ANALYSE DE LA CONFORMITE

1. Introduction

La phase d'analyse de la conformité a pour objectif de vérifier si un objet d'apprentissage est conforme à un patron de conception reflétant un ensemble de règles à respecter. Cela complète les étapes d'analyse par des moyens de contrôle plus forts et qui peuvent le cas échéant être appliqués non pas a postériori sur les objets déjà produits mais a priori sur les objets en cours de construction. Ainsi, il est important d'intégrer dans notre approche la possibilité d'assister l'auteur pour qu'il puisse vérifier la conformité de son objet soit par rapport à une certaine vision de la qualité, soit par rapport à des principes d'harmonisation édictés par des autorités relatives au domaine de l'apprentissage en ligne.

Le chapitre est organisé comme suit. Nous commençons par donner les principes de notre approche par règles de conformité puis nous présentons le langage tout d'abord informellement par des exemples puis de manière formelle. Une notation graphique permettant aux experts / auteurs de construire ou comprendre des règles est ensuite présentée. Les principes de vérification sont ensuite posés (sachant qu'ils seront détaillés dans le chapitre suivant) et enfin un exemple illustratif permet de montrer tous les langages et notations proposés ici.

2. Principes de la vérification de la conformité

2.1. Introduction

En fait, cette phase de notre approche permet de vérifier la conformité d'un objet d'apprentissage composé par rapport à un patron de conception (i.e. un ensemble de règles). Il faut noter que cette phase se distingue des autres phases par le fait qu'elle ne remet pas forcement en question l'effort de composition d'un objet d'apprentissage composé. En effet, les experts / administrateurs peuvent éventuellement avoir définis plusieurs ensembles de règles de conformité et alors un objet n'a pas à être conforme à tous ces ensembles mais seulement à l'un (au moins) d'entre eux.

Par exemple, supposant qu'un premier ensemble de règles reflète une pédagogie d'apprentissage par compétence et qu'un deuxième ensemble de règles reflète une pédagogie par l'exemple. Si un objet donné est soumis pour une vérification de conformité par rapport au premier ensemble et qu'il n'est pas conforme alors l'auteur peut en déduire qu'il n'est pas destiné à un apprentissage par compétence. En même, temps ce même objet peut être conforme au deuxième ensemble de règles et donc être adapté à un apprentissage par l'exemple. Les résultats négatifs d'une analyse de conformité peuvent faire découvrir à l'auteur les éléments à modifier pour devenir conforme (dans l'exemple ci-dessus comment adapter l'objet à un apprentissage par compétences).

La figure ci-dessus résume le processus de vérification de la conformité. Des experts proposent des règles de conformités exprimées via une notation graphique. Celle-ci permet à la fois de simplifier la définition des règles mais aussi la communication des règles aux auteurs. A la notation graphique correspond un langage formel qui permet la traduction des règles dans le langage F-logique. Ainsi, l'auteur peut soumettre son objet d'apprentissage pour une vérification automatique de conformité.

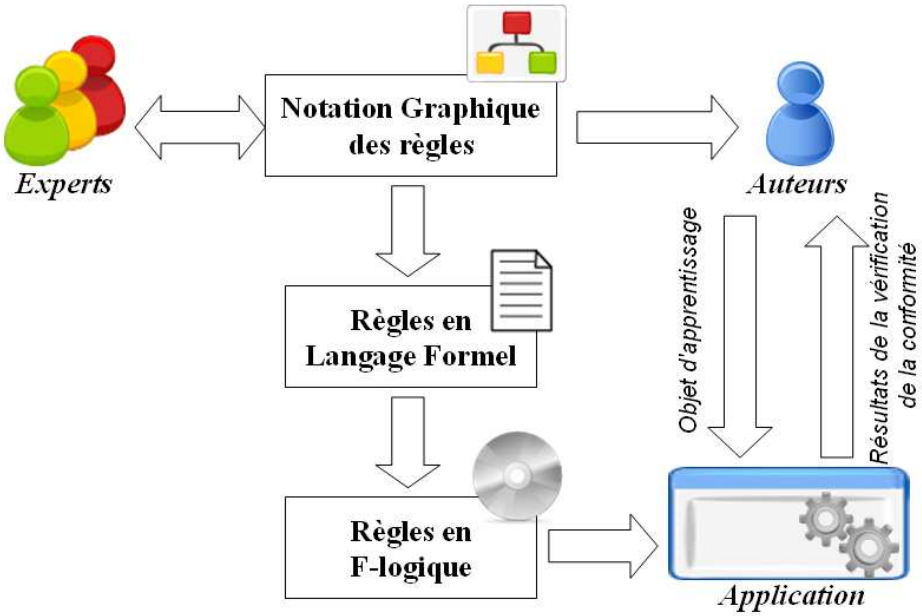

Figure 35 : Différents formats de description des règles de conformité

2.2. Règles de conformité

Les règles de conformité peuvent être définies par un ou plusieurs intervenants. Par exemple, il se peut que certaines règles soient définies au niveau du rectorat, d'autres au niveau de

l'institut d'autres au niveau du département et d'autres par l'auteur lui-même. Il se peut aussi que toutes les règles soient définies par une seule autorité.

Les règles qui peuvent être définies par rapport à un objet d'apprentissage font référence à la description que l'on peut observer. Ainsi, ces règles se basent sur les métriques introduites au niveau de la phase d'analyse et plus généralement sur les métadonnées des objets ainsi que leurs graphes de composition. Rappelons que nous avons classé ces règles en quatre catégories : les règles d'annotation, les règles structurelles, les règles sémantiques et les règles hybrides.

Afin de décrire les règles de conformité il faut définir un langage formel suffisamment expressif pour pouvoir décrire les quatre catégories de règles et flexible pour pouvoir s'adapter aux critères de qualité des experts et auteurs. En effet, on est loin d'avoir une seule vision de la qualité pour les objets d'apprentissage. Dans son rapport intitulé « la qualité en e-learning » [Ehlers et al., 2006] le centre européen pour le développement de la formation professionnelle affirme que 35% des organisations interrogées sur la qualité en e-learning utilisent leurs propres approches alors que 26% utilisent des approches externes. En fait, cette étude montre que 24% des interrogés travaillent dans des organisations qui laissent à leurs collaborateurs le soin de développer la qualité.

2.3. Notation graphique pour les règles

Il est important de prévoir un moyen permettant de définir (pour les experts) et de présenter (pour les auteurs) de manière compréhensible les règles de conformité. La compréhension de ces règles par les auteurs aura comme conséquence qu'il les prendra mieux en compte lors des phases de sélection d'objets à réutiliser et de construction du graphe de composition abstrait. Ceci évitera un nombre élevé d'itérations avant de mettre en place un objet d'apprentissage conforme.

Proposer une notation graphique représente un moyen approprié pour répondre à ce besoin. Cette notation doit offrir à la fois deux propriétés nécessaires pour communiquer clairement les règles même pour les non spécialistes. La première est qu'elle doit représenter les règles d'une façon organisée selon leurs natures : règles hybrides, règles sémantiques, règles structurelles et règles sur les métadonnées. La deuxième est qu'elle doit présenter les règles dans un format succinct et précis compréhensible par les auteurs.

2.4. Vérification automatique des règles

Faire vérifier manuellement la conformité d'un objet d'apprentissage composé par rapport aux règles de conformité présente plusieurs inconvénients. En effet, d'une part nous avons le risque de se tromper ou d'oublier la vérification d'une ou de plusieurs règles et d'autre part c'est une tâche fastidieuse et consommatrice de temps.

