de modélisation pour les perspectives qu'il fournit (aspects sociaux, organisationnels, temporels, etc.).
- LDL (Learning Design Language) : c'est un langage de modélisation pédagogique qui a été conçu pour modéliser des activités collaboratives. Il puise ses racines dans les sciences sociales, principalement la linguistique, la sociologie et l'ethnométhodologie. Il propose sept concepts qui permettent aux concepteurs pédagogiques de construire le modèle d'une activité d'apprentissage collaboratif. Il a à la fois un une représentation visuelle et une notation textuelle, cette dernière étant lisible par ordinateur [MVF+06].

Le langage LDL est présenté en détails dans la section suivante.

I.4 L'EML LDL et l'infrastructure LDI associée

LDL (Learning Design Language) est un langage de modélisation pédagogique qui a été conçu pour décrire des activités d'apprentissage coopératives hétérogènes dans lesquelles plusieurs dimensions de l'activité sont prises en compte : l'apprentissage, l'organisation et le suivi, l'entraînement et l'évaluation [MVFD06].

I.4.0.3 Le méta-modèle LDL

La figure I.8 représente le méta-modèle conceptuel de l'EML LDL. Comme nous pouvons le constater sur cette figure, le concept scénario est le noeud principal. Tout scénario LDL doit donc comporter une instance de ce dernier.

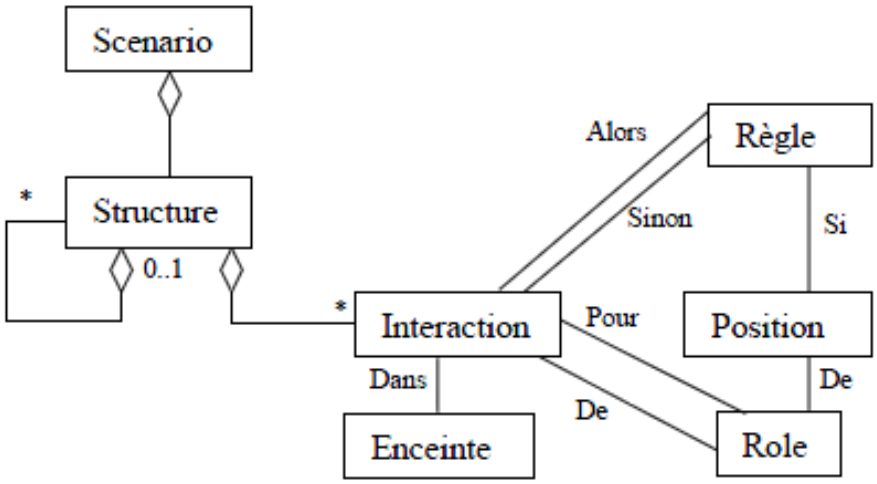

Figure I.8 : Méta-modèle de l'EML LDL

Le concept *scénario*

Comme illustré dans la figure I.8, le concept de scénario représente la *racine* du méta-modèle LDL. Tout modèle de scénario LDL doit présenter au moins une instance du concept "scénario".

Le concept *Structure*

Dans LDL, une étape dans l'activité est spécifiée par le concept de structure. L'organisation des structures permet de décrire l'enchainement des phases. Une structure peut référencer de manière récursive des structures ou des interactions au sein de l'activité. LDL propose trois types de structures qui permettent d'organiser les phases de manière :

- **séquentielle :** les interactions des participants doivent être exécutées en séquentiel. Exemple : lire l'énoncé d'un exercice, répondre à des questions sur l'exercice et voir la solution ;
- **sélective :** dans une structure sélective les participants sont maîtres du choix des interactions qui la composent ;
- **parallèle :** toutes les interactions peuvent être jouées sans ordre prédéfini.

Le concept *Interaction*

Les interactions s'insèrent dans les structures précédemment décrites. Elles spécifient au sein du scénario les différents échanges qui auront lieu entre les participants [MVFD06]. Une interaction est une action orientée d'un participant vers un autre. L'initiateur de l'action est le *destinateur* et le receveur le *destinataire*. L'interaction est un concept clé dans une activité modélisée par un scénario LDL, les échanges et les collaborations sont construits essentiellement sur ce concept. Les interactions sont dites situées, c'est-à-dire qu'elles portent sur des contenus (document, page Web, vidéo, image) ou se produisent via des services particuliers (forum, chat, wiki, courrier électronique, etc.). L'interaction (comme la structure) peut être :

- *visible* ou pas. Quand elle est visible, elle est automatiquement présentée au participant qui dispose du rôle adéquat. Ce dernier peut par conséquent la démarrer ;
- *démarrée* ou non. Quand elle est démarrée, l'interaction est en cours d'exécution ;
- *terminée* ou pas. Quand elle est terminée, l'interaction devient automatiquement invisible pour le participant.

Le concept *Rôle*

Les interactions renvoient thématiquement à un rôle *élève* ou *enseignant* (exemples de rôles dans une activité d'apprentissage). Dans LDL, le rôle se définit par l'ensemble des interactions spécifiques qu'il regroupe dans le modèle d'une activité [MVFD06]. Un participant peut jouer plusieurs rôles dans une même activité. L'enseignant peut par exemple avoir en même temps le rôle de tuteur et celui d'évaluateur. Il pourra ou devra intervenir dans la limite des interactions qui définissent ses rôles dans l'activité.

Le concept *Enceinte*

Les enceintes d'une activité spécifient l'endroit où se dérouleront les interactions des participants. Une enceinte peut être de type *contenu* ou *service*. Un forum, un moteur de recherche, un chat, un instrument de mesure ou une encyclopédie coopérative (du type de Wikipédia) seront spécifiés dans le scénario comme des enceintes de type *service*. Un cours, un devoir, un exercice, un album d'images ou un site Web seront spécifiés comme des enceintes de type *contenu*.

Le concept *Règle*

Dans LDL, ce sont les règles qui régissent le démarrage et la fin des activités, l'ouverture ou la clôture des interactions, le début ou la fin des structures [MVFD06]. Le démarrage ou la fin

des interactions dépendent des réponses apportées par les participants.

Le concept *Position*

Le concept de position permet de savoir ce qui se passe lors des interactions entre les participants au sein de l'activité. La position représente le résultat des réactions d'un participant, comme son point de vue, sa réponse ou encore sa présence dans une enceinte. Les valeurs des positions sont testées par les règles qui elles mêmes testent des conditions. Les résultats des tests effectués sur ses conditions déterminent quelles interactions, structures, contenus ou services doivent être présentés aux participants.

Les positions peuvent être *déclarées* ou *observées*. Une position est déclarée quand elle est apportée intentionnellement par le participant qui déclenche une interaction. Elle est dite observée quand elle est au contraire collectée par le système de manière implicite.

Une position LDL a des attributs qui définissent ses propriétés. La *portée* permet par exemple de déterminer la visibilité d'une position. Grâce à cette propriété il devient possible de limiter la visibilité de la position :

- pour les participants d'une seule et unique activité ;
- pour tous les participants à des activités issues du même scénario ;
- pour tous les participants à des activités issues de différents scénarios.

La *partition* permet de spécifier si la position est individuelle ou collective. Quand elle est individuelle, la position appartient à un seul participant et n'influe que l'activité de ce dernier. Au contraire, lorsqu'elle est collective, la position peut influencer les activités d'un groupe de participants. La position de chaque participant est alors valable pour tout le groupe.

Le concept *Observable*

La notion d'observable répond à la nécessité de disposer de points d'observation sur l'activité et les différents éléments qui la composent [MVFD06]. Tous les concepts LDL sont par principe observables. Autrement dit, tous les éléments modélisés dans un scénario peuvent être observés et notamment les propriétés qui les caractérisent. Par exemple, la valeur d'une position (réponse, choix, etc.) ou l'état d'une interaction (structure).

Exemple : modélisation d'un scénario LDL

La démarche de conception d'un scénario LDL est la suivante :

- l'identification et la construction de sa structure interactionnelle, c'est à dire de la manière dont les échanges vont s'organiser et se dérouler ;
- la définition des différents rôles qui prendront part au sein de l'activité ;
- la définition des enceintes qui sont les lieux dans lesquels va se dérouler l'activité ;
- la définition des règles auxquelles les participants vont se conformer ;
- la définition des positions des participants, c'est à dire des différents points de vue qu'ils auront à exprimer au cours de l'activité.

Prenant un petit exemple de situation d'apprentissage dans laquelle des apprenants doivent répondre à des Questionnaire à Choix Multiples (QCM). Ces apprenants passeront d'un QCM à un autre en fonction de leurs réponses. Voyons les étapes de modélisation du scénario de cette situation à travers l'instanciation du méta-modèle LDL :

- instancier le concept scénario pour représenter l'activité d'apprentissage. Celui-ci référencera les autres éléments ;
- instancier une structure LDL pour représenter la phase durant laquelle s'enchaineront les différentes interactions des apprenants. Il faut aussi définir cette structure comme *sélective*, car l'ordre d'exécution des interactions (réponse aux questions) doit dépendre des réponses données par les apprenants ;
- instancier un rôle LDL *Apprenant* ;
- instancier une enceinte LDL pour chaque question (QCM) (Question 1, Question 2, Question 3, etc.). Celles-ci accueilleront les différents QCM ;
- instancier des positions LDL afin de représenter les réponses que donneront les apprenants (Réponse 1, Réponse 2, Réponse 3, etc.). Il faut également choisir l'enceinte dans laquelle sera déclarée ou observée la valeur de la position. Par exemple, la position *réponse 1* sera déclarée dans l'enceinte *Question1* ;
- instancier une règle LDL pour conditionner le passage d'un QCM à un autre en fonction de la réponse (valeur de la position) choisie par l'apprenant. La règle évalue une condition dans laquelle elle compare la valeur de la position à la réponse juste du QCM. La règle peut être *vraie* ou *fausse* selon la réponse choisie par l'apprenant ;
- instancier une interaction pour chaque question et cibler le rôle *Apprenant* en tant que *Destinateur*. Préciser également :

- l'enceinte dans laquelle se déroulera l'interaction. Par exemple, l'interaction *Répondre à la Question 1* dans l'enceinte *Question 1*,
- la position qui sera déclarée par l'apprenant quand il aura effectué l'interaction. Par exemple, la position *Réponse 1* pour l'interaction *Répondre à la Question 1*,
- les règles de démarrage et de terminaison de l'interaction. Par exemple, pour démarrer l'interaction *Répondre à la Question 2*, il faut que la règle qui évalue la position *Réponse 1* soit vraie. L'interaction *Répondre à la Question 1* sera terminée si sa règle de terminaison est vraie. Cette dernière sera alors vraie si *Position 1* contient une réponse.

Nous compléterons et détaillerons ce scénario dans l'exemple fil rouge que nous présenterons dans le chapitre II.

Personnalisation à l'aide du *trio* concepts position, règle et observable

L'observation, les positions et les règles sont liées. Ensemble, elles permettent de contrôler la présentation des interactions et des structures aux participants. Ce sont des concepts importants pour la personnalisation et l'adaptation dynamique de l'activité. En effet, les positions sont d'abord capturées grâce au concept d'observable (pour chaque participant). Une fois capturées, ces positions sont évaluées par les règles. Les valeurs obtenues (pour chaque participant) qui peuvent être vraies ou fausses permettent d'influencer la visibilité des interactions et des structures. Dans la mesure où le scénario le prévoit, ce mécanisme permet donc de personnaliser les parcours des participants ou les contenus pédagogiques mis à leur disposition en fonction des positions qu'ils ont prises.

I.4.1 LDI : une infrastructure pour l'opérationnalisation et l'exécution de scénarios LDL

LDI (*Learning Design Infrastructure*) est une infrastructure conçue pour permettre l'opérationnalisation et l'exécution des scénarios LDL. Elle possède une base de données pour la sauvegarde des scénarios LDL, une sorte de référentiel pour les scénarios. Tout scénario LDL opérationnalisé dans LDI y est alors sauvegardé. Après leur importation, il est possible de les consulter au format XML, et de les partager avec d'autres utilisateurs qui ont accès à LDI.

I.4.1.1 Architecture de l'infrastructure LDI

D'un point de vue technique, *LDI* a été développé en utilisant les technologies Zope [Zop11] et Plone [Plo11]. Zope est un serveur d'application web orienté objet, libre et écrit dans le langage de programmation Python. Plone est un module zope qui fournit un environnement de gestion et de contenus (documents, e-mail, messages textuels, etc.) en ligne. Il est facilement extensible et adapté au développement rapide d'applications Web. Il constitue donc un outil flexible, puissant et idéal pour développer des solutions de partage de contenu sur Internet et LDI en est un exemple.

Concept de produit dans *Zope*

Le produit est un concept clé qui apporte de la modularité à *ZOPE*. En effet, dans *Zope*, tout est intégré sous forme de produit. Les produits sont alors enregistrés dans une base de données appelée Zope Object Database (ZODB). L'infrastructure LDI a été ainsi implantée sous forme de produit ZOPE.

Produit LDI

LDI est un produit qui vient s'intégrer à Plone et à Zope. Ceci permet au produit LDI de bénéficier des services de gestion de contenus déjà existants dans Plone. Comme le montre la figure I.9, le produit LDI est aussi composé de sous-produits, chacun apportant une fonction spécifique. Par exemple, le moteur d'exécution est un produit qui est responsable de l'exécution des scénarios LDL.

Principe de fonctionnement du moteur d'exécution

Le produit *moteur d'exécution* ou *LD Engine* exécute des requêtes sur la Zope Object Data Base pour récupérer les valeurs des règles *LDL* et ainsi déterminer la visibilité de chaque interaction pour tous les participants. Quand une interaction est visible, elle est présentée au participant via son Player. Le Player est une interface graphique qui permet au participant de voir et d'exécuter les interactions du scénario dans la limite de son rôle, de l'exécution du scénario et de son activité ou l'activité des autres participants (collaboration). Chaque participant a son propre Player. La communication entre le moteur d'exécution (LDI) et chaque *Player* s'effectue dans une configuration client serveur. Les *Player* des participants sont des clients et LDI le serveur qui reçoit et exécute leurs requêtes (cf. Figure I.10). Nous donnerons plus de détails et d'exemples sur la visibilité dans le chapitre V.

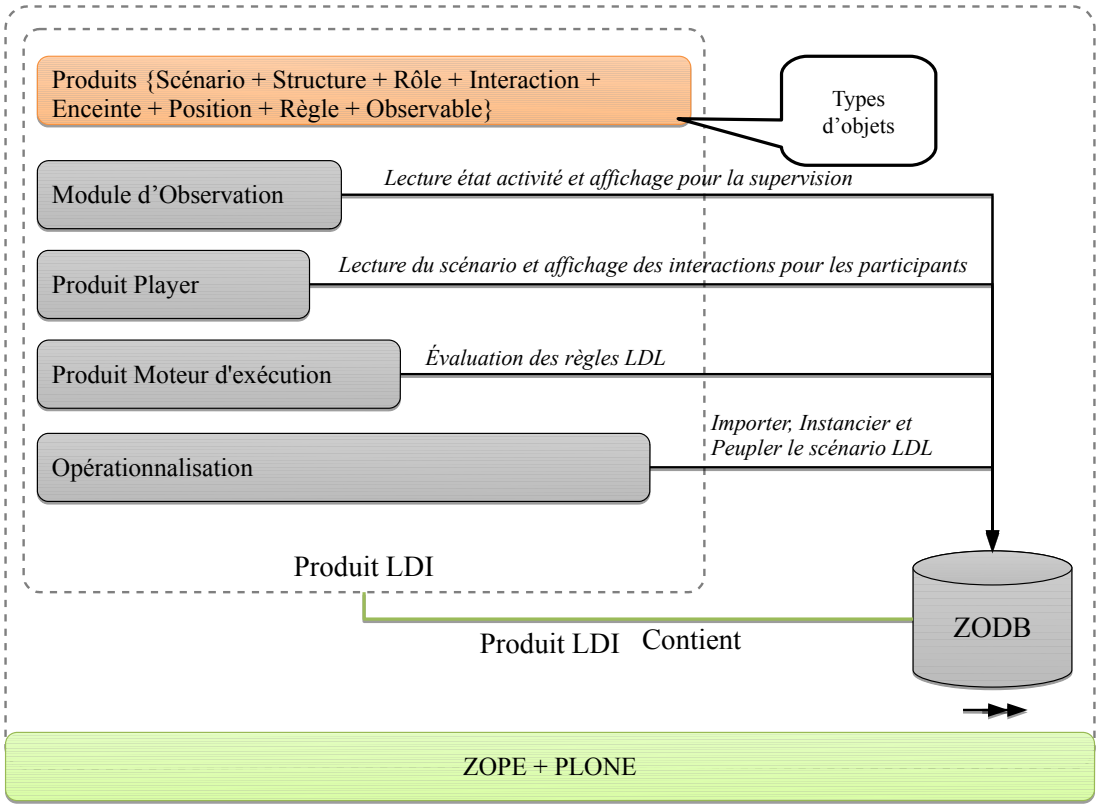

Figure I.9 : Architecture de l'infrastructure *LDI*

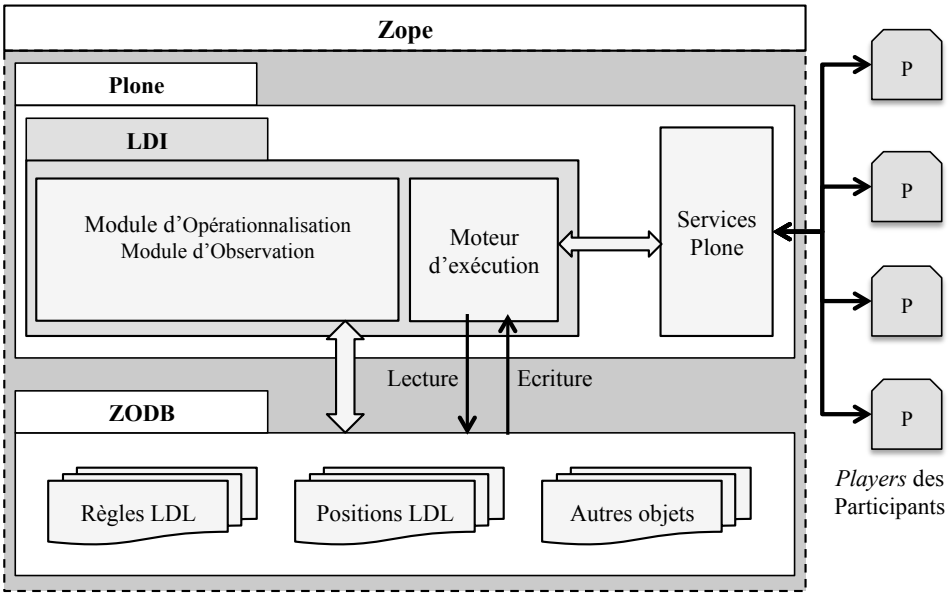

Figure I.10 : Intégration du moteur d'exécution dans *LDI*

I.4.1.2 Gestion de l'exécution des scénarios dans LDI

LDI fournit certaines fonctionnalités permettant de gérer les scénarios en cours d'exécution. Il s'agit d'opérations de base permettant par exemple le démarrage, la suspension, l'arrêt ou la suppression de l'exécution des scénarios. La gestion inclut aussi la sauvegarde des différents états par lesquels passe l'activité pendant l'exécution de son scénario. Cette fonctionnalité permet de naviguer dans les différents points de sauvegarde pour revenir à un point antérieur si une erreur survenait pendant l'exécution. Par ailleurs, c'est cette fonctionnalité qui pourra être exploitée pour restaurer l'état de l'activité lors d'une adaptation nécessitée par l'observation d'un événement imprévu.

I.4.1.3 L'observation dans LDI

L'observation de l'activité dans LDI est possible grâce au concept d'observable. Tous les objets (ou leurs propriétés) peuvent être observés. Ceci est très important pour la régulation de l'activité. Il est donc possible de voir via l'interface de *LDI*, chaque événement lié aux interactions des participants dans l'activité. Ces événements constituent des traces d'exécution des interactions. Elles permettent par exemple de savoir qui fait quelles interactions et à quelles dates elles sont démarrées, visibles ou terminées. Toutes ces informations réunies représentent l'état de l'activité I.11.

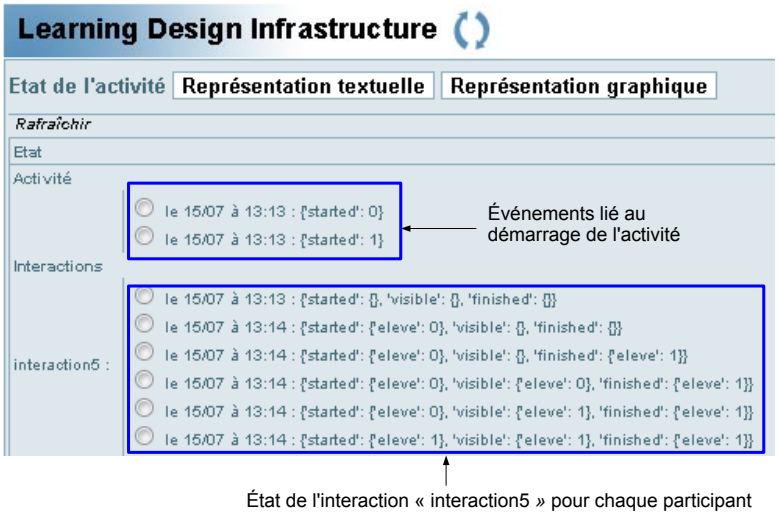

Figure I.11 : Observation de l'état de l'activité dans LDI

Grâce au mécanisme présenté ci-dessus, LDI offre certaines possibilités de personnalisation

permettant à l'activité de s'adapter aux réactions des participants, et aussi d'influencer l'exécution des scénarios. Toutefois, ses facultés d'adaptation restent malgré tout limitées. Nous en discutons dans la section suivante.

I.4.1.4 Quelques possibilités d'adaptation dans LDI

L'observation permet d'avoir un retour sur les réactions des participants dans l'activité. Elle permet par exemple de voir l'avancement de chaque apprenant, les positions déclarées et leurs valeurs. Si à travers les observations produites par LDI, l'enseignant ou le tuteur qui supervise l'activité détecte une situation imprévue qui bloque l'exécution du scénario, il devrait être possible d'intervenir afin d'y remédier. LDI fournit pour cela quelques possibilités de modifications via l'interface d'opérationnalisation. Il est ainsi possible de :

- configurer les rôles à la volée (ajouter ou supprimer des participants) pendant l'exécution du scénario (cf. Figure I.12) ;
- configurer les URL qui référencent les ressources et les services requis pour l'exécution du scénario. Par exemple, référencer un autre document que celui déjà choisi (cf. Figure I.13).

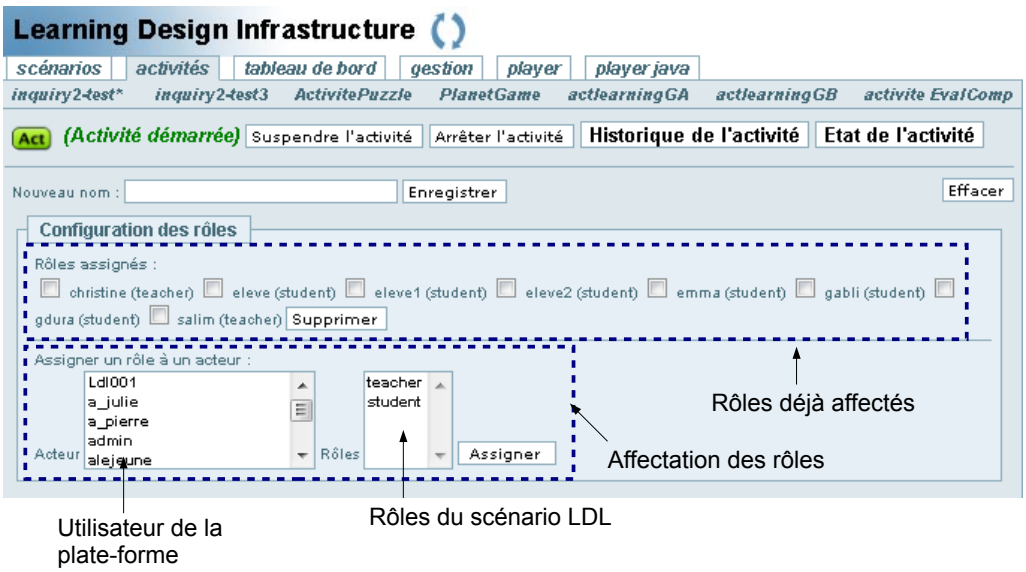

Figure I.12 : Configuration des rôles dans LDI

Nous constatons que les possibilités d'adaptation offertes par LDI sont :

- limitées, car ne couvrant pas toutes les modifications qu'il pourrait être nécessaire d'apporter à une activité en cours d'exécution, notamment lors de l'apparition de situations imprévues. Par exemple, il est impossible d'apporter des modifications au scénario ou

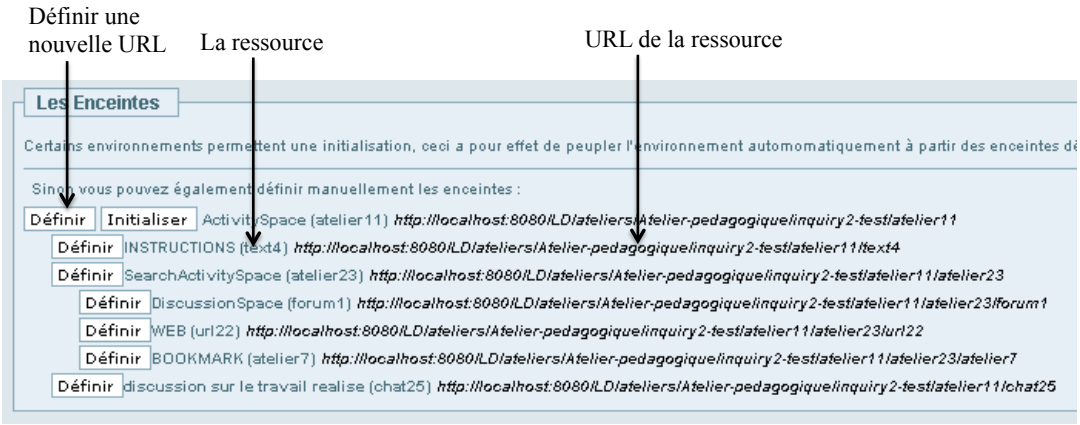

Figure I.13 : Configuration des enceintes dans LDI

encore de modifier le parcours d'un participant pour dévier celui-ci de ce qui est prévu dans le scénario ;

– insuffisantes, car n'offrant pas de support pour cadrer les adaptations et s'assurer de leurs succès. Il est possible de faire certaines modifications, mais sans en connaitre les conséquences.

Ces limitations sont au cœur de notre problématique qui fait l'objet du chapitre suivant.

I.5 Conclusion

Nous avons présenté dans ce chapitre le contexte de notre recherche. Nous avons introduit l'approche EML, le langage LDL et l'infrastructure LDI qui représentent notre terrain d'expérimentation et la base sur laquelle nous nous sommes appuyés dans nos travaux.

Dans le chapitre suivant, nous présenterons notre problématique de recherche. Nous allons par conséquent nous intéresser aux situations d'apprentissage dans le contexte d'une approche EML. Il s'agit d'étudier les verrous qui empêchent aujourd'hui l'adaptation des activités d'apprentissage mises en place en utilisant l'approche EML.

Chapitre II

Définition et étude de la problématique

Ce chapitre est organisé comme suit :

- Dans la section II.1, nous nous intéressons à l'imprévisibilité de l'activité du point de vue des sciences sociales, de la théorie de l'activité et des sciences de l'éducation ;
- La section II.2 présentera le cas d'une expérimentation d'un scénario modélisé conformément au langage LDL, au cours de laquelle s'est révélée la nécessité de disposer de mécanismes d'adaptation en cours d'exécution ;
- Dans la section II.3, nous étudierons différentes situations pouvant survenir lors de l'exécution d'un scénario et en empêcher le déroulement prévu. Ces situations ont fait l'objet d'une confrontation entre des chercheurs internationaux dans le domaine du Learning Design au cours d'un Workshop auquel nous avons participé pendant la conférence CSCL 2009. Nous nous appuierons sur ces différentes situations pour illustrer nos propositions dans la suite de ce mémoire ;
- Dans la section II.4, nous situerons plus précisément notre problématique en regard des différents types d'adaptation d'une activité d'apprentissage résultant de l'exécution d'un scénario modélisé avec un EML ;
- Nous présenterons dans la section II.5 une première proposition de caractérisation des situations imprévues et de leur adaptation à la volée ;
- Dans la section II.6 nous présenterons le cadre dans lequel nous avons étudié la problématique d'adaptation à la volée et les conditions que nous avons considérées dans les travaux que nous avons réalisés ;
- La dernière section résumera les questions abordées dans ce chapitre.

II.1 Imprévisibilité d'une activité

Nous considérons que l'imprévisibilité de l'activité est au premier plan responsable de la nécessité de pouvoir en ajuster le déroulement. Nous allons nous intéresser à cette imprévisibilité en considérant les points de vue suivants :

- les sciences sociales ;
- la Théorie de l'Activité (TA) ;
- les sciences de l'éducation.

II.1.1 Du point de vue des sciences sociales

Dans [Suc94], *L. Suchman* propose une étude dans laquelle elle soutient la thèse que l'interaction est un phénomène anthropologique où l'action humaine est constamment construite et reconstruite en fonction d'interactions dynamiques avec les mondes matériel et social. Autrement dit, l'action ne dépend pas uniquement de son plan (préalablement prescrit) mais aussi des conditions dans lesquelles elle est située. Ainsi, écrit Suchman dans [Suc87] : "j'introduis l'expression *action située* pour souligner que tout cours d'action dépend de façon essentielle de ses circonstances matérielles et sociales" (écriture traduite en français dans [BC04]). L'agent aura beau tout planifier, envisager les *alternatives* entre lesquelles choisir à chaque étape (dans le plan), l'accomplissement de l'action ne pourra être la simple exécution d'un plan. Il faudra s'ajuster aux circonstances, traiter les contingences, agir au bon moment en saisissant les occasions favorables [RST04].

II.1.2 Du point de vue de la Théorie de l'Activité

La Théorie de l'Activité (TA) a été introduite au début du siècle par *A. Leont'ev* et *Vygotsky*, deux théoriciens de l'école historique et culturelle soviétique de la psychologie. La TA a permis à Engeström en 1987 de proposer une structure de base (cf. Figure II.1) pour modéliser des activités collectives [Eng87].

Niveaux d'une activité

A. Leont'ev a décrit l'activité comme étant décomposable en trois niveaux hiérarchiques (cf. Figure II.2) : l'activité (plus haut niveau, elle a un motif) qui se décompose en actions (niveau intermédiaire lié à un but) qui elles mêmes se décomposent en opérations (procédures compilées conditionnées) [KC02]. Il ajoute que ces niveaux ne sont pas statiques et qu'ils peuvent par conséquent se transformer. Pour expliquer cette transformation, [Bou00] propose l'exemple sui-

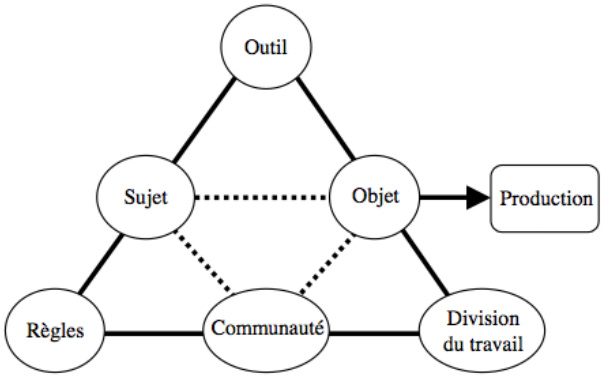

Figure II.1 : La structure basique d'une activité. Source [Bou00]

vant : "lorsque l'on descend un escalier dont les marches sont trop serrées, *l'opération* de descente doit revenir au niveau d'une *action* consciente, demandant plus d'attention, et une exécution plus lente". Cet exemple montre que face à un imprévu, l'opération qui consiste à descendre les marches de façon automatique (compilée et maitrisée), doit s'adapter et devenir une *action* réfléchie.

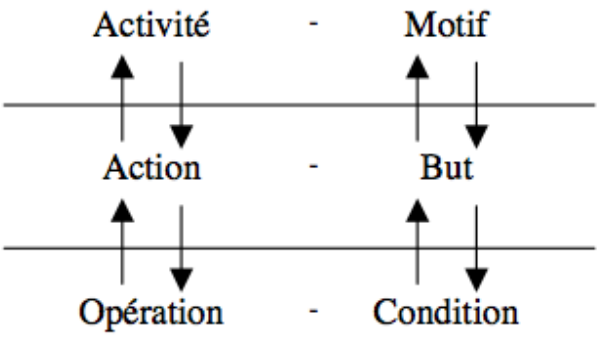

Figure II.2 : Les niveaux hiérarchiques d'une activité. Source [Bou00]

Expansivité de Kuutti

[Kuu91, Kuu96] considère l'imprévisibilité d'une activité en prenant en compte son influence sur le support de l'activité du point de vue des technologies de l'information. Il propose donc une classification (cf. Figure II.3) dans laquelle il montre l'évolution des types basiques de support de l'activité (outil, règle, division du travail, etc. cf. Figure II.1) par rapport aux états (passif, actif et expansif) de l'individu dans l'activité. Nous remarquons par exemple que dans l'état expansif (troisième ligne du tableau représenté figure II.1), le sujet (première colonne du tableau de la

figure II.1), fait évoluer l'outil en créant de nouveaux automatismes. Le sujet s'adapte et adapte l'outil qu'il utilise lorsqu'il fait face à une situation qu'il n'a jamais rencontrée (imprévu).

