MODELISATION ET PROTOTYPE

1. Introduction

Dans ce chapitre nous décrivons l'implémentation que nous avons faite des différentes propositions faites dans ce travail. L'intégration logicielle est faite dans le logiciel COLORS que nous avons développé et qui est un environnement auteur d'objets d'apprentissage SCORM et IMS travaillant sur un ensemble de banques d'objets distantes. Après avoir décrit COLORS, nous décrivons son extension aux objets SIMBAD. La plus grande partie des analyses présentées dans ce travail est implémentée via la logique des frames. Nous allons présenter le langage F-logique et la plateforme OntoBroker que nous avons utilisés pour la mise en œuvre de notre approche. Puis, nous allons présenter notre modélisation des objets SIMBAD conformément à ce formalisme. Enfin nous décrivons l'implémentation des différentes analyses en se focalisant sur les règles de conformité.

2. Système COLORS

2.1. Contexte

De nos jours le concept d'objet d'apprentissage est considéré comme l'un des concepts les plus prometteurs pour l'avenir de la formation en ligne. L'un des principaux apports de ce concept est le fait de promouvoir la réutilisation des contenus éducatifs. En fait, la mise en place des formations en ligne est marquée par le coût élevé de la réalisation d'un contenu sous une forme numérisée. En plus, force est de constater une forte ressemblance entre les contenus des formations similaires. En conséquence, la réutilisation de l'existant est une piste prometteuse pour tout organisme souhaitant mettre en place une formation en ligne avec des facteurs de succès et de rentabilité notamment un retour sur investissement et une capitalisation de l'existant.

Cependant, il est nécessaire d'allier aux objets d'apprentissage les outils informatiques nécessaires pour une exploitation réelle, efficace et rentable. Parmi les outils concernés nous

pouvons citer les outils d'édition (RELAOD, eXe, etc.), les LCMS[10] (MOODLE, INES, CLAROLINE, etc.) et les banques d'objets d'apprentissage (eRIB, PLANET, ARIADNE, DOOR, etc). C'est aux banques d'objets d'apprentissage que nous nous somme intéressés.

Les banques d'objets d'apprentissage représentent une composante fondamentale dans l'infrastructure logicielle de toute institution adoptant la formation en ligne. Leur rôle principal est de rassembler en un endroit unique, au moins d'un point de vue logique et pas forcément physique, les objets d'apprentissage. Il est alors plus facile de trouver, de gérer et d'utiliser les objets d'apprentissage à partir d'un point d'accès unique. Ce point d'accès, qui est la banque d'objets d'apprentissage, permet d'éviter une anarchie inévitable avec des contenus redondants, invisibles ou fictifs.

Toutefois, ce n'est pas le concept intrinsèque de banque d'objets d'apprentissage qui a suscité notre attention, d'un point de vue recherche, mais plutôt le questionnement sur son utilisation dans un cadre de coopération inter-organisationnelle et à grande échelle. Encore une fois, cette perspective à été depuis quelques années étudiée. En effet, le concept même d'objets d'apprentissage se manifeste comme catalyseur pour la réutilisation et l'interopérabilité. D'autre part, ces deux éléments représentent les ingrédients de base pour toute coopération entre les institutions œuvrant dans la sphère de la formation en ligne. Ainsi, le concept d'objets d'apprentissage doit être un élément clé pour favoriser une synergie entre ces institutions. Ceci explique la mise en place de projets de création de réseaux de banques d'objets d'apprentissage de part et d'autre du monde.

Cependant, les réseaux qui ont été mis en place souffrent d'une contextualisation forte au point qu'il est difficile de les réutiliser ailleurs. C'est-à-dire ils n'ont pas été faits généralement dans une optique de transfert ou d'adaptation à d'autres bénéficiaires. Autrement dit, la seule possibilité qui peut être négociée c'est celle d'intégrer ces réseaux déjà développés sans avoir la possibilité de créer ses propres réseaux. Notre réponse à cette situation c'est le projet intitulé COLORS pour **CO**operative **L**earning **O**bject **R**epository **S**ystem.

En fait, l'objectif est de permettre à plusieurs établissements universitaires de partager à la fois leurs objets d'apprentissage en interne tout en ayant la possibilité d'intégrer un réseau de

[10] LCMS : Learning Content Management System

banques d'objets d'apprentissage dans un cadre de partenariat et de coopération avec des partenaires externes. Il s'agit de permettre la réutilisation des objets d'apprentissage localement et éventuellement à grande échelle. C'est ainsi que nous avons décidé de mettre en place une infrastructure logicielle de partage d'objets d'apprentissage à une échelle multi-organisationnelle supportant l'esprit de coopération et d'ouverture. Afin d'atteindre cet objectif COLORS doit être évolutif, interopérable avec l'existant, flexible et portable. Présentons tout d'abord COLORS avant de revenir sur la façon avec laquelle nous avons répondu à ces objectifs.

2.2. Présentation

COLORS est une banque d'objets d'apprentissage Open source (License GNU GPL : GNU GENERAL PUBLIC LICENSE Version 2, June 1991) développée en langage PHP à base du code source de DOOR[11] qui est également une banque d'objets Open Source. Par rapport à son prédécesseur, COLORS garde le fait qu'il y a trois acteurs : l'administrateur, l'auteur et le simple utilisateur. Côté fonctionnalités, COLORS a hérité les fonctions de gestion des utilisateurs, des objets d'apprentissage et les fonctions de recherche et d'exploration du contenu de la banque.

Toutefois, COLORS se distingue par une prise en compte native de la langue arabe. Elle est interopérable avec MOODLE[12] grâce à un plugin spécifique. En plus, elle est complètement conforme aux spécifications IMS Content Package 1.1.3, SCORM 1.2 et SCORM 2004. Cependant, l'originalité la plus marquante de COLORS est qu'elle offre la possibilité à une banque COLORS de s'intégrer à un ou plusieurs réseaux de banques d'objets d'apprentissage COLORS. De plus amples détails sont donnés dans l'annexe C.

[11] http://door.sourceforge.net
[12] http://moodle.org

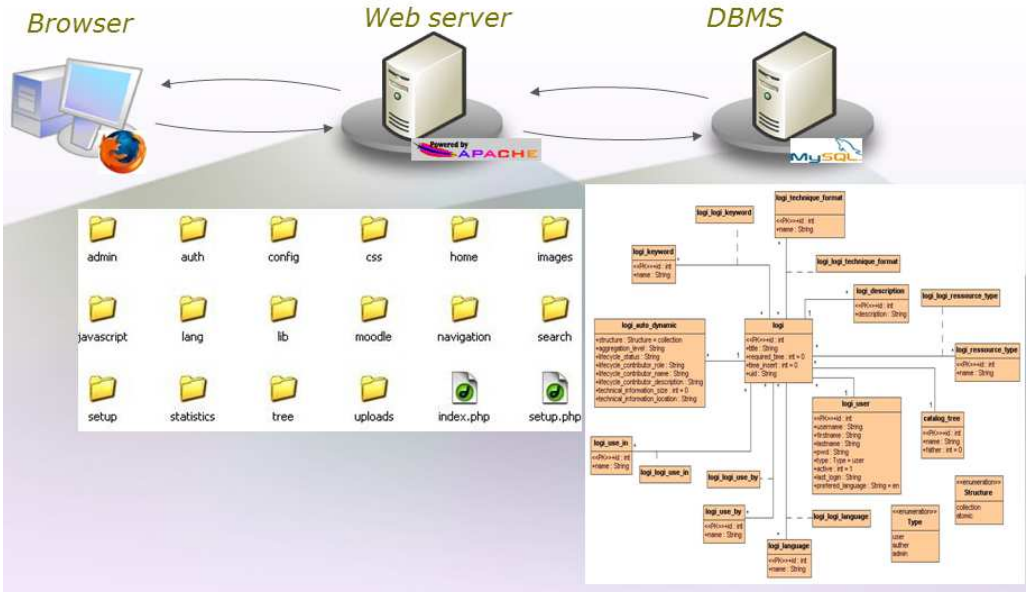


Figure 39 : Architecture logicielle du système COLORS

2.3. Coordinateur de banques COLORS

En fait, pour mettre en place un réseau de partage d'objets d'apprentissage il faut plusieurs instances de COLORS et un coordinateur pour le réseau de banques. Ce coordinateur, que nous avons développé, est une application Web open source simple à administrer. Elle joue le rôle d'un médiateur entre les banques d'un même réseau. Elle permet de gérer la procédure d'inscription permettant à des institutions de mutualiser leurs objets d'apprentissage. C'est au niveau de cette composante, qui forme le cœur d'un réseau de banques d'objets d'apprentissage, que nous avons implémenté une approche de fédération de ressources et que nous avons fixé les règles de propagation des requêtes et d'agrégation des résultats.

