République Algérienne Démocratique et Populaire & Ministère de l'Enseignement Supérieur Et de La Recherche Scientifique

Université UHL Batna

Faculté des sciences

Département d'Informatique

N° d'ordre :……

Série :……

MÉMOIRE

Présenté en vue de l'obtention du diplôme de Magistère

Spécialité : Informatique

Option : Système d'Information et de Connaissance

Par :

Aouachria Moufida

SUJET

Une approche basée ontologie pour la réutilisation des connaissances dans le processus d'affaires (Business Process)

Composition de jury :

Pr Bilami azzeddine	Président	Université de Batna
Dr Maamri Ramdane	Rapporteur	Université Mentouri -Constantine
Dr Lahlouhi Ammar	Examinateur	Université de Batna
Dr Belatar Brahim	Examinateur	Université de Batna

Année universitaire 2011/2012

Résumé

Lorsqu'on discute la notion de réutilisation dans les systèmes d'information, on parle généralement sur la réutilisation au niveau des composants logiciels. Pourtant, la tendance actuelle est orientée vers la réutilisation à des niveaux d'abstraction plus élevés, comme l'ingénierie d'entreprise, l'ingénierie de domaine et l'ingénierie d'application. De l'autre côté, les enjeux du processus d'affaires occupent une partie importante dans les méthodologies avancées de l'ingénierie d'entreprise, ainsi que l'importance des ontologies dans la représentation et la réutilisation des connaissances.

Notre travail se situe dans le domaine de la réutilisation de connaissance dans les processus d'affaires en général et dans l'utilisation de l'ontologie dans la réutilisation de connaissances en particulier. Pour cela, nous proposons une approche semi-automatique basée sur les ontologies pour permettre la réutilisation des connaissances du processus d'affaires durant la phase de l'ingénierie de domaine d'application puis dans l'environnement d'exécution approprié. La notion de processus d'affaires générique est introduite pour désigner une famille du processus d'affaires similaires. Selon la conceptualisation de la variabilité entre les membres de même famille, nous distinguons dans notre approche, trois phases qui sont : la phase d'ingénierie du domaine de processus, la phase d'ingénierie applicative du processus et la phase d'exécution du processus. Afin de réaliser l'approche proposée, nous devons tout d'abord séparer l'ontologie de processus d'affaires de celle du domaine d'application. Ensuite réutiliser l'ontologie de processus dans les différents domaines d'application pour qu'il soit exécuté dans un environnement d'exécution choisi. C'est pourquoi, nous proposons d'enrichir deux types d'ontologies réutilisables de haut niveau : Ontologie du domaine d'application de haut niveau et Ontologie du processus d'affaires de haut niveau. Les deux ontologies sont basées sur un système de méta-concepts, qui constitue une ontologie de haut niveau, que nous traitons également. Nous mettons beaucoup plus l'accent sur la conceptualisation de variabilité, où nous proposons un modèle pour résoudre les points de variation statiques et dynamiques, qui est notre contribution principal dans ce mémoire. Finalement, nous discutons une étude de cas.

Mots clés : Ontologie, Processus d'Affaires Générique, Réutilisation, Ingénierie des connaissances, Ingénierie de domaine, la configuration, Feature Model.

ملخص

عندما نناقش مفهوم إعادة الاستخدام في أنظمة المعلومات، نتحدث في الغالب على إعادة استخدام بعض العناصر أو المكونات البرمجية. إلا أن التوجه الحالي يميل نحو إعادة الاستخدام في مستويات تجريد أعلى، مثل هندسة الأعمال، هندسة المجال و كذا هندسة التطبيقات. من جهة أخري نلاحظ أن مواضيع سير أو إجراءات الأعمال في الآونة الأخيرة أصبحت تشغل جزءا هاما في الطرق المتطورة الخاصة بهندسة المؤسسة، بقدر أهمية الانطولوجيا في تمثيل و إعادة استخدام المعرفة.

عملنا في هذه المذكرة ينطوي تحت مجال إعادة استخدام المعرفة في سير الأعمال بصفة عامة، و استعمال الانطولوجيا في إعادة استخدام المعارف بصفة أخص. من اجل ذلك، نقترح منهجية جديدة شبه تلقائية ترتكز علي الانطولوجيا للسماح بإعادة استخدام المعارف الخاصة بسير الأعمال، أثناء هندسة مجال تطبيقها وكذا أثناء تنفيذها في المحيط المناسب. أدخل هنا مفهوم السير العام للأعمال للدلالة على عائلة لسير الأعمال المتماثلة. وفقا لمفهوم التباين بين أفراد العائلة الواحدة، سنميز في هذه المنهجية ثلاث مراحل هي: مرحلة هندسة مجال سير الأعمال، مرحلة هندسة مجال تطبيق سير الأعمال، مرحلة تنفيذ سير الأعمال. لتحقيق هذه المنهجية، يجب أولا فصل انطولوجيا سير الأعمال عن انطولوجيا مجال تطبيقها، ثم إعادة استخدام هذه الانطولوجيا ـ أي انطولوجيا سير الأعمال ـ في مختلف مجالات تطبيقها لتنفيذها في المحيط المحدد بشكل جيد. لذا، فإننا نقترح تحسين نوعين من الانطولوجيا ذات المستوى العالي و التي يمكن إعادة استخدامها : انطولوجيا مجال التطبيق ذات المستوى العالي؛ انطولوجيا سير الأعمال ذات المستوى العالي. وتستند الاثنتين على بعض المفاهيم الفوقية، و التي تمثل انطولوجيا المستوى العالي، التي نناقشها أيضا. سوف نركز أكثر على مفهوم التباين بين أفراد العائلة الواحدة. أين نقترح نموذجا لحل نقاط الاختلاف الثابتة والديناميكية، حيث أنه يمثل مساهمتنا الأساسية في هذه المذكرة. أخيرا، نقترح مثالا لمناقشة وتطبيق هذه المنهجية.

الكلمات المفتاحية: الانطولوجيا، السير العام للأعمال ، إعادة الاستخدام، هندسة المعرفة، هندسة المجال، تمثيل الخصائص.

Remerciements

*Ce travail a été réalisé sous la direction du **Docteur Maamri Ramdane**, Maître de Conférences à l'université Mentouri de Constantine.*

*Mes plus sincères remerciements vont à mon encadreur **M. Maamri Ramdane**, pour avoir accepté de diriger mon mémoire de Magistère. Je le remercie vivement pour le temps qu'il a consacré à mon mémoire. Merci pour vos explications, pour vos conseils et notamment pour votre patience.*

Je tiens à remercier très sincèrement l'ensemble des membres du jury qui me font le grand honneur d'avoir accepté de juger mon travail.

*Je remercie vivement Monsieur **Pr Bilami azzeddine** d'avoir accepté de présider ce jury.*

*Mes vifs remerciements à **Dr Lahlouhi Ammar**, d'avoir accepté de juger ce travail.*

*Je tiens également à remercier **Dr Belatar Brahim**, c'est un grand honneur pour moi de vous compter parmi les membres du jury.*

Je tiens également à remercier Dr.Toufik Zaidi, pour ses conseils, ses remarques et ses idées intéressantes.

Sans oublier Dr. S.Bouame de ces observations et encouragements ainsi que M.Mohammed Betta pour ces encouragements et aides depuis l'ingéniorat.

Je souhaite remercier également M. S.Mansouri spécialiste en Management des affaires, pour la documentation, pour ses conseils, ses idées et son aide.

Je remercie, M. C. Lattari qui a corrigé les fautes d'orthographes que j'avais commises. Ainsi que, Dr ben boulaid pour la traduction Anglais/Français.

Je ne saurais oublier dans ces remerciements tous mes collègues et amies .Je leur exprime ma profonde sympathie et leur souhaite beaucoup de bien.

Pour terminer, je voudrais remercier ma famille, en particulier ma mère, merci pour votre encouragement et pour votre patience.

TABLE DES MATIÈRES

Introduction Générale

Partie -01- L'État de l'art

Chapitre -01-

Représentation de la connaissance et ingénierie ontologique

Chapitre -02-

Processus d'affaires (Business Process)

Chapitre -03-

Réutilisation des connaissances ontologiques dans le processus d'affaires

Partie -02- Contribution

Chapitre -04-

Proposition

Chapitre -05-

Étude de cas et Implémentation

LISTE DES FIGURES

Chapitre -01-

Chapitre -02-

Chapitre -04-

Chapitre -05-

LISTE DES TABLEAUX

Chapitre -01-

Chapitre -04-

Chapitre -05-

Introduction Générale

« Vous voyez les choses qui existent et vous dites : pourquoi ?
Moi, je rêve des choses qui n'ont jamais existé et je dis pourquoi pas ? »
Georges Bernard Shaw

1. Contexte de travail et problématique

La majorité des systèmes d'informations actuels repose sur la notion de réutilisation, laquelle a été restreinte au niveau des composants logiciels tels que (fonction, objet,…) ; Toutefois, la tendance actuelle est orientée vers la réutilisation à des niveaux d'abstraction plus élevés, tels que l'ingénierie d'entreprise, l'ingénierie de domaine et l'ingénierie d'application. Cela est dû à la complexité, à la diversité et à l'augmentation des exigences et contraintes économiques.

Récemment les enjeux de *l'ingénierie du processus d'affaires* sont devenus une partie importante dans les méthodologies avancées de l'ingénierie d'entreprise. En particulier, on recense la Gestion des Processus d'affaires (BPM) [SMIT03] et l'Architecture Orientée Service (SOA) [ERL05], qui facilitent la réutilisation des logiciels [CAPL03]. Cependant, la représentation des connaissances sur les familles des processus d'affaires similaires et leur réutilisation dans des projets de l'ingénierie d'entreprise reste un problème posé [ČIUK07, ČIUK07a].

Dans ce contexte, un système intégré des ontologies pour soutenir la modélisation d'entreprise a été développé à Toronto Virtual Enterprise (TOVE) projet [FOX92]. Ce système se compose d'un certain nombre d'ontologies génériques de base, y compris une ontologie d'activité, une ontologie des ressources, une ontologie d'organisation, et une ontologie de produits. L'approche TOVE est critiquée puisqu'elle exige trop d'efforts pour instancier le modèle d'une entreprise particulière. Afin de faciliter cette tâche, les auteurs travaillent sur le développement des outils de son automatisation [CAPL03].

Dans le même aspect, un autre travail [USCH96] qui décrit un projet d'entreprise à l'Université d'Édimbourg propose une ontologie (Ontologie D'Entreprise) pour la modélisation d'entreprise et qui vise à soutenir un environnement de l'ingénierie d'entreprise. Cet environnement est conçu comme un ensemble intégré de méthodes et d'outils pour saisir et analyser les aspects principaux d'une entreprise. L'ontologie d'entreprise fournit cinq classes de haut niveau pour l'intégration des différents aspects d'une entreprise : une méta-ontologie, activités et processus, organisation, stratégie et le

marketing. La méta-ontologie définit les concepts de base de modélisation (entité, relation, rôle, acteur, état d'affaire).

Les termes sont exprimés sous une forme restreinte et structurée de la langue naturelle complétée par quelques axiomes formels utilisant Onto lingua. Par conséquent, il ne supporte pas le raisonnement automatique [CAPL03].

Nous avons aussi constaté que les travaux de Caplinskas et ses coéquipiers [CAPL02, CAPL03, CAPL03a, CAPL04, ČIUK06, ČIUK07, ČIUK07a] sont très intéressants, dans la mesure où ils ont proposé une approche à base d'ontologies qui permet la réutilisation des connaissances au niveau du processus d'affaires. Cette approche, qui est en cours d'évolution, consiste à combiner les différentes techniques de réutilisations développées dans l'ingénierie de domaine [CZAR00], l'ingénierie de connaissances [CHAN86] et l'ingénierie des systèmes basés sur les ontologies [DAVI06].

L'idée générale de l'approche proposée est d'abord de séparer l'ontologie de processus d'affaires de l'ontologie de domaine d'application, ensuite réutiliser l'ontologie de processus dans les différents domaines d'application. C'est pourquoi, ils ont proposé deux types d'ontologies réutilisables de haut niveau : Ontologie du domaine d'application de haut niveau et Ontologie du processus d'affaires de haut niveau. Les deux ontologies sont basées sur un système de méta-concepts, qui constitue une ontologie de haut niveau, qu'ils ont également traité. La notion de processus d'affaires générique est introduite pour désigner une famille de processus d'affaires similaires. Selon la conceptualisation de la variabilité entre les membres de même famille, ils ont distingué dans cette approche, deux phases pour localiser le processus d'affaires générique dans un domaine d'application bien choisi. Les deux phases sont : l'ingénierie du domaine de processus (développement pour la réutilisation) et l'ingénierie de processus (développement avec la réutilisation). La réutilisation présentée dans cette approche permet la résolution de variabilité d'une manière statique c'est à dire durant la construction du processus d'affaires localisé dans un domaine d'application choisi, sans tenir compte de la variabilité qui peut avoir lieu durant l'exécution de ce processus.

En plus, d'après JIANQI YU [JIAN10], les approches pour la réutilisation à deux phases développées dans l'ingénierie de domaine et adaptées par Caplinskas, sont mises en place avec succès ; mais elles ne proposent pas de dynamisme à l'exécution. De plus, elles ne conviennent pas très bien aux environnements dynamiques changeants. De ce fait, nous trouvons qu'il a pris en considération la variabilité qui aura lieu durant l'exécution

d'applications, où il a proposé une approche à trois phases : Une phase d'ingénierie domaine, Une phase d'ingénierie applicative et Une phase d'exécution.

2. Notre contribution

Pour les raisons citées précédemment, nous présenterons dans ce mémoire une nouvelle approche, semi automatique, qui se base sur les ontologies pour permettre la réutilisation des connaissances du processus d'affaires durant l'ingénierie de domaine d'application puis dans l'environnement d'exécution choisi. C'est-à-dire nous allons distinguer deux types des points de variation, statiques et dynamiques. Les points de variation statiques seront résolus durant la construction du processus dans le domaine d'application choisi et les autres points de variations dits dynamiques seront résolus durant l'exécution de ce processus dans l'environnement d'exécution choisi aussi. Cela pour tenir compte de la variabilité qui peut avoir lieu durant l'exécution de ce processus.

Nous allons nous baser sur l'approche proposée par JIANQI YU sur les lignes de production et largement sur celle de Caplinskas et ses coéquipiers sur les processus d'affaires, qui ont été toutes les deux visées à la réutilisation, pour construire notre approche.

Donc notre contribution, consiste à :

Premièrement, distinguer deux parties dans la phase de l'ingénierie de processus proposée par Caplinskas : la phase de l'ingénierie applicative de processus et la phase de l'exécution de processus. Ainsi, l'architecture de l'approche que nous allons proposer comporte trois phases : la phase d'ingénierie du domaine de processus, la phase de l'ingénierie applicative de processus et la phase de l'exécution de processus.

La première phase, ou l'ingénierie du domaine de processus, vise à mettre en place au sein d'un domaine un ensemble d'éléments réutilisables fondés autour des notions de modèles des caractéristiques et d'ontologies. Ces notions représentent un processus d'affaires générique, que nous présenterons aussi dans ce chapitre.

La seconde phase, ou l'ingénierie applicative du processus, concernant l'exploitation des éléments communs précédemment définis. Son objectif est de spécifier et localiser, de façon plus précise, les processus d'affaires dans un domaine d'application choisi.

Enfin, *la troisième phase*, a pour but d'exécuter le processus d'affaires spécifié, en identifiant et en utilisant aussi au dernier moment les services ou les activités appropriées et disponibles pour l'exécution des autres activités convenables. Un système de gestion des Workflow est invoqué pour gérer l'exécution du processus d'affaires. Donc l'objectif de la troisième phase est d'intégrer le processus d'affaires, localisé dans un domaine d'application déjà choisi, dans un environnement d'exécution bien spécifié.

Deuxièmement, cette approche se concentre sur les notions de modèles de caractéristiques et des ontologies, celles-ci, nous les définissons de façon précise à l'aide du méta modèles ou de ce qu'on appelle les ontologies de haut niveau. C'est pourquoi, nous proposons d'enrichir deux types d'ontologies réutilisables de haut niveau : Ontologie du domaine d'application de haut niveau et Ontologie du processus d'affaires de haut niveau. Nous mettrons beaucoup plus l'accent sur la conceptualisation de variabilité, où nous allons proposer un modèle pour résoudre les points de variation statiques et dynamiques, qui sera notre contribution principale dans ce mémoire. Les deux ontologies sont basées sur un système de méta-concepts, qui constitue une ontologie de haut niveau, que nous traiterons également.

Troisièmement, nous présenterons une étude de cas pour illustrer cette approche. Nous allons prendre l'exemple d'un processus de conférence simplifié qui sera considéré comme un processus générique. Cela signifie qu'il pourra être réutilisé dans différentes universités, collèges et pour faire une présentation lors des conférences. *Tout d'abord,* nous donnerons une description détaillée de l'application de cette approche sur le processus de conférence, qui sera considéré comme un processus d'affaires générique. Cette description sera durant les trois phases à proposer : l'ingénierie du domaine de processus, l'ingénierie d'applicative du processus ainsi que la phase de l'exécution de processus. *Ensuite*, nous ferons un aperçu schématisé sur l'aspect d'implémentation de cette approche. Nous utiliserons l'outil Protégé pour implémenter toutes les ontologies déjà mentionnées dans cette approche, avec les différentes relations, restreintes et règles de dépendance entre les différents concepts. Le raisonneur *Racer* permettra de tester la cohérence et la consistance de ces ontologies. L'outil Protégé ainsi que le raisonneur *Racer*, nous permettront aussi d'utiliser la logique de description pour résoudre le problème de la configuration de processus. Cela sera réalisé en utilisant les différentes relations, restreintes et règles de dépendances entre les différents concepts, notons que le plugin *SWRL RULES* nous permettra de définir ces règles de dépendance. Pour l'attribution des rôles, nous introduirons la fonction *Map* de plugin *Prompt* de l'outil

Protégé. L'implémentation de modèle exécutable de processus cours sera possible via un langage d'exécution des processus tel que WSBPEL, qui devra être exécuté par un système de gestion des Workflow tels que Intalio|BPMS ou BONITA.

3. Plan du mémoire

Ce mémoire est organisé comme suit, après cette introduction, le développement est en deux parties : l'état de l'art et la proposition.

La première partie présente l'état de l'art. Les travaux de ce mémoire se situent à la convergence de trois domaines, la représentation des connaissances ontologiques, les processus d'affaires et finalement la réutilisation des connaissances. Pour cette raison, l'état de l'art est divisé en trois chapitres :

Chapitre 01 : rappel sur la représentation de la connaissance d'entreprise, la définition et le développement de l'ontologie, le rôle des ontologies dans la gestion des connaissances ainsi que la mise en correspondance d'ontologies et les différentes techniques d'appariement entre ces ontologies (Matching).

Chapitre 02 : présente une description détaillée des processus d'affaires, des exemples et des concepts fondamentaux dans l'ingénierie d'entreprise, ainsi que les termes et méthodologies principales pour faire la modélisation ou l'exécution de ces processus d'affaires.

Chapitre 03 : décrit la notion de réutilisation, les différentes techniques de réutilisation. Il présente aussi un état de l'art des travaux existant auteur de la réutilisation des connaissances ontologiques dans le domaine d'ingénierie d'entreprise et plus exactement dans les processus d'affaires.

La deuxième partie introduit notre contribution et est structurée en deux chapitres :

Chapitre 04 : présente en détail le principe de notre proposition qui permettra la réutilisation en trois phases : *la phase d'ingénierie du domaine de processus, la Phase d'ingénierie applicative et celle d'exécution.* Ensuite, pour soutenir cette proposition, nous allons enrichir les deux types d'ontologies réutilisables de haut niveau : l'une pour la représentation des connaissances d'un processus d'affaires générique, pendant la phase de l'ingénierie de domaine. L'autre ontologie sera désignée pour la représentation des connaissances d'un domaine d'application particulier, pendant la phase d'ingénierie applicative du processus. Les deux ontologies seront basées sur une ontologie de haut niveau, que nous exposerons aussi. Nous mettrons beaucoup plus l'accent sur la conceptualisation de variabilité, où nous allons proposer un modèle pour résoudre les

points de variation statiques et dynamiques, qui sera notre contribution principale dans cette approche.

Chapitre 05 : présente une étude de cas pour illustrer notre approche. *En premier lieu*, nous allons donner une description ou une conceptualisation détaillée de l'application de cette approche sur le processus conférence, qui sera considéré comme un processus d'affaires générique. *En second lieu*, nous allons donner un aperçu sur l'aspect de l'implémentation de cette approche. L'outil Protégé avec ses différents plugin (*SWRL RULES*, fonction *Map* de plugin *Prompt*, *etc.*) ainsi que le raisonneur *Racer*, nous permettront de construire les différents ontologies, tester leur cohérence, utiliser la logique de description pour résoudre le problème de la configuration de processus, construire les règles de dépendance entre les différents concepts par le plugin *SWRL RULE*, attribution des rôles avec la fonction *Map* de plugin *Prompt*. L'implémentation de modèle exécutable de processus généré sera possible via un langage d'exécution des processus tel que WSBPEL, ce dernier devra être exécuté par un système de gestion des Workflow tel que Intalio|BPMS ou BONITA.

On termine ce mémoire par une conclusion générale dans laquelle on présente une synthèse de notre contribution. Et aussi on identifie les perspectives de ce travail.

Chapitre -01-

Représentation de la connaissance et ingénierie ontologique

1. Introduction

"Representation is a stylized version of the world. Depending on how it is to be used, a representation can be quite simple, or quite complex". Charniak et McDermott

Avec le développent croissant des connaissances et la complexité de leur gestion au sein de l'entreprise, une meilleure technique et langage de représentation des connaissances doivent être présentés. Pour pouvoir représenter des connaissances propres à un domaine particulier, on doit décrire et coder les entités de ce domaine sous une forme qui puisse être interprétée et exploitée par l'homme et par la machine.

Il existe plusieurs familles de langages de représentation des connaissances, les expressions du langage reposent sur une *syntaxe*, un *procédé* de construction de formules bien formées et une *sémantique*. La sémantique donne un sens aux formules et justifie la validité des opérations effectuées par le système de représentation sur les formules.

Une ontologie est une conceptualisation d'un domaine, elle vise à représenter cette connaissance en étant à la fois interprétable par l'homme et par la machine.

Dans ce chapitre, nous allons faire un rappel sur la représentation de la connaissance d'entreprise, la définition et le développement de l'ontologie ainsi que le rôle des ontologies dans la gestion des connaissances.

2. Qu'est-ce qu'une Connaissance ?

Avant toute réflexion sur les connaissances, nous devons tout d'abord différencier les termes de données, d'informations et de connaissances.

La donnée est le moins porteur de sens de tous ces termes. Les données ne sont ni vraies, ni fausses, ni significatives à moins d'être récupérées, représentées et réinterprétées. Elles sont transmises à un système ou un programme qui les traite, les modifie et les fait évoluer [HERN05], Exemple : 35, âge, Omar,…etc.

Toute **information** est issue de données qui sont structurées pour constituer une information. Les données deviennent informations quand elles prennent un sens soit pour le système soit pour l'utilisateur [HERN05], Exemple : l'âge d'Omar égale à 35.

L'information, constituée de données, devient **connaissance** à partir du moment où elle sert de fondement à une inférence, au déclenchement d'un processus [LAME02], Exemple : Il faut que Omar soit âgé moins de 35 ans pour qu'il puisse avoir un contrat de pré-emploi. Dans ce qui suit, nous définissons les deux catégories des connaissances d'entreprise.

L'élément de tout système basé sur la connaissance est une base de connaissances. **Une base de connaissances** est capable de stocker des données mais est également capable de leur associer une représentation formelle. La représentation formelle de connaissances a pour objectif de rendre celles-ci interprétables aussi bien par une machine que par un être humain.

3. La représentation des connaissances :

Il est possible d'identifier cinq rôles principaux joués par la représentation des connaissances [OLIV03] :

~ *Une représentation des connaissances est un substitut :* les objets physiques, les évènements et les relations sont représentés par des symboles.

~ *Une représentation des connaissances est un ensemble de conventions ontologiques :* l'ontologie est l'étude de l'existence, elle détermine les catégories des choses qui existent ou qui sont susceptibles d'exister pour le domaine d'étude. Ces catégories représentent les conventions ontologiques des concepteurs.

~ *Une représentation des connaissances est une théorie fragmentaire sur le raisonnement :* pour permettre le raisonnement sur les éléments d'un domaine, la représentation des connaissances doit décrire leurs comportements et leurs interactions.

~ *Une représentation des connaissances est un moyen de faire efficacement des calculs :* une représentation des connaissances doit pouvoir être exploitée efficacement.

~ *Une représentation des connaissances est un moyen d'expression pour les humains :* elle devrait faciliter la communication entre les ingénieurs de connaissances et les experts du domaine.

3.1. Les niveaux de représentation des connaissances

Guarino [GUAR95] a révisé les distinctions introduites dans [BRAC79], où les langages de représentation des connaissances ont été classés selon les types de primitives offertes à l'utilisateur. Donc, Guarino a proposé l'introduction d'un autre niveau - le niveau ontologique - intermédiaire entre l'épistémologique et le conceptuel (Tableau 1.1).

Niveau	Primitives	Interprétation	Caractéristique
Logique	Prédicats, fonctions	Arbitraire	Formalisation
Épistémologique	Relations structurantes	Arbitraire	Structure
Ontologique	Relations ontologiques	Contrainte	Signification
Conceptuel	Relations conceptuelles	Subjective	Conceptualisation
Linguistique	Termes linguistiques	Subjective	Dépend du langage

Tableau 1.1 Niveaux de la représentation des connaissances (selon Guarino [GUAR95])

Au niveau logique (premier ordre) (le niveau de la formalisation), les primitives de base sont des prédicats et des fonctions, qui ont donné une sémantique formelle en termes de relations entre les objets d'un domaine. Cependant, aucune hypothèse particulière n'est faite sur la nature de ces relations, qui sont tout à fait générales et le contenu indépendant. Le niveau logique est le niveau de formalisation : il permet une interprétation formelle des primitives, mais leur interprétation est cependant tout à fait arbitraire (par exemple : $\exists x$. *Ballon(x)* ∧ *hasColor(x, Rouge*).

Le niveau épistémologique (le niveau de la structure) : Brachman [BRAC79] a proposé l'introduction d'un langage situé à un niveau intermédiaire, où les primitives nous permettent de préciser «la structure formelle des unités conceptuelles et de leurs interrelations comme des unités conceptuelles (indépendantes de toutes connaissances qui y sont exprimées)" [BRAC79]. Il repose sur un ensemble de primitives indiquant, par exemple que Ballon est un concept, et que *hasColor* est une relation, sans donner de sens à ces primitives.

Le niveau ontologique est le niveau de la signification : ces engagements ontologiques associés aux primitives du langage sont spécifiés explicitement. Une telle spécification peut être effectuée de deux manières : soit par restreindre convenablement la sémantique des primitives, ou en introduisant des propositions de signification exprimées dans le langage lui-même. Dans les deux cas, l'objectif est de limiter le nombre d'interprétations possibles. Les définitions fournies au niveau ontologique peuvent ainsi éviter d´écrire Rouge comme un concept par exemple.

Au niveau conceptuel, les primitives sont définies de façon cognitive, et la structure du domaine modélisé (son squelette dans le texte) est fixée. Les connaissances sont modélisées en spécialisant ce squelette. Dans un certain domaine d'application, l'utilisateur est obligé d'exprimer la connaissance sous forme d'une spécialisation de ce squelette.

Le niveau linguistique fait référer directement aux termes du langage.

Parmi les formalismes de représentation des connaissances, nous trouvons plusieurs grandes familles aux propriétés similaires, même si les aspects syntaxiques sont variables : Logiques (propositions, prédicats, descriptions), Graphes et Réseaux sémantiques…etc.

Nous décrivons ci après quelques uns de ces formalismes.

3.2. Les langages de représentation des connaissances

Le but de la représentation des connaissances est de rendre compte d'un domaine particulier de telle sorte que cette représentation soit manipulable par la machine. Donc nécessité d'un langage qui modélise le domaine tels que :

3.2.1. Les Réseaux sémantiques

Leur origine est attribuée à Quillian qui a proposé en 1968 de construire un modèle de la « mémoire humaine » fondé sur un réseau sémantique de mots construits à partir d'expériences en psycholinguistique. Les réseaux sémantiques représentent un graphe orienté et étiqueté, ils constituent une manière de représenter les relations entre les objets (nœuds) de domaine modélisé. Ces nœuds (objets) sont reliés entre eux, ces liens ont un sens. Les liens sont orientés car la relation n'est pas symétrique.

Les Réseaux sémantiques ont mené à la définition de nouveaux formalismes tels que les frames, les logiques de description et les graphes conceptuels.

3.2.2. Les frames Minsky [MINS75]

Les cadres conceptuels (Frame) [MINS75] sont des structures de données qui peuvent décrire des connaissances stéréotypées. Les Frames sont organisés en classes et en sous-classes. Il y a donc ici une hiérarchie entre les classes qui contiennent les sous-classes. Les sous-classes héritent des propriétés (attributs) des classes. Les attributs sont des données de type varié ; ils peuvent être aussi des procédures qui manipulent les attributs du frame ou d'autres frames.

3.2.3. Les graphes conceptuels

Les graphes conceptuels développés par Sowa [SOWA84] sont des formalismes de représentation logique basés sur les graphes existentiels de Charles Sanders Peirce et les réseaux sémantiques. Ils furent créés afin de permettre une représentation du langage naturel. De façon générale, un graphe conceptuel est défini comme un graphe qui a deux sortes de nœuds : *Les nœuds concepts* qui représentent des entités, des attributs, des états, des événements...etc. *Les nœuds relations conceptuelles* qui symbolisent les liens qui existent entre deux concepts. Ainsi, un graphe conceptuel est : *Orienté. Fini :* tout graphe dans une mémoire d'un ordinateur ne peut avoir qu'un nombre fini de nœuds. *Connexe :* si deux parties n'étaient pas connectées entre elles on aurait deux graphes conceptuels. *Bipartie :* il ne possède que deux sortes de nœuds - les concepts et les relations conceptuelles- chaque arc reliant une sorte de nœud à l'autre sorte de nœud.

3.2.4. Les Logiques De Description

Les logiques de description [BARC77] sont nées pour pallier au manque de sémantique formelle des réseaux sémantiques et des systèmes à base de frames, qui sont basés sur une lecture intuitive de constructeurs proches du langage naturel [NARD03].

Les logiques de description [BAAD91] forment une famille de langages de représentation de connaissances d'un domaine d'application d'une façon structurée et d'une sémantique formelle, l'une des raisons qui nous amène à l'utiliser pendant la phase de configuration dans l'approche à proposer.

Les logiques de description utilisent les notions de concept, de rôle et d'individu. Les concepts correspondent à des classes d'individus, les rôles sont des relations entre ces individus. **Exemple 01 :** Cet exemple décrit les relations entre les membres d'une famille : Concepts : *Femme, Homme, Personne ...* Rôles : *enfant-de, père-de ...*Instances : *Karim, Amina ....*

En particulier, les logiques de description (LD) sont apparentées aux formalismes de représentation des connaissances par objets. La sémantique d'une telle logique s'exprime grâce à des notions ensemblistes. Ainsi, étant donné une interprétation *I*, un concept C s'interprète comme un ensemble I (C), un rôle r s'interprète comme une relation binaire I (r) et une instance a comme un individu I (a). Les opérations ∩, U sur les ensembles sont représentées par les connecteurs ∩, U, sur les concepts. Par exemple, étant donné deux concepts C et D et une interprétation *I*, on a $(C \cap D)^I = (C)^I \cap (D)^I$.

Les logiques de description ont une base commune enrichie de différentes extensions :

~ *ℒ (Attributive Language):* c'est le langage de base défini à partir des éléments syntaxiques suivants :

A: Concept atomique, ⊤ : Le concept universel Top, ⊥ : Le concept vide Bottom, ¬ A : Négation d'un concept atomique, C ∩ D: Conjonction de concepts,

∀ *r.C* : Quantificateur universel, ∃*r :* Quantificateur existentiel non typé.

D'autres langages, plus expressifs, peuvent être définis en rajoutant d'autres constructeurs au langage AL, nous avons par exemple [CHAA09] :

~ *ℒ𝒰 = ℒ 𝒰 {C 𝒰 D}* : Disjonction de concepts.

~ *ℒℰ = ℒ 𝒰* {∃r.C} : Quantificateur existentiel typé.

~ *ℒ𝒞* (Attributive Language with Complement) : c'est la logique la plus importante, elle constitue la base de toutes les logiques de descriptions « expressives ».

ℒ𝒞 = ℒ 𝒰 {¬ C} : Ici, C est un concept primitif ou défini. On note qu'on peut coder la disjonction resp. la quantification existentielle à l'aide de la négation et la conjonction

resp. La quantification universelle, pour obtenir $\mathcal{ALU}$ resp. $\mathcal{ALE}$ comme sous-logiques d'$\mathcal{ALC}$.

~ $\mathcal{ALN} = \mathcal{AL} \cup \{\leq nr; \geq nr\}$: Les restrictions de nombres, désignées par la lettre n et notées par nr(restriction a moins de *n*) et *nr* (restriction a plus de *n*) ou *n* représente un entier positif.

Il existe plusieurs LDs dont la différence est de posséder plus ou moins de constructeurs et de type d'axiomes. Nous présentons dans le Tableau 1.2 les principaux constructeurs des LDs :

Nom	Syntaxe LD	Syntaxe abstraite	Sémantique
Nom de classe Top, Thing Bottom, Nothing	C ⊤ ⊥	C (URI) owl: Thing owl : Nothing	$C^I \subset A^I$ $\top^I = \Delta^I$ $\bot^I = \emptyset$
Intersection Union Négation Enumération	C ∩ D C U D ¬C {a, b ...}	intersection0f(C D) union0f(C D) complement0f(C) one0f *(a b ...)*	$(C \cap D)^I = C^I \cap D^I$ $(C \cup D)^I = C^I \cup D^I$ $(\neg C)^I = \Delta^I \setminus C^I$ $\{a, b \ldots\}^I = \{a^I, b^I \ldots\}$
Quantificateur existentiel Quantificateur universel Restriction à une valeur Restriction non qualifiée de cardinalité	∃p.C ∀p.C ∋p.a = n p ≥n p ≤n p	restriction(p someValuesFrom(C)) restriction(p allValuesFrom(C)) restriction(p hasValue(a)) restriction(p cardinality(C)) restriction(p minCardinality(C)) restriction(p maxCardinality(C))	$(\exists p.C)^I = \{x \mid \exists y, (x, y) \in p^I$ et $y \in C^I\}$ $(\forall p.C)^I = \{x \mid si\ \forall y, (x, y) \in p^I$ alors $y \in C'\}$ $(\ni p.a)^I = \{x \mid (x, a^I) \in p^I\}$ $(= n\ p)^I = \{x \mid card\{y \mid (x, y) \in p^I\} = n\}$ $(\geq n\ p)^I = \{x \mid card\{y \mid (x, y) \in p^I\} \geq n\}$ $(\leq n\ p)^I = \{x \mid card\{y \mid (x, y) \in p^I\} \leq n\}$
Quantificateur existentiel Quantificateur universel Restriction à une valeur Restriction non qualifiée de cardinalité	∃d.T ∀d.T ∋d.a = n d ≥ n d ≤n d	restriction(d someValuesFrom(T)) restriction(d allValuesFrom(T)) restriction(d hasValue(a)) restriction(d cardinality(T)) restriction(d minCardinality(T)) restriction(d maxCardinality(T))	$(\exists d.T)^I = \{x \mid \exists y, (x, y) \in d^I \wedge y \in T^I\}$ $(\forall d.T)^I = \{x \mid \forall y, (x, y) \in d^I \rightarrow y \in T^D\}$ $(\ni d.a)^I = \{x \mid (x, d^D) \in d^I\}$ $(= n\ d)^I = \{x \mid card\{y\ (x, y) \in d^I\} = n\}$ $(\geq n\ d)^I = \{x \mid card\{y\ (x, y) \in d^I\} \geq n\}$ $(< n\ d)^I = \{x \mid card\{y\ (x, y) \in d^I\} \leq n\}$

Tableau 1.2 Constructeurs de classes OWL et leur correspondance en LD. C et D sont des classes, T est un type de donnée, n est un nombre, a et b sont des individus, p une proprieté d objet (ObjectProperty) et d une proprieté de données (DatatypeProperty) (Selon Adrien Coulet [ADRI07]).

Une base de connaissances BC en LD est généralement composée de deux parties (niveaux) une TBox et une ABox :

1. **Niveau terminologique Tbox (Terminological Box) :** le niveau générique (global) vrai dans tous les modèles et pour tous les individus [CHAA09] ; La TBox, ou base

terminologique, est un ensemble d'axiomes terminologiques, qui peuvent être de la forme C⊆D ou de la forme C≡ D, C et D étant deux concepts [ADRI07].

Exemple 2. Cet exemple d'écrit une Tbox:

Femelle ⊆ ⊤∩ ┐Mâle, Mâle ⊆ ⊤∩ ┐Femelle, Animal ≡ Mâle U Femelle

Humain ⊆ Animal, Femme ≡ Humain∩ Femelle, Homme ≡ Humain ∩┐Femelle.