	Outil	Règles	Division du travail	Sujet «pensant»	Objet	Communauté
Passif	Automatisation de routine	Contrôle implicite et imposé	Fixée et imposée	Déclenche des actions prédéterminées	Données	Implicite et invisible
Actif	Support aux actions de transformation et de manipulation. Rendre les outils et procédures visibles	Pensées partagées et visibles	Coordination, Organisation visible du travail	Recherche l'information	Matériel partagé	Réseau visible Support à la communication
Expansif	Automatisation De nouvelles routines ou construction d'outil	Construction, négociation des règles	Réorganisation	Apprend, comprend	Construction de l'objet	Construction de la, ou d'une nouvelle communauté

Figure II.3 : Une classification des types basiques de support au travail. (Source [Bou00])

II.1.3 Du point de vue des sciences de l'éducation

Selon [Alt02] la pratique enseignante peut être définie comme la manière de faire singulière d'une personne, sa façon réelle, propre, d'exécuter une activité professionnelle : l'enseignement. Si l'enseignement a la particularité d'être singulier, l'apprentissage ou la situation d'apprentissage et tout ce qui en résulte comme notamment les interactions des apprenants, le sont aussi. En effet :

- selon [TL99], ce sont les trames interactionnelles entre l'enseignant et les apprenants qui commandent la logique des *événements* en classe ;
- les événements en classe sont caractérisés [Doy86] par par la multiplicité, la simultanéité, l'immédiateté, l'*imprévisibilité* et l'*historicité* ;

Par ailleurs, selon [Alt02], même si les routines quotidiennes et les règles de fonctionnement de la classe incluent ces trames d'interactions avec les apprenants et si le déroulement du cours en classe est planifié (scénario préétabli), il n'en demeure pas moins que l'enseignant doit s'adapter en permanence aux situations toujours singulières et imprévisibles.

Impact de l'imprévisibilité de l'activité sur l'exécution d'un scénario pédagogique : Contrairement à une *situation traditionnelle* où l'enseignant peut réagir face à une situation imprévue et adapter son plan de cours en fonction, le déroulement des situations d'apprentissage

instrumentées (utilisant l'approche EML) est contraint par le modèle de scénario. Ces situations sont par conséquent sensibles à l'imprévu. Selon [SM95] une *situation imprévue* est un cas particulier qui ne peut être traité correctement par le système sans l'intervention manuelle d'un agent humain (enseignant, tuteur, informaticien, etc.). Dans la suite de ce mémoire, il faudra prendre sous ce sens l'expression *situation imprévue*. Nous donnons un exemple de situation imprévue dans la section suivante.

II.2 Leçons apprises : l'expérimentation du scénario *LearnElec*

Dans le contexte d'une situation d'apprentissage utilisant une approche EML, nous nous appuyons sur une expérimentation qui a eu lieu dans le cadre de l'initiative MATES (Methodology And Tools for Experimentation Scenario), pour présenter un exemple de situation imprévue et bloquante constatée lors de l'exécution du scénario *LearnElec*.

II.2.1 Le scénario *LearnElec*

LearnElec concerne l'apprentissage de notions élémentaires d'électricité sur les circuits en série (loi d'Ohm, additivité des tensions, unicité de l'intensité). Il a été expérimenté au printemps 2007 dans des classes de troisième de la région Rhône Alpes. Ce scénario alterne des étapes de travail individuel et de débat en groupe supervisé par un enseignant-tuteur. Il est constitué de différentes phases généralement construites selon le schéma présenté sur la figure II.4.

Le passage d'une étape à une autre (ex : passage de Travail individuel à Débat) est soumis à l'évaluation de règles modélisées en LDL sous la forme $si < condition > alors < action >$ qui s'expliquent comme suit :

- condition : *tous les élèves ont soumis leur travail individuel*. Dans le cas de LDL, il s'agit de la déclaration d'une position ;
- action : permettre à l'élève d'aller à l'étape suivante. Dans le cas de LDL cela consiste à rendre visibles les interactions de l'étape suivante (débat pour chaque groupe et suivi du débat pour les tuteurs).

II.2.2 Situation bloquante

Le débat dans chaque groupe ne peut se faire que quand tous ses membres ont soumis leur travail individuel. Lors de l'expérimentation certains groupes ont été grandement pénalisés (et n'ont pas pu terminer dans les temps l'activité prévue) suite au blocage engendré par l'absence

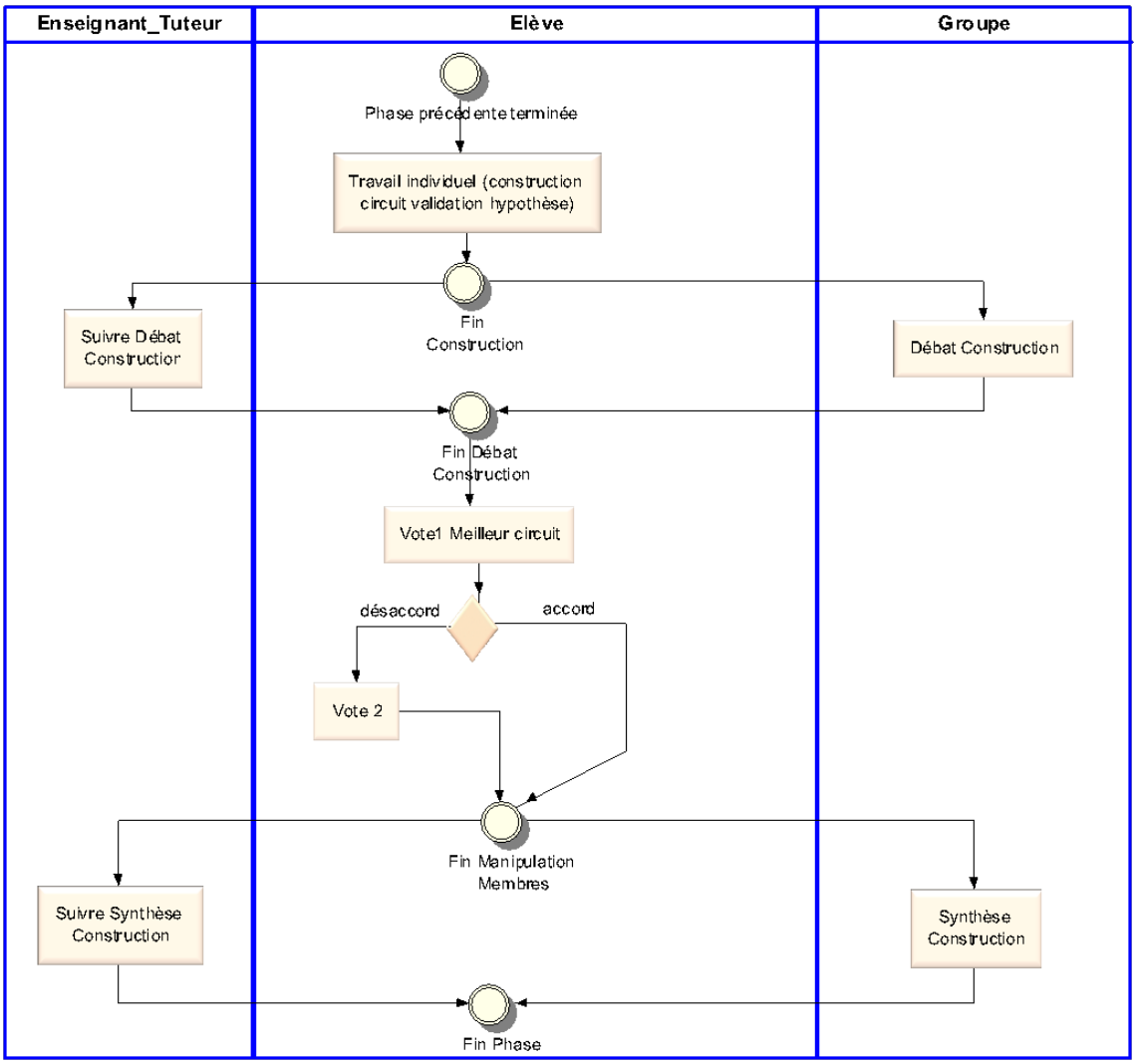

Figure II.4 : Pattern de déroulement des phases du scénario LearnElec

de soumission du travail d'un ou plusieurs de leurs membres. Pour pallier ce problème, il aurait fallu pouvoir intervenir à la volée, par exemple en prenant la main sur le travail des élèves en retard (n'ayant pas soumis leur travail) et effectuer le rendu en l'état. Cela aurait ainsi permis de débloquer la situation.

Nous présentons dans la section suivante l'exemple fil rouge que nous allons conserver tout au long de ce mémoire. Cet exemple qui s'inspire de l'exemple du scénario LearElec présenté ci-dessus, a été proposé comme étude de cas lors d'un workshop sur le thème de l'adaptation dans le Learning Design (LD : approche EML) organisé lors de la conférence CSCL 2009, auquel nous avons participé.

II.3 Exemple fil rouge

L'objectif majeur du Workshop Adapting Activities Modeled by CSCL Scripts (AAMCS) était de comparer les différentes approches d'adaptation proposées dans le Learning Design et, de manière générale, dans le CSCL.

II.3.1 L'étude de cas proposée dans le workshop AAMCS

Le scénario pédagogique proposé comme étude de cas pour le workshop AAMCS a été davantage conçu pour mettre en évidence des situations particulières qui pourraient en perturber le déroulement que pour sa pertinence pédagogique. Il a servi de *benchmark* de référence pour la comparaison des propositions et solutions apportées par les différents participants du *Workshop*.

Scénario de l'étude de cas

Le scénario pédagogique implique deux rôles *learner* et *teacher*. L'activité associée au rôle *learner* est composée de trois étapes qui alternent entre le travail collaboratif et individuel :

1. **Choix d'un thème :** Les apprenants se concertent pour le choix d'un thème en relation avec l'environnement. Les thèmes proposés dans le scénario sont : l'économie d'énergie, la biodiversité, le traitement des déchets et les transports en commun. Le scénario doit de fait permettre un choix collectif ;

2. **Réponses aux questions :** Dans cette étape, chaque apprenant évolue individuellement en répondant à des QCM relatifs au thème choisi dans la première étape. Dans cette étape les réponses des apprenants déterminent le jeu de questions qui leur sont posées. Par exemple s'il répond correctement à la question A, l'apprenant se verra proposer la question B et dans le cas contraire il se verra proposer la question C (cf. Figure II.5). Cette étape terminée, les apprenants ayant effectué des parcours identiques, sont automatiquement affectés au même groupe ;

3. **Troisième étape collaborative :** cette troisième et dernière étape est collective, car les apprenants de chaque groupe (même parcours dans la deuxième étape) doivent partager une discussion autour des réponses aux questions proposées dans les QCM. L'enseignant peut intervenir dans les discussions de chaque groupe.

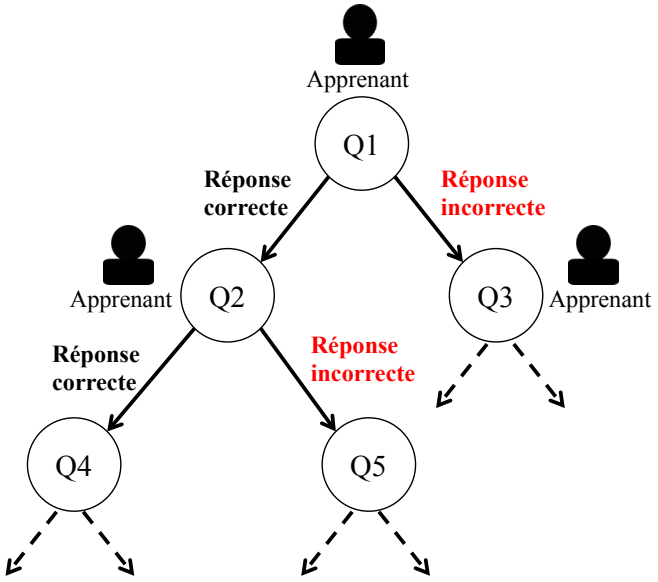

Figure II.5 : Parcours des apprenants adaptés aux réponses données

II.3.2 Situations imprévues proposées dans l'étude de cas

Ces *situations imprévues* sont des exemples de cas particuliers pouvant être rencontrés lors de l'exécution du scénario de l'étude de cas présentée ci-dessus.

Apprenant en retard

Dans cette situation imprévue, un apprenant qui n'arrive pas à rejoindre l'activité à cause d'un problème de connexion, bloque ses camarades qui l'attendent pour pouvoir accéder à la troisième étape du scénario.

Ajouter un nouveau participant à l'activité

L'enseignant veut inviter un nouvel apprenant (non prévu initialement) à participer à l'activité alors que le scénario est déjà en cours d'exécution.

Pas de rôle expert

L'enseignant veut inviter un *expert en environnement* à rejoindre l'activité pour répondre aux questions des apprenants. Celui-ci pourra par exemple consulter les réponses données par les apprenants et réagir en conséquence. D'autre part, l'expert aimerait pouvoir compter sur des services supplémentaires qui n'ont pas été prévus dans le scénario initial, comme par exemple la possibilité de faire visionner un film aux apprenants.

Dans la section suivante, nous reviendrons sur les exemples présentés ci-dessus pour expliquer et illustrer notre problématique.

II.4 Définition de la problématique d'adaptation

II.4.1 Adaptation à la volée

Nous nous intéressons à la possibilité donnée à un superviseur (enseignant, tuteur, informaticien, etc.) de modifier en cours d'exécution une activité se déroulant selon un scénario modélisé conformément à un EML. Nous désignerons par *adaptation à la volée* ce type d'adaptation d'une situation d'apprentissage instrumentée.

La figure II.6 situe l'adaptation à la volée dans le cycle de vie d'un scénario pédagogique (cf. Chapitre I).

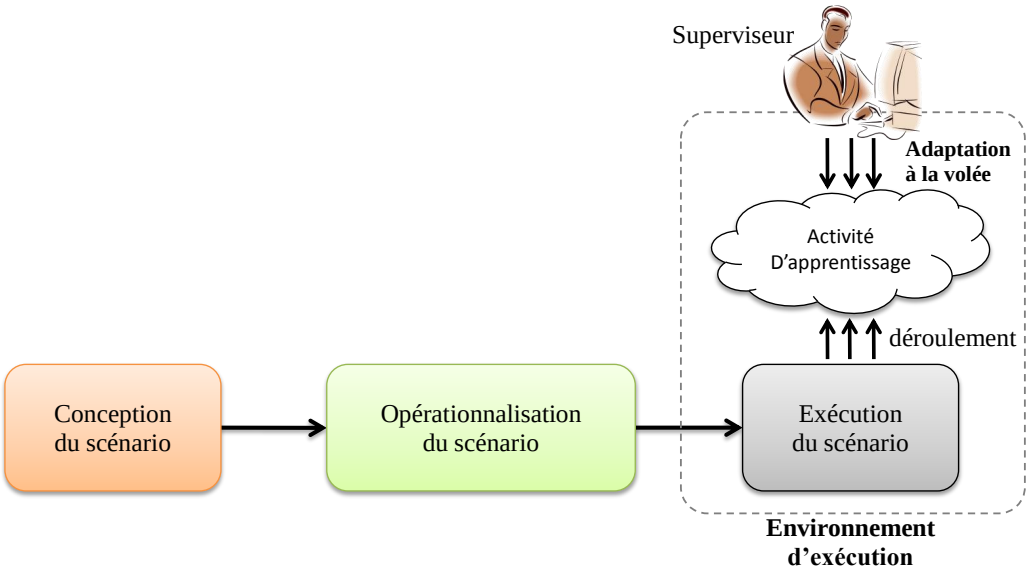

Figure II.6 : Positionnement de l'adaptation à la volée dans le cycle de vie d'un scénario pédagogique

En plus de permettre la modification des activités d'apprentissage mises en place en utilisant l'approche EML, adapter une activité au cours de son exécution ne peut être envisagé que sous

réserve de garantir la préservation des données nécessaires à cette activité. La cohérence des informations relatives aux interactions des participants ainsi qu'à leur progression doit dans tous les cas être maintenue.

II.4.2 Ce qui n'est pas de l'adaptation à la volée

Pour mieux situer notre problématique, nous étudions ici deux autres types d'adaptation que nous ne traitons pas dans nos travaux : l'adaptation prédéfinie et l'adaptation par conception itérative.

II.4.2.1 Adaptation prédéfinie

L'adaptation prédéfinie ou adaptation par modélisation avancée [Bur07] consiste à modéliser des alternatives directement dans le scénario. Les adaptations prédéfinies sont spécifiées lors de la phase de conception du scénario comme le montre la figure II.7. Il s'agit le plus souvent de règles sur lesquelles s'appuie le système pendant l'exécution du scénario pour personnaliser les parcours, les contenus et les services proposés aux participants [BS06, SB06, BSB+04].

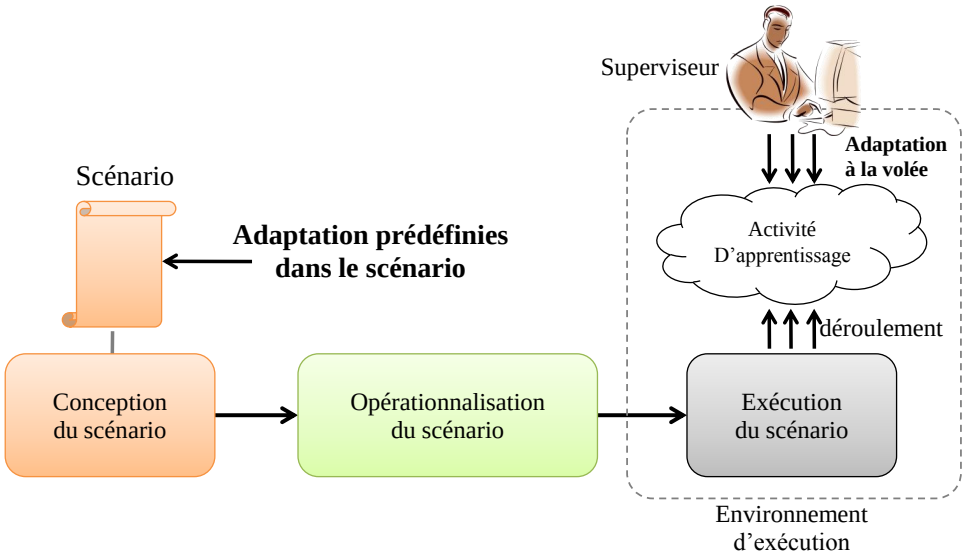

Figure II.7 : Positionnement de l'adaptation prédéfinie dans le cycle de vie d'un scénario

Exemple d'adaptation prédéfinie

Dans l'exemple fil rouge (cf. Section II.3), les apprenants choisissent un thème dans une pre-

mière étape. Ils sont ensuite redirigés automatiquement sur les questions associées à ce thème. Dans le scénario que nous avons implémenté pour le workshop AAMCS, nous avons utilisé des règles de la forme *si alors* pour mettre en place des alternatives qui seront sélectionnées à l'exécution en fonction des interactions des participants. L'alternative sélectionnée correspondra alors au thème choisi.

Toujours dans le même scénario, nous avons également mis en place le même mécanisme d'adaptation prédéfinie, pour rediriger les apprenants de question en question en fonction de leur réponse à chaque QCM. Ainsi, les apprenants auront la même question de départ, mais pas forcement le même parcours. Les questions sont alors sélectionnées selon le schéma représenté par la figure II.8.

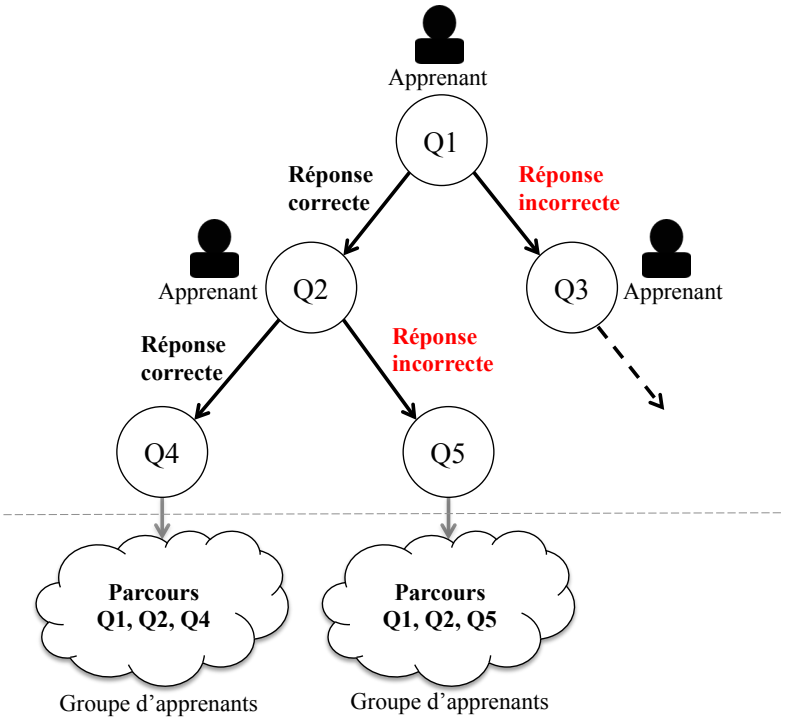

Figure II.8 : Calcul des parcours en fonction des réponses des apprenants

Même si elle permet de rendre possible une certaine flexibilité lors de l'exécution des scénarios, l'adaptation prédéfinie reste limitée à un ensemble d'alternatives fixées lors de la phase de conception et ne permet donc pas de pallier les événements imprévus qui empêchent le bon déroulement du scénario lors de son exécution.

II.4.2.2 Adaptation par conception itérative

L'adaptation par conception itérative est une approche descriptive qui consiste à spécifier les adaptations durant l'exécution du scénario. Un exemple de ce type d'adaptation est présenté dans [ZDFPD07].

La différence entre ce type d'adaptation et l'adaptation prédéfinie réside dans le fait que les adaptations ne sont pas définies au même moment. Dans l'adaptation par conception itérative, on spécifie les adaptations durant l'exécution du scénario. Ce n'est qu'une fois l'exécution terminée que l'on intègre les adaptations au scénario. Celles-ci seront alors prises en compte dans les prochaines exécutions de ce scénario. Les adaptations sont définies et intégrées à chaque itération du cycle de vie du scénario. Autrement dit, quand le superviseur observe une situation imprévue, il décrit immédiatement (pendant l'exécution) les modifications qui devraient être apportées au scénario sans affecter son exécution (ré-itération de l'étape de conception après l'exécution). La figure II.9 positionne ce type d'adaptation dans le cycle de vie du scénario.

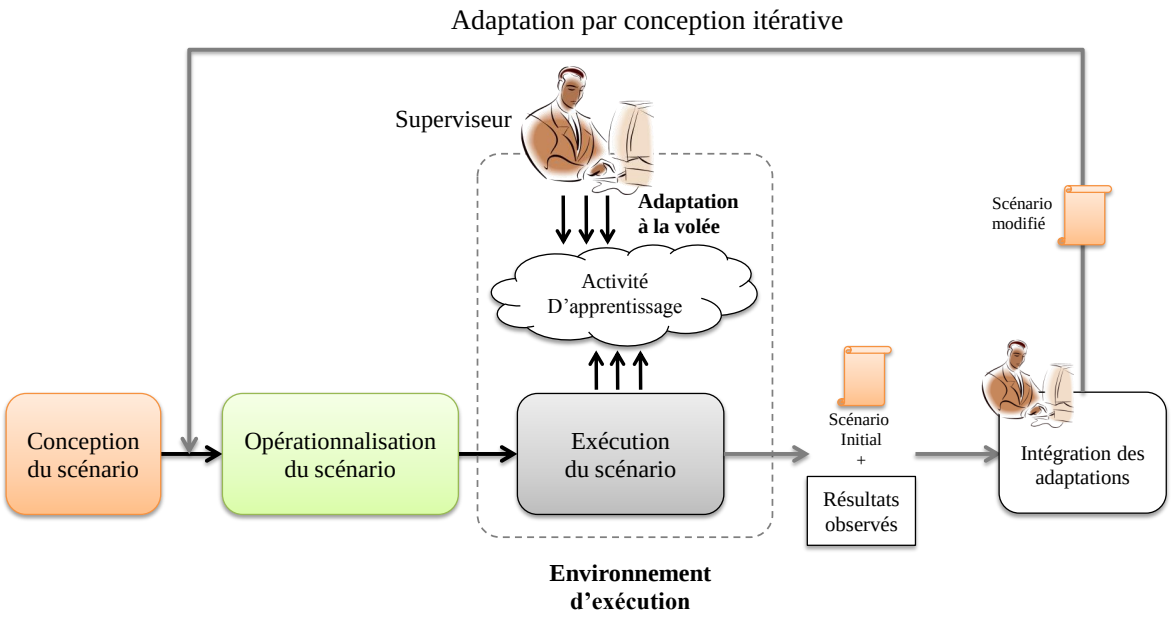

Figure II.9 : Positionnement de l'adaptation par conception itérative dans le cycle de vie d'un scénario

On s'inspire ainsi de la façon dont procède l'enseignant dans sa classe, lorsqu'il note les amé-

liorations qu'il pourrait apporter à son cours pour les prochaines séances.

Ce type d'adaptation permet d'améliorer et d'affiner le scénario au fil des exécutions, mais reste limitée, car ne permettant pas de répondre à la problématique d'adaptation à la volée.

II.5 Caractérisation de l'adaptation à la volée

Au titre de première contribution, nous avons caractérisé l'adaptation à la volée :

- du point de vue des situations imprévues dont nous dégagerons les propriétés ;
- du point de vue des modifications qu'elle requiert en nous intéressons notamment aux éléments ciblés.

II.5.1 Tensions ou opportunités

Nous avons distingué deux types de situations imprévues selon l'objectif de l'intervention du superviseur :

- **tension :** si elle n'est pas prise en compte, l'exécution du scénario et le déroulement de l'activité peuvent être perturbés ;
- **opportunité :** si elle est prise en compte, c'est uniquement pour améliorer l'exécution du scénario et le déroulement de l'activité.

Ainsi l'adaptation à la volée de la situation imprévue de l'apprenant en retard (cf. Section II.3.2), est de type "réponse à une tension". Au contraire, l'adaptation à la volée qui permettrait de répondre à la situation imprévue *pas de rôle expert* (cf. Section II.3.2), est de type "réponse à une oppportunité", car son objectif est d'améliorer le déroulement de l'activité.

II.5.2 Caractère bloquant d'une situation imprévue

Une situation imprévue peut être bloquante ou non. Lorsqu'elle est bloquante, la situation doit être adaptée, car elle bloque ou pourrait bloquer l'exécution du scénario. L'exemple *apprenant en retard* (cf. Section II.3.2) est une situation bloquante.

II.5.3 Temporalité d'une situation imprévue

La temporalité d'une situation imprévue est liée à la progression de l'exécution du scénario lorsqu'une situation imprévue est observée. Cette progression dépend des deux points d'exécution et de modification du scénario que nous définissons comme suit :

- **point d'exécution :** est défini par la progression du participant ou des participants concernés par la situation imprévue. Dans le cas où ces progressions seraient différentes, c'est le participant qui a le plus avancé qui déterminera le point d'exécution. Dans l'exemple de l'apprenant en retard (cf. Section II.3.2) le point d'exécution coïncide avec le point de synchronisation où sont bloqués les apprenants à la fin de la première étape ;
- **point de modification :** est défini par la position de l'élément à modifier dans le scénario pour faire face à une situation imprévue en adaptant l'activité. Dans l'exemple de situation imprévue de l'apprenant en retard (cf. Section II.3.2), le point de modification coïncide avec le point d'exécution représenté par le point de synchronisation qui empêche les apprenants d'accéder à l'étape suivante. Ce qui n'est pas le cas dans l'exemple de la situation *pas de rôle expert* (cf. Section II.3.2), car le point de modification (troisième étape) se situe au delà du point d'exécution (deuxième étape).

Le point de modification et le point d'exécution nous permettent d'introduire ce que nous avons appelé la *temporalité* de la situation imprévue. En fonction de ces deux-points et du moment où a été observée la situation imprévue, la temporalité peut être :

- **antérieure :** le point d'exécution se situe après le point de modification de la situation imprévue (cf. Figure II.10). Visiblement la situation n'a pas bloqué l'exécution du scénario, elle est donc non bloquante. Toutefois, l'activité pourrait être adaptée pour éviter que cette situation n'arrive à nouveau (Ex. boucle dans le scénario) ;
- **en cours :** le point de modification de la situation imprévue coïncide avec le point d'exécution (cf. Figure II.10). L'adaptation aura un impact direct sur l'activité en cours ;
- **postérieure :** le point d'exécution n'est pas encore passé par le point de modification (cf. Figure II.10). Il pourrait être nécessaire d'intervenir pour éviter un blocage de l'exécution du scénario avant que le point d'exécution ne passe par le point de modification. La situation *pas de rôle expert* a une temporalité postérieure.

II.5.4 Portée d'une situation imprévue

La portée désigne les participants concernés par la situation imprévue et par l'adaptation à la volée. Dans l'exemple de situation *pas de rôle expert* (cf. Section II.3.2), la portée est limitée aux apprenants qui seront les seuls concernés par l'adaptation. L'activité de l'enseignant restera inchangée.

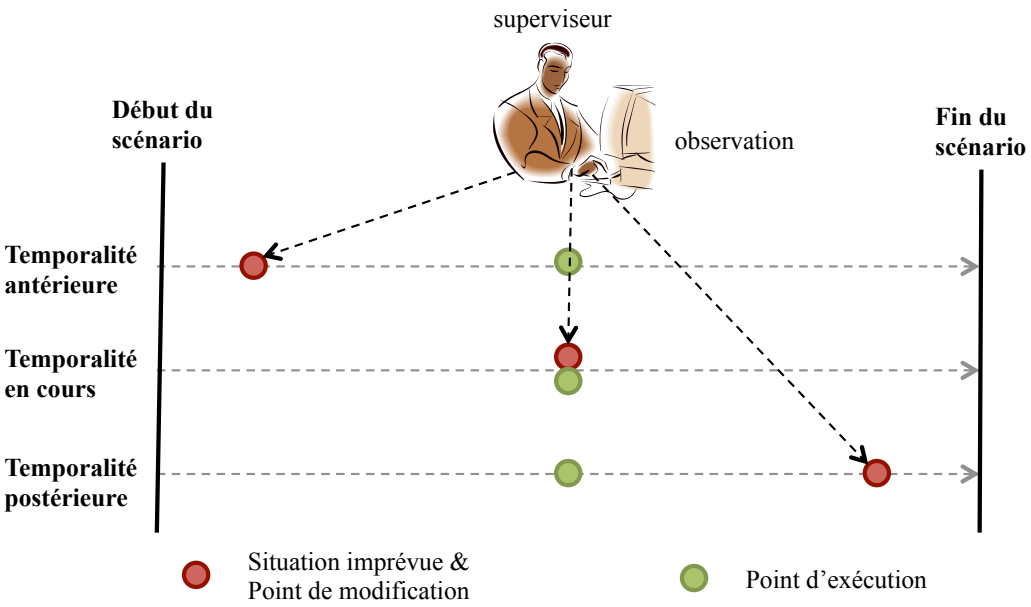

Figure II.10 : Temporalité d'une situation imprévue

II.5.5 Nature de ce qui est adapté

Il s'agit des éléments à modifier pour adapter une situation imprévue. Ces éléments sont définis pas :

- **les informations liées à la conception du scénario** : ce sont tous les éléments définis dans le modèle du scénario ;
- **les informations liées à l'opérationnalisation du scénario** : ce sont toutes les informations spécifiées lors de phase d'opérationnalisation du scénario, elles permettent de désigner :
  - les participants de l'activité,
  - le rôle de chaque participant dans l'activité,
  - les ressources, les services et les outils utilisés dans l'activité,
- **les informations liées à l'exécution du scénario :** ce sont toutes les informations générées par l'exécution du scénario et les interactions des participants : progressions, résultats, réponses, choix, etc. Elles définissent aussi l'état de l'activité d'apprentissage.

A titre d'exemple, nous donnons ci-dessous la nature des différents cas d'adaptation de l'exemple fil rouge II.3 :

1. ***pas de rôle expert* :** requiert de modifier le scénario en cours d'exécution pour y ajouter le nouveau rôle *expert* et ses interactions ;

2. ***ajouter un participant*** **:** nécessite de modifier les informations liées à l'opérationnalisation du scénario, en affectant le rôle apprenant à un nouvel utilisateur pour lui permettre de rejoindre l'activité ;
3. ***apprenant en retard*** **:** requiert de modifier les informations liées à l'exécution du scénario, en modifiant les progressions des apprenants pour leur permettre d'aller directement à l'étape suivante.

II.6 Cadrage de la problématique

Nous proposons dans cette section un cadrage qui rassemble les conditions que nous avons prises en compte dans nos contributions et dans les réalisations de nos travaux.

II.6.1 Questions pour l'adaptation à la volée

La figure II.11 expose les questions et les réponses (conditions) que nous respecterons dans le reste de ce mémoire.

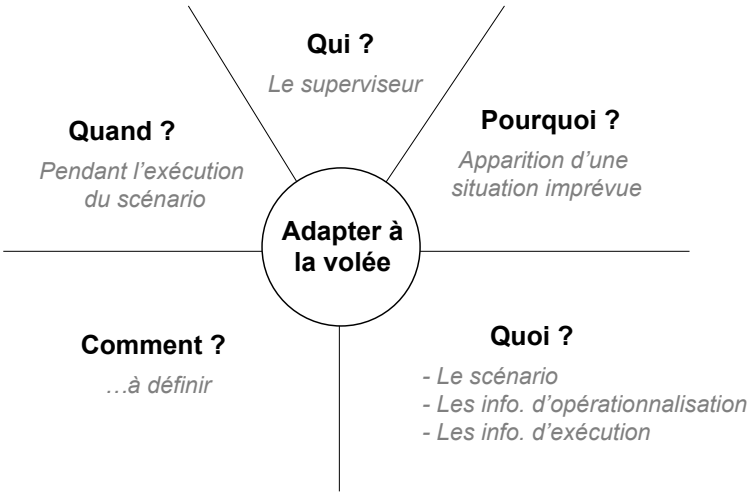

Figure II.11 : Cadrage de la problématique d'adaptation à la volée

Qui adapte à la volée : nous désignerons par *superviseur de l'activité*, la personne responsable de l'adaptation à la volée. Ce peut être l'enseignant, le tuteur, le concepteur du scénario ou encore l'ingénieur pédagogique. C'est aussi le superviseur qui observe l'activité et qui juge de la nécessité d'intervenir pour adapter. Il doit donc planifier, définir et mettre en place toute modification.