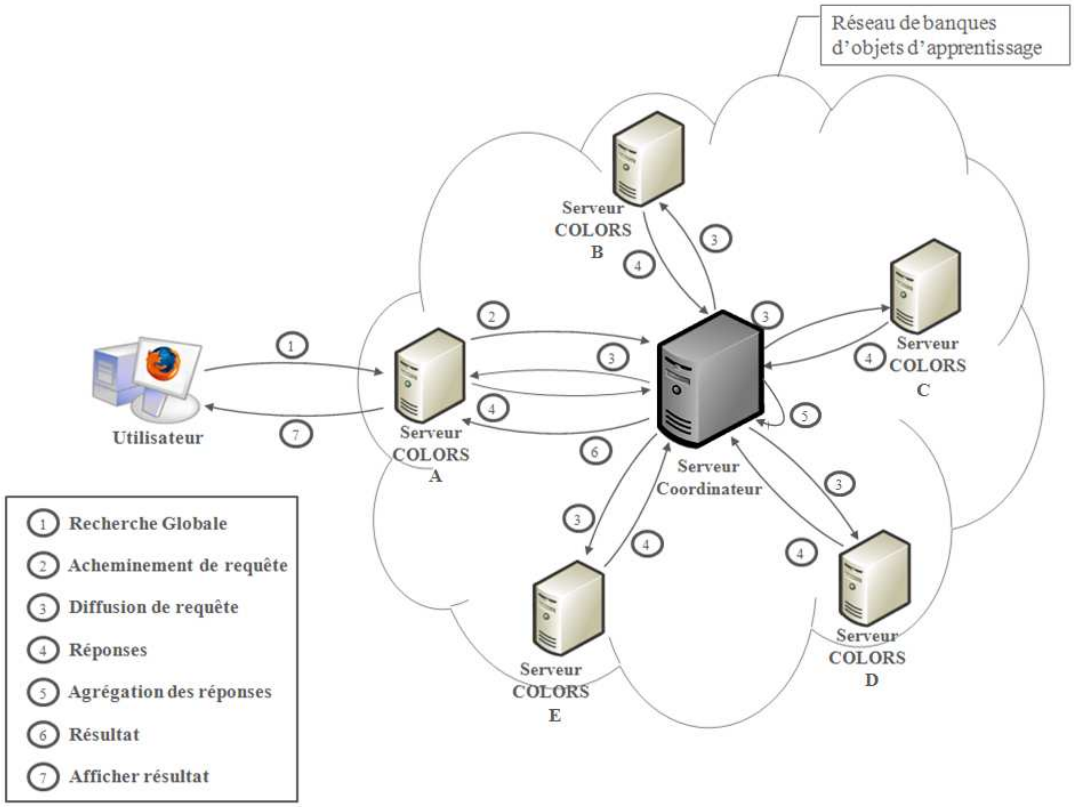


Figure 40 : Gestion des requêtes au niveau d'un réseau de banques COLORS

Ce coordinateur supporte deux types d'acteurs. Le premier c'est l'administrateur qui a comme rôle principale de gérer (activer, bloquer ou supprimer) les instances de COLORS qui forment le réseau de banques, de consulter des statistiques, de modifier les conditions d'utilisation, de reconfigurer le système, et de gérer les paramètres de son compte. Le deuxième acteur c'est tout utilisateur ou utilisateur potentiel. Il pourra consulter les conditions d'utilisation fixées par la communauté qui détient le réseau de banques. Il peut également consulter la liste des institutions qui forment le réseau.

Figure 41 : Interface de gestion des banques d'un réseau COLORS par le coordinateur

2.4. Assistance aux auteurs via COLORS

Les banques d'objets d'apprentissage offrent aux auteurs des fonctionnalités classiques liées au dépôt des objets d'apprentissage. Elles consistent le plus souvent à l'importation de l'objet et à la gestion des métadonnées. Avec COLORS nous souhaitons offrir aux auteurs des outils supplémentaires issus de notre approche d'assistance.

En fait, dans la version actuelle du système COLORS nous avons introduit un module simple d'analyse qui porte uniquement sur les métadonnées éducatives. L'objectif étant d'étudier dans un premier temps la façon avec laquelle l'analyse pour la compréhension des objets d'apprentissage peut être intégrée et présentée comme un service supplémentaire aux auteurs. Dans ce contexte nous avons inclus un module permettant de comparer les propriétés de l'objet par rapport à ceux des dix objets les plus consultés dans la banque COLORS [Farhat et Jemni, 2009].

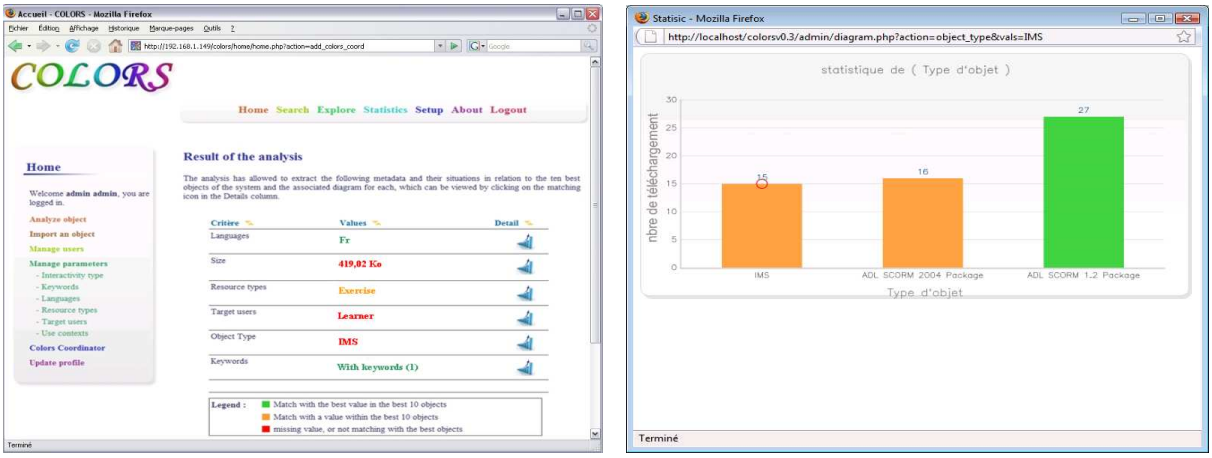


Figure 42 : Interfaces COLORS relatives à l'analyse des propriétés des objets d'apprentissage

Comme le montre la capture d'écran à gauche, les résultats de l'analyse sont affichés par critère et en trois couleurs. La couleur verte indique que pour ce critère l'objet a une valeur qui correspond à celle de l'objet le plus consulté dans la banque. Si la couleur est orangée, la valeur correspond à l'une des valeurs des dix objets les plus consultés. La couleur rouge indique que la valeur ne correspond à aucune valeur des dix objets les plus téléchargés. L'auteur peut consulter les détails en cliquant sur le lien correspondant. Par exemple, il peut voir les détails concernant le type de l'objet d'apprentissage. Ainsi, l'auteur peut constater que la plupart des utilisateurs de cette banque d'objets d'apprentissage préfèrent des objets au format SCORM 1.2 plutôt que IMS ou SCORM 2004 comme le montre la capture d'écran à droite. Ceci peut être expliqué par le fait que les utilisateurs utilisent plutôt des systèmes de gestion de l'apprentissage compatible avec les objets SCORM 1.2. Donc, pour favoriser l'usage de ses objets d'apprentissage l'auteur a intérêt à concevoir des objets dans ce format.

2.5. Extension de COLORS aux objets SIMBAD

Notre approche d'assistance prend réellement intérêt dès que la structure et la sémantique des objets d'apprentissage atteint un certain niveau de complexité. C'est le cas des objets composés SIMBAD. Ainsi, nous avons proposé une extension pour le système COLORS qui prévoit la gestion des objets composés SIMBAD. Pour cela, nous avons remplacé le module d'analyse en PHP de COLORS par un ensemble de modules construits avec la logique des frames et le logiciel Ontobroker. Ces modules supposent que les objets d'apprentissage en entrées soient conformes au format SIMBAD (exprimé en F-logique). Nous avons donc ajouté un module de traduction SCORM et IMS vers SIMBAD, ce qui permet d'analyser un grand nombre d'objets d'apprentissage, même si cette analyse est moins fouillée que celle des objets SIMBAD.