2. **Niveau assertionnel Abox (Assertional Box):** fournit des instances des concepts et des rôles [CHAA09]. La ABox est un ensemble d'assertions qui peuvent être sous la forme C(a) ou r (à, b) où a et b sont deux instances, C est un concept et r est un rôle. Le premier type d'assertion correspond à une instanciation de concept, le second, à une instanciation de rôle [ADRI07].

   Exemple 03 : Cet exemple représente une Abox :

   Karim : Homme, Amina : Femme, épouse_de (Karim, Amina).

Exemple d'une base de connaissance écrite en LD, le concept " tarte aux pommes et aux noix dont toutes les pâtes sont feuilletées ou brisées" peut être représenté sous la forme suivante en LD [ADRI07] :

Tarte ∩∃ingrédient.Pomme ∩∃ingrédient.Noix∩ ∀pâte.(Feuilletée U Brisée).

Ce concept s'appuie sur la base de connaissances du Tableau 1.3. À titre d'exemple, l'axiome [Axl] indique qu'une Tarte est une Préparation Culinaire, l'axiome [Ax7] désigne que Pomme et Noix sont des concepts incompatibles (il n'existe pas d'objet qui soit à la fois une pomme et une noix) et l'axiome [Ax8] indique qu'une Tarte Sucrée est une Tarte ayant au moins un ingrédient qui soit un Produit Sucré.

(Axl) Tarte ⊆ PréparationCulinaire	(Ax7) Pomme ∩ Noix ⊆L
(Ax2) Dessert ⊆ PréparationCulinaire	(Ax8) TarteSucrée≡Tarte∩ ∃ ingrédient.ProduitSucrée
(Ax3) ProduitSucré ⊆ Produit	(Ax9) TarteSalée ≡ Tarte ∩┐TarteSucrée
(Ax4) Fruit ⊆ ProduitSucré	(Axl0) TarteSucrée ⊆ Dessert
(Ax5) Pomme ⊆ Fruit	
(Ax6) Noix ⊆ Fruit	

(a)TBox

(A1) (Tarter∩ ∃ ingrédient.Pomme)(tartel) (A3) ingrédient(tartel,noixl)

(A2) Noix(noixl)

(b)ABox

Tableau 1.3 Une base de connaissance écrite en *LD* (Selon Adrien Coulet [ADRI07])

4. Ontologie

4.1. Différentes notions du terme Ontologie

Guarino et Giaretta [GUAR95a] ont soulevé des préoccupations que le terme **«ontologie»** a été utilisé de manière incohérente. Ils ont trouvé au moins sept notions différentes attribuées à ce terme :

1. Ontologie en tant que discipline philosophique.
2. Ontologie en tant qu'un système conceptuel informel.
3. Ontologie en tant que sémantique formelle.
4. Ontologie comme une spécification d'une conceptualisation.
5. Ontologie comme une représentation d'un système conceptuel via une théorie logique :
    a. Caractérisée par des propriétés formelles.
    b. Caractérisée seulement par ses buts spécifiques.
6. L'Ontologie en tant que vocabulaire utilisé par une théorie logique.
7. L'Ontologie en tant que spécification (méta-niveau) d'une logique théorie·

Donc, nous pouvons dire que le mot « ontologie » provient du domaine de la philosophie, qui sert à désigner une théorie basée sur l'étude de l'être, d'une coté, et il peut aussi être interprété comme l'ensemble de ce qui existe avec ses relations, restrictions, axiomes et vocabulaires dans le domaine de l'Intelligence Artificiel.

La définition d'ontologie la plus connue dans le domaine de l'intelligence Artificielle (IA), est celle qui a été présentée par Gruber [GRUB93a], « une ontologie est une spécification explicite d'une conceptualisation ». [BORS97] a ajouté quelques précisions au concept de [GRUB93a] et définit une ontologie comme « une spécification formelle et explicite d'une conceptualisation partagée » : **Conceptualisation :** modèle abstrait d'un phénomène du monde pour lequel on a identifié les principaux concepts. **Explicite :** les définitions déclaratives de concepts utilisés et les contraintes quant à leurs usages sont explicitement définies. **Formelle :** l'ontologie doit être traduite dans un langage compréhensible par la machine. **Partagée :** l'ontologie doit être consensuelle et acceptée par un groupe.

Plus formellement une ontologie O est un système de symboles consistant en :

- Un ensemble C, de concepts, et un ensemble R de relations binaires (D, B), entre domaines et co-domaines (qui sont deux sous-ensembles de C) ; N.B. : Un concept représente un ensemble (fini ou infini).

- Une hiérarchie H, où les concepts et relations sont hiérarchiquement reliés par la relation de subsomption, i.e. une relation d'ordre partiel note $\subsetneq$ ou $C_1 \subseteq C_2$ signifie que C_1 est un sous-concept de C_2, et $r_1 \subseteq r_2$ signifie que r_1 est une sous-relation de r_2.
- Un ensemble d'axiomes A qui introduisent les concepts et les relations.

4.2. Les éléments constituant une ontologie

Une ontologie est constituée des éléments suivants [GÓME99] : **Les concepts (classes) :** sont utilisés dans un Large sens, ils peuvent être abstraits ou concrets, élémentaires ou composés, réels ou fictifs. Un concept peut aussi être la description d'une tâche, fonction, action, stratégie, processus, Raisonnement, etc. appartenant à un segment de la réalité, c'est-à-dire le domaine. **Les relations :** qui traduisent les associations existant entre les concepts présents dans ce domaine. **Les fonctions :** soit les cas particuliers de relations dans lesquelles un élément de la relation, le (x-ième), est défini en fonction des (x-1) éléments précédents. **Les axiomes :** qui constituent des affirmations, acceptées comme des vérités, concernant des abstractions traduites de ce domaine par l'ontologie. **Les instances (modèles) :** qui désignent la description en extension de l'ontologie, véhiculant les connaissances d'un domaine.

4.3. Différents types d'ontologies

Une classification des ontologies a été proposée selon leurs niveaux de généralités [GUAR98] : **Les ontologies de haut niveau** sont celles qui décrivent des concepts généraux (espace, temps, matière, objets, événements, actions, etc.) indépendants d'un problème ou d'un domaine d'application particulier. **Les ontologies de domaine** et **les ontologies de tâche,** expriment une conceptualisation qui les rend spécifiques à un domaine déterminé de la connaissance (comme la médecine ou les automobiles), ou une tâche, ou une activité générique (comme le diagnostic ou la vente), en spécialisant les concepts présentés dans les ontologies de haut niveau. **Les ontologies d'application** contiennent les définitions requises pour modéliser la connaissance dans une application, par exemple identifier les maladies du cœur, d'après une ontologie du domaine de la cardiologie. Ces concepts correspondent souvent aux rôles joués par des entités de domaine tout en exécutant une certaine activité, comme l'unité remplaçable ou le composant disponible.

On peut distinguer les ontologies selon le formalisme utilisé pour les exprimer : **Ontologie Informelle :** l'ontologie est exprimée en langage naturel. Cela peut permettre de rendre plus compréhensible l'ontologie pour l'utilisateur, mais cela peut rendre plus

difficile la vérification de l'absence de redondances ou de contradiction. **Ontologie Semi-informelle :** l'ontologie est exprimée dans une forme restreinte et structurée de la langue naturelle ; cela permet d'augmenter la clarté de l'ontologie tout en réduisant l'ambigüité. **Ontologie Semi-formelle :** l'ontologie est exprimée dans un langage artificiel définit formellement. **Ontologie Formelle :** l'ontologie est exprimée dans un langage artificiel disposant d'une sémantique formelle, permettant de prouver des propriétés de cette ontologie.

4.4. Les applications d'ontologies

Les applications d'ontologies ont été classifiées en quatre catégories principales, et chaque application peut s'insérer dans plus d'une de ces catégories [USCH99] : **L'autorité neutre :** une ontologie se développe dans une seule langue, qui est ensuite traduite pour des formats différents et utilisés dans des applications possédant des objectifs multiples. **Les ontologies comme spécification :** une ontologie est créée pour un domaine déterminé et doit fournir un vocabulaire visant à spécifier des besoins nécessaires à une ou plusieurs applications cibles. En fait, l'ontologie est utilisée comme une base de spécification et de développement de logiciel, qui permet ainsi sa réutilisation. **L'accès commun à l'information :** une ontologie est utilisée en vue de permettre à de multiples applications cibles, ou à des personnes, d'avoir accès à des sources hétérogènes d'informations, qui se trouvent exprimées dans un vocabulaire varié ou un format inaccessible. **La recherche basée sur des ontologies :** une ontologie est utilisée à des fins de recherche, dans un répertoire d'informations, selon des moyens souhaités, afin d'améliorer la précision et de réduire la perte inutile de temps durant cette recherche.

4.5. La construction d'une ontologie

4.5.1. Critères pour construire une ontologie

Le processus de construction d'une ontologie doit respecter certains principes de base qui permettent d'obtenir une ontologie susceptible de répondre aux objectifs de l'ontologie. Gruber [GRUB93], présente certains critères pour la construction d'un projet d'ontologie. **La clarté :** La définition d'un concept doit faire passer le sens voulu du terme, de manière aussi objective que possible (indépendante du contexte). **La Complétude :** Une définition exprimée par des conditions nécessaires et suffisantes est préférée à une définition partielle (définie seulement par une condition soit nécessaire ou bien suffisante). **La cohérence :** Une ontologie cohérente doit permettre des inférences conformes à ces définitions. **L'extensibilité :** Il doit être possible d'ajouter de nouveaux concepts sans avoir à toucher aux fondations de l'ontologie. **Engagements ontologiques minimaux :**

L'ontologie devrait spécifier le moins possible la signification de ses termes, donnant aux parties qui s'engagent dans cette ontologie la liberté de spécialiser et d'instancier l'ontologie comme elles le désirent. **Principe de distinction ontologique :** les classes dans une ontologie devraient être disjointes. Le critère utilisé pour isoler le noyau de propriétés considérées comme invariables pour une instance d'une classe est appelé le critère d'Identité. **Modularité :** Ce principe vise à minimiser les couplages entre les modules. **Réduire au minimum la distance sémantique** entre les concepts enfants de mêmes parents. Les concepts similaires sont groupés et représentés comme des sous-classes d'une classe, et devraient être définis en utilisant les mêmes primitives, considérant que les concepts qui sont moins similaires sont représentés plus loin dans la hiérarchie. **Normaliser :** il est préférable de normaliser les noms autant que possible.

4.5.2. Méthodologie de développement d'une ontologie

On entend par méthodologie, les procédures de travail, les étapes, qui décrivent le pourquoi et le comment de la conceptualisation puis de l'artefact construit. L'ingénierie ontologique ne propose à l'heure actuelle, aucune méthode normalisée ou méthodologie générale de construction d'ontologies, ce qui rend le processus d'élaboration des ontologies long et coûteux. Cependant certains auteurs ont proposé des méthodologies inspirées de leur expérience de construction d'ontologies [FOX92] et [USCH95]. Ces méthodologies proposent à travers un ensemble d'étapes, un cycle de développement d'ontologies qui peut être adopté lors de la construction d'une nouvelle ontologie.

a. La méthode Tove (TOrento Virtual Enterprise), cette méthode est basée sur l'expérience du développement de l'ontologie du projet TOVE (TOrento Virtual Enterprise) [FOX92]. Elle aboutit à la construction d'un modèle logique de connaissance. L'ontologie est développée selon les étapes suivantes :

- Identification de scénario (problèmes) dépendant d'une application.
- Formulation de questions informelles (basées sur les scénarios) auxquelles l'ontologie doit permettre de répondre.
- Spécification d'une terminologie à partir des termes apparaissant dans les Spécification formelle (en Knowledge Interchange Format (KIF)) des axiomes et des définitions pour les termes de la terminologie.
- Évaluation de la complétude de l'ontologie.

b. La méthode Entreprise (The enterprise ontology), Uschold et King [USCH98] qui ont proposé une méthode de construction d'ontologie basée sur l'expérience acquise lors du développement de l'ontologie d'Entreprise, *The enterprise ontology*.

Pour construire une ontologie conformément à l'approche d'Uschold et King, les étapes suivantes doivent être respectées :

- Identification des objectifs et du contexte de l'ontologie
- Construction de l'ontologie
- Capture de l'ontologie
- Codage de l'ontologie
- Intégration d'ontologies existantes
- Évaluation de l'ontologie
- Documentation de l'ontologie.

Nous allons utiliser, dans notre apport, quelques concepts et définitions cités dans l'ontologie d'entreprise proposée dans [USCH98].

c. D'autre part et selon Gomez [GÓME99], pour développer une ontologie il y a cinq étapes à suivre : **Étape de spécification :** qui consiste à identifier la raison d'être de l'ontologie, les objectifs et l'utilisation. **Étape de conceptualisation** : consiste à produire la description informelle des concepts avec leurs propriétés et les relations entre les concepts. **Étape d'intégration** : qui consiste à uniformiser l'ontologie développée pour l'intégrer à d'autres ontologies déjà existantes. **Étape d'implantation :** qui consiste à décrire d'une façon formelle des extraits de la conceptualisation. **Étape d'évaluation et de documentation :** ces étapes sont effectuées tout au long du processus de développement.

4.5.3. Langages informatiques pour construire et manipuler des ontologies

Dans le but de mettre au point un langage standardisé, le W3C[1] a créé en novembre 2001 un groupe de travail, WebOnt rassemblant les acteurs du domaine dont la DARPA (*Defense advanced Research Projects Agency*) qui avait mis au point le langage DAML+OIL basé sur XML[2] et RDF[3]. Le travail de ce groupe a abouti à la recommandation OWL en février 2004. OWL[4] définit donc une syntaxe RDF pour décrire et construire des vocabulaires pour créer des ontologies [XAVI05]. OWL peut être utilisé pour représenter explicitement les sens des termes des vocabulaires et les relations entre ces termes.

OWL vise également à rendre les ressources sur le Web aisément accessibles aux processus automatisés [FVHD04], d'une part en les structurant d'une façon compréhensible et standardisée, et d'autre part en leur ajoutant des méta-informations.

[1] http://www.w3.org/
[2] http://www.w3.org/XML/
[3] http://www.w3.org/RDF/
[4] http://www.w3.org/2004/OWL/

Pour cela, OWL a des moyens plus puissants pour exprimer la signification et la sémantique que XML, RDF, et RDF-S[5]. De plus, OWL tient compte de l'aspect diffus des sources de connaissances et permet à l'information d'être recueillie à partir de sources distribuées, notamment en permettant la mise en relation des ontologies et l'importation des informations provenant explicitement d'autres ontologies.

a. OWL et les autres langages

XML fournit une syntaxe pour des documents structurés, mais n'impose aucune contrainte sémantique à la signification des documents.

RDF est un modèle de données pour représenter les objets et les relations entre eux, fournissant une sémantique simple pour ce modèle qui peut être représenté dans une syntaxe XML.

RDF-Schéma est un langage de définition de vocabulaire pour la description de propriétés et de classes représentées par des ressources RDF. RDF-S permet de définir des graphes de triplets RDF, avec une sémantique de généralisation/hiérarchisation de ces propriétés et de ces classes.

OWL ajoute du vocabulaire pour la description des propriétés et des classes, des relations entre classes (par exemple disjointness), des cardinalités et des caractéristiques de propriétés (par exemple symmetry). OWL est développé comme une extension du vocabulaire de RDF et il est dérivé du langage d'ontologies DAML + OIL.

b. Les éléments de base de langage

OWL est composé de trois parties [FVHD04] : **Une classe,** c'est à dire un groupe d'individus partageant les mêmes caractéristiques. Les classes peuvent être organisées hiérarchiquement selon une taxonomie (classification). Les classes définies par l'utilisateur sont d'ailleurs toutes des enfants de la « super-classe » OWL : Thing. Peut être comparé à une table dans le domaine des bases de données relationnelles. **Une propriété** qui permet de définir des faits ou des relations entre ces classes. Il existe en **OWL** deux types de propriétés : *propriété d'*objet (owl:ObjectProperty) qui définit une propriété entre deux individus d'une classe ou de plusieurs classes, et une *propriété de type de données* (owl:DataTypeProperty), c'est à dire une relation entre une valeur ou donnée et un individu d'une classe, l'équivalent d'un champ d'une table dans une base de données relationnelles. Les propriétés peuvent aussi être organisées hiérarchiquement. **Une instance,** c'est à dire un individu d'une classe qui peut prendre les caractéristiques définies par les propriétés.

[5] http://www.w3.org/TR/rdf-schema/

c. Sous langages d'OWL

OWL a trois sous langages de plus en plus expressifs : OWL Lite, OWL DL, et OWL Full : **OWL Lite :** est destiné aux utilisateurs ayant principalement besoin d'une hiérarchie de classification et de contraintes simples. Il vise plus particulièrement les développeurs d'outils, lesquels souhaitent mettre en œuvre le langage OWL en commençant avec un ensemble de base relativement simple des fonctionnalités du langage. **OWL DL :** est destiné aux utilisateurs qui demandent une expressivité maximale tout en retenant la complétude du calcul (toutes les inférences sont garanties calculables) et la décidabilité (tous les calculs s'achèveront dans un intervalle de temps fini). Le langage OWL DL (DL étant les initiales pour Description Logic) a été conçu pour prendre en charge la logique descriptive et fournir un sous-ensemble du langage offrant les propriétés de calcul nécessaires aux systèmes de raisonnement. Ainsi, il vise plus particulièrement les développeurs de systèmes de raisonnement puissants qui utilisent les ontologies construites selon les restrictions demandées pour OWL DL. OWL DL utilise toutes les primitives fournies par OWL LITE. **OWL Full :** est destiné aux utilisateurs qui veulent une expressivité maximale et la liberté syntaxique de RDF sans garantie de calcul (par exemple, dans OWL Full, une classe peut se traiter simultanément comme une collection d'individus ou comme un individu à part entière). OWL Full rend alors disponibles des fonctionnalités qui peuvent servir à nombre de base de données et de systèmes de représentation des connaissances mais qui, pourtant, violent les contraintes des raisonneurs de logique descriptive.

Comme il existe d'autres langages tels que : KIF (Knowledge Interchange Format), RIF (Rule Interchange Format) et la F-logique, Gellish, etc.

4.5.4. Outils pour la manipulation des ontologies

Protégé[6] est éditeur d'ontologies développé à l'université de Stanford, qui offre une infrastructure permettant d'éditer des ontologies de domaines et de décrire des données sous forme de l'ontologie créée [MARI06].

Le programme Protégé est basé sur la technologie Java ; il peut être étendu pour créer des applications à base de connaissances. Protégé possède une interface graphique qui offre un ensemble de fonctionnalités qui permet d'éditer les ontologies. Des extensions permettent de sauvegarder les ontologies crées sous le format OWL-DL, RDF et RDFS.

Les programmes Racer[7], Jess[8] et Jena[9], sont des outils qui aident à développer des applications à base d'ontologie. Racer est un moteur d'inférence utilisé pour vérifier la

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

cohérence d'une ontologie ; il est basé sur les modèles de raisonnement logique. Jess et Jena sont basés sur la modélisation du raisonnement par règles de production. Jess fait un lien entre l'ontologie et les règles, parce qu'il sert à la modélisation du raisonnement de règles et Jena est une plateforme en Java qui a pour but de construire des applications permettant la manipulation d'ontologies décrites en RDF, RDFS et OWL. Jena s'adresse aux développeurs d'applications, parce qu'il permet de lire, modifier et sauvegarder des ontologies.

4.6. Mise en correspondance d'ontologies

La Correspondance ou le Mapping entre les entités de deux ontologies est nécessaire car :

- Il existe des ontologies pour de petites parties de l'univers.
- Il n'existe pas d'ontologie globale pour un domaine, donc il doit pouvoir exister plusieurs visions d'un même domaine.

Plusieurs technique et méthodes de mapping ont été proposées. Avant de citer quelques-unes, nous devons définir quelques termes à savoir **Correspondances ou Mappings, Appariement ou Matching, Les méthodes de comparaison ou matchers, Alignement d'ontologies**:

Correspondances ou Mappings ; les mappings sont des relations entre les éléments de deux représentations (ontologies, schémas de bases de données, etc.), indiquant une similarité relative selon une mesure donnée [KLEI01].

La correspondance formelle [ADRI07] ; Soit deux ontologies O et O', une correspondance M entre O et O' est un quintuplet < id, e, e', R, n > tel que

- id est un identifiant unique de l'élément de mapping,
- e et e' sont des entités de O et O' (des concepts ou des rôles par exemple),
- R est une relation (par exemple d'équivalence (≡) ; de généralisation (⊆) ; de spécialisation (⊇) ; de disjonction (⊥)),
- n est une mesure de confiance contenue dans une structure mathématique (typiquement dans l'intervalle [0,1].

Appariement ou Matching ; le matching d'ontologies est le processus de définition d'un ensemble de fonctions permettant de spécifier des « correspondances » entre termes [CHAN03, EMBL03].

[7] http://protege.stanford.edu/
[8] http://herzberg.ca.sandia.gov/jess/
[9] http://jena.sourceforge.net

Les méthodes de comparaison ou matchers ; un matcher est une fonction utilisée pour calculer la distance entre deux entités. Les matchers sont des fonctions qui peuvent être combinées dans le processus de matching.

Alignement d'ontologies ; l'alignement d'ontologies est le processus d'établissement de liens de correspondances entre deux ontologies originales.

L'alignement formelle [ADRI07] *:* Soit deux ontologies *O et O',* l'alignement A entre *e et* e_1 est :

- Un ensemble de correspondances entre *O et O',*
- Associées à une multiplicité *: 1-1, 1-*, etc.*
- Associées à des métadonnées additionnelles (une méthode, une date, des propriétés, etc.).

4.6.1. Les techniques de l'appariement (matching)

Pour faire correspondre les différentes ontologies, deux étapes sont nécessaires [ELBY09] :

- S'abstraire de la différence entre les langages d'ontologies utilisés (par exemple en traduisant les ontologies dans un même formalisme de représentation), puis ;
- Chercher les concepts équivalents à apparier en tenant compte des différences de conceptualisation, de description de cette conceptualisation et de terminologie.

Les différentes méthodes de comparaison (Matchers) utilisées dans le processus de matching sont organisées selon la classification ci-dessous, présentée dans la Figure 1.2 [ELBY09].

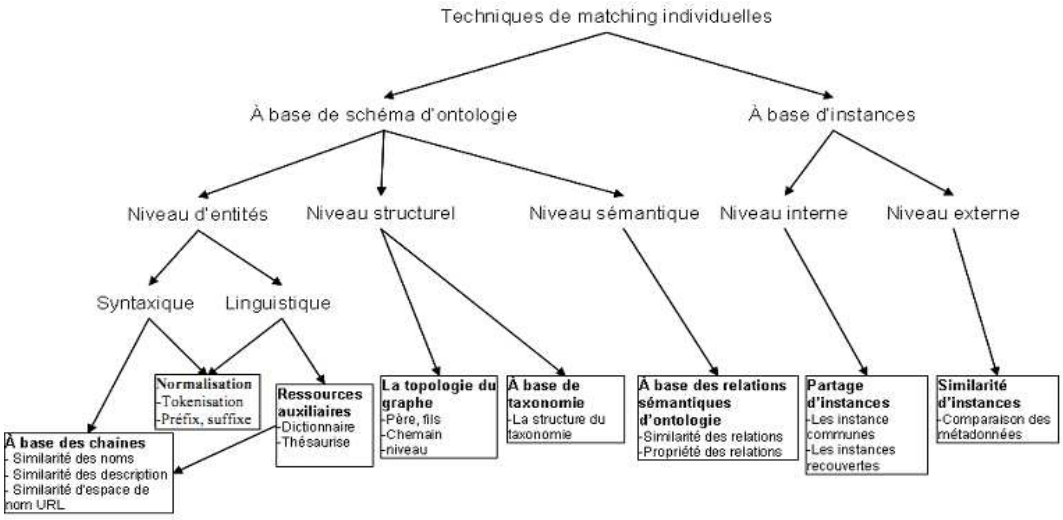

Figure 1.1 Classification des matchers (selon Elbyed [ELBY09])

Les différents matchers sont classés suivant leur entrée, c'est-à-dire le type d'entités à comparer et la méthode utilisée pour calculer la similarité. Par exemple, un des matchers

syntaxiques compare deux entités (ou deux concepts) suivant l'égalité de leur noms (par exemple, les labels des concepts). L'appariement entre deux entités ontologiques est soit basé sur le schéma d'ontologie ou basé sur les instances. Plusieurs niveaux, que Elbyed [ELBY09] explicite ci-dessous, sont alors considérés :

1) Matcher au niveau entité, le matcher au niveau entité compare les noms des entités en regardant le label ou l'identifiant d'un concept. Les noms peuvent être comparés, après une normalisation et parfois une concaténation avec le chemin depuis la racine. Nous trouvons à ce niveau deux approches de matching : l'approche syntaxique et l'approche lexicale, appelée aussi linguistique :

- **L'approche syntaxique** effectue la correspondance à travers les mesures de dis-similarité des chaînes de caractères (par exemple, la distance de Hamming).
- **L'approche lexicale** ou linguistique effectue la correspondance à travers les relations lexicales (par exemple, synonymie, hyponymie, etc.). C'est généralement à partir des labels que l'on fait appel à des ressources auxiliaires telles que les dictionnaires de synonymes et d'hyponymes comme WordNet 2, mais aussi à des thésaurus et des ressources sémantiques ainsi qu'à des dictionnaires spécifiques aux domaines étudiés comme (UMLS3).

2) Matcher au niveau structurel et sémantique, le matcher au niveau structurel et sémantique compare les structures internes des entités (par exemple, intervalle de valeur, cardinalité d'attributs, etc.).

Nous trouvons également à ce niveau de matching un autre type d'appariement qui utilise la structure externe des entités, c'est-à-dire qui compare les entités au sein de leur topologie (par exemple, en comparant leurs pères, fils, etc.).

3) Matcher à base d'instances, deux approches existent pour comparer les ontologies à partir des instances associées aux concepts d'ontologies :

- Soit les deux ontologies à comparer référencent les mêmes instances et dans ce cas le matcher génère une similarité entre les concepts qui partagent les mêmes instances.
- Soit les deux ontologies à comparer ne référencent pas les mêmes instances et dans ce cas le matcher fait des recherches par mots-clés dans les instances (souvent des documents ou autres fichiers). La similarité est ensuite calculée entre les instances à l'aide de ces mots-clés. Les classes (concepts) liées à ces instances sont ensuite « appariées ».

5. Conclusion

Dans ce chapitre, nous avons fait un rappel sur la gestion des connaissances dans le domaine de l'ingénierie des connaissances. La définition de la notion de gestion des connaissances nous a guidés vers la représentation des connaissances. Nous avons par la suite décrit, les différents niveaux de représentation ainsi que les divers langages et formalisme utilisées pour représenter ces connaissances. Le chapitre se termine par la description des ontologies et la présentation des méthodes et outils d'aide à leur construction. Aborder les objectifs, les composants, le cycle de développement et l'utilisation des ontologies, facilite la compréhension de leur apport dans la gestion des connaissances. Le langage utilisé pour représenter formellement une ontologie a un impact direct sur le niveau de formalisation de cette ontologie, les logiques de description sont particulièrement adaptées à la représentation des connaissances formalisées.

Nous avons dressé au cours de ce chapitre, l'intérêt de l'utilisation des ontologies comme méthodes de modélisation et réutilisation des connaissances dans le contexte de la gestion des connaissances de l'entreprise. De ce fait, notre choix s'est porté sur la représentation des connaissances à travers les ontologies.

Récemment, les processus d'affaires sont devenus une partie importante dans le domaine de l'ingénierie d'entreprise. Alors, il avait préconisé que nous devrons examiner la gestion des connaissances dans le processus d'affaires entier (qui est décrété pour manipuler une transaction de bout en bout), recherchant des façons d'optimiser le processus d'affaires dans sa totalité. Dans le deuxième chapitre nous allons présenter une description détaillée des processus d'affaires, des exemples et des concepts fondamentaux dans l'ingénierie d'entreprise, ainsi que les termes et méthodologies pour la description et l'amélioration de ces processus d'affaires.

Chapitre -02-
Processus d'affaires (Business Process)

1. Introduction

"The economists will have to revise their theories of value."
[Albert Einstein]

Dans une vision traditionnelle, une entreprise est considérée comme une organisation hiérarchique qui reflète à la fois la décomposition fonctionnelle de l'entreprise et la chaîne de commandement. Différents départements spécialisés dans des fonctions d'affaires spécifiques (par exemple, le marketing, la production ou la comptabilité), et au sein de chaque département, les sous-départements, les équipes et les individus se spécialisent dans des sous-fonctions. Le traitement d'une commande du client en général traverse les frontières des différents départements : Ventes (pour prendre l'ordre), Planification (pour planifier la fabrication du produit ou la reconstitution des stocks), production, l'expédition et la comptabilité.

Le développement de la technologie de l'information dans les années 1990 et sa capacité à surmonter la distance géographique a engendré un nouveau type de monde, et un environnement des affaires radicalement différent. Dans un marché en pleine évolution, les clients ont plus de liberté de la demande, du niveau de service qu'ils voulaient ; si une entreprise ne pouvait pas livrer dans les délais requis, une autre ailleurs peut probablement le faire. D'un autre côté, les chercheurs ont commencé à analyser ce qui pouvait s'adapter à cet environnement dynamique permettant ainsi d'offre des résultats plus rapides et qui satisfont de plus en plus les besoins du client. Ils avaient préconisé qu'on devrait regarder le processus d'affaires tout en entier qui est décrété pour manipuler une transaction de bout en bout, recherchant ainsi différentes façons d'optimiser ce processus dans sa totalité. Donc, les entreprises avaient besoin d'une nouvelle approche, d'où la naissance de l'approche des processus d'affaires.

Dans ce chapitre, nous allons, dans un premier temps, définir l'approche par processus dans les entreprises, ensuite nous donnerons une description générale sur les processus d'affaires et essayer d'éliminer l'ambigüité dans la définition d'un processus d'affaires, en donnant les différentes définitions existantes dans la littérature. Finalement, nous allons exposer les différents méthodes et techniques de modélisation et d'exécution d'un processus d'affaires.

2. Objectif de modélisation par processus

L'approche par processus est l'un des huit principes de management de la qualité identifiés dans la norme ISO 9000:2000, cette dernière précise que "*pour qu'un organisme fonctionne de manière efficace, il doit identifier et gérer de nombreuses activités corrélées. Toute activité utilisant des ressources et gérée de manière à permettre la transformation d'éléments d'entrée en éléments de sortie, peut être considérée comme un processus. L'élément de sortie d'un processus constitue souvent l'élément d'entrée du processus suivant. L'approche processus désigne l'application d'un système de processus au sein d'un organisme, ainsi que l'identification, les interactions et le management de ces processus*". [FRAN01]

Selon [RAPH03], l'approche processus est une réponse à un environnement en perpétuel mouvement. Elle constitue une réponse aux préoccupations permanentes d'accroissement de la compétitivité, de la souplesse, de la réactivité, tout en restant fidèle à la finalité des nouvelles normes ISO, la satisfaction du client :

- Elle doit permettre à l'entreprise de préparer les évolutions technologiques et réglementaires et d'anticiper les attentes du client.
- Elle renforce le rôle des collaborateurs au sein de l'entreprise et replace la notion de valeur ajoutée au cœur des préoccupations de la direction.
- Elle instaure une vision transversale orientée client.
- Elle clarifie les missions et les contributions des acteurs à la prestation délivrée au client.
- Elle attire l'attention de l'entreprise sur les objectifs et la valeur ajoutée des activités.
- Elle facilite la détection et la correction des dysfonctionnements.
- Elle prévient les risques en les identifiant en amont.
- Elle assure une meilleure compréhension des besoins et des contraintes de chacun par le partage de l'information et des ressources.

3. L'ingénierie d'entreprise

Caplinskas [CAPL03] considère une entreprise comme une ou plusieurs organisations partageant une certaine mission, buts et objectifs pour offrir une sortie sous forme d'un produit ou d'un service. Pour donner un sens aux processus d'affaires dans un système d'entreprise, il faut distinguer entre les deux termes «*Entreprise*» et «*Système d'Entreprise* » :

Une entreprise est une collection d'entités fonctionnelles (structures organisationnelles, employés, systèmes logiciels, machines, etc.) capables d'accomplir certaines tâches [CAPL03].

Un système d'entreprise est une sorte de système d'exploitation destiné à effectuer certaines affaires (nous interprétons le terme «Affaires», tout intérêt commercial, industriel, gouvernemental ou autre). Le système d'entreprise est exécuté par l'entreprise, plus exactement, par ses entités fonctionnelles [CAPL03].

3.1. Niveaux hiérarchique d'un système d'entreprise

Caplinskas [CAPL03] a proposé trois niveaux hiérarchiques du système d'entreprise : *Niveau des affaires, Niveau du traitement d'information et Niveau de la technologie d'information*. Il a défini ces niveaux comme suit :

1. *Le niveau des affaires* exploite un certain nombre de systèmes d'affaires. Tout système d'affaires est un sous-système du système d'entreprise. Il est implémenté comme un ensemble des processus d'affaires liés, qui sont réalisés par une entreprise pour atteindre certains objectifs de longue durée tout en produisant des produits particuliers et / ou des services fournis aux clients. *Un processus d'affaires*, à son tour, est mis en œuvre comme un ensemble d'activités inter reliées, qui manipulent (acquérir, créer, stocker, transformer, transporter, livrer, etc.) des instances d'entités d'affaires en particulier. Il est régie par un certain nombre de règles d'affaires, met en œuvre une fonctionnalité particulière et répond à un certain nombre de contraintes non-fonctionnelles assurant l'utilisation rationnelle de ressources, la fiabilité et d'autres propriétés préférables de ce processus.

2. *Le niveau du traitement d'information* exploite un certain nombre de systèmes d'information. Habituellement, chaque système d'information prend en charge un système d'affaires particulier, mais parfois il peut être partagé par plusieurs systèmes d'affaires. Le système d'information est implémenté comme un ensemble des processus d'information, effectués par les entités fonctionnelles d'entreprise, afin de fournir un certain nombre de services d'information nécessaires pour soutenir les processus d'affaires. *Un processus d'information* est une sorte de processus de soutient. Il est implémenté comme un ensemble d'activités inter reliées, qui manipulent (créer, copier, stocker, transformer, transporter, diffuser, etc.) des objets particuliers de l'information, et il est régie par un certain nombre de règles du traitement d'information. L'objet d'information est un élément des connaissances d'affaires.

3. Le *Niveau de la technologie d'information* gère un certain nombre de systèmes logiciels (applications, portails, données systèmes entrepôts, systèmes experts, messagerie, etc.) Tout logiciel est un composant d'un système d'information (SI) particulier (parfois, il peut être partagé par plus d'un système d'information (SI)) et il est utilisé pour mettre en œuvre, au moins partiellement, une entité fonctionnelle particulière ou un outil particulier, qui est utilisé pour manipuler des objets logiciels. Les objets logiciels sont des objets d'informations représentées sous forme numérique.

Pour être efficace tous les trois niveaux du système d'entreprise doivent être intégrés correctement.

4. Description d'un processus d'affaires

4.1. Définitions d'un processus d'affaires

Récemment, les processus d'affaires sont devenus une partie importante dans le domaine de l'ingénierie d'entreprise. Pour cette raison plusieurs définitions ont été proposées dans la littérature, nous présentons quelques-unes ci-après, avec des exemples qui les illustrent :

Raphaël SIBLER [RAPH03], présente une description d'un processus qui comporte trois composantes principales :

Ses caractéristiques : Un processus est défini par : Un intitulé, Une finalité, Un propriétaire ou pilote, dont le rôle est : de garantir l'efficacité du processus tout en s'assurant qu'il produit les résultats attendus par rapport aux objectifs fixés par la direction ; de veiller à la bonne utilisation des ressources allouées, Des données d'entrée, Des données de sortie visant à satisfaire les clients du processus, clairement identifiés, Des ressources :

- ~ Humaines, en termes de compétences nécessaires pour accomplir l'activité, Financières, Matérielles (équipements, logiciels,...), Informationnelles (expériences, connaissances, savoir-faire,...).

Sa vitalité : La vitalité d'un processus permet de suivre le dynamisme et l'amélioration des résultats du processus. Elle est définie par :

- Des objectifs provenant de l'identification des besoins et des attentes du client, ainsi que la mise en place des objectives qualités définis au préalable par la direction. La présence d'objectifs a pour but d'améliorer les performances du processus. Ils sont définis par une action, une échéance et des critères d'appréciation des résultats.

- Des indicateurs permettant de mesurer l'atteinte des objectifs. Les indicateurs doivent être pertinents, c'est-à-dire en cohérence avec les objectifs et la finalité du processus. En règle générale, ils s'appuient sur les données de sortie.

Sa représentation : Afin d'assurer la compréhension du processus, il est préférable de privilégier les modes de représentation graphique (logigrammes,...) car ils permettent une compréhension simple et synthétique du processus. Les représentations peuvent être :

- Macroscopiques : visualisation des étapes du processus,
- Détaillées : visualisation des tâches et / ou jalons nécessaires.

Dans bien des cas, l'analyse du processus issu de la norme ISO 9000 version 94 donnait lieu à une description statique des tâches, répondant aux questions qui fait quoi ? et comment ? [RAPH03].