Ceci justifie le recours à l'automatisation de cette phase de notre approche afin d'assister les auteurs dans la vérification de leurs objets aux règles de conformité. Pour y parvenir nous avons fait appel aux techniques de la programmation logique afin de pouvoir décider si un objet est en conformité ou pas. Ainsi, les propriétés de l'objet d'apprentissage composé doivent être décrites par des faits et les règles de conformité par des règles logiques. Ensuite, le moteur d'inférence peut décider automatiquement si l'objet est conforme à toutes les règles ou non.

3. Règles de conformité

3.1. Règles pour les métadonnées

Les règles portant sur des métadonnées éducatives (ou règles d'annotation) permettent de vérifier la conformité à un profil d'application LOM. Les vérifications permettent de s'assurer que certaines entrées ou catégories obligatoires sont renseignées et qu'il y en a assez de métadonnées éducatives pour que l'objet soit accessible aux utilisateurs. Ces règles ont été classifiées en trois familles qui peuvent être utilisées séparément ou sous forme d'une combinaison.

3.1.1. Catégories LOM obligatoires

Les métadonnées, comme décrites dans le standard LOM [IEEE, 2002], sont organisées en catégories. En effet, dans le cas de ce standard les éléments à renseigner sont organisés en neuf catégories. Une catégorie est un groupe d'éléments en relation [IEEE, 2002].

Dans un profil d'application LOM on peut procéder par catégorie pour définir ce qui est obligatoire ou pas. Voici un exemple de règles pour imposer le renseignement de certaines catégories de métadonnées :

RM_1 : *La catégorie « General » est obligatoire.*

RM_2 : *La catégorie « Technical » est obligatoire.*

3.1.2. Eléments LOM obligatoires

Les profils d'application LOM les plus connus sont définis élément par élément. En effet, pour chaque élément LOM on indique s'il est obligatoire ou optionnel. Pour vérifier la conformité d'un objet d'apprentissage à un profil d'application défini de cette façon il faut définir une règle par élément obligatoire. Voici un exemple :

RM_3 : *L'élément « Title » est obligatoire.*

3.1.3. Règles quantitatives

La troisième famille de règles sur les métadonnées traite l'aspect quantitatif. Il ne s'agit plus de vérifier la conformité à un profil d'application LOM mais de s'assurer que l'auteur a renseigné assez de métadonnées éducatives pour que l'objet d'apprentissage soit accessible et qu'il puisse être découvert suite à une requête de la part de l'utilisateur. Voici deux exemples de règles, dont l'une a comme portée une catégorie LOM alors que l'autre a comme portée toutes les métadonnées :

RM_4 : *Au moins 50% des éléments de la catégorie « Educational » doivent être renseignés.*

RM_5 : *Au moins 20% des éléments de métadonnées éducatives doivent être renseignés.*

3.2. Règles structurelles

3.2.1. Forme du graphe

Au niveau structurel chaque objet d'apprentissage composé est décrit par un graphe de composition abstrait. Ce graphe est directement conçu par l'auteur de l'objet et sa forme reflète une stratégie de navigation au sein du contenu éducatif.

Une stratégie de navigation peut favoriser les accès alternatifs, les accès parallèles aux ressources. Elle permet également à l'auteur de définir un ordre de navigation permettant d'indiquer quelle ressource doit être étudiée en premier afin d'assimiler au mieux les suivantes.

Ainsi, d'un point de vue métier un ensemble de règles peuvent être proposées à ce niveau. En fait, pour augmenter le degré de personnalisation au niveau d'un objet d'apprentissage composé il faut s'assurer que l'auteur a conçu le graphe de composition avec assez d'opérateurs d'alternatives (ALT). D'autre part, pour permettre à l'apprenant d'avoir une certaine liberté dans le choix du chemin de navigation il faut s'assurer que l'auteur a introduit assez d'opérateurs parallèles (PAR) au sein du graphe.

Voici un exemple de règles structurelles qui permettent d'influencer la stratégie de navigation :

RS_1 : *Au moins deux opérateurs ALT doivent être présent au sein du graphe de composition abstrait.*

RS_2 : *Au moins un opérateur PAR doit être présent au sein du graphe de composition abstrait.*

Ces différentes règles peuvent être également décrites pour le graphe de composition concret. Rappelons que celui-ci est obtenu suite au remplacement de chaque objet d'apprentissage composé au sein du graphe de composition par son graphe de composition abstrait de façon récursive. Ce graphe ne comporte que des objets d'apprentissage non composés (ou atomiques). La stratégie de navigation à ce niveau est une combinaison de la stratégie décrite par l'auteur au niveau du graphe de composition abstrait et des stratégies de navigation des auteurs des objets d'apprentissage composés réutilisés. L'usage de règles à ce niveau est justifié par le fait de vouloir avoir un certain contrôle sur les propriétés du graphe de composition concret qui présente une facette de l'objet d'apprentissage non directement visible pour l'auteur. A titre d'exemples, voici quelques règles illustratives :

RS_3 : *Au plus cinq opérateurs ALT doivent être présent au sein du graphe de composition concret.*

RS_4 : *Au plus cinq opérateurs PAR doivent être présent au sein du graphe de composition concret.*

Le graphe concret peut se décomposer en un ensemble de graphes affichables qui peuvent être empruntés par les apprenants qui vont accéder à cet objet d'apprentissage. Certaines portions sont communes à tous les graphes affichables d'un objet d'apprentissage composé alors que d'autres parties sont spécifiques et s'adaptent mieux à certains profils d'apprenants. Le nombre de graphes affichables représente un aspect significatif d'un point de vue métier qui peut être contrôlé via des règles. Par exemple :

RS_5 : *Au moins trois graphes affichables par objet d'apprentissage composé.*

RS_6 : *Au plus six graphes affichables par objet d'apprentissage composé.*

3.2.2. Dimensions et densité du graphe de composition

Les règles structurelles peuvent également porter sur la taille du graphe permettant ainsi de contrôler ses dimensions ainsi que sa densité. Ceci permet de traduire certaines règles métier

qui concernent le volume de contenu dispensé aux apprenants via un objet d'apprentissage composé.

La taille du graphe est exprimée en termes de largeur et de profondeur. La profondeur d'un graphe correspond au plus long chemin. La largeur du graphe correspond à la plus grande distance entre deux sommets. Les distances sont mesurées en nombre d'arcs. Des règles portant sur les dimensions d'un graphe de composition peuvent être formulées par exemple comme suit :

RS_7 : *la profondeur du graphe de composition abstrait ne doit pas dépasser vingt.*

RS_8 : *la largeur du graphe de composition concret ne doit pas dépasser cinq.*

La densité du graphe en terme de nombre de sommets de type objets d'apprentissage est une propriété supplémentaire qui peut être contrôlée au niveau des différents graphes associés à un objet d'apprentissage composé. Des règles comme les suivantes peuvent être formulées :

RS_9 : *Le nombre de sommets de type objet d'apprentissage ne doit pas dépasser 50 au niveau du graphe de composition concret.*

RS_10 : *Le nombre de sommets de type objet d'apprentissage au niveau de chaque graphe affichable ne doit pas dépasser vingt.*

Les graphes affichables représentent pour les apprenants la partie visible de l'objet d'apprentissage composé. Ceci justifie le fait de vouloir s'assurer que certaines règles ont été respectées. A ce niveau, les règles peuvent imposer une certaines similarité entre les graphes affichables. A titre d'exemples voici deux règles illustratives :

RS_11 : *La différence entre la plus petite profondeur et la plus grande profondeur des différents graphes affichables ne doit pas dépasser 30%.*

RS_12 : *La différence de densité en sommets entre le graphe affichable le plus dense et le moins dense ne doit pas dépasser 30%.*