Voici l'architecture que nous proposons :

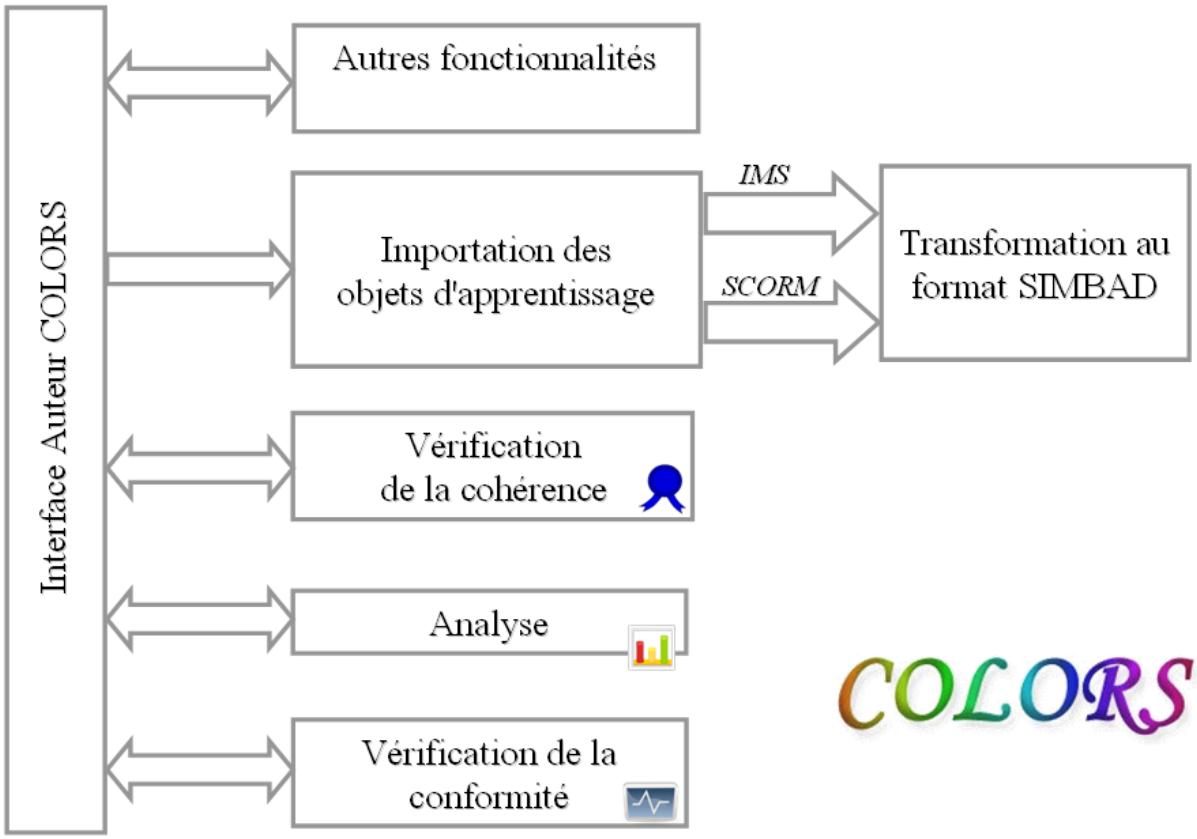


Figure 43 : Architecture de l'extension de COLORS

L'implémentation de notre approche est basée sur l'utilisation de la programmation logique. Ainsi, nous allons introduire le langage F-logique et le moteur d'inférence OntoBroker.

3. Logique des frames

3.1. Introduction

D'après [Fensel 2000] si la logique des prédicats offre comme primitive de modélisation la notion de prédicat, d'autres approches proposent une modélisation basée sur d'autres primitives. En effet, il existe des approches basées sur les « Frames » (classes) et les attributs comme primitives de modélisation. La logique des Frames, ou F-logique, fait partie de ces approches.

L'utilisation de la F-logique est motivée tout d'abord par sa capacité à répondre à nos besoins en termes d'expression et d'inférence. En effet, elle est capable d'exprimer les différentes composantes et propriétés d'un objet d'apprentissage. En plus, elle est capable d'inférer

différentes informations permettant la mise en place de notre approche d'assistance aux auteurs d'objets d'apprentissage composés.

Outre les caractéristiques intrinsèques de la F-logique, son utilisation dans le cadre de cette thèse a été favorisée par le fait qu'il s'agit d'un formalisme déjà utilisé au sein du projet SIMBAD. Ceci nous permet de profiter de l'expertise et du savoir faire existant. D'autre part, nous disposons d'une licence du système OntoBroker qui est un moteur d'inférences efficace pour la F-logique.

3.2. Présentation de la F-logique

D'après [Fensel 2000] la F-logique permet la spécification de bases de données orientées objet, de systèmes à base de classes et de faire de la programmation logique. Son principal apport, par rapport à des approches concurrentes c'est l'intégration de constructeurs conceptuels de modélisation : les classes, les attributs, les restrictions de domaine et de portée, l'héritage et les axiomes. Les axiomes logiques permettent d'exprimer, dans le cas de modélisation d'ontologies, les relations entre les éléments de cette ontologie et leurs instances. Toutes ces primitives de modélisation sont introduites dans un cadre logique cohérent.

L'alphabet de la F-logique consiste en un ensemble de symboles de fonctions et un ensemble de variables. Un terme est un terme usuel de la logique de premier ordre composé de symboles de fonctions et de variables.

3.3. Langage F-logique

Le langage associé à la F-logique est constitué d'un ensemble de formules construites à partir des symboles de l'alphabet. Comme dans plusieurs autres logiques, les formules peuvent être construites à partir de formules simples par l'utilisation des connecteurs usuels (non / et /ou) et des quantificateurs universel et existentiel [Ontoprise 2006b].

Avec le langage F-logique il est possible de définir trois types d'expressions : les faits, les règles et les requêtes.

- Exemple de fait : L0_001 est un objet d'apprentissage

```
LO_001 : LearningObject.
```

- Exemple de règle : Pour tout X et Y tel que X est le concept « algèbre relationnelle » et Y est le concept « calcul relationnel », on a X est relié à Y par la relation rhétorique « contraste ».

```
FORALL X,Y   X[contrast->Y] <- X:RelationalAlgebra AND Y:RelationalCalculus.
```

- Exemple de requête : Donnez l'ensemble de tous les objets d'apprentissage.

```
FORALL X <- X:LearningObject
```

4. Modélisation des objets d'apprentissage SIMBAD en F-logique

4.1. Description conceptuelle

Afin de pouvoir mettre en œuvre notre approche d'assistance des auteurs des objets d'apprentissage composés dans le cadre de l'approche SIMBAD, nous avons commencé par identifier les éléments que nous devons modéliser en F-logique et les relations entre eux sous forme d'un diagramme de classes.

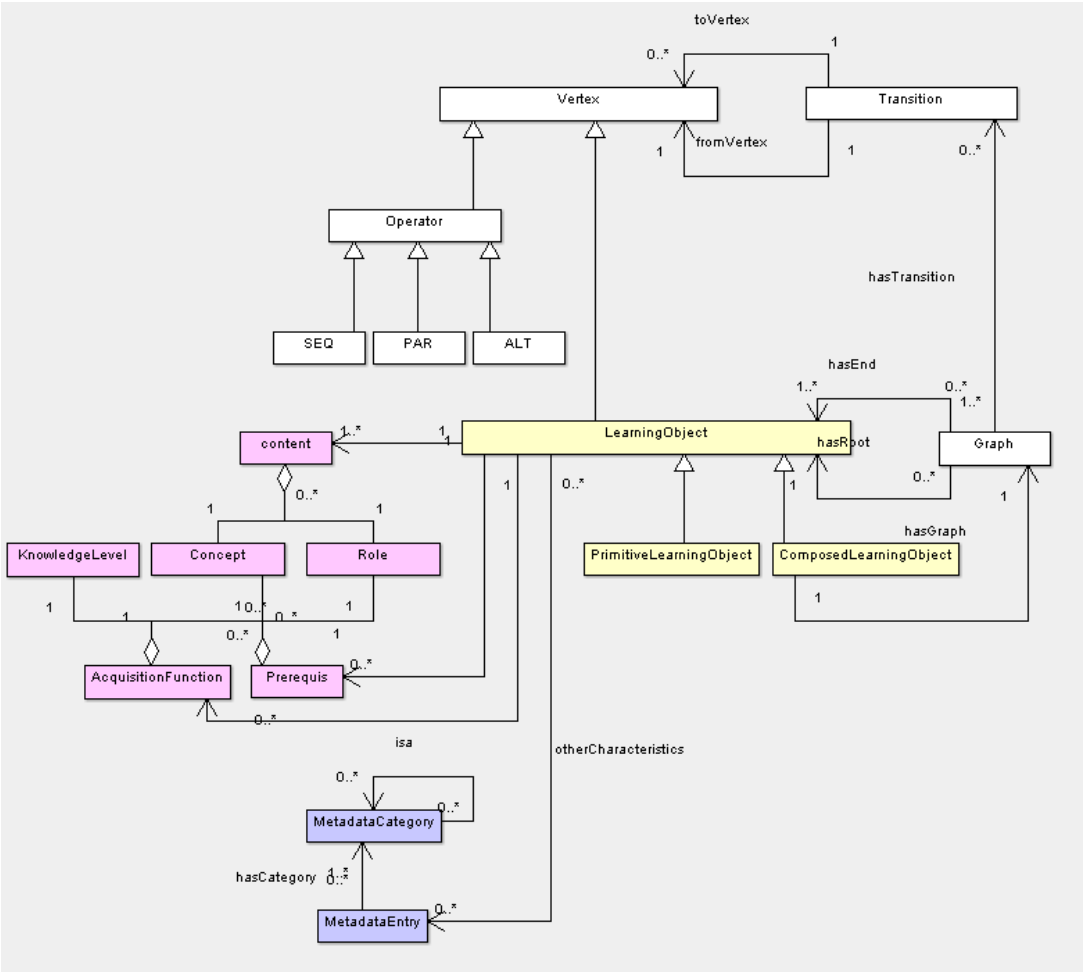


Figure 44 : Différents éléments relatifs aux objets d'apprentissage SIMBAD

L'élément central à considérer est l'objet d'apprentissage qui se décline en deux variantes : les objets simples et les objets composés. Il est décrit par un contenu, des pré-requis et une fonction d'acquisition. Ces éléments représentent les métadonnées sémantiques à qui sont associées les métadonnées LOM constituées par des entrées organisées en catégories.