Quand adapter-t-on à la volée : pendant l'exécution du scénario, par opposition à l'adaptation prédéfinie (cf. Section II.4.2.1).

Pourquoi adapter à la volée : soit pour améliorer le déroulement de l'activité (opportunité) ou alors pour éviter que l'exécution du scénario ne soit perturbée ou bloquée (tension).

Qu'adapte-t-on à la volée : selon la nature de l'adaptation à la volée, toutes les informations liées à la conception, à l'opérationnalisation ou à l'exécution du scénario.

Comment adapter à la volée : c'est l'objet de notre recherche. Notre travail consiste à proposer une approche et des outils pour permettre l'adaptation à la volée observée lors de l'exécution d'un scénario.

II.6.2 Quels moyens pour l'observation

La trace d'activité est le résultat d'une situation d'apprentissage : résultats des apprenants, leurs copies, tableau non effacé, etc. Dans une situation informatisée, c'est le système qui s'occupe de la collecte et de la sauvegarde de ces traces. On parle dans ce cas là de collecte de traces numériques ou de traces d'apprentissage. Une fois collectées, les traces sont sauvegardées dans une base de données (base de traces). Ces traces peuvent alors être exploitées pour :

- l'observation : collectes des traces numériques pour suivre l'activité d'apprentissage et savoir ce qui s'y passe. Ces traces peuvent servir au calcul d'indicateur d'activité qui aideront les enseignants à comprendre, à évaluer, à suivre et à soutenir l'apprentissage [DMR+09]. Elle peuvent également servir à la personnalisation de l'apprentissage [BS06, SB06, BSB+04] ;
- l'analyse : il s'agit d'analyser les traces collectées une fois l'activité d'apprentissage achevée, par exemple pour inférer ou vérifier des hypothèses sur l'appropriation effective des connaissances par les apprenants [PL04]. Elles sont aussi utilisées pour concevoir les indicateurs d'activité qui seront exploités durant l'activité d'apprentissage.

Dans notre contexte, les traces sont produites par l'exécution du scénario pédagogique et exploitées pour observer en temps réel les interactions des participants.

Nous supposerons dans la suite de ce mémoire, que nous disposons de moyens d'observation de traces et de calcul d'indicateurs permettant de détecter et de rendre compte en temps réel, de l'apparition de situations imprévues et de la nécessité d'adapter à la volée. LDI implémente des mécanismes de capture de traces et d'observation (cf. Chapitre I), mais il revient au superviseur de détecter les tensions et de déceler les opportunités.

II.7 Bilan et conclusion

Nous avons introduit dans ce chapitre l'adaptation à la volée dans un contexte d'apprentissage instrumenté et utilisant l'approche EML. De cette problématique nous avons exclu d'autres types d'adaptation comme l'adaptation prédéfinie et l'adaptation par conception itérative, car elles ne répondent pas à la problématique d'adaptation à la volée.

Nous avons présenté des exemples de situations imprévues (cf. Section II.3) et nous les avons caractérisés relativement à notre problématique. Ainsi, nous avons défini les caractéristiques suivantes :

pour les situations imprévues :

- **caractère bloquant :** qui détermine si une situation imprévue bloque ou non l'exécution du scénario ;
- **temporalité :** dépend des points de modification et d'exécution. Elle peut être antérieure, en cours ou postérieure ;
- **tension ou opportunité :** la tension peut bloquer l'exécution du scénario et nécessiter une adaptation à la volée. Au contraire, une opportunité est uniquement une occasion d'améliorer l'activité.

pour l'adaptation à la volée : la nature qui détermine ce qui doit être modifié pour adapter une situation imprévue : le scénario, les informations liées à son opérationnalisation ou les informations liées à son exécution.

Les caractéristiques dégagées nous ont permis ensuite de cibler les besoins auxquels il faut répondre pour adapter à la volée. L'exemple fil rouge et les caractéristiques présentés dans ce chapitre nous serviront de référentiel pour présenter et comparer les travaux relatifs à l'adapta-

tion dans le chapitre suivant.

Chapitre III

Etat de l'art : adaptation à la volée

Dans ce chapitre, nous nous positionnons dans le cadre des travaux menés sur la problématique d'adaptation. Nous aborderons cette problématique en considérant aussi bien les travaux menés dans le domaine du Learning Design (cf. la section III.1) que ceux menés dans le domaine du *Workflow* (cf. la section III.2). La problématique de la flexibilité dans les *workflows* est en effet assez proche de celle de l'adaptation dans le Learning Design. Pour le Learning Design, nous nous intéresserons tout naturellement aux travaux des chercheurs ayant relevé le défi du workshop "Adapting Activities Modeled by CSCL Scripts" (AAMCS) organisé lors de la conférence CSCL'2009 [FH09] (cf. Section II.3.1) en s'y engageant en tant que compétiteur. Ce workshop a en effet atteint son objectif de rassembler la communauté internationale travaillant sur cette problématique. Nous y avons nous-même participé et exposé nos premières propositions (présentées dans le chapitre IV).

Nous étudierons tous ces travaux en prenant en compte l'exemple "fil rouge" et les caractéristiques dégagées dans l'étude de la problématique (cf. Chapitre II). Cet exemple nous servira en quelque sorte de "benchmark" permettant d'indiquer dans quelle mesure les propositions formulées fournissent, ou pas, des solutions complètes pour l'adaptation à la volée. Les trois situations imprévues envisagées dans l'exemple "fil rouge" (cf. Section II.3) sont en effet des situations représentatives des trois natures d'informations existant lors de l'exécution du scénario : informations liées à sa conception, informations liées à son opérationnalisation, informations liées à son exécution (cf. Section II.5.5). Une solution est de fait complète pour l'adaptation à la volée si elle permet de traiter les trois cas. Nous reproduirons ainsi dans ce chapitre la démarche qui fut celle du Workshop AAMCS.

III.1 De l'adaptation dans le domaine du Learning Design

Lorsqu'il est question d'adaptation dans le domaine du Learning Design, la plupart des travaux traitent d'adaptation prédéfinie. Il nous a de fait semblé difficile de les passer sous silence, même si, comme nous l'avons déjà expliqué dans la section II.4, ce type d'adaptation ne résout pas le problème de l'adaptation à la volée, parce qu'il nécessite d'anticiper sur tout événement qui pourrait survenir lors d'une activité et de le formaliser dans le scénario. Nous décrivons de fait succinctement ces propositions dans la première partie de cette section (Section III.1.1), en complément de ce que nous avons déjà présenté auparavant, dans la section II.4.2.

Exposer ces travaux nous donnera par ailleurs l'occasion de rappeler les bases des principaux EMLs sur lesquels des propositions qui relèvent effectivement de l'adaptation à la volée ont été formulées. Nous présenterons ces propositions dans la seconde partie de cette section (cf. Section III.1.2.

Il est à noter que les auteurs des travaux décrits dans ce qui suit ont tous participé au workshop AAMCS [1] et que nous nous appuyons en grande partie sur les éléments de solution qu'ils ont présentés alors, certaines de ces solutions ayant été, depuis, complétées ou améliorées du fait de la poursuite de leurs travaux.

III.1.1 Approches d'adaptation prédéfinie

L'adaptation prédéfinie suppose de tout prévoir à l'avance. Tout est en effet directement "codé" dans le scénario durant la phase de conception. Les deux exemples de situations *prévues* proposées dans l'exemple "fil rouge" sont typiques du genre de situations qui peuvent être modélisées dans un scénario en utilisant ce type d'approche :

- **situation 1 :** les apprenants choisissent un thème dans la liste proposée. Le scénario est conçu de manière à déclencher l'alternative correspondant aux questions associées au thème choisi ;
- **situation 2** : les apprenants répondent individuellement à un ensemble de questions et leur parcours, d'une question à l'autre, dépend des réponses qu'ils apportent. Le scénario

1. Etaient présents lors de ce workshop, organisé dans le cadre de CSCL'2009 en juin à Rhodes, des chercheurs montrant des solutions couvrant, le plus souvent partiellement, les différentes situations d'adaptation de l'exemple "fil rouge", travaillant sur les EMLs IMS-LD, PoEml, Lams et bien sûr LDL. Une liste des compétiteurs est disponible en ligne à l'adresse : `http://www.lama.univ-savoie.fr/wiki/index.php/CSCL_Workshop_Challenge_on_Adaptation`

est conçu de manière à déclencher l'alternative correspondant à la réponse (cf. Figure II.5).

Cela explique de fait que tous les participants au workshop AAMCS, quel que soit l'EML utilisé pour formaliser le scénario, ont pu traiter ces deux situations. La démarche la plus communément répandue relève de la *modélisation avancée* (cf. Section III.1.1.1) qui exploite les capacités des langages à exprimer des enchaînements conditionnels. Chaque EML a alors ses spécificités pour décrire des alternatives (cf. Section III.1.1.2). Nous décrivons cela dans ce qui suit. Nous nous attarderons ensuite sur une approche plus originale : celle de la *modélisation différée* (cf. Section III.1.1.3).

III.1.1.1 Adaptation par modélisation avancée

La modélisation avancée, que l'on qualifie aussi d'adaptation statique [PRCRARLN10], permet aux concepteurs d'envisager des alternatives (situation prévues) lors de la phase de conception [PRCRAR10]. Dans ce cas-là, toutes les alternatives autorisées à l'exécution sont modélisées lors de la phase de conception. Il s'agit donc d'une approche *prescriptive*. Les objectifs définis dans le modèle de l'activité peuvent être atteints par plusieurs chemins (parcours) possibles. La sélection de ces chemins dépend ainsi des choix effectués par les utilisateurs ou par le Système pendant l'exécution du scénario. On parle d'*Adaptativité* ou d'*adaptabilité* selon que ce choix est effectué par le système ou par l'utilisateur [Bru01, ABH04, HN04].

Que ce soit pour l'adaptabilité ou l'adaptativité, la modélisation avancée dépend complètement de l'EML utilisé, de son pouvoir d'expression et de sa capacité à modéliser ce qui peut s'apparenter à des règles de la forme $siconditionalorsaction$. D'un EML à l'autre, les concepts proposés pour exprimer ce type de règles sont différents, plus ou moins simples à manipuler, mais tous ont pour objectif de permettre de spécifier, lors de la phase de conception, des alternatives qui seront interprétées à l'exécution.

Cela présente de fait l'avantage de contribuer à donner au scénario une certaine flexibilité, sans sortir cependant du cadre de situations totalement anticipées et prévues. Par ailleurs, le modèle produit peut vite devenir très complexe, le nombre de possibilités spécifiées lors de la conception du scénario pouvant être important.

III.1.1.2 Des EMLs supportant l'approche de modélisation avancée

Adaptation par modélisation avancée avec IMS-LD

Le méta-modèle d'IMS-LD [KOA03] est structuré en trois niveaux, A, B et C. Chaque niveau permet de modéliser certaines caractéristiques de l'activité [BS06]. Les niveaux B et C proposent des concepts pouvant être utilisés pour la modélisation avancée. Ces concepts sont les propriétés, les conditions, les éléments globaux et les services fournis par l'environnement d'exécution. Il est par exemple possible d'utiliser les propriétés pour stocker des valeurs qui sont testées ensuite par des conditions simples se basant sur la structure *si, alors sinon*. Ces dernières permettent aussi de modifier les valeurs des propriétés afin de masquer et de révéler des éléments à l'exécution. D'autres concepts tels que les environnements globaux, permettent de récupérer les entrées utilisateurs (formulaire, documents, réponses QCM), de les formater, d'afficher ou de masquer des contenus spécifiques à l'aide de calques HTML/XHTML. Les autres éléments du niveau C permettent de spécifier des adaptations bien plus proches de l'environnement. Ainsi les services de supervision et les systèmes de notifications permettent de capturer les valeurs des propriétés affectées aux utilisateurs et de déclencher des notifications lorsque certains événements sont détectés.

Adaptation par modélisation avancée avec LDL

Dans LDL [MVFD06], les concepts clés pour la modélisation avancée sont les règles, les positions et les observables. Une position peut modéliser le résultat d'une interaction à l'exécution (une réponse, un choix, uns préférence, etc.). Elle peut être observée par le système ou prise par un participant. Dans les deux cas, le recours au concept d'observable permet de récupérer les valeurs des postions, qui sont testées par des règles dont le résultat est utilisé pour adapter le processus d'apprentissage. Ce sont ces concepts qui sont exploités dans le scénario LDL décrit dans [LDM$^+$07] pour donner aux apprenants la possibilité de voter, d'utiliser des QCM ou encore pour générer dynamiquement des groupes d'élèves en fonction de leurs réponses (positions). Par exemple, une position peut être utilisée pour représenter le choix d'une réponse par un apprenant qui répond à un QCM. Le moteur qui interprète le scénario LDL évalue la réponse en utilisant une règle LDL qui compare la valeur de la position déclarée par l'apprenant et la bonne réponse du QCM. Le résultat de l'évaluation du résultat (vrai ou faux) de cette règle est ensuite utilisé pour sélectionner l'interaction qui sera présentée à l'apprenant. Cette interaction qui représente le prochain QCM que traitera l'apprenant est une alternative sélectionnée par le système en s'appuyant sur la valeur de la position observée sur l'interaction de l'apprenant.

Adaptation par modélisation avancée avec PoEML

PoEML (Perspective-oriented Educational Modeling Language) [CRLNAR06] est un EML qui propose une approche basée sur les perspectives tout en proposant divers niveaux d'abstraction pouvant être regroupés selon plusieurs aspects. Le méta-modèle PoEML est organisé en plusieurs paquets en conformité avec les perspectives et les aspects identifiés [CRMLN+07]. Les perspectives permettent de regrouper les éléments du méta-modèle en sous-ensemble de modèles ayant chacun un rôle déterminé lors de la modélisation. Les aspects ont été définis pour contrôler le déroulement du scénario, ils sont responsables de la modélisation avancée dans le méta-modèle PoEML. L'aspect *constante* permet par exemple de définir des constantes initialisées dans le scénario et devant rester inchangées lors des différentes instanciations de ce dernier. Un autre aspect basé sur les données permet d'influencer le déroulement du scénario en prenant en compte des valeurs ou des propriétés collectées et sauvegardées par le système. Par exemple les connaissances acquises par l'apprenant dans des sessions antérieures. L'aspect basé sur les événements est responsable de la détection des changements dans le scénario et ses instances. Le dernier aspect basé sur les décisions est impliqué dans la gestion de la prise de décision lors qu'il s'agit d'effectuer des changements liés à la structure ou le comportement d'une unité d'apprentissage.

Adaptation par modélisation avancée dans LAMS

LAMS [Dal03] (Learning Activity Management System) propose une approche qui se distingue de celles déjà présentées ci-dessus, car il n'explicite pas de méta-modèle conceptuel. Il propose une notation graphique et un éditeur pour la modélisation des scénarios. LAMS n'est donc pas un langage de modélisation, mais un environnement de modélisation, d'opérationnalisation et d'exécution d'activités d'apprentissage. Pour ce qui est de la conception, LAMS propose un *outil de modélisation graphique* qui s'appuie sur le principe de séquences d'activités. Ces séquences sont alors créées à partir d'activités représentées par des outils (Un Wiki, un Chat, un forum, un Questionnaire, etc.), reliées entre elles par des transitions. L'adaptation par modélisation avancée dans LAMS consiste à configurer les activités et les transitions pour permettre à l'activité de s'adapter automatiquement aux réactions des apprenants. L'adaptation est réalisée grâce à une transition spéciale appelée transition de branchement. Cette transition prend en entrée des options représentées par des variables prédéfinies lors de la phase de conception et évaluées à l'exécution. Le résultat obtenu permet alors de déterminer quelles activités seront proposées aux apprenants.

III.1.1.3 Adaptation par modélisation différée

Dans ce type d'adaptation, certaines parties du scénario de l'activité sont complétées à l'exécution [PRCRAR10]. Les parties incomplètes appelées *placeholder* peuvent être vues comme des boites noires, devant (ou pouvant) être complétées à l'exécution en prenant en compte les réactions des utilisateurs. Le fait de compléter le scénario à l'exécution introduit plus de flexibilité rendant possible quelques adaptations et permettant ainsi de prendre en compte les réactions des utilisateurs (inconnues lors de la conception du scénario). Par exemple, le choix d'une durée d'examen à l'exécution en fonction des exercices choisis par les apprenants. Dans ce cas là, lors de la conception du scénario, un *placeholder* est prévu pour recevoir la valeur de la durée de l'examen à l'exécution. L'exécution du scénario est suspendu jusqu'à ce que la valeur de cette durée soit précisée par l'enseignant (*placeholder* contient une valeur). Cette approche permet donc de supporter centaines situations à condition de prévoir les placeholders requis pendant la conception du scénario. Elle ne permet donc pas d'anticiper et de gérer des situations imprévues, car toute réaction provoquant une adaptation à l'exécution doit être prévue à l'avance à travers la spécification d'un placeholder dans le scénario.

III.1.1.4 Conclusion

Les approches d'adaptation prédéfinie se focalisent uniquement sur la définition d'alternatives permettant de faire face à des situations *prévues*. Elles ne traitent pas le cas où des situations imprévues surviennent à l'exécution du scénario. Elles ne rentrent donc pas dans le cadrage de la problématique d'adaptation à la volée.

III.1.2 Approches d'adaptation à la volée dans le domaine du LD

Par opposition à l'adaptation prédéfinie, l'adaptation à la volée se focalise sur des adaptations qui sont effectuées lors de l'exécution du scénario. Pour chaque approche qui relève à notre avis de l'adaptation à la volée dans le LD et qui est présentée dans ce qui suit, nous fournirons tout d'abord une description. Nous nous attacherons ensuite à indiquer en quoi cette approche propose, ou pas, des solutions d'adaptation pour les trois situations imprévues de l'exemple "fil rouge" : *apprenant en retard*, *ajout d'un participant*, *pas de rôle expert*. Cela nous permettra de comparer ces approches entre elles, et avec les propositions que nous formulons dans cette thèse. Il sera en effet aisé de constater qu'aucune des solutions présentées ne traitent les trois situations. Or elles ont été choisies, nous le rappelons, en tant que situations représentatives de la nature des informations à modifier. Les traiter toutes les trois revient de fait à pouvoir

apporter une solution pour l'adaptation à la volée. C'est le challenge que nous nous sommes fixé dans cette thèse.

III.1.2.1 Poches d'adaptation : proposition de Zarraonandia

Description

[ZDF06] propose une approche d'adaptation à la volée permettant d'introduire de petites variations lors de l'exécution d'une Unité d'Apprentissage (UA) spécifiée avec IMS-LD. Il s'appuie dans cette approche sur un mécanisme qu'il appelle *poche d'adaptation*. En fait, une poche d'adaptation encapsule (dans un fichier) l'ensemble des modifications requises pour l'adaptation à la volée. Ces modifications regroupent des ajouts, des suppressions, des modifications, ainsi que la définition de nouveaux éléments IMS-LD (le cas échéant) et les nouvelles ressources (le cas échéant) requis pour une adaptation spécifique. L'auteur de l'approche propose également de modifier le moteur *Coppercore* [OUN11] pour lui permettre d'intégrer les *pokes* (modifications) à l'UA initiale en temps réel.

Traitement de l'exemple fil rouge

L'utilisation des poches permet de supporter certaines adaptations à l'exécution mais reste limitée à des variations minimes dans l'UA en cours d'exécution [ZDF06]. Voyons ces limitations avec l'exemple fil rouge (cf. Chapitre II) :

- **apprenant en retard :** cette situation peut être gérée en utilisant les poches d'adaptation. Il est possible de créer une poche d'adaptation qui encapsule les modifications qui vont présenter (modifier des propriétés IMS-LD pour rendre visible) les éléments requis permettant aux apprenants d'accéder directement à la phase suivante ;
- **ajouter un apprenant :** cette situation nécessite de modifier les informations liées à l'opérationnalisation du scénario. L'approche des poches d'adaptation supporte la définition de nouvelles ressources et de leur association avec des environnements ou des activités existantes. Toutefois, ces modifications n'incluent pas les utilisateurs inscrits dans l'environnement d'exécution. Il est donc impossible de faire les modifications nécessaires pour l'ajout d'un nouvel apprenant à l'activité ;
- **pas de rôle expert :** cette situation nécessite de modifier le scénario (la définition de l'UA). L'auteur des poches d'adaptation précise clairement dans [ZDF06] que l'approche ne considère pas les modifications qui touchent au scénario. Il précise en disant que "l'introduction de nouveaux rôles, d'actes, de conditions complexes ou de changements structu-

raux requièrent la description de fichiers d'adaptation complexes (les poches d'adaptation sont spécifiées dans un fichier), ce qui fait de la re-conception (nouveau cycle) de l'UA une approche plus appropriée".

III.1.2.2 Adaptation à la volée dans le cadre du projet *TENCompetence*

Déscription

TENCompetence est un projet financé par la Commission européenne [TEN11]. Il développe et utilise une infrastructure dont l'objectif et de supporter, d'accompagner et de faire évoluer les compétences des individus, des groupes et des organisations [TEN11]. L'infrastructure proposée regroupe plusieurs outils implémentés pour supporter la spécification IMS-LD. [KM08] s'intéresse aux capacités d'adaptation de cette infrastructure. L'auteur insiste sur le fait que de nombreux mécanismes d'adaptation peuvent être construits en exploitant les propriétés et les conditions proposées par IMS-LD (niveau B IMS-LD cf. I.2.3). Les conditions et les propriétés proposés par IMS-LD (à l'instar des règles et des conditions de LDL) peuvent en effet servir pour mettre en place des mécanismes d'adaptation permettant aux tuteurs (ou enseignants), aux apprenants ou encore au système (Adaptabilité ou Adaptativité cf. Section III.1.1.1) de choisir des alternatives prédéfinies.

Traitement de l'exemple fil rouge

Le problème qui se pose avec l'approche proposée ci-dessus est que les alternatives ne sont pas mises en place à l'exécution, mais pendant la conception du scénario. Il est par exemple difficile d'accéder aux informations liées à l'exécution afin d'y effectuer des modifications. Voyons avec l'exemple fil rouge :

- **apprenant en retard :** le seul moyen de gérer cette situation en utilisant IMS-LD et l'infrastructure proposée dans *TENCompetence* est de prédéfinir un booléen qui une fois positionné à vrai par l'enseignant permettra à l'apprenant d'aller directement à la phase suivante. Seulement, le fait de prédéfinir le booléen relève de l'adaptation par modélisation avancée (prédéfinie) et non pas de l'adaptation à la volée. En effet, l'infrastructure de *TENCompetence* ne permettant pas l'accès et la modification des informations liées à l'exécution du scénario, ne pourra pas être utilisée pour gérer la situation imprévue de l'apprenant en retard.
- **ajouter un apprenant :** c'est l'unique situation qu'il est possible de gérer en utilisant IMS-LD et l'infrastructure proposée dans *TENCompetence*. En effet, cette dernière pro-

pose la possibilité d'ajouter un nouvel apprenant à l'activité en cours, en créant et en lui affectant une nouvelle instance conforme au scénario prévu dans l'UA. L'apprenant pourra joindre l'activité et y participer sans problème.

- **pas de rôle expert :** cette situation nécessite de modifier la conception du scénario prévu initialement, or l'infrastructure proposée dans *TENCompetence* n'offre aucun mécanisme pour cela.

III.1.2.3 L'adaptation à la volée et PoEML : proposition de Perez-Rodriguez

Description

[PRCRARLN10] propose une approche d'adaptation à la volée centrée sur la modification ad hoc du scénario à l'exécution. Il s'appuie dans cette proposition sur le langage PoEML et l'orientation aspects que procure ce dernier. Il sépare ainsi entre les préoccupations. Par exemple, il propose un sous-modèle pour les activités individuelles, un sous-modèle pour les activités collaboratives, un autre pour l'organisation et l'enchainement des activités, etc. Ces sous-modèles sont ensuite combinés pour construire le scénario de l'activité d'apprentissage.

Dans cette approche, on considère que l'adaptation par modification ad hoc est une adaptation dynamique pouvant être effectuée à l'exécution en introduisant les changements ad hoc sur les modèles en cours d'exécution. PoEML propose plusieurs modèles (ou sous-modèles) pour décrire l'activité : le modèle utilisateur qui décrit l'activité individuelle de ce dernier, modèle de l'activité collaborative ou de l'organisation des activités individuelles, etc. Les adaptations sont classifiées en fonction du modèle qu'elles ciblent. L'auteur parle d'adaptation de :

- *Niveau A* : à ce niveau, seules les modifications des modèles utilisateurs sont autorisées. Ces modifications ont donc un impact limité à l'activité d'un utilisateur ou d'un groupe d'utilisateur désignés. Dans certains cas, la modification d'un ou plusieurs modèles utilisateurs peut avoir un impact sur les autres modèles restés inchangés ;
- *Niveau B* : à ce niveau, la modification du modèle de l'activité est aussi autorisée. Il peut s'agir par exemple de la définition des activités elles-mêmes ou de leur enchainement.

[PRCRARLN10] propose une infrastructure qui implémente cette approche. L'architecture de cette infrastructure s'appuie sur un moteur d'exécution orienté services (Approche SOA) conçu autour du langage PoEML, et sur des interfaces auteur pour modifier les modèles en faisant appel à des primitives simples : *ajouter une activité*, *supprimer une activité*, *remplacer*

une activité, etc. Le méta-modèle d'exécution proposé s'appuie sur un Automate à États Finis (AEF) pour représenter et contrôler l'exécution des processus associés aux modèles de l'activité. Autrement dit, ce méta-modèle permet d'instancier les éléments du scénario PoEML et de les représenter sous forme d'AEF, ce dernier étant interprété et exécuté par le moteur d'exécution.

L'AEF est utilisé pour encapsuler les éléments du scénario PoEML. L'instanciation des éléments PoEML dépend de l'évaluation des contraintes rattachée aux nœuds de l'AEF et des objectifs fixés dans le scénario. Chaque objectif doit être accompli dans une situation (une tâche ou sous activité). Une fois la contrainte évaluée, l'automate déclenche la transition qui permettra à l'utilisateur concerné d'aller à l'activité suivante. Les éléments PoEML de la nouvelle situation sont alors instanciés à la demande de cette transition. Autrement dit, les instances des utilisateurs sont créées au fur et à mesure en fonction des transitions déclenchées par l'AEF.

Pour modifier les éléments PoEML correspondants à l'instance ciblée lors de l'adaptation, le système bloque tout accès utilisateur. Les instances deviennent accessibles à nouveau lorsque toutes les primitives de modification (ajouter, modifier, supprimer) sont appliquées, les utilisateurs concernés peuvent alors continuer leurs progression.

Traitement de l'exemple fil rouge

L'originalité de cette approche réside dans le fait que les éléments du scénario sont instanciés à la demande pendant l'exécution des processus utilisateurs, en fonction de leur progression. Les éléments instanciés dépendent donc du parcours de chaque utilisateur, contrôlé par l'AEF et les contraintes définies dans ses nœuds. Cependant, cette approche a l'inconvénient de l'accès au point de modification. En effet, comme avec l'approche d'adaptation par modélisation différée, les utilisateurs concernés par l'adaptation doivent être en dehors du point de modification. Voyons cela avec l'exemple fil rouge :

- **apprenant en retard :** cette situation ne peut pas être gérée en utilisant cette approche, car les apprenants bloqués par l'apprenant en retard sont déjà sur le point de modification. Il est par conséquent trop tard pour bloquer l'accès à ce point. Pour y remédier, il faut pouvoir déplacer les apprenants à un point situé avant le point de modification, mais là encore l'approche ne permet pas de modifier les informations liées à l'exécution ;
- **ajouter un apprenant :** l'approche présentée ci-dessus ne peut pas être utilisée pour modifier les informations liées à l'opérationnalisation du scénario, celle-ci cible uniquement

le scénario à travers l'utilisation de primitives (ajouter, supprimer, modifier) déclenchées par l'AEF ;

- **pas de rôle expert :** c'est la seule situation qui peut être gérée par l'approche présentée ci-dessus. Il est en effet possible de définir des primitives pour intégrer les éléments requis pour l'ajout du rôle. Par exemple : ajouter le rôle et ses activités, les environnements dont il aura besoin, etc. Cependant, là aussi, il faut que les modifications soient situées en dehors du point de modification pour pouvoir effectuer l'adaptation.

III.1.2.4 Adaptation à la volée dans LAMS

Description

LAMS propose une approche qui s'appuie sur l'utilisation d'une transition spéciale permettant de bloquer et d'isoler la partie du modèle qui doit être modifiée (avant le point de modifications). Les apprenants sont donc mis en attente avant ce type de transition. Ils poursuivent leurs activités une fois les modifications terminées. C'est une approche intéressante, car elle permet de modifier le modèle de l'activité pendant l'exécution.

Traitement de l'exemple fil rouge

- **apprenant en retard :** cette situation ne peut pas être gérée en bloquant l'accès à la partie du modèle qui doit être modifiée. En effet, les apprenants bloqués sont justement sur la partie qui doit être modifiée et il n'existe pas de moyen permettant de les déplacer. LAMS ne permet donc pas la modification des informations liées à l'exécution ;
- **ajouter un apprenant :** LAMS permet d'ajouter des apprenants à l'activité. Celui-ci pourra rejoindre l'activité et y participer même si les autres apprenants ont déjà entamé leurs activités ;
- **pas de rôle expert :** LAMS ne possède pas de concept rôle, il est par conséquent impossible de permettre à un expert de participer à l'activité en cours.

L'approche d'adaptation utilisée dans LAMS est similaire à celles proposée par [PRCRARLN10]. Les deux approches utilisent en effet le même mécanisme qui permet de bloquer l'accès à la partie du modèle qui doit être modifiée pour l'adaptation. La modification des informations pendant l'exécution du scénario étant impossible, les situations de l'*apprenant en retard* et l'*ajout d'un apprenant* ne pourront pas être gérées par LAMS.

III.2 De l'adaptation dans le domaine du Workflow

Nous avons choisi de nous intéresser au domaine du Workflow car il a de nombreuses similitudes avec le LD. Le problème de l'adaptation à la volée se pose notamment de la même façon, sous la désignation de "gestion des exceptions" dans le domaine du workflow. Nous présentons dans ce qui suit (cf. la section III.2.4) des solutions qui ont été proposées pour rendre les workflows plus flexibles de manière à permettre le traitement de certaines de ces exceptions, dans certains cas. Nous rappelons au préalable ce qu'est un workflow III.2.1 pour en fixer le vocabulaire, établissons le parallèle avec le LD pour justifier de l'étude de ce domaine (cf. la section III.2.2) et donnons des éléments de définition, caractérisation et classification des exceptions III.2.3.

III.2.1 Ce qu'est le workflow

Le terme *workflow* (ou flux de travail) fait référence à la manière dont on automatise l'exécution de processus dans lesquels sont impliqués une personne, un groupe de personnes, ou un organisme. Ces derniers sont alors amenés à réaliser de façon collaborative ou non des opérations ou des tâches en utilisant des outils (traitement de texte, tableur, modeleur, etc.) et des services (forum, chat, messagerie, etc.). On utilise également le terme *workflow* pour désigner la représentation des étapes à travers lesquelles seront réalisées les différentes tâches affectées aux acteurs qui les exécutent, ainsi que les délais et les conditions de cheminement des flux (produits, documents, informations qui transitent entre les étapes) produits.

La représentation du workflow se fait dans un *modèle de workflow* spécifié dans un langage de workflow tel que BPMN [Whi04]. C'est la phase de conception du workflow. Pour pouvoir exploiter et exécuter le workflow, il faut passer par une étape dite d'*enactment*. Durant cette étape, le modèle de workflow est instancié, des utilisateurs et des ressources (documents, image, pages web, etc.) sont choisis et les processus de workflow créés. Le modèle workflow est alors prêt à être exécuté afin de permettre aux différents utilisateurs d'effectuer les tâches qui leur sont attribuées dans la limite des délais et des conditions prévues initialement.

Remarque : dans ce qui suit, nous utiliserons le terme *Workflow* pour faire référence au domaine. Le *workflow* étant le flux de travail modélisé par un *modèle de workflow*.

III.2.2 Parallèle avec le Learning Design

Nous rassemblons dans cette section des précisions sur les raisons qui nous ont amené à étudier les approches d'adaptation proposées dans le domaine du Workflow. Dans la section III.2.2.1 nous présentons certaines similitudes entre l'approche utilisée dans le domaine du LD et celle utilisée dans le domaine du Workflow. Ces similitudes nous permettent de justifier le choix que nous avons fait en nous intéressant au Workflow. La section III.2.2.3 complète ensuite ces similitudes avec un alignement entre les vocabulaires utilisés dans les deux domaines.

III.2.2.1 Les similitudes entre Learning Design et Workflow

Le Workflow s'appuie sur une approche similaire à celle utilisée dans le LD. Cette similitude réside dans l'utilisation d'un modèle conçu à l'aide d'un langage de modélisation formel dans le but de mettre en place de manière automatique ou semi-automatique une activité qui se déroulera au sein d'un environnement d'exécution cible. Il existe d'ailleurs des travaux qui proposent d'utiliser un EML comme langage de workfow. Par exemple, dans [SBdCS07], c'est le langage IMS-LD qui a servi de langage de modélisation de workflow.