3.2.3. Niveaux d'abstraction

Les règles peuvent porter sur des propriétés liées aux niveaux d'abstraction du graphe de composition d'un objet d'apprentissage composé. Le nombre de niveau d'abstraction peut être limité pour éviter le risque d'une discordance structurelle importante entre le graphe de composition abstrait et le graphe de composition concret qui peut influencer de façon négative la stratégie de navigation choisie par l'auteur. La règle suivante permet d'exprimer ceci :

RS_13 : *Le nombre de niveaux d'abstraction ne doit pas dépasser quatre.*

Certaines règles peuvent limiter les aspects en relation avec la discordance structurelle. En effet, il est possible de cibler des propriétés précises. Les différences en termes de nombre de sommets, d'opérateurs PAR et d'opérateurs ALT entre le graphe de composition abstrait et le graphe de composition concret peuvent être ainsi contrôlées. Les différences entre ces deux graphes en termes de profondeur, de largeur et de densité peuvent également être contrôlées. Voici des exemples de règles :

RS_14 : *Le nombre de sommets du graphe de composition concret ne doit pas dépasser le double du nombre de sommets du graphe de composition abstrait.*

3.2.4. Similarité des graphes affichables

A un même objet d'apprentissage composé est associé un certain nombre de graphes affichables qui correspondent aux différents parcours possibles du graphe de composition concret. Ces graphes affichables sont en quelque sorte des versions différentes d'un même objet. Ainsi, ils doivent présenter une certaine similarité. Cette similarité peut être contrôlée via des règles spécifiques. Ces règles doivent spécifier le degré de similarité en donnant son étendue minimum. Voici quelques exemples de règles :

RS_15 : *Tous les graphes affichables d'un objet d'apprentissage composé doivent avoir en commun au moins les trois premiers sommets.*

RS_16 : *Tous les graphes affichables doivent avoir en commun au moins 50% des leurs sommets (sommets du graphe de composition concret sans les opérateurs ALT).*

3.3. Règles sémantiques

Concernant les aspects sémantiques les règles peuvent porter sur le contenu, les pré-requis et la fonction d'acquisition. Au niveau du contenu et pour des considérations pédagogiques il est intéressant de pouvoir s'assurer que le contenu n'est pas trop chargé en concepts, que certains rôles éducatifs sont présents et éventuellement que certains rôles éducatifs sont assurés pour chaque concept traité. Quant aux pré-requis, il est généralement souhaitable qu'ils ne soient pas trop nombreux ni trop restrictifs. En effet, ceci peut rendre l'objet inaccessible à la plupart des apprenants et donc aller à l'encontre de son rôle éducatif. Finalement, la fonction d'acquisition ne doit pas être trop forte en offrant à l'apprenant un niveau élevé sur plusieurs concepts et plusieurs rôles simultanément.

3.3.1. Contenu

Le contenu d'un objet d'apprentissage est décrit par un ensemble de couples <concept, rôle>. A ce niveau le nombre de concepts différents qui sont traités peut être contrôlé afin d'éviter un contenu trop chargé. Ceci doit permettre de favoriser une meilleure assimilation par les apprenants. Ceci peut être exprimé via des règles semblables à celle-ci :

RSe_1 : *Le nombre de concepts traités au niveau du contenu ne doit pas dépasser cinq.*

Les rôles éducatifs assurés par l'objet d'apprentissage composé peuvent être également contrôlés pour s'assurer par exemple de la présence de certains rôles spécifiques. Voici un exemple de règles de ce type :

RSe_2 : *Les rôles « exercice » et « exemple » doivent être présents au niveau du contenu.*

Il est également envisageable de vouloir s'assurer de la présence d'un ou de plusieurs rôles pour chaque concept traité au niveau de l'objet d'apprentissage composé. Voici une règle de ce type à titre d'exemple :

RSe_3 : *Le rôle « définition » doit être assuré pour chaque concept dans le contenu.*

Il est possible également de procéder d'une façon plus générale en définissant une règle limitant le nombre de doublets qui décrivent le contenu. Voici un exemple :

RSe_4 : *le nombre d'entrées qui décrivent le contenu ne doivent pas dépasser dix.*

3.3.2. Pré-requis

Pour les pré-requis ils sont décrit par des triplets de la forme <concept, rôle, niveau>. S'assurer que les pré-requis ne sont pas trop restrictifs peut être contrôlé en appliquant certaines règles.

Le nombre de concept traités au niveau des pré-requis peut faire le sujet d'une règle permettant de s'assurer qu'il ne dépasse pas un certain seuil. Par exemple, voici un exemple d'une règle qui put être formulée afin d'assurer cette propriété :

RSe_5 : *Le nombre de concepts différents demandés au niveau des pré-requis ne doit pas dépasser cinq.*

Le nombre de rôles différents demandés au niveau des pré-requis peut également être contrôlé pour s'assurer que l'objet reste accessible aux apprenants. Il est ainsi possible de limiter le nombre de rôles différents pour mettre un seuil supérieur. Voici un exemple de formulation d'une telle règle :

RSe_6 : *Le nombre de rôles éducatifs différents demandés au niveau des pré-requis ne doit pas dépasser trois.*

Le niveau de maîtrise d'un concept suivant un certain rôle, décrit par un triplet au niveau des pré-requis, permet d'indiquer si l'objet sera accessible à un nombre restreint ou important d'apprenants. En effet, si le niveau de maîtrise demandé est élevé alors moins d'apprenants seront en mesure d'accéder à l'objet d'apprentissage. Certaines règles peuvent contrôler cet aspect comme la règle suivante :

RSe_7 : *Le nombre de pré-requis avec un niveau de maîtrise « élevé » ne doit pas dépasser trois.*

Le nombre de pré-requis de façon générale peut également être contrôlé via une règle qui limite le nombre d'entrées. Voici un exemple :

RSe_8 : *Le nombre d'entrées qui décrivent les pré-requis nécessaires à l'apprenant ne doit pas dépasser cinq.*

3.3.3. Fonction d'acquisition

La fonction d'acquisition est un ensemble de triplets <concept, rôle, niveau> qui représente ce qu'un apprenant a acquis grâce au contenu d'un objet d'apprentissage. Cette information permet de mettre-à-jour le profil de l'apprenant.

Ce qui peut être contrôlé à ce niveau c'est principalement l'aspect quantitatif. En fait, proposer trop de connaissances à la fois à l'apprenant est souvent considéré comme étant à l'encontre d'une meilleure assimilation des concepts traités par le contenu. Ceci peut être contrôlé via une règle de la forme suivante :

RSe_9 : *Le nombre de triplets dans la fonction d'acquisition ne doit pas dépasser cinq.*

Le contrôlé peut être raffiné en se focalisant sur le nombre de triplets dont le niveau de connaissance est « élevé ». Par exemple, voici une règle qui permet d'appliquer ce type de contrôle :

RSe_10 : *Le nombre de triplets dans la fonction d'acquisition ayant comme niveau de connaissance « élevé » ne doit pas dépasser trois.*

3.4. Règles hybrides

Les règles hybrides représentent une façon plus expressive pour contrôler des aspects qui mettent en jeux la sémantique et la structure simultanément. Ainsi, des règles didactiques plus

élaborées que celles purement structurelles ou purement sémantiques peuvent être exprimées et contrôlées.