A chaque objet d'apprentissage composé est associé un graphe de composition abstrait. Celui-ci comporte des arcs et des sommets. Ces derniers se déclinent en opérateurs et sommets avec leurs différentes variantes comme décrit par le diagramme de classe ci-dessous.

4.2. Objet d'apprentissage

En F-logique on utilise la notion de signature pour définir les méthodes applicables aux instances de classes. Ainsi, la signature déclare la méthode et spécifie les restrictions concernant les paramètres et les résultats.

Pour distinguer les signatures des données, les flèches utilisées pour les signatures sont doubles (=) et leurs sommets indiquent si le résultat est une valeur (=>) ou un ensemble de valeurs (=>>). Par exemple, pour spécifier que la méthode « hasContent » est applicable à un objet d'apprentissage et qu'elle a pour paramètre un rôle et comme résultat des concepts, nous devons utiliser la signature suivante :

```
LearningObject [hasContent@(Role) =>> Concept].
```

Une instance d'un objet d'apprentissage prend la forme suivante :

```
PrimitiveLearningObject :: LearningObject

LO_001 : PrimitiveLearningObject.

LO_001 [hasContent@(introduction) -> database].
```

Ici nous avons spécifié que LO_001 est une instance de la classe « PrimitiveLearningObject ». Cette classe est une sous-classe de la classe « LearningObject » et donc hérite de ses méthodes. Ainsi, nous avons pu spécifier que cet objet a comme contenu le concept « database » avec le rôle « introduction ». Voici les signatures relatives aux objets d'apprentissage :

```
LearningObject
      [
            hasContent@(Role) =>> Concept;
            hasPrerequisite@(Role,KnowledgeLevel) =>> Concept;
            hasAcquisitionFunction@(Role,KnowledgeLevel) =>> Concept;
```

```
            otherCharacteristics =>> MetadataEntry
      ].


PrimitiveLearningObject :: LearningObject.
PrimitiveLearningObject
      [
            hasURL=>String
      ].


ComposedLearningObject :: LearningObject.
ComposedLearningObject
      [
            hasGraph => Graph;
      ].
```

Par la suite il faut définir les instances des objets d'apprentissage conformément à la signature.

L'instanciation d'un objet d'apprentissage simple prend ainsi la forme suivante :

```
lo_001 : PrimitiveLearningObject.
lo_001
[
      hasContent@(introduction) -> database;
      hasPrerequisite@(introduction, medium) ->> informatics ;
      hasAcuisitionFunction@(introduction, beginner)->> database ;
      hasURL -> "http://localhost/lo/lo_001.html" ;
      otherCharacteristics   ->>   {lo_001_Category,   lo_001_Entry,   lo_001_Title,
lo_001_Title, lo_001_Language}
].
```

L'instanciation d'un objet d'apprentissage composé prend la forme suivante :

```
lo_901 : ComposedLearningObject.
lo_901
[
```

```
	hasPrerequisite@() -> ;
	hasContent@(introduction) -> database;
	hasAcquisitionFunction@(introduction, beginner) -> database;
	hasGraph -> graph_901;
	otherCharacteristics ->> {lo_901_Catalog, lo_901_Entry, lo_901_Title, lo_901_Language}
].
```

4.3. Graphe de composition

Chaque objet d'apprentissage composé est décrit par un graphe de composition abstrait (« *Graph* »). Celui-ci est décrit par l'ensemble des arcs (« *Transition* ») qui le décrit. De plus, chaque graphe de composition se distingue par un sommet racine et des sommets feuilles qui sont des objets d'apprentissage.

```
Graph :: ContentNode.
Graph
[
	hasRoot => LearningObject;
	hasEnd =>> LearningObject;
	hasTransition =>> Transition;
].
```

Chaque arc à une origine et une destination qui sont des sommets (*« Vertex »*). Chaque sommet est soit un opérateur (*« Operator »*) ou un objet d'apprentissage (*« LearningObject »*). Voici la signature que nous avons définie pour un graphe en F-logique :

```
Transition :: GraphObject.
Transition
[
	toVertex => Vertex;
	fromVertex => Vertex;
].
```

```
Operator ::  Vertex .

LearningObject :: Vertex .
```

D'autres éléments relatifs à SIMBAD ont été décrits en F-logique tels que le modèle de domaine ou les métadonnées LOM. Nous allons les présenter au niveau de l'annexe A.

4.4. Exemple illustratif

Supposant que l'objet d'apprentissage composé LO_901, introduit dans la section **3.1** de ce chapitre, a le graphe de composition abstrait suivant :

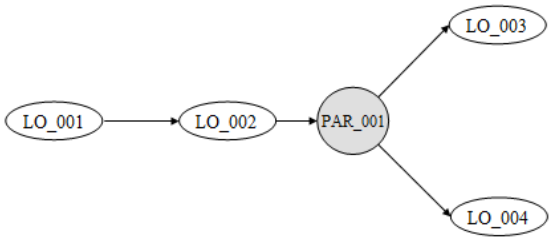


Figure 45 : Exemple d'un graphe de composition abstrait

Voici la description du graphe ci-dessus :

```
graph_001 : Graph.

graph_001 [

      hasRoot -> LO_001;

      hasEnd ->> {LO_003, LO_004};

      hasTransition    ->>    {transition_001,    transition_002,    transition_003,
transition_004}

].

transition_001 : Transition.

transition_001 [

      fromVertex -> LO_001;

      toVertex -> LO_002

].

transition_002 : Transition.

transition_002 [

      fromVertex -> LO_002;

      toVertex -> op_par_001

].
```

```
transition_003 : Transition.

transition_003 [

       fromVertex -> op_par_001;

       toVertex -> LO_003

].

transition_004 : Transition.

transition_004 [

       fromVertex -> op_par_001;

       toVertex -> LO_004

].
```

5. Analyse des objets SIMBAD

5.1. Architecture

Afin, de mettre en œuvre les différents mécanismes d'assistance aux auteurs des objets d'apprentissage composés nous avons mis au point un ensemble de modules développés en langage Java et basés sur l'utilisation de la programmation logique via le langage F-logique. Le schéma suivant décrit l'architecture suivie :

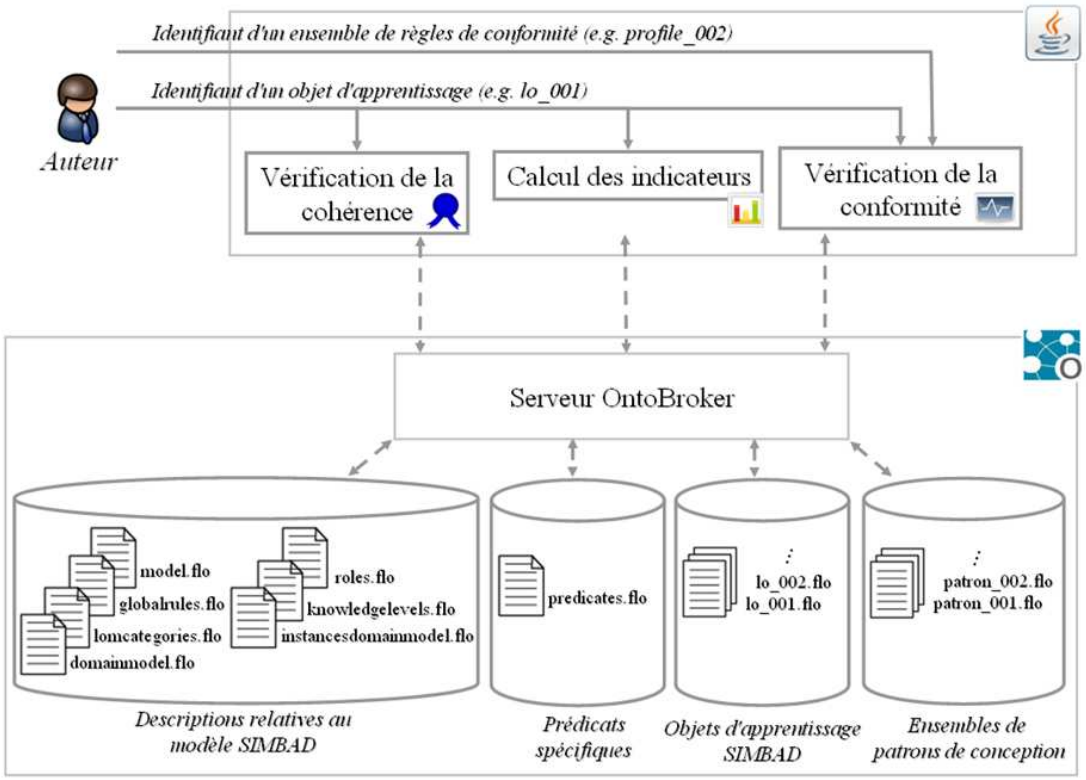


Figure 46 : Architecture expérimentale

Il s'agit d'une architecture client/serveur avec une partie client développée en langage Java et en une partie serveur formée d'OntoBroker le moteur d'inférence associé au langage F-logique.