Tandis que **Bill Curtis** [CURT92], défini un processus comme un ensemble partiellement ordonné des tâches ou des mesures prises pour atteindre un objectif spécifique**.**

Davenport [THOM93], décrit un processus d'affaires comme :

"Un ensemble structuré et mesuré d'activités conçues pour produire un résultat spécifique pour un client ou un marché particulier. Il implique une emphase forte sur la façon dont le travail est effectué dans une organisation,…. Un processus est ainsi un ordre spécifique des activités professionnelles à travers le temps et l'espace, avec un commencement et une fin, et des entrées et des sorties bien définies : une structure pour l'action. … Adopter une approche de processus implique adopté le point de vue du client. Les processus sont la structure par laquelle une organisation fait ce qui est nécessaire pour produire la valeur pour ses clients."

On peut considérer la définition de **Hammer & Champy's** [MICH93] comme un sous ensemble de celle de Davenport [THOM93]. Ils définissent un processus comme : " Une collection d'activités qui prend un ou plusieurs genres d'entrée et crée un résultat qui est précieux pour le client."

Pour sa part, **Johansson** [HENR93] considère un processus comme :

"Un ensemble d'activités liées qui prennent une entrée et la transforment pour créer une production. Idéalement, la transformation qui arrive dans le processus doit ajouter la valeur à l'entrée et créer une production qui est plus utile et efficace au destinataire en amont ou en aval."

Rummler & Brache [RUMM95], Emploient une définition qui englobe clairement un centre sur les clients externes de l'organisation, en l'exposant :

"Un processus d'affaire est une série d'étapes conçues pour produire un produit ou un service…. Quelques processus aboutissent à un produit ou un service qui est reçu par le client externe d'une organisation. Nous appelons ces processus *primaires*. D'autres processus produisent les produits qui sont invisibles au client externe, mais essentiels à la gestion efficace des affaires. Nous appelons ces processus *d'appui*."

Jay Cousins and Tony Stewart, RivCom Ltd [JAYC02], en parlant du concept de processus d'affaires, ils définissent un processus d'affaires comme est un jeu des activités d'affaires liées logiquement, qui se combinent pour livrer quelque chose de valeur (par exemple des produits, des marchandises, des services ou l'information) à un client.

Dans tutoriel UML [SPAR04], un processus d'affaire : A un But, A des entrées spécifiques, A des productions spécifiques, Ressources d'utilisations, A quelques activités qui sont exécutées dans quelque ordre, Peut affecter plus qu'une unité organisationnelle. Impact horizontal organisationnel, Crée la valeur de quelque sorte pour le client. Ce dernier peut être interne ou externe. La Figure 2.1 ci après représente cette définition :

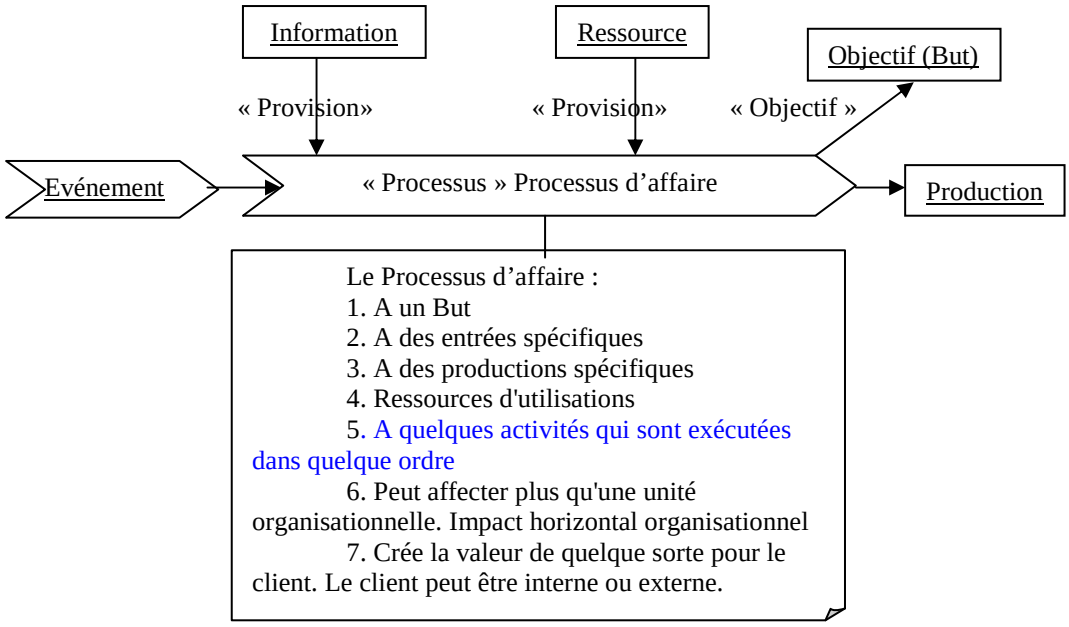

Figure 2.1 Définition du Processus d'affaires dans leur environnement (selon Sparx [SPAR04])

Pour sa part, Milli [HAFE04] a essayé de proposer une définition qui englobe certaines définitions déjà proposées : « Les activités d'un processus métier (processus d'affaires) sont réalisées par des acteurs jouant des rôles particuliers, en consommant des ressources et la production d'autres. Les activités peuvent être déclenchées par des événements et peuvent, à leur tour, générer d'autres événements. Les activités d'un processus peuvent être liées par des dépendances de ressources (producteur-consommateur

dépendances) ou des dépendances de contrôle (une activité déclenchant une autre). Les acteurs opèrent dans le cadre de frontières organisationnelles. Les organismes exercent des fonctions métier spécifiques. Les rôles peuvent fonctions support. La Figure 2.2 montre un diagramme UML qui représente un méta modèle de ce qu'est un processus.

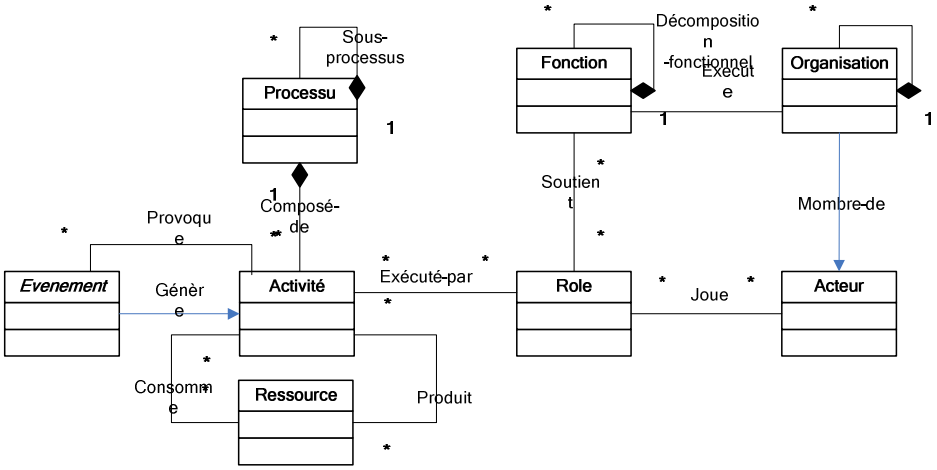

Figure 2.2 Méta modèle qui définit le processus d'affaires dans leur environnement (Selon Hafedh Milli [HAFE04])

Dans le domaine des Workflows, Hollingsworth [HOLL99] a aussi sa contribution, sur la définition du processus d'affaires et leur utilisation, présenté avec les points suivants :

- « Un processus d'affaire est considéré donc comme un ensemble d'une ou plusieurs procédures liées ou des activités qui, collectivement, réalisent un objectif d'affaires ou un but politique, généralement dans le contexte d'une structure organisationnelle définissant les rôles et les relations fonctionnels. »
- Un processus d'affaires est généralement associé à des objectifs opérationnels et des relations d'affaires, par exemple dans une assurance, processus de réclamation, ou processus de Développement de l'Ingénierie. Un processus peut être entièrement contenu dans une seule unité organisationnelle ou peut s'étendre sur plusieurs organismes différents, dans une relation client-fournisseur.
- Un processus d'affaires définit les conditions de déclenchement de son initiation dans chaque nouvelle instance (par exemple l'arrivée d'un sinistre) et défini ses produits à son achèvement.
- Un processus d'affaires peut impliquer des interactions formelles ou informelles entre les participants ; sa durée peut également varier largement.
- Un processus d'affaires peut concerner des activités automatisées, capable de la gestion du workflow, et / ou des activités manuelles, qui se situent en dehors de la gestion de workflow.

Parmi les promoteurs dans ce domaine, ***Van der Alste*** [AALS02] déclare que le processus consiste en un certain nombre de tâches qui doivent être réalisées et un ensemble de conditions qui déterminent l'ordre des tâches. Toujours selon Alste [AALS02], un processus peut également être appelé une procédure. Une tâche est une unité logique de travail qui est réalisé comme un tout unique par une seule ressource. Une ressource est le nom générique pour une personne, une machine ou un groupe de personnes ou de machines qui peuvent exécuter des tâches spécifiques. Cela ne signifie pas toujours que la ressource effectue nécessairement la tâche indépendamment, mais qu'elle en est responsable.

Alste [AALS02] a donné cet exemple qui est illustré dans la Figure 2.3, dans lequel il a examiné comment une compagnie d'assurance traite une réclamation. Il a identifié les tâches de ce processus comme suit :

1) L'enregistrement de la réception de la réclamation ;
2) Établir le type de réclamation (par exemple, incendie, automobile, voyage, professionnel) ;
3) Vérifier la politique du client, pour confirmer qu'il couvre vraiment en principe ce qui a été revendiqué pour ; c'est le cas dans la couverture principe de ce qui a été réclamé pour ;
4) Vérifier la prime, pour confirmer que les paiements sont à jour ;
5) Rejeter, si la tâche 3 ou 4 a un résultat négatif ;
6) Produire une lettre de refus ;
7) Estimation de montant à payer, basée sur les détails de réclamation ;
8) Nomination d'un assesseur pour étudier les circonstances des dégâts et d'établir sa valeur ;
9) Examen des mesures d'urgence afin de limiter d'autres dégâts ou de relever la détresse ;
10) Prestation de mesures d'urgence si elle est approuvée dans le cadre de la tâche 8 ;
11) Établissement ou révision du montant à payer et offre au client ;
12) Enregistrement de la réaction du client : l'acceptation ou l’objection ;
13) Évaluation de l'objection et la décision de réviser (tâche 11) ou engager une procédure légale (tâche 14) ;
14) Procédure légale ;
15) Paiement de réclamation et
16) Fermeture de réclamation : dépôt.

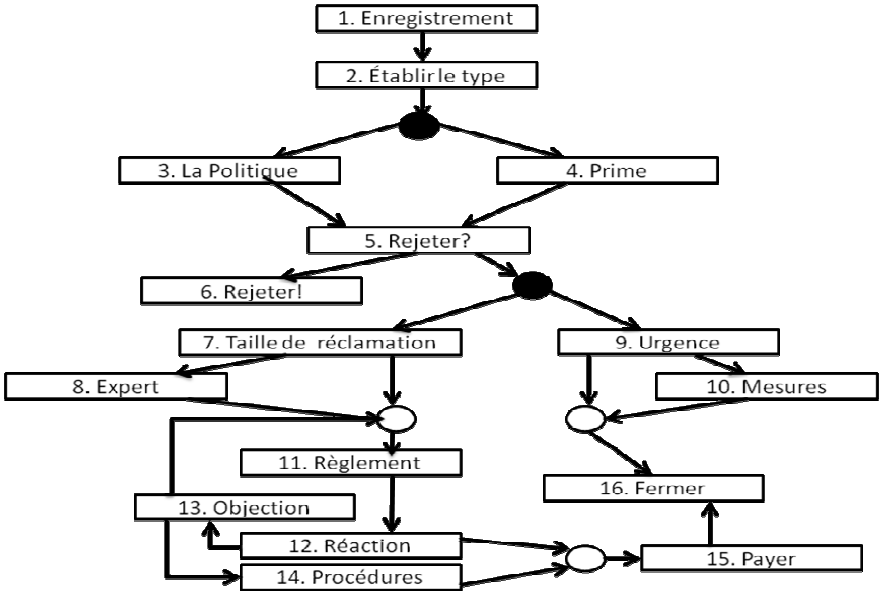

Figure 2.3 Exemple de processus proposé par *Alste* (qui examine comment une compagnie d'assurance traite une réclamation (Selon *Alste*[AALS02])

Dans cet exemple et selon *Alste* [AALS02] plusieurs remarques peuvent être faites :

- Deux ou plusieurs tâches qui doivent être effectuées dans un ordre strict sont appelées une séquence. Parfois, un choix entre deux ou plusieurs tâches peuvent être faites, ce que nous appelons une sélection. Il ya aussi des tâches qui peuvent être effectuées en parallèle, par exemple vérifier la politique et le contrôle des primes. Ces tâches doivent toutes les deux être achevées avant que la tâche "Rejet" ne puisse commencer, c'est ce qu'on appelle la synchronisation.
- Cet exemple d'un processus inclut également l'itération, à savoir l'évaluation répétée d'une objection ou la révision du montant qui doit être payé.
- Les cercles indiquent où des workflows particuliers se jointent ou se séparent. Les cercles gris ont plusieurs tâches antécédentes et une seule tâche ultérieure. Ils indiquent que seulement une des tâches antécédentes doit être effectuée pour continuer. Les cercles noirs ont une seule tâche antécédente et plusieurs tâches ultérieures. Ils montrent que toutes les tâches ultérieures doivent être exécutées (Les cercles peuvent être considérés comme des tâches) [AALS02].
- Donc, nous pouvons identifier quatre différents mécanismes de base dans les structures de processus : séquence, sélection, parallélisme, et l'itération.

Définition globale

Bien qu'il y ait plusieurs définitions du processus d'affaires, ou bien processus métier comme il avait mentionné dans certaines traductions, N. R. JENNINGS [JENN00] a

déclaré qu'un processus d'affaires peut être divisé en un certain nombre de composants constitutifs, qui sont illustrés dans la Figure 2.4 :

Premièrement, il doit y avoir une définition des processus d'affaires (branche gauche de la Figure 2.4). Il décrit, dans un langage de spécification, les activités qui doivent être effectuées, les participants qui pourraient ou devraient les exécuter, et les interdépendances qui existent entre eux. Les langages de spécification doivent fournir un ensemble de concepts utiles pour les processus décrivant leurs tâches, les dépendances entre les tâches et les rôles requis qui peuvent accomplir les tâches spécifiées [JENN00]. Les activités dans la description du processus peuvent être automatisées ou impliquent des êtres humains qui interagissent avec les ordinateurs.

Deuxièmement, le processus d'affaires doit être exécuté et géré (branche droite de la Figure 2.4). Un système logiciel, doit être conçu, capable d'assurer que la description du processus est réalisée dans la pratique. Ce système doit permettre aux humains et les activités manuelles pour être affectés de façon appropriée ; fournir un accès à des outils logiciels (par exemple, bases de données, tableurs, logiciels de conception, etc.) requises pour effectuer les tâches et assurer que les dépendances entre les tâches sont satisfaisantes [JENN00].

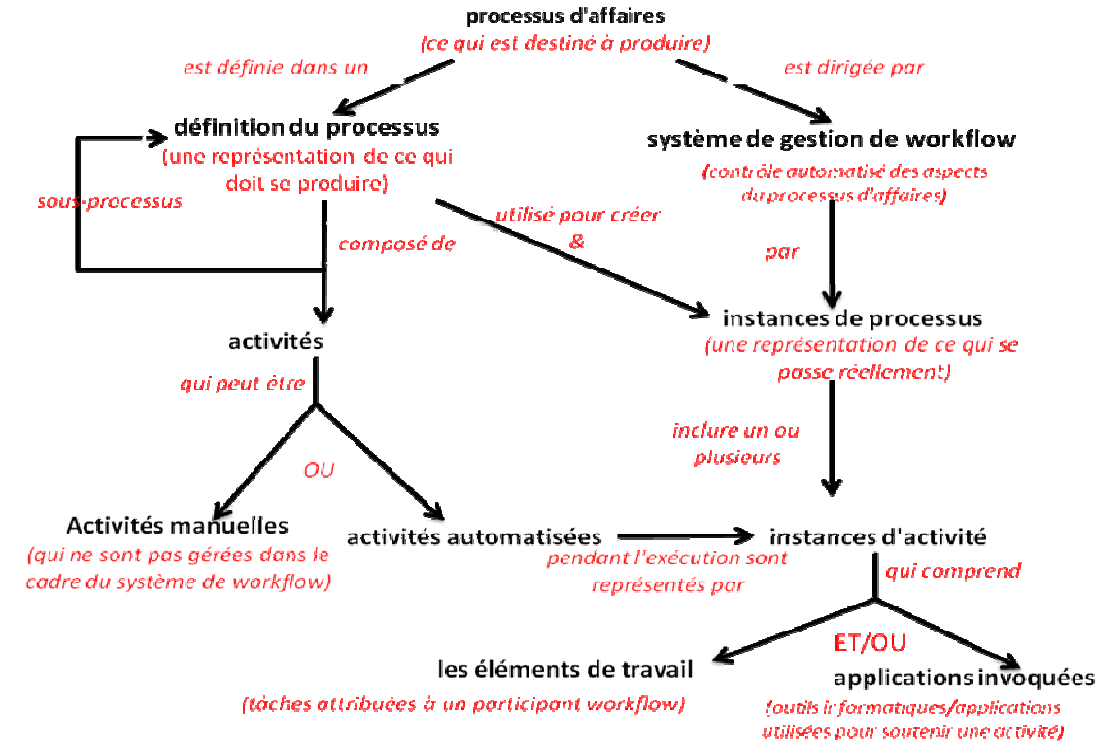

Figure 2.4 Composantes d'un processus métier (selon David Hollingsworth **[**HOLL**99])**

4.2. Typologies de processus

Les normes ISO 9000:2000 [RAPH03] ont proposé une typologie de processus d'affaires, selon la qualité, constituée de trois classes :

- **Les processus de "réalisation"** qui concernent directement la réalisation du produit de l'étape de conception jusqu'à la commercialisation.
- **Les processus "supports"** qui assurent le bon fonctionnement des processus de réalisation en leur fournissant les ressources nécessaires (maintenance, ressources humaines, contrôle de gestion,...).
- **Les processus de "pilotage" (Management)** qui agissent sur le fonctionnement de l'entreprise et sa dynamique d'amélioration. Ils orientent et assurent la cohérence des processus de réalisation et de support.

Cependant Medina Mora [MEDI93] a classé les processus dans une organisation en :

- **Processus matériels,** l'assemblage des composants physiques ou la livraison des produits physiques.
- **Processus d'information,** liées aux tâches automatisées et partiellement automatisées qui créent, traitent, gèrent et fournissent des informations.
- **Processus d'affaires**, les descriptions de marché centrées sur les activités d'une organisation, mis en œuvre comme processus d'information et / ou processus matériels.

Pour autant, il n'existe pas de catalogue de processus. Il appartient à chaque entreprise, en fonction de sa stratégie, son activité et ses particularités, de définir ses propres processus.

5. Termes et méthodologies pour la modélisation ou l'exécution du processus d'affaires

5.1. L'ingénierie des processus d'affaires

L'ingénierie des processus d'affaires est comprise comme la tentative de définir les processus d'affaires selon les exigences logiques et organisationnelles [VOLK94].

5.2. La réingénierie des processus d'affaires

La réingénierie des processus d'affaires est similaire à l'ingénierie des processus métier, mais elle s'intéresse davantage aux processus d'affaires déjà existants et leur mise en œuvre effective. La réingénierie des processus d'affaires est comprise comme la réinvention des processus d'affaires. Réinvention -au lieu de l'amélioration- signifie que les

structures de base du processus ne sont pas nécessairement laissées intactes, mais peuvent être changées suite à de nouvelles technologies ou les opportunités des affaires sont changées [VOLK94].

5.3. La gestion des processus d'affaires (Business Process Management (BPM))

Le BPM est la discipline qui fournit l'ensemble des méthodes, technologies et outils destinés à améliorer l'efficacité, la traçabilité et l'agilité des processus métiers au sein desquels collaborent des systèmes, des logiciels, des personnes et aussi des clients, des fournisseurs et des partenaires…. La BPM traite du cycle d'ingénierie des processus [XAVI09].

D'après cette définition, nous constatons que BPM est une approche qui permet la gestion du cycle de vie d'un processus d'affaires, depuis sa création jusqu'à son exécution.

5.3.1. Cycle de vie de BPM

Le cycle de vie du BPM (Figure 2.5) impose une vue itérative permettant de veiller à ce que les processus puissent évoluer et être optimisés dans des cycles courts. Chaque auteur propose les différentes étapes du cycle de vie de BPM qu'il voit nécessaires. Jean-Noël Gillot [JEAN08] propose quatre étapes qui sont :

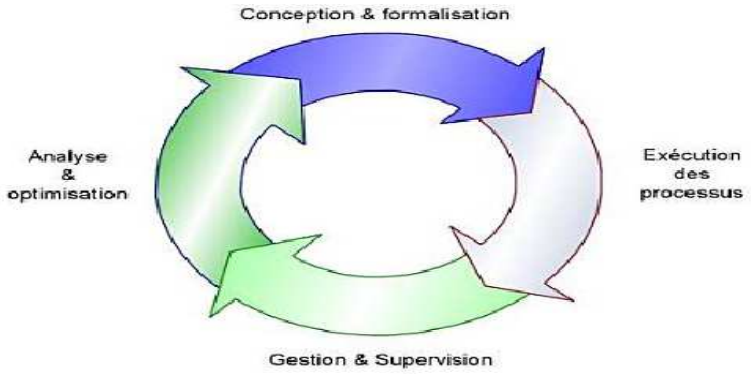

Figure 2.5 Cycle de vie de BPM (Selon Jean-Noël Gillot [JEAN08])

1. **Conception et formalisation :** Les processus sont en général modélisés à l'aide d'un outil graphique dont sa capacité doit permettre la réutilisabilité des modèles dans un outil de BPM. Cela permet l'utilisation de l'outil BPM par les experts métiers à partir de vues qui leur soient propres et compréhensibles pour modéliser. La phase de conception doit permettre de modéliser les processus, définir les activités et règles métiers associées, définir les éléments de coût du processus pour chacune des activités et simuler le processus pour effectuer un premier niveau d'optimisation. À ce niveau, la documentation du processus est très importante, car elle permet d'échanger avec tous

les acteurs pour rechercher la meilleure version du processus à implémenter dans l'outil.

2. **Exécution des processus :** Les processus sont intégrés au système d'information en faisant appel à des outils permettant la mise en œuvre de l'automatisation des processus, les fonctions de Workflow, l'intégration des applications et des règles métiers. Les processus seront alors testés et déployés dans l'environnement d'exécution.
3. **Gestion et supervision :** La partie gestion englobe l'administration et la maintenance corrective des processus. La partie supervision qui est de deux types : la supervision du déroulement des processus et la supervision de la performance des processus qui mettent en œuvre des tableaux de bord représentant les indicateurs de performance.
4. **Analyse et optimisation :** Après un certain temps de fonctionnement des processus, les données collectées vont pouvoir servir à analyser leur fonctionnement. L'analyse portera sur la performance de chacune des activités du processus, mais également sur le délai de traitement de chacune de celle-ci et le délai de traitement global du processus. L'analyse effectuée, on pourra dresser un plan d'investigation qui permettra de trouver les causes, qu'elles soient conjoncturelles, humaines ou systèmes. Un plan d'action est alors dressé pour permettre l'amélioration du processus

5.3.2. La modélisation d'un processus d'affaires

Le regroupement des informations collectées reflète l'organisation et son mode de fonctionnement. Le nombre important d'intervenants, de tâches, d'interactions et d'échange de message rendent éventuellement difficile la compréhension au premier regard des informations collectées. La maîtrise des processus métiers nécessite une simplification de cette réalité en structurant l'information sous forme de modèles employés dans toutes les étapes du cycle de vie des processus métiers [BRIO08].

5.3.2.1 Le modèle de processus métiers, un modèle représente une abstraction de la réalité reflétant une certaine façon de réduire la complexité naturelle d'un phénomène. Un modèle peut prendre différentes formes comme les feuilles de calcul et les diagrammes. Il est important de distinguer la notion de modèle et celle de diagramme. Le modèle reprend l'ensemble de l'information du phénomène étudié offrant la possibilité de simuler une situation réelle. Un diagramme constitue une représentation graphique du phénomène facilitant sa compréhension et sa communication. La cartographie des processus métiers regroupe l'ensemble des diagrammes de processus métiers en différents niveaux d'information [BRIO08].

5.3.2.2 Représentation graphique Des Processus Métiers, il n'existe pas de règles absolues de représentation et de modélisation des processus métiers de l'organisation. Cependant, il existe plusieurs diagrammes et modèles fondamentaux illustrant la situation courante d'une organisation [BRIO08] :

- Le diagramme de processus représentant les séquences logiques et chronologiques des tâches des processus métiers.
- Le diagramme de la chaîne de valeur représente une perspective macroscopique de la création de valeur dans l'organisation.
- L'organigramme représente la structure organisationnelle hiérarchisée de l'entreprise.
- Le tableau des règles métiers. La représentation des règles métiers globales complète les enchaînements des tâches décrits dans les diagrammes des processus métiers.

Il est important, dès le début de la modélisation, d'organiser les modèles en plusieurs niveaux ; d'un niveau élevé global et simple, vers des niveaux plus détaillés. Comme ils sont illustrés dans la Figure 2.6.

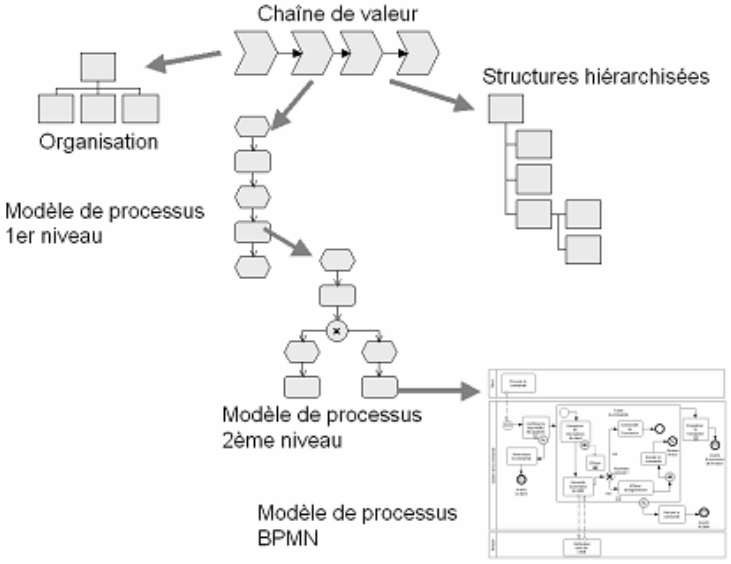

Figure 2.6 L'organisation des diagrammes et modèles de processus métiers (Selon Briol [BRIO08])

5.3.2.3 Les objectifs de la modélisation d'un processus d'affaires, la modélisation des processus d'affaires est utile pour trois raisons essentielles, qui peuvent à leur tour soutiennent plusieurs objectifs d'affaires [HAFE04] :

1) *Décrire un processus :* nous allons modéliser un processus pour être en mesure de le décrire. Ces descriptions, pour les humains doivent être compréhensibles, pour les machines, doivent être formalisées.
2) *Analyse d'un processus :* tout simplement, l'analyse des processus consiste à évaluer les propriétés d'un processus. La réingénierie et l'amélioration des processus s'appuient

sur l'analyse des processus existants pour identifier les étapes redondantes ou sous-optimale.

3) *Adopter un processus :* nous pouvons adopter un processus à des fins de simulation ou de fournir un certain niveau de soutien à l'exécution des processus. Selon le langage, ce soutien peut prendre différentes formes : la réaction aux événements déclenchés par l'exécution du processus, vérifiant que les contraintes spécifiques sont satisfaites, conduisant l'exécution du processus.

5.3.2.4 Les vues de la modélisation d'un processus d'affaires, comme les processus d'affaires sont complexes, les concepteurs fournissent des vues de modélisation différentes, chacune se concentrant sur un aspect du processus. Nous identifions quatre vues, résumées ci-dessous [HAFE04] :

1) *La vue fonctionnelle :* cette vue présente les dépendances fonctionnelles entre les éléments de processus (activités, sous-processus, etc.). Ces dépendances sont typiquement incorporées dans le fait que quelques éléments de processus consomment (ou le besoin) des données (ou des ressources) produites par d'autres. Les notations typiques utilisées dans la vue fonctionnelle incluent des diagrammes de flux de données.

2) *La vue dynamique :* la vue dynamique fournit l'ordonnancement et le contrôle des informations d'un processus, c'est-à-dire, quand certaines activités sont exécutées (synchronisation, conditions préalables) et comment elles sont exécutées (par exemple, en décrivant la logique de contrôle).

3) *La vue informationnelle :* inclut la description des entités qui sont produites, consommées ou autrement manipulées par le processus. Ces entités incluent des données pures, artefacts, produits etc.

4) *La vue organisationnelle :* décrit qui va exécuter chaque tâche ou fonction et dans quel endroit dans l'organisation il va l'exécuter (fonctionnellement et physiquement).

Une autre distinction intéressante est celle, présenté plus spécifiquement dans le contexte de services Web, entre la vue d'orchestration et la vue de chorégraphie. BPDM [BPDM08] supporte deux vues fondamentales et complémentaires du processus «orchestration» et «Chorégraphie» :

- *L'orchestration se* réfère à un processus spécifique, à savoir, comment ce processus doit être exécuté, uniquement à partir de la perspective de ce processus [HAFE04]. Les concepts d'orchestration dans BPDM [BPDM08] sont représentés à travers des séquences d'activités »qui produisent des résultats avec des branchements et de la

synchronisation. L'orchestration est généralement représentée comme organigrammes, diagrammes d'activités, ou des notations similaires d'une tâche ou une activité après une autre. L'orchestration de processus décrit ce qui se passe et quand, afin de mieux gérer un processus sous l'autorité d'une entité [BPDM08].

- *La chorégraphie* se réfère à l'échange de messages entre les diverses parties et sources, à savoir, l'orchestration suit la trace des séquences de message dans les parties impliquées aux diverses transactions [HAFE04]. Dans BPDM [BPDM08], la chorégraphie décrit comment les entités semi-indépendantes collaborant travaillent ensemble dans un processus, dont chacun peut avoir ses propres processus internes. La chorégraphie capture les interactions des rôles avec des responsabilités bien définies au sein d'un processus donné. La chorégraphie est la base de l'architecture orientée services (SOA) [ERL05].

Tandis que quelques langages peuvent seulement exprimer une de ces vues, généralement l'orchestration, d'autres (par exemple, BPEL4WS) [BPEL07] peuvent exprimer toutes les deux [HAFE04].

5.3.2.5 Classification des langages de la modélisation de processus d'affaires, ces langages on été subdivisé selon les différentes vues de processus d'affaires (dynamique, fonctionnelle, informationnelle, organisationnelle) comme suit [HAFE04]:

1) *Les langages classiques de modélisation de processus :* Ces langages ne sont généralement pas formels, mais peuvent se prêter à diverses analyses informelles ou heuristiques. Les langages dans cette catégorie comprennent l'IDEF, les réseaux de Pétri, chaînes de processus de l'évènement (CPE), Diagrammes d'activité de rôle**,** Resource-Event-Agent (REA), et récemment Business Process Modeling Language.

2) *Les langages modélisant les Workflows:* le système de gestion de workflow est un système informatique qui gère un processus d'affaires en assignant les activités du processus aux bonnes ressources, en déplaçant les éléments de travail (par exemple, documents, commandes, etc.) d'une étape de traitement vers une autre, et en suivant le progrès du processus [WfMC02]. Les langages de modélisation des workflows sont des langages de script, décrivent les Workflows, afin qu'ils puissent être supportés par un système de gestion des Workflows. Ces langages, pour la plupart, sont formels et exécutables. Nous parlerons, principalement, du Langage de Description du Processus de Workflow (WPDL) [HOLL99].

3) *Les langages d'intégration de processus :* l'apparition du commerce électronique interentreprises (B2B) a stimulé l'intérêt dans les langages modélisant le processus pour

buts d'intégrer les processus de deux ou plusieurs partenaires d'affaires. Ces langages se concentrent typiquement sur des mécanismes d'intégration en termes d'abstraction, technologies indépendantes, interfaces de programmation et des formats d'échange de données. Les langages dans cette catégorie peuvent aussi capturer des différents niveaux de la sémantique des processus sous-jacents. Exemple de ces langages : RosettaNet, ebXML [ebXML03] et BPEL4WS [BPEL07].

4) *Les langages orientés objet :* la modélisation orientée objet a été une façon naturelle de représenter le monde réel dans une voie où, tant les experts de domaine que les experts en informatique, peuvent le représenter [HAFE04]. UML 2 [OMG01] est parmi ces langages [HAFE04].

5.4. Concepts de base sur le système d'exécution du processus métier

Un système d'exécution de processus métier a été défini comme est une solution logicielle d'exécution et de supervision des processus métiers. Il regroupe généralement plusieurs composants [BRIO08] : *Un outil de définition et de modélisation des processus métiers, Un moteur d'exécution des enchaînements des opérations entre individus communément appelé moteur de « Workflow », Un moteur de règles métiers, Un service d'intégration d'applications et d'autres systèmes informatiques existants en employant des protocoles standards de communication, Un composant de contrôle et de supervision des processus métiers, Un outil d'analyse et de génération automatique de rapports, Un outil de mesure des performances.*

5.4.1. Le système de gestion des processus métiers

Le système de gestion des processus métiers ou « Business Process Management System – BPMS » en anglais, est une solution logicielle d'élaboration, de déploiement, d'exécution, de supervision et de gestion des processus métiers. Il a confirmé qu'un système de gestion des processus métiers reprend généralement les fonctionnalités suivantes [BRIO08] : *La modélisation des processus métiers, Le développement collaboratif entre intervenants, La documentation des processus métiers, La simulation des processus métiers, L'intégration des applications internes ou externes à l'organisation, l'automatisation des processus métiers, La collaboration entre entreprises partenaire, Le déploiement des processus métiers, L'analyse de l'exécution des processus métiers, La production de tableaux de bord et de rapports, la gestion des connaissances de l'organisation.*

5.4.2. Notions sur les Workflow

5.4.2.1 Définition de Workflow, c'est l'automatisation d'un processus métier, en tout ou en partie, au cours de laquelle des documents, des informations ou des tâches sont transmises d'un participant à un autre, selon un ensemble de règles procédurales [HOLL99]. Il fournit en outre, à chacun des acteurs, les informations nécessaires pour la réalisation de sa tâche [BRIO08].

5.4.2.2 Typologie de Workflow et modèle de référence, on distingue généralement deux types de Workflow [HOLL99] :

Le Workflow procédural (aussi appelé Workflow de production ou Workflow directif) correspondant à des processus métiers connus de l'entreprise et faisant l'objet de procédures préétablies : le cheminement du Workflow est plus ou moins figé ; valable au début des années 90. **Le Workflow ad hoc** basé sur un modèle collaboratif dans lequel les acteurs interviennent dans la décision du cheminement : le cheminement du Workflow est dynamique.

5.4.2.3 Systèmes de gestion des workflows, un système qui définit, crée et gère l'exécution des workflows grâce à l'utilisation du logiciel, fonctionnant sur un ou plusieurs moteurs de workflow, qui est capable d'interpréter la définition de processus, d'interagir avec les participants de workflow et, le cas échéant, invoquer l'utilisation d'outils informatiques et des applications [HOLL99].

Un moteur de Workflow est l'outil permettant de créer, gérer et exécuter des instances de Workflow. Ce type d'outil permet ainsi de formaliser les règles métier de l'entreprise afin d'automatiser la prise de décision, c'est-à-dire la branche du Workflow à choisir, en fonction du contexte donné. Ce système permet aussi de s'interfacer avec des outils d'administration, des outils de suivi, des applications clientes ou d'autres systèmes de gestion de Workflow [BRIO08].

Le WfMC (la coalition pour la gestion de Workflow) a publié un modèle de référence architecturale, décrivant la structure et les interfaces d'un système de gestion de Workflow.

5.4.2.4 Architecture de WFMS (Système de gestion de workflow), le standard WFMC (la coalition pour la gestion de Workflow) définit une architecture générique pour les WFMS. Comme elle est illustrée dans la Figure 2.7 :

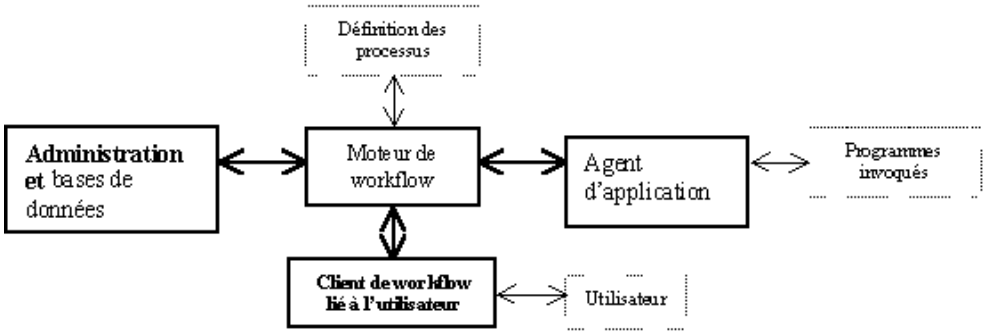

Figure 2.7 Architecture générique pour les WFMS **(**Système de gestion de workflow(Selon DEMAR [DEMA97])

Les 4 entités importantes dans un WFMS sont les suivantes [DEMA97] : Le moteur de Workflow, l'agent d'application, le DBMS (Système de gestion des bases de données), et le client de serveur de workflow : *Les agents d'application* gèrent les programmes informatiques qui sont associés aux activités. *Les clients-Utilisateurs* sont la passerelle entre le WFMS et les utilisateurs qui utilisent le système. *Le moteur de workflow* permet de séquencer les différentes instances de processus et communiquent avec les agents d'application pour réaliser une activité, les clients-utilisateurs et le système de gestion de base de données. *La base de données* permet de stocker les informations relatives aux processus de workflow.