3.4.1. Rôles du sommet racine et des sommets feuilles

Le sommet racine et les sommets feuilles ont une importance particulière. En effet, le sommet racine est le premier objet d'apprentissage qui sera offert aux apprenants. Concernant les sommets feuilles sont les derniers objets d'apprentissage qui sont offerts aux apprenants. Il est donc intéressant de vouloir leur imposer des rôles précis via des règles spécifiques. Voici deux règles à titre d'exemple :

RH_1 : *Le sommet racine doit avoir le rôle « introduction ».*

RH_2 : *Chaque sommet feuille doit avoir le rôle « exercice ».*

3.4.2. Patron de conception

Les règles de conception peuvent être très strictes en imposant un patron de conception commun aux objets d'apprentissage avec cependant une certaine marge de liberté. Un patron de conception représente une structure « flexible » du graphe de composition couplée avec des règles portant sur le contenu. Ceci permet d'avoir des objets d'apprentissage très homogènes et qui présente une structure et une sémantique homogènes. Voici deux patrons de conception possibles :

RH_3 : *L'objet doit commencer par un objet d'apprentissage simple avec le rôle « Introduction » suivi par un objet d'apprentissage avec le rôle « présentation ». Il doit être suivi par trois alternatives d'objets d'apprentissage ayant le rôle « Exemple ». Ensuite, il faut un accès parallèle à deux objets d'apprentissage avec le rôle « Exercice ». Puis, l'objet doit présenter un objet d'apprentissage avec le rôle « Conclusion ».*

RH_4 : *L'objet doit commencer par une séquence de trois objets d'apprentissage simples et se terminer par un objet d'apprentissage avec le rôle « Conclusion ».*

4. Langage formel pour la description des règles

4.1. Eléments du langage

Afin de pouvoir décrire les règles d'une façon formelle il faut définir un langage capable de supporter les différentes règles de conformité. Ainsi, nous avons défini un langage qui suit la grammaire décrite ci-dessous.

En fait, chaque ensemble de règles est décrit par un identificateur, éventuellement une spécification du graphe de composition abstrait et une ou plusieurs règles de conformité.

Rules := Identifier = [Graph_spefication ;] [Rule ;]$^{+}$

Le graphe est décrit par l'ensemble de ses arcs. Chaque arc est décrit par un couple de sommets. Le premier est le sommet de départ et le deuxième c'est le sommet d'arrivée.

*Graph_specification := Edge [, Edge]**
Edge := Node Node

Chaque sommet peut représenter l'un des éléments suivants : un objet d'apprentissage (quelconque, simple ou composé), un opérateur de graphe ou un sommet générique ANY qui représente tout graphe valide. Les opérateurs de graphe sont ALT pour les alternatives, PAR pour les accès parallèles et ANY_Op qui représente l'un ou l'autre des deux opérateurs.

Node := Id_Node : LO | SLO | CLO | Graph_Op | ANY
Graph_Op := ALT | PAR | ANY_Op

Les règles sont simples ou complexes. Chaque règle simple consiste en une fonction dont le résultat est comparé à une constante ou au résultat de l'évaluation d'une autre fonction. Une règle complexe est formée d'une quantification qui porte sur un ensemble généré par une fonction et qui doit vérifier soit une propriété décrite par un quantificateur qui porte sur un ensemble généré également par une fonction, soit une comparaison entre le résultat d'une fonction qui s'applique sur l'ensemble généré et une constante.

Rule := SimpleRule | ComplexRule
SimpleRule := LearningObject->Characteristic->Function Comparator [Function | Constant]
LerningObject := LO | Id_Node
Characteristic := Metadata | AbstractCG | ConcreteCG | DeliveryG | Prerequisite | Content | AcquisitionFunction
Comparator := = | ≠ | < | > | ≤ | ≥ | IN | NOT IN | ⊇ | ⊆
ComplexRule := Quantification Function => [Quantification Function | Function Comparator Constant
Quantification := ∀var IN | ∃var IN
Constant := string or numbers or booleans

4.2. Fonctions

Chaque fonction a un nom et d'éventuels paramètres donnés entre parenthèse. Chaque fonction retourne un résultat et s'applique à une caractéristique d'un objet d'apprentissage :

LearningObject -> Caracteristic -> FunctionName ([Parameter [, Parameter]])*

L'objet d'apprentissage est soit l'objet d'apprentissage composé sujet de la vérification de la conformité ou un objet d'apprentissage utilisé dans le graphe de composition abstrait. Dans le premier cas il faut noter l'objet par LO (représente l'objet en cours de validation) et dans le deuxième cas il faut donner l'identifiant du sommet qui représente l'objet au sein du graphe.

La liste des caractéristiques sur lesquelles portent les règles est la suivante :

« ***Metadata*** » : désigne l'ensemble des métadonnées informationnelles de l'objet d'apprentissage.

« ***AbstractCG*** » : désigne le graphe de composition abstrait de l'objet d'apprentissage.

« ***ConcreteCG*** » : désigne le graphe de composition concret de l'objet d'apprentissage.

« ***DeliveryG*** » : désigne l'ensemble de graphes affichables de l'objet d'apprentissage.

« ***Content*** » : désigne l'ensemble des doublets <contenu, rôle > qui donnent une description sémantique du contenu de l'objet d'apprentissage.

« ***Prerequisites*** » : désigne l'ensemble des triplets <contenu, rôle, niveau> qui donnent une description sémantique des pré-requis nécessaires pour aborder cet objet d'apprentissage.

« ***AcquisitionFunction*** » : désigne l'ensemble des triplets <contenu, rôle, niveau> qui donnent une description sémantique des acquis apportés au profil de l'apprenant par cet objet d'apprentissage.

Nous allons introduire par la suite les différentes fonctions ainsi que des exemples de leur utilisation pour chaque famille de règles.

4.2.1. Fonctions pour les règles de métadonnées

Nous allons utiliser la fonction ***list(<paramètre>)*** avec la caractéristique « Metadata ». Elle peut prendre comme paramètre le terme « Categories » et alors elle génère l'ensemble des catégories de métadonnées renseignées au sein de l'objet. Sinon, elle prend comme paramètre le terme « Entries » afin de générer l'ensemble des entrées de métadonnées renseignées.

A titre d'exemples voici comment on note les règles présentées dans la section précédente :

RM_1 : *LO ->Metadata ->list('Categories') ⊇ {'General'}*

RM_2 : *LO -> Metadata -> list('Categories') ⊇ {'Technical'}*

RM_3 : *LO -> Metadata -> list('Entries') ⊇ {'Title'}*

Pour les règles quantitatives qui portent sur le pourcentage des entrées de métadonnées renseignées nous allons utiliser la fonction ***percentageOfFilledEntries(***<paramètre>*)* avec la caractéristique « Metadata ». Elle donne un pourcentage. S'il s'agit du pourcentage par rapport à tous les entrées, le paramètre à indiquer est « All ». Par contre s'il s'agit du pourcentage par rapport à tous les entrées d'une catégorie le paramètre est alors le nom de la catégorie de métadonnées.

RM_4 : *LO -> Metadata -> percentageOfFilledEntries('Educational') ≥ 50%*

RM_5 : *LO -> Metadata -> percentageOfFilledEntries('All') ≥ 20%*

4.2.2. Fonctions pour les règles structurelles

Les règles structurelles portent sur trois caractéristiques : le graphe de composition abstrait (***AbstractCG***), le graphe de composition concret (***ConcreteCG***) et les graphes de composition (***DeliveryG***).

Une première fonction qui s'applique à tous ces caractéristiques est la fonction ***count(***<*paramètre*>***)***. Elle permet de comptabiliser les opérateurs d'alternatives au sein du graphe si 'ALT' est donné comme paramètre. Elle permet de comptabiliser le nombre d'opérateurs parallèle si le paramètre donné est 'PAR'. Si le paramètre donné est 'LO' ou 'SLO' ou 'CLO' alors elle comptabilise respectivement le nombre d'objets d'apprentissage ou le nombre d'objet d'apprentissage simples ou le nombre d'objets d'apprentissage composés au sein du graphe. Finalement, le paramètre 'ALL' permet de comptabiliser le nombre de sommets au sein du graphe.