Le langage Java est utilisé principalement pour simplifier la manipulation pour l'utilisateur (il n'a pas à connaitre le langage F-logique). Toutefois, les algorithmes que nous utilisons pour la vérification de la cohérence, l'analyse multicritères et sémantique et la vérification de la conformité nécessite un langage qui supporte les expressions conditionnelles et itératives. Ainsi, les indicateurs et les données nécessaires pour l'exécution des traitements sont demandées à OntoBroker via des requêtes en F-logique.

Le serveur OntoBroker est lancé avec une configuration basée sur un ensemble de fichiers comportant les éléments nécessaire pour pouvoir assister l'auteur dans la composition d'un objet d'apprentissage composé.

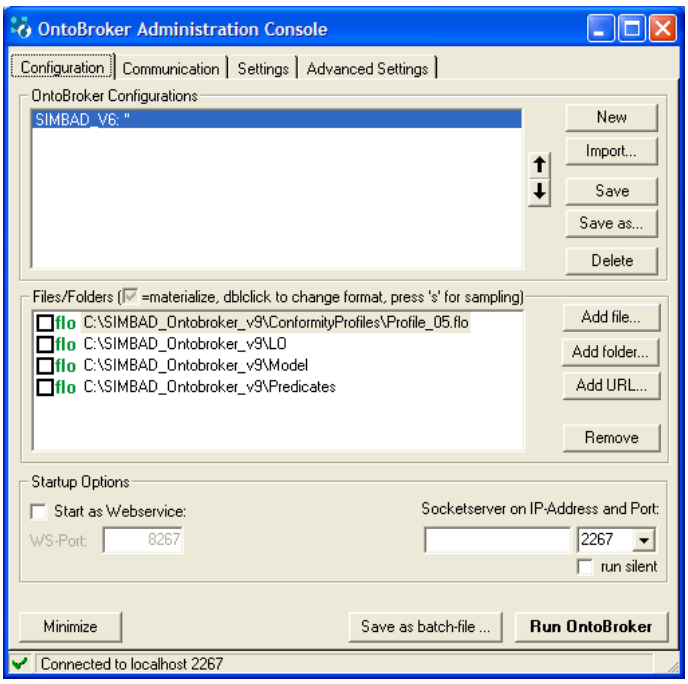


Figure 47 : Console d'administration OntoBroker 5.0

Les fichiers sont organisés en quatre répertoires :

– *Model* : Il contient sept fichiers qui décrivent les éléments du modèle SIMBAD

  – *Model.flo* : contient les signatures nécessaires pour définir des instances d'objets d'apprentissage, de graphes de composition et de métadonnées.

  – *GlobalRules.flo* : contient les règles qui permettent de classer les objets d'apprentissage en conformes ou pas conformes aux règles de conformité.

- *LOMCategories.flo* : contient la signature et les instances des catégories et des entrées de métadonnées LOM.
- *DomainModel.flo* : contient les signatures relatives au modèle de domaine en termes de concepts et de relations inter-concepts.
- *InstancesDomainModel.flo* : contient les instances relatives au modèle de domaine.
- *Roles.flo* : contient la définition des rôles éducatifs.
- *KnowledgeLevels.flo* : contient les niveaux de connaissances à considérer pour la définition des pré-requis et des fonctions d'acquisition.

- *LO* : contient les fichiers qui décrivent les différents objets d'apprentissage.
- *Predicates* : contient les différents prédicats de haut niveau qui ont été défini pour simplifier les manipulations spécifiques aux objets d'apprentissage.
- *ConformityProfiles* : contient les différents ensembles de règles de conformité.

5.2. Implémentation

Nous avons implémenté la partie client de l'architecture en langage Java afin d'expérimenter les différents algorithmes proposés et pour tester les mécanismes d'assistance aux auteurs que nous avons proposés.

Le diagramme de classe proposé dans cette section montre les différentes classes que nous avons implémentées et les relations établies entre elles. Pour des raisons de lisibilité du diagramme nous avons montré uniquement les attributs et les méthodes publiques.

La classe « ComposedLearningObject » est la classe centrale dans notre implémentation. Elle offre les différents services d'assistance via ses méthodes. A cette classe est associé la classe « CompositionGraph » qui offre les services relatifs aux aspects structurels des graphes de composition. Deux classes héritent d'elle : « AbstractCompositionGraph » et « ConcreteCompositionGraph ». La première offre en plus des services spécifiques au graphe de composition abstrait. La deuxième est relative au graphe de composition concret généré par la classe « ConcreteGraphGenerator ». La classe nommée « DeliveryGraphGenerator » permet de générer les graphes affichables à partir d'un graphe de composition concret.

A l'objet d'apprentissage composé est associé les classes « LOM » et « SemanticMetadata » qui héritent de la classe « Metadata ». Chacune (des deux classes) offre des services relatifs à chaque famille de métadonnées via leurs méthodes respectives.

La classe « SemanticSpace » offre des services relatifs à l'espace sémantique de l'objet d'apprentissage composé. Elle fait appel aux services relatifs aux métadonnées sémantiques et au graphe de composition abstrait.

La classe « OntoBrokerClient » permet aux différents objets de notre application de communiquer avec le serveur OntoBroker. Cette classe présente plusieurs services dont la possibilité d'envoyer des requêtes ainsi que d'ajouter ou de supprimer des faits de la base de connaissances OntoBroker.

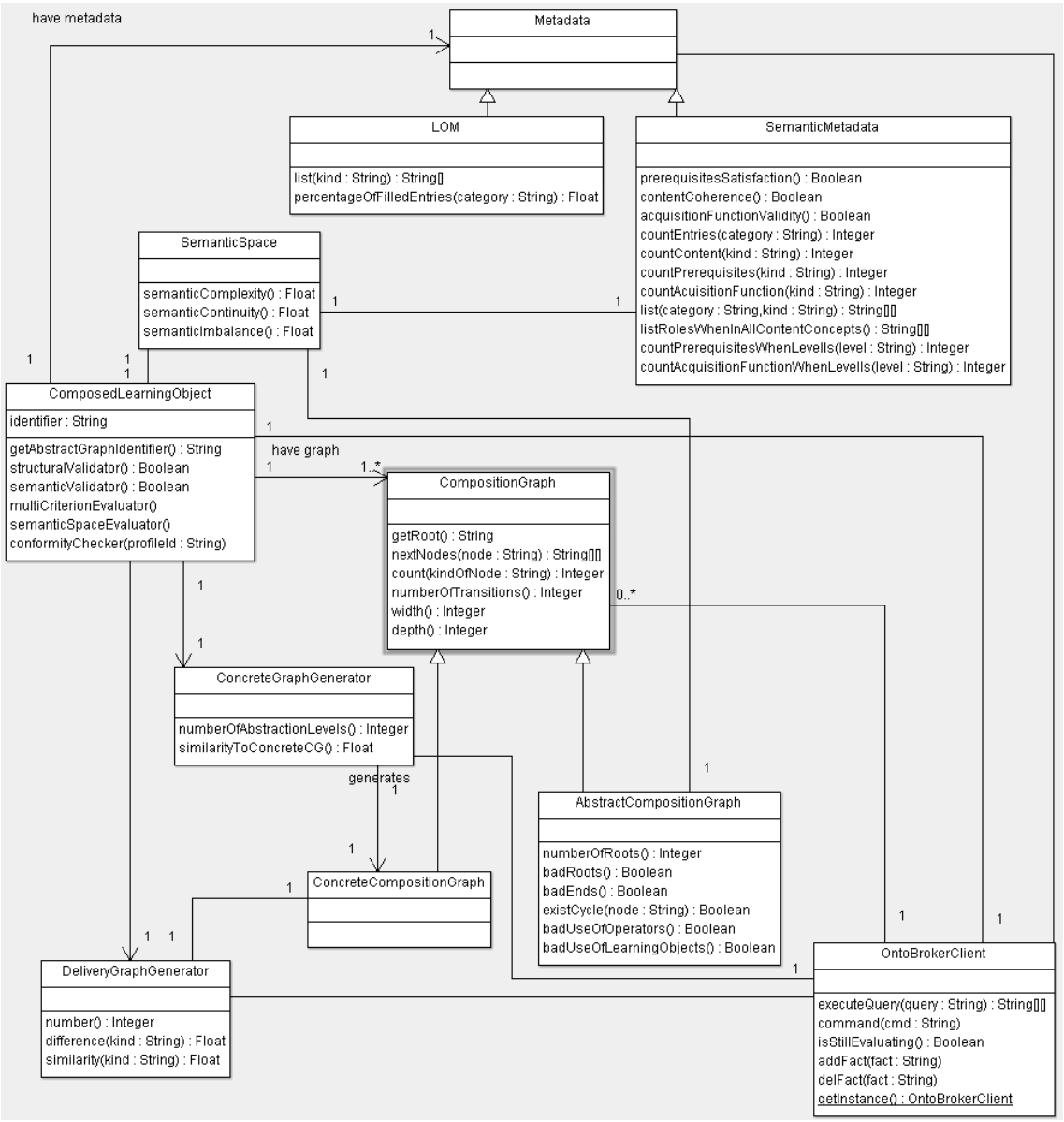


Figure 48 : Diagramme de classe du prototype implémenté

6. Vérification de la cohérence

6.1. Expérimentation

Pour expérimenter le module de vérification de la cohérence (sémantique et structurelle) nous avons défini un ensemble d'objets d'apprentissage chacun avec une anomalie spécifique parmi les anomalies structurelles et sémantiques possibles. Nous avons soumis ces différents objets à notre prototype afin de vérifier la capacité des algorithmes que nous avons défini à détecter les incohérences.