5.4.2.5 Types de données dans les systèmes de gestion de workflow, nous distinguons trois types de données. Comme ils sont illustrés dans la Figure 2.8 **[**HOLL99] :

1. Les données d'application, les données qui sont spécifiques à l'application et ne sont pas accessibles par le système de gestion de workflow.

2. Données concernant le Workflow, les données qui sont utilisées par un système de gestion de workflow pour déterminer les transitions d'état d'une instance de workflow. Par exemple dans les pré-et post-conditions, les conditions de transition ou l'affectation des participants de workflow.

3. Données de contrôle de workflow, les données qui sont gérées par le système de gestion de workflow et / ou un moteur de workflow. Ces données sont internes au système de gestion de workflow et ne sont pas normalement accessibles aux applications.

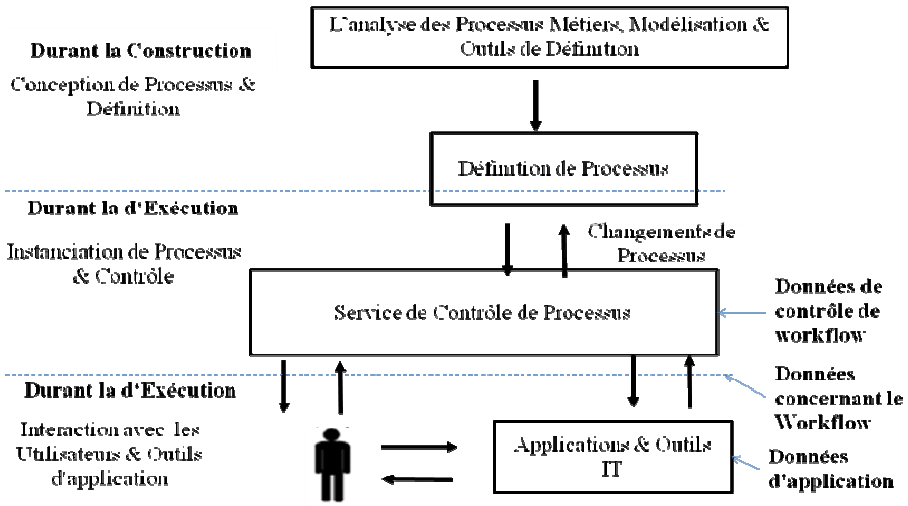

Figure 2.8 Types de données dans les systèmes de gestion de workflow(selon David Hollingsworth [HOLL99])

5.5. Service Orienté Architecture (SOA)

Le BPM permet aux entreprises de mieux appréhender leurs processus métier, ***l'architecture SOA*** permet aussi de résoudre les problèmes de réutilisabilité et d'élimination des données dupliquées dans les infrastructures du système informatique. L'association de ces deux approches améliore significativement le système d'information et la gestion des processus métier [BRIO08].

Les entreprises ont toujours cherché une solution pour intégrer les systèmes existants dans le but d'avoir un support IT pour les « business process ». SOA – Offre une telle architecture. Elle unifie les « business process » en structurant de grandes applications dans une collection des modules plus petits qu'on appelle des services. Différents groupes d'utilisateurs (à l'intérieur ou extérieur de l'entreprise) peuvent utiliser ces applications.

En plus, les nouvelles applications qui sont construites autour de ces concepts ont beaucoup plus de flexibilité et d'uniformité.

Le service peut : être codé dans n'importe quel langage, s'exécuter sur n'importe quelle plate-forme (matérielle et logicielle).

Le service doit : offrir un ensemble d'opérations dont les interfaces sont publiées, être autonome (disposer de toutes les informations nécessaires à son exécution : pas de notion d'état), respecter un ensemble de contrats (règles de fonctionnement), correspondre aux processus métier et fonctions mutualisables au niveau de l'entreprise afin d'aligner l'informatique aux changements des décisions stratégiques et tactiques.

5.6. Standardisation du BPM

La standardisation de la représentation des processus est un enjeu majeur pour faciliter l'intégration entre les outils de BPM. La standardisation a lieu à différents niveaux [JEAN08] : Au niveau de la modélisation des processus, au niveau de l'exécution des processus et au niveau de la communication avec le Système d'information.

5.6.1. La modélisation des processus métiers avec BPMN

La spécification BPMN ou « Business Process Modeling Notation » décrit une notation standard de modélisation des processus métier. Élaborée en 2001, elle a été publiée une première fois en 2004. Depuis 2005, le consortium OMG déjà propriétaire du langage UML a repris la maintenance et l'évolution de la spécification de la notation BPMN. Une brève description sur la modélisation des processus métiers avec BPMN [BRIO08] :

- La notation BPMN a pour objectif de proposer un moyen simple et visuel de communication entre les différents intervenants chargés de mettre en œuvre une approche de gestion des processus métiers dans l'organisation.
- Les concepteurs de la notation BPMN ont cherché à combler formellement le vide existant entre la définition des processus métiers et leur mise en œuvre. Ce vide est comblé en proposant deux perspectives distinctes dans la spécification.
- La première partie de la spécification est consacrée à la description de la notation BPMN et de ses éléments fondamentaux. La seconde partie décrit la traduction des différents éléments de la notation BPMN en éléments du langage d'exécution des processus métiers BPEL.
- Deux rôles distincts se chargent respectivement de ces deux perspectives. L'analyste métier se charge d'élaborer les diagrammes de processus métiers sans tenir compte des aspects techniques et d'exécution des processus modélisés. L'informaticien prend la responsabilité d'ajouter les informations nécessaires à la transcription des modèles dans un langage d'exécution.
- La spécification de la notation BPMN propose deux niveaux d'abstraction :
  - Un niveau sommaire composé uniquement d'éléments essentiels à l'analyste métier en omettant les détails de transcription dans le langage d'exécution des processus.
  - Un niveau détaillé complétant les éléments fondamentaux du niveau sommaire avec des informations techniques destinées aux moteurs d'exécution des processus métiers.

- La notation BPMN couvre uniquement la description des éléments de la notation sans préciser de méthodologie particulière de sa mise en œuvre. La Figure 2.9 représente un simple exemple de diagramme BPMN.

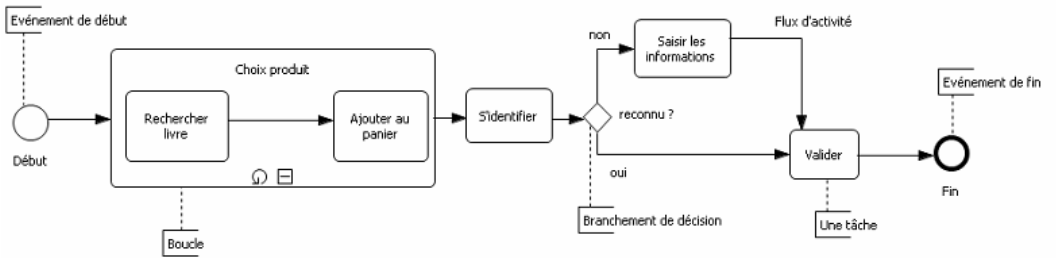

Figure 2.9 Exemple de diagramme BPMN (Selon Briol **[BRIO08**])

- Les éléments du langage destiné aux analystes représentent les éléments de base ne gardant que les aspects génériques de modélisation des processus métiers. Ces éléments de base sont ensuite complétés par l'informaticien en y ajoutant les informations nécessaires à ces exécutions.

5.6.2. Business Process Definition Metamodel (BPDM)

Il existe maintenant un atout substantiel des descriptions de processus d'affaires, notations, implémentations et des machineries. Mais un grand nombre de ces derniers sont isolées. Isolées d'une technologie, méthodologie, ou d'une notation particulière.

BPDM [BPDM08] offre la possibilité de représenter et de modéliser les processus métier indépendante de notation ou de méthodologie, mettant ainsi ces différentes approches ensemble dans une tendance cohésive. Ceci est fait en utilisant un «méta-modèle", un modèle sur la façon de décrire les processus d'affaires - un genre de vocabulaire partagé des processus avec des rapports bien définis entre les termes et les concepts.

Le méta modèle BPDM utilise la norme "Meta Object Facility" (MOF) [MOF06] de OMG pour reproduire d'une manière très générale les processus d'affaires, et de donner une syntaxe XML pour stocker et transformer les modèles de processus d'affaires entre les outils et les infrastructures [BPDM08]. Divers outils, méthodes et technologies peuvent, par la suite, cartographier leurs façons de voir, comprendre, et d'implémenter les processus vers et à travers le standard BPDM. Pour arriver à cet objectif, BPDM support deux vues fondamentales et complémentaires de processus (cf. la section 5.3.2.4.).

Pour savoir plus sur les modèles proposés par OMG avec la norme BPDM voir cette référence [BPDM08].

5.6.3. Business Process Execution Language (ou BPEL)

BPEL (Business Process Execution Language) [BPEL07] est une initiative de la BPMI dont le but est de proposer une représentation XML des activités liées à l'exécution d'un processus. Là où la notation BPMN s'attache à décrire statiquement les processus, le langage BPEL décrit la dynamique d'ensemble.

«WS BPEL définit un modèle et une grammaire pour décrire le comportement d'un processus métier basé sur les interactions entre le processus et ses partenaires. L'interaction avec chaque partenaire se fait par des interfaces de service web…» [BPEL07].

Concrètement, WS-BPEL est une nouvelle couche au-dessus de WSDL. Un document WSDL définit les types des messages et des ports qui représentent les opérations proposées par le service défini et les modalités des différentes interactions où il peut intervenir. Ces informations sont utilisées par WS-BPEL pour spécifier le flot des actions à exécuter. Un document WS-BPEL est basé sur XML et peut être exécuté par un moteur d'orchestrations qui agit comme un coordinateur central. Le moteur lit le document WS-BPEL et invoquera les différents services dans l'ordre spécifié. Le service composite est lui même présenté comme un Service Web qui peut à son tour être invoqué de la même manière [ABOU10].

La structure d'un processus WS-BPEL est décrite dans un fragment de XML-Schema W3C. Le fichier BPEL définit le processus, ou l'enchaînement et la logique des actions qui seront exécutées par le moteur d'orchestration. La structure du fichier BPEL est la même que celle du processus. Ce fichier est véritablement le code source de l'application que constitue le processus, le moteur d'orchestration agissant comme une machine virtuelle capable d'exécuter le code BPEL.

La balise <process> : la balise <process> est l'élément racine (au sens XML) du fichier BPEL. C'est à l'intérieur de cette balise que se retrouvera la description complète du processus. Grâce à l'attribut name, on peut donner un nom au processus. *Exemple:*

```
<process
 name="processName" xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
 targetNamespace=http://example.com
 xmlns:tns="http://example.com"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"> [...]    </process>
```

Les moteurs WS-BPEL : Il n'existe pas de standard pour l'architecture des moteurs WS-BPEL et les applications qui existent sont le résultat d'une interprétation libre

de la spécification WS-BPEL faite par les différents concepteurs [BRIO08] : ***Projets Open source (***Active BPEL ActiveBPEL (2007) ; Apache ODE Apache (2007)). ***Produits commerciaux*** (Oracle BPEL Process Manager Oracle (2007) ; IBM WebSphere ibm (2008)).

5.6.4. La technologie des systèmes de gestion des processus métiers

L'emploi de protocoles standards de communication assure une portabilité entre systèmes de gestion des processus métiers [BRIO08]. La Figure 2.10 illustre l'architecture combinant ces protocoles, elle est décomposée en trois niveaux complémentaires :

- La fondation constituée essentiellement du langage XML et de ses spécifications comme le schéma XML, XPath, XSLT, XForm.
- L'intégration des fonctionnalités des applications sous-jacentes en employant les services Web composés essentiellement des protocoles de communication comme SOAP, WSDL et UDDI.
- La définition de la logique de séquence d'exécution supportée avec des langages XML de haut niveau comme WSCI ou BPEL représentant chacun un mode d'interaction entre services Web : la chorégraphie ou l'orchestration.

L'utilisation importante du langage XML garantit l'ouverture d'échange et de traitements d'informations provenant de diverses origines tant depuis l'interface utilisateur que des applications sous-jacentes [BRIO08].

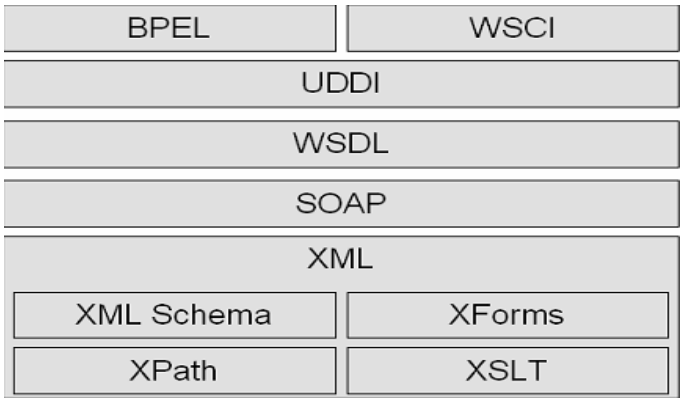

Figure 2.10 Architecture et protocoles standards (Selon Briol [BRIO08])

6. Conclusion

Dans ce chapitre, nous avons présenté une description détaillée du processus d’affaires, ou ce que d’autres ont appelé processus métier, dans son environnement. Ainsi que les différentes termes et méthodologies liées à ce concept, tel que le système de gestion des Workflows qui définit, crée et gère l'exécution des workflows grâce à l'utilisation du logiciel, qui fonctionnent sur un ou plusieurs moteurs de workflow. Ces moteurs de workflow sont capables d'interpréter la définition de processus.

Les sujets du processus d’affaires occupent une partie importante dans les méthodologies avancées de l’ingénierie d’entreprise. Récemment, on recense la Gestion de Processus d’affaires (BPM) et l’Architecture Orientée Service (SOA), qui facilitent la réutilisation durant le développement des logiciels [ČIUK07a]. Cependant, représentant les connaissances sur les familles des processus d'affaires similaires et la réutilisation de ces connaissances dans des projets de l’ingénierie d'entreprise reste un problème encore posé.

De ce fait dans le chapitre suivant, nous présenterons la notion de réutilisation, différentes techniques de réutilisation, ainsi qu’un état de l’art sur les travaux existent sur la réutilisation des connaissances ontologiques dans le domaine de l’ingénierie d’entreprise et exactement dans les processus d’affaires. Cela afin de cerner les lacunes trouvées dans ces travaux et essayer de les remédier par la suite de ce mémoire.

Chapitre -03-

Réutilisation des connaissances ontologiques dans le processus d'affaires

1. Introduction

"Most software development is mostly redevelopment."

D.M Weiss and C.T.R Lai,

Software Product-Line Engineering:

A Family-Based Software Development Process,

Addison Wesley, 1999

La quasi-totalité des systèmes d'informations actuels repose sur la notion de réutilisation, laquelle a été restreinte au niveau des composants logiciels tels que (fonction, objet,….) ; Toutefois, la tendance actuelle est orientée vers la réutilisation à des niveaux d'abstraction plus élevés. Dans ce cas, Albertas Caplinskas et ses co-équipiers [CAPL02, CAPL03, CAPL03a, CAPL04, ČIUK06, ČIUK07, ČIUK07a] ont déclaré que, dans l'ingénierie du système d'information, la réutilisation peut avoir lieu dans l'ingénierie d'entreprise, l'ingénierie de domaine et l'ingénierie d'application.

Dans ce chapitre, après la définition de la notion de réutilisation, nous traiterons quelques techniques pour la réutilisation. Ensuite nous exposerons et évaluerons quelques travaux principaux existants sur la réutilisation des connaissances ontologiques dans les processus d'affaires.

2. Définitions de réutilisation

Plusieurs et différents points de vue existent sur la réutilisation. Nous donnons quelques exemples de la littérature.

Pour **Freeman** la réutilisation est l'utilisation de toute information qui peut avoir besoin d'un développeur dans le processus de création de logiciels [JOHA97].

De même **Basili** et **Rombach** voient la réutilisation des logiciels comme l'utilisation de tout ce qui est associé à un projet de logiciel, y compris les connaissances [JOHA97].

Nous adoptons la définition de **Krueger** qui donne un sens large à la réutilisation des logiciels [JIAN10] **:**

"Réutilisation de logiciels est le processus de création des systèmes logiciels à partir d'artefacts logiciels existants plutôt qu'à les construire à partir de zéro. Typiquement, la réutilisation implique la sélection, la spécialisation et l'intégration des artefacts, bien que

les différentes techniques de réutilisation puissent accentuer ou souligner certains de ces éléments."

Cette définition a une vue large et générale, c'est-à-dire que la réutilisation de logiciel ne se limite pas à la seule réutilisation de code. Elle concerne en effet tous les artefacts définis, et éventuellement implantés, produits lors d'une première réalisation logicielle. Ces artefacts comprennent des codes mais aussi des connaissances liées, par exemple, à la conception, à l'architecture, aux conditions et contextes d'exécution, aux cas de tests, etc [JIAN10]. La réutilisation peut ainsi être mise en œuvre tout au long du cycle de vie d'un développement logiciel. De même pour le développement d'un modèle exécutable d'un processus d'affaires.

3. Techniques de réutilisation

Nous examinons ici quelques approches importantes dans la réutilisation, toutes ces approches cherchent à améliorer la qualité et la productivité du développement logiciel en construisant des applications par des artefacts logiciels déjà existants :

3.1. Réutilisation ad-hoc

La plus ancienne, Il s'agit pour un programmeur de réutiliser de façon opportuniste un morceau de code ou un schéma de réutilisation qui n'avait pas été préparé dans ce sens. La réutilisation est ici non préparée, entièrement décidée et mise en œuvre par celui qui réutilise, en fonction de ses besoins du moment [JIAN10].

Avantage

- Elle ne coûte rien au niveau de la préparation.

Inconvénient

- La qualité et l'adéquation des artefacts réutilisés ne sont pas garanties. Elle dépend complètement du talent, et de la chance, de celui qui réutilise.

C'est la forme de réutilisation la plus utilisée, encore aujourd'hui.

3.2. Objet

Elle utilise les objets comme élément de base et leurs interactions pour constituer des applications logicielles et des systèmes informatiques. Un objet est un représentant d'un type abstrait, une classe, qui contient des attributs permettant de décrire les propriétés de l'objet et des interfaces permettant d'opérer sur ses attributs et de collaborer avec d'autres objets [JIAN10].

Avantages

- Elle repose sur les deux notions : **L'héritage,** l'un des mécanismes permettant explicitement la réutilisation et l'adaptation de type par surcharge et redéfinition ; et **Le polymorphisme** qui participe également à l'amélioration de la réutilisation en permettant l'utilisation de mêmes interfaces sur des objets différents.

Inconvénients

- L'adaptation d'une application construite suivant l'approche orientée objets, doit être réalisée à un niveau architectural trop bas - celui des classes. Il est en effet difficile, au niveau d'un langage de programmation, de déterminer des éléments réutilisables. Les classes sont en effet très liées à leur environnement immédiat qui change de projet en projet. L'approche orientée objets permet donc la réutilisation dans un cadre très limité.

3.3. Composants

Les composants ont en effet pour vocation de servir comme éléments de base pour la construction d'applications logicielles. Plus précisément, les approches à composants perçoivent le développement d'applications logicielles comme un assemblage de composants, et gèrent la maintenance et l'évolution d'applications par la personnalisation et le remplacement de composants réutilisables [JIAN10].

Avantages

- Ils se situent à un niveau architectural plus élevé que celui proposé par des objets. Typiquement, un composant peut être constitué de plusieurs objets pour la réalisation d'une tâche spécifique.
- Il existe plusieurs gains de la réutilisation dans l'approche à composant, par exemple, une bibliothèque réutilisable fournie par le système d'exploitation, une base de données et interface utilisateur graphique uniforme (GUI[10]), etc.

Inconvénients

- Le développement d'une bibliothèque de composants réutilisables est souvent coûteux.
- Les approches à composants ne définissent généralement pas de mécanismes systématiques permettant de concevoir de façon stratégique le développement de composants de sorte qu'ils soient réutilisables lors de développements futurs [JIAN10].

[10] GUI: Graphical user interface

3.4. Services

Ce paradigme utilise la notion de service comme élément de base pour la construction d'applications logicielles. Un service est défini comme une entité logicielle qui peut être utilisée de façon statique ou dynamique pour la réalisation d'une application logicielle. Un consommateur, ou client, sélectionne un service à partir de sa description. Il l'utilise sans avoir connaissance de la technologie sous-jacente nécessaire à son implantation ni de sa plate-forme d'exécution. Inversement, le service ne connaît pas le contexte dans lequel il va être utilisé par un client [JIAN10].

Avantages

- Cette indépendance à double sens est une propriété forte des services qui facilite le faible couplage.
- Une architecture à service (**SOA** pour **Service Oriented Architecture** [ERL05]) regroupe un ensemble de services et des mécanismes d'assemblage permettant le développement d'applications basées sur la réutilisation de services.
- Les caractéristiques des applications à services, tels que la substituabilité transparente, le faible couplage, la liaison retardée et la technologie d'implémentation neutre, sont particulièrement intéressantes dans de nouveaux domaines d'applications ayant de fortes contraintes de dynamisme.

Inconvénients

- Dans certaines situations où de nombreux services sont disponibles, la stratégie de recherche, sélection et composition de services est difficile à spécifier pour les développeurs.
- Certaines applications distribuées en temps réel désirent faire collaborer plusieurs services pour réaliser une transaction en temps opportun. Mais, le dynamisme des services soulève le défi de la gestion de la qualité des applications distribuées sous la forme de composition de services.
- La composition de services pour former de nouvelles applications demeure une activité forte complexe.

L'approche à service seule est encore insuffisante pour une réutilisation effective [JIAN10].

3.5. Modèles

Les approches précédentes se concentrent sur la réutilisation de code. La réutilisation ne se limite pas au code mais concerne tous les artefacts d'un développement

logiciel. Les modèles (une simplification d'une réalité, il est crée dans un but bien précis.), en particulier sont des éléments de réutilisation intéressants. De nombreux modèles sont créés lors d'un développement logiciel, tels que les exigences, les architectures, les spécifications de sécurité, etc. Les modèles possèdent une grande valeur intrinsèque et les réutiliser présente une forte valeur ajoutée [JIAN10].

Avantages

- MDE (Model Driven Engineering [SCHM06]) et MDA (Model Driven Architecture[11]), ces deux approches prônent l'étude et l'utilisation systématique de modèles, elles fournissent des technologies permettant aux développeurs de s'abstraire de complexités croissantes liées aux langages et plates-formes d'exécution.
- Les modèles à l'exécution maintiennent une vue de l'environnement d'exécution d'un programme et sont généralement utilisés pour adapter, à l'exécution, le comportement d'un programme. Le but de ces modèles est de cacher la complexité des phénomènes se produisant durant l'exécution pour permettre à des agents (des programmes) de faire évoluer le système au mieux (en déployant de nouveaux artefacts par exemple).
- La réutilisation de modèles a donné naissance aux patterns de conception, ou design patterns [JIAN10].

Inconvénients

- La réutilisation de modèles n'est pas si simple. Les modèles constituent une abstraction partielle, très focalisée, d'un système et sont ainsi très liés à un contexte. Pour les réutiliser directement, il faut donc se trouver dans les mêmes conditions de développement (qui ne sont souvent pas formalisées).
- Les modèles sont surtout réutilisables quand on reste à un haut niveau d'abstraction.

3.6. Modèle des deux cycles de vie

Les modèles de conception, les frameworks et les composants sont conçus pour développer des applications simples, qu'on peut utiliser pour le développement, ultérieur, d'applications similaires. Ceci est en fait un contraste avec l'ingénierie de domaine, qui est conçu pour modeler des familles d'application [CZAR00].

L'auteur de [CZAR00] définit l'ingénierie de domaine comme *:* « *l'activité de collection, organisation et de stockage de l'expérience passée dans la construction des systèmes ou des parties de systèmes dans un domaine particulier sous forme d'artefacts*

[11] MDA: Models-driven Architecture http://www.omg.org/mda/

réutilisables (c'est-à-dire, produits réutilisables de travail), ainsi que de prévoir un moyen adéquat pour la réutilisation de ces d'artefacts (c'est-à-dire, l'extraction, la qualification, la diffusion, l'adaptation, l'assemblage, et ainsi de suite) lors de la construction de nouveaux systèmes. »

L'ingénierie de domaine englobe les trois activités suivantes : l'analyse de domaine, conception de domaine, et la mise en œuvre de domaine.

Analyse de domaine : définissant la portée de domaine et l'ensemble d'exigences réutilisables et configurables pour les systèmes de ce domaine.

Conception de domaine : développant une architecture commune pour les systèmes dans le domaine et concevant un plan de production.

Implémentation de domaine : Implémentant des artefacts réutilisables, par exemple, composants réutilisables, langages spécifiques de domaine, générateurs, une infrastructure de réutilisation, et un processus de production.

Les résultats de l'ingénierie de domaine, sont réutilisés lors de l'ingénierie d'application, qui est, le processus de la production de systèmes concrets utilisant les artefacts réutilisables développés lors de l'ingénierie de domaine.

Nous avons donc deux activités pour le développement : l'ingénierie de domaine qui représente l'activité de développement pour la réutilisation, et l'ingénierie d'application qui représente l'activité de développement avec la réutilisation. Ces deux activités constituent le modèle à deux cycles de vie [JOHA97]. Nous allons utiliser cette technique dans notre contexte de travail.

4. Certains travaux sur l'intégration des ontologiques dans l'ingénierie de domaine et l'ingénierie d'entreprise

La tendance actuelle dans l'ingénierie des systèmes d'information, nécessite d'examiner les enjeux de réutilisation dans les niveaux d'abstraction plus élevés, à savoir : l'ingénierie d'entreprise (IE), l'ingénierie de domaine (ID), et de l'ingénierie d'application (AE*).*

L'ingénierie d'entreprise (IE) devrait soutenir la réutilisation de tous les objets créés dans le domaine de l'entreprise, y compris les architectures de données et d'informations, les définitions de la ligne de produits, et les modèles d'affaires [CAPL03].

L'ingénierie de domaine (ID) devrait soutenir la réutilisation des produits créés dans certains domaines y compris les modèles génériques de domaine et les architectures

génériques de logiciels. Le terme «domaine» désigne ici «un espace de connaissances ou d'activités caractérisées par un ensemble de concepts et terminologie comprise par les praticiens dans cet espace» [CAPL03].

Le but de *l'ingénierie d'application (IA)* est de produire des produits pour la livraison utilisant des artefacts créés par l'ingénierie d'entreprise et l'ingénierie de domaine.

La structure flexible des ontologies (comme nous l'avons déjà mentionné dans le premier chapitre) permet de présenter les connaissances d'une façon réutilisable, dans ce qui suit nous allons présenter quelques travaux qui utilisent les ontologies pour la représentation des connaissances dans l'ingénierie de domaine et l'ingénierie d'entreprise.

1. Une approche ontologique à l'ingénierie de domaine

Falbo, Guizzardi, et Duarte ont proposé une approche basée sur les ontologies **(Ontology Domain Engineering ODE)** autour de l'ingénierie de domaine [CAPL03]. Le principe de cette approche est d'utiliser *l'ontologie de domaine* en tant que *modèle de domaine* et de dériver la structure d'objet pour eux. En ODE, l'analyse du domaine est considéré comme la phase de développement d'ontologie, la conception de domaine représente cette ontologie par un modèle objet (spécifications d'infrastructure), et l'implémentation de domaine en tant que phase de développement des composants Java (l'Exécution d'infrastructure). ODE fournit l'intégration de l'ontologie en cours de développement avec les ontologies plus générales précédemment développées. Elle permet une réutilisation plus large des connaissances de domaine [CAPL03].

L'approche d'ODE emploie quatre types d'axiomes : *Épistémologiques, Consolidation, Ontologique,* et *Définitions* [CAPL03].

Les axiomes épistémologiques décrivent la structure des concepts (les relations : partie-entier, est-un, etc.).

Les axiomes de Consolidation imposent des contraintes pour ces relations.

Les axiomes Ontologiques représentent les connaissances déclaratives de base. Les contraintes définissent la plupart du temps les pré-conditions qui doivent être satisfaites pour qu'une relation soit uniformément établie.

Les axiomes de définition sont employés pour définir de nouveaux concepts et relations. La logique de premier ordre est employée pour spécifier les axiomes.

2. Les Ontologies dans l'ingénierie d'entreprise

Un système intégré des ontologies pour soutenir la modélisation d'entreprise a été développé à **Toronto Virtual Enterprise (TOVE) projet**. Ce système se compose d'un certain nombre d'ontologies génériques de base, y compris une ontologie **d'activité**,

l'ontologie des ressources, **l'ontologie** d'organisation, et une ontologie de produits. Il comprend également un ensemble d'extensions à ces ontologies génériques qui définissent des concepts supplémentaires (coût, qualité, etc.). Récemment, la tentative a été faite pour développer une ontologie complémentaire de processus d'affaires [GRÜN00].

L'approche TOVE est critiquée parce qu'elle exige trop d'efforts pour instancier le modèle d'une entreprise particulière. Afin de faciliter cette tâche, les auteurs travaillent sur le développement des outils d'automatisation [CAPL03].

Une autre contribution importante à l'ingénierie d'entreprise basée sur l'ontologie, a été faite dans **le projet d'entreprise à l'Université d'Edimbourg** [USCH96]. Dans ce projet une ontologie (**Ontologie D'Entreprise**) pour la modélisation d'entreprise qui vise à soutenir un environnement de l'ingénierie d'entreprise a été développée [CAPL03]. Cet environnement est conçu comme un ensemble intégré des méthodes et des outils pour saisir et analyser les aspects principaux d'une entreprise. L'ontologie d'Entreprise fournit cinq classes de haut niveau pour l'intégration des différents aspects d'une entreprise : **méta-ontologie**, **activités et processus, organisation, stratégie et marketing**. Méta-ontologie définit les concepts de base de modélisation (Entité, Relation, Rôle, Acteur, État D'affaire). Donc, l'ontologie d'entreprise est un système d'ontologies intégrées. Elle est semi formelle, les termes sont exprimés sous une forme restreinte et structurée de la langue naturelle et ils sont complétés par quelques axiomes formels utilisant **Ontolingua**. Par conséquent, cette ontologie ne supporte pas le raisonnement automatique [CAPL03].

D'après l'analyse qui a été faite par Caplinskas [CAPL03] sur les approches présentées précédemment, Caplinskas a montré que ces approches de réutilisation des assets (artefacts) dans l'ingénierie d'entreprise ne sont pas suffisamment établies. Elles souffrent de nombreuses lacunes :

- Absence de contextes théoriques, comment construire la superstructure employée pour définir les ontologies de bas niveau dans le problème aussi bien que dans le domaine de système.
- Couverture insuffisante des vues de modélisation, en particulier dans le domaine du système, qui est un obstacle important à l'élaboration de nombreux assets réutilisables (architecture de l'information, architecture des données, architecture du réseau, etc.).
- Écart sémantique important entre la modélisation des processus d'affaires, processus d'informations et processus d'applications.

- Forte orientation des techniques de l'ingénierie de domaine vers la réutilisation des solutions logicielles et, par conséquent, l'incapacité de modeler les points communs et variables au niveau des affaires et au niveau du traitement de l'information.
- Niveau trop abstrait des modèles et, par conséquent, nécessité d'effectuer des opérations manuelles dans la personnalisation et l'instanciation de la conception réutilisables.
- Accent trop fort sur des aspects structurels et, par conséquence, l'incapacité de réutiliser des aspects dynamiques, particulièrement au niveau des affaires et au niveau de traitement de l'information.
- Structure monolithique des modèles de problème et, par conséquence, l'incapacité de réutiliser les artefacts liés aux domaines verticaux[12] (par exemple, les processus génériques) dans les différents systèmes d'information d'entreprise.

Pour remédier ces problèmes, des solutions ont été proposées par Caplinskas [CAPL03] et ont été élaborées en détail par Caplinskas et Ciuksys dans [CAPL04, CAPL06], donc Caplinskas et Ciuksys voient que [CAPL06] :

- Les systèmes d'informations, pour la plupart, sont utilisés pour soutenir certaines affaires et doivent être alignés avec ces affaires dans tous les aspects. Toute affaire, du point de vue de l'ingénierie des systèmes d'informations, peut être considérée comme un système qui peut être décomposé en deux couches interdépendantes, toutefois, relativement indépendantes :
    - **Les entités de base,** qui sont généralement visées comme des unités enregistrées [CAPL98] et ;
    - **Les processus,** dans lequel certaines entités de base agissent comme des acteurs, d'autres sont utilisées comme ressources pour le processus ou elles sont manipulées par les processus comme des objets traités, qui réalisent des transactions d'affaires (Business Objects). Bien que, chaque processus d'affaires ait sa propre vision sur les entités de base, celles-ci ont également des propriétés ontologiques indépendantes de processus.
- Dans les systèmes d'informations, les propriétés ontologiques d'entités de base sont enregistrées et stockées dans des bases de données indépendantes de processus appelé registres [CAPL98]. La notion de registre comprend également les services nécessaires pour créer et maintenir la base de données appropriée [CAPL06].

[12] Domaines verticaux contiennent des systèmes complets par contre les domaines horizontaux contiennent seulement des parties des systèmes dans la portée de domaine [CZAR00].

- Ils ont aussi utilisé le terme *domaine d'application* pour désigner les connaissances sur les entités de base et leurs propriétés ontologiques. Et le terme *domaine problème* pour désigner les connaissances d'un processus d'affaires particulier, y compris des connaissances sur les propriétés spécifiques d'entités de base, qui sont nécessaire uniquement dans le contexte de ce processus. Donc, chaque système d'informations a un seul domaine d'application et plusieurs domaines problème. Le domaine d'application est implémenté par un nombre de registres et chaque domaine problème par un sous système fonctionnel du système d'informations approprié. Ils supposent que ces registres sont implémentés comme des composants et sont partagés par plusieurs différents sous systèmes [CAPL06].
- Les ontologies dans l'ingénierie des systèmes d'informations (SI) sont utilisées pour soutenir à la fois la réutilisation des connaissances du domaine d'application (concepts de domaine) et la réutilisation des connaissances du domaine de processus (concepts de processus). Habituellement, les deux types de connaissances sont mixtes et interdépendants, de ce fait, on trouve que les processus d'affaires sont décrits en termes d'un domaine d'application particulier [CAPL06].
- D'autre part et d'après les mêmes auteurs [CAPL06], pour généraliser les connaissances des processus, il est possible de séparer les connaissances sur les entités d'affaires et celles sur les processus d'affaires. En décrivant les processus en termes de rôles abstraits (les acteurs, les ressources, les objets métier, etc.) joués par les entités de base, mais pas directement en termes d'entités de base elles-mêmes. En d'autres termes, il est possible de décrire les processus d'affaires génériques et de les réutiliser dans les différents domaines d'application.

Donc, nous avons remarqué que les travaux de Caplinskas et ses co-équipiers [CAPL02, CAPL03, CAPL03a, CAPL04, ČIUK06, ČIUK07, ČIUK07a] sont très intéressants, dans la mesure où ils ont proposé une approche à base d'ontologies qui permet la réutilisation des connaissances au niveau du processus d'affaires. Cette approche, qui est en cours d'évolution, consiste à combiner les différentes techniques de réutilisations développées dans l'ingénierie de domaine [CZAR00], l'ingénierie de connaissances [CHAN86] et l'ingénierie des systèmes basés sur les ontologies [DAVI06].

Leur contribution principale est de proposer une façon de séparer les connaissances du processus d'affaires et les connaissances du domaine d'application au cours de l'ingénierie pour la réutilisation (l'ingénierie du domaine de processus). En proposant deux types d'ontologies réutilisables de haut niveau : Ontologie du domaine d'application de

haut niveau et Ontologie du processus d'affaires de haut niveau. Les deux ontologies sont basées sur un système de méta-concepts, qui constitue une ontologie de haut niveau, est également proposé.

Ils ont également proposé une façon de combiner les ontologies du processus d'affaires et les ontologies du domaine d'application au cours de l'ingénierie avec la réutilisation (l'ingénierie du processus), dans un domaine d'application particulier. Ainsi que, au cours de ce processus, la façon d'associer les connaissances du domaine problème avec les composants logiciels réutilisables indépendantes du domaine. Afin de construire le sous-système fonctionnel supportant le processus d'affaires correspondant.