Grace à cette fonction les règles suivantes peuvent être exprimées :

RS_1 : *LO -> AbstractCG -> count('ALT') ≥ 2*

RS_2 : *LO -> AbstractCG -> count('PAR') ≥ 1*

RS_3 : *LO -> ConcreteCG -> count('ALT') ≤ 5*

RS_4 : *LO -> ConcreteCG -> count('PAR') ≤ 5*

RS_9 : *LO -> ConcreteCG -> count(‘LO’) ≤ 50*

RS_10 : *LO -> DeliveryG -> count(‘LO’) ≤ 20*

Deux fonctions sans paramètres ***width()*** et ***depth()*** permettent de retourner respectivement la largeur et la profondeur d’une des trois catégories de graphes associés à un objet d’apprentissage composé.

Grace à cette fonction les règles suivantes peuvent être exprimées :

RS_7 : *LO -> AbstractCG -> depth() ≤ 20*

RS_8 : *LO -> AbstractCG -> width() ≤ 5*

Les graphes affichables ont leurs propres fonctions. La fonction ***number()*** permet de donner le nombre de graphes affichables. Alors que la fonction ***difference(***<paramètre>***)*** permet de retourner le pourcentage de différence entre la valeur la plus petite et la plus grande entre les graphes affichables par rapport à l’un des paramètres suivants : ‘Depth’ pour la profondeur, ‘Width’ pour la largeur, ‘LO’ pour le nombre d’objets d’apprentissage, ‘PAR’ pour le nombre d’opérateurs parallèle et ‘ALL’ pour le nombre total de sommets.

La fonction ***similarity(***<paramètre>***)*** permet de retourner suivant le paramètre donné une évaluation concernant la similarité entre les graphes affichables. Si le paramètre donné est ‘Beginning’ la fonction retourne le nombre des sommets qu’on trouve au début de tous les graphes affichables. Si le paramètre donné est ‘ALL’ alors la fonction doit retourner le pourcentage de sommets partagés entre les graphes affichables par rapport à l’ensemble de leurs sommets. Ses sommets sont ceux du graphe de composition concret sans les opérateurs ALT.

Grace à ces fonctions il est possible d’exprimer les contraintes suivantes qui portent sur les graphes affichables :

RS_5 : *LO -> DeliveryGraph -> number()≥ 3*

RS_6 : *LO -> DeliveryGraph -> number()≤ 6*

RS_11 : *LO -> DeliveryGraph -> difference(‘Depth’) ≤ 30%*

RS_12 : *LO -> DeliveryGraph -> difference(‘ALL’) ≤ 30%*

RS_15 : *LO -> DeliveryGraph -> similarity(‘Beginning’) ≥ 3*

RS_16 : *LO -> DeliveryGraph -> similarity(‘ALL’) ≥ 50%*

Le graphe de composition abstrait dispose de deux fonctions spécifiques. La première est la fonction ***numberOfAbstractionLevels()***. Cette fonction donne le nombre de niveaux d'abstraction. La deuxième est la fonction ***similarityToConcreteCG()*** qui permet de donner le pourcentage de similarité entre le graphe de composition abstrait et le graphe de composition concret en termes d'objets d'apprentissage.

Ces deux fonctions permettent d'exprimer les règles suivantes :

RS_13 : *LO -> AbstractCG -> numberOfAbstractionLevels() ≤ 4*

RS_14 : *LO -> AbstractCG -> similarityToConcretteCG() ≤ 50%*

4.2.3. Fonctions pour les règles sémantiques

Les règles sémantiques portent sur trois caractéristiques : le contenu (***Content***), les pré-requis (***Prerequisites***) et la fonction d'acquisition (***AcquisitionFunction***).

Une première fonction qui s'applique à tous ces caractéristiques est la fonction ***number()*** qui permet de retourner le nombre d'entrées relatives à chacune des trois catégories de métadonnées sémantiques.

Cette fonction permet d'exprimer les règles suivantes :

RSe_4 : *LO -> Content -> number() ≤ 10*

RSe_8 : *LO -> Prerequisites -> number() ≤ 5*

RSe_9 : *LO -> AcquisitionFunction -> number()≤ 5*

Une deuxième fonction applicable à ces caractéristiques est la fonction ***count(***<paramètre>***)***. Si le paramètre est 'Concept' ou 'Role' alors la fonction retourne respectivement le nombre de concepts ou le nombre de rôles distincts au sein de la catégorie de métadonnées considérée.

RSe_1 : *LO -> Content -> count('Concept') ≤ 5*

RSe_5 : *LO -> Prerequisites -> count('Concept') ≤ 5*

RSe_6 : *LO -> Prerequisites -> count('Role') ≤ 3*

Nous proposons également la fonction ***list(****<paramètre>****)*** qui permet selon le paramètre de lister soit l'ensemble des concepts avec le paramètre 'Concept', soit l'ensemble des rôles avec le paramètre 'Role', soit l'ensemble de niveaux avec le paramètre 'Level'.

Cette fonction permet d'exprimer des règles de conformité comme la suivante :

RSe_2 : *LO -> Content -> list('Role') ⊇ {'Exercie', 'Exemple'}*

RSe_3 : *∀ x IN LO->Content->list('Concept') => ∃ <x, 'Définition'> IN LO->Content-> list()*

RSe_7 : *∀ <x, y, 'Elevé'> IN LO->Prerequisites->list() => count() ≤ 3*

RSe_10 : *∀ <x, y, 'Elevé'> IN LO->AcquisitionFunction-> list() =>count()≤3*

En fait les règles RSe_3, RSe_7 et RSe_10 sont des exemples de règles complexe où on fait appel aux quantificateurs.

4.2.4. Formulation des règles hybrides

Les règles hybrides associent à la fois des aspects structurels et sémantiques. Elles ont une forte capacité expressive. Voici comment les règles hybrides données comme exemple précédemment sont exprimées :

RH_1 : *Le sommet racine doit avoir le rôle « introduction ».*

R_001 =

N1 : LO N2 : ANY ;

…

N1 -> Content -> list (Role) ⊇ {'Introduction'},

…

RH_2 : *Chaque sommet feuille doit avoir le rôle « exercice ».*

R_002 =

N1 : ANY N2 : LO

…

N2 -> Content -> list (Role) ⊇ {'Conclusion'}

…

RH_3 : *L'objet doit commencer par objet d'apprentissage simple avec le rôle « Introduction » suivi par un objet d'apprentissage avec le rôle « présentation ». Il doit être suivi par trois alternatives d'objets d'apprentissage ayant le rôle « Exemple ». Ensuite, il faut un accès parallèle à deux objets d'apprentissage avec le rôle « Exercice ». Puis, l'objet doit présenter un objet d'apprentissage avec le rôle « Conclusion ».*

R_003 =

N1 : SLO N2 : LO,

N2 : LO N3 : ALT,

N3 : ALT N4 : LO,

N3 : ALT N5 : LO,

N3 : ALT N6 : LO,

N4 : LO N7 : PAR,

N5 : LO N7 : PAR,

N6 : LO N7 : PAR,

N7 : PAR N8 : LO,

N7 : PAR N9 : LO,

N8 : LO N10 : LO,

N9 : LO N10 : LO ;

N1 -> Content -> list (Role) ⊇ {'Introduction'},

N2 -> Content -> list (Role) ⊇ {'Présentation'},

N4 -> Content -> list (Role) ⊇ {'Exemple'},

N5 -> Content -> list (Role) ⊇ {'Exemple'},

N6 -> Content -> list (Role) ⊇ {'Exemple'},

N8 -> Content -> list (Role) ⊇ {'Exercice'},

N9 -> Content -> list (Role) ⊇ {'Exercice'},

N10 -> Content -> list (Role) ⊇ {'Conclusion'} ;

***RH_4 :** L'objet doit commencer par une séquence de trois objets d'apprentissage simples et se terminer par un objet d'apprentissage avec le rôle « Conclusion ».*

R_004 =

N1 : SLO N2 : SLO,

N2 : SLO N3 : SLO,

N3 : SLO N4 : ANY,

N4 : ANY N5 : LO ;

N5 -> Content -> list (Role) ⊇ {'Conclusion'} ;

4.3. Expressivité du langage

Nous avons étudié l'expressivité du langage par rapport à nos objectifs, et nous avons réussi à formuler toute les règles qui nous intéressent. Toutefois, il faut noter qu'il ne supporte pas des règles qui intègrent des contraintes conditionnelles.