6.2. Test et résultats

Concernant les anomalies structurelles nous avons soumis les objets suivants à la vérification automatique de la cohérence structurelle :

LO_902 : Objet d'apprentissage sans anomalies.

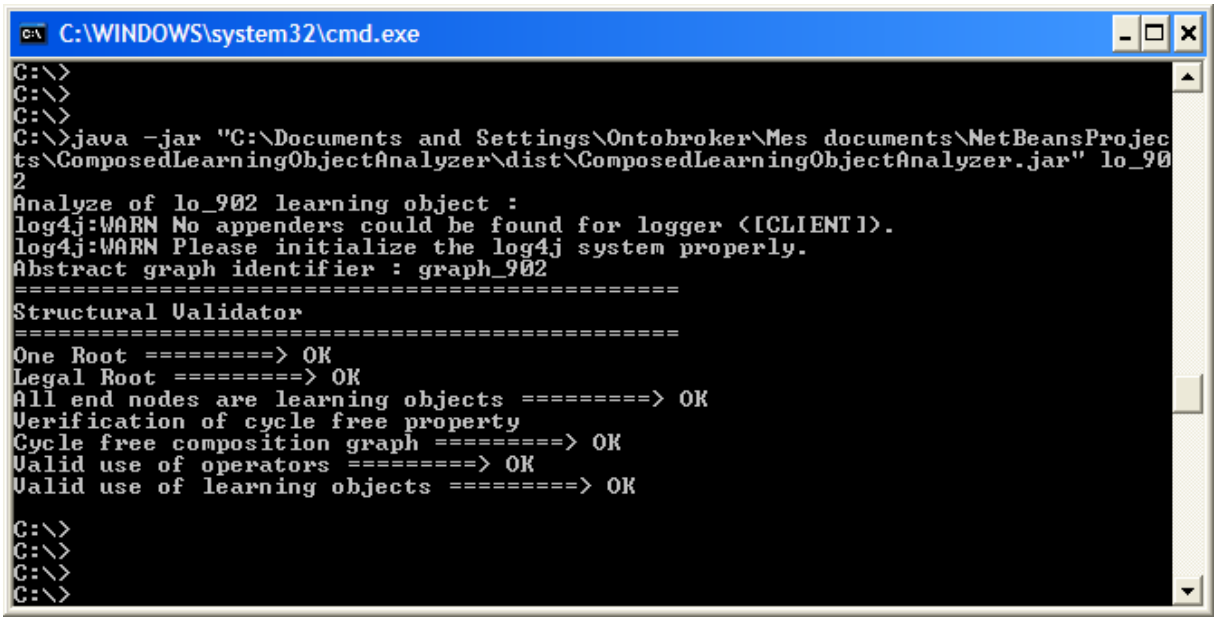


Figure 49 : Vérification de la conformité structurelle de l'objet LO_902

LO_1001 : Il comporte deux composantes connexes.

LO_1002 : Il comporte un sommet de type objet d'apprentissage qui est à l'origine de deux arcs.

LO_1003 : Il comporte un sommet feuille dans son graphe de composition abstrait qui est de type opérateur.

LO_1004 : Il comporte plusieurs sommets racines.

LO_1005 : Il comporte un sommet racine qui est de type opérateur.

LO_1006 : Il comporte un sommet de type opérateur qui est à l'origine d'un seul arc.

LO_1007 : Il comporte un cycle au sein du graphe de composition abstrait.

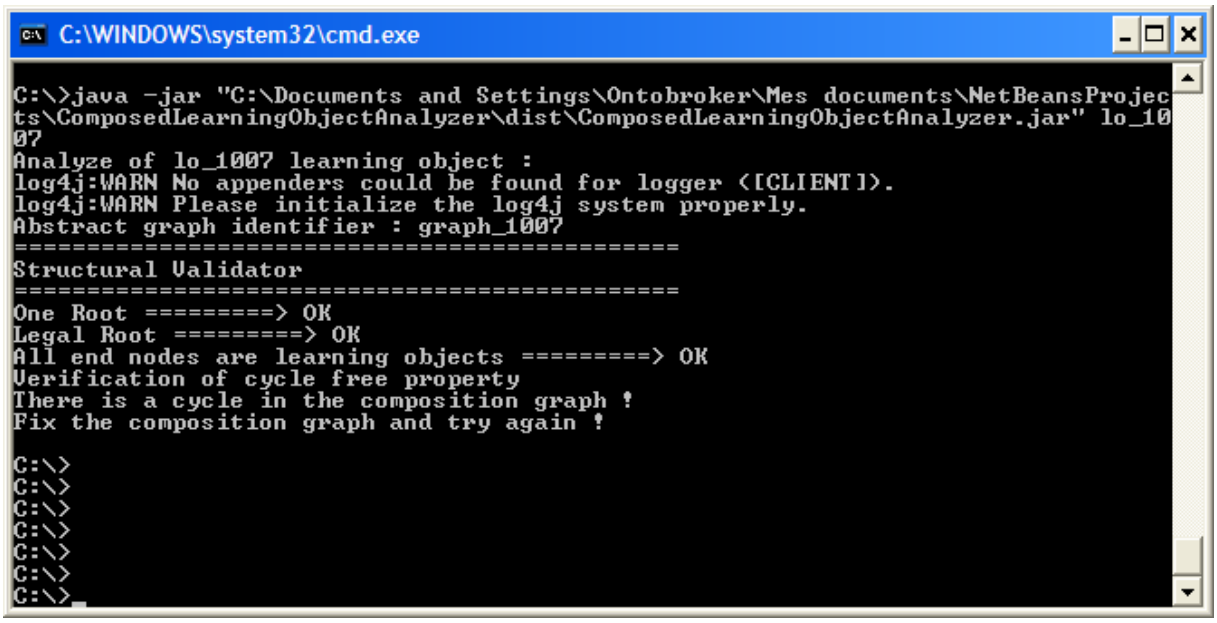


Figure 50 : Vérification de la conformité structurelle de l'objet LO_1007

Les différentes simulations ont montré que les algorithmes que nous avons définis permettent de détecter les différentes anomalies structurelles qui peuvent être présentes dans un graphe de composition abstrait.

Pour les aspects sémantiques nous avons préparé deux objets d'apprentissage avec des anomalies sémantiques :

LO_1011 : Un objet d'apprentissage au sein du graphe de composition abstrait comporte un pré-requis non satisfait ni par les objets qui le précédent ni par les pré-requis de l'objet composé.

LO_1012 : Le graphe de composition abstrait comporte un objet d'apprentissage dont le contenu comporte un concept qui n'est pas parmi les concepts du contenu de l'objet composé.

Ces deux anomalies sémantiques ont été également détectées par l'analyseur. Ce qui permet de valider les algorithmes qui ont été défini pour assurer cet objectif.

7. Assistance à la compréhension

7.1. Analyse générale

Afin de vérifier que la phase d'assistance à la compréhension permet d'avoir des indicateurs significatifs nous avons préparé un ensemble d'objets d'apprentissage composés ayant chacun des caractéristiques précises par rapport aux aspects suivant :

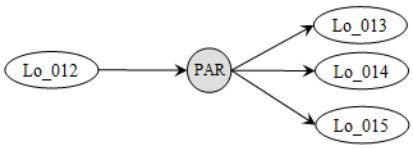
- Cohérence structurelle
- Capacité à être adapté
- Capacité à être réutilisé
- Consistance didactique

Ensuite, nous avons soumis ces objets d'apprentissage composés au processus d'analyse automatique. Nous avons fixé les coefficients utilisés dans les formules à la valeur un et les mesures sont ramenées à une échelle de zéro à dix.