La notion de processus d'affaires générique est introduite pour désigner une famille de processus d'affaires similaires. Selon la conceptualisation de la variabilité entre les membres de même famille, ils ont distingué dans cette approche deux phases pour localiser le processus d'affaires générique dans un domaine d'application bien choisi. Les deux phases sont : l'ingénierie du domaine de processus (développement pour la réutilisation) et l'ingénierie de processus (développement avec la réutilisation). La réutilisation présentée dans cette approche permet la résolution de la variabilité d'une manière statique c'est à dire durant la construction du processus d'affaires localisé dans un domaine d'application choisi. Cela, sans tenir en compte à la variabilité qui peut avoir lieu durant l'exécution de ce processus.

En plus, d'après JIANQI YU [JIAN10], les approches pour la réutilisation à deux phases développées dans l'ingénierie de domaine et adaptées par Caplinskas, sont mises en place avec succès ; mais elles ne proposent pas de dynamisme à l'exécution. De plus, elles ne conviennent pas très bien aux environnements dynamiques changeants. De ce fait, nous trouvons qu'il a pris en considération la variabilité qui aura lieu durant l'exécution d'applications, où il a proposé une approche à trois phases pour remédier ces problèmes, il a proposé de structurer le développement des applications à services en trois phases :

Une phase d'ingénierie domaine dont le but est de définir des services et des architectures de référence pour la composition de ces services. L'architecture de référence est caractérisée par des parties variables laissant la place à des adaptations lors des phases plus amont ;

Une phase d'ingénierie applicative dont le but est de définir une architecture à services apportant une solution à un problème précis. Cette architecture à service reprend une architecture de référence et élimine certains points de variabilité. Elle peut conserver certains aspects variables qui seront résolus lors de la phase suivante ;

Une phase d'exécution qui exécute de façon autonomique une application à services en fonction du contexte à l'exécution et des contraintes architecturales exprimées par l'architecture applicative, définie précédemment. Cette phase repose sur une machine d'exécution à service supportant la découverte, la sélection et la liaison de services hétérogènes, conformément à une architecture.

Pour les raisons citées précédemment, en vue de construire notre approche, nous allons nous baser sur l'approche proposée par JIANQI YU sur les lignes de production et largement sur celle de Caplinskas et ses coéquipiers sur les processus d'affaires qui ont été toutes les deux visées à la réutilisation.

Donc, dans ce mémoire nous proposerons une nouvelle approche, semi automatique, qui se base sur les ontologies pour permettre la réutilisation des connaissances du processus d'affaires durant l'ingénierie de domaine d'application puis dans l'environnement d'exécution choisi. C'est-à-dire nous allons distinguer deux types des points de variation, statiques et dynamiques. Les points de variation statiques seront résolus durant la construction du processus dans le domaine d'application choisi et les autres points de variations dites dynamiques seront résolus durant l'exécution de ce processus dans l'environnement d'exécution choisi aussi. Cela pour tenir compte la variabilité qui peut avoir lieu durant l'exécution de ce processus.

Donc dans cette approche, qui fera l'objet de la deuxième partie de ce mémoire, nous distinguerons deux parties dans la phase de l'ingénierie de processus proposée par Caplinskas : la phase de l'ingénierie applicative de processus et la phase de l'exécution de processus. Alors, nous aurons donc une approche à trois phases : la phase d'ingénierie du domaine de processus, la phase de l'ingénierie applicative de processus et la phase de l'exécution de processus.

5. Conclusion

L'intégration des ontologies dans l'ingénierie de domaine et l'ingénierie de connaissances fournit une combinaison forte pour la représentation et la réutilisation des connaissances.

Dans ce chapitre, nous avons fait une synthèse des techniques de réutilisation et nous avons présenté quelques travaux relatifs à l'utilisation des ontologies dans l'ingénierie d'entreprise ainsi que dans l'ingénierie de domaine. Nous avons aussi présenté quelques travaux de Caplinskas et ses coéquipiers, qui ont proposé une approche basée sur l'ontologie à l'ingénierie du domaine qui permet la réutilisation des connaissances de processus. Cette approche consiste à combiner les différentes techniques de réutilisations développées dans l'ingénierie de domaine, l'ingénierie de connaissances et l'ingénierie des systèmes basés sur les ontologies. Elle est constituée de deux phases : L'ingénierie du domaine de processus (développement pour la réutilisation) et l'ingénierie de processus (développement avec la réutilisation).

Nous avons présenté aussi le travail de JIANQI YU qui a déclaré que les approches pour la réutilisation à deux phases développées dans l'ingénierie de domaine et adaptées par Caplinskas, sont mises en place avec succès ; mais elles ne proposent pas de dynamisme à l'exécution. Pour y remédier, JIANQI YU propose une approche à trois phases, il a proposé de structurer le développement des applications à services en trois phases : Une phase d'ingénierie domaine, Une phase d'ingénierie applicative, Une phase d'exécution. Nous avons aussi défini en brève l'objectif de ce mémoire qui consiste à proposer une nouvelle approche, nous concilierons l'approche proposée par JIANQI YU sur les lignes de production et celle de Caplinskas et ses coéquipiers sur les processus d'affaires. Cette approche fera l'objet des prochains chapitres.

Chapitre -04-

Proposition

1. Introduction

Shotgun surgery: You whiff this when every time you make a kind of change,
you have to make a lot of little changes to a lot of different classes.
Kent Beck and Martin Fowler [FBB+99] describing
one of the "bad smells" often found in OO code

Dans la première partie de ce mémoire nous avons présenté un état de l'art sur la représentation des connaissances inhérentes aux ontologies. Une vue détaillée du domaine de représentation de ces connaissances, qui est l'ingénierie d'entreprise et exactement les processus d'affaires. Ainsi que certains travaux sur la réutilisation dans ce domaine. Dans la seconde partie, nous allons présenter notre proposition qui vise à concilier l'approche proposée par JIANQI YU [JIAN10] sur les lignes de production et celle de Caplinskas et ses co-équipiers sur les processus d'affaires, qui ont été toutes les deux visées à la réutilisation.

L'approche proposée dans ce mémoire est structurée en trois phases de développement :

La première phase, ou ingénierie du domaine de processus, vise à mettre en place au sein d'un domaine un ensemble d'éléments réutilisables fondé autour des notions de modèles des caractéristiques et d'ontologies représentant un processus d'affaires générique, que nous présenterons aussi dans ce chapitre.

La seconde phase, ou ingénierie applicative du processus, concernant l'exploitation des éléments communs précédemment définis. Son objectif est de spécifier et localiser, de façon plus précise, les processus d'affaires dans un domaine d'application choisi.

Enfin, *la troisième phase*, a pour but d'exécuter le processus d'affaires spécifié, en identifiant et utilisant aussi au dernier moment les services ou les activités appropriées et disponibles pour l'exécution des autres activités convenablement. Un système de gestion des Workflow est invoqué pour gérer l'exécution du processus d'affaires. Donc l'objectif de la troisième phase est d'intégrer le processus d'affaires, localisé dans un domaine d'application déjà choisi, dans un environnement d'exécution bien choisi.

Cette approche se concentre sur les notions de modèles de caractéristiques et les ontologies, celles-ci, nous les définissons de façon précise à l'aide du méta modèles ou de ce qu'on appelle les ontologies de haut niveau. La conceptualisation de toutes les ontologies, proposées dans ce mémoire, est faite par des diagrammes UML. Comme UML ne propose pas des services de méta modélisation, nos diagrammes ne doivent pas être considérés comme des diagrammes UML propres.

De ce fait, cette partie est structurée en deux chapitres, Chapitre 04 et Chapitre 05, qui approfondissent notre travail. Le Chapitre 04, rappelle notre contexte de travail et présente une vision détaillée de notre approche. Celle-ci repose sur la définition de trois phases successives pour permettre la réutilisation des connaissances au niveau du processus d'affaires. L'étude de cas utilisant cette approche sera donnée dans le chapitre suivant.

2. Rappel sur le contexte

Notre recherche se situe dans le domaine de la réutilisation de connaissance dans les processus d'affaires en général et dans l'utilisation de l'ontologie dans la réutilisation des connaissances en particulier. En d'autre terme, ce travail vise à simplifier la représentation et la réutilisation de connaissances des processus d'affaires en utilisant les ontologies.

Comme nous l'avons présenté dans la première partie de l'état de l'art, la mise en place des ontologies permet la réutilisation effective de connaissances. Ainsi que le langage utilisé pour représenter formellement une ontologie a un impact direct sur le niveau de formalisation de cette ontologie, les logiques de description sont particulièrement adaptées à la représentation des connaissances formalisées.

Dans la deuxième partie, nous avons vu l'importance des processus d'affaires dans les méthodologies avancées de l'ingénierie d'entreprise. Bien qu'on a recensé la Gestion de Processus d'affaires(BPM) [SMIT03] et l'Architecture Orientée Service (SOA) [ERL05], qui facilitent la réutilisation durant le développement des logiciels, la représentation des connaissances sur les familles des processus d'affaires similaires et la réutilisation de ces connaissances dans des projets de l'ingénierie d'entreprise qui reste un problème encore posé [ČIUK07a]. Ce qui nous conduit également, dans la troisième parie de l'état de l'art, de faire une synthèse des techniques de réutilisation. Où nous avons présenté quelques travaux relatifs à l'utilisation des ontologies dans l'ingénierie d'entreprise ainsi que dans l'ingénierie de domaine. Nous avons aussi présenté quelques travaux de Caplinskas et ses coéquipiers [CAPL02, CAPL03, CAPL03a, CAPL04, ČIUK06, ČIUK07, ČIUK07a], qui ont proposé une approche basée sur l'ontologie à l'ingénierie du domaine qui permet la réutilisation des connaissances de processus. Qui est une combinaison de différentes techniques de réutilisations développées dans l'ingénierie de domaine [CZAR00], l'ingénierie de connaissances [CHAN86] et l'ingénierie des systèmes basée sur les ontologies [DAVI06]. Elle est constituée de deux phases :

L'ingénierie du domaine de processus (développement pour la réutilisation) et l'ingénierie de processus (développement avec la réutilisation). Nous avons présenté aussi le travail de JIANQI YU [JIAN10] qui a déclaré que les approches pour la réutilisation à deux phases développées dans l'ingénierie de domaine et adaptées par Caplinskas, sont mises en place avec succès, mais elles ne proposent pas de dynamisme à l'exécution. Pour remédier à ces problèmes, JIANQI YU [JIAN10] propose une approche à trois phases, il a proposé de structurer le développement des applications à services en trois phases : Une phase d'ingénierie de domaine, Une phase d'ingénierie applicative et une phase d'exécution. Nous avons aussi brièvement défini l'objectif de ce mémoire qui consiste à proposer une nouvelle approche, nous concilierons l'approche proposée par JIANQI YU sur les lignes de production et celle de Caplinskas et ses coéquipiers sur les processus d'affaires.

3. Présentation de l'approche

Avant de présenter notre approche, nous allons tout abord rappeler la notion de processus d'affaires générique (appelé aussi domaine de processus).

3.1. Processus d'affaires générique

Comme nous avons déjà défini, un processus d'affaires est un ensemble partiellement ordonné d'activités liées qui crée une valeur en transformant une entrée en une sortie plus précieuse. Les entrées et les sorties peuvent être des artefacts et / ou des informations et la transformation peut être effectuée par des acteurs humains, machines, ou les deux [ČIUK07a].

Un processus d'affaires générique est défini comme étant une abstraction d'une famille de processus d'affaires. Les membres de cette famille sont caractérisés par des points (parties) communs importants, et chaque membre particulier est également spécifié par des points supplémentaires (variabilité) [ČIUK07a].

Un processus d'affaires générique est décrit par une sorte de modèle des caractéristiques [KANG90] et par une ontologie. Le processus d'affaires générique ne comprend pas les connaissances de contrôle sur la séquence des activités d'affaires. Les connaissances de contrôle sont ajoutées plus tard, réutilisant l'ontologie de processus dans un domaine particulier d'application. Les processus génériques devraient être exprimés en termes de rôles abstraits (acteurs, entrées, sorties, ressources, requises). Ces processus génériques sont utilisés pour générer des processus particuliers (c'est- à -dire les membres de la famille) qui sont alors situés dans le domaine d'application choisi [ČIUK07a].

L'objectif de génération est de produire la configuration requise du processus, en d'autres termes, pour décider quelles variabilités qui ne sont pas appropriées à ce membre particulier de famille pour les rejeter. Par la suite, les rôles doivent être substitués par les entités du domaine d'application dans lequel ce membre de la famille de processus est situé. Ils ont appelé cette activité Attribution de rôle [ČIUK07a].

3.2. Principe de l'approche

Dans le cadre de notre travail, nous avons proposé une approche semi-automatique qui permet la réutilisation des connaissances dans le processus d'affaires. Cette approche est basée sur le travail de Caplinskas, l'ingénierie de domaine, l'ingénierie des connaissances et l'ingénierie des systèmes basés sur les ontologies.

L'idée générale de l'approche proposée est d'abord de séparer l'ontologie de processus d'affaires de l'ontologie de domaine d'application, ensuite réutiliser l'ontologie de processus dans les différents domaines d'application [ČIUK07a]. C'est pourquoi, nous proposons d'enrichir deux types d'ontologies réutilisables de haut niveau : Ontologie du processus d'affaires de haut niveau et Ontologie du domaine d'application de haut niveau. Les deux ontologies sont basées sur un système de méta-concepts, qui constitue une ontologie de haut niveau, que nous traiterons également. Dans ce cas la, d'après ces ontologies de haut niveau, nous pouvons construire des ontologies du domaine de processus ainsi que les ontologies de domaine d'application de processus respectivement. L'ontologie du domaine de processus ou bien l'ontologie générique du processus d'affaires constitue des points communs et les points de variation, pour des raisons de simplicité, nous avons seulement choisi la variabilité d'activités.

Pour réutiliser cette ontologie dans un domaine d'application particulier, nous devons configurer cette ontologie, en résoudre certaines variabilités déjà définies par le choix des variantes (c'est-à-dire les activités) correspondant au domaine d'application choisi. Nous les appelons variantes statiques, qui correspondent aux points de variation statiques. Nous avons donc l'ontologie du processus dans le domaine d'application choisi (en termes d'entités) et l'ontologie du processus d'affaire configuré (en termes de rôles) avec des points de variation non résolus, dits dynamiques. L'étape suivante est l'attribution des rôles, c'est à dire attribuer chaque entité au rôle correspondant. Selon l'algorithme décrit dans la section détaillée de cette étape.

Pendant l'exécution d'un processus d'affaires, selon l'approche proposée dans [ČIUK07a], en cas d'absence d'un service ou une ressource pour l'exécution d'une activité, ils n'ont pas proposé un autre choix. La différence majeur entre notre proposition

et celle dans [ČIUK07a], c'est que, en cas d'absence de ces services ou ressources, nous proposons de créer les services nécessaires à l'exécution de cette activité ou bien d'exécuter une autre activité avec un service ou une ressource d'exécution disponible. Pour cette raison, nous pouvons distinguer un autre type de points de variations qui se compose de points de variation dynamiques. Ils sont reliés à l'environnement d'exécution, donc la variabilité sera résolue durant la phase de l'exécution de processus.

Par la suite, nous devons définir l'ordre d'exécution des activités de ce processus en prenant en considération leur environnement d'exécution. Durant la phase d'ingénierie applicative du processus, nous pouvons définir d'autres points de variation dynamiques, qui seront résolus dans la phase d'exécution de ce processus. Un modèle de processus d'affaires exécutable est produit. Il est décrit par un langage tel que WSBPEL [BPEL07] et peut être exécuté par un système de gestion des Workflows**.**

Ainsi, l'approche proposée comporte trois phases : la phase d'ingénierie du domaine de processus (A), la phase de l'ingénierie applicative de processus (B) et la phase de l'exécution de processus (C). La Figure 4.1 illustre le flow d'activités principales durant chaque phase.

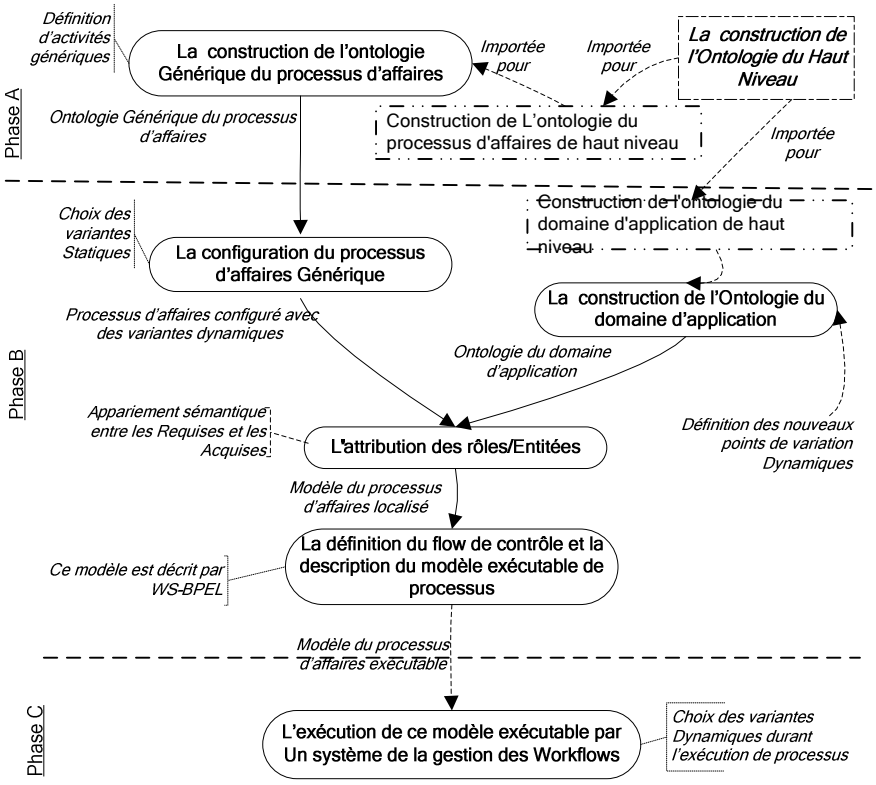

Figure 4.1. Flow d'activités principales durant chaque phase de l'approche proposée

Les activités principales de cette approche sont brièvement énumérées comme suit :

1) **La phase d'ingénierie du domaine de processus** (A), l'objectif de cette phase est la définition d'un domaine du processus d'affaires particulier. À cet égard, selon l'ingénierie de domaine, nous avons défini trois sous-activités qui sont l'analyse, la conception et l'implémentation d'un domaine de processus [ČIUK07a] :
   - **L'analyse du domaine de processus,** la construction du modèle des caractéristiques (**Feature Model**) à partie de la définition des points communs et les points de variations.
   - **La conception du domaine de processus,** la construction de l'ontologie du processus d'affaires générique à partir de l'ontologie du processus d'affaires de haut niveau (décrite en termes de rôles) (cf. section 3.4.3).
   - **L'implémentation du domaine de processus**, la création des assets (artefacts) réutilisables à partir du modèle de caractéristiques et l'ontologie de processus.

2) **La Phase d'ingénierie applicative du processus d'affaires (B),** la réutilisation des éléments déjà définis dans la phase précédente afin de générer le modèle du processus d'affaires localisé dans un domaine d'application choisi et par la suite un modèle exécutable de ce processus décrit par un langage tel que WSBPEL. Cette phase commence par deux activités parallèles :
   - **L'analyse du domaine applicative de processus**, le résultat de cette étape est une Ontologie du domaine d'application (décrite en termes d'entités). Elle est basée sur l'ontologie du domaine d'application de haut niveau (cf. section 3.4.2).
   - **La configuration du processus d'affaires générique,** le processus d'affaires configuré à partir du choix des variantes statiques.

   Pendant l'étape de la configuration, certains points de variabilité restent non résolus, ils seront résolus lors de la phase d'exécution du processus.

   Le processus d'affaires configuré et l'ontologie du domaine d'application sont des entrées pour la prochaine étape, qui est l'attribution des rôles.
   - **L'attribution des rôles,** Attribuer chaque entité au rôle correspondant. Selon l'algorithme décrit dans la section suivante.
   - **La définition de l'ordre d'exécution d'activités de ce processus** en prenant compte leur environnement d'exécution. Durant la phase d'ingénierie applicative du processus, nous pouvons définir d'autres points de variation dynamiques, qui seront résolus dans la phase d'exécution de ce processus.

- Un modèle de processus d'affaires exécutable est produit. Il est décrit par un langage tel que WSBPEL [BPEL07] et peut être exécuté par un système de gestion des Workflows**.**

3) La phase d'exécution du processus d'affaires (C), L'exécution de Modèle du processus d'affaires exécutable (décrite par WS BPEL) avec un système de gestion des Workflows, en choisissant les variantes dynamiques (activités à exécuter) selon les services et l'environnement requises.

3.3. Description détaillée de l'approche

La Figure 4.2 représente l'architecture détaillée de l'approche proposée, avec les différentes phases. Chaque phase est constituée d'un ensemble d'activités qui ont comme but d'arriver à leur objectif principal. De même, pour chaque activité un objectif ou résultat, qui est en relation avec les autres activités de même phase ou celles des autres phases. Ci-après, nous décrivons en détail cette architecture.

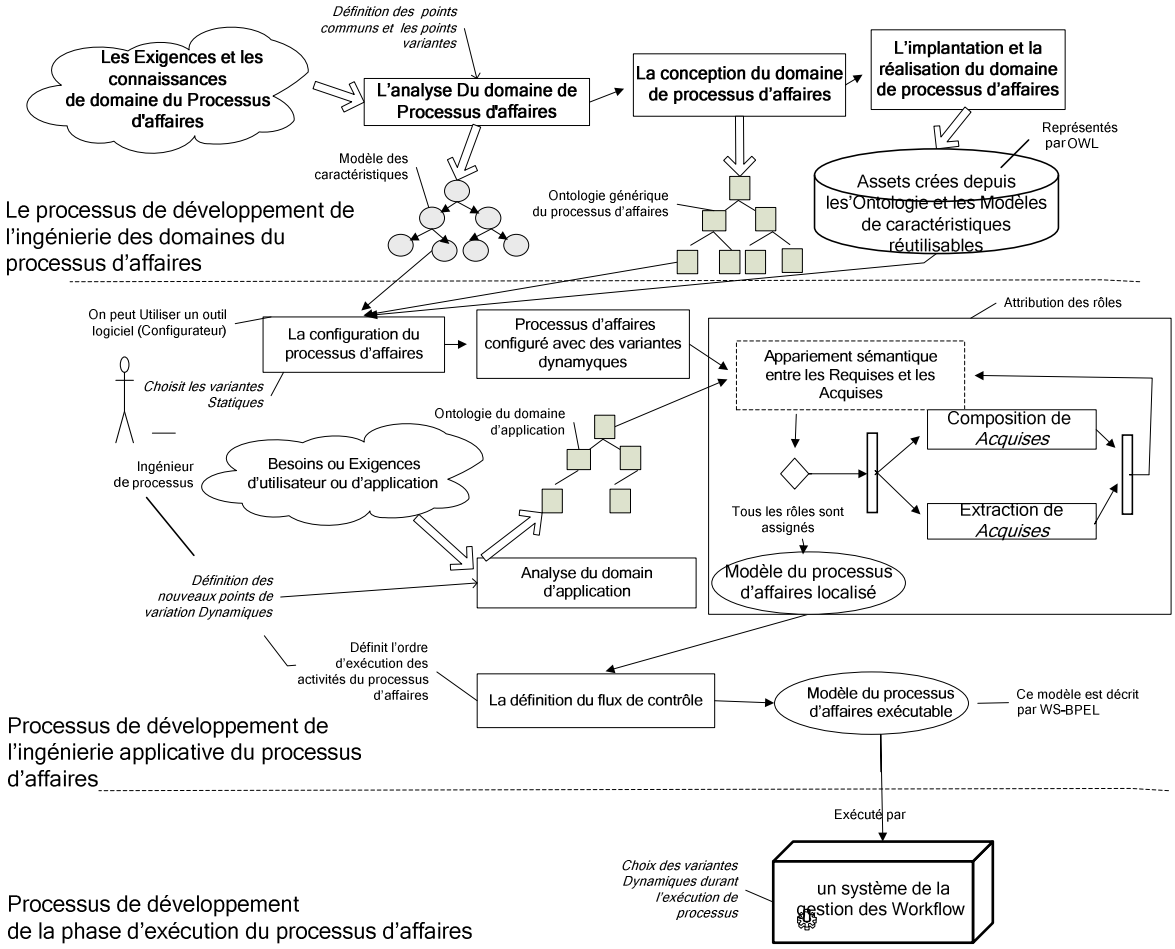

Figure 4.2. Architecture détaillée de l'approche proposée

3.3.1. La Phase d'ingénierie du domaine de processus d'affaires

L'objectif de cette phase est la définition d'un domaine du processus d'affaires particulier. À cet égard, selon l'ingénierie de domaine qui a été proposée dans [ČIUK07a], nous définissons trois sous-activités qui sont : l'analyse, la conception et l'implémentation d'un domaine de processus [ČIUK07a]. La Figure 4.3, présente ces trois activités.

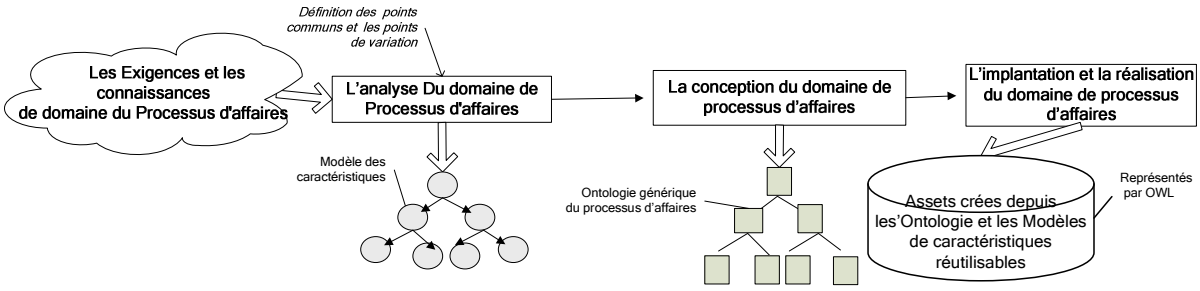

Figure 4.3. Le processus de développement de l'ingénierie de domaine du processus d'affaires (selon Donatas CIUKSYS [ČIUK07a])

3.3.1.1. L'analyse de domaine du processus d'affaires

Le but de ***l'analyse du domaine*** est :

- ✓ Étudier le domaine de processus d'affaires ;
- ✓ Identifier les éléments communs ou variables entre les processus de ce domaine.
- ✓ Parmi les méthodes d'analyse du domaine, la plus connue c'est « *Feature-Oriented Domain Analysis (FODA)* » [KANG90], où le domaine est décrit sous forme d'un modèle de caractéristiques (Feature Model).
- ✓ Un modèle des caractéristiques est une représentation explicite spécifiée sous forme d'arbre dont les nœuds représentent les caractéristiques de domaines (concepts de domaine) et les arcs spécifient les liens entre ces caractéristiques [KANG90].
- ✓ Donc le résultat de l'analyse du domaine est le modèle des caractéristiques, qui décrit les éléments communs et variables entre les processus de ce domaine. Ces processus constituent une famille de processus.

Exemple

On prend les concepts d'activité du processus de commande (Figure 4.4, décrite en utilisant UML) suivant :

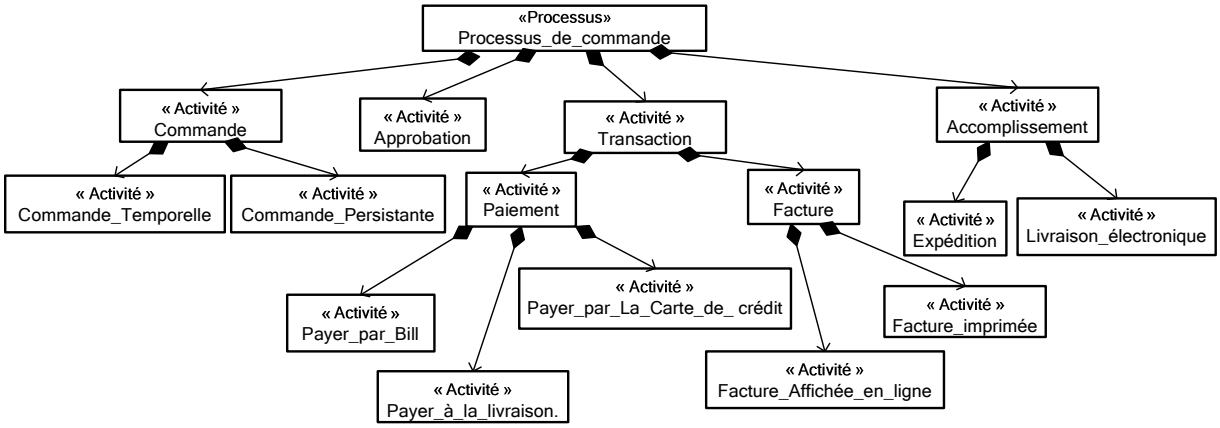

Figure 4.4. Concepts d'activité du processus de commande

On peut représenter aussi ces activités par un modèle des caractéristiques (Figure 4.5), pour bien décrire les éléments (Activités) communs et variables au sein d'une famille du processus de Commande.

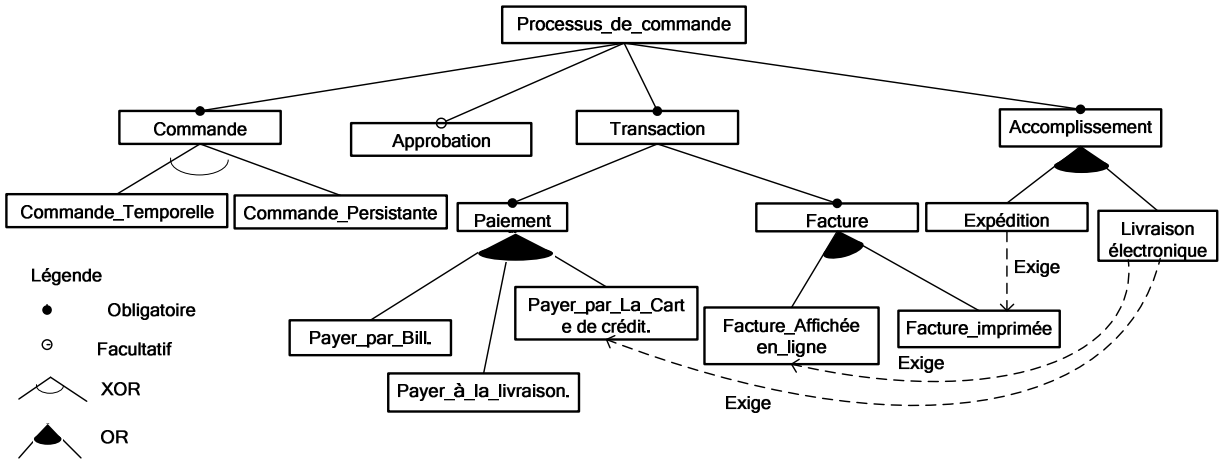

Figure 4.5. Modèle de caractéristiques du processus de commande **(**Selon Donatas CIUKSYS [ČIUK07a])

3.3.1.2. La conception du domaine de processus

Le but de cette étape, est la construction de l'ontologie du processus d'affaires générique à partir de l'ontologie du processus d'affaires de haut niveau (décrite en termes de rôles) (cf. section 3.4.3).

- ✓ Raffine les termes définis par le modèle des caractéristiques ;
- ✓ Ajoute aux connaissances l'épistémique d'ontologie ;
- ✓ L'ontologie produite est basée sur l'ontologie du processus d'affaires de haut niveau, qui définit des concepts tels que : *activité, entrée, sortie, ressource, requise, etc*. Ces concepts sont exigés pour modéliser n'importe quel processus d'affaires générique. Un fragment d'ontologie de l'activité «*Payer_Par_La_Carte_Crédit* » et un autre de

l'activité «*Commande_Temporelle* » du processus de commande, sont présenté dans les figures : Figure 4.6 et Figure 4.7 respectivement.

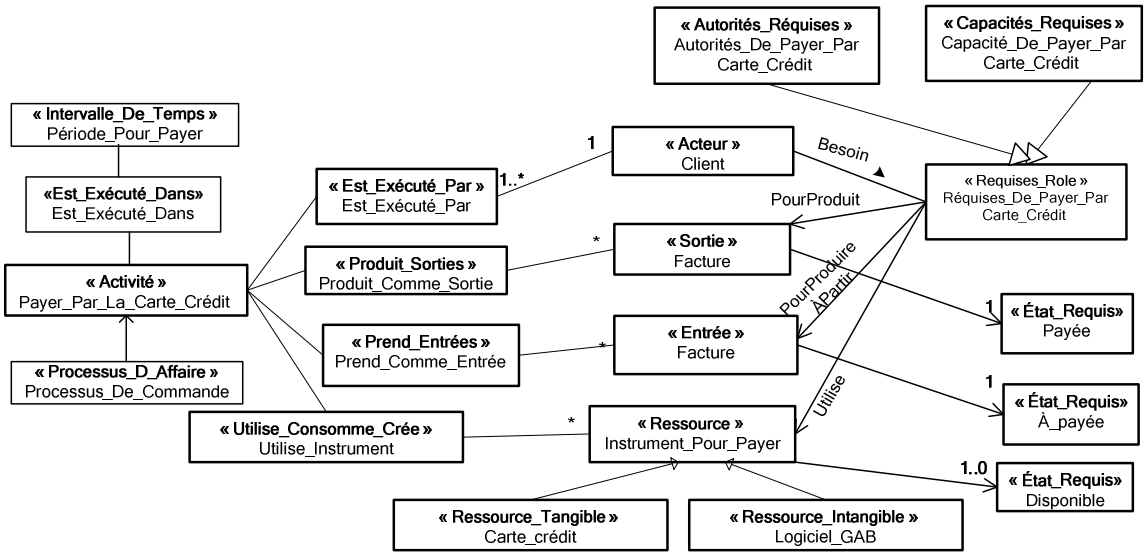

Figure 4.6. Un fragment d'ontologie de l'activité «*Payer_Par_La_Carte_Crédit* »

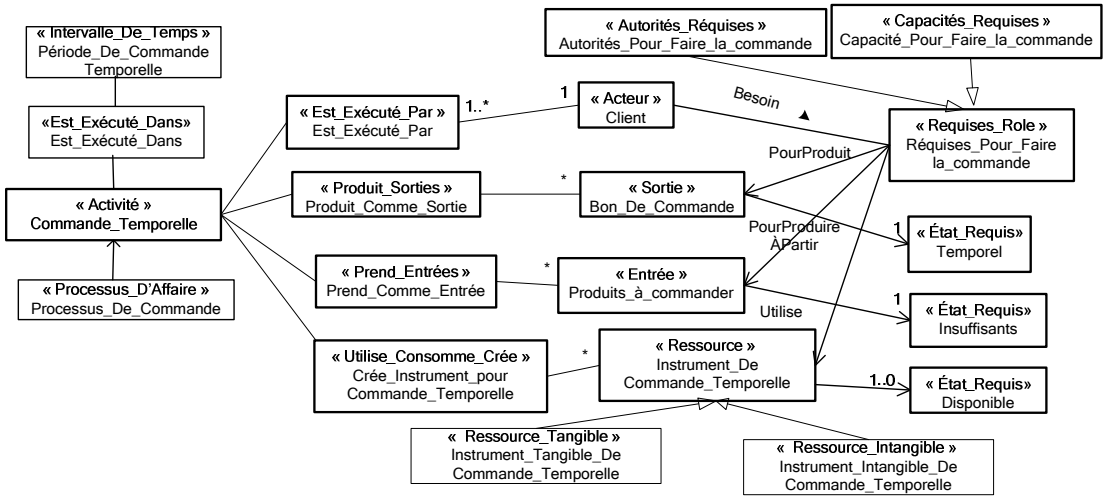

Figure 4.7. Un fragment d'ontologie de l'activité «*Commande_Temporelle* »

L'ontologie du processus d'affaires de haut niveau (décrite en termes des rôles), décrite en détail dans la section d'ontologie du processus d'affaires de haut niveau.

3.3.1.3. L'implémentation du domaine de processus,

Le but de cette étape, est la création des assets (artefacts) réutilisables à partir du modèle de caractéristiques et de l'ontologie de processus.

- ✓ Empaqueter le modèle des caractéristiques et l'ontologie de processus dans un paquet d'assets réutilisables.

- ✓ L'ontologie de processus est représentée comme assets réutilisable, en utilisant Web-Ontology Langage (OWL) à base de la Logique de Description (DL), avec l'outil Protégé.

Figure 4.8, Figure 4.9 et Figure 4.10 représentent la construction d'ontologie du processus de commande à partir d'ontologie du processus d'affaires de haut niveau.