En ce qui concerne les travaux présentés dans ce mémoire, voici les raisons pour lesquelles nous nous sommes intéressés au domaine du Workflow :

- comme dans une approche EML où on décrit une situation d'apprentissage dans un scénario pédagogique, le *modèle de workflow* décrit un flux de travail qui représente l'ordre logique d'exécution des tâches par les différents acteurs qui seront impliqués dans l'activité ;
- comme dans une approche EML, le Workflow a recours à une approche de conception descriptive de l'activité, en se basant sur un méta-modèle conceptuel d'un langage de modélisation comme : BPMN [Whi04] ou des langages de description de processus exécutables tels que BPEL [Slo06], XPDL [Sha02] ou encore BPEL4WS [Jur06] pour les Services Web [2] ;
- comme dans une approche EML, la phase de conception du workflow est suivie par une phase d'opérationnalisation appelée *Enactment*. L'objectif de cette phase est alors d'instancier le modèle de workflow et de créer le ou les processus qui vont s'exécuter dans un environnement d'exécution appelé système de workflows [Man02] ;

2. Un Service Web est une entité logicielle accessible sur Internet, indépendamment de toute technologie de tout langage de programmation

– comme dans une approche EML, les modèles de workflow sont interprétés par un moteur d'exécution appelé *moteur de workflow*.

Il existe tout de même une certaine différence entre les deux approches. Cette différence se ressent dans les objectifs qui motivent les deux approches. Dans le domaine du workflow, on a tendance à se focaliser davantage sur la modélisation de workflows normés et rationalisés, ce qui n'est pas le cas dans le LD.

III.2.2.2 L'imprévisibilité dans le Workflow

Le workflow se heurte à l'imprévisibilité des activités tout comme le LD. Dans le workflow, on parle d'exceptions, que l'on peut mettre en parallèle, dans une certaine mesure, avec les situations imprévues (cf. Chapitre II).

Dans le monde réel, les processus de travail sont plus complexes et plus riches en variations que dans un processus décrit par un modèle de workflow. Les activités modélisées par le workflow sont imprévisibles. Le problème est de fait le même que pour le LD : il peut y avoir des situations auxquelles on n'a pas pensé ou qui sortent du cadre "normal" du travail (par exemple, une personne qui tombe subitement malade, ou qui démissionne, et qui intervient dans un processus donné). Les utilisateurs de ces workflows ont besoin d'ajuster et de modifier à la volée ces modèles pour répondre au mieux à leurs attentes et exigences [KB97].

L'imprévisibilité des activités requiert un support pour l'adaptation à la volée comme c'est le cas dans le LD. Ce support doit permettre de gérer, de prendre en compte des exceptions et de s'y adapter dynamiquement [KBT+00].

III.2.2.3 Alignement du vocabulaire

Pour unifier et ainsi faciliter la lecture des sections suivantes, nous proposons d'aligner les termes utilisés dans le LD et dans le Workflow. Le tableau présenté dans la figure III.1 montre trois colonnes qui présentent dans l'ordre, les termes utilisés dans le Workflow, les termes utilisés dans le LD et les termes que nous utiliserons dans le reste de ce mémoire.

Termes utilisés dans le Workflow	**Termes utilisés dans une approche EML**	**Termes que nous utiliserons**
langage de workflow	EML	langage de modélisation
le modèle de workflow	le scénario	le modèle de l'activité
enactment	opérationnalisation	opérationnalisation
instances ou processus de workflow	processus d'apprentissage	instances
utilisateur du Workflow	participant	utilisateur
exception imprévue	situation imprévue	exception
exception prévue	situation prévue	exception prévue

Figure III.1 : Alignement de vocabulaire entre l'approche EML et le Workflow

III.2.3 Les exceptions dans le Workflow

III.2.3.1 Définition

Dans le contexte de l'exécution d'un processus de workflow, une exception est une situation exceptionnelle qui empêche le déroulement d'une activité ou un workflow complet [MA07]. On trouve plusieurs définitions de ces exceptions dans la littérature :

- [AL89] décrit l'exception comme étant la disponibilité d'un ensemble de règles pour gérer une condition ;
- [SM95] donne une autre définition dans laquelle il définit l'exception comme un cas particulier qui ne peut être traité correctement par le système, sans une intervention manuelle ;
- [SMS94] parle de variations prévues pour décrire le processus normal et des exceptions comme des événements qui ne peuvent pas être gérés par le flux prévu dans le processus principal ;
- [BW95] décrit les exceptions comme des événements non modélisés par le système ou de déviations du flux normal prescrit par son modèle ;
- [KBT+00] considère quand à lui l'exception comme étant un *événement imprévu* dont l'adaptation à la volée consisterait à proposer des mécanismes (outils) pour assister leurs transformations en des *variations prévues*, en faisant évoluer les processus de workflow ou leurs modèles

Dans le contexte de notre problématique nous pouvons comparer les exceptions aux *situations imprévues* étudiées et caractérisées dans le chapitre II. Notre définition se rapproche alors de

celle proposée de [KBT+00] (cinquième définition proposée dans la liste ci-dessus).

III.2.3.2 Caractérisation des exceptions

La caractérisation des exceptions est cruciale pour la prise de décision lors de l'adaptation à la volée [MA07]. Nous avons proposé dans le chapitre II une étude qui nous a permis de caractériser les exceptions. Nous donnons dans ce qui suit d'autres caractéristiques de ces exceptions.

[MA07] ont identifié les caractéristiques suivantes pour décrire les exceptions et tenter de les catégoriser :

- **portée :** les instances affectées par l'exception ;
- **détection :** peut être automatique (par le système) ou manuelle (le superviseur) ;
- **type d'événement :** essentiel pour la prise en charge de l'exception, par exemple pour son adaptation à la volée :
  - **les événements de données :** violation de règle d'utilisation de données,
  - **les événements temporels :** quand un événement prédéfini survient,
  - **les événements de workflows :** liés à des situations spéciales qui concernent le début ou la fin d'une activité,
- **impact organisationel :** les participants concernés par l'exception : un participant, un groupe ou tous les participants ;
- **différence par rapport à des règles organisationnelles :** elle peut être :
  - **établie :** des règles existent pour gérer l'événement mais elles sont introuvables,
  - **non établie :** des règles existent, mais elles ne sont pas applicables à l'événement,
  - **vraie :** il n'existe aucune exception pour gérer l'événement,
- **complexité de la solution :** facile ou difficile à gérer selon le temps nécessaire à sa prise en charge. Elle est difficile si la solution ne peut pas être obtenue dans un temps acceptable ;
- **temps de réaction :** rapide, relâché ou long selon la rapidité avec laquelle doit être pris en compte l'événement. Dans l'exemple de l'apprenant en retard, les apprenants bloqués doivent être débloqués rapidement ;
- **délai avant d'aboutir à une solution :** rapide ou relaxé selon la complexité de l'exception.

L'identification de la valeur de ces caractéristiques doit permettre de catégoriser les exceptions et d'établir des priorités dans leur traitement. Ces priorités vont dépendre du contexte

"métier" pour lequel le workflow est modélisé et dans lequel il est utilisé. On attribuera ainsi davantage de poids à telle caractéristique plutôt qu'à telle autre en fonction de ce contexte. Par exemple, dans le domaine bancaire, l'accent sera mis sur le temps de réaction et le délai avant d'aboutir à une solution : ils devront être très courts pour réduire les conséquences de l'exception.

III.2.3.3 Classification des exceptions

Il existe dans la littérature de nombreuses études qui s'intéressent aux exceptions dans le domaine du Workflow [DGJH+98, HT04, MA07]. Certaines de ces études s'intéressent notamment à leur catégorisation comme nous l'avons constaté dans la section précédente. D'autres études proposent de catégoriser les exceptions selon leur nature, leur impact sur l'exécution du workflow, les erreurs qu'elles occasionnent, etc.

[BCCT05] propose une classification dans laquelle il prend en considération deux caractéristiques : l'origine de l'exception et le moment de sa détection. Pour ce qui est de l'origine, il considère que l'exception peut avoir les trois origines suivantes :

- **comportementale :** ce sont des exceptions générées par l'utilisateur. Ce dernier n'a pas respecté l'ordre d'exécution des activités et des tâches prévus dans le modèle du workflow. Par exemple, dans l'exemple fil rouge, si un apprenant utilise le bouton précédent pour accéder à un QCM auquel il a déjà répondu (ancienne page) ;
- **sémantique :** dite aussi de type *application*, car provoquée par l'échec de l'exécution de l'exception qui n'atteint pas les objectifs attendus. Par exemple, l'utilisateur qui n'a pas poursuivi ses paiements périodiques ;
- **système :** erreur système due à un mauvais fonctionnement aussi bien côté client que côté serveur au niveau de l'environnement d'exécution. C'est par exemple le cas d'une panne du système ou d'un problème sur le réseau qui lie le poste de l'utilisateur au serveur qui héberge le système de workflow. C'est aussi le cas de l'exemple que nous avons présenté dans le chapitre II et qui concerne un élève en retard qui n'arrive pas joindre l'activité, à cause d'un problème de connexion.

Pour ce qui est du moment de la détection, il peut être :

- **synchrone :** les exceptions synchrones surviennent pendant l'exécution de l'activité de l'utilisateur. Elles sont liées à l'état courant de l'activité. Exemple, une erreur qui survient

alors que la session liée à l'activité de l'utilisateur est toujours en cours. L'utilisateur clique sur un lien et le serveur ne répond pas. L'exception doit être traitée dans l'immédiat ;

– **asynchrone :** l'exception asynchrone peut arriver à tout moment lors de l'exécution du processus de workflow. Elle est indépendante de l'état de l'activité en cours. Quel que soit l'état de la session de l'utilisateur, le traitement de l'exception peut être différé.

Nous pouvons de fait considérer que les approches de gestion des exceptions (adaptation) peuvent être classées et que leur classification s'appuie sur des caractéristiques utiles à leur étude, à leur détection et leur prise en charge. Ainsi, les caractéristiques que nous avons dégagées dans le chapitre II peuvent également servir dans cette optique là.

III.2.4 Gestion des exceptions dans le Workflow

III.2.4.1 Adaptation prédéfinie par configuration de modèles de workflow

Comme dans un EML, les langages de workflow supportent la spécification d'alternatives dans les modèles produits. Ces alternatives sont sélectionnées à l'exécution en fonction de l'état de l'activité, du profil de l'utilisateur, de ses choix, de ses préférences, etc. La flexibilité du workflow dépend alors de règles (*si alors sinon*) évaluées par le moteur de workflow. Dans la même idée que les approches d'adaptation prédéfinies présentées dans le domaine du LD (cf. Chapitre II), nous nous intéressons à une approche qui élargit les possibilités d'adaptation en reportant les choix pour les alternatives lors de la phase d'opérationnalisation (d'enactment) du modèle, sans pour autant traiter du problème de l'adaptation à la volée : les *workflow configurables*.

La configuration consiste à paramétrer le modèle de worklow avant son exécution. Ce paramétrage permet ainsi de sélectionner les parties du modèle de workflow qui seront exécutées. La configuration peut par exemple être utilisée pour opérer des petits changements dans le modèle du workflow afin de le façonner et de l'adapter aux règles de l'entreprise. C'est une approche utilisée dans les grandes entreprises qui souvent ont besoin de plusieurs configurations possibles pour mieux exploiter les modèles de workflows qu'ils utilisent [GVDAJVLR08]. Cette approche peut être exploitée dans l'exemple fil rouge en laissant le choix au superviseur de sélectionner le jeu de QCM (thème) qui sera proposé aux apprenants. Dans le cas contraire (utilisation directe sans paramétrage), ce sont les apprenants eux-mêmes qui choisiront le thème des QCM.

Configuration par les variants : approche de Gottschalk

[GVDAJVLR08] propose d'augmenter un langage de workflow existant en utilisant des *variants* comme support de paramétrage. Le variant agit comme un *aiguilleur* qui va sélectionner la sous partie du modèle de workflow à exécuter en fonction des paramètres qu'on lui fournit avant l'exécution.

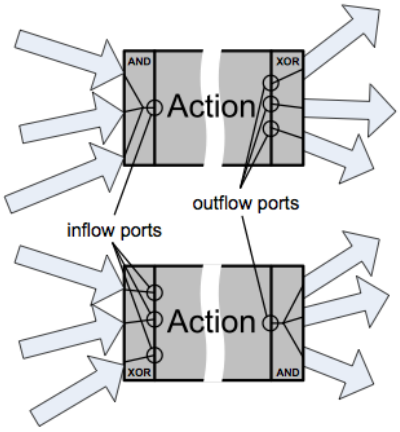

Figure III.2 : Ports de configuration des variants. Source [GVDAJVLR08]

Chaque variant est associé à une action configurable qui possède trois ports d'entrée et trois ports de sortie (cf. Figure III.2). Il existe plusieurs types de variants. Chaque type possédant des possibilités de sélection différentes en fonction de l'opérateur appliqué. Cet opérateur peut être un AND ou un XOR. Appliqué sur un groupe de ports, le type AND permet l'activation d'un seul port à la fois, alors que le XOR permet d'en activer plusieurs.

Les entrées et sorties possibles de chaque variant sont déterminées par la configuration des actions. Plusieurs configurations sont alors possibles. Comme le montre l'exemple illustré par la figure III.3, les ports sont des éléments qui peuvent être activés, bloqués, ou masqués :

- **port activé :** à l'état activé, le port permet le déclenchement de l'action associée au variant ;
- **port bloqué :** qu'il soit en entrée ou en sortie, le port ne permet aucune continuité du flux ;
- **masqué :** possible sur un port en entrée uniquement, il permet d'inhiber l'action du variant, qui est alors ignorée.

La configuration apporte une certaine flexibilité qui permet d'augmenter la réutilisabilité des

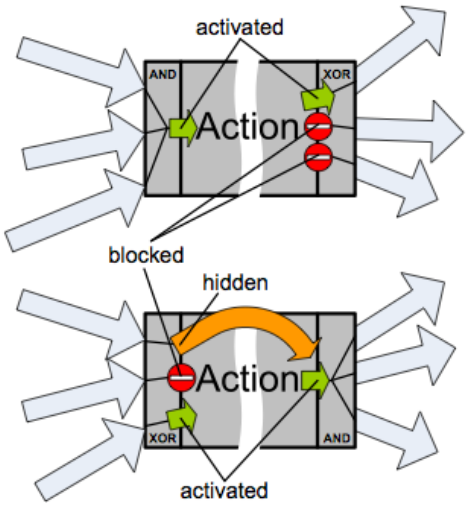

Figure III.3 : Configurations des actions d'entrée/sortie d'un variant

modèles de workflow. Cela reste cependant limité et n'apporte notamment pas de solutions pour l'adaptation à la volée. Ainsi, par exemple, les situations imprévues présentées dans l'exemple fil rouge ne pourront pas être traitées. Elles ne sont en effet connues qu'à l'exécution du modèle de l'activité. Or, cette approche permet de configurer le workflow à l'opérationnalisation. Une fois l'exécution lancée, aucune modification n'est possible.

Nous nous intéressons dans la section suivante à des approches d'adaptation à la volée dans le Workflow.

III.2.4.2 Adaptation à la volée dans le workflow

Les approches que nous présentons dans cette section traitent de l'adaptation à la volée. Elles ne couvrent pas la totalité du problème mais y répondent en partie. Elles s'attachent à la modification du modèle de workflow associé aux instances en cours d'exécution et traitent dans ce cadre de la migration de ces instances.

Adaptation par migration d'instances

L'adaptation par migration d'instance, aussi appelée *adaptation de type*, est le déplacement d'un ensemble d'instances d'un modèle source à un modèle destination. Le modèle destination est une nouvelle version obtenue par modification du modèle source (cf. Figure III.4).

La migration est alors conditionnée par le choix des instances qui seront impactées par les modifications dues à l'adaptation (ajout, suppression, modification d'éléments dans le modèle

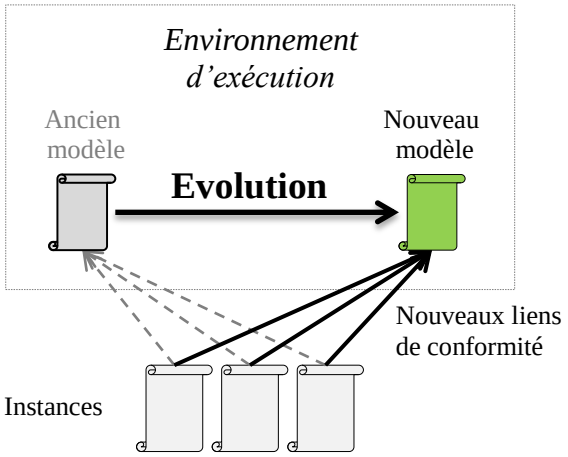

Figure III.4 : Migration d'instances d'un modèle de workflow source à un modèle de workflow destination

de workflow). [HJ98] ont identifié deux cas possibles, selon que les instances sont concernées ou pas par les modifications. Dans le premier cas (instances concernées par les modifications), il faut décider quelles instances seront impactées par les changements :

- les futures instances du modèle de workfow. Cela revient à faire de l'adaptation par conception itérative. Les modifications appliquées au modèle de workflow seront valables lors des prochaines exécutions ;
- les instances qui n'exécutent pas la partie qui doit être modifiée. La partie modifiée dans le modèle n'est pas (ou a déjà été) exécutée par ces instances.

Pour l'adaptation des instances, les auteurs considèrent plusieurs cas possibles selon les points d'exécution et de modification (cf. Chapitre II Section II.5.3). Ils considèrent alors l'évolution de l'exécution de l'instance du workflow (point de modification/point d'exécution) afin de décider si l'adaptation peut être appliquée ou pas. Trois cas sont à considérer en fonction du point d'exécution et du point de modification dans ce cas là :

- le point de modification est avant le point d'exécution : dans ce cas-là les instances peuvent continuer à s'exécuter et les modifications seront répercutées sur les futures instances du modèle de workflow ;
- le point de modification coïncide avec le point d'exécution. Sans mécanisme d'annulation, il est difficile d'apporter les modifications qui risquent de perturber l'exécution des instances ;
- le point de modification vient après le point d'exécution de l'instance. La modification est

possible parce qu'il n'y a pas de conflits entre l'exécution et la partie à modifier dans le modèle de workflow.

Remarque : la migration peut être faite en rattachant les instances au nouveau modèle cible. Ceci peut se faire en exploitant la relation *est une instance de* afin de déplacer les instances du modèle source au modèle cible.

La proposition présentée ci-dessus considère trois cas à identifier avant de migrer les instances du modèle source au modèle cible, mais cette migration est limitée, car elle ne traite pas le cas où le point de modification et le point d'exécution coïncident. C'est ce que nous avons appelé la temporalité en cours (cf. Chapitre II Section II.5.3). De plus, l'auteur suppose que les instances correspondent à des activités individuelles et indépendantes (les instances s'exécutent de manière indépendante). Il ne considère donc pas les aspects collaboratifs. Il considère aussi que le modèle de workflow est simple et ne contient pas de boucle, c'est-à-dire qu'un point d'exécution ne passe pas deux fois sur le même point de modification. De plus, les modifications étant limitées au modèle, cela exclut les situations qui requièrent de modifier les informations liées à l'opérationnalisation et à l'exécution. Les situations *apprenant en retard* et *ajout d'un apprenant* de l'exemple fil rouge (cf. Chapitre II) ne peuvent pas être traitées avec une telle approche.

Instanciation dynamique pour la migration

Dans ce type d'approche, l'instanciation du modèle de workflow se fait progressivement et dynamiquement pendant l'exécution. Par opposition à l'approche présentée dans la section précédente (cf. Section III.2.4.2), les instances sont partielles au début et sont complétées à l'exécution. Ceci permet d'instancier des sous-parties différentes du modèle de workflow en fonction de l'état de l'activité, des choix des utilisateurs, de leurs préférences, etc. L'instanciation dynamique permet donc d'adapter à la volée la partie du modèle de workflow qui n'a pas encore été instanciée.

[Wes99] propose un *Framework* qui implémente l'approche d'instanciation dynamique dans le contexte d'un projet nommé *WASA* [BMVW95]. Le *Framework* s'appuie principalement sur la vérification de la conformité des instances par rapport au modèle de workflow. En effet, cette conformité doit être préservée tout au long de l'exécution et notamment avant toute modification (adaptation) du modèle de workflow.

Chaque modification du modèle de l'activité est précédée par une suspension de l'exécution des instances concernées (instances différentes implique des points d'exécution différents). Le système vérifie alors si les instances sont toujours conformes au modèle obtenu une fois les modifications appliquées au modèle initial. Cette conformité dépend du point de modification et du point d'exécution des instances (cf. Chapitre II), en cherchant un mapping comme dans l'approche par migration d'instances (cf. III.2.4.2). La figure III.5 illustre un exemple avec deux instances i et j conformes à un modèle de workflow S (ou schéma). S' est le nouveau modèle de workflow obtenu par modification du modèle de workflow S.

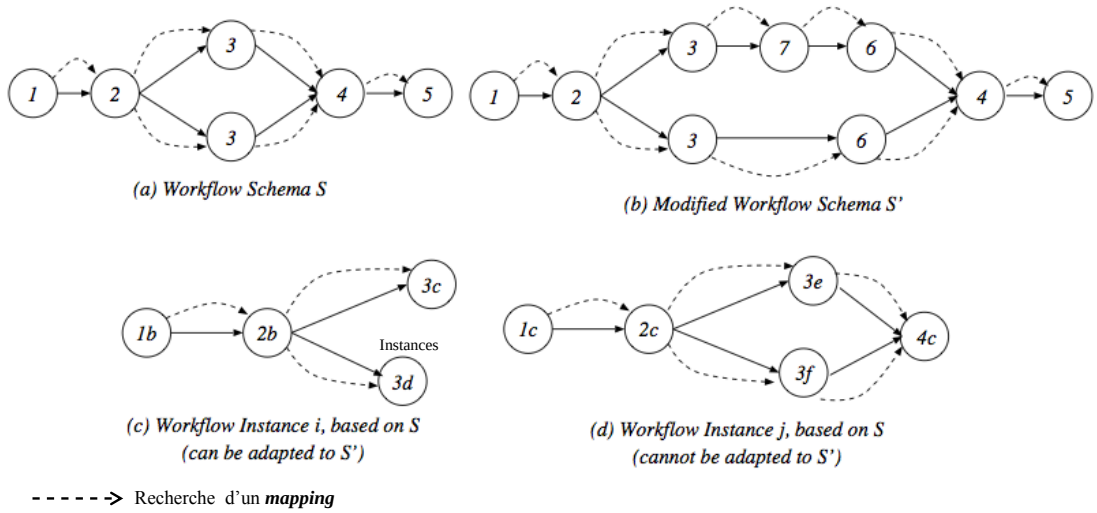

Figure III.5 : Instances i et j conformes à un modèle de workflow (ou schéma) S, que l'on cherche à transformer en un modèle de workflow S'. (Source [Wes10])

Pour savoir si les instances i et j peuvent être adaptées au nouveau modèle S', le système cherche un *mapping*. On dit qu'il y a un *mapping* si le sous-modèle déjà exécuté par l'instance est conforme avec le nouveau modèle obtenu après la modification. C'est par exemple le cas de l'instance i dont le point d'exécution est le nœud $3c$. L'instance j dont le point d'exécution est le nœud $4c$ ne possède pas de *mapping* possible avec le modèle S'.

Une fois les instances conformes repérées, le système procède à leur migration. L'implémentation étant basée sur la programmation orientée objet, la migration se fait par de simples modifications sur des objets. Dans le cas de l'exemple ci-dessus, il s'agit de modifier les relations

inter objets de type *instance de* existantes.

L'implémentation présentée ci-dessus exploite des avantages offerts par le fait que les modèles et leurs instances sont implémentés dans un contexte orienté objet. Les relations de type *instance-de* sont modifiables à la volée pour permettre la migration des instances. La recherche de *mapping* est aussi facilitée puisqu'elle est réduite à un calcul d'ensembles sur des nœuds représentés par des objets liés entre eux.

Cette proposition est intéressante et permet plus de flexibilité que l'approche de migration d'instances, mais elle reste limitée, car elle ne propose pas de solutions pour la migration des instances non conformes (aucun *mapping*) et qui ont un point d'exécution qui coïncide avec le point de modification. Par ailleurs, tout comme la précédente proposition (cf . Section III.2.4.2), elle considère que les activités sont indépendantes les unes des autres. Elle ne prend de fait pas en considération les situations où les instances doivent être synchronisées dans le contexte d'une activité collaborative. C'est le cas dans l'exemple fil rouge avec la situation de l'*apprenant en retard*, qui requiert de modifier les informations liées à l'exécution. La modification de l'activité de cet apprenant aura certainement un impact sur les activités des autres participants. Or, pour adapter cette situation en utilisant la proposition décrite ci-dessus, il faudrait intervenir avant que les apprenants n'atteignent le point de modification et donc anticiper sur le blocage de la situation, ce qui est dans la réalité impossible.

III.3 Conclusion sur l'adaptation dans le LD et dans le Workflow

Nous nous sommes intéressés dans ce chapitre à des approches d'adaptation proposées dans le LD et dans le Workflow. Ces approches sont regroupées selon la phase du cycle du vie du scénario durant laquelle sont effectuées les adaptations (cf. Figure III.6). Nous pouvons constater que les approches d'adaptation (prédéfinies et à la volée) proposées dans le LD sont rangées dans les mêmes catégories que celles proposées dans le Workflow. En effet, nous classons la proposition :

- de Zarraonandia et de *TENCompetence* dans les approches d'adaptation par modification ad-hoc d'instances ;
- de LAMS et de Perez-Rodriguez dans les approches d'adaptation par migration d'instances ;

– de Perez-Rodriguez dans les approches d'instanciation dynamique pour la migration.

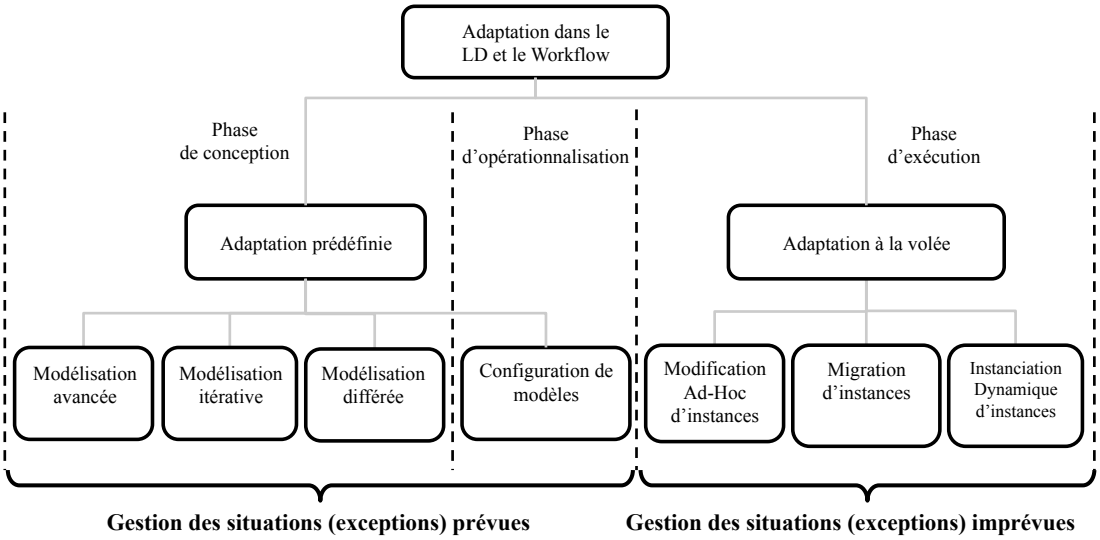

Figure III.6 : Positionnement des approches d'adaptation dans le cycle de vie du modèle de l'activité

La figure III.7 représente une classification des approches d'adaptations à la volée selon leur nature et la temporalité des situations imprévues qu'elles permettent de gérer. Nous pouvons ainsi constater que les approches présentées ne permettent pas de gérer toutes les natures possibles. Par exemple, l'adaptation par modification ad-hoc d'instances est la seule approche capable de supporter des situations qui requièrent de modifier le modèle (scénario ou workflow) et les informations liées à son opérationnalisation. Les autres approches sont limitées à la modification du modèle et dans une temporalité antérieure uniquement.

	Temporalité			**Nature**		
	Antérieure	**En cours**	**Postérieure**	**MO**	**OP**	**EX**
Modification Ad-Hoc d'instances	X		X	X		X
Migration d'instances	X			X		
Instanciation dynamique d'instances	X			X		

MO Modèle
OP Informations liées à l'opérationnalisation
EX Informations liées à l'exécution

Figure III.7 : Classification des approches d'adaptation à la volée selon leur nature et la temporalité des situations qu'elles gèrent

En plus de ne pas supporter les situations dont la temporalité est en cours, l'inconvénient des approches présentées est qu'elles ne considèrent pas les aspects collaboratifs dans l'activité. Elles ne supportent donc que des modifications sur des modèles qui s'exécutent de façon indépendante (sans synchronisation). Par ailleurs, quand les utilisateurs ont des progressions différentes, il faut disposer de moyens permettant de modifier leurs progressions. Notre point de vue est différent dans le sens où l'approche que nous proposons s'appuie sur les caractéristiques que nous avons dégagées dans le chapitre II et notamment la nature des adaptations. Selon cette nature, l'adaptation peut cibler de modifier le modèle de l'activité, les informations liées à son opérationnalisation ou les informations liées à son exécution. Nous présentons notre approche dans le chapitre suivant.

Chapitre IV

Première proposition : adaptation par propagation

Nous présentons dans ce chapitre notre première proposition pour l'adaptation à la volée. Ce chapitre est organisé comme suit :

Dans la section IV.1, nous présenterons notre approche pour l'adaptation à la volée en nous appuyant sur les exemples de situations proposés dans le cadre du workshop AAMCS.

La section IV.2 sera consacrée à la description du prototype qui implémente notre approche d'adaptation à la volée. Nous montrerons également comment s'applique l'approche proposée à travers la présentation du déroulement d'un exemple d'adaptation.

IV.0.1 Situations imprévues proposées dans l'étude de cas

Ces *situations imprévues* sont des exemples de cas particuliers pouvant être rencontrés lors de l'exécution du scénario de l'étude de cas :

- **un apprenant en retard :** dans cette situation imprévue, un apprenant qui n'arrive pas à se connecter à l'activité bloque ses camarades qui l'attendent pour pouvoir accéder à la troisième étape du scénario. Une situation identique a été présentée dans le cadre de l'exemple présenté dans la section II.3.2 ;
- **ajouter un nouveau participant à l'activité :** l'enseignant veut inviter un nouvel apprenant à participer à l'activité alors que le scénario est déjà en cours d'exécution ;
- **inviter un expert à l'activité :** l'enseignant veut inviter un *expert en environnement* à rejoindre l'activité pour répondre aux questions des apprenants. Celui-ci pourra par

exemple consulter les réponses données par les apprenants et réagir en conséquence. L'expert aimerait pouvoir compter sur des images, des vidéos et des témoignages sonores afin de sensibiliser les apprenants par rapport à la protection de l'environnement.

Nous reviendrons sur l'adaptation à la volée des trois situations présentées ci-dessus une fois que nous aurons fait notre proposition dans la section suivante.

IV.1 Première proposition : adaptation par propagation

Notre première proposition pour répondre à la question de l'adaptation à la volée s'appuie sur la caractérisation que nous avons proposée dans le chapitre II. Nous avons en effet proposé de caractériser chaque situation imprévue par sa nature qui peut alors concerner :

- le scénario qui est aussi le modèle qui décrit la situation d'apprentissage ;
- les informations liées à l'opérationnalisation de ce scénario ;
- les informations liées à l'exécution de ce scénario.

IV.1.1 Hypothèse : l'adaptation à la volée doit prendre en compte la nature de la situation imprévue

Dans notre approche nous considérons que l'adaptation à la volée dépend de la nature de la situation imprévue. Il convient alors d'orienter les modifications en fonction des informations liées à la nature de la situation imprévue qui doit être traitée :

- le scénario pédagogique ;
- les informations liées à l'opérationnalisation du scénario ;
- les informations liées à l'exécution du scénario.

Nous pouvons par conséquent distinguer les adaptations selon les informations qu'elles ciblent.

IV.1.2 Niveaux des informations liées à l'exécution du scénario

Les informations liées à la nature de la situation imprévue peuvent être mises en relation avec les différentes phases du cycle de vie du scénario (cf. Chapitre I). Nous utiliserons le terme *niveaux* pour désigner une organisation verticale de ces informations. Nous considérerons par conséquent que ces informations sont organisées en trois niveaux :

- **le niveau conception :** ce niveau porte les informations liées à l'étape de conception du modèle du scénario ;

- **le niveau opérationnalisation :** ce niveau porte les informations liées à l'opérationnalisation du scénario. Ces informations permettent de répondre aux questions suivantes :
  - quels utilisateurs jouent quels rôles,
  - quelles ressources (Ex. une image, un didacticiel, etc.) et quels services (Ex. un chat, un forum de discussion) sont associés à l'exécution du scénario,
- **le niveau exécution :** ce niveau porte les informations liées à l'exécution du scénario. Ce sont des informations qui permettent de connaitre la progression de chaque participant de l'activité. Elles n'existent qu'à l'exécution du scénario et elles représentent l'état de l'activité. Les réponses formulées par les apprenants sont des exemples d'informations que porte ce niveau.

La figure IV.1 montre cette organisation en niveaux par rapport au cycle de vie du scénario. Nous désignerons par *niveau supérieur* le niveau conception et par *niveaux inférieurs* les niveaux opérationnalisation et le niveau exécution. Le niveau opérationnalisation est le niveau inférieur *direct* du niveau conception, et le niveau exécution est le niveau inférieur direct du niveau opérationnalisation.