7.2. Tests et résultats

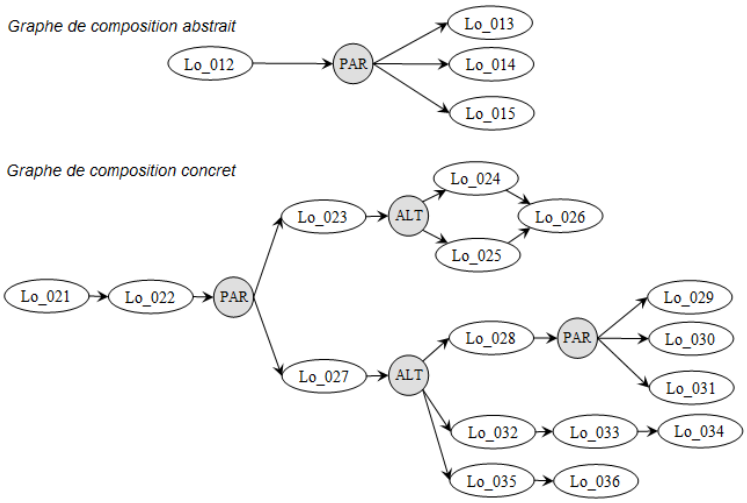
L'objectif des tests que nous avons mené sur l'aspect analyse générale consiste à vérifier que les résultats retournés correspondent bien à la vraie nature des objets d'apprentissage. Afin, d'effectuer cette évaluation nous avons préparé des objets d'apprentissage composés avec des caractéristiques particulières afin de vérifier si les valeurs estimatives calculées des indicateurs les caractérisent correctement. Voici la liste des objets expérimentés et leurs caractéristiques :

LO_920 : Concordance structurelle élevée.



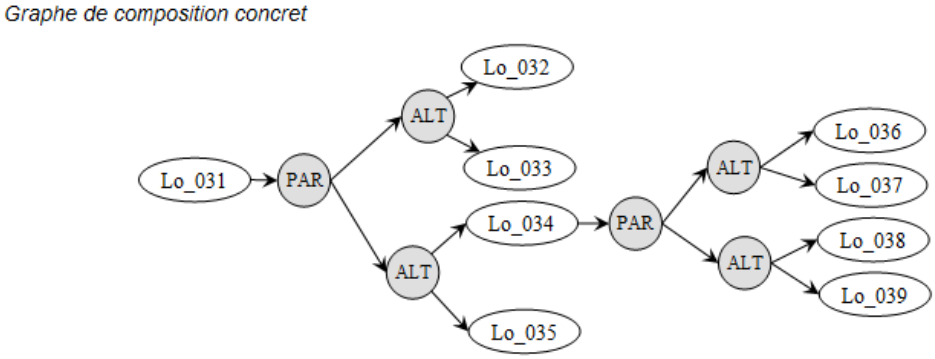
Cet objet est formé d'objets d'apprentissage simples. Ainsi, les graphes de composition abstrait et concret sont identiques.

LO_921 : Concordance structurelle faible.



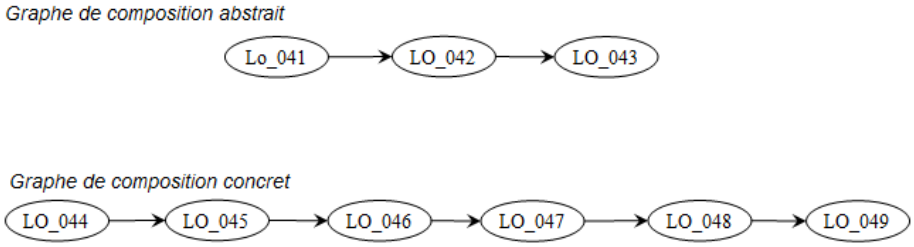
L'objet a été conçu de telle façon qu'il y en a une différence importante entre les graphes de composition abstrait et concret comme le montre la figure.

LO_922 : Capacité à être adapté élevée.



L'objet Lo_922 a le graphe de composition concret suivant qui produit dix graphes affichables différents.

LO_923 : Capacité à être adapté faible.



Le graphe de composition concret ne comporte aucun opérateur et donc la capacité d'adaptation et nulle.

LO_924 : Capacité à être réutilisé élevée.

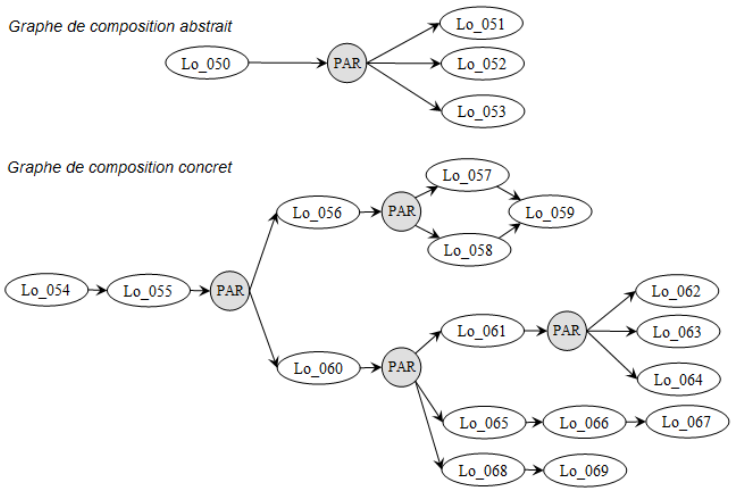
Cet objet est identique au niveau structurel à l'objet LO_921. Cependant son contenu traite un seul concept et ne demande aucun pré-requis. En plus, toutes les métadonnées sont renseignées.

LO_925 : Capacité à être réutilisé faible.

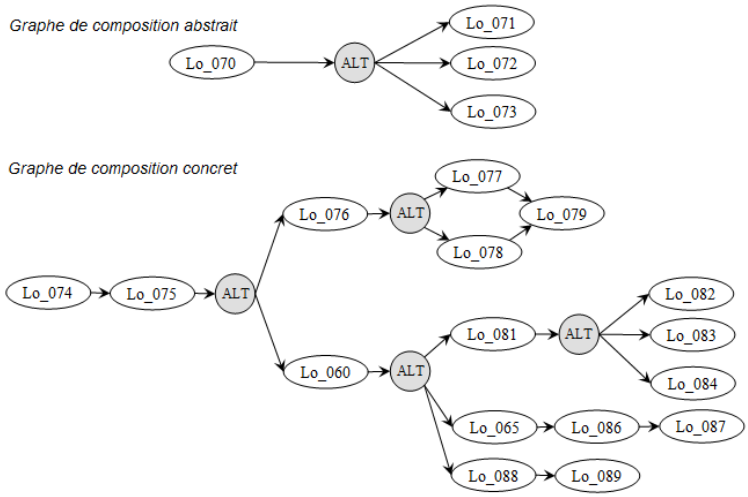
Cet objet a aussi les mêmes caractéristiques structurelles que l'objet LO_921. En revanche, son contenu et ses pré-requis portent sur dix concepts. La seule entrée de métadonnées renseignée est le titre.

LO_926 : Richesse didactique élevée.

Le graphe de composition concret de cet objet ne comporte pas d'opérateurs ALT et donc on a un seul graphe affichable. En plus, il comporte quatre rôles par concept et quatre sommets par rôle en moyenne.



LO_927 : Richesse didactique faible.



Cet objet a en moyenne un rôle par concept et quatre objets par rôle. En plus, seulement les deux premiers sommets sont partagés par tous les graphes affichables.

Si une valeur est entre 0 et 3,32 alors elle est considérée comme faible (F). Si elle est entre 3,33 et 6,65 alors elle est considérée comme moyenne (M). Finalement, si la valeur est entre 6,66 et 10 alors elle est considérée comme élevée (E).

Le résultat des tests est résumé dans le tableau ci-dessous. Chaque colonne comporte les mesures estimatives relatives à un objet d'apprentissage composé. Le titre de la colonne indique l'identifiant d'objet alors que chaque ligne indique l'aspect vérifié. La dernière ligne indique si les valeurs obtenues correspondent ou pas aux caractéristiques de chaque objet.

	LO_920	***LO_921***	***LO_922***	***LO_923***	***LO_924***	***LO_925***	***LO_926***	***LO_927***
Concordance structurelle	**E** (10)	**F** (3.16)	M (5,31)	F (1,83)	F (3.16)	F (3.16)	F (3.16)	F (3.16)
Capacité à être adapté	F (1,25)	M (5,25)	**E** (7,83)	**F** (0)	M (5,25)	M (5,25)	M (5,25)	M (5,25)
Capacité à être réutilisé	M (6,26)	M (6,26)	M (6,26)	M (6,26)	**E** (9,8)	**F** (2,19)	M (6,26)	M (6,26)
Richesse didactique	M (5)	E (7,5)	F (0,18)	M (5)	M (5,16)	M (5,16)	**E** (8,33)	**F** (2,8)
Correspondance	☑	☑	☑	☑	☑	☑	☑	☑

Comme on peut le constater l'évaluation automatique donne des estimations correctes concernant les caractéristiques des objets d'apprentissage. La précision des valeurs peut être améliorée selon le contexte, le domaine et l'avis des experts en réglant les coefficients de chaque formule d'évaluation.