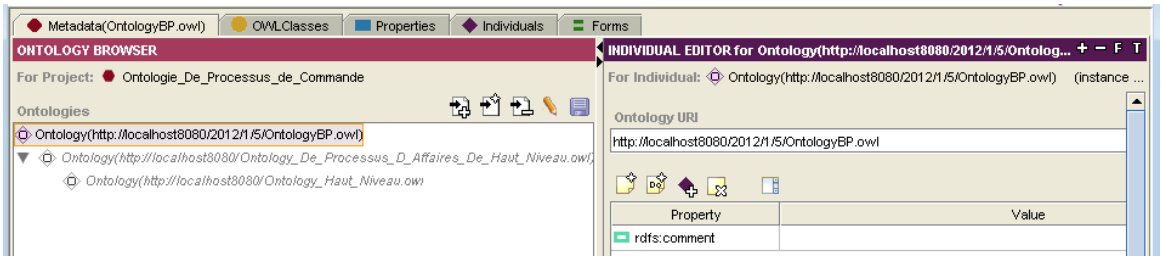

Figure 4.8. Importer l'ontologie du processus d'affaires de haut niveau pour construire l'ontologie du processus de commande

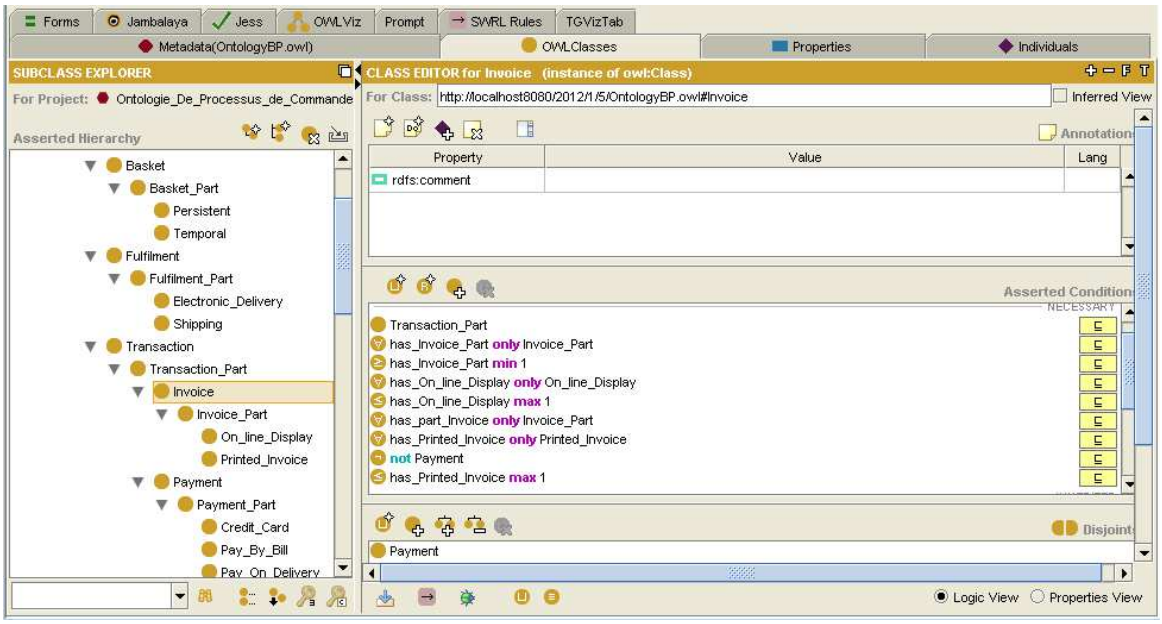

Figure 4.9. Une partie de vue logique de l'activité 'Invoice' (Facture dans le modèle de caractéristiques du processus de commande.)

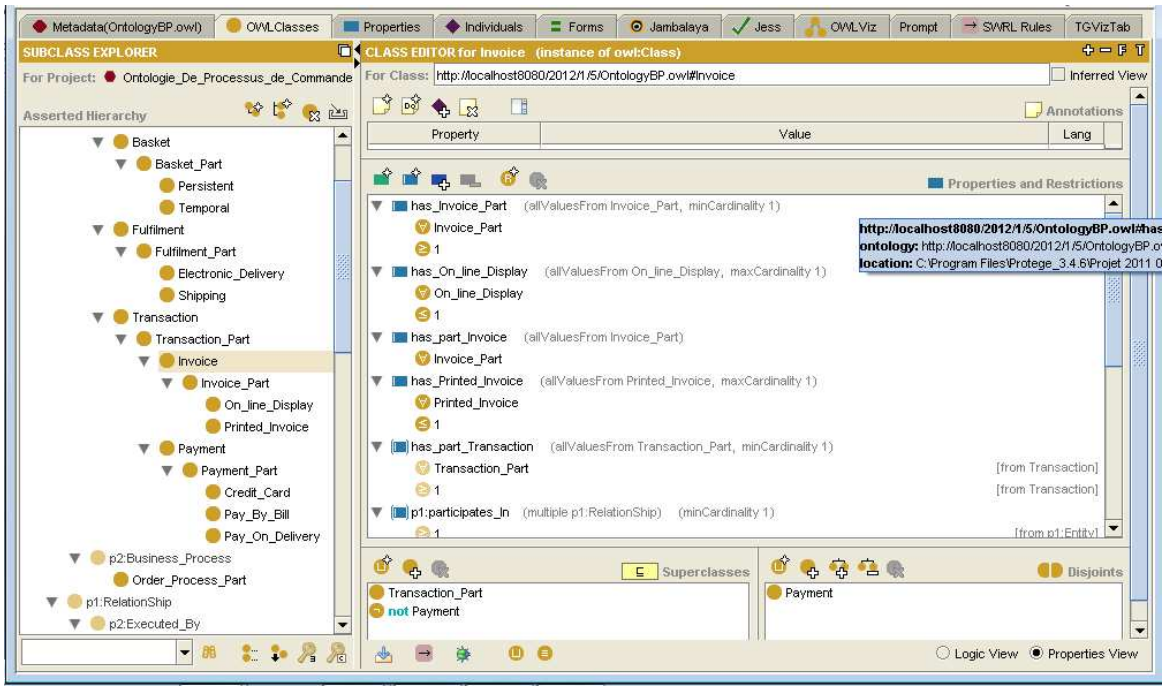

Figure 4.10. Une partie des propriétés de l'activité 'Invoice' (Facture dans le modèle de caractéristiques du processus de commande)

3.3.2. La Phase d'ingénierie applicative du processus d'affaires

La réutilisation des éléments, déjà définis dans la phase précédente. Cela pour générer le modèle du processus d'affaires localisé dans un domaine d'application choisi et par la suite un modèle exécutable de ce processus décrit par un langage tel que WSBPEL. La Figure 4.11 représente le processus de développement de l'ingénierie applicative du processus d'affaires.

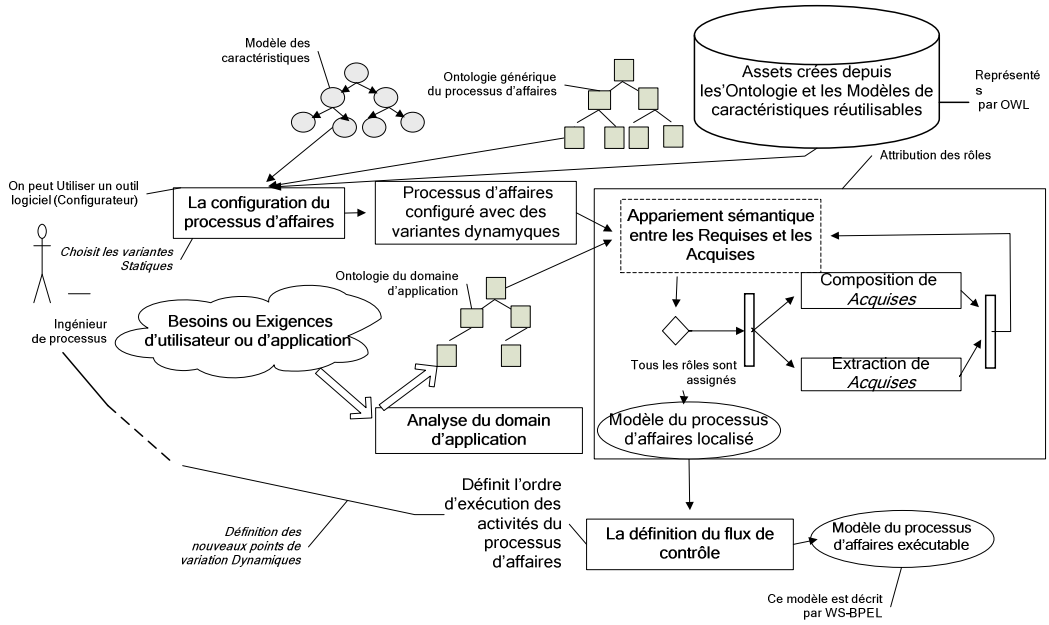

Figure 4.11. Le processus de développement de l'ingénierie applicative du processus d'affaires **(**Selon Donatas CIUKSYS [ČIUK07a])

Cette phase commence par deux activités parallèles :

3.3.2.1. L'analyse de domaine applicative du processus d'affair*es*

Le résultat de cette étape est une Ontologie du domaine d'application (décrite en termes d'entités). Elle est basée sur l'ontologie du domaine d'application de haut niveau (cf. la section 3.4.2).

JIANQI YU [JIAN10], voit l'apparition de trois types d'exigences durant l'analyse des exigences d'application, de même pour l'analyse de domaine applicative du processus d'affaires, on distingue trois types d'exigences :

- Les exigences qui sont incluses dans les exigences du domaine classifiées lors de l'ingénierie du domaine.
- Les exigences spécifiques qui sont prévues dans l'analyse du domaine sous forme de variabilités. Celles-ci donnent lieu à des configurations particulières du processus d'affaires, via les points de variabilité, et des composants réutilisables.
- Les exigences spécifiques qui n'ont pas été prévues dans l'analyse du domaine. Comme l'a souligné JIANQI [JIAN10], ces exigences posent généralement problème. Elles constituent des demandes ou des contraintes qui donnent lieu à des configurations particulières, non guidées, des artefacts réutilisables ou à de nouvelles implantations.

Donc, Nous tirerons profit du modèle de domaine existant et nous décrirons les besoins du client utilisant les caractéristiques à partir du modèle de domaine.

Cependant les nouvelles exigences des clients qui ne figurent pas dans le modèle de domaine nécessitent un développement personnalisé. Ces nouvelles exigences devraient également être renvoyées à l'ingénierie de domaine pour affiner et étendre les assets réutilisables.

Le résultat de cette étape est une Ontologie du domaine d'application (décrit en termes d'entités). Elle est basée sur l'ontologie du domaine d'application de haut niveau (cf. la section 3.4.2). Nous donnerons un exemple dans le chapitre de l'étude de cas.

Donc, durant cette étape, nous pouvons aussi définir d'autres points de variation dynamiques, qui seront résolus dans la phase d'exécution de ce processus. La définition de ces points de variation peut être faite durant l'implémentation de ce processus.

3.3.2.2. La configuration du processus d'affaires générique

Le processus d'affaires est configuré à partir du choix des variantes statiques.

- La réutilisation des connaissances du processus exige que le processus soit configuré, en tenant compte particularités du domaine d'application choisi. Le but principal de configuration du processus est de guider l'ingénieur de processus à travers la sélection de variantes jusqu'à ce que toutes les variabilités, ici la variabilité correspondante aux point de variation statique, soient résolues. La configuration devrait également empêcher la violation des dépendances entre les variantes du modèle de caractéristiques ; Traiter les activités comme des composants et les dépendances comme contraintes, nous pouvons définir la tâche qui devrait être réalisée durant la configuration comme un problème de conception de la configuration [ČIUK07a].
- Pour résoudre le problème de la configuration de processus, nous suivrons l'approche à base de la Logique de Description (DL) [BAAD03] décrite dans [ČIUK07a], parce que nous définissons toutes les ontologies proposées utilisant l'Ontology Web Langage (OWL) à base de la Logique de Description (DL). Une solution au problème de configuration peut être proposée pour être un modèle de la base de connaissance donnée de type Logique de Description (DL) [BAAD03].
- Dans ce cas, l'espace de configuration est défini par le TBox qui décrit la hiérarchie des concepts et la hiérarchie des rôles. ABox initial inclut exclusivement des variantes obligatoires (points communs). Utilisant cette approche, le choix des variantes optionnelles ou alternatives nécessite que ABox soit augmenté avec les individus correspondants et qu'un raisonneur OWL DL (nous utiliserons dans notre cas d'étude le raisonneur Racer) soit sollicité pour vérifier si ABox est conforme à l'espace de configuration. Dans le cas contraire le choix doit être défait. Il est important, dans notre cas, que le problème de processus de la configuration soit résolu en mode interactif et que le processus de résolution des problèmes soit itératif. Pour assurer les dépendances entre les variantes, nous avons utilisé les règles de Horn [MOTI05, ČIUK07a].

Exemple :

Voici un exemple des espaces de configuration (TBox) pour les problèmes de configuration dans le processus de commande, qui sont présentés par le modèle des caractéristiques dans la Figure 4.5 :

Partie_D_Accomplissement ⊆ (Expédition U Livraison_Electronique)
⊆ =1 Partie_De ∩ Partie_De.Accomplissement
Expédition ⊆ ┐Livraison_Electronique (01)
Livraison_Electronique ⊆ ┐Expédition
A_ Une_ Partie_D_Accomplissement ⊆ A_une_partie
A_Une_Expédition ⊆ A_Une_Partie_D_Accomplissement (02)
A_Une_Livraison_Electronique ⊆ A_Une_Partie_D_Accomplissement
T⊆ ∀ A_Une_Partie_D__Accomplissement.Partie_D_Accomplissement
T ⊆ ∀ A_Une_Expédition.Expédition (03)
T ⊆ ∀ A_Une_Livraison_Electronique.Livraison_Electronique
Accomplissement ⊆ Partie_Processus_de_commande (04)
∀ A_Une_Partie. Partie_D_Accomplissement
∩ ≥ 1 A_Une_Partie_D_Accomplissement
∩ ≤ 1 A_Une_Expédition
∩ ≤ 1 A_Une_Livraison_Electronique

Tableau 4.1 L'espace de configuration (TBox) pour le problème de configuration Accomplissement**(**Selon Donatas CIUKSYS **[ČIUK07a]**)

Dans le **Tableau 4.1**, qui représente l'espace de configuration (TBox) pour le problème de configuration Accomplissement. Le **TBox** commence par donner ce qu'on appelle l'axiome de couverture pour le concept ***Partie_D_Accomplissement***, aussi bien qu'exiger de ***Partie_D_Accomplissement*** d'être une et une seule partie de concept ***Accomplissement.*** Les axiomes additionnels assurent la disjonction des concepts ***Expédition*** *et* ***Livraison_Électronique*** (01). Alors **TBox** présente la hiérarchie de rôle, déclarant que le rôle ***a_une_Partie_D_Accomplissement*** est un sous rôle de rôle ***a_une_Partie,*** et les deux rôles ***a_une_Expédition*** et ***a_une_Livraison_Électronique*** sont des sous rôles de rôle ***a_une_Partie_D_accomplissement*** (02).

La suite de **TBox** est une série de restrictions imposantes sur les rôles (03). Enfin des exigences sont énoncées pour le concept ***Accomplissement*** (04). Il doit être une ***Partie du processus de commande***, toutes les Parties qu'il a doivent être des instances de concept ***Partie_D_Accomplissement***, et il doit avoir au moins une ***Partie_D_Accomplissement***, au plus une ***Expédition,*** et au plus une ***Livraison_Électronique***.

Partie_De_Transaction ⊆ (Paiement ∩ Facture)	
⊆ =1 Partie_De ∩ Partie_De.Transaction	
Paiement ⊆ ¬ Facture	(05)
Facture ⊆ ¬ Paiement	
A_Une_ Partie_De_Transaction ⊆ A_Une_Partie	(06)
A_Un_Paiement ⊆ A_Une_Partie_De_Transaction	
A_Une_Facture ⊆ A_Une_Partie_De_Transaction	
T ⊆ ∀ A_Une_ Partie_De_Transaction. Partie_De_Transaction	
T ⊆ ∀ A_Un_Paiement.Paiement	(07)
T ⊆ ∀ A_Une_Facture. Facture	
Transaction ⊆ Partie_Processus_de_commande	
∀ A_Une_Partie. Partie_De_Transaction	(08)
∩ ≥ 1 A_Une_Partie_De_Transaction	
∩ ≤ 1 A_Un_Paiement	
∩ ≤ 1 A_Une_Facture	

Tableau 4.2 L'espace de configuration (TBox) pour le problème de configuration Transaction

Dans le **Tableau 4.2**, qui représente l'espace de configuration (TBox) pour le problème de configuration ***Transaction.*** Le **TBox** continue à donner également l'axiome de couverture pour le concept ***Partie_De_Transaction***, aussi bien qu'exiger de ***Partie_De_Transaction*** d'être une et une seule partie de concept ***Transaction***. Les axiomes additionnels assurent la disjonction des concepts ***Paiement*** *et* ***Facture*** (05). Alors **TBox** présente la hiérarchie de rôle, déclarant que le rôle ***a_une_Partie_De_Transaction*** est un sous rôle de rôle ***a_une_Partie***, et les deux rôles ***a_un_Paiement*** et ***a_une_Facture*** sont des sous rôles de rôle ***a_une_Partie_De_Transaction*** **(06).**

La suite de **TBox** est une série de restrictions imposantes sur les rôles (07). Enfin des exigences sont énoncées pour le concept ***Transaction*** (08). Il doit être une ***Partie du processus de commande***, toutes les Parties qu'il a doivent être des instances de concept ***Partie_De_Transaction***, et il doit avoir au moins une ***Partie_De_Transaction***, au plus un ***Paiement***, et au plus une ***Facture***.

Partie_De_Facture ⊆ (Facture_Affichée_en_ligne U Facture_imprimée)	
⊆ =1 Partie_De ∩ Partie_De. Facture	
Facture_Affichée_en_ligne ⊆ ┐ Facture_imprimée	(09)
Facture_imprimée ⊆ ┐ Facture_Affichée_en_ligne	
A_Une_Partie_De_Facture ⊆ A_Une_Partie	(10)
A_Une_Facture_Affichée_en_ligne ⊆ A_Une_Partie_De_Facture	
A_Une_Facture_imprimée ⊆ A_Une_Facture_Partie	
T ⊆ ∀ A_Une_Partie_De_Facture. Partie_De_Facture	(11)
T⊆∀A_Une_Facture_Affichée_en_ligne. Facture_Affichée_en_ligne	
T ⊆ ∀ A_Une_Facture_imprimée. Facture_imprimée	
Facture ⊆ Partie_Processus_de_commande	
∀ A_Une_Partie. Partie_De_Facture	
∩ ≥ 1 A_Une_Partie_De_Facture	(12)
∩ ≤ 1 A_Une_Facture_Affichée_En_ligne	
∩ ≤ 1 A_Une_Facture_Imprimée	

Tableau 4.3 L'espace de configuration (TBox) pour le problème de configuration Facture

Dans le **Tableau 4.3**, qui représente l'espace de configuration (TBox) pour le problème de configuration ***Facture***, **TBox** continue à donner aussi l'axiome de couverture pour le concept ***Partie_De_Facture***, aussi bien qu'exiger de ***Partie_De_Facture*** d'être une et une seule partie de concept ***Facture***. Les axiomes additionnels assurent la disjonction des concepts ***Facture_Affichée_en_ligne*** *et* ***Facture_imprimée*** (09). Alors **TBox** présente la hiérarchie de rôle, déclarant que le rôle **a_une_*Partie_De_Facture*** est un sous rôle de rôle ***a_une_Partie***, et les deux rôles ***a_une_Facture_Affichée_en_ligne*** et ***a_une_Facture_imprimée*** sont des sous rôles de rôle **a_une_*Partie_De_Facture* (10).**

La suite de **TBox** est une série de restrictions imposantes sur les rôles (11). Enfin des exigences sont énoncées pour le concept ***Facture*** (12). Il doit être une ***Partie du processus de commande***, toutes les Parties qu'il a doivent être des instances de concept ***Facture***, et il doit avoir au moins une ***Partie_De_Facture***, au plus une ***Facture_Affichée_en_ligne***, et au plus une ***Facture_imprimée***.

Partie_De_Paiement ⊆ (Payer_par_Bill U Payer_à_la_livraison U Payer_par_Carte_de_crédit)	(13)
⊆ =1 Partie_De ∩ Partie_De.Paiement	
Payer_par_Bill ⊆ ┐ (Payer_à_la_livraison∩ Payer_par_Carte_de_crédit)	
Payer_A_la_livraison ⊆ ┐ (Payer_par_Bill ∩ Payer_par_Carte_de_crédit)	
Payer_par_Carte_de_crédit ⊆ ┐ (Payer_par_Bill ∩ Payer_A_la_livraison)	
A_Une_Partie_De_Paiement ⊆ A_Une_Partie	
A_Un_Paiement_par_Bill⊆ A_Une_Partie_De_Paiement	(14)
A_Un_Paiement_A_la_livraison ⊆ A_Une_Partie_De_Paiement	
A_Un_Paiement_par_Carte_de_crédit. ⊆ A_Une_Partie_De_Paiement	
T ⊆ ∀ A_Une_Partie_De_Paiement. Partie_De_Paiement	
T ⊆ ∀ A_Un_Paiement_par_Bill. Payer_par_Bill	(15)
T ⊆ ∀ A_Un_Paiement_A__la_livraison. Payer_A__la_livraison	
T ⊆ ∀ A_ Un_Paiement_par_Carte_de_crédit.Payer_par_Carte_de_crédit.	
Paiement ⊆ Partie_Processus_de_commande	
∀ A_Une_Partie. Partie_De_Paiement	
∩ ≥ 1 A_Une_Partie_De_Paiement	
∩ ≤ 1 A_Un_Paiement_par_Bill	(16)
∩ ≤ 1 A_Un_Paiement_A__la_livraison	
∩ ≤ 1 A_Un_Paiement_par_Carte_de_crédit.	

Tableau 4.4 L'espace de configuration (TBox) pour le problème de configuration Paiement

Dans le **Tableau 4.4**, qui représente l'espace de configuration (TBox) pour le problème de configuration **Paiement**, **TBox** continue à donner aussi l'axiome de couverture pour le concept ***Partie_De_Paiement***, aussi bien qu'exiger de ***Partie_De_Paiement*** d'être une et une seule partie de concept ***Paiement***. Les axiomes additionnels assurent la disjonction de concepts ***Payer_par_Bill***, ***Payer_à_la_livraison*** et ***Payer_par_Carte_de_crédit*** (13). Alors **TBox** présente la hiérarchie de rôle, déclarant que le rôle ***a_une_Partie_De_Paiement*** est un sous rôle de rôle ***a_une_Partie***, et les trois rôles ***a_Un_Paiement _par_Bill , A_Un_Paiement_par_Carte_de_crédit*** et ***a_Un_Paiement _à_la_livraison*** sont des sous rôles de rôle ***a_une_Partie_De_Paiement*** (14).

La suite de **TBox** est une série de restrictions imposantes sur les rôles (15). Enfin des exigences sont énoncées pour le concept ***Paiement*** (16). Il doit être une *Partie du*

processus de commande, toutes les Parties qu'il a doivent être des instances de concept ***Partie_De_Paiement***, et il doit avoir au moins une ***Partie_De_Paiement***, au plus un ***Paiement_par_Bill***, au plus un ***Paiement_à_la_livraison et*** au plus un ***Paiement_par_Carte_de_crédit.***

Partie_De_Commande ⊆ (Commande_Temporelle U Commande_Persistante)	
⊆ =1 Partie_De ∩ Partie_De.Commande	(17)
Commande_Temporelle ⊆ ¬ Commande_Persistante	
Commande_Persistante ⊆ ¬ Commande_Temporelle	
A_Une_Partie_De_Commande ⊆ A_Une_Partie	(18)
A_Une_Commande_Temporelle⊆ A_Une_Partie_De_Commande	
A_Une_Commande_Persistante ⊆ A_Une_Partie_De_Commande	
T ⊆ ∀ A_Une_Partie_De_Commande. Partie_De_Commande	(19)
T ⊆∀ A_Une_Commande_Temporelle. Commande_Temporelle	
T ⊆ ∀ A_Une_Commande_Persistante. Commande_Persistante	
Commande ⊆ Partie_Processus_de_commande	
∀ A_Une_Partie. Partie_De_Commande	
∩ ≥ 1 A_Une_Partie_De_Commande	(20)
∩ ≤ 1 A_Une_Commande_Temporelle	
∩ ≤ 1 A_Une_Commande_Persistante	

Tableau 4.5 L'espace de configuration (TBox) pour le problème de configuration Commande

Dans le **Tableau 4.5**, qui représente l'espace de configuration (TBox) pour le problème de configuration **Commande**, le **TBox** continue à donner aussi l'axiome de couverture pour le concept ***Partie_De_Commande***, aussi bien qu'exiger de ***Partie_De_Commande*** d'être une et une seule partie de concept ***Commande***. Les axiomes additionnels assurent la disjonction de concepts ***Commande_Temporelle*** *et* ***Commande_Persistante*** (17). Alors **TBox** présente la hiérarchie de rôle, déclarant que le rôle ***a_une_Partie_De_Commande*** est un sous rôle de rôle ***a_une_Partie,*** et les deux rôles ***a_une_Commande_Temporelle*** *Et* ***a_une_Commande_Persistante*** sont des sous rôles de rôle ***a_une_Partie_De_Commande*** (18).

La suite de **TBox** est une série de restrictions imposantes sur les rôles (19). Enfin des exigences sont énoncées pour le concept ***Commande*** (20). Il doit être une *Partie du processus de commande*, toutes les Parties qu'il a doivent être des instances de concept

Partie_De_Commande, et il doit avoir au moins une ***Partie_De_Commande***, au plus une ***Commande_Temporelle*** , et au plus une ***Commande_Persistante***.

Dans notre exemple l'ABox initial est très simple : ***A = {c : Commande, a : Accomplissement, t : Transaction, p : Paiement, f : Facture}.*** Si l'utilisateur choisit les variantes : ***Commande_Persistante*** (résout la variabilité actuelle dans l'activité Commande), ***Livraison_Electronique*** (résout la variabilité actuelle dans l'activité Accomplissement), ***Facture_Affichée_en_ligne*** (résout la variabilité actuelle dans l'activité facture)***, Paiement_par_Carte_de_crédit*** (résout la variabilité actuelle dans l'activité Paiement).

ABox deviendra:

A = {c : *Commande,* ***cp :*** *Commande_Persistante, a_une_Commande_Persistante* ***(c ; cp)****,* ***a :*** *Accomplissement,* ***le :*** *Livraison_Electronique ; a_une_Livraison_Electronique* ***(a ; le)****,* ***t :*** *Transaction,* ***p :*** *Paiement,* ***f*** *: Facture, a_une_Facture* ***(t ; f)****, a_un_Paiement* ***(t ; p)****,* ***fael :*** *Facture_Affichée_en_ligne, a_une_Facture_Affichée_en_ligne* ***(f ; fael)****,* ***ppcc :*** *Paiement_par_Carte_de_crédit, a_un_Paiement_par_Carte_de_crédit****(p ;ppcc)}.***

La base de connaissances devrait être testée pour assurer sa cohérence et, par conséquent le choix de l'utilisateur serait validé.

Notre base de connaissance manque encore de dépendances entre les variantes. Comme nous pouvons voir dans le modèle des caractéristiques (Figure 4.5), le choix de variante *Livraison_Electronique* nécessite de choisir la variante *Facture_Affichée_en_ligne*. Pour tenir une telle contrainte dans notre base de connaissances nous avons besoin de règles.

L'un des formalismes basés sur les règles, qui sont décidables et souvent utilisés, sont les règles de Horn à libres fonctions [MOTI05].

Exemple de la règle de Horn :

a_un_Parent(x; y) et *a_un_frère(y; z)→ a_un_Oncle(x; z).*

Comme il était confirmé par caplinska, le langage de description OWL-DL a été étendu avec des règles de Horn dans [HORR04], mais cette extension est indécidable [HORR04]. Dans [MOTI05] une combinaison décidable OWL-DL avec ces règles a été proposée, où la décidabilité est obtenue en limitant les règles qu'on appelle les règles DL-Safe. Dans les règles de DL-safe, les concepts et les rôles ont permis de se produire dans

les corps et les têtes des règles en tant qu'attributs unaires et binaires, mais chaque variable d'une règle est exigée d'être liée seulement aux individus présents dans ABox [Motik05]. En d'autres termes, la règle ci-dessus d'oncle sera DL-safe, si le raisonnement sera exécuté seulement sur l'ensemble des individus d'ABox.

Donc, nous allons ajouter des règles dans notre base de connaissances. La règle selon laquelle le choix de la variante *Livraison Électronique* nécessite le choix des variantes *Facture_Affichée_En_Ligne* et *Paiement_par_Carte_de_crédit* , peut être écrite comme suit :

a_une_Livraison_Électronique (a; le) → a_une_Facture_Affichée_En_Ligne (f; fael) & a_un_Paiement_par_Carte_de_crédit (p; ppcc) **(I)**

Ici *a, le, f, fael, p, ppcc* sont des individus des concepts : *Accomplissement, Livraison_Électronique, Facture, Facture_Affichée_En_Ligne, Paiement, Paiement_par_Carte_de_crédit,* respectivement.

Si notre choix est fait sur la variante *Expédition (e),* la règle selon ce choix nécessite la sélection des variantes *Facture_Imprimée (fi)* et *Paiement_à_la_Livraison (pl)* peut être écrite comme suit :

a_une_Expédition (a; e)→ a_une_Facture_Imprimée (f ; fi) & a_un_Paiement_à_la_Livraison (p; pl) **(II)**

De même, toutes les dépendances requises peuvent être codées en tant que règles. Les dépendances mutuellement exclusives sont codées en ajoutant la négation à tous les rôles dans la tête de règles. La présence des règles dans la base de connaissances signifie que le raisonneur doit être capable de les comprendre et les exécuter au besoin.

Pour l'implémentation de cette partie, l'outil Protégé permet de définir l'espace de configuration (TBox) pour un problème de configuration en ajoutant des conditions sur les différent concepts qui appartiennent au problème de configuration. Ici, on prend par exemple le problème de configuration *Paiement* de modèle des caractéristiques du processus de commande. Les figures Figure 4.12, Figure 4.13, Figure 4.14, Figure 4.15 et Figure 4.16 représentent les conditions ajoutées sur les concepts : *Paiement, Partie_De_Paiement, Paiement_Carte_crédit, Paiement_par_Bill* et *Paiement_à_la_Livraison* respectif.

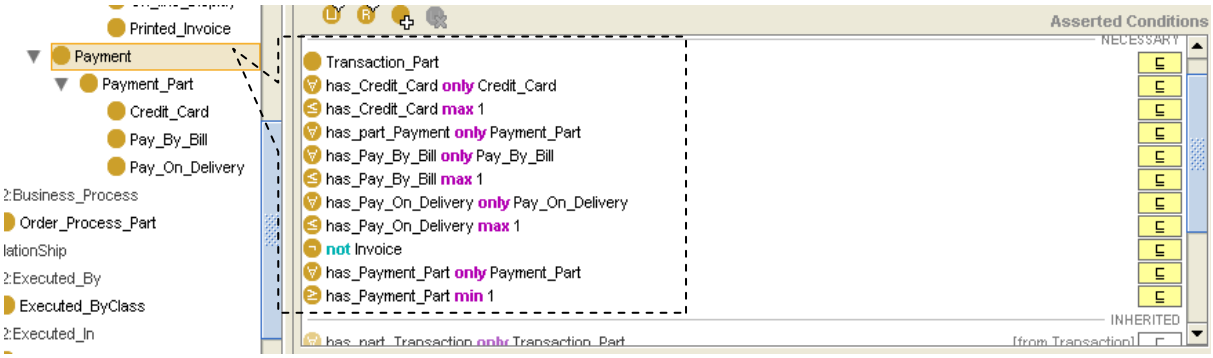

Figure 4.12. Conditions ajoutées sur le concept *Paiement* (Payment) avec l'outil Protégé

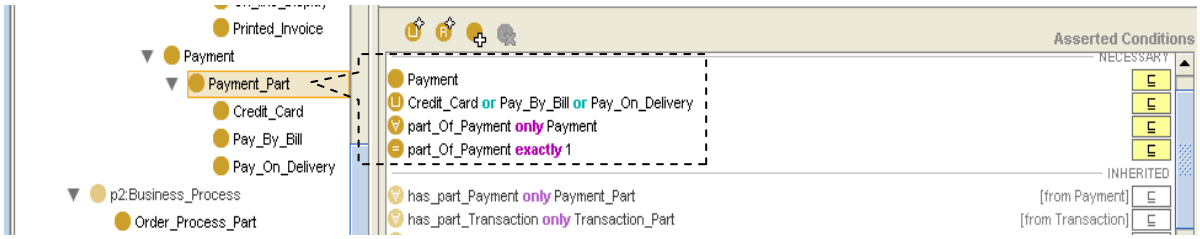

Figure 4.13. Conditions ajoutées sur le concept *Partie_De_Paiement* (Payment_Part) avec l'outil Protégé

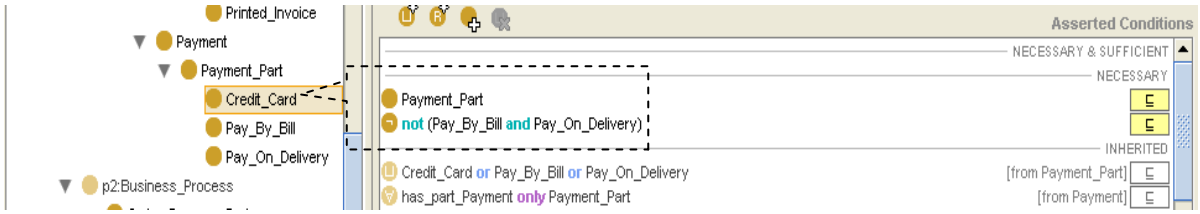

Figure 4.14. Conditions ajoutées sur le concept Payer_Carte_crédit (Credit_Card) avec l'outil Protégé

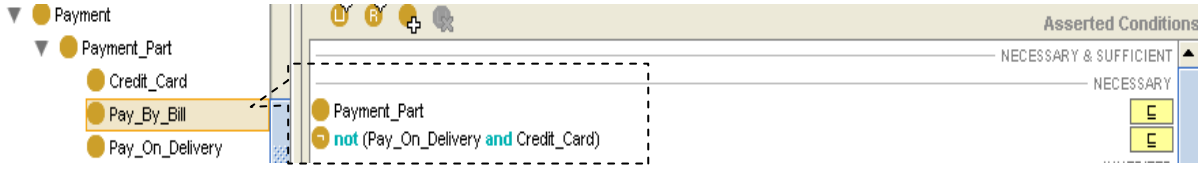

Figure 4.15. Conditions ajoutées sur le concept Payer_par_Bill (Pay_By_Bill) avec l'outil Protégé

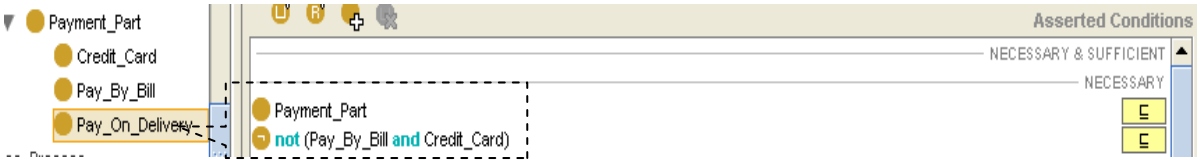

Figure 4.16. Conditions ajoutées sur le concept Payer_à_la_Livraison (Pay_On_Delivery) avec l'outil Protégé

Pour l'implémentation d'ABox, il inclut exclusivement les variantes obligatoires. C'est-à-dire on doit instancier seulement ces variantes, ensuite et comme nous l'avons mentionné auparavant, le choix des variantes optionnelles ou alternatives nécessite que ABox soit augmenté avec les individus correspondants et les relations qui existent entre eux. Cela veut dire, qu'il faut instancier aussi et seulement les variantes optionnelles ou alternatives choisies durant la résolution de variabilités. Sans oublier d'instancier aussi les relations (Rôles) entre ces variantes. La Figure 4.17, représente le choix de la variante *Carte_Crédit* dans le problème de configuration *Paiement* avec l'outil Protégé.