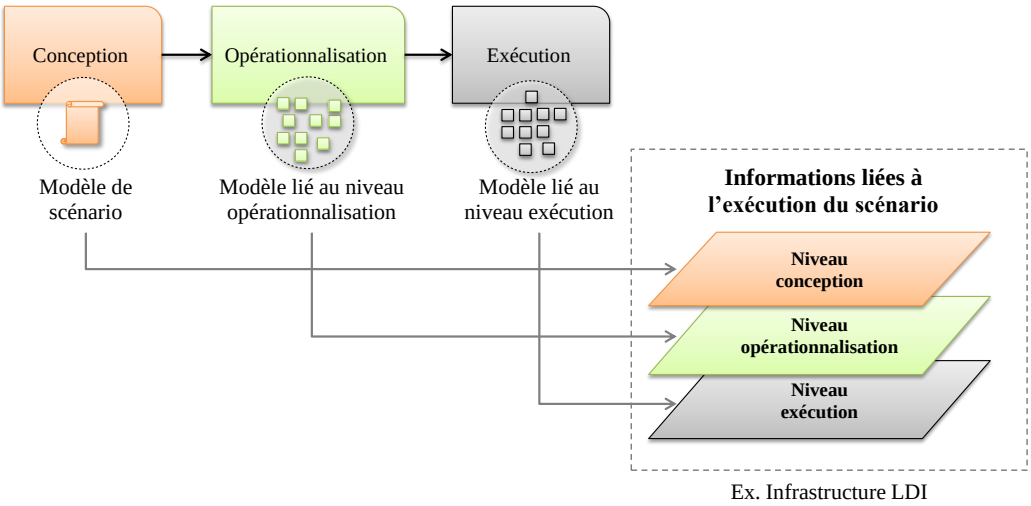

Figure IV.1 : Organisation en niveaux des informations liées à l'exécution d'un scénario

IV.1.3 Relations entre les niveaux d'une activité

En plus de leur organisation verticale, les niveaux d'une activité sont liés. Ces liens sont nécessaires à l'exécution du scénario. Exemple d'un scénario LDL qui s'exécute dans LDI :

- les *rôles* sont définis dans le scénario et donc au niveau conception ;

– les *utilisateurs* qui jouent ces *rôles* et participent à l'activité, sont définis au niveau opérationnalisation ;
– les progressions de ces *utilisateurs* sont dans le niveau exécution.

La figure IV.2 montre certains de ces liens. Dans le cas de l'exécution d'un scénario LDL, ces liens représentent des références qui identifient et font la correspondance entre les éléments de chaque niveau.

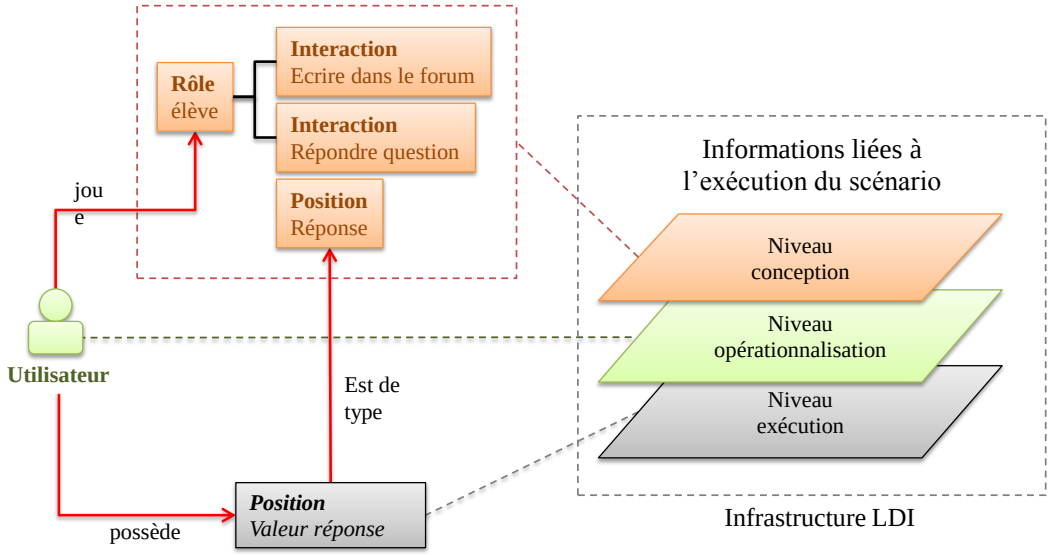

Figure IV.2 : Exemple de liens entre les trois niveaux d'une activité se déroulant conformément à un scénario LDL

Dans l'exemple de la figure IV.2 :

– **le lien *joue* :** permet de connaître les interactions que peut jouer l'utilisateur qui a le rôle *learner* ;
– **le lien *possède* :** permet de faire l'association entre cet utilisateur et les valeurs de ses positions dans l'activité (réponses, choix, préférences, etc.) ;
– **le lien *est de type* :** permet de connaître la sémantique de la position LDL dans le scénario.

IV.1.4 Niveau d'une adaptation à la volée

Maintenant que nous avons introduit l'organisation en niveaux des informations liées à l'exécution d'un scénario, nous nous focalisons à nouveau sur l'adaptation à la volée. Comme la

situation imprévue, l'adaptation à la volée peut être caractérisée par les informations qu'elle cible. En effet, il est possible de caractériser chaque adaptation à la volée par le niveau d'informations qu'elle doit modifier. Nous dirons alors qu'une adaptation à la volée est :

- **de niveau conception :** si elle nécessite de modifier les informations liées à la phase de conception du scénario, comme :
  - l'ajout d'un nouveau rôle,
  - la suppression d'une interaction pour restreindre le champ d'actions d'un rôle donné,
  - l'ajout d'une nouvelle règle,
- **de niveau opérationnalisation :** si elle nécessite de modifier les informations liées à la phase d'opérationnalisation, comme :
  - l'ajout d'un nouveau participant durant l'exécution du scénario,
  - la substitution d'une ressource, un service ou un outil choisis lors de la phase d'opérationnalisation,
- **de niveau exécution :** si elle nécessite de modifier les informations liées à la phase d'exécution, comme :
  - la modification de la progression d'un participant,
  - la modification de la valeur de la position d'un participant.

IV.1.5 Propagation d'une adaptation à la volée

Etant donnés les liens qui existent entre les niveaux associés à l'exécution d'un scénario, une modification qui cible une information appartenant à un niveau N_i pourrait avoir un impact sur les informations d'un autres niveau N_j. La modification d'informations de niveau conception pourrait par exemple nécessiter la modification d'informations de niveau opérationnalisation ou exécution. Nous parlerons alors d'*adaptation par propagation.*

Dans l'exemple de situation imprévue *pas de rôle Expert* proposée dans l'étude de cas (cf. Chapitre II), l'ajout du rôle expert au scénario (niveau conception) nécessite une propagation sur le niveau *opérationnalisation*, car le rôle introduit doit en plus être affecté à l'expert. Dans le cas contraire, l'expert ne pourra pas participer à l'activité.

Sens d'une propagation : considérons une situation imprévue S_i de niveau N_i (N_i = Conception, Opérationnalisation ou Exécution) et son adaptation à la volée A de niveau N_i. Si au moment d'appliquer les modifications de cette adaptation, il s'avère que celles-ci sont insuffi-

santes et qu'il faut modifier en plus des informations d'un autre niveau disons N_j alors :

- si $N_i > N_j$: il faut une propagation et pour cela les modifications effectuées sur le niveau N_i doivent être propagées sur le niveau N_j et donc sur un niveau inférieur (opérationnalisation ou exécution) ;
- si $N_i < N_j$: le niveau de S_i et de A n'est pas N_i mais plutôt N_j. Autrement dit, l'adaptation A_i est de niveau supérieur (opérationnalisation ou conception). Il faut par conséquent identifier les informations à modifier dans le niveau N_j.

La propagation se fait toujours d'un niveau supérieur à un niveau inférieur. Dans notre cas du niveau conception au niveau opérationnalisation ou/et exécution, ou alors du niveau opérationnalisation au niveau exécution. La figure IV.3 montre le sens des propagations possibles entre les trois niveaux.

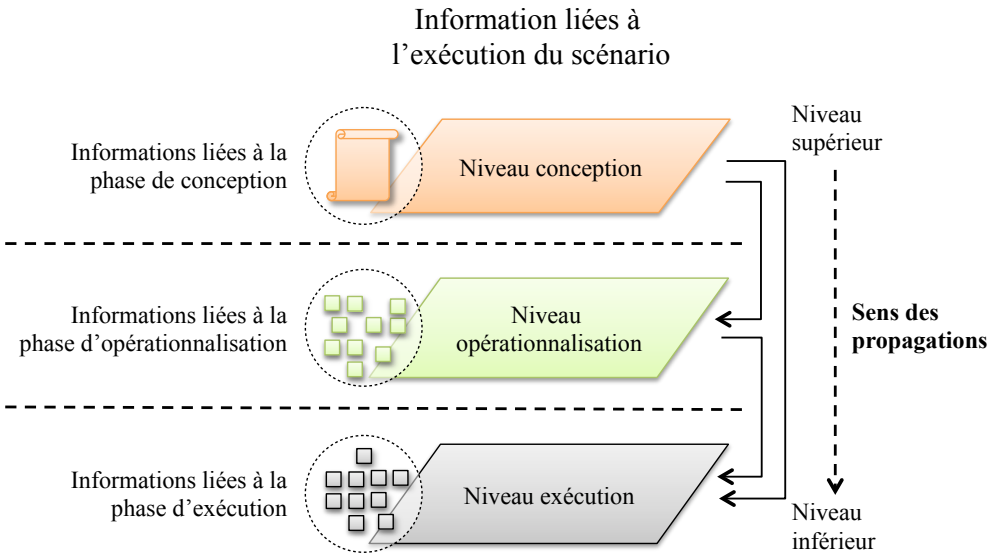

Figure IV.3 : Sens de propagation des modifications d'une adaptation à la volée

IV.1.5.1 Processus d'une adaptation par propagation

Pour adapter une situation imprévue, il est nécessaire de définir l'adaptation à la volée associée à travers les étapes suivantes :

1. identifier le niveau de l'adaptation à la volée et les informations qui doivent être modifiées dans celui-ci ;
2. définir les modifications à apporter aux informations identifiées ;

3. déterminer s'il est nécessaire de faire des propagations à partir du niveau identifié précédemment. Si oui alors :
   - identifier le ou les niveaux concernés et les informations à modifier,
   - définir les modifications à apporter aux informations identifiées.

Les étapes proposées ci-dessus peuvent être itérées de manière récursive sur chacun des niveaux conception, opérationnalisation, jusqu'à l'identification de toutes les propagations requises.

IV.1.6 Adaptation par propagation complète

Une adaptation à la volée dont la propagation est complète est une adaptation dont les modifications portent sur les trois niveaux : conception, opérationnalisation et exécution. Cette propagation commence au niveau conception (supérieur) et se termine au niveau exécution (inférieur) en passant par le niveau opérationnalisation.

Exemple de propagation complète : nous avions évoqué précédemment l'exemple de situation imprévue *pas de rôle expert*. Contrairement à ce qui est proposé dans l'exemple de l'étude de cas (cf. Chapitre II), supposons que l'expert soit disponible pour une durée limitée et qu'il ne peut attendre que les apprenants aient fini de répondre aux QCM (deuxième étape du scénario). Il faudrait alors mobiliser les apprenants et les déplacer directement à la troisième étape pour qu'ils puissent discuter sans attendre avec l'expert. Dans une activité modélisée par un scénario LDL, l'adaptation à la volée par propagation complète de la situation *pas de rôle expert* doit passer par les modifications suivantes :

1. **au niveau conception :** modifier le scénario en y ajoutant un nouveau rôle *Expert* et lui affecter l'interaction (existante) qui lui permettra de discuter avec les apprenants dans la troisième étape (discussion) ;
2. **propagation sur le niveau opérationnalisation :** à ce niveau il faut affecter le nouveau rôle *expert* à l'utilisateur expert en environnement ;
3. **propagation sur le niveau exécution :** à ce niveau il faut modifier les progressions des apprenants pour leur permettre d'aller directement à la troisième étape. Avec le couple LDL et LDI, ceci peut être effectué en modifiant la visibilité des interactions pour les apprenants. Il faut donc leur permettre de voir l'interaction qui leur permettra d'aller directement à la troisième étape de discussion.

Les seules modifications possibles dans LDI sont celles qui concernent les informations liées à l'opérationnalisation du scénario LDL. Il n'y a donc aucun moyen pour modifier les informations des autres niveaux. Pour répondre à ce besoin, nous avons implémenté un prototype qui étend LDI pour expliciter et modifier les informations liées au niveau conception (scénario) et au niveau exécution (progressions des apprenants). Nous présentons ce prototype dans la section suivante.

IV.2 Un premier prototype pour l'adaptation par propagation

Nous avons implémenté un premier prototype pour permettre les modifications liées à cette adaptation par propagation. Ce prototype implémenté dans LDI, permet d'expliciter et de modifier les informations liées aux niveaux conception et exécution. Il constitue une extension de l'interface d'opérationnalisation existante (cf. Chapitre I).

IV.2.1 Architecture du prototype

Nous allons présenter l'architecture du prototype dont nous avons fait une démonstration lors du workshop AAMCS pour l'adaptation à la volée de l'étude de cas proposée (cf. Chapitre II).

LDI ayant été développé selon les technologies ZOPE [Zop11] et *Plone* [Plo11], les extensions ont été ajoutées sous forme de produits [1]. Ces extensions sont représentées par les modules *1* et *2* dans la figure IV.4.

Rôle du module 1 : ce module permet de modifier un scénario en cours d'exécution. Dans le cas de l'exemple *ajouter rôle expert*, l'implémentation du prototype est limitée à l'ajout de nouveaux concepts au scénario. D'un point de vue conceptuel, ce module offre des fonctionnalités de base pour instancier, initialiser et configurer des concepts LDL.

Rôle du module 2 : ce deuxième module permet de visualiser et de modifier les informations liées à l'exécution d'un scénario. Par *visualiser* nous sous entendons l'observation en temps réel des progressions des participants. Pour connaître ces progressions, nous utilisons les informations liées à l'état de visibilité des interactions (structures) des participants. Une interaction a en effet un état qui permet de savoir si elle est *démarrée*, *terminée* ou *visible* pour à un participant donné, avec un rôle donné dans l'activité. En plus de permettre l'observation du niveau exécution, ce

1. module utilisé dans le serveur d'application Zope

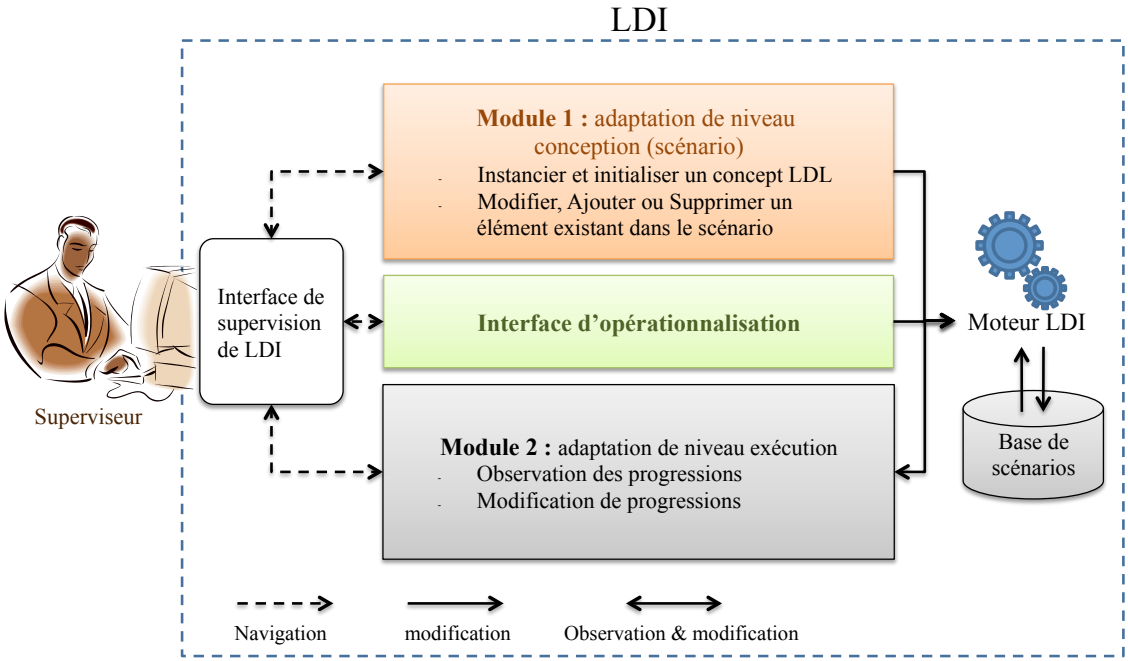

Figure IV.4 : Architecture du prototype d'adaptation par propagation

module permet d'en modifier les informations. LDI sauvegarde un état de visibilité (booléen : vrai = visible, faux = invisible) pour chaque interaction (structure) et pour chaque rôle et participant de l'activité. Nous pouvons définir la visibilité d'une interaction i par V_i. La valeur de V_i est déterminée par une fonction de la forme $V(i, u, r)$ avec :

- i : une interaction du scénario ;
- u : un utilisateur de l'environnement d'exécution ;
- r : un rôle affecté à u.

Ce module permet donc de modifier la visibilité des interactions en utilisant une fonction de la forme $forceV(i, u, r, vf)$ avec :

- i : une interaction du scénario ;
- u : un utilisateur de l'environnement d'exécution ;
- r : un rôle affecté à l'utilisateur u (le couple (u, r) représente un participant) ;
- vf : nouvelle valeur de visibilité pour l'interaction i. Si vf est à *vrai*, l'interaction i devient visible dans le player de l'utilisateur u par rapport au rôle r. Elle est masquée dans le cas contraire.

Des fonctions similaires ont été implémentées pour permettre à ce module de gérer le démar-

rage et l'arrêt des interactions (structures).

IV.2.2 Exemple d'utilisation du prototype

Revenons sur l'exemple de situation imprévue *pas de rôle Expert*, proposée dans l'étude de cas du *Workshop AAMCS*. Nous allons montrer comment effectuer une adaptation par propagation en utilisant le prototype proposé.

La situation imprévue *par de rôle Expert* nécessite une adaptation à la volée de niveau conception avec une propagation complète. Nous allons décrire tout le processus d'adaptation à la volée, de la modification du niveau conception à la modification du niveau exécution.

Afin d'effectuer cette adaptation, nous avons déterminé l'ensemble des modifications requises pour le niveau conception et calculé les propagations sur les autres niveaux opérationnalisation et exécution :

Modification du niveau conception : dans cette situation nous souhaitons ajouter un nouveau rôle *Expert* au scénario de l'étude de cas. Pour effectuer cette modification nous utilisons le *module 1* (cf. Figure IV.4). D'un point de vue conceptuel, ce module permet :

- de supprimer un élément du scénario ;
- de modifier les propriétés d'un élément du scénario ;
- d'ajouter un élément au scénario LDL.

Nous avons implémenté la fonction qui permet d'instancier un nouveau concept et de l'ajouter au scénario en cours d'exécution. Dans notre cas c'est l'action *ajouter* qui permet d'instancier et d'ajouter une instance du concept *Rôle LDL* au scénario. Une fois le rôle ajouté, il faut le configurer en choisissant ses interactions dans le scénario. Dans notre cas, il s'agit d'une interaction qui existe déjà et qui permet de discuter avec les apprenants dans la troisième étape du scénario (cf. Figure IV.5).

Adaptation de niveau opérationnalisation : cette modification peut être faite directement à travers l'interface d'opérationnalisation de LDI (cf. Figure IV.6). Il s'agit d'affecter le nouveau rôle *Expert* à l'utilisateur expert dans LDI. Une fois cette affectation effectuée, l'expert peut se connecter et participer à l'activité. Il aura par conséquent un accès direct à la troisième étape

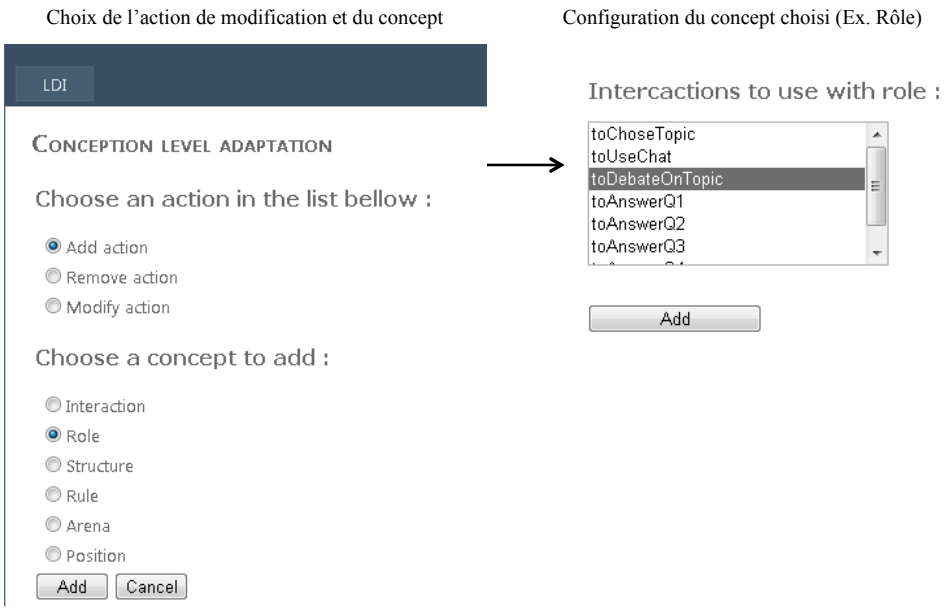

Figure IV.5 : Modification du niveau conception dans *module 1*

du scénario à travers l'interaction *Discuss in the chat*.

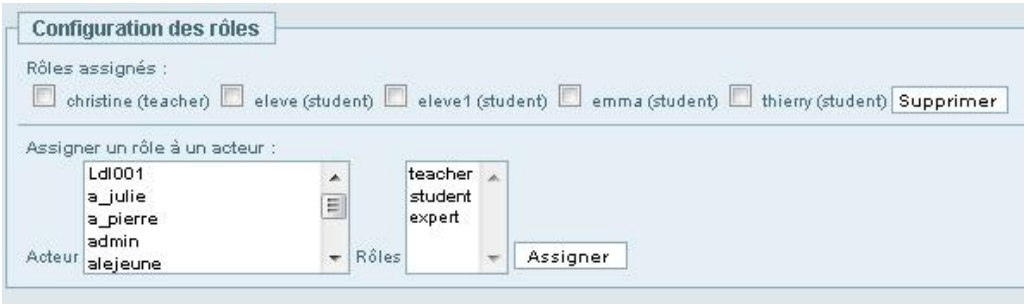

Figure IV.6 : Modification du niveau opérationnalisation dans LDI

Adaptation de niveau exécution : au niveau exécution, les modifications concernent les informations liées aux visibilités des interactions des apprenants. Comme nous l'avons précisé lors de la présentation de l'exemple de propagation complète (cf. Section IV.1.6), nous supposons que l'expert est disponible pour une période restreinte qui débute alors que les apprenants sont toujours en train de répondre aux QCM. Nous avons donc modifié les valeurs de visibilités de leurs interactions pour leur permettre d'accéder directement à la troisième étape et de discuter avec l'expert qui les attend. La figure IV.7 montre l'interface du *module 2* pour l'observation et

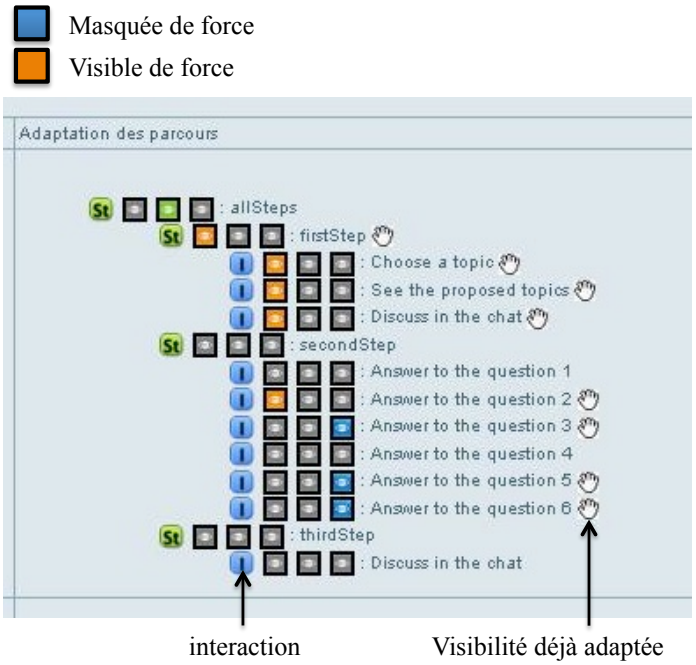

Figure IV.7 : Interface du *module 2* : observation et d'adaptation des informations liées au niveau exécution

la modification des valeurs de visibilité des interactions. La figure IV.8 montre l'état de l'interface du module 2 après l'adaptation. Nous pouvons constater l'interaction *Discuss in the chat* qui devient visible (bleu).

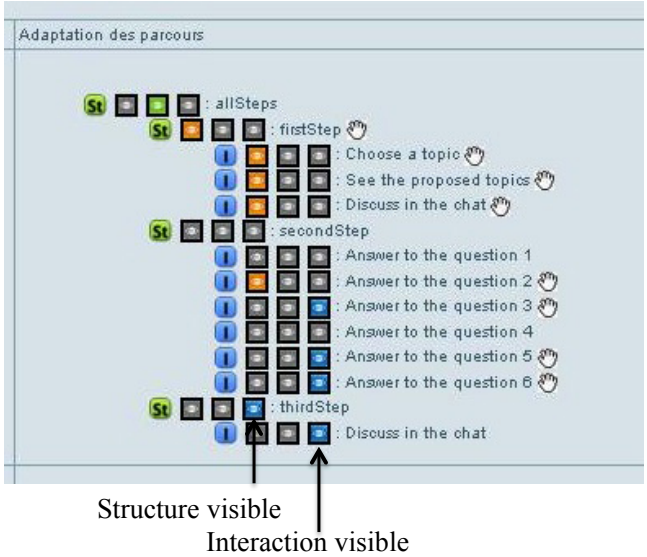

Figure IV.8 : Etat de visibilité de l'interaction *Discuss in the chat* après l'adaptation

Lors de la modification de la visibilité d'une interaction (ou structure), le superviseur peut choisir la portée de l'adaptation et cocher une case qui désigne *le rôle concerné*, *un groupe de participants* ou *tous les participants*. Dans notre exemple, ce sont tous les participants qui ont le rôle *learner* qui sont concernés par l'adaptation.

Nous utilisons le module 2 (cf. Figure IV.4) pour forcer la visibilité de l'interaction *Discuss in the chat* qui permet d'aller à la troisième étape. La figure IV.9 montre les deux états du player d'un apprenant, avant et après l'adaptation. Une fois l'interaction affichée pour les apprenants, il faut leur envoyer une notification pour les prévenir (en utilisant un chat de discussion par exemple, et leur demander de choisir la nouvelle interaction visible dans leurs *players* respectifs.

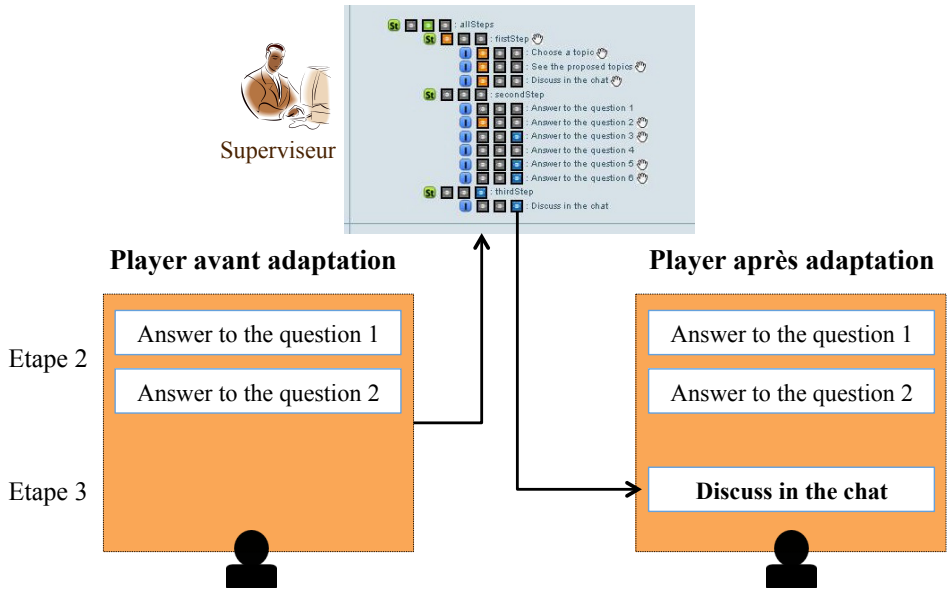

Figure IV.9 : Modification de la visibilité des interactions dans le *player* des participants

IV.3 Conclusion

Nous avons proposé dans ce chapitre notre approche pour l'adaptation à la volée d'une activité se déroulant conformément à l'exécution d'un scénario. Dans cette approche, nous avons proposé une organisation verticale et en niveaux des informations liées à l'exécution d'un scénario :

- le niveau conception représenté par le scénario ;
- le niveau opérationnalisation représenté par les informations liées à la phase d'opérationnalisation du scénario ;

– le niveau exécution représenté par les informations liées à la phase d'exécution du scénario.

Nous avons par ailleurs mis en évidence des liens entre les informations de chaque niveau et avons soulevé la nécessité de procéder à des modifications par propagation.

Nous avons également implanté l'approche proposée dans un prototype qui étend l'infrastructure LDI pour l'adaptation à la volée par propagation. Cependant, ce prototype n'implémente pas complètement l'approche d'adaptation par propagation, car il n'explicite pas toutes les informations liées à l'exécution du scénario et notamment les informations liées à la phase d'opérationnalisation. De plus, le prototype offre la possibilité d'adapter par propagation, mais n'exploite pas les liens entre les niveaux pour cadrer et assister le superviseur durant l'adaptation. Dans le prochain chapitre, nous présenterons une deuxième proposition répondant à ce problème. Nous expliquerons alors comment nous nous sommes appuyés sur l'Ingénierie Dirigée par les Modèles pour mettre en œuvre un Framework indépendant de LDI pour l'adaptation par propagation.

Chapitre V

Deuxième proposition : utilisation de l'IDM pour l'adaptation par propagation

Nous avons présenté dans le chapitre précédent (cf. Chapitre IV) une approche pour l'adaptation à la volée d'une activité qui se déroule conformément à l'exécution du scénario pédagogique. Dans l'approche proposée, nous avons montré que l'exécution du scénario dépend de trois niveaux d'informations et que l'adaptation à la volée consiste en des modifications séparées des informations contenues dans chaque niveau. Nous avons montré également que la modification de certains niveaux requiert des propagations que nous avons illustrées à travers l'adaptation d'une situation imprévue en utilisant le prototype que nous avons implémenté.

Nous avons par ailleurs évoqué certains problèmes liés à l'implémentation du prototype proposé, car ce dernier :

- est complètement dépendant de l'infrastructure LDI ;
- n'explicite pas toutes les informations liées aux trois niveaux : conception, opérationnalisation et exécution ;
- n'explicite pas les liens existants entre les informations de chaque niveau, ce qui rend difficile le calcul des propagations. En effet, en explicitant ces liens, il sera possible de trouver les dépendances entre les différents niveaux. Ces dépendances serviront ensuite à identifier les éléments qui doivent être modifiés lors des propagations. Par exemple, si l'élément B qui appartient à un niveau N_B dépend d'un autre élément A qui lui appartient à un niveau N_A, alors si A est modifié par une adaptation, l'élément B devrait être impacté

et une propagation du niveau N_A au niveau N_B devra être effectuée ;
- n'offre pas de moyen permettant de cadrer les modifications apportées aux différents niveaux ;
- ne permet pas de garder les traces des modifications effectuées.

Pour répondre aux limites identifiées ci-dessus, nous avons mis en œuvre un *Framework* qui s'appuie sur l'IDM pour l'adaptation par propagation. L'IDM a fourni les concepts et les outils qui nous ont permis d'implémenter des méta-modèles pour :
- spécifier les informations liées aux niveaux opérationnalisation et exécution ;
- spécifier les liens entre les informations des différents niveaux ;
- construire des outils de visualisation des modèles de chaque niveau ;
- manipuler les informations à partir de leurs modèles dans un outil découplé de l'infrastructure LDI ;
- modéliser les adaptations et en garder une trace lisible et ré-utilisable.

Dans la section V.1, nous présentons l'hypothèse selon laquelle la référence de chacun des niveaux - conception, opérationnalisation, exécution - à son méta-modèle et l'expression des relations inter méta-modèles doit permettre l'adaptation par propagation. La section V.2 donne une brève description du rôle de l'IDM dans notre proposition et notamment les concepts que nous avons exploités pour l'implémentation du *Framework*. Les sections V.3 et V.4 seront dédiées à la description des méta-modèles proposés dans notre approche. Dans la section V.5 nous présentons un autre méta-modèle qui permet de spécifier des modifications pouvant être utilisées dans une adaptation par propagation. Dans la section V.6, nous présenterons le *Framework* qui implémente ces méta-modèles et son utilité pour l'adaptation par propagation. Enfin, dans la section V.6.3, nous présenterons une validation de l'approche proposée en utilisant le *Framework* implémenté à travers un exemple d'adaptation.

V.1 Hypothèse : un méta-modèle pour chaque niveau

Les informations liées à chaque niveau peuvent être considérées comme des instances de concepts décrits par un méta-modèle propre à ce niveau.

Exhiber les méta-modèles de chaque niveau devrait permettre d'une part, d'introduire des outils pour créer, manipuler et visualiser des modèles représentant

les informations liées à l'exécution d'un scénario, et d'autre part, d'introduire un méta-modèle pour spécifier les adaptations à l'aide de modèles.

Notre hypothèse consiste donc à ramener l'adaptation à la volée et notamment l'adaptation par propagation (cf. Chapitre IV) à un ensemble de modifications sur des modèles, à l'aide d'un méta-modèle. Les modèles instanciés à partir de ces méta-modèles seront utilisés comme des leviers pour manipuler et modifier les informations liées à l'exécution du scénario.