7.3. Analyse spécifique de la sémantique du contenu

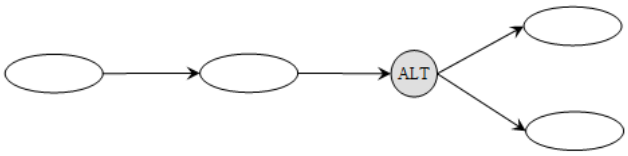
Afin d'expérimenter l'analyse spécifique de la sémantique du contenu nous avons procédé de la même façon que dans l'expérimentation de l'analyse multicritères. En effet, nous avons préparé des objets d'apprentissage avec des caractéristiques particulières par rapport aux trois propriétés objets de l'analyse : complexité sémantique, continuité sémantique et déséquilibre sémantique du contenu.

Par la suite nous avons soumis ces différents objets à l'analyse afin de vérifier que les valeurs des indicateurs calculés automatiquement sont bien ceux attendu. Nous avons fixé les valeurs des coefficients à un et les mesures sont ramenées à une échelle de zéro à dix.

7.4. Tests et résultats

Pour ces tests nous avons considérer des espaces sémantiques de taille 2. Pour calculer la complexité sémantique du contenu nous avons considéré que le seuil de complexité maximal pour les concepts (α) est égal à 10 et que le seuil de complexité maximal pour les composantes sémantiques (β) est également égal à 10.

Tous les objets utilisés dans cette simulation ont le même graphe de composition abstrait :



Voici la liste des objets expérimentés et leurs caractéristiques, sachant que nous avons utilisé l'ontologie de l'algorithmique (voir le chapitre V) :

LO_930 : Complexité sémantique élevée.

En fait, l'espace sémantique de cet objet comporte dix concepts répartis en dix composantes sémantiques. Ces valeurs sont celles des seuils maximaux de complexité α et β.

LO_931 : Complexité sémantique faible.

L'espace sémantique de cet objet contient un seul concept et donc on a une seule composante sémantique.

LO_932 : Continuité sémantique élevée.

L'espace sémantique de cet objet contient treize concepts et cinq composantes sémantiques. Toutefois, les concepts font partie d'un seul graphe connexe et on a sept intersections entre les composantes sémantiques.

LO_933 : Continuité sémantique faible.

L'espace sémantique de cet objet (qui contient vingt quatre concepts) est éclaté en trois graphes connexes et il n'y a qu'une seule intersection entre les quatre composantes sémantiques.

LO_934 : Déséquilibre sémantique élevée.

Le poids maximal des concepts est quatre alors que le poids minimal est un. Ces mêmes valeurs correspondent aux poids maximal et minimal des composantes sémantiques. Le

graphe de composition abstrait de l'objet comporte quatre sommets de type objet d'apprentissage.

LO_935 : Déséquilibre sémantique faible.

Les poids maximal et minimal des concepts et des composantes sémantiques est un. Le graphe de composition abstrait de l'objet comporte quatre sommets de type objet d'apprentissage.

Si une valeur est entre 0 et 3,32 alors elle est considérée comme faible (F). Si elle est entre 3,33 et 6,65 alors elle est considérée comme moyenne (M). Finalement, si la valeur est entre 6,66 et 10 alors elle est considérée comme élevée (E).

	LO_930	***LO_931***	***LO_932***	***LO_933***	***LO_934***	***LO_935***
Complexité	**E** (10)	**F** (0)	E (8,49)	E (8,01)	M (5,73)	F (3,01)
Continuité	F (0,5)	M (10)	**E** (10)	**F** (1,81)	M (3,33)	M (10)
Déséquilibre	M (4,3)	F (0)	M (4,3)	M (4,3)	**E** (8,61)	**F** (0)
Correspondance	☑	☑	☑	☑	☑	☑

Comme on peut le constater l'évaluation automatique donne des estimations correctes concernant les caractéristiques des objets d'apprentissage au niveau sémantique. La précision des valeurs peut être améliorée selon le contexte, le domaine et l'avis des experts en réglant les coefficients de chaque formule d'évaluation et en réglant les seuils α et β.

8. Vérification de la conformité

8.1. Expérimentation

Afin d'expérimenter la vérification automatique de la conformité d'un objet d'apprentissage composé à un ensemble de règles de conformité nous avons considéré les règles suivants définissant un patron de conception :

R_210 =

N1 : SLO N2 : ANY ;

LO -> Content -> count('Concept') ≤ 5 ;

LO -> AbstractCG -> abstractionLevels() ≤ *4* ;

LO -> ConcreteCG -> count(‘LO’) ≤ 20 ;

LO -> DeliveryGraph -> list(‘Role’) ⊇ {‘introduction’, ‘Exemple’, ‘Exercice’};

LO -> DeliveryGraph -> number() ≤ 5 ;

N1 -> Content -> list (‘Role’) ⊇ {‘Introduction’} ;

N2 -> Content -> list (‘Role’) ⊇ {‘Exercice’} ;

LO -> DeliveryGraph -> similarity(‘Beginning’) ≥ 2 ;

Ensuite, nous avons transformé ces règles en F-logique de la façon suivante :

FORALL LO,R,C,N

LO:R1Conforms<- LO[hasContent@(R)->>C] and count(LO,C,N) and lessorequal(N,5).

Tout objet d’apprentissage qui a un contenu qui traite le concept C selon le rôle R, le nombre de concepts dans le contenu de LO est N et N est inférieur ou égal à 5 alors c’est un objet conforme à la première règle de conformité.

Les autres règles fonctionnent selon le même principe :

FORALL LO, G, L, N

LO:R2Conforms<- LO[hasGraph->G] and p_g_levels(G,L) and count(G,L,N) and less(N,4).

FORALL LO, G, V, N

LO:R3Conforms<- LO[hasConcreteGraph->G) and p_cg_vertices(G,V) and V:AtomicLearningObject and count(G,V,N) and less(N,20).

FORALL LO, G

LO:R4Conforms<- LO[hasDeliveryGraph->>G] and p_dg_existsRole(G,”introduction”) and p_dg_existsRole(G,”exercise”) and p_dg_existsRole(G,”exemple”).

FORALL LO, G, D, N

LO:R5Conforms<- LO[hasGraph->G] and p_g_dg(G,D) and count(G,D,N) and less(N,5).

FORALL LO, R, C1, C2, V

LO:R6Conforms<- LO[hasContent@(“exercise”)->>C1] and LO[hasRoot ->V] and V[hasContent@(“introduction”) ->> C2].

FORALL LO, R, G, N

LO:R7Conforms<- LO[hasDGrah->G] and p_dg_beginSimilarity(G,N) and lessorequal(2,N).

Pour le moment nous n'avons pas proposé un module automatique pour transformer des règles en langage formel vers des règles F-logique. Nous nous somme plutôt intéressé à la faisabilité de la vérification automatique de la conformité. La définition des prédicats de haut-niveau que nous avons utilisé pour implémenter les règles de conformité sont données au niveau de l'annexe B.

8.2. Tests et résultats

Afin de vérifier la capacité du système à détecter la conformité ou la non-conformité à un patron de conception, nous avons défini un ensemble d'objets d'apprentissage dont chacun viole l'une de ses règles :

LO_1021 : Objet qui traite sept concepts au niveau du contenu et donc qui viole la première règle de conformité.

LO_1022 : Objet qui comporte cinq niveaux d'abstraction et donc qui viole la deuxième règle de conformité.

LO_1023 : Objet qui comporte vingt trois objets d'apprentissage au niveau du graphe de composition concret et donc qui viole la troisième règle de conformité.

LO_1024 : Objet qui comporte un graphe affichable qui ne comporte pas le rôle « Exemple » et donc qui viole la quatrième règle de conformité.

LO_1025 : Objet qui comporte six graphes affichables et donc qui viole la cinquième règle de conformité.

LO_1026 : Objet avec un graphe de composition dont le sommet racine est un objet d'apprentissage composé et donc qui viole la sixième règle de conformité.

LO_1027 : Objet dont les graphes affichables ne partagent que le premier sommet et donc qui viole la septième règle de conformité.

La soumission de ces objets à notre système a permis de vérifier que les algorithmes qui ont été définis permettent effectivement de vérifier la conformité d'un objet d'apprentissage composé à un ensemble de règles de conformité.

9. Conclusion

Au niveau de ce chapitre nous avons étudié la possibilité d'utiliser F-logique comme support de modélisation et l'OntoBroker comme moteur d'inférence afin de vérifier notre approche d'assistance aux auteurs d'objets d'apprentissage composés.

Comme approche de test nous avons effectué des simulations avec à chaque fois un ensemble d'objets d'apprentissage afin de vérifier le comportement du système. Les résultats ont permis de confirmer que le système retourne les résultats attendu.

Nous avons également présenté dans ce chapitre le système COLORS qui est une banque d'objet d'apprentissage. Nous avons montré via ce système la possibilité d'intégrer certains éléments de notre approche dans des systèmes outre les environnements de conception d'objets d'apprentissage. Pour le moment nous avons réussi d'intégrer dans COLORS certains éléments relatifs à l'analyse comparative des métadonnées informationnelles. Ceci s'explique par le fait que pour le moment les formats utilisés pour coder les objets d'apprentissage ne comportent pas une structure complexe et pas de métadonnées sémantiques.