Exemple :

« *Si le nombre d'objets d'apprentissage d'objets dépasse dix au niveau du graphe de composition abstrait alors le nombre d'opérateurs ALT au niveau de ce graphe ne doit dépasser trois.* »

Ceci ne présente pas une limite par rapport à notre contexte d'utilisation du moment qu'il est possible de définir plusieurs ensembles de contraintes. Chaque ensemble peut être conçu pour une catégorie particulière d'objets d'apprentissage. En effet, nous pouvons par exemple considérer que les objets d'apprentissage dont le graphe de composition comprend plus que dix objets sont des objets de grande taille. Et il est possible de définir un ensemble de règles pour les objets de grande taille qui spécifie que le nombre d'opérateurs ALT doit être supérieur à trois. Ce profil peut être formulé de la façon suivante :

R_005 =

…

LO -> AbstractCG -> count ('LO') >10,

LO -> AbstractCG -> count ('ALT') > 3,

…

Dans cet extrait de profil on peut constater qu'uniquement les objets d'apprentissage qui comportent plus que dix objets au niveau de leurs graphes de composition abstraits sont traités par le reste des règles de conformité. Et ils ne sont considérés comme valides que si le nombre d'opérateurs ALT est supérieur à trois.

Les règles itératives sont une autre forme de règles non supportées par notre langage formel. En effet, certaines règles que nous pouvons énoncer nécessitent l'exécution d'un traitement itératif.

Exemple :

« Le graphe de composition abstrait ne doit pas comporter de cycles. »

Cette règle ne peut être exprimée avec notre langage formel. En fait, plusieurs règles de validité par rapport au modèle SIMBAD nécessitent des traitements itératifs. Ceci explique le fait que nous avons isolé ces règles au niveau d'une phase à part et que nous l'avons pas considéré comme une forme de règles de conformité.

Il faut également noter que les règles doivent être basées sur des propriétés de l'objet d'apprentissage. Et donc certaines règles qui touchent à des propriétés relatives par exemple à

l’apprenant ou au modèle de domaine (et donc pas à l’objet d’apprentissage) ne peuvent pas être exprimées.

Exemple :

Tous les concepts présents dans l’objet d’apprentissage composé doivent être d’un niveau inférieur à trois dans le modèle de domaine (i.e. des concepts de haut niveau).

En fait, prendre en compte dans le langage formel des entités autres que les objets d’apprentissage, comporte plus de risques que d’avantages. En effet, le langage risque de devenir trop complexe alors que ce type de règles semble trop restrictif et pas fondamental par rapport à notre objectif de départ.

5. Notation graphique

5.1. Objectif

Communiquer les règles à respecter aux concepteurs des objets d’apprentissage composés est une étape importante pour permettre aux auteurs de comprendre les règles et donc d’anticiper dès la conception pour favoriser la production d’objets conformes.

Une représentation graphique des règles doit permettre d’abord de les organiser par catégorie. En plus, elle doit permettre de simplifier la formulation de ces règles grâce à une notation à la fois expressive et concise. Ceci doit éviter une mauvaise compréhension ou interprétation des règles.

Certaines des règles hybrides, et spécialement les patrons de conception, sont naturellement exprimées via la notation graphique. Nous définissons ci-après les éléments d’une représentation graphique qui tient compte de ces différents éléments.

5.2. Conteneur pour les règles

L’idée de la notation graphique est de structurer les règles autour du graphe de composition. Certaines règles portent sur l’objet tout entier alors que d’autres portent sur un sommet particulier du graphe.

Un conteneur va spécifier dans sa première partie (appelée règles hybrides) la structure du graphe de composition sur lequel il s’applique. Si cette partie est laissée vide c’est qu’il n’y a pas de contraintes particulières sur la forme du graphe de composition. Si une structure est donnée alors elle est décrite entre autres par des conteneurs.

La portée du conteneur englobant est l'objet d'apprentissage composé à contrôler. Alors que la portée d'un conteneur utilisé pour décrire la structure d'un graphe de composition est le sommet qu'il représente (le plus souvent le sommet racine ou un sommet feuille).

Les autres parties du conteneur contiennent les autres règles qui s'appliquent sur le graphe et classifiées en règles sémantiques, structurelles et sur d'annotation.

Au conteneur est associée une étiquette qui lui attribut un identifiant unique (il permet de le désigner d'une façon unique et donc l'ensemble de règles associées) et un type (objet simple (SLO), complexe(CLO), quelconque(LO) ou n'importe quel graphe valide (ANY)).

Figure 36 : Conteneur utilisé dans la notation graphique

5.3. Patrons de conception

L'expression des règles hybrides est basée sur une présentation graphique. Celle-ci contient uniquement deux formes : les conteneurs et les opérateurs. Ces derniers sont représentés par des formes circulaires. Chaque forme prend soit le libellé PAR pour représenter un opérateur « parallèle », soit le libellé ALT pour représenter l'opérateur « alternatif » ou le libellé ANY_Op pour représenter tout opérateur.

A titre d'exemple soit un ensemble de règles de conformité (R_150) qui comporte uniquement la règle hybride *RH_3.*

RH_3 : *L'objet doit commencer par objet d'apprentissage simple avec le rôle « Introduction » suivi par un objet d'apprentissage avec le rôle « présentation ». Il doit être suivi par trois alternatives d'objets d'apprentissage ayant le rôle « Exemple ». Ensuite, il faut*

un accès parallèle à deux objets d'apprentissage avec le rôle « Exercice ». Puis, l'objet doit présenter un objet d'apprentissage avec le rôle « Conclusion ».

Voici l'expression de R_150 en langage formel :

R_150= N1 : SLO N2 : LO, N2 : LO N3 : ALT, N3 : ALT N4 : LO,
N3 : ALT N5 : LO, N3 : ALT N6 : LO, N4 : LO N7 : PAR,
N5 : LO N7 : PAR, N6 : LO N7 : PAR, N7 : PAR N8 : LO,
N7 : PAR N9 : LO, N8 : LO N10 : LO, N9 : LO N10 : LO ;

N1 -> Content -> list (Role) ⊇ {'Introduction'},

N2 -> Content -> list (Role) ⊇ {'Présentation'},

N4 -> Content -> list (Role) ⊇ {'Exemple'},

N5 -> Content -> list (Role) ⊇ {'Exemple'},

N6 -> Content -> list (Role) ⊇ {'Exemple'},

N8 -> Content -> list (Role) ⊇ {'Exercice'},

N9 -> Content -> list (Role) ⊇ {'Exercice'},

N10 -> Content -> list (Role) ⊇ {'Conclusion'} ;

Et voici la notation graphique pour cet exemple :