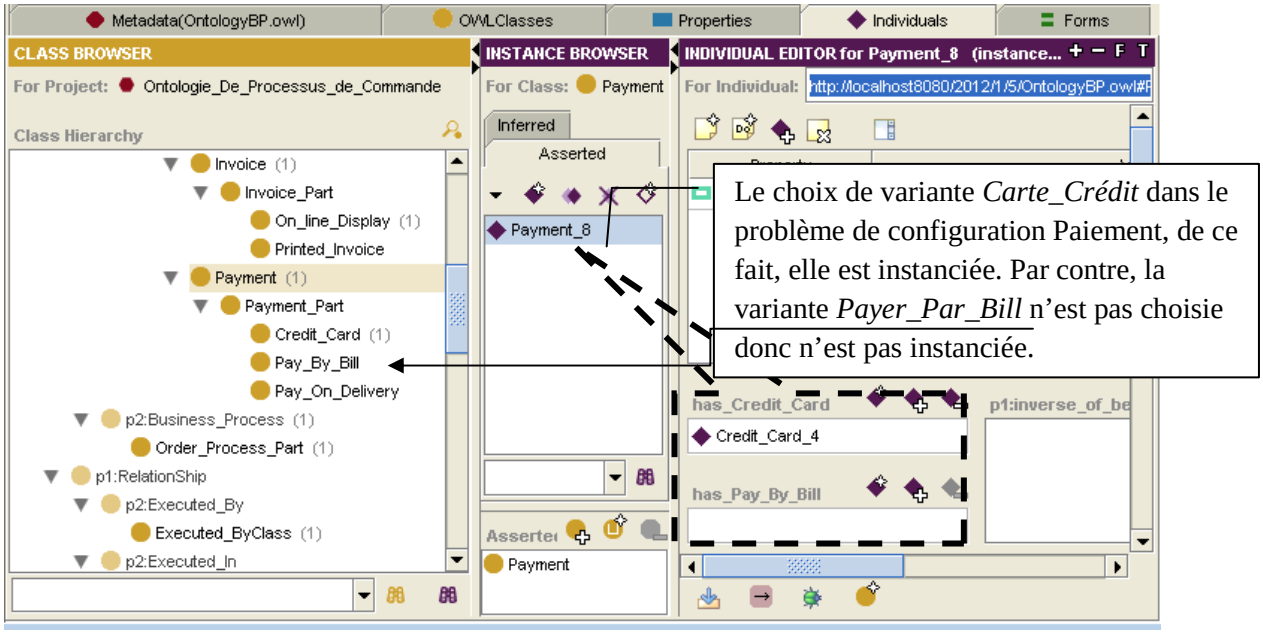

Figure 4.17. Choix de variante *Carte_Crédit* dans le problème de configuration *Paiement* avec l'outil Protégé

Les variantes ***Payer_Par_Bill*** et ***Payer_à_La_Livraison*** non ne sont pas choisis, de ce fait, ne sont pas instanciés avec l'outil Protégé.

Pour les règles qui assurent les dépendances entre les variantes, protégé offre aussi un plugin ***SWRL Rules*** pour construire ces règles, l'implémentation de la règle (I) est illustrée dans la Figure 4.18 :

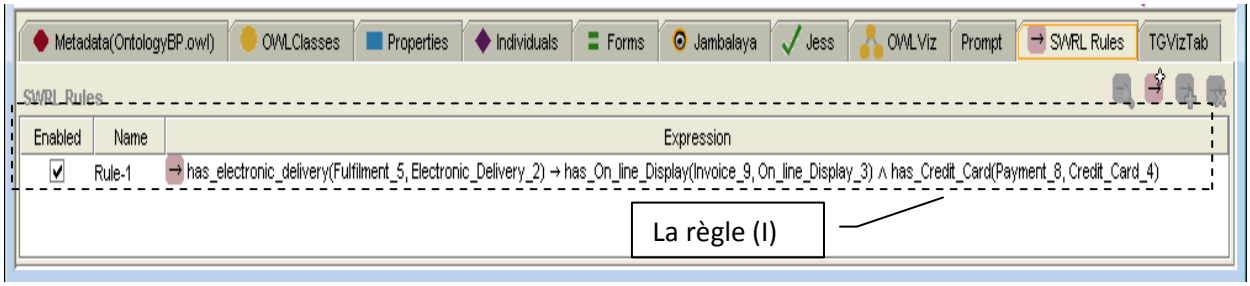

Figure 4.18. Implémentation de règle **(I)** avec l'onglet SWRL Rules d'outil Protégé

Nous remarquons que pendant cette étape, certains points de variation, peuvent être restés non résolus, la résolution ou la prise de décisions sur ces points de variations sera durant la phase d'exécution du processus.

Le processus d'affaires configuré (par la résolution de points de variations statiques) et l'ontologie du domaine d'application sont des entrées pour la prochaine étape, qui est l'attribution des rôles.

3.3.2.3. L'attribution des rôles

C'est-à-dire, attribuer chaque entité au rôle correspondant selon l'algorithme décrit dans le Tableau 4.6.

- Le processus d'affaires est toujours décrit en termes de rôles (acteurs qui exécutent les activités de processus).
- Les exigences pour que les acteurs puissent jouer ces rôles sont exprimées par le concept Requises (Capacités et autorités requises). L'ontologie du domaine d'application définit les entités actives et leurs acquisitions (Capacités et autorité acquises).
- Ainsi, rôles et entités, ces deux sont caractérisés en termes de capacités et autorités. L'attribution des rôles est réalisée en correspondant les requises aux acquises, selon l'algorithme suivant (Tableau 4.6), qui est proposé dans [ČIUK07a] et est enrichie par les concepts que nous avons ajoutés :

Répéter
1 *Rôle. Requises ← Capacités et autorités requises ;*
2 ***Répéter***
Entité. Acquises←Capacités et autorité acquises ;
Si *(Entité. Acquises = Rôle. Requises)* ***alors***
Attribution de rôle, Rôle ← Entité ; aller à 1 ; ***FinSi***
Si *(Entité. Acquises englobant Rôle. Requises)* ***alors***
Cette entité doit être reconstruite <u>Et</u> divisée en plusieurs entités spécifiques (la spécification des Acquises) ; aller à 2 ; ***FinSi***
Si *(Entité. Acquises sont englobées par les Rôle. Requises)* ***alors***
Ces entités doivent être composées d'une qui est plus grande, entité composite (la généralisation des Acquises) ; aller à 2 ; ***FinSi***

Jusqu'à ce que toutes les Entités soient vérifiées ;

Si *(aucun candidat d'entité active n'a joué un certain rôle)* ***alors***
Une nouvelle entité doit être créée ; ***FinSi***
Jusqu'à ce que tous les rôles soient attribués.

Tableau 4.6 Algorithme de l'attribution des rôles

Pour l'implémentation de cette étape, nous illustrerons durant l'étude de cas.

Le résultat de cette phase est un modèle du processus d'affaires localisé dans un domaine d'application choisi avec des points de variation non résolus.

3.3.2.4. La définition du flow de contrôle et la description du modèle exécutable de processus

Après avoir localisé le processus d'affaires dans un domaine d'application choisi. Dans cette étape, nous devons définir l'ordre d'exécution des activités de ce processus, par conséquent, nous ajouterons les connaissances de contrôle qui définissent l'ordre d'exécution de ces activités [ČIUK07a]. Donc, un modèle exécutable de processus d'affaires est produit.

Dans cette approche, les connaissances de contrôle sont considérées comme une partie des exigences d'implémentation et utilisées uniquement lors de la phase d'ingénierie applicative du processus [ČIUK06]. Par conséquent, cette approche élimine l'inconvénient des approches qui utilisent les connaissances de contrôle dans une phase très tôt (pendant la conception de processus du domaine) et ne peuvent pas être ajustées plus tard (les connaissances de contrôle sont durement codé dans des composants génériques de processus). Ceci est l'un d'inconvénients les plus importants de paquets ERP (Enterprise Resource Planning (ERP)) [ČIUK06].

De ce fait, les connaissances de contrôle (Figure 4.19 (Décrite en utilisant UML)) pour le processus générique de commande, par exemple, sont définies séparément. Elles sont ajoutées au processus de conférence localisé dans le domaine d'application choisi pour avoir un modèle exécutable de ce processus.

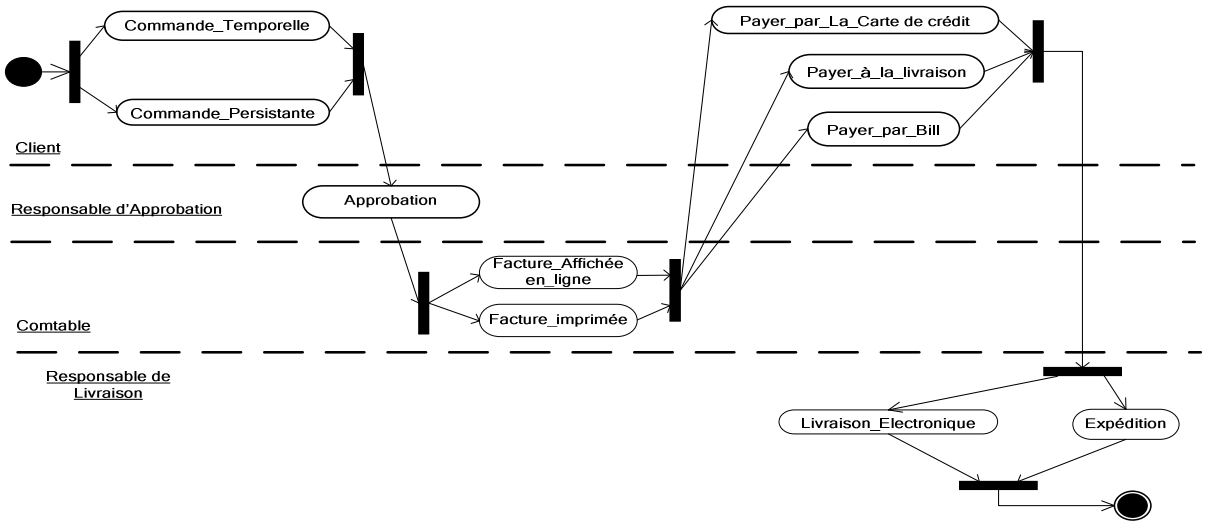

Figure 4.19. Connaissances de Contrôle pour le processus de conférence

Nous pouvons utiliser, un langage d'exécution de processus tel que WSBPEL, qui va exécuter par un système de gestion de Workfows. Ce système illustre l'exécution d'activités du processus.

Comme nous l'avons déjà mentionné, selon l'approche proposée dans [ČIUK07a] en cas d'absence d'un service ou d'une ressource pour l'exécution d'une activité pendant l'exécution d'un processus d'affaires, ils n'ont pas proposé un autre choix. En d'autres termes, ils n'ont pas proposé ou montré la résolution des points de variations qui peuvent avoir lieu durant l'exécution de processus.

La différence majeure entre notre proposition et celle dans [ČIUK07a], c'est que, en cas d'absence de ces services ou ressources, nous devons permettre la création des services nécessaires pour exécuter cette activité ou bien nous devons exécuter une autre activité avec un service ou une ressource d'exécution disponible.

Pour cette raison, nous pouvons distinguer un autre type de points de variations, qui se compose des points de variation dynamiques reliés à l'environnement d'exécution. Cela veut dire, que certains services ou ressources, dans cet environnement, doivent être disponibles pour qu'une activité s'exécute. En cas d'absence de ces services ou ressources, comme nous avons déjà mentionné, nous avons la possibilité ou le choix d'exécuter une autre activité avec un service ou une ressource d'exécution est disponible ou bien de créer ces services manquants.

3.3.3. La Phase d'exécution du processus d'affaires

La troisième phase (Figure 4.20) de notre approche a comme but de gérer l'exécution de processus d'affaires défini précédemment par un langage tel que WSBEL. Cela par la résolution des variabilités qui existent encore dans ce modèle exécutable du processus d'affaires.

- Comme nous l'avons déjà dit, certains points de variation, peuvent être restés non résolus, on les appelle points de variation dynamiques. Ces points de variations peuvent être des points de variation qui ont été défini dans la phase de domaine du processus, qui ne sont pas résolus durant la phase précédente, ou des points de variation qui peuvent être définis durant la phase d'ingénierie applicative du processus.
- Dans cette étape, il faut prendre les décisions correspondant au choix d'activités à exécuter, comme elles sont définies dans les phases précédentes et selon la disponibilité des services exigés pour l'exécution de chaque activité.

- Ces services peuvent être fournis par d'autres activités de processus lui-même ou disponible sur l'environnement d'exécution.
- Le modèle exécutable du processus d'affaires exécuté par un système de gestion des workflows.

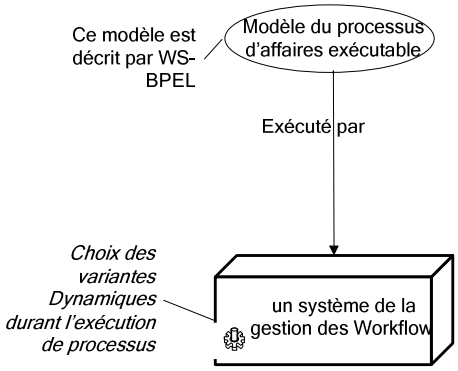

Figure 4.20. Le processus de développement de la phase d'exécution du processus d'affaires **(**Selon Donatas CIUKSYS [ČIUK07a])

Dans l'exemple du processus de commande, si nous n'avons pas un protocole ou un service pour faire une livraison électronique, nous pouvons faire une *Expédition* durant l'exécution de ce processus dans un domaine d'application et environnement d'exécution bien choisi. Dans ce cas là, nous allons choisir l'exécution de l'activité *Expédition* ou lieu d'exécuter l'activité *Livraison_Électronique.*

Dans la section suivante, nous exposerons les ontologies réutilisables de haut niveau qui ont été proposées dans cette approche. Elles permettent la réutilisation des connaissances pour localiser le processus d'affaires générique dans un domaine d'application et environnement d'exécution choisis.

3.4. Les ontologies de haut niveau utilisées

L'approche, déjà présentée dans ce mémoire, est principalement basée sur trois types d'ontologies :

3.4.1. L'ontologie de haut niveau

- Tant l'ontologie du domaine d'application que l'ontologie de processus, elle devrait être décrite par un certain système commun de méta concepts. Cela signifie qu'une certaine ontologie de niveau plus haut est exigée. La différence entre l'ontologie de haut niveau dans [ČIUK07a] et notre ontologie c'est que nous avons introduit le concept de Temps, que nous considérons important, surtout durant l'exécution du processus d'affaires.

Cette ontologie présente les concepts génériques qui sont partagés par toutes les ontologies de niveau inférieur et reflètent la réalité d'entreprise (le discours d'intérêt).

- L'ontologie de Haut-niveau proposé dans [ČIUK07a] a été influencée par l'ontologie d'entreprise d'Uschold [USCH98]. La différence importante entre Uschold et leurs ontologies c'est qu'ils ont permis les Etats aux Rôles.
- *L'ontologie de Haut-niveau* (Figure 4.21) est une ontologie à deux niveaux :

Le niveau supérieur fournit au seul concept nommé "Concept" qui est utilisé pour définir les concepts de deuxième niveau "Entité", "Relation", " Rôle" et "État_D_Affaire", "Temps".

- Ces concepts, à leur tour, sont utilisés pour définir les concepts du troisième niveau dans l'ontologie du domaine d'application aussi que dans l'ontologie du processus d'affaires. Cette idée est fournie par *Meta Object Facility (MOF)* [MOF06], selon cette approche les instances de méta- concept sont des concepts aussi.

Dans cette ontologie de Haut niveau et d'autres, décrits ci-dessous, sont représentés dans des diagrammes informels UML. Pour l'implémentation nous allons utiliser la notation formelle d'OWL.

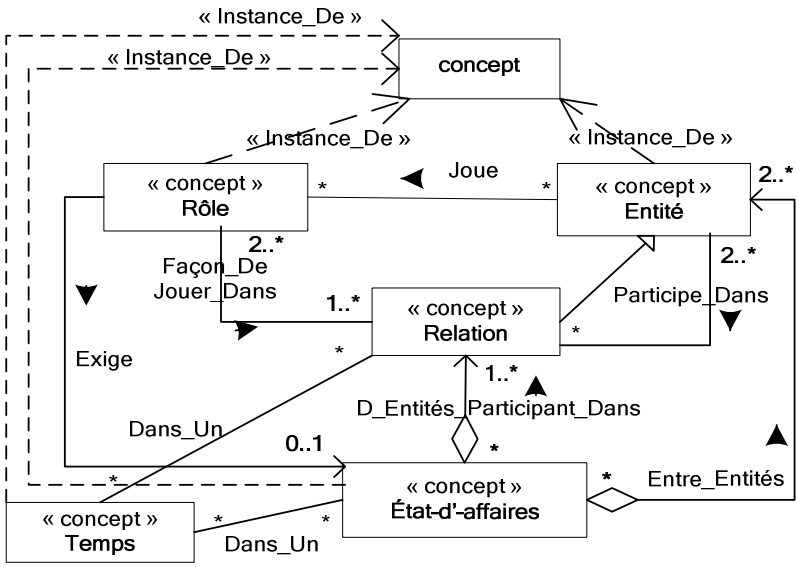

Figure 4.21. L'ontologie de haut niveau

3.4.2. L'ontologie du domaine d'application de haut niveau

- Nous nous basons sur l'ontologie proposée par les auteurs de [ČIUK07a] où nous avons trouvé que : Tous les concepts définis par cette ontologie sont des instances de concepts définis par l'ontologie de haut niveau.

- Cette ontologie (Figure 4.22) est considérée comme une base pour définir les concepts de l'ontologie d'un domaine d'application particulier. L'ontologie affine la notion d'entité et classifie toutes les entités en deux types : entités actives et entités passives. Ces concepts sont organisés de cette façon pour faciliter l'attribution des rôles, pendant la phase de l'ingénierie applicative [ČIUK07a].
- *L'entité active* a comme sous types : *Postes de travail, Systèmes d'application* et *Unités organisationnelles.* Tous ces types d'entités, peuvent fournir des capacités et peuvent aussi être candidats à jouer des rôles définis par l'ontologie de processus. Ils peuvent changer les états des *entités passives* dans un *Intervalle de temps* spécifié (qu'on a ajouté comme nouveau concept).
- *L'objectif d'affaires* est un état d'entité passive, qui consiste à jouer le rôle de production dans l'ontologie de processus d'affaires. Le concept *"Objectif"* modélisé l'objectif de l'entreprise. Ils forment une hiérarchie. C'est-à-dire, nous supposons que dans un domaine d'application (un domaine spécifique de l'entreprise) prévoit explicitement des objectifs d'affaires qui doivent être atteints pour la bonne marche de l'entreprise dans cette zone d'affaires.
- *Par exemple, un poste de travail* peut fournir une capacité d'écriture et par conséquent la possibilité de préparer un document (changement de son état). De même, un système de commande peut fournir une capacité du traitement des commandes et être capable de changer l'état d'une commande (de l'état non traitée vers état traitée).
- La différence entre notre ontologie et celle proposée par les auteurs de [ČIUK07a] c'est que nous avons ajouté les concepts suivants *: L'intervalle de Temps,* qui est une instance de concept Temps, représente l'intervalle du temps estimé pour changer l'état d'une entité passive par une entité active. Il peut aussi représenter l'Intervalle de temps pour atteindre l'Objectif ou les Sous Objectif d'une Activité d'affaires. *Intervalle de Temps* est dans un sens confus ; vous ne savez pas de quelle longueur il est ou bien nécessaire. *Exemple : « Toujours »* son intervalle de l'infini loin dans le passé jusqu'à l'infini dans un avenir lointain.
- Et *Autorité*, dans certain cas les entités actives ont comme Acquises des Autorités au lieu de Capacités ou bien les deux en même temps, qui peuvent changer l'état d'une Entité passive.

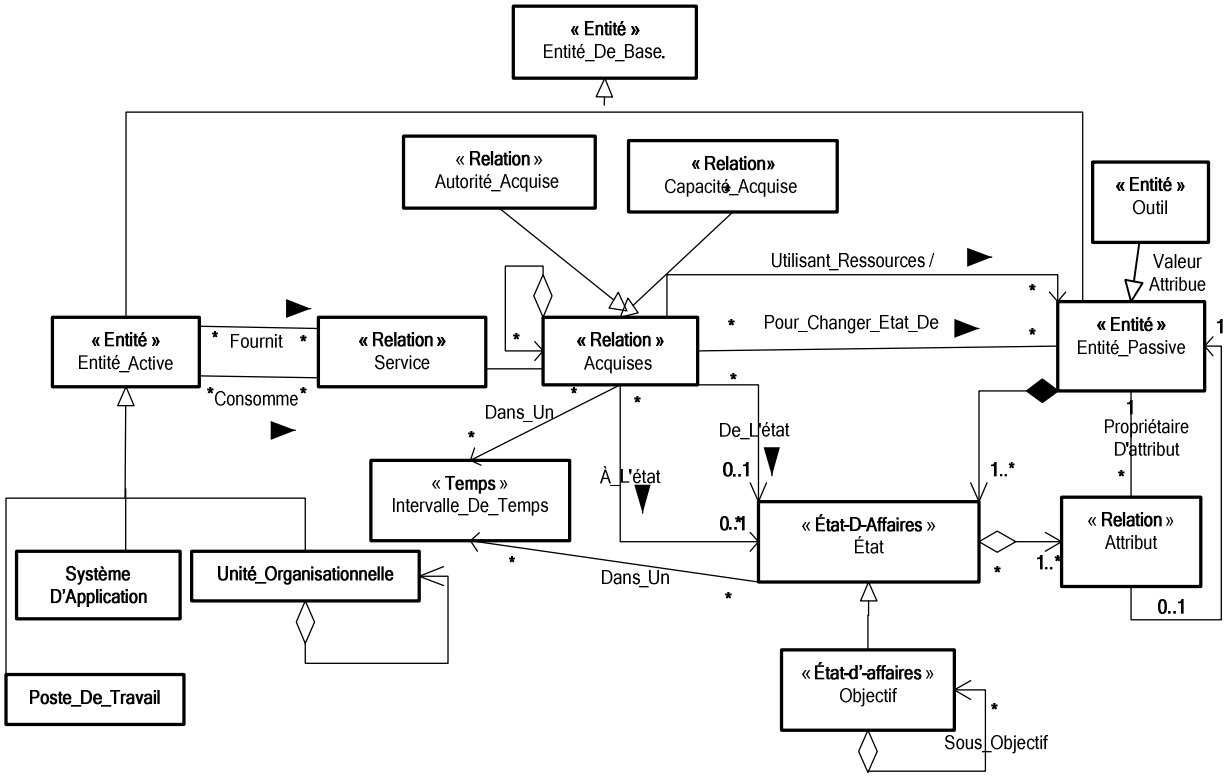

Figure 4.22. L'ontologie du domaine d'application

3.4.3. L'ontologie du processus d'affaires de haut niveau

En 2003, le consortium OMG a annoncé une initiative, qui vise à normaliser la conceptualisation du processus d'affaires et à développer ce qu'on appelle Business Process Definition Métamodèle (BPDM) [BPDM08], la version finale est livrée en 2008. BPDM définit plus de 100 concepts.

- L'ontologie du processus de haut niveau qui a été proposée par Caplinskas [ČIUK07a] est un sous-ensemble de BPDM [BPDM08]. Elle comprend les concepts nécessaires pour décrire tous les types de rôles fournis par le processus d'affaires. Acteurs, entrées, sorties et les ressources sont modélisées comme des rôles. Les entités du domaine doivent être assignées à ces rôles quand le processus d'affaires est localisé dans un domaine particulier d'application : les entités actives peuvent être candidates aux rôles d'acteur, entités passives pour entrer, les entités passives aux rôles d'entrées, de sorties et de ressources. Notre contribution dans cette partie d'ontologie, est d'ajouter aux requises l'autorité pour exécuter une activité ou une sous-activité, dans un Intervalle de temps estimé (Figure 4.23).
- Ces concepts ne sont pas suffisants pour représenter les variabilités fournies par les modèles des caractéristiques. Ainsi, les concepts tels que : *Variabilité*, *Variante*,

*Point de variation,…*doivent être inclus dans l'ontologie du processus de haut niveau [ČIUK07a].

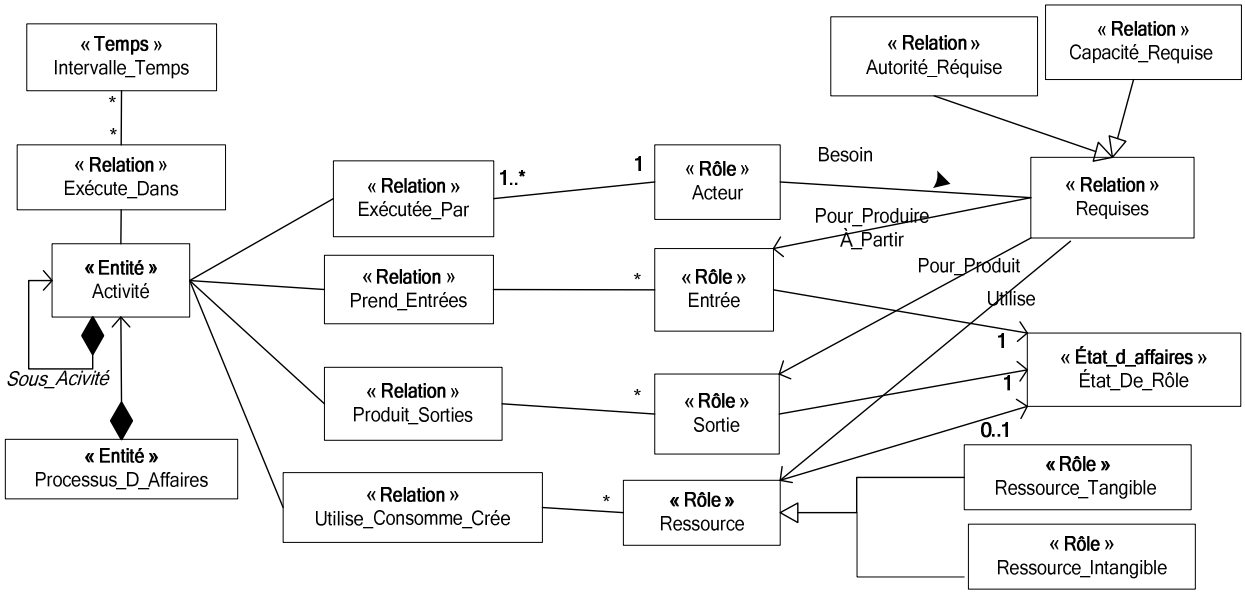

Figure 4.23. L'ontologie du processus d'affaires de haut niveau (Conceptualisation de processus)

3.4.4. Conceptualisation de variabilité

Notre apport dans la conception de variabilité est d'adapter le modèle général de variabilité dans les lignes de produits, qui est proposé par Becker [BECK03], aux processus d'affaires. La conceptualisation de la variabilité au sein d'un processus d'affaires générique est montrée dans la Figure 4.24. Nous présentons une description de notre apport dans cette conception :

1. **La Variabilité :** représente une capacité de changer ou adapter un système [GURP01] ; dans notre contexte, le processus avec variabilités peut être adapté dans divers domaines d'application.
   - *Le Point de variation* : est une place dans une asset logiciel où la variabilité se produit [BECK03]. *La Variabilité* correspond à un ensemble de points de variation, qui définissent l'étendue de variabilité (c'est-à-dire ; plus le nombre de variantes, plus le degré d'adaptabilité du système). Pour la raison de simplicité, nous permettons à la variation du processus de se produire seulement dans les activités, et d'appeler ces activités des activités génériques. D'après le modèle général de variabilité dans une famille de produits, qui a été proposé par Becker [BECK03], il avait distingué deux types des *points de variation :*

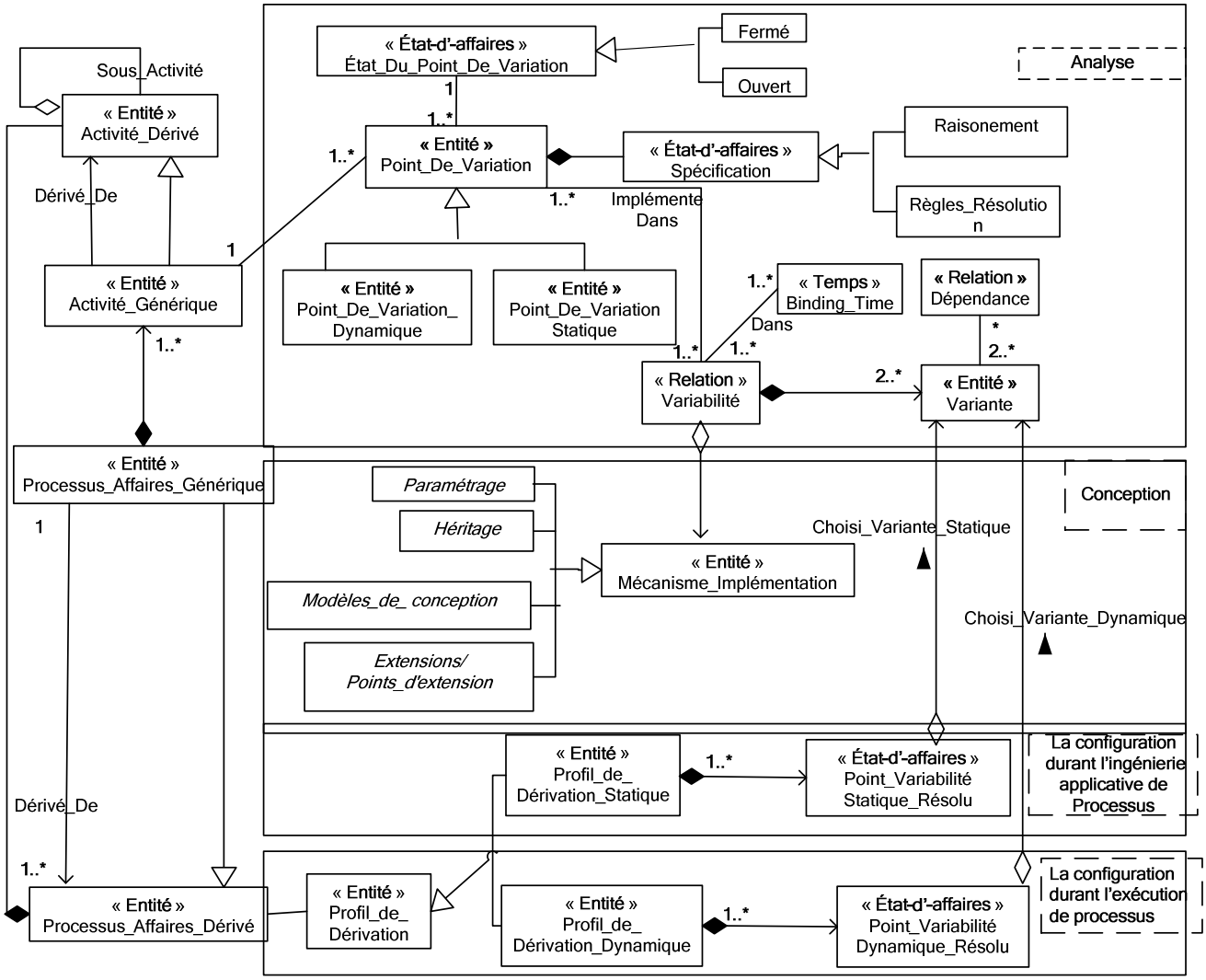

Figure 4.24. Conceptualisation de variabilité

a. *Points de Variation Statiques :* délimitent une solution dans une Asset, qui permet de gérer la variabilité pendant le cycle de vie de produit.

b. *Points de Variation dynamiques :* délimitent une solution dans une Asset, qui permet de gérer la variabilité à la fin de cycle de la vie du produit, c'est à dire après la livraison.

Dans notre cas, nous supposons que les points de variation statiques seront résolus pendant la phase de l'ingénierie applicative. Par contre les points de variation Dynamiques seront résolus pendant la phase d'exécution, afin d'adapter aux changements de contexte, pendant l'exécution d'activités d'un processus d'affaires. Nous nous sommes basés sur [JIAN10], pour distinguer deux états pour un point de variation, fermé ou ouvert :

c. *Le point de variation est « ouvert. »* dans ce cas, il est possible pendant l'ingénierie applicative d'ajouter une nouvelle variante ou de modifier des variantes existantes.

d. *Le point de variation est « fermé. »* dans ce cas, il n'est pas possible de sortir des choix proposés par le point de variation.

2. **La résolution de la variabilité :** est le choix de la variante convenable. Ce choix doit être effectué pour chaque variabilité.

- *Les dépendances* restreignent les choix des variantes, c'est- à -dire le choix d'une variante peut exiger le choix ou l'élimination de l'autre (par exemple le choix du type de paiement "Carte de crédit» au sein du processus d'affaires E-Shop peut rendre l'exigence de l'activité optionnelle "Se connecter à la banque").
- Les Choix des variantes sont enregistrés dans *le Profil de Dérivation.*
- *La Variante* choisie doit être intégrée dans le système au Point de Variation. Pour réaliser cette intégration, *des Mécanismes d'Implémentations* précisent des techniques qui dépendent du type de variantes et le lieu où l'intégration doit être effectuée.
- Les moments exacts, où les décisions retardées prises, sont variables. Ces moments, généralement appelés "Binding Time" [JIAN10].
- *Un profil de dérivation* comporte un ensemble d'affectations. Chaque affectation représente une décision prise (Choix de variante, que ce soit statiques ou dynamiques). Par exemple, La *variante* (**A)** a été choisie au *point de variation* (**B)** pour la *variabilité* (**C)** au *BindingTime* (**D).** Si aucune affectation n'est disponible pour une variabilité, donc la Variabilité n'est pas liée au *profil de dérivation*.

Dans le Tableau 4.7 ci-dessous, un algorithme qui illustre la résolution de variabilité au sein d'un processus d'affaires générique :

Modèle de processus d'affaires génériques {contient des points de variation qui doivent être résolus afin de configurer ce processus *d'abord* dans un domaine d'application et *ensuite* dans un environnement d'exécution bien choisi.}
{La résolution des points de variation statiques :}

Répéter
Si *(Points De Variation = Points De Variation Statiques)* ***alors***
- *Résolution des points de variation (Choix de variantes statiques, des dépendances restreignent ce choix) {c'est à dire la configuration durant la phase de l'ingénierie applicative de processus} ;*
- *Ajouter ces variantes statiques au profil de dérivation statiques ;*

FinSi

Jusqu'à ce que tous les Points De Variation soient vérifiés ;

Modèle de processus d'affaires localisé dans un domaine d'application bien choisi ;

{La résolution des points de variation Dynamiques **:}**

Répéter

Si *(Points De Variation = Points De Variation Dynamiques)* ***alors*** *:*

- *Résolution des points de variation (Choix de variantes Dynamiques, des dépendances restreignent ce choix) {c'est à dire la configuration durant la phase de l'ingénierie d'exécution de processus} ;*
- *Ajouter ces variantes Dynamiques au profil de dérivation Dynamiques ;*

FinSi

Jusqu'à ce que tous les Points De Variation soient vérifiés ;

Modèle de processus d'affaires exécutable {Localisé dans un environnement d'exécution bien choisi.}, qui va exécuter par un système de gestion des Workflows.

Tableau 4.7 Un algorithme qui illustre la résolution de variabilité au sein d'un processus d'affaires générique

D'après ***Capliniska***, la conception de la variabilité est conçu durant les trois phases ***l'analyse du domaine dans l'ingénierie de domaine, la conception de domaine et la configuration,*** comme suit :

- ***L'une des tâches de l'analyse du domaine dans l'ingénierie de domaine*** est *l'identification des points de variation, les variabilités, les variantes et les dépendances.* Toutes ces informations seront représentées par le modèle des caractéristiques de la méthode Feature-Oriented Domain Analysis (FODA) [KANG90]. Pour raison de simplicité, ils ont permis que la *Variation*, dans le

processus, se produire uniquement dans les activités et on les a appelés *Activités génériques*. La raison de cette simplification est que la majorité des variabilités de processus se trouvent notamment dans les *activités*. Par conséquent, les activités génériques sont effectivement le seul type des points de *variation* que nous considérons (et le concept *ActivitéGénérique* spécialise le concept *Variante*. Les deux, *Activité et Activité générique* spécialisent le concept *ActivitéAbstaite*. Cela leur permet d'être interchangeables, ce qui est :

a. *Durant l'analyse du domaine de processus,* l'Ingénieur de processus va remplacer certaines activités par d'autres qui sont génériques,
b. *Pendant la configuration de processus, les Activités Génériques seront remplacés par* d'autres activités non-génériques (c'est-à-dire variante choisie).

Par conséquent, ils ont proposé un seul type de variantes, qui est Activités Abstraites (le concept *Activités Abstraites* s'étend le concept *Variante*). *L'Activité Générique* représente une famille d'activités. Comme membres de la famille, ils ont considéré non seulement les activités simples, mais aussi les autres activités qui sont génériques. De cette façon, nous pouvons avoir la hiérarchie des *Activités Génériques* (il y a une association d'agrégation entre *l'Activité Générique* et *l'Activité Abstraite*).

- ***Dans l'étape de la conception de domaine,*** l'ingénieur de processus doit préciser comment les activités seront dérivées des activités génériques (comment la variabilité sera résolue ?), c'est-à-dire, choisir les *Mécanismes de l'Implémentation* de variabilité.

Puhlmann discute les mécanismes de l'implémentation de variabilités appropriés aux processus d'affaires comme suit [ČIUK07a] :

a. *Le Paramétrage* permet l'introduction de la variabilité du comportement d'*activité générique,* en mettant les paramètres à certains endroits. Ensuite, nous pouvons résoudre la variabilité en assignant des valeurs à ces paramètres.
b. *L'héritage* permet le changement de *l'Activité Générique* par une autre activité spécialisée. *Activité spécialisée* doit être conforme à l'interface de *l'Activité Générique* (au moins les entrées et sorties doivent être identiques) et fournit généralement un comportement supplémentaire.

c. *Les modèles de conception* sont basés sur la dissimulation et l'héritage d'informations, comme le modèle de la conception de stratégie, peut être représenté dans les processus en utilisant l'héritage. Par exemple, le modèle de la conception de stratégie comprend ce que l'on appelle *l'activité abstraite* qui définit l'interface (Entrées et Sorties) et plusieurs activités d'implémentation. Une de ces activités doit être choisie et remplacera *l'activité abstraite.*

d. *Extensions/ Points d'extension.* Les points d'extension sont les lieux où le processus peut être étendu avec des comportements supplémentaires. Ils peuvent être représentés par ce qu'on appelle *l'Activité Nulle- Activités Sans Comportement. Extensions* (activités non vide) peut être choisi pour remplacer l'activité nulle. *Une activité s'étendant* doit avoir une interface compatible pour être intégrable dans le processus au point d'extension correspondante.