Les méta-modèles introduits sont présentés dans la figure V.1. Comme nous pouvons le constater sur cette figure, à chaque phase du cycle de vie du scénario, correspond un méta-modèle. Nous avons donc étendu l'utilisation des méta-modèles à la phase d'opérationnalisation et la phase d'exécution, le méta-modèle associé à la phase de conception étant celui du langage choisi (dans notre cas LDL).

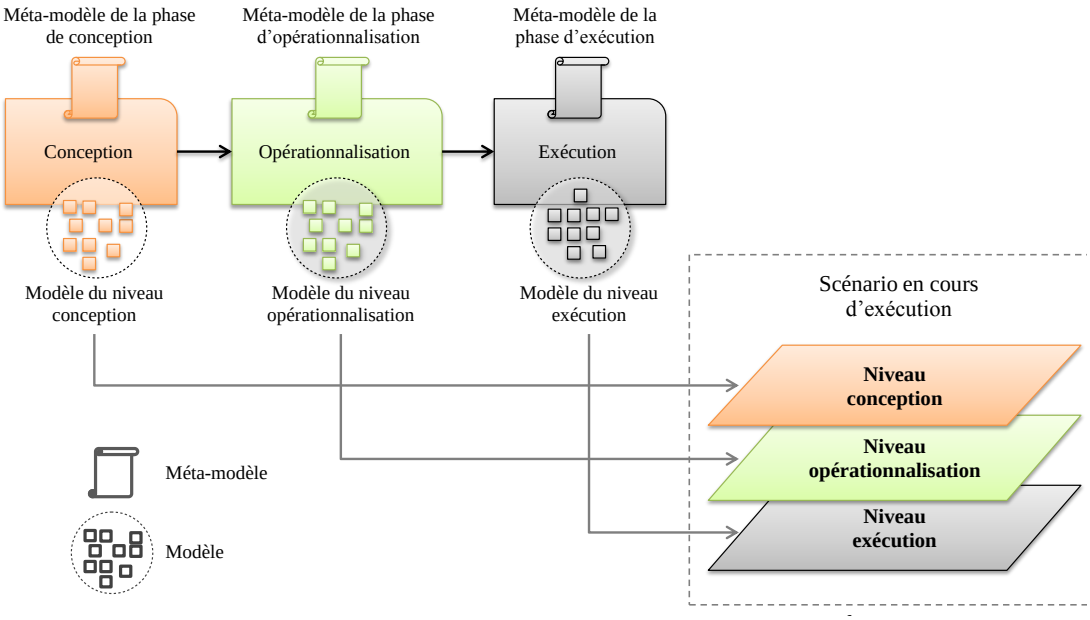

Figure V.1 : Positionnement des Méta-modèles liés à l'opérationnalisation et à l'exécution dans le cycle de vie d'un scénario LDL

Dans la section suivante, nous présenterons les concepts de l'IDM que nous avons exploités dans notre approche.

V.2 Rôle de l'Ingénierie Dirigée par les Modèles

L'Ingénierie Dirigée par les Modèles est déjà utilisée dans le cycle de vie du scénario et de manière plus générale dans la production des logiciels. C'est un paradigme qui propose des modèles, des outils et une approche spécifique permettant de théoriser sur les manipulations de modèles par le biais de transformations successives pour la génération de code. L'IDM propose d'étendre et de renforcer l'utilisation de modèles et de méta-modèles à travers une nouvelle vision dont l'objectif majeur est de rassembler des techniques et des approches connues dans le domaine de la modélisation. Certaines communautés en Informatique et notamment l'Ingénierie des EIAH s'appuient sur cette approche.

Le *Framework* que nous avons proposé se base sur l'IDM pour permettre de représenter les informations et concevoir des outils permettant de manipuler les informations que nous ciblons indépendamment des mécanismes spécifiques à l'infrastructure LDI. Les concepts que nous avons exploités sont les suivants :

- **le modèle et la relation *Représente* :** un modèle est une représentation abstraite d'un objet de la réalité. La relation *Représente*, notée μ, lie le modèle à l'élément modélisé. Elle se limite alors à certaines propriétés de l'élément modélisé. Par exemple, dans un scénario LDL, l'enceinte *représente* (modélise) le lieu ou se déroule une interaction ;
- **Le méta-modèle et la relation *EstConformeA* :** un méta-modèle est un modèle qui définit un langage de modélisation. Le concept de méta-modèle mène a une relation notée χ et appelée **EstConformeA**. Par exemple, tout scénario est conforme (dans notre approche) au méta-modèle conceptuel du langage de modélisation utilisé (EML). La relation *EstConformeA* est définie dans [Fav05] ;
- **la relation *EstTransforméEn* :** notée τ, la relation *EstTransforméEn* permet d'exprimer un lien entre un concept appartenant à un *modèle source* et un autre concept d'un autre *modèle cible* à des fins d'automatisation. La relation **EstTransforméEn** consiste donc à faire la *mise en correspondance* entre des éléments appartenant à des modèles différents. Les transformations sont exprimées à l'aide de *langages de transformation* comme ATL [BDJ+03] ou QVT [OMG05] ;
- **le *niveau d'abstraction* d'un modèle :** en général plus on se trouve proche de la plate-forme d'exécution, moins on est abstrait [BB02]. L'*ingénierie directe* est une transformation (relation τ) qui permet de réduire le niveau d'abstraction d'un modèle, c'est la technique utilisée lors de l'opérationnalisation d'un scénario sur une plate-forme. L'*ingénirie*

inverse permet au contraire, d'augmenter le niveau d'abstraction, en passant du code source à un modèle plus abstrait.

Plus de détails sur ces concepts sont présentés dans [BDJ+03, SK03, JAB+06].

V.3 Méta-modèle lié à la phase d'opérationnalisation

Le méta-modèle lié à la phase d'opérationnalisation permet de représenter les informations liées au niveau opérationnalisation. C'est dans ce méta-modèle que nous retrouverons par exemple les informations liées à l'infrastructure cible, comme les utilisateurs, les ressources (Ex. une image, un didacticiel, etc.) et les services (Ex. un chat, un forum de discussion). En plus de spécifier ces informations, ce méta-modèle définit des liens avec le niveau conception. La figure V.2 présente le méta-modèle lié à l'opérationnalisation d'un scénario LDL.

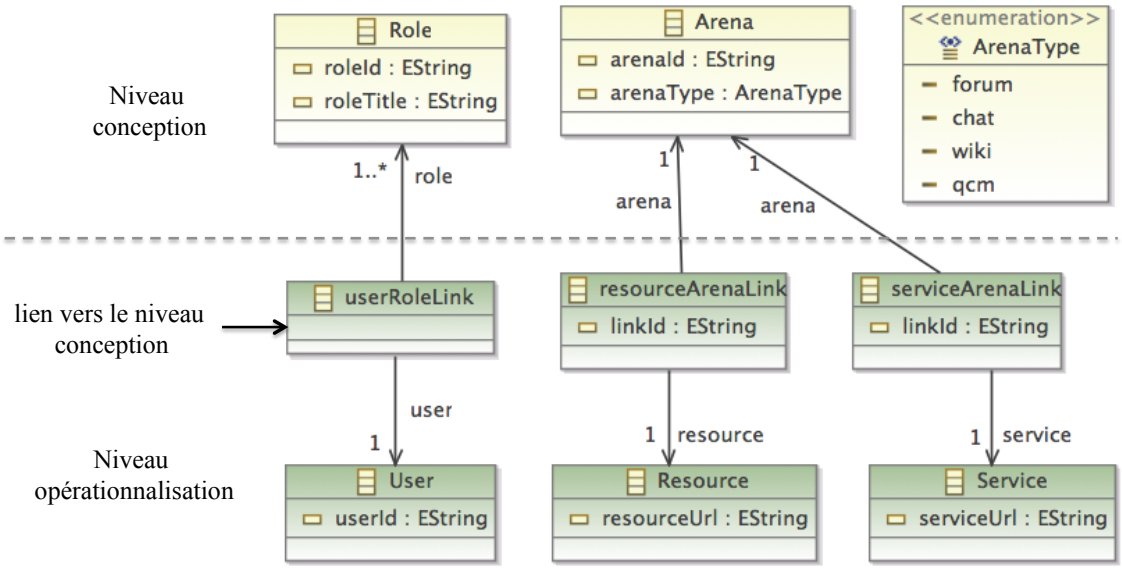

Figure V.2 : Méta-modèle lié à la phase d'opérationnalisation d'un scénario LDL dans LDI

Ce méta-modèle définit les méta-concepts : *Resource*, *Service* et *User* et les méta-concepts qui les lient au niveau conception : *userRoleLink*, *resourceArenaLink*, *serviceArenaLink*. Par exemple, un concept $< Role : enseignant >$ instance du méta-concept *Role*, peut être associé à un concept $< User : stephane >$ instancié à partir du méta-concept *User*. Le concept $< User : stephane >$ (stephane étant l'identifiant d'un utilisateur) sera alors porteur des propriétés de son méta-concept et permettra l'identification de l'utilisateur et des rôles qui lui sont

affectés à travers les liens *userRoleLink*.

De la même manière, un concept < *Arena* : *chat* > instancié à partir du méta-concept *Arena* (Enceinte), peut être lié à un concept < *Service* : *chat* > instancié à partir du méta-concept *Service* à travers les liens *serviceArenaLink*. Le concept < *Service* : *chat* > portera alors une URL [1] permettant l'accès à un service chat dans LDI.

V.4 Méta-modèle lié à la phase d'exécution

Le méta-modèle lié à la phase d'exécution permet de représenter les informations liées à la phase d'exécution. Il permet également de lier les informations du niveau exécution aux niveaux supérieurs (opérationnalisation et conception). Ces liens permettent par exemple d'associer les informations de progression d'un utilisateur par rapport aux rôles qui lui sont affectés dans le scénario. La figure V.3 montre le méta-modèle lié à la phase d'exécution d'un scénario LDL.

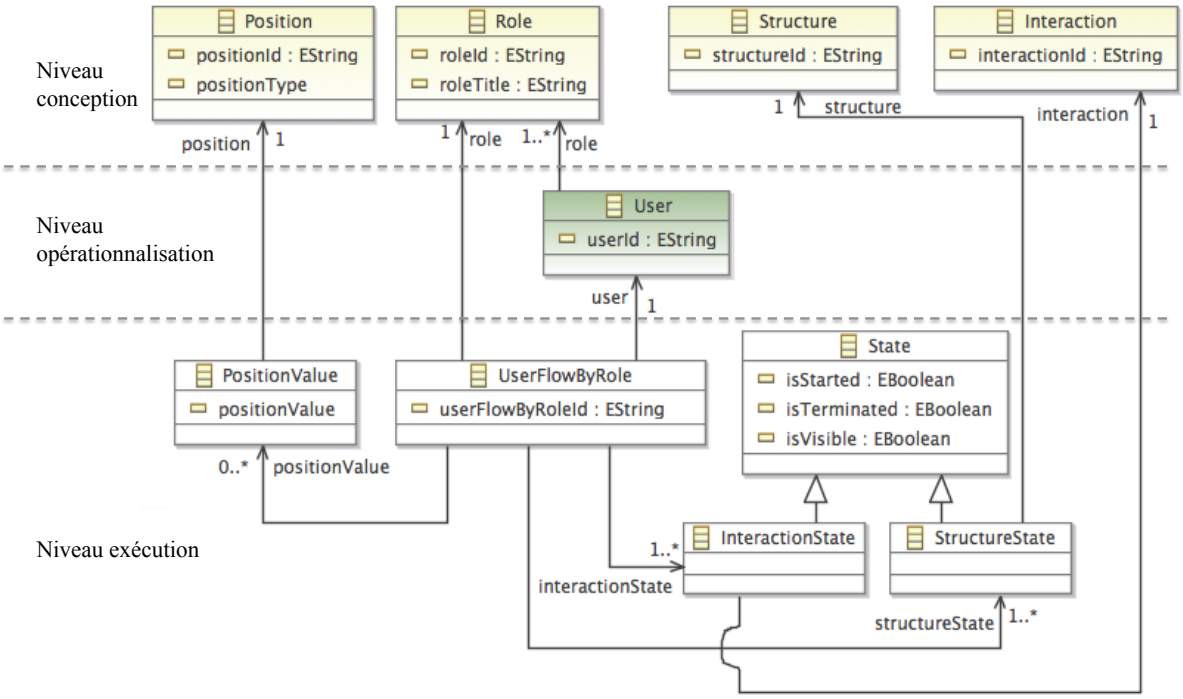

Figure V.3 : Méta-modèle lié à la phase d'exécution d'un scénario LDL dans LDI

1. Uniform Resource Locator : chaine de caractères permettant de référencer des ressources sur Internet

Le méta-modèle lié à la phase d'exécution définit les méta-concepts suivants :

- **PositionValue :** représente la valeur d'une position déclarée ou observée pour un participant. Exemple : valeur d'un choix effectué par un participant qui répond à un QCM ;
- **InteractionState :** hérite de la méta-classe *State*, elle représente l'état d'une interaction pour un participant. Elle porte par exemple la propriété *isVisible* qui permet de savoir si cette interaction est visible ou pas pour le participant ;
- **StructureState :** de la même manière que *InsteractionState*, le méta-concept *StructureState* hérite de la méta-classe *State* et représente l'état d'une Structure. Il porte aussi la propriété *isVisible* qui permet de déterminer la visibilité de cette structure pour le participant ;
- **UserFlowByRole :** ce méta-concept permet de lier les autres méta-concept aux niveaux opérationnalisation et conception : il permet de lier les informations définies par les méta-concepts *Position-State*, *InteractionState* et *StructureState* à un utilisateur jouant un rôle donné. C'est ce méta-concept qui permet de déterminer la progression de chaque participant, notamment à travers les liens *userRoleLink* définis dans le méta-modèle lié à la phase d'opérationnalisation (cf. Section V.3).

Les deux méta-modèles des niveaux opérationnalisation et exécution permettent de définir les informations liées à l'exécution d'un scénario LDL dans l'infrastructure LDI. L'approche proposée peut être généralisée et utilisée avec un autre langage de modélisation et une autre infrastructure. Elle propose un cadre conceptuel permettant d'exploiter les concepts de l'IDM pour construire des outils pour l'observation et l'adaptation par propagation. Dans notre cas, il s'agit de construire des outils capables de créer et de manipuler des modèles à partir des méta-modèles introduits, afin de disposer de *leviers* permettant d'influencer et d'adapter l'exécution du scénario.

Par ailleurs, nous considérons que la modification des modèles obtenus devrait être guidée par l'outil qui assiste le superviseur pendant l'adaptation par propagation. Afin de permettre ce guidage nous avons proposé un troisième méta-modèle pour spécifier des modifications pouvant être apportées aux modèles instanciés à partir des méta-modèles introduits. Ce méta-modèle est présenté dans la section suivante.

V.5 Un méta-modèle pour spécifier des modifications

Le méta-modèle des modifications permet de guider la définition des modifications à apporter aux modèles instanciés à partir des trois méta-modèles présentés précédemment : le modèle instancié à partir de ce méta-modèle spécifiera les modifications à apporter pour adapter les modèles liés à la phase de conception (scénario), d'opérationnalisation et d'exécution. La figure V.4 positionne ce méta-modèle dans le cycle de vie du scénario et décrit un processus d'instanciation pour la définition des modifications et leurs propagations entre les niveaux et leurs modèles.

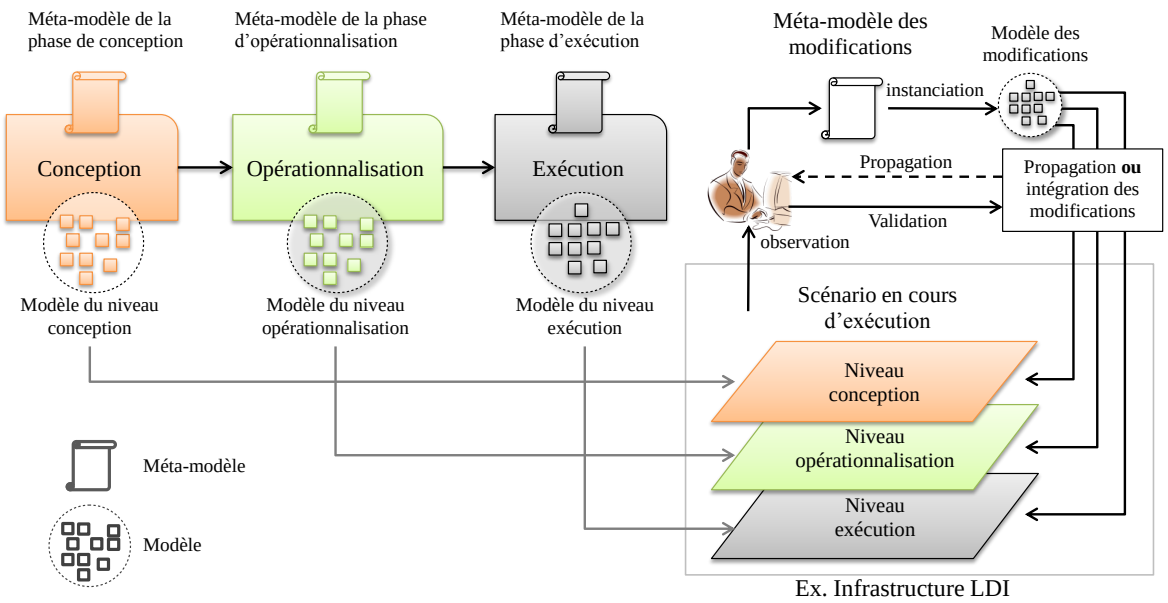

Figure V.4 : Processus de spécification des modifications pour l'adaptation par propagation

Ce processus est composé des étapes suivantes :

1. observation et détection de la situation imprévue qui nécessite une adaptation. A ce niveau, le superviseur doit analyser la situation et déterminer :
   (a) sa temporalité. Si elle est *en cours*, il devra alors suspendre l'activité en attendant l'adaptation,
   (b) le niveau de l'adaptation et par conséquent le modèle à cibler par les modifications,
2. déterminer les modifications qui devraient être apportées au modèle du niveau ciblé précédemment ;
3. instancier le méta-modèle des modifications pour spécifier les modifications identifiées précédemment ;

4. s'il y a lieu de faire des propagations sur les autres niveaux, instancier à nouveau le méta-modèle des modifications afin de compléter le modèle des modifications spécifié précédemment ;
5. s'il n'y a plus de propagations alors intégrer les modifications aux modèles qu'elles ciblent après leurs validation par le superviseur. Ce dernier doit pour cela s'assurer (le cas échéant à l'aide d'un informaticien, d'un ingénieur pédagogique) que toutes les modifications dégagées lors de l'analyse de la situation imprévue, ont été spécifiées dans le modèle des modifications, et que leur application permettra d'adapter la situation imprévue ;
6. reprendre l'exécution du scénario avec les nouvelles modifications dues à l'adaptation par propagation.

V.5.1 Avantages d'un modèle de spécification des modifications

Le fait de spécifier les changements avant de les appliquer, permet d'avoir une trace lisible qu'il est possible de vérifier et de valider avant son utilisation. La validation devrait permettre d'éviter des erreurs dans les modèles après l'adaptation. Par ailleurs, le fait de garder une trace des actions de modifications, permettrait de maintenir un historique qui pourrait être exploité pour annuler les changements effectués en cas d'erreur. Ceci suppose de mettre en place des mécanismes pour la sauvegarde des modèles de modification et des modèles de chaque niveau avant toute adaptation et leur restauration en cas d'échec.

Une autre utilité de l'utilisation d'un méta-modèle des modifications est de permettre l'échange des modifications sous forme de modèles pouvant être consultés ou réutilisés par d'autres superviseurs (enseignants, tuteur, concepteurs de scénarios). Les modèles de modification représentent donc des méta-informations liées à l'activité de supervision. Cette traçabilité s'appuie sur les propriétés définies dans le méta-concept *Adaptation* et le méta-concept *Exception* (cf. Figure V.5). Ces propriétés sont les suivantes :

- ***actorId*** **:** identifiant du superviseur qui a décrit les modifications ;
- ***level*** **:** le niveau de l'adaptation et donc le modèle auquel s'appliqueront les modifications ;
- ***isOpportunity*** **:** quand il est vrai, il permet de savoir que les modifications ne répondent pas à une situation imprévue qui pourrait bloquer l'exécution du scénario, mais à un besoin d'amélioration ;
- ***description*** **:** une description textuelle pour documenter les modifications et décrire notamment la situation imprévue ;

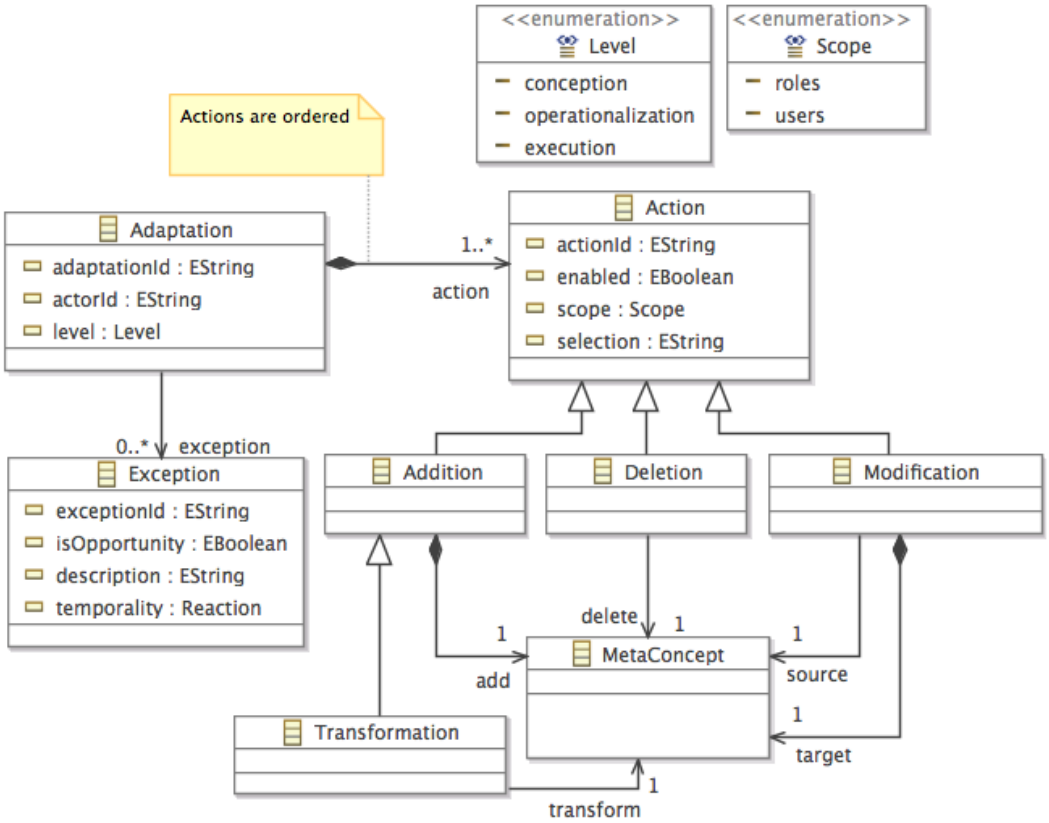

Figure V.5 : Métal-modèle des modifications pour l'adaptation par propagation

– ***temporality*** **:** temporalité de la situation imprévue. Quand elle est *en cours*, l'exécution du scénario doit obligatoirement être suspendue en attendant l'intégration de toutes les modifications.

V.5.2 Les actions de base du méta-modèle des modifications

Comme le montre la figure V.5, la description des modifications s'appuie sur les actions de base suivantes :

– ***addition*** **:** qui permet d'ajouter une instance au modèle ;
– ***deletion*** **:** qui permet de supprimer une instance dans le modèle ;
– ***modification*** **:** qui permet de modifier une instance dans le modèle ;
– ***transformation*** **:** qui permet de créer une copie à partir d'une instance dans le modèle et de la modifier. Cette action n'est pas indispensable, car elle peut être remplacée par un enchaînement des actions *addition* et *modification* dans l'ordre.

Les actions présentées ci-dessus nécessitent la sélection au préalable du méta-modèle correspondant au niveau à modifier. Cette sélection sera exprimée dans le modèle instancié (modèle des modifications), à travers la propriété *level* du méta-concept *Adaptation*.

Nous donnerons plus de détails sur les autres concepts et leurs propriétés dans l'exemple qui suit.

V.5.3 Un exemple d'utilisation du méta-modèle des modifications

Nous allons présenter un exemple d'utilisation du méta-modèle des modifications à travers son instanciation pour l'adaptation par propagation de la situation imprévue *pas de rôle expert*, proposée dans l'étude de cas du Workshop AAMCS (cf. Chapitre II). Cette situation nécessite une adaptation par propagation complète qui requiert donc d'instancier le méta-modèle des modifications pour chacun des niveaux conception, opérationnalisation et exécution.

La spécification des modifications débute par l'instanciation des méta-concept *Adaptation* et *Exception*. Pour ce qui est du méta-concept *Adaptation* il faut préciser les propriétés suivantes :

- ***actorId* :** l'identifiant du superviseur ;
- ***level* :** le niveau conception, car la situation *par de rôle expert* nécessite la modification du scénario ;

En ce qui concerne le méta-concept *Exception*, il faut préciser les propriétés suivantes :

- ***isOportunity* :** qui doit être à *vrai*, car la situation imprévue est une amélioration ;
- ***temporality* :** dans la description de la situation imprévue *par de rôle expert*, les apprenants doivent discuter immédiatement avec l'expert, car ce dernier n'est pas disponible pour longtemps. La temporalité doit par conséquent prendre la valeur *en cours*. Les modifications auront donc un impact immédiat sur l'exécution du scénario.

La prochaine étape consiste en l'instanciation des méta-concepts de type *Action* pour spécifier les modifications qui doivent être apportées au scénario et les propagations sur les autres niveaux (opérationnalisation et exécution).

Pour spécifer les modifications du niveau conception : l'ordre des actions est important et doit être vérifié et maintenu tout au long de la spécification du modèle des modifications

1. instancier le méta-concept de l'action *Add* pour ajouter un concept $< Role : expert >$;

2. instancier le méta-concept de l'action *Add* pour ajouter un concept $< Arena : questions >$. Il s'agit du type d'enceinte qui va référencer la page web sur laquelle seront visibles les questions posées aux apprenants dans la deuxième étape du scénario ;
3. instancier le méta-concept de l'action *Add* pour ajouter un concept $< Interaction : seeQuestions >$ et lui affecter le concept $< Role : expert >$ en choisissant le concept $< Arena : questions >$ comme étant l'enceinte où se déroulera l'interaction.

Pour spécifier les modifications du niveau opérationnalisation : à ce niveau il faut instancier les liens qui vont permettre d'associer les concepts ajoutés précédemment aux concepts adéquats au niveau opérationnalisation. Voici les modifications à spécifier :

- instancier le méta-concept de l'action *Add* pour ajouter un concept $< Resource : pageQuestions >$ avec l'URL de la page web qui présente les questions posées aux apprenants,
- instancier le méta-concept de l'action *Add* pour ajouter un concept $< User : nicolas >$ (nicolas étant l'identifiant de l'expert en environnement) ;
- instancier le méta-concept de l'action *Add* pour ajouter un concept $< UserRoleLink : nicolasExpertLink >$ qui est un lien vers le rôle *expert* ajouté au niveau conception. Nous affectons ainsi le rôle d'expert à l'utilisateur *nicolas* ;
- instancier le méta-concept de l'action *Add* pour ajouter un concept $< ResourceArenaLink : pageQuestionsQuestionsLink >$ qui est un lien vers l'enceinte *pageQuestions* ajoutée au niveau conception ;
- instancier le méta-concept *Modification* pour modifier l'interaction *discussInTheChat* qui permet d'utiliser le chat de discussion dans la troisième étape du scénario. Cette modification consiste en l'ajout du rôle *expert* à la liste des rôles autorisés à faire l'interaction (propriété *adressers* d'une interaction LDL).

Pour spécifier les modifications du niveau exécution : afin d'effectuer ces modifications, il va falloir modifier les valeurs de visibilité des interactions des apprenants. L'objectif étant de les autoriser à faire l'interaction qui leur permettra d'aller directement discuter avec l'expert. Tous les apprenants sont donc concernés. Il faut par conséquent définir la propriété *scope* de chaque action à *roles* et la propriété *selection* au rôle *learner*. Cela permettra de limiter la portée des actions de modification aux interactions des apprenants. Voici la seule action à spécifier : instancier le méta-concept *Modification* et préciser le concept $< IntercationState : discussInTheChat >$ comme cible dans le modèle du scénario (niveau conception). Il faut ensuite renseigner la pro-

priété *target* ciblant le même concept $< InteractionState : discussInTheChat >$, mais avec la propriété *isVisible* à vrai pour permettre aux apprenants d'y accéder.

Comme nous l'avons précisé au début de ce chapitre, nous avons implémenté un *Framework* orienté IDM pour créer et manipuler les modèles des différents niveaux introduits ci-dessus et les modèles de modifications pour leurs adaptations. Les détails de mise en œuvre de ce *Framework* sont présentés dans la section suivante.

V.6 Mise en œuvre d'un *Framework* pour l'adaptation par propagation

Le *Framework* implémente les méta-modèles proposés et permet de les instancier en prenant en compte l'organisation en niveaux : conception, opérationnalisation et exécution. Il permet notamment l'observation de l'évolution de ces modèles en temps réel et leur modification à travers l'instanciation du méta-modèle des modifications.

V.6.1 Architecture du *Framework*

Le *Framework* a été développé pour être utilisé avec LDL et LDI, mais l'approche adoptée peut être étendue à d'autres langages de modélisation et d'autres infrastructures. Pour cela, il faut soit :

- implémenter les méta-modèles adaptés au langage de modélisation et à l'infrastructure cible ;
- s'appuyer sur des techniques de transformation de modèles pour permettre l'intéropérabilité des langages et des infrastructures.

La première solution nécessite d'implémenter plusieurs méta-modèles, ce qui peut être lourd et complexe vu le nombre de langages et d'infrastructures qui existent. La deuxième solution semble donc plus préférable sous réserve d'existence de *mapping*[2] possibles.

Le recours à l'IDM nous a permis d'implémenter le *Framework* de façon découplée de l'infrastructure LDI. La communication entre le *Framework* et LDI s'établit à travers des appels de

2. Correspondances entre des éléments de modèles conformes à des méta-modèles différents

services[3] utilisant des requêtes XML RPC[4]. Comme le montre la figure V.6, le *Framework* est composé de différentes couches et différents modules.

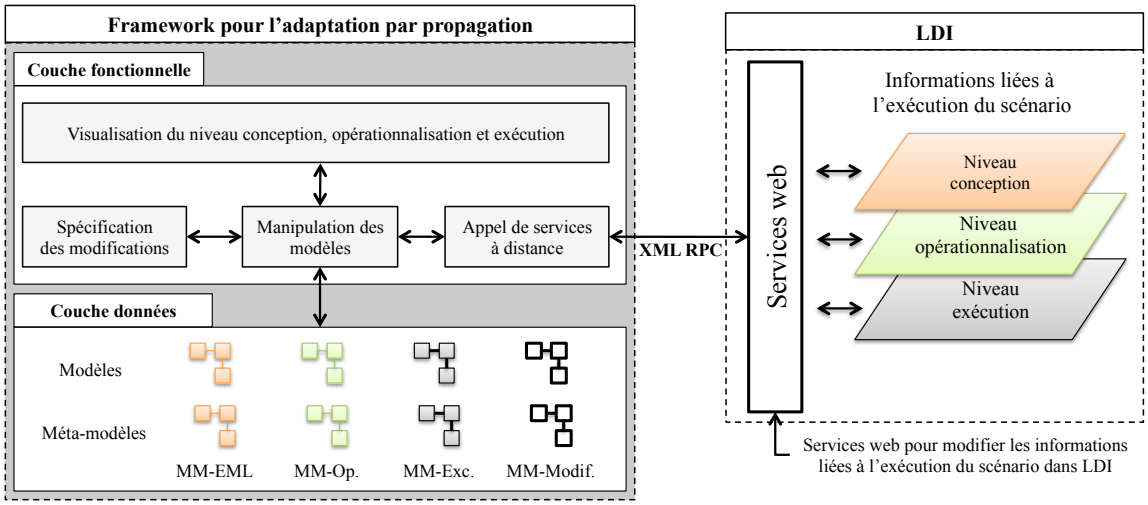

Figure V.6 : Architecture du *Framework* d'adaptation par propagation

La *couche données* englobe les différents méta-modèles et modèles introduits ci-dessus. La *couche fonctionnelle* regroupe les différents modules qui implémentent les fonctionnalités du *Framework* :

- **le module de manipulation des modèles :** permet l'instanciation des méta-modèles pour la transformation des informations liées à l'exécution du scénario en des modèles manipulables dans le *Framework*, et notamment l'instanciation du méta-modèle des modifications pour la spécification des modifications. Ce module permet également de lire et de modifier tout les modèles de la couche des données ;
- **le module de spécification des modifications :** permet de spécifier les actions de modification qui doivent être instanciées par le module de manipulation des modèles ;
- **le module de visualisation :** permet d'observer l'évolution des modèles dans le temps. La récupération des informations liées à l'exécution du scénario et leur transformation en modèles pour l'observation se fait alors à la volée et à chaque demande du superviseur. Ce module est composé de trois sous-modules pour l'observation des trois modèles liés

3. Programme informatique permettant la communication et l'échange de données entre applications et systèmes hétérogènes dans des environnements distribués.