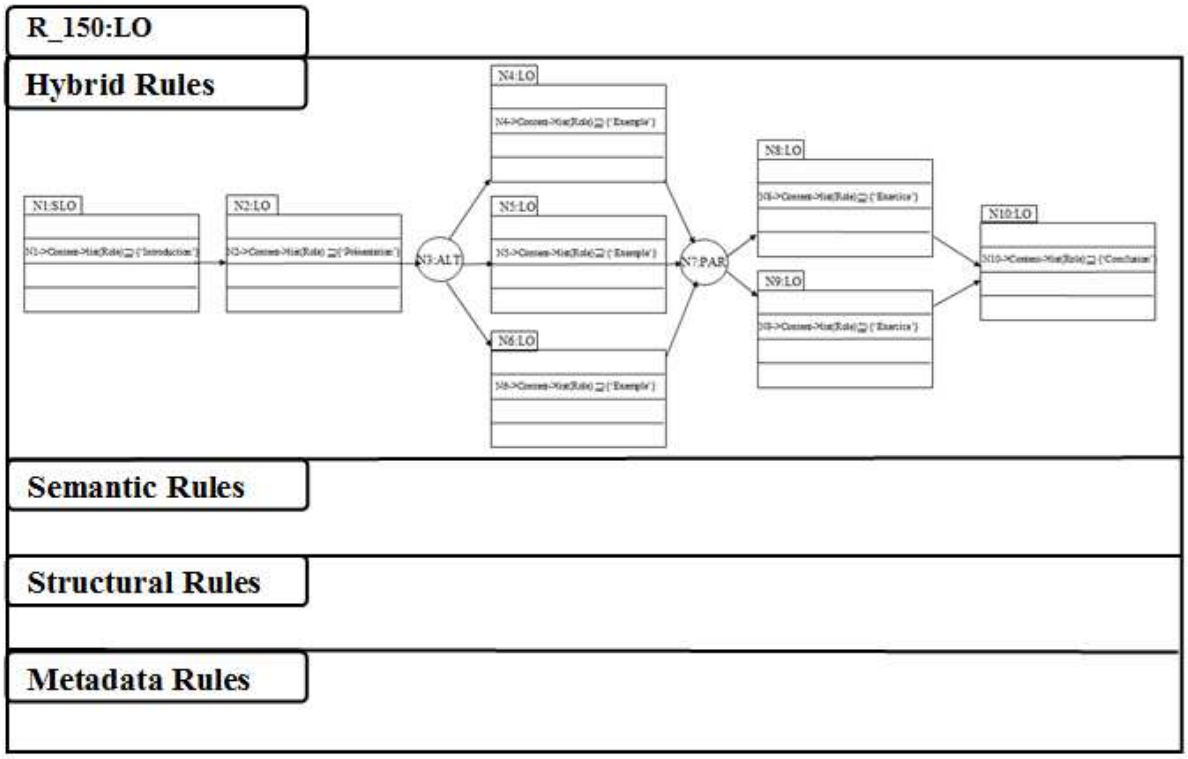

Figure 37 : Exemple de notation graphique pour les règles de conformité

5.4. Expressivité de la notation graphique

En fait, les règles sémantiques, structurelles et d'annotation sont exprimées de la même façon qu'avec le langage formel. Ainsi, on a à ce niveau le même niveau d'expressivité que le langage formel. L'apport de la notation graphique à ce niveau consiste en l'organisation de ces règles par catégorie afin de faciliter leur compréhension par les utilisateurs.

L'intérêt majeur de cette notation est l'expression des règles hybrides. Celles-ci se formulent beaucoup plus simplement autour du graphe de composition. Il reste donc à discuter de l'expressivité de la notation graphique par rapport aux règles hybrides, ce qui revient à discuter de la capacité à décrire tout patron de conception valide avec cette notation.

En fait, la description des règles hybrides repose sur une structure qui est décrite par un ensemble d'arcs. Chaque arc a un sommet de départ et un sommet d'arrivée. Et les sommets se déclinent en des objets d'apprentissage, des opérateurs ou de sommets génériques comme le montre cet extrait de la grammaire du langage formel :

*Graph_specification := Edge [, Edge]**
Edge := Node Node
Node := Id_Node : LO | SLO | CLO | Graph_Op | ANY
Graph_Op := ALT | PAR | ANY_Op

La notation graphique permet de représenter tous ces éléments et de leur associer les caractéristiques que l'on souhaite. En effet, puisque la notation graphique est capable de représenter chaque type de sommet et ainsi que les arcs alors elle est capable d'exprimer toute règle hybride conformément à la grammaire du langage formel.

Cette équivalence entre la notation graphique et le langage formel permet de garantir la faisabilité de la traduction des règles de la première forme vers la deuxième et vice versa.

6. Exemple illustratif

Supposons qu'une université a décidé d'adopter une solution d'apprentissage en ligne pour ses étudiants et pour son personnel. Le système SIMBAD est retenu comme support de formation en ligne. Durant une période d'essai l'administration a constaté un taux d'abondant assez important. Un sondage fait auprès des apprenants a montré que la majorité des critiques concernent les objets d'apprentissage mis à leur disposition.

Cette université peut réagir en demandant à des experts en conception d'objets d'apprentissage et en pédagogie, de définir des règles auxquelles tout objet d'apprentissage doit être conforme pour qu'il soit accepté et adopté.

Ceci va engendrer la définition d'un ensemble de règles en langage naturel. Ces règles doivent être traduites en une notation adaptée pour leur communication aux auteurs potentiels d'objets d'apprentissage. Ces mêmes règles doivent être également traduites en un langage informatique afin d'assurer leur vérification automatique à chaque fois qu'un objet est proposé par un auteur.

6.1. Règles en langue naturelle

Les règles, comme nous l'avons mentionné, peuvent être le résultat de l'accumulation de directives édictées par plusieurs entités. Par exemple, les deux règles suivantes peuvent être définies à l'échelle d'un établissement universitaire :

(R1) *Le nombre de concept traités au niveau du contenu ne doit pas dépasser cinq concepts.*

(R2) *Le nombre de niveaux d'abstraction ne doit pas dépasser quatre niveaux.*

Par la première règle l'établissement peut vouloir s'assurer que le contenu des objets d'apprentissage ne présente pas une surcharge cognitive importante pour les apprenants. En fait, c'est l'un des facteurs qui peuvent justifier un taux important d'abondant.

Alors que par la deuxième règle, l'établissement souhaite éviter que la composition du graphe concret soit très différente de la composition du graphe abstrait ce qui peut engendrer certains problèmes. En effet, l'auteur peut constater que ce qu'il obtient au niveau concret ne correspond pas à son objectif initial. Ceci peut entraîner l'abandon de la conception par réutilisation des objets d'apprentissage et donc des coûts plus élevés en termes d'efforts pour les enseignants et donc pour l'établissement.

A l'échelle des départements d'autres règles plus spécifiques aux disciplines enseignés et à leurs spécificités peuvent être définies. Voici un exemple de trois règles qui peuvent être définies par un département:

(R3) *Le nombre d'objets d'apprentissage au niveau du graphe concret ne doit pas dépasser vingt objets.*

(R4) *Chaque graphe affichable doit inclure des objets ayant les rôles « Introduction », « Exercice » et « Exemple ».*

(R5) *Le nombre des graphes affichables ne doit pas dépasser cinq.*

Le ou les responsables d'un module spécifique peuvent également proposer des règles supplémentaires auxquelles les objets d'apprentissage qui traitent ce module doivent être conformes. Les deux règles suivantes peuvent être définies à ce niveau :

(R6) *Le graphe de composition abstrait doit commencer par un objet d'apprentissage simple ayant le rôle « introduction » et doit contenir un objet avec le rôle « Exercice ».*

(R7) *Tous les graphes affichables doivent partager les deux premiers objets d'apprentissage.*

Les règles sont donc définies par un ou par plusieurs intervenants. Elles peuvent être issues d'une approche pédagogique et/ou d'une expertise. Leurs énoncés sont formulés en langage naturel ce qui donne certes une flexibilité mais en contre partie peut engendrer des risques de mauvaise interprétation ou de confusion. Ce risque prend plus d'ampleur lorsque leur nombre devient élevé.