- *Le configurateur* guidé par l'analyste des processus d'affaires devrait être en mesure de résoudre toutes les variabilités (c'est à dire choisir exactement une variante pour chaque variabilité) et d'intégrer les variantes choisies dans le processus en utilisant les mécanismes de l'implémentation appropriés à la variabilité. Par conséquent, une description du processus configuré (c'est à dire sans variabilité) est entièrement produite et enregistrée dans le *Profil de Dérivation*.

De notre part, et puisque nous avons deux types de configuration, celui de domaine d'application et l'autre durant la phase d'exécution, donc notre contribution c'est que :

- *Le configurateur* devrait être en mesure de distinguer les points de variation statiques et dynamiques et de résoudre les variabilités dont les points de variation sont statiques dans la phase de l'ingénierie d'application, et celles dont les points de variation sont dynamiques dans la phase d'exécution (c'est à dire choisir exactement une variante pour chaque variabilité).
- *Le configurateur* devrait être en mesure aussi d'intégrer les variantes choisies dans le processus en utilisant les mécanismes de l'implémentation appropriés à la variabilité. Par conséquence, une description du processus configuré (c'est à dire sans variabilité statique dans la première configuration de l'approche

proposée et sans variabilité dynamique dans la deuxième configuration) est entièrement produite et enregistrée dans le *Profil de Dérivation*.

- ***Nous pouvons aussi distinguer deux type de profile de dérivation (***statique et dynamique***) pour chaque type de configuration.***

La Figure 4.25, représente la conceptualisation du processus de variabilité proposé dans [ČIUK07a] et que nous avons enrichi par les concepts suivants: Point de variation dynamique, Point de variation dynamique résolu, Point de variation statique, Point de variation statique résolu.

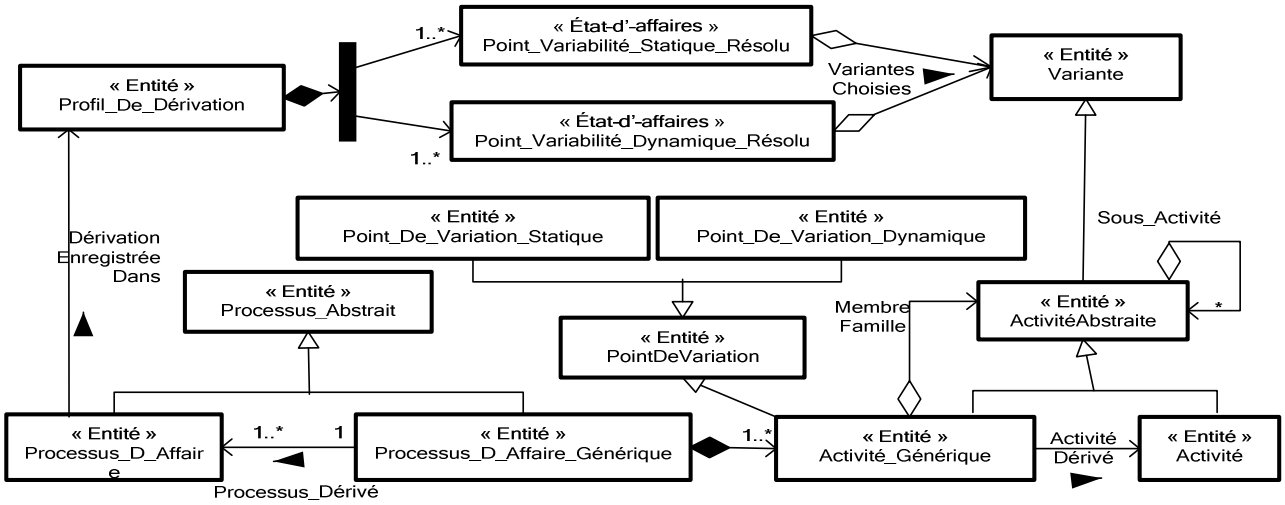

Figure 4.25. La conception du processus de variabilité.

4. Conclusion

Selon l'approche proposée dans [ČIUK07a] en cas d'absence d'un service ou une ressource pour l'exécution d'une activité pendant l'exécution d'un processus d'affaires, ils n'ont pas proposé un autre choix.

La différence majeur entre notre proposition et celle dans [ČIUK07a], c'est que, en cas d'absence de ces services ou ressources, on a la possibilité d'exécuter une autre activité avec un service ou une ressource d'exécution disponible.

Pour cette raison, nous avons distingué un autre type des points de variations, qui est les points de variation dynamiques, relie à l'environnement d'exécution. Cela veut dire, que certaines services ou ressources, dans cet environnement, doivent être disponibles pour qu'une activité s'exécutée. En cas d'absence de ces services ou ressources, comme nous l'avons déjà cité, on a la possibilité ou le choix d'exécuter une autre activité avec une service ou une ressource d'exécution est disponible. Dans ce cas cas-là nous avons deux types de points de variation : ceux qui sont résolus durant la configuration dans la phase applicative du processus, dites statiques, et ces derniers qui sont les points de variation dynamiques.

Donc notre contribution majeure c'était la proposition d'une conceptualisation de variabilité qui support ces deux types de points de variation ainsi que l'algorithme de la résolution de cette variabilité.

Ce qui nous a permis de proposer une approche à trois phases, donc nous avons scindé en deux parties la phase de l'ingénierie de processus, proposée par Caplinskas : la phase d'ingénierie applicative et la phase d'exécution. Donc, l'architecture de l'approche que nous avons proposée comporte trois phases : ***la phase d'ingénierie du domaine de processus (A), la phase d'ingénierie applicative du processus (B) et la phase d'exécution du processus (C).***

Pour soutenir et consolider l'architecture déjà citée, nous avons proposé d'enrichir deux types d'ontologies réutilisables de haut niveau : Celle qui représente les connaissances d'un processus d'affaires générique pendant la phase d'ingénierie du domaine (nous avons mis beaucoup plus l'accent sur la conceptualisation de variabilité) et l'autre ontologie qui a été désignée à la représentation des connaissances d'un domaine d'application pendant la phase d'ingénierie applicative du processus. Les deux ontologies ont été basées sur un système de méta-concepts, qui constitue une ontologie de haut niveau, que nous avons traité également.

Les principaux concepts que nous avons ajoutés sont : le concept de **Temps** (il est important surtout durant l'exécution du processus d'affaires) qui est considéré, dans l'ontologie de haut niveau, comme instance de concept **Concept,** et qui avait une relation entre le concept **Relation. P**uisque les deux autres ontologies sont considérées comme instanciations de l'ontologie de haut niveau, le concept de temps est apparu dans les deux ontologies sous le nom d'**Intervalle de Temps.** Il avait une relation avec l'entité ***Activité*** dans l'ontologie de processus et deux relations dans l'ontologie de domaine d'application, l'une avec la relation ***Acquises*** et l'autre avec l'état d'affaires **État**.

Nous avons distingué deux ***types des points de variation*** : ***statiques et dynamiques.*** Les points de variation statiques sont résolus durant la configuration de processus d'affaires générique lors de phase (B), par contre les points de variation dynamiques sont résolus lors de la phase (C). Nous nous sommes basés sur [JIAN10], nous avons aussi distingué deux états pour un point de variation, fermé ou ouvert : *Le point de variation est «ouvert»* dans ce cas, il est possible pendant l'ingénierie applicative d'ajouter une nouvelle variante ou de modifier des variantes existantes. *Le point de variation est «fermé»* dans ce cas, il n'est pas possible de sortir des choix proposés par le point de variation.

Puisque nous avons deux types de configuration, celui de domaine applicative du processus et l'autre durant la phase d'exécution du ce processus, donc notre contribution dans ces deux types de configuration était comme suit :

- **Le configurateur** devrait être en mesure de distinguer les points de variation statiques et dynamiques et de résoudre les variabilités dont les points de variation sont statiques dans la phase de l'ingénierie applicative du processus, et celles dont les points de variation sont dynamiques dans la phase d'exécution du ce processus (c'est-à-dire choisir une variante pour chaque variabilité selon les contraintes mis sur elle).
- **Le configurateur** devrait être en mesure aussi d'intégrer les variantes choisies dans le processus en utilisant les mécanismes de l'implémentation appropriés à la variabilité. De ce fait, une description du processus configuré (c'est-à-dire sans variabilité statique dans la première configuration de l'approche proposée ou sans variabilité dynamique dans la deuxième configuration) est entièrement produite et enregistrée dans le Profil de Dérivation.

Dans le chapitre suivant, nous présenterons une étude de cas pour illustrer l'approche proposée. Nous allons étudier un processus de conférence simplifié qui sera considéré comme un processus générique.

Chapitre -05-

Étude de cas et Implémentation

1. Introduction

"The value of achievement lies in the achieving."[Albert Einstein]

Pour bien assimiler l'approche proposée, nous allons présenter dans ce chapitre l'exemple d'un processus de conférence simplifié qui est considéré comme un processus générique et qu'il peut être réutilisé dans différentes universités et collèges ainsi que pour faire une présentation lors des conférences. Nous commençons tout d'abord par une description de cette étude de cas, puis nous présentons la partie d'implémentation.

2. Description d'application de l'approche proposée sur le processus générique de Conférence

Nous allons discuter l'application de l'approche proposée, sur le processus de Conférence à travers les trois phases :

2.1. La phase d'ingénierie du domaine de processus de conférence

Constituée de trois activités suivantes :

2.1.1. Analyse de domaine du processus de conférence

On prend par exemple les concepts d'activité du processus de conférence (Figure 5.1, décrite en utilisant UML) suivant :

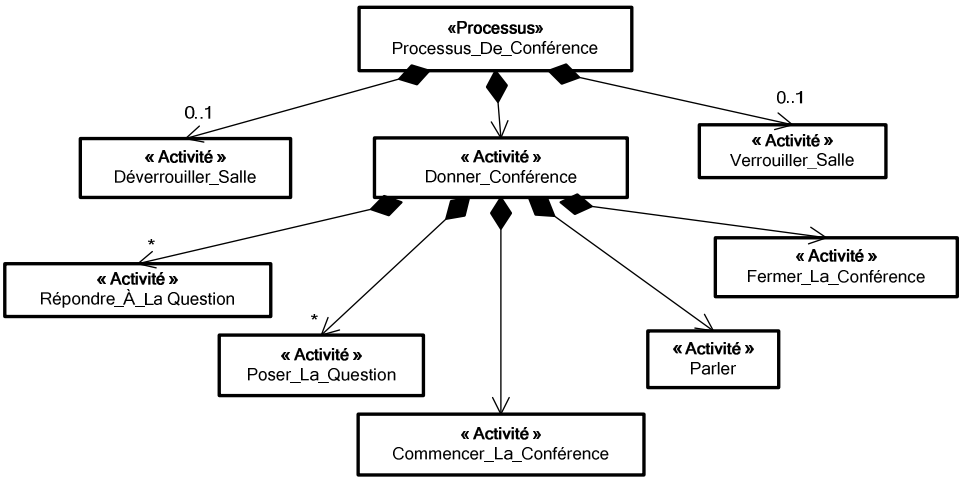

Figure 5.1. Concepts d'activité du processus de conférence**(**Selon Donatas CIUKSYS **[ČIUK06]**)

On peut représenter aussi ces activités par un modèle de caractéristiques (Figure. 5.2), pour bien décrire les éléments (Activités) communs et variables au sein d'une famille du processus de conférence :

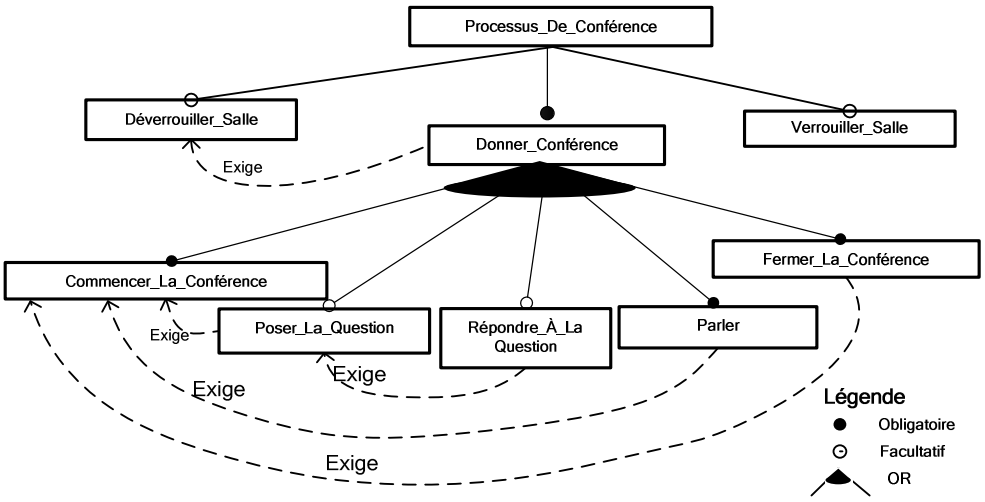

Figure 5.2. Modèle de caractéristiques du processus de conférence

2.1.2. Conception de domaine du processus de conférence

La construction de l'ontologie pour chaque activité du processus de conférence (Figure. 5.1). Les figures (Figure. 5.3, Figure. 5.4, Figure. 5.5, Figure. 5.6, Figure. 5.7, Figure. 5.8, Figure. 5.9) représentent les activités de processus Conférence, utilisant le méta-modèle présenté sous forme de l'Ontologie du processus d'affaires de haut niveau (décrit en termes de rôles) proposée dans cette approche (cf. Section 4.3.3 de chapitre 04).

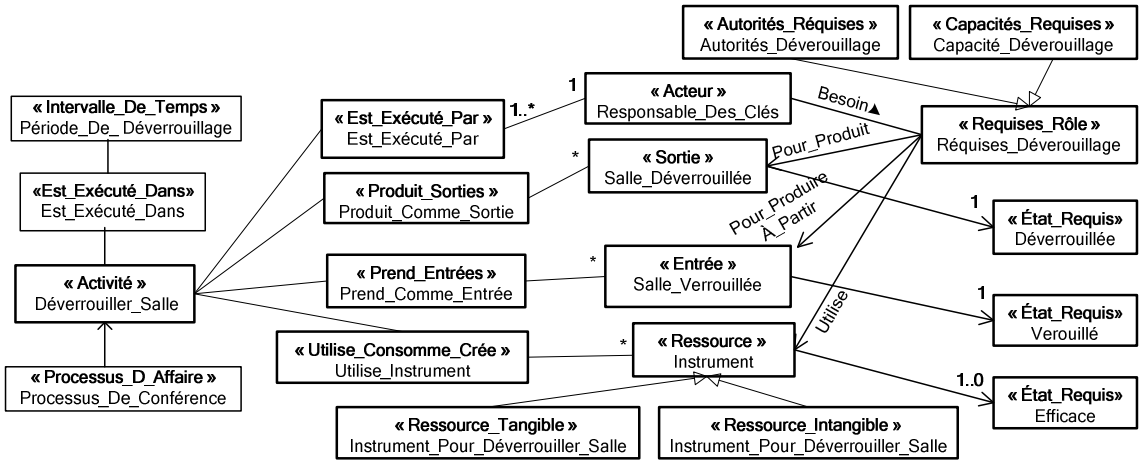

Figure 5.3. Un fragment d'ontologie de l'activité «Déverrouiller_Salle »

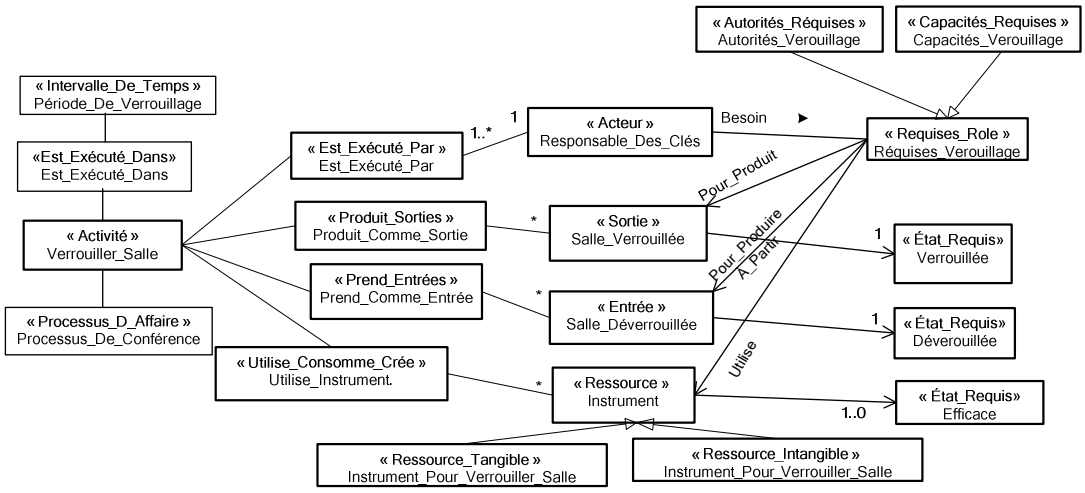

Figure 5.4. Un fragment d'ontologie de l'activité «Verrouiller_Salle »

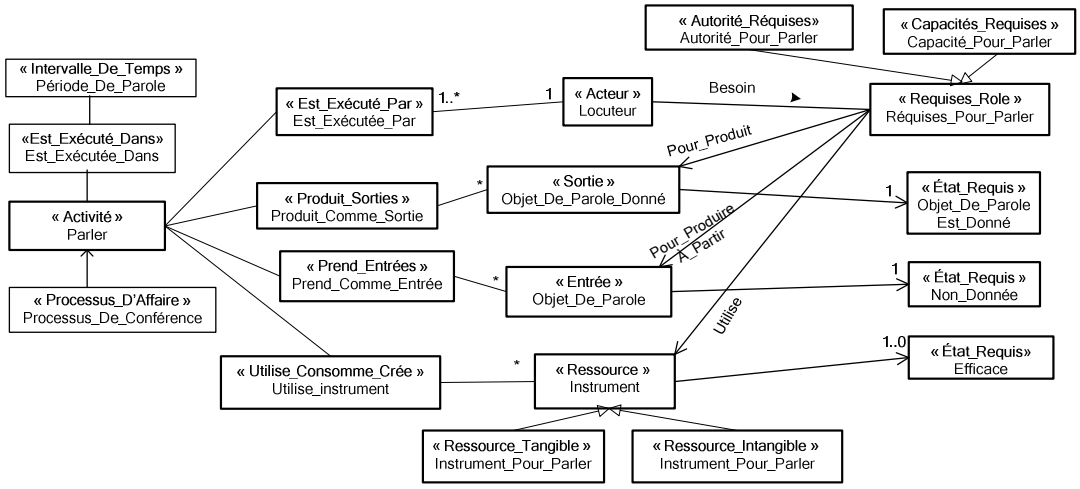

Figure 5.5. Un fragment d'ontologie de l'activité «Parler »

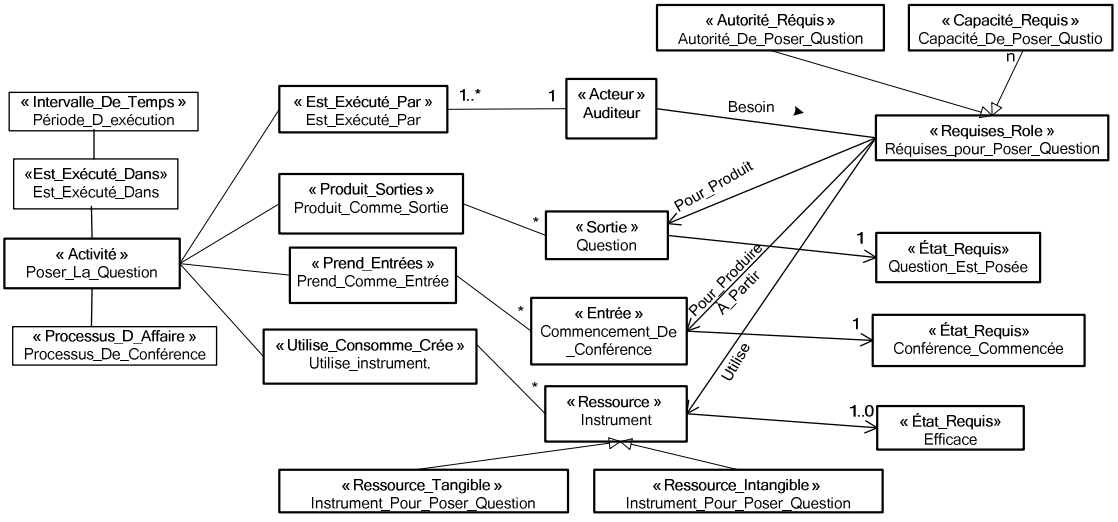

Figure 5.6. Un fragment d'ontologie de l'activité «Poser_la_question »

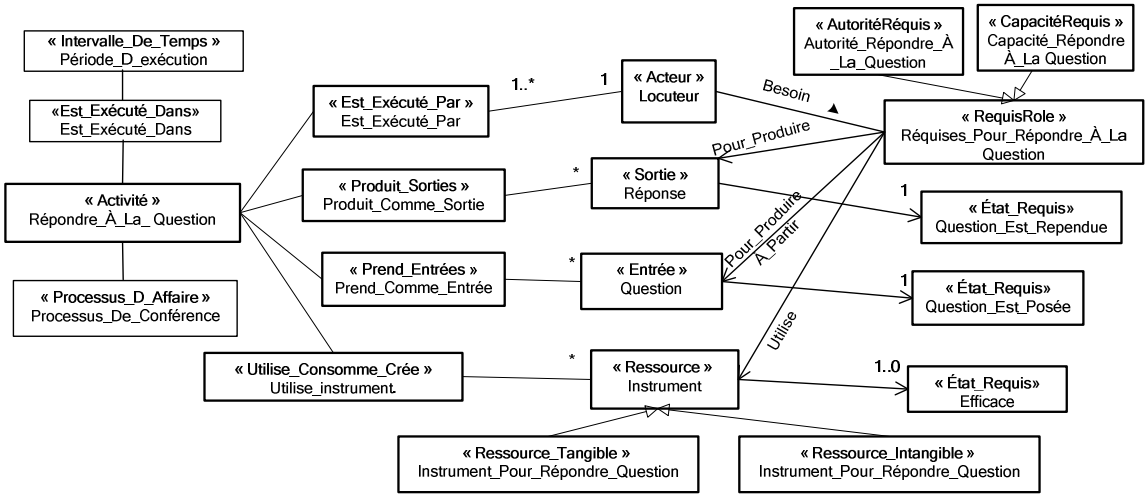

Figure 5.7. Un fragment d'ontologie de l'activité «Répondre_à_la_question »

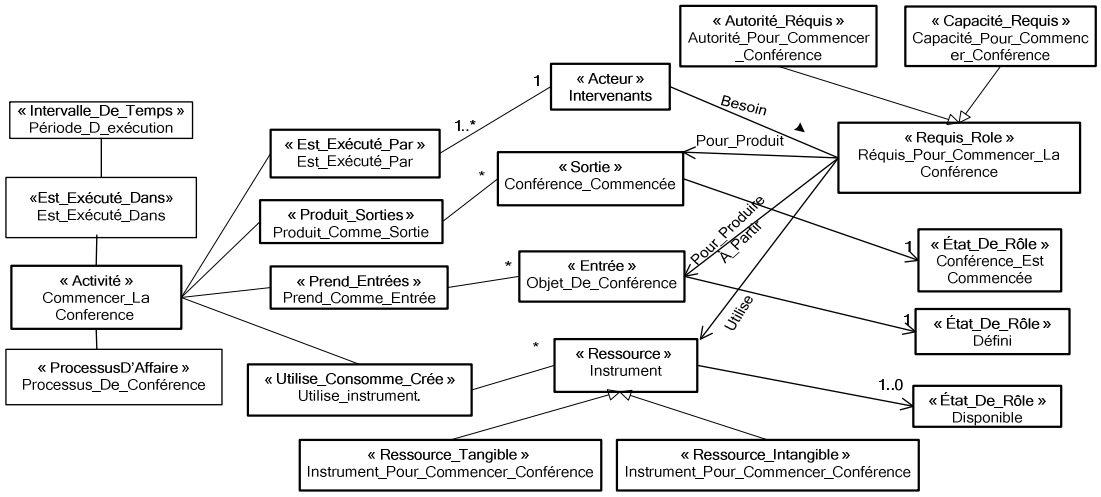

Figure 5.8. Un fragment d'ontologie de l'activité «Commencer_la_conférence »

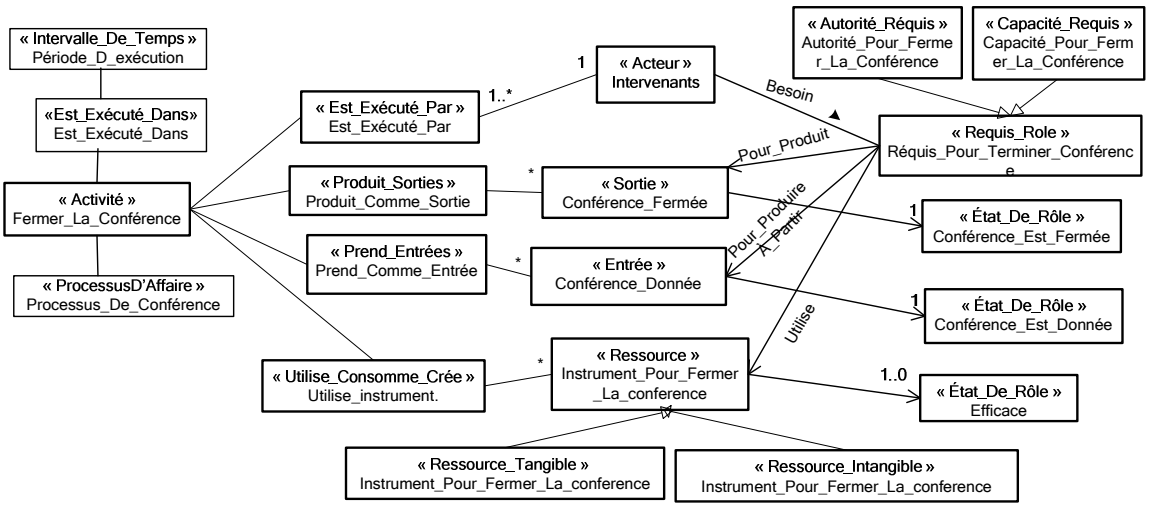

Figure 5.9. Un fragment d'ontologie de l'activité «Fermer_la_conférence»

2.1.3. Implémentation de domaine du processus de conférence

C'est d'empaqueter le modèle des caractéristiques et l'ontologie de processus dans un paquet d'assets réutilisables. L'ontologie de processus est représentée comme assets réutilisable, en utilisant Web-Ontology Langage (OWL) à base de la Logique de Description (DL).

Nous avons utilisé l'outil Protégé pour cette Implémentation. Après avoir construit l'ontologie de haut niveau, nous l'avons importée pour construire l'ontologie du domaine de processus d'affaires de haut niveau. Cette ontologie est considérée comme un méta modèle pour construire n'importe quelle ontologie de processus d'affaire. Dans ce cas là, en utilisant ce méta modèle, nous avons la possibilité de construire l'ontologie correspondant aux activités du processus de conférence et ensuite construire les différentes règles de dépendance entre ces activités. Comme nous allons voir dans la section d'implémentation.

2.2. La Phase d'ingénierie Applicative du processus de conférence

La réutilisation des éléments, déjà définis dans la phase précédente.

2.2.1. L'analyse du domaine d'application du processus de conférence

Nous tirerons profit du modèle de domaine de conférence existant et nous décrirons les besoins du client en utilisant les caractéristiques à partir du modèle de domaine de conférence. Cependant les nouvelles exigences des clients, qui ne figurent pas dans le modèle de conférence, nécessitent un développement personnalisé. Le résultat de cette étape est une Ontologie du domaine d'application qui est le processus de Cours ou d'une Vidéo conférence dans une université particulière. Nous utilisons le méta-modèle présenté sous forme de l'Ontologie du domaine d'application de haut niveau (décrite en termes d'entités) proposée dans cette approche, pour la construction de l'ontologie du domaine d'application choisi. Nous proposons cette ontologie représentée dans le diagramme **(**Figure. 5.10**)** suivant :

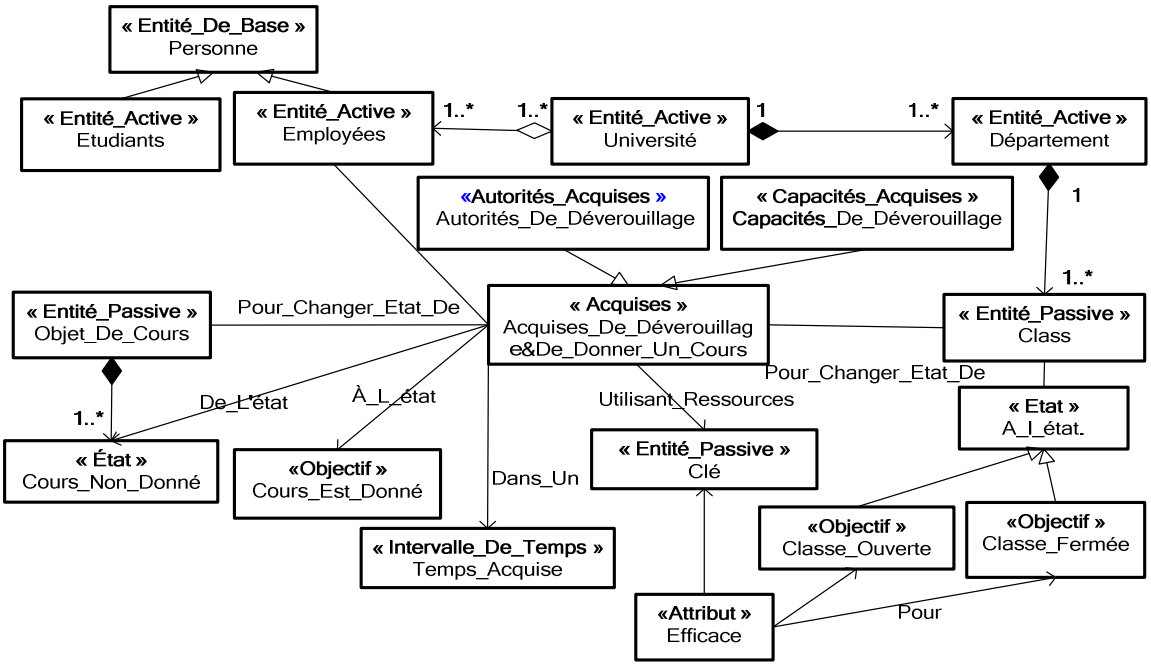

Figure 5.10. Un fragment d'ontologie du domaine d'application

2.2.2. La configuration du processus de Conférence

Pour avoir un Processus de Cours. La réutilisation des connaissances du processus exige que le processus soit configuré en tenant compte des particularités du domaine d'application choisi.

Comme nous avons présenté dans le chapitre précédent, pour résoudre le problème de la configuration de processus, nous suivrons l'approche à base de la Logique de Description (DL) [BAAD03] décrite dans [ČIUK07a], parce que nous définissons toutes les ontologies proposées utilisant l'Ontology Web Langage (OWL) à base de la Logique de Description (DL). Une solution au problème de configuration peut être proposée pour être un modèle de la base de connaissance donnée de type Logiques de Description (DL) [BAAD03].

Dans ce cas, l'espace de configuration est défini par le TBox qui décrit la hiérarchie des concepts et la hiérarchie des rôles. ABox initial inclut exclusivement des variantes obligatoires. Utilisant cette approche, le choix des variantes optionnelles ou alternatives nécessite que ABox soit augmenté avec les individus correspondants et qu'un raisonneur OWL DL (nous utiliserons dans notre cas d'étude le raisonneur Racer) soit sollicité pour vérifier si ABox est conforme à l'espace de configuration. Dans le cas contraire le choix doit être défait. Pour assurer les dépendances entre les variantes, nous avons utilisé les règles de Horn [MOTI05, ČIUK07a].

De ce fait, l'espace de configuration (TBox) pour le problème de la configuration dans le processus de conférence, qui est présenté par le modèle des caractéristiques Figure 5.2, est présenté dans le Tableau 5.1 :

Partie_De_Donner_Conférence ⊆ (Commencer_La_Conférence U Parler U Fermer_La_Conférence U Poser_La_Question U Répondre_à_La Question)

⊆ =1 Partie_De ∩ Partie_De.Donner_Conférence

Commencer_La_Conférence ⊆ ┐ (Parler ∩ Fermer_La_Conférence ∩ Poser_La_Question ∩ Répondre_à_La_Question)

Parler ⊆ ┐ (Commencer_La_Conférence ∩ Fermer_La_Conférence ∩ Poser_La_Question ∩ Répondre_à_La_Question) **(01)**

Fermer_La_Conférence ⊆ ┐ (Commencer_La_Conférence ∩ Parler ∩ Poser_La_Question ∩ Répondre_à_La_Question)

Poser_La_Question⊆ ┐ (Commencer_La_Conférence ∩ Parler ∩ **Fermer_La_Conférence** ∩ Répondre_à_La Question)

Répondre_à_La_Question ⊆ ┐ (Commencer_La_Conférence ∩ Parler ∩ Fermer_La_Conférence ∩ Poser_La_Question)

A_Une_Partie_De_Donner_Conférence ⊆ A_Une_Partie.

A_ Commencer_La_Conférence ⊆ A_Une_Partie_De_Donner_Conférence

A_Parler ⊆ A_Une_Partie_De_Donner_Conférence **(02)**

A_Fermer_La_Conférence ⊆ A_Une_Partie_De_Donner_Conférence

A_Poser_La_Question ⊆ A_Une_Partie_De_Donner_Conférence

A_ Fermer_La_Conférence ⊆ A_Une_Partie_De_Donner_Conférence

A_Répondre_à_La_Question ⊆ A_Une_Partie_De_Donner_Conférence

T ⊆ ∀ A_Une_Partie_De_Donner_Conférence . Partie_De_Donner_Conférence

T ⊆ ∀ A_Commencer_La_Conférence. Commencer_La_Conférence

T ⊆ ∀ A_Parler. Parler **(03)**

T ⊆ ∀ A_Fermer_La_Conférence. Fermer_La_Conférence

T ⊆ ∀ A_Poser_La_Question. Poser_La_Question

T ⊆ ∀ A_Répondre_à_La_Question. Répondre_à_La_Question

Donner_La_Conférence ⊆ Partie_Processus_de_commande		
	∀ A_Une_Partie. Partie_De_Donner_Conférence	
	∩ ≥ 1 A_Une_Partie_De_Donner_Conférence	
	∩ ≤ 1 A_Commencer_La_Conférence	**(04)**
	∩ ≤ 1 A_Parler	
	∩ ≤ 1 A_Fermer_La_Conférence	
	∩ ≤ 1 A_Poser_La_Question	
	∩ ≤ 1 A_Répondre_à_La Question	

Tableau 5.1 L'espace de configuration (TBox) pour le problème de configuration Donner_Conférence

Dans le **Tableau 5.1**, qui représente l'espace de configuration (TBox) pour le problème de configuration *Donner_Conférence*, le **TBox** commence par donner ce qu'on appel l'axiome de couverture pour le concept *Partie_De_Donner_Conférence* , aussi bien qu'exiger de *Partie_De_Donner_Conférence* d'être une et une seule partie de concept *Donner_Conférence*. Les axiomes additionnels assurent la disjonction de concepts *Commencer_La_Conférence, Parler, Fermer_La_Conférence, Poser_La_Question et Répondre_à_La Question* (01). Alors **TBox** présente la hiérarchie de rôle, déclarant que le rôle *a_Une_Partie_De_Donner_Conférence* est un sous rôle de rôle *a_une_Partie,* et les rôles *a_Commencer_La_Conférence, a_Parler, a_Fermer_La_Conférence, a_Poser_La_Question et a_Répondre_à_La Question* sont des sous rôles de rôle a_*Une_Partie_De_Donner_Conférence* (02).

La suite de **TBox** est une série de restrictions imposantes sur les rôles (03). Enfin des exigences sont énoncées pour le concept *Donner_Conférence* (04). Il doit être une *Partie du processus de Conférence*, toutes les Parties qu'il a doivent être des instances de concept *Partie_De_Donner_Conférence,* et il doit avoir au moins une *Partie_De_Donner_Conférence* , au plus *l'activité Commencer_La_Conférence,* au plus *l'activité Parler,* au plus *l'activité Fermer_La_Conférence,* au plus *l'activité Poser_La_Question et* au plus *l'activité Répondre_à_La Question.*

Dans notre exemple l'ABox initial est très simple :

A = {dc : Donner_Conférence}.

Si l'utilisateur choisit les variantes : ***Commencer_La_Conférence, Parler, Fermer_La_Conférence*** (résout la variabilité actuelle dans l'activité ***Donner_Conférence***). Ainsi que le choix d'activité ***Déverouiller_