4. Le protocole RPC (*Remote Procedure Call*) est une spécification simple et un ensemble de codes qui permettent à des processus s'exécutant dans des environnements différents de faire des appels de méthodes à travers un réseau.

aux niveaux conception, opérationnalisation et exécution ;

- **le module d'appel des services :** assure la communication via des services pour l'échange d'informations entre le *Framework* et LDI. Cet échange se fait dans deux sens :
  - **de LDI vers le *Framework* :** pour la récupération à distance des informations liées à l'exécution du scénario au sein de l'infrastructure LDI. Les services utilisés permettent de récupérer les informations liées à l'exécution du scénario dans LDI. Elles permettent notamment l'observation de l'évolution de l'activité en invoquant ces services de manière régulière et continue. Une fois ces informations récupérées, le *Framework* les traite et les transforme (au sens transformation IDM cf. Section V.2) en modèles conformes aux trois méta-modèles introduits. Ils peuvent alors servir à l'observation et à la spécification des modifications pour l'adaptation par propagation,
  - **du *Framework* vers LDI :** pour la mise à jour des informations liées à l'exécution du scénario dans LDI. Le module d'appel des services interprète les modèles de modification, détermine les services à appeler et construit une liste des requêtes qui permettront la mise à jour des informations sur LDI.

Remarque : dans la version actuelle du *Framework*, le module de spécification n'est pas complètement fonctionnel. Il permet uniquement d'importer et de lire les modifications déjà spécifiées dans un fichier XML externe. Ce dernier représente alors un modèle conforme au méta-modèle des modifications. Nous présentons un exemple de modèle de modification dans la section suivante.

V.6.2 Processus d'adaptation par propagation dans le *Framework*

Comme nous l'avons précisé dans la section précédente, le *Framework* est composé de plusieurs modules. L'adaptation par propagation dans le *Framework* s'appuie sur une utilisation des différents modules dans un ordre bien précis. Cet ordre est décrit par le diagramme d'activité proposé dans la figure V.7. En utilisant le *Framework*, l'adaptation par propagation se fait à travers les étapes suivantes :

1. la récupération des informations liées à l'exécution du scénario LDL sélectionné par le superviseur ;
2. la création des modèles liés aux trois niveaux : conception, opérationnalisation et exécution à travers l'instanciation des méta-modèles appropriés ;
3. la génération des trois vues qui vont permettre l'observation des modèles créés ;

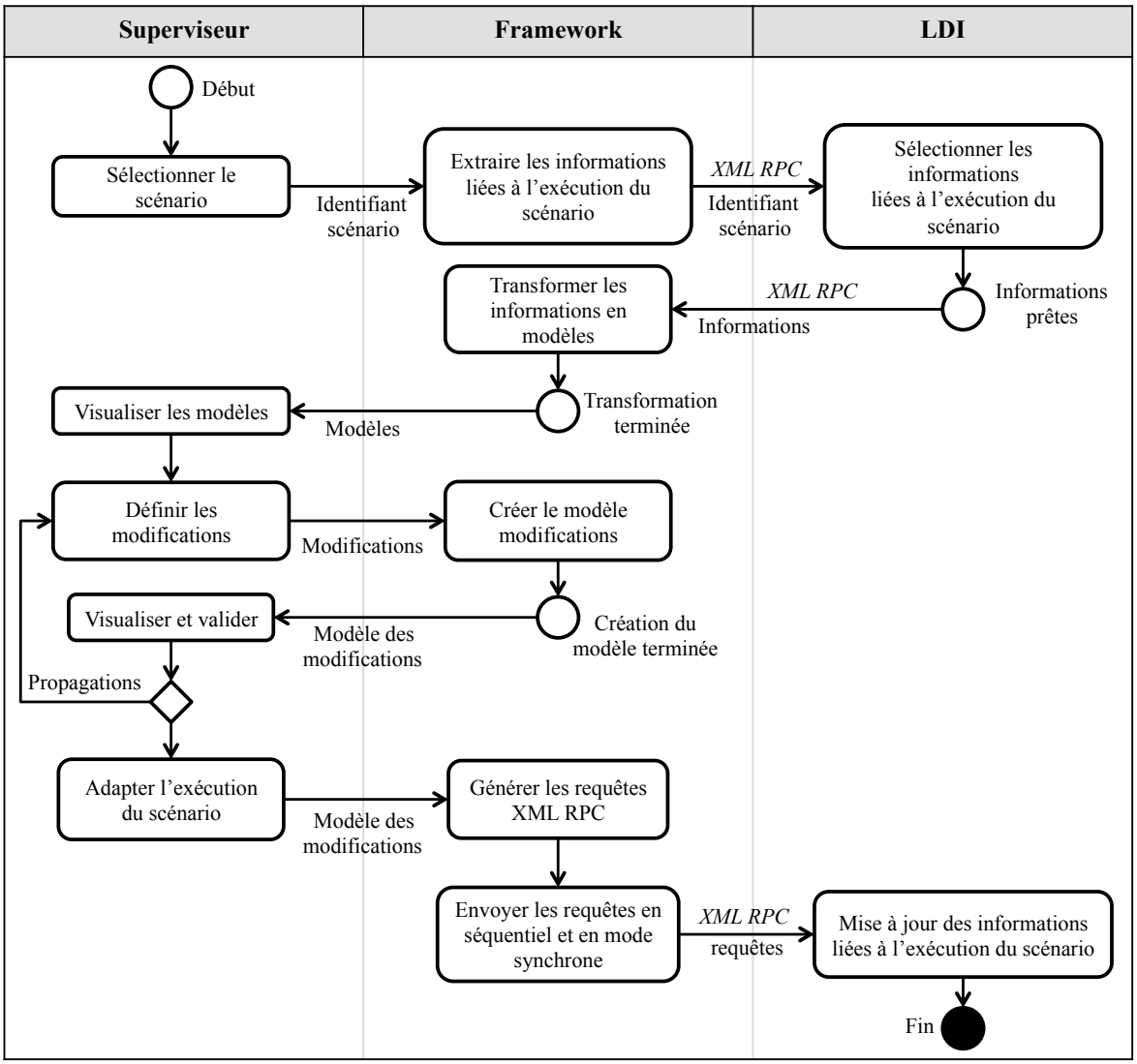

Figure V.7 : Diagramme d'activité d'une adaptation par propagation dans le *Framework*

4. l'instanciation du méta-modèle des modifications pour la spécification d'un modèle qui représente les modifications qui doivent être apportées aux modèles créés ;

5. la validation des modifications par le superviseur qui doit vérifier notamment l'existence de propagations. Toutes les propagations nécessaires à l'adaptation doivent être prises en considération et spécifiées dans l'ordre au sein du modèle des modifications ;

6. la transformation automatique des modifications spécifiées dans le modèle des modifications pour la génération des requêtes XML RPC qui serviront à mettre à jour les informations liées à l'exécution du scénario. Ces requêtes doivent alors être envoyées de manière à respecter l'ordre des modifications dans le modèle des modifications et de façon synchrone.

V.6.3 Déroulement d'un exemple d'adaptation par propagation dans le *Framework*

Dans l'approche proposée dans ce chapitre, les modifications associées aux trois niveaux conception, opérationnalisation et exécution sont spécifiées dans un modèle instancié à partir du méta-modèle des modifications. Nous avons déjà spécifié les modifications relatives à la situation imprévue *par de rôle expert* dans la section V.5.3. Nous allons reprendre les mêmes modifications pour effectuer une adaptation par propagation en utilisant notre *Framework*. Nous suivrons alors les étapes décrites par le diagramme d'activité de la figure V.8.

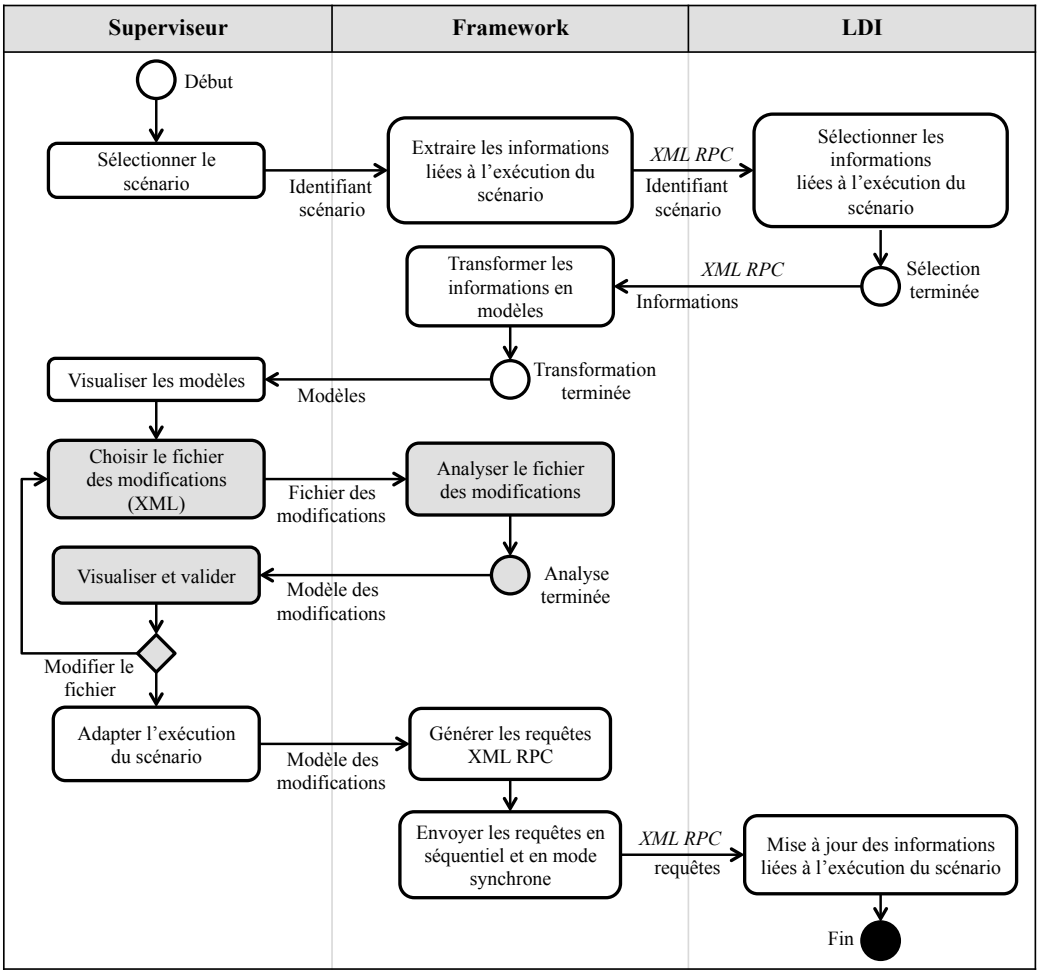

Figure V.8 : Importation et analyse d'un modèle de modification au format XML

Ce diagramme diffère de celui présenté précédemment (cf. Figure V.7), car les modifications ne sont pas spécifiées directement dans le *Framework*. Elles sont au contraire importées dans un fichier XML. Les étapes représentées en boites grisées dans le diagramme d'activité font référence

aux étapes permettant d'importer, d'analyser et transformer ce fichier XML en modèle de modifications. Toutes les modifications ainsi que les propagations doivent être spécifiées dans ce fichier.

Exemple de modification du niveau conception : la figure V.9 montre un fragment de code XML qui correspond à l'action *Add* du rôle *expert*. Cette action définit l'identifiant du rôle et son titre conformément au méta-modèle du niveau conception.

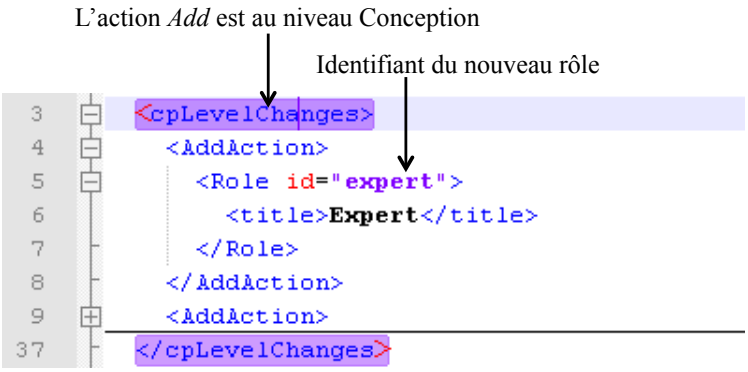

Figure V.9 : Fragment XML de l'action *Add* pour l'ajout du rôle *Expert*

Exemple de modification du niveau opérationnalisation : la figure V.10 montre un autre fragment de code XML associé à action *Add* du lien *nicolasExpertLink* qui modifiera le niveau opérationnalisation afin d'affecter le rôle *expert* à l'utilisateur *nicolas*.

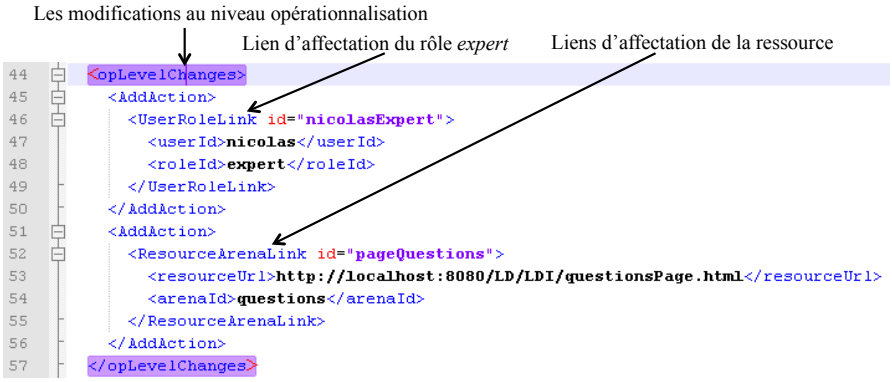

Figure V.10 : Fragment XML pour l'ajout des liens d'opérationnalisation

Exemple de modification du niveau exécution : enfin, la figure V.11 montre un fragment de code XML qui définit l'action *Modification* qui permettra de modifier la visibilité de l'interaction *toDebateOnTopic*. Cette action doit alors spécifier l'identifiant de l'interaction à modifier et sa

nouvelle définition. Les paramètres *scope* et *selection* limitent la modification aux utilisateurs ayant le rôle *learner*. Le paramètre *isVisible* modifie la visibilité de l'interaction à *True*. Cette dernière devient donc visible pour les apprenants.

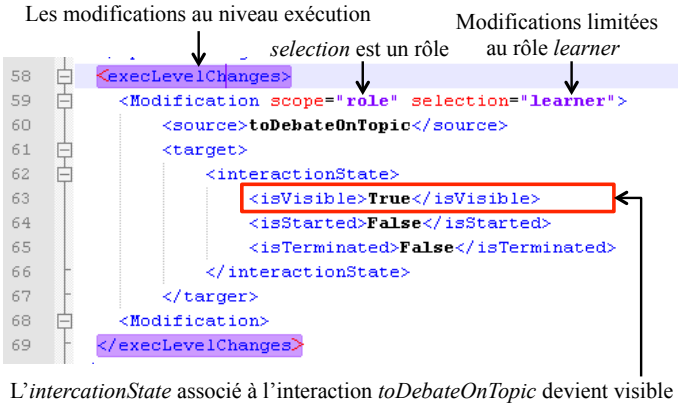

Figure V.11 : Fragment XML de l'action *modification* au niveau exécution

Sélection du scénario :

Au lancement, le *Framework* se connecte à l'infrastructure LDI et récupère les identifiants de tous les scénarios qui y sont enregistrés.

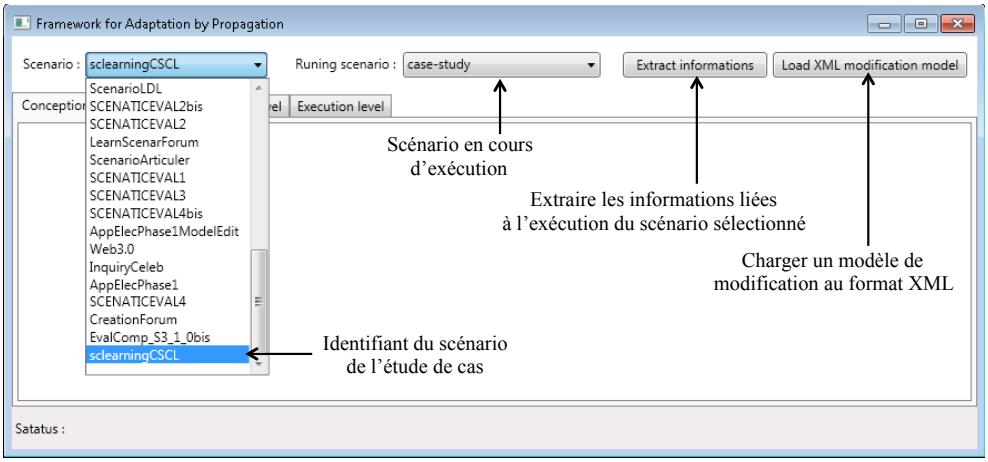

Figure V.12 : Sélection de l'identifiant du scénario dans le *Framework*

La Figure V.12 montre cette liste. Le scénario qui nous intéresse est référencé par l'identifiant *cscl-caseStudy*. Une autre liste permet de sélectionner le scénario en cours d'exécution avant de lancer l'extraction des informations liées à son exécution.

Extraction des informations liées à l'exécution du scénario :

Une fois le scénario en cours d'exécution sélectionné dans l'interface du *Framework*, la prochaine étape consiste en l'extraction des informations liées à son exécution. Pour extraire ces informations, le *Framework* envoie une requête XML RPC à LDI. LDI renvoie ces informations dans un fichier XML qui est analysé ensuite par le *Framework* pour en extraire les informations de chaque niveau et les transformer en modèles. La figure V.13 montre une vue arborescente du modèle lié au niveau conception.

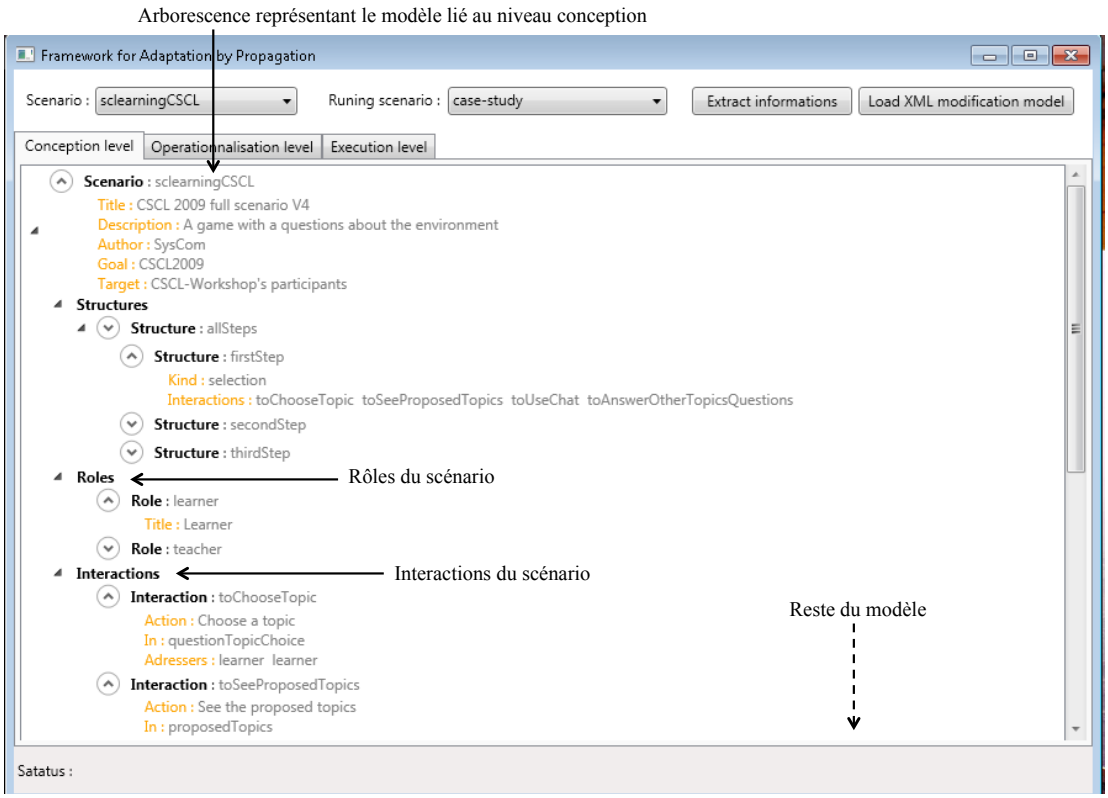

Figure V.13 : Vue arborescente du modèle lié au niveau conception

La figure V.14 montre la vue associée au niveau opérationnalisation. Il est par exemple possible d'y voir les utilisateurs et leurs rôles ainsi que les ressources et les services associés à l'exécution du scénario.

La figure V.15 montre la vue associée au niveau exécution. Il est possible de constater sur cette vue les interactions des participants par rôle. Il est notamment possible de savoir quelles interactions sont visibles. Sur l'exemple présenté, nous pouvons voir que l'interaction *toDebateOnTopic* n'est pas visible pour *aude* et c'est aussi le cas pour tous les autres apprenants.

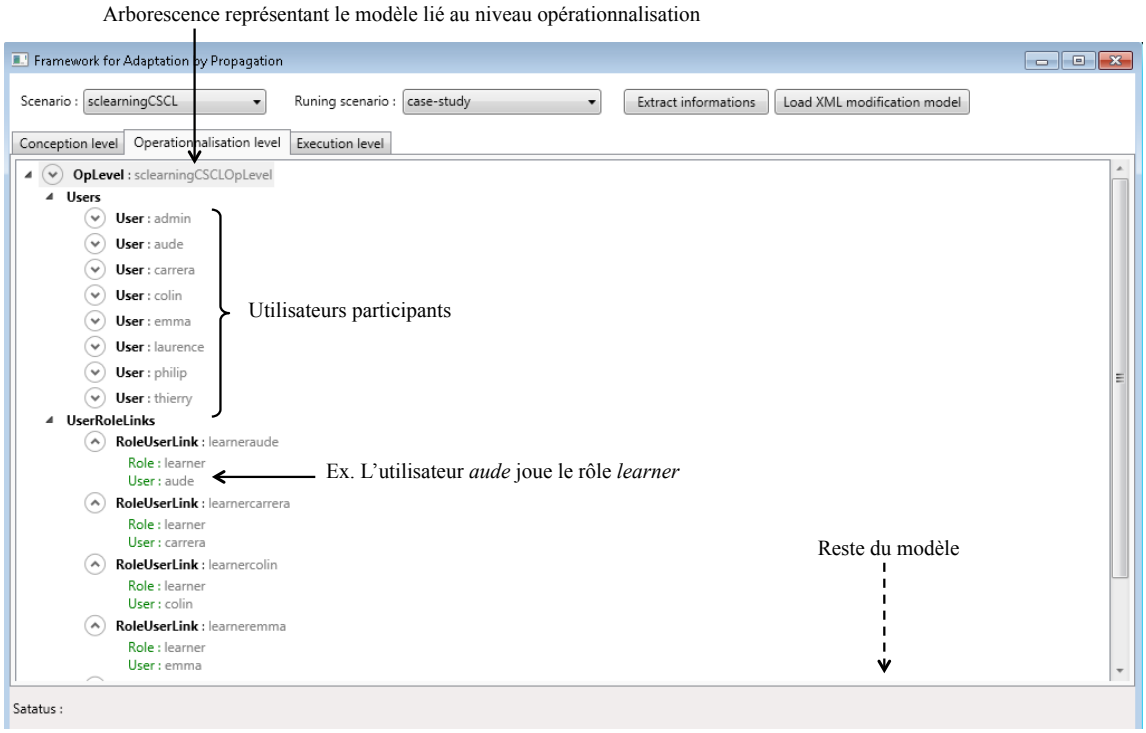

Figure V.14 : Vue arborescente du modèle lié au niveau opérationnalisation

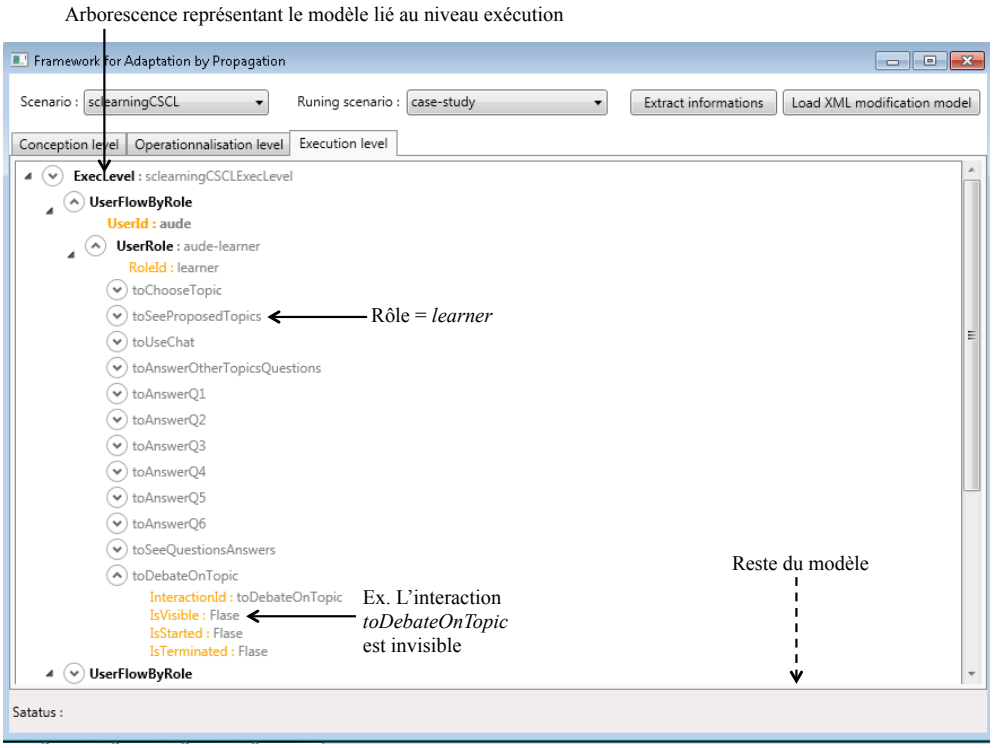

Figure V.15 : Vue arborescente du modèle lié au niveau exécution

Chargement du modèle des modifications

Le chargement du modèle des modifications dans le *Framework* va permettre d'adapter l'exécution du scénario et de répondre à la situation *pas de rôle expert*. Comme nous l'avons expliqué précédemment, le modèle des modifications est spécifié dans un fichier XML qui sera chargé et analysé dans le *Framework*. La figure V.16 montre le résultat affiché par le *Framework* une fois le fichier chargé et analysé. La fenêtre affichée à l'écran propose au superviseur de voir la liste des modifications. Une fois les modifications vérifiées et validées par le superviseur, l'adaptation peut être appliquée. Les modifications sont alors transformées en requêtes qui vont modifier les informations liées à l'exécution du scénario dans LDI.

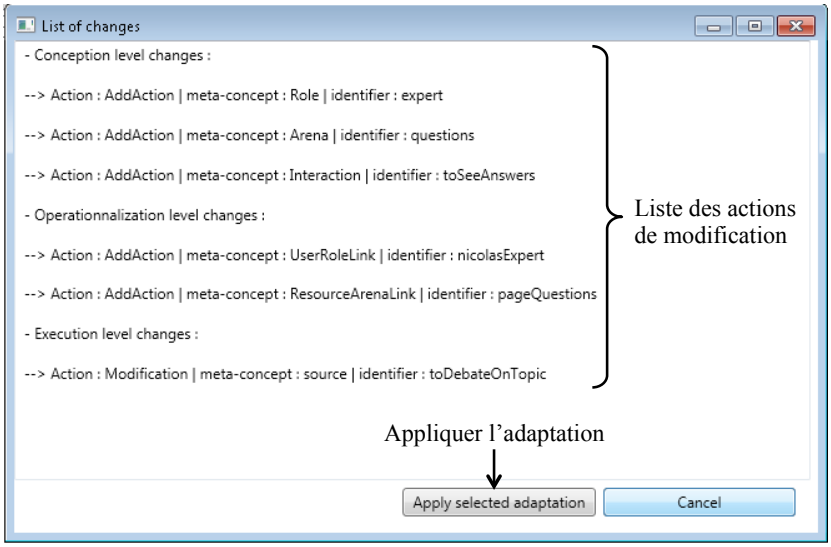

Figure V.16 : Validation des modifications avant l'adaptation par propagation

Extraction des informations liées à l'exécution du scénario après l'adaptation :

Une nouvelle extraction des informations liées à l'exécution du scénario montre que les modifications ont bien été apportées aux trois niveaux conception, opérationnalisation et exécution.

Au niveau conception, nous pouvons constater l'apparition d'un nouveau rôle *expert*. Au niveau opérationnalisation, le nouveau rôle est bien affecté à l'utilisateur *nicolas* avec l'apparition d'un *userRoleLink* qui référence le rôle *expert*. Enfin, au niveau exécution, l'interaction *toDebateOnTopic* est devenue visible comme le montre la figure V.17. Nous pouvons aussi constater ces même modifications au niveau de LDI et notamment au niveau du player de chaque apprenant qui affiche désormais la nouvelle interaction *toDebateOnTopic* (cf. Figure V.18). Les apprenants

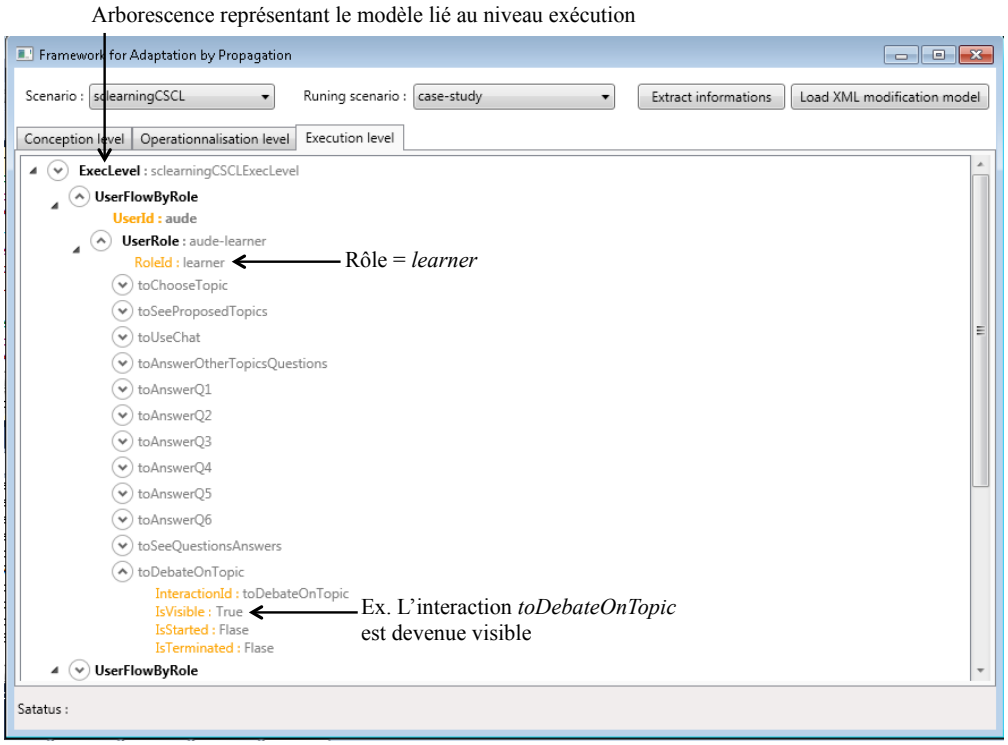

Figure V.17 : Vue sur le niveau exécution après l'adaptation

peuvent sélectionner l'interaction pour participer directement à la discussion avec l'expert.

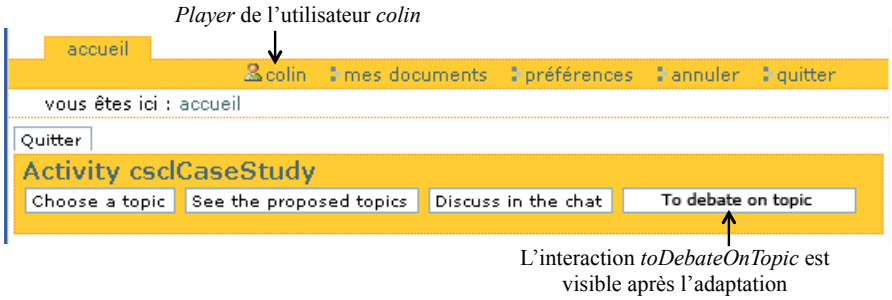

Figure V.18 : Etat du *player* après l'adaptation

Au moment où le superviseur prépare l'adaptation, les apprenants sont en train de répondre aux questions de la deuxième étape (QCM). Dans le cas de l'exemple traité ci-dessus, l'adaptation à la volée de l'exécution du scénario n'aura pas d'impact direct sur l'activité des apprenants. Cependant, afin d'éviter que les informations liées à l'exécution du scénario ne changent à nouveau, il pourrait être préférable de suspendre l'activité. Par ailleurs, la durée du processus d'adaptation n'étant pas connue à l'avance, il serait par conséquent indispensable de notifier les

apprenants pour les tenir au courant du délai qu'ils auront à observer en attendant la fin de l'adaptation.

Remarque : la fonction de suspension de l'exécution du scénario n'étant pas implémentée dans le *Framework*, le superviseur devra le faire directement via l'interface d'opérationnalisation de LDI. Comme perspective d'amélioration du *Framework*, nous envisageons l'implémentation de fonctionnalités de contrôle de l'exécution des scénarios (démarrage, suspension, arrêt) en exploitant des appels de services web déjà implémentés dans LDI.