6.2. Règles en langage formel

Pour formuler les règles suivant notre notation il faut les classifier suivant leurs catégories. Pour notre cas d'étude nous avons des règles qui couvrent trois catégories parmi les quatre possibles. En fait, cet ensemble de règles ne s'intéresse pas aux aspects relatifs aux métadonnées mais à des aspects structurels, sémantiques et hybrides.

En ce qui concerne l'aspect structurel nous avons quatre règles (R2), (R3), (R5) et (R7). Elles sont notées de la façon suivante :

(R2) : LO -> AbstractCG -> abstractionLevels()≤ 4

(R3) : LO -> ConcreteCG -> count('LO') $\leq$ 20

(R5) : LO -> DeliveryGraph -> number() $\leq$ 5

(R7) : LO -> DeliveryGraph -> similarity('Beginning') $\geq$ 2

Pour ce qui relève de la sémantique nous avons deux règles (R1) et (R4). Elles sont notées de la façon suivante :

(R1) : LO -> Content -> count('Concept') $\leq$ 5

(R4) : LO -> DeliveryGraph -> list('Role') $\supseteq$ {'introduction', 'Exemple', 'Exercice'}

Finalement, la règle (R6) est une règle hybride qui est notée de la façon suivante :

(R6) : N1 : SLO N2 : ANY ;

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

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

6.3. Notation graphique pour les règles

L’ensemble des règles est décrit sous la forme suivante :

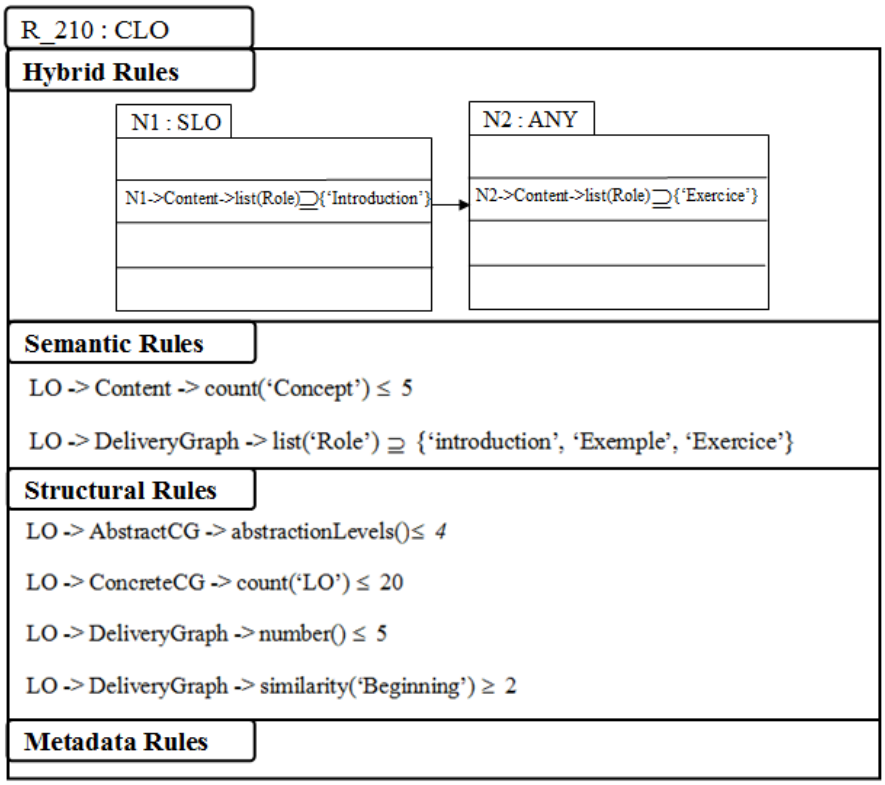

Figure 38 : Notation graphique de règles de conformité

6.4. Vérification automatique des règles

Plusieurs arguments justifient le recours à l’automatisation de la vérification des règles de conformité. D’abord, la vérification manuelle comporte plusieurs risques tels que l’oubli de la vérification de la conformité par rapport à une règle ou l’erreur en considérant ce qui est conforme comme non conforme ou l’inverse. De plus, si le nombre de règles est élevé ou le nombre d’objets d’apprentissage est élevé l’automatisation doit permettre de gagner un temps considérable. Nous pouvons ajouter à ceci le fait que dans un processus de conception itératif et incrémental d’objets d’apprentissage composés, comme c’est le cas dans notre approche d’assistance, faire passer l’objet au niveau de chaque itération au test de la conformité d’une façon non automatisée peut décourager l’auteur à adopter cette approche.

Ainsi, l'automatisation est un élément clé dans notre approche. Pour y parvenir nous avons fait recours à la programmation en F-logique (logique de frames) pour formuler les règles. Ensuite, la vérification automatique se fait par un système tel qu'Ontobroker[9] dans notre cas. Les détails concernant la F-logique et Ontobroker seront présentés dans le chapitre suivant sur la partie expérimentale.

Voici la façon avec laquelle les différentes règles seront exprimées en F-logique :

(R1) *FORALL LO,R,C,N LO:R1Conforms<- LO[hasContent@(R)->>C] and count(LO,C,N) and lessorequal(N,5).*

(R2) *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).*

(R3) *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).*

(R4) *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,"example").*

(R5) *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).*

(R6) *FORALL LO,R,C1,C2,V LO:R6Conforms<- LO[hasContent@("exercise")->>C1] and LO[hasRoot ->V] and V[hasContent@("introduction") ->> C2].*

(R7) *FORALL LO,R,G,N LO:R7Conforms<- LO[hasDGrah->G] and p_dg_beginSimilarity(G,N) and lessorequal(2,N).*

Il faut noter que nous avons défini des prédicats spécialisés. Nous avons donné des noms à ces prédicats qui commencent par « p_ ». Par exemple, p_g_levels(X,Y) est un prédicat qui indique que le graphe de composition X à Y niveaux d'abstraction. Sans la définition de ces prédicats la formulation de la conformité aux règles sera une tâche très difficile avec les prédicats de base de la F-logique.

[9] www.ontoprise.de/en/home/products/ontobroker/

7. Conclusion

Tout au long de ce chapitre nous avons traités les différents éléments relatifs à la phase d'analyse de la conformité des objets d'apprentissage composés par rapport à un patron de conception (ensemble de règles). Nous avons souligné l'importance de cette phase dans un processus de conception d'objets d'apprentissage. Nous avons également présenté une étude des différents aspects qui peuvent être contrôlés par des règles de conformité avec des exemples à l'appui.

Par la suite, nous avons introduit le langage formel de description de règles via sa grammaire et en reprenant les différents exemples énoncés en langage naturel en les décrivant en langage formel. Nous avons également introduit une notation graphique plus adaptée pour la communication des règles de conformité aux auteurs. Les différents éléments de structuration et de description de patrons de conception ont été introduits. L'expressivité du langage formel et de la notation graphique a été discutée. Enfin, nous avons terminé le chapitre avec un exemple complet qui dévoile les différents éléments relatifs à la vérification de la conformité.

Dans le chapitre suivant nous allons, entre autres, présenter et discuter les différents éléments relatifs à l'implémentation et à l'expérimentation du module de la vérification de la conformité et plus largement de notre approche.