V.7 Conclusion

Nous avons présenté dans ce chapitre un *Framework* pour l'adaptation par propagation de situations d'apprentissage se déroulant conformément à l'exécution d'un scénario pédagogique. Ce *Framework* permet de récupérer les informations liées à l'exécution des scénarios sur LDI. Il permet ainsi d'expliciter ces informations en les transformant de manière automatique et transparente en des modèles plus faciles à manipuler. L'exploitation de ces modèles dans le *Framework* structurent et guident l'activité de supervision en permettant d'une part, l'observation des trois niveaux conception, opérationnalisation et exécution en temps réel, et d'autre part la spécification des actions de modification requises pour l'adaptation par propagation et leur intégration au sein de l'infrastructure LDI. Ce *Framework* intègre ainsi les fonctions de supervision en permettant notamment la traçabilité de l'activité d'adaptation et le partage des modèles de modification pour la réutilisation. En plus de couvrir le processus de supervision, l'utilisation de l'approche proposée (adaptation par propagations) combinée à une approche IDM offre des capacités d'extensions pouvant être exploitées pour supporter différents EML.

Chapitre VI

Conclusion et perspectives de nos travaux

Dans le cadre de cette thèse, nous avons étudié la problématique d'adaptation d'activités d'apprentissage informatisées. Une approche d'adaptation par propagation basée sur l'IDM a été proposée. Nous présentons dans ce dernier chapitre, le bilan de nos travaux et les résultats obtenus, ainsi qu'une analyse de nos travaux et les améliorations que nous envisageons comme perspectives.

Bilan de nos travaux

Nos travaux se sont focalisés sur une problématique dans laquelle nous nous sommes confrontés à une difficulté qui dépasse les frontières de notre contexte (Ingénierie des EIAH). Nous retrouvons en effet le problème de l'adaptation à la volée aussi bien dans l'ingénierie des EIAH que dans le Workflow. La raison de cela est liée au cycle de vie à travers lequel sont conçus, opérationnalisés et exécutés les modèles. En effet, le point de départ est un modèle spécifié par instanciation d'un méta-modèle conceptuel. Le modèle obtenu est abstrait et indépendant de la plate-forme cible. Une fois cette transformation effectuée, le modèle est exécuté. Se pose alors le problème de modifier ce modèle une fois l'exécution entamée. Même si nous situons nos travaux dans un contexte d'Ingénierie des EIAH, les travaux et les résultats obtenus peuvent être exploités dans le Workflow ou dans tout autre domaine exploitant une approche qui utilise des modèles calculables pouvant être opérationnalisés et exécutés.

Dans le chapitre II, nous avons présenté une étude de la problématique d'adaptation à la volée et des situations imprévues. Grâce à cette étude, nous avons pu dégager certaines caractéristiques

que nous avons utilisées dans notre approche. Nous nous sommes intéressés à la nature de la situation imprévue qui définit les éléments qui doivent être modifiés par l'adaptation à la volée. Nous considérons que les informations liées à l'exécution du scénario peuvent être organisées en trois niveaux : conception, opérationnalisation et exécution. Ainsi, selon les informations qu'elle cible, l'adaptation à la volée peut être de :

- *niveau conception* si elle requiert de modifier les informations liées à la phase de conception ;
- *niveau opérationnalisation* quand elle doit modifier les informations liées à l'opérationnalisation du scénario ;
- *niveau exécution* quand elle nécessite de modifier les informations liées à l'exécution du scénario.

Nous avons également mis en évidence les liens de dépendance entre les informations portées par chacun des niveaux introduits. Partant de l'idée que les différents niveaux d'informations sont liés, nous avons établi que la modification de chaque niveau pourrait nécessiter la modification des autres niveaux. C'est ce que nous avons appelé la propagation des modifications. L'adaptation par propagation consiste donc à modifier un des niveaux (conception, opérationnalisation ou exécution) et à propager les modifications sur les autres niveaux. Une adaptation de niveau conception peut par exemple se propager sur les niveaux opérationnalisation ou sur le niveau exécution. La propagation est dite complète quand elle modifie les trois niveaux. Nous avons par ailleurs implémenté un prototype qui étend l'infrastructure LDI pour l'adaptation par propagation complète de la situation imprévue *par de rôle expert* (cf. Section **??**). Ce prototype nous a permis de valider l'approche d'adaptation par propagation qui représente notre première contribution.

Notre deuxième contribution consiste en l'utilisation de l'IDM pour l'adaptation par propagation. Nous avons proposé d'étendre l'utilisation des méta-modèles pour la représentation des informations liées à la phase d'opérationnalisation et d'exécution. Chaque niveau (conception, opérationnalisation, exécution) est par conséquent représenté par son méta-modèle. En plus de représenter les informations liées à son niveau, chaque méta-modèle spécifie des liens qui référencent des méta-concepts du niveau supérieur. Par exemple, chaque méta-modèle du niveau opérationnalisation référence des méta-concepts du niveau conception. Le méta-modèle du niveau exécution référence des méta-concepts des niveaux opérationnalisation et conception. L'instanciation de chacun de ces méta-modèles produit un modèle qui explicite les informations

liées à chaque niveau et les liens entre eux. Ces modèles servent de leviers pour manipuler et modifier les informations liées à l'exécution du scénario.

Pour permettre le guidage des manipulations effectuées sur ces modèles, nous avons introduit un méta-modèle des modifications. Celui-ci permet de spécifier les modifications sous forme d'actions simples sur les autres modèles. L'ensemble de ces actions constitue ainsi un modèle des modifications qui vont permettre l'adaptation par propagation.

Enfin, pour supporter l'approche d'adaptation par propagation, nous avons implanté un *Framework* découplé de l'infrastructure LDI. Celui-ci propose des modules qui permettent de récupérer les informations liées à l'exécution du scénario, et de les transformer en modèles. D'autres modules permettent ensuite de visualiser ces modèles pour observer par exemple les éléments qui doivent être modifiés avant l'adaptation. Par ailleurs, afin d'appliquer une adaptation par propagation, un autre module permet d'importer un fichier XML qui spécifie les modifications et de le transformer en modèle. Ce modèle peut être visualisé et validé par le superviseur afin d'appliquer les modifications. Le *Framework* étant découplé de LDI, les modifications s'effectuent par l'exécution de requêtes distantes en utilisant le protocole *XML RPC*. Nous avons validé l'approche d'adaptation par propagation et le *Framework* à travers l'exemple traité dans le chapitre V. Nous avons ainsi traité une adaptation par propagation complète.

Perspectives de nos travaux

L'approche d'adaptation par propagation permet de guider et d'organiser les modifications des informations liées à l'exécution du scénario. Cette organisation passe par une séparation des modifications de chacun des niveaux conception, opérationnalisation et exécution. La propagation permet ensuite de répercuter les modifications entre les niveaux. Le recours à l'IDM permet d'utiliser des modèles afin de construire des outils pour l'observation et notamment l'adaptation.

L'observation

Pour ce qui est de l'observation, les méta-modèles liés à la phase d'opérationnalisation et d'exécution permettent de capturer l'état des informations liées à l'exécution du scénario en temps réel. Nous pourrions apporter encore des améliorations à ce niveau :

- du point de vue de l'approche : étendre le méta-modèle lié à la phase d'exécution afin de capturer plus d'informations (Ex. Résultats (positions) des participants, leurs choix, leurs

objectifs). Aussi, de pouvoir désactiver un lien d'opérationnalisation momentanément. Ceci pourrait être utilisé pour isoler une ressource, un service ou un utilisateur avant d'effectuer des modifications par exemple ;

- du point de vue du *Framework* : l'observation s'appuie sur des vues arborescentes construites sur les modèles des trois niveaux conception, opérationnalisation et exécution. Il est possible de visualiser chaque modèle et donc chaque niveau de manière isolée. Nous pourrions exploiter davantage ces modèles pour construire d'autres modules permettant par exemple :
  - d'observer les informations liées à l'exécution du scénario de façon synchrone et continue, car la version actuelle du *Framework* le fait à chaque demande du superviseur,
  - de désactiver/activer un lien d'opérationnalisation avant une adaptation pour bloquer l'accès à une ressource (un service) qui doit être remplacée, ou alors, pour isoler un utilisateur et l'empêcher de participer à l'activité,
  - de construire des vues plus intuitives pour visualiser en temps réel les progressions des participants,
  - de détecter un écart important de progression et de le signaler sur la vue associée au niveau exécution. Ceci permettrait par exemple de détecter la situation de l'apprenant en retard (cf. Section II.3.2),
  - de pouvoir exploiter les liens qui existent entre les trois niveaux pour proposer au superviseur de visualiser les autres concepts liés à un concept sélectionné. Par exemple de sélectionner un utilisateur au niveau opérationnalisation et de voir tous les concepts liés à celui-ci au sein d'un même niveau et dans les autres niveaux : les interactions, les rôles et les règles au niveau conception ou les *InteractionState* au niveau exécution. Ceci pourrait aider le superviseur à constater l'impact d'une modification sur le concept sélectionné et par conséquent les propagations qui seront nécessaires.

L'adaptation

Pour permettre la spécification des modifications d'une adaptation par propagation, nous avons proposé un méta-modèle de modifications. L'instanciation de ce méta-modèle permet de créer un modèle qui spécifie les modifications qui doivent être apportées à chaque niveau. Dans le cas du *Framework* que nous avons proposé, ce modèle est créé à partir d'un fichier XML. Afin de guider le superviseur dans la spécification des modifications, le *Framework* pourrait être augmenté d'un module permettant d'instancier et de créer les modifications requises et de les

exporter ensuite pour garder une trace (XML), pour une réutilisation ultérieure ou tout simplement pour documenter l'activité d'adaptation du superviseur. Nous envisageons donc un travail dans cette optique là.

Une autre perspective concerne la validation sémantique des modifications. Le *Framework* pourrait aider le superviseur dans la sélection des modifications. Dans la version actuelle, les propagations ne sont pas calculées automatiquement, mais spécifiées manuellement. Cette aide pourrait se manifester sous forme de suggestions des changements possibles en fonction de chaque modification sélectionnée par le superviseur. Ceci permettrait un gain de temps important pour l'adaptation d'autant plus que l'adaptation à la volée se fait en temps réel et dans un temps relativement court. De plus, le fait d'assister le superviseur permettrait d'éviter des erreurs lors de l'adaptation, car celles-ci pourraient provoquer à leur tour, des situations qui nécessiteraient d'être adaptées.

Pour guider et assister d'avantage l'adaptation par propagation, il serait intéressant d'avoir la possibilité de passer par des *patterns* d'adaptation (ou modèle d'adaptation), qui une fois instanciés et initialisés, permettraient de générer les modifications nécessaires à l'adaptation d'une situation imprévue identifiée et connue. [KD09] propose une approche dans laquelle il se base sur des patterns d'adaptation. Ces patterns représentent un référentiel de situations d'apprentissage particulières et les moyens requis pour leur adaptation. Il suppose alors que ces situations pourraient survenir en se basant sur des observations et des expérimentations faites sur activités réelles. Les patterns sont donc utilisés pour permettre au système d'initier des processus d'adaptation bien définis lorsque des conditions spécifiques sont identifiées. Notre utilisation de ces patterns sera différente. Chaque *pattern* sera réutilisable dans des *situations imprévues* identifiées et connues (exemple situation de l'*apprenant en retard* cf. Section II). La différence réside dans le fait que, dans notre cas, le pattern est sélectionné et utilisé par le superviseur pour gérer une situation imprévue bien spécifique. Ceci passe par un travail de caractérisation des situations imprévues pour en déduire les patterns qui permettraient de les adapter. Ces patterns pourraient être implémentés sous forme de modèles permettant de représenter les informations requises (que doit fournir le superviseur à leur instanciation) pour adapter par propagation une situation imprévue ciblée. La figure VI.1 présente un exemple simple de pattern d'adaptation qui pourrait être utilisé pour créer un nouveau rôle LDL en réutilisant des interactions existantes. Ce pattern permettrait de générer un modèle de modifications pour l'ajout de ce rôle.

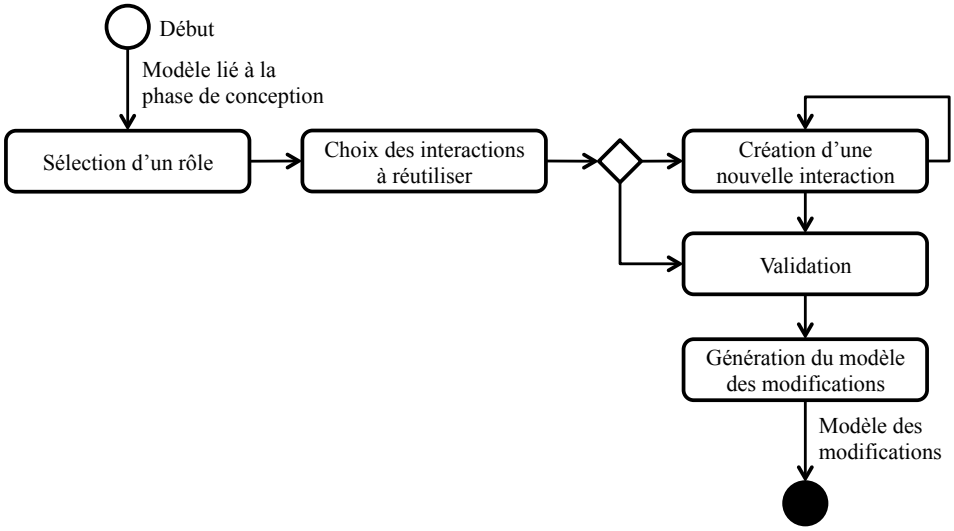

Figure VI.1 : Pattern d'adaptation pour l'ajout d'un rôle LDL par réutilisation

Table des figures

I.1 Méta-modèle de la spécification IMS-LD. Source [Lej04] 17
I.2 Modèle relationnel d'IMS LD. Source [Lej04] 18
I.3 Cycle de vie d'un scénario pédagogique 20
I.4 Cycle de vie d'un scénario *LDL* 21
I.5 Outils utilisés dans le cycle de vie d'un scénario pédagogique 22
I.6 Diagramme des cas d'utilisation d'un éditeur LD. Source [KT05] 23
I.7 Positionnement de *Model Editor* et de *Relaod LDE* dans le cycle de vie du scénario 24
I.8 Méta-modèle de l'EML LDL 26
I.9 Architecture de l'infrastructure *LDI* 32
I.10 Intégration du moteur d'exécution dans *LDI* 32
I.11 Observation de l'état de l'activité dans LDI 33
I.12 Configuration des rôles dans LDI 34
I.13 Configuration des enceintes dans LDI 35

II.1 La structure basique d'une activité. Source [Bou00] 39
II.2 Les niveaux hiérarchiques d'une activité. Source [Bou00] 39
II.3 Une classification des types basiques de support au travail. (Source [Bou00]) . . 40
II.4 Pattern de déroulement des phases du scénario LearnElec 42
II.5 Parcours des apprenants adaptés aux réponses données 44
II.6 Positionnement de l'adaptation à la volée dans le cycle de vie d'un scénario pédagogique 45
II.7 Positionnement de l'adaptation prédéfinie dans le cycle de vie d'un scénario . . 46
II.8 Calcul des parcours en fonction des réponses des apprenants 47
II.9 Positionnement de l'adaptation par conception itérative dans le cycle de vie d'un scénario 48
II.10 Temporalité d'une situation imprévue 51

II.11 Cadrage de la problématique d'adaptation à la volée 52

III.1 Alignement de vocabulaire entre l'approche EML et le Workflow 71
III.2 Ports de configuration des variants. Source [GVDAJVLR08] 75
III.3 Configurations des actions d'entrée/sortie d'un variant 76
III.4 Migration d'instances d'un modèle de workflow source à un modèle de workflow destination 77
III.5 Instances i et j conformes à un modèle de workflow (ou schéma) S, que l'on cherche à transformer en un modèle de workflow S'. (Source [Wes10]) 79
III.6 Positionnement des approches d'adaptation dans le cycle de vie du modèle de l'activité 81
III.7 Classification des approches d'adaptation à la volée selon leur nature et la temporalité des situations qu'elles gèrent 81

IV.1 Organisation en niveaux des informations liées à l'exécution d'un scénario . . . 85
IV.2 Exemple de liens entre les trois niveaux d'une activité se déroulant conformément à un scénario LDL 86
IV.3 Sens de propagation des modifications d'une adaptation à la volée 88
IV.4 Architecture du prototype d'adaptation par propagation 91
IV.5 Modification du niveau conception dans *module 1* 93
IV.6 Modification du niveau opérationnalisation dans LDI 93
IV.7 Interface du *module 2* : observation et d'adaptation des informations liées au niveau exécution 94
IV.8 Etat de visibilité de l'interaction *Discuss in the chat* après l'adaptation 94
IV.9 Modification de la visibilité des interactions dans le *player* des participants . . 95

V.1 Positionnement des Méta-modèles liés à l'opérationnalisation et à l'exécution dans le cycle de vie d'un scénario LDL 99
V.2 Méta-modèle lié à la phase d'opérationnalisation d'un scénario LDL dans LDI . 101
V.3 Méta-modèle lié à la phase d'exécution d'un scénario LDL dans LDI 102
V.4 Processus de spécification des modifications pour l'adaptation par propagation 104
V.5 Métal-modèle des modifications pour l'adaptation par propagation 106
V.6 Architecture du *Framework* d'adaptation par propagation 110
V.7 Diagramme d'activité d'une adaptation par propagation dans le *Framework* . . 112
V.8 Importation et analyse d'un modèle de modification au format XML 113

V.9 Fragment XML de l'action *Add* pour l'ajout du rôle *Expert* 114
V.10 Fragment XML pour l'ajout des liens d'opérationnalisation 114
V.11 Fragment XML de l'action *modification* au niveau exécution 115
V.12 Sélection de l'identifiant du scénario dans le *Framework* 115
V.13 Vue arborescente du modèle lié au niveau conception 116
V.14 Vue arborescente du modèle lié au niveau opérationnalisation 117
V.15 Vue arborescente du modèle lié au niveau exécution 117
V.16 Validation des modifications avant l'adaptation par propagation 118
V.17 Vue sur le niveau exécution après l'adaptation 119
V.18 Etat du *player* après l'adaptation 119

VI.1 Pattern d'adaptation pour l'ajout d'un rôle LDL par réutilisation 126

Bibliographie

[ABH04] A. Ahmad, O. Basir, and K. Hassanein. Adaptive user interfaces for intelligent e-Learning : issues and trends. In *Fourth International Conference on Electronic Business, ICEB2004, Beijing*, 2004.

[AL89] E. Auramaki and M. Leppanen. Exceptions and office information systems. In *Proceedings of the IFIP WG*, volume 8, 1989.

[Alt02] M. Altet. Une démarche de recherche sur la pratique enseignante : l'analyse plurielle. *Revue française de pédagogie*, 138(1) :85–93, 2002.

[BAK05] D. Burgos, M. Arnaud, and R. Koper. IMS Learning Design : la flexibilité pédagogique au service des besoins de la e-formation. `http://edutice.archives-ouvertes.fr/docs/00/28/50/55/HTML/index.html`, 2005.

[BB02] J. Bézivin and X. Blanc. Promesses et interrogations de l'approche MDA. *Developpeur Reference*, 2002.

[BC04] P. Béguin and Y. Clot. L'action située dans le développement de l'activité. *revue électronique*, page 35, 2004.

[BCCT05] M. Brambilla, S. Ceri, S. Comai, and C. Tziviskou. Exception handling in workflow-driven Web applications. In *Proceedings of the 14th international conference on World Wide Web*, pages 170–179. ACM, 2005.

[BDBF06] L. Botturi, M. Derntl, E. Boot, and K. Figl. A classification framework for educational modeling languages in instructional design. In *6th IEEE International Conference on Advanced Learning Technologies (ICALT 2006), Kerkrade (The Netherlands)*, 2006.

[BDJ+03] J. Bézivin, G. Dupé, F. Jouault, G. Pitette, and J.E. Rougui. First experiments with the ATL model transformation language : Transforming XSLT into XQuery. In *2nd OOPSLA Workshop on Generative Techniques in the context of Model Driven Architecture*, page 50. Citeseer, 2003.

[BMVW95] C. Bauzer-Medeiros, G. Vossen, and M. Weske. WASA : A Workflow-Based Architecture to Support Scientific Database Applications. *Lecture Notes in Computer Science*, pages 574–574, 1995.

[Bot06] L. Botturi. E 2 ML : A visual language for the design of instruction. *Educational Technology Research and Development*, 54(3) :265–293, 2006.

[Bou00] G. Bourguin. Un support informatique à l'activité coopérative fondé sur la Théorie de l'Activité : le projet DARE. *Unpublished PhD Thesis, Université des Sciences et technologies de Lille1, Lille*, 2000.

[Bru01] P. Brusilovsky. Adaptive hypermedia. *User modeling and user-adapted interaction*, 11(1) :87–110, 2001.

[BS06] D. Burgos and M. Specht. Adaptive e-Learning Methods and IMS Learning Design : An Integrated Approach. In *Advanced Learning Technologies, 2006. Sixth International Conference on*, pages 1192–1193. IEEE, 2006.

[BSB+04] J. Boticario, O. Santos, C. Barrera, E. Gaudioso, F. Hernandez, A. Rodriguez, P. Van Rosmalen, and R. Koper. Defining adaptive learning design templates for combining design and runtime adaptation in aLFanet. *Artificial Intelligence*, 2004.

[Bur07] D. Burgos. How to represent adaptation in e-learning with IMS learning design. *Interactive Learning Environments*, 15(2) :161–170, 2007.

[BW95] P. Barthelmess and J. Wainer. Workflow systems : a few definitions and a few suggestions. In *Proceedings of conference on Organizational computing systems*, pages 138–147. ACM, 1995.

[CRLNAR06] M. Caeiro-Rodriguez, M. Llamas-Nistal, and L. Anido-Rifón. The PoEML proposal to model services in educational modeling languages. *Groupware : Design, Implementation, and Use*, pages 187–202, 2006.

[CRMLN+07] M. Caeiro-Rodríguez, M.J. Marcelino, M. Llamas-Nistal, L. Anido-Rifón, and A.J. Mendes. Supporting the modeling of flexible educational units poeml : A separation of concerns approach. *Journal of Universal Computer Science*, 13(7) :980–990, 2007.

[Dal03] J. Dalziel. Implementing learning design : The learning activity management system (LAMS). In *AscilitE*, pages 1–10. Citeseer, 2003.

[Der05] M. Derntl. *Patterns for person centered e-learning.* Aka, 2005.

[DGJH+98] W. Deiters, T. Goesmann, K. Just-Hahn, T. L "offeler, and R. Rollers. Support for exception handling through workflow management systems. *CSCW, Towards Adaptive Workflow, Seattle, WA*, 1998.

[DMR+09] T. Djouad, A. Mille, C. Reffay, M. BENMOHAMED, L. LIRIS, B. LIFC, and A. LIRE. Ingénierie des indicateurs d'activités à partir de traces modélisées pour un environnement informatique d'apprentissage humain. *month*, 2009.

[Doy86] W. Doyle. Paradigmes de recherche sur l'efficacité des enseignants. *L'art et la science de l'enseignement*, pages 435–481, 1986.

[Eng87] Y. Engeström. *Learning by Expanding. An Activity-theoretical approach to developmental research.* Orienta-Konsultit Oy, Helsinki, 1987.

[Fav05] J.M. Favre. Foundations of meta-pyramids : Languages vs. metamodels–episode II : Story of thotus the baboon. *Language Engineering for Model-Driven Software Development*, 4101, 2005.

[FH09] C. Ferraris and A. Harrer. Competitive Challenge on Adapting Activities Modeled by CSCL Scripts. `http://www.isls.org/cscl2009/Pre-Conference7.htm`, 2009.

[FLVD05] C. Ferraris, A. Lejeune, L. Vignollet, and J.P. David. Modélisation de scénarios pédagogiques collaboratifs. In *Conférence EIAH*, volume 5, pages 285–296, 2005.

[GVDAJVLR08] F. Gottschalk, W.M.P. Van Der Aalst, M.H. Jansen-Vullers, and M. La Rosa. CONFIGURABLE WORKFLOW MODELS. *International Journal of Cooperative Information Systems*, 17(2) :177–221, 2008.

[HJ98] S. Horn and S. Jablonski. An approach to dynamic instance adaption in workflow management applications. In *Conference on Computer Supported Cooperative Work (CSCW)*, 1998.

[HN04] N. Henze and W. Nejdl. A logical characterization of adaptive educational hypermedia. *New review of hypermedia and multimedia*, 10(1) :77–113, 2004.

[HT04] S.Y. Hwang and J. Tang. Consulting past exceptions to facilitate workflow exception handling. *Decision Support Systems*, 37(1) :49–69, 2004.

[IMS11] Consorcium IMS. IMS Global Learning Consortium. `http://www.imsglobal.org/`, 2011.

[JAB+06] F. Jouault, F. Allilaire, J. Bézivin, I. Kurtev, and P. Valduriez. ATL : a QVT-like transformation language. In *Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications*, page 720. ACM, 2006.

[Jur06] M.B. Juric. *Business Process Execution Language for Web Services BPEL and BPEL4WS 2nd Edition*. Packt Publishing, 2006.

[KB97] MM Kwan and PR Balasubramanian. Dynamic workflow management : a framework for modeling workflows. In *System Sciences, 1997, Proceedings of the Thirtieth Hawaii International Conference on*, volume 4, 1997.

[KBT+00] P.J. Kammer, G.A. Bolcer, R.N. Taylor, A.S. Hitomi, and M. Bergman. Techniques for supporting dynamic and adaptive workflow. *Computer Supported Cooperative Work (CSCW)*, 9(3) :269–292, 2000.

[KC02] V. Kaptelinin and M. Cole. Individual and collective activities in educational computer game playing. *CSCL 2, carrying forward the conversation*, page 303, 2002.

[KD09] A. Karakostas and S. Demetriadis. Adaptation patterns in systems for scripted collaboration. In *Proceedings of the 9th international conference on Computer supported collaborative learning-Volume 1*, pages 477–481. International Society of the Learning Sciences, 2009.

[KM04] R. Koper and J. Manderveld. Educational modelling language : modelling reusable, interoperable, rich and personalised units of learning. *British Journal of Educational Technology*, 35(5) :537–551, 2004.

[KM08] R. Koper and Y. Miao. Using the ims ld standard to describe learning designs. *2008). Handbook of research on learning design and learning objects : Issues, applications and technologies. Information Science Reference*, 2008.

[KO04] R. Koper and B. Olivier. Representing the learning design of units of learning. *Educational Technology and Society*, 7(3) :97–111, 2004.

[KOA03] R. Koper, B. Olivier, and T. Anderson. IMS learning design information model. *IMS Global Learning Consortium*, 2003.

[Kop01] Rob Koper. Modeling units of study from a pedagogical perspective the pedagogical meta-model behind eml. *In Practice*, pages 1–40, 2001.

[KT05] R. Koper and C. Tattersall. *Learning design : A handbook on modelling and delivering networked education and training.* Springer Berlin Heidelberg, 2005.

[Kuu91] K. Kuutti. The concept of activity as a basic unit of analysis for cscw research. In *Proceedings of the second conference on European Conference on Computer-Supported Cooperative Work*, pages 249–264. Kluwer Academic Publishers, 1991.

[Kuu96] K. Kuutti. Activity theory as a potential framework for human-computer interaction research. *Context and consciousness : Activity theory and human-computer interaction*, pages 17–44, 1996.

[LDM+07] A. Lejeune, J.P. David, C. Martel, S. Michelet, and N. Vezian. To set up pedagogical experiments in a virtual lab : methodology and first results. In *International Conference ICL*, September 2007.

[Lej04] A. Lejeune. Ims learning design. *Distances et savoirs*, 2(4) :409–450, 2004.

[LPD04] D H Leo, J I A Perez, and Y A Dimitriadis. Ims learning design support for the formalization of collaborative learning patterns. *IEEE International Conference on Advanced Learning Technologies 2004 Proceedings*, pages 350–355, 2004.

[MA07] H. Mourão and P. Antunes. Supporting effective unexpected exceptions handling in workflow management systems. In *Proceedings of the 2007 ACM symposium on Applied computing*, pages 1242–1249. ACM, 2007.

[MAL07] S. Michelet, J.M. Adam, and V. Luengo. Adaptive learning scenarios for detection of misconceptions about electricity and remediation. *International Journal of Emerging Technologies in Learning (iJET)*, 2(1), 2007.

[Man02] D.A. Manolescu. Workflow enactment with continuation and future objects. In *Proceedings of the 17th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, pages 40–51. ACM, 2002.

[MBS05] C.D. Milligan, P. Beauvoir, and P. Sharples. The reload learning design tools. *Journal of Interactive Media in Education*, 2005(1), 2005.

[MVF+06] C. Martel, L. Vignollet, C. Ferraris, J.P. David, and A. Lejeune. Ldl : an alternative eml. In *Advanced Learning Technologies, 2006. Sixth International Conference on*, pages 1107–1108. IEEE, 2006.

[MVFD06] C. Martel, L. Vignollet, C. Ferraris, and G. Durand. Ldl : a language to model collaborative learning activities. In *EDMEDIA 2006, World Conference on Educational Multimedia, Hypermedia and Telecommunications*, pages 838–844, 2006.

[OMG05] OMG. *MOF QVT Final Adopted Specification*. Object Management Group, June 2005.

[OUN11] OUNL. The IMS Learning Design Engine. `http://coppercore.sourceforge.net/index.shtml`, 2011.

[PG06] J.P. Pernin and H. Godinet. Scénariser l'enseignement et l'apprentissage : une nouvelle compétence pour le praticien. `http://www.inrp.fr/publications/edition-electronique/documents-travaux-recherche-education/BR056.pdf`, 2006.

[PL04] J.P. Pernin and A. Lejeune. Dispositifs d'apprentissage instrumentés par les technologies : vers une ingénierie centrée sur les scénarios. *TICE 2004*, pages 407–414, 2004.

[Plo11] Plone. Système de gestion de contenu Web. `http://plone.org/`, 2011.

[PRCRAR10] R. Perez-Rodriguez, M. Caeiro-Rodriguez, and L. Anido-Rifon. Adaptation in a PoEML-based e-learning platform. In *Education Engineering (EDUCON), 2010 IEEE*, pages 1681–1690. IEEE, 2010.

[PRCRARLN10] R. Perez-Rodriguez, M. Caeiro-Rodriguez, L. Anido-Rifon, and M. Llamas-Nistal. Execution Model and Authoring Middleware Enabling Dynamic Adaptation in Educational Scenarios Scripted with PoEML. *Journal of Universal Computer Science*, 16(19) :2821–2840, 2010.

[RST04] M. Relieu, P. Salembier, and J. Theureau. Introduction au numéro spécial «activité et action/cognition située». *Revue Activites. org*, 1(2) :1–10, 2004.

[RVRK+02] Adrian Rawlings, P Van Rosmalen, Rob Koper, Miguel Rodriguez-Artacho, and Paul Lefere. Survey of educational modelling languages (emls). *CENISSS Learning Technologies Workshop*, page 80, 2002.

[SB06] M. Specht and D. Burgos. Implementing Adaptive Educational Methods with IMS Learning Design. *Adaptive Hypermedia 2006*, 2006.

[SBdCS07] O.C. Santos, J.G. Boticario, E. del Campo, and M. Saneiro. IMS-LD as a workflow to provide personalized support for disabled students in Higher Edu-

cation institutions. In *11th International Conference on User Modeling–UM 2007 Corfu, Greece*, page 41. Citeseer, 2007.

[Sha02] R. Shapiro. A technical comparison of xpdl, bpml and bpel4ws. *Cape Visions*, 2002.

[SK03] S. Sendall and W. Kozaczynski. Model transformation : The heart and soul of model-driven software development. *IEEE software*, 20(5) :42–45, 2003.

[Slo06] A. Slomiski. On using BPEL extensibility to implement OGSI and WSRF grid workflows. *Concurrency and Computation : Practice and Experience*, 18(10) :1229–1241, 2006.

[SM95] D.M. Strong and S.M. Miller. Exceptions and exception handling in computerized information processes. *ACM Transactions on Information Systems (TOIS)*, 13(2) :206–233, 1995.

[SMS94] H. Saastamoinen, M. Markkanen, and V. Savolainen. *Survey on exceptions in office information systems*. University of Colorado, Boulder, Dept. of Computer Science, 1994.

[Suc87] L.A. Suchman. *Plans and situated actions : the problem of human-machine communication*. Cambridge Univ Pr, 1987.

[Suc94] L.A. Suchman. *Plans and situated actions : The problem of human-machine communication*. Cambridge University Press, 1994.

[Tch02] P. Tchounikine. Pour une ingénierie des Environnements Informatiques pour l'Apprentissage Humain. *Revue I3*, 2(1) :59–95, 2002.

[Tch09] P. Tchounikine. Précis de recherche en Ingénierie des EIAH. `http://hal.archives-ouvertes.fr`, 2009.

[TEN11] TENCompetence. TENCompetence Foundation. `http://www.tencompetence.org/web/guest`, 2011.

[TL99] M. Tardif and C. Lessard. Le travail enseignant au quotidien. *Contribution à l'étude du travail dans les métiers et les professions d'interactions humaines. Belgique : De Boeck*, 1999.

[VM06] H. Vogten and H. Martens. CopperCore. *Retrieved February 9th*, 2006.

[Wes99] Mathias Weske. *Adaptive Workflows based on Flexible Assignment of Workflow Schemas and Workflow Instances Ű Extended Abstract*, pages 1–7. 1999.

[Wes10] M. Weske. Adaptive Workflows based on Flexible Assignment of Workflow Schemas and Workflow Instances. http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-24/wes99.pdf, 2010.

[Whi04] S.A. White. Introduction to BPMN. *IBM Cooperation*, pages 2008–029, 2004.

[ZDF06] T. Zarraonandia, J.M. Dodero, and C. Fernandez. Crosscutting Runtime Adaptations of LD Execution. *JOURNAL OF EDUCATIONAL TECHNOLOGYAND SOCIETY*, 9(1) :123, 2006.

[ZDFPD07] T. Zarraonandia, J. Dodero, C. Fernández, and I. Paloma Díaz. Iterative design of learning processes. *Computers and Education*, pages 163–177, 2007.

[Zop11] Zope. Serveur d'application web orienté objet. http://www.zope.org/, 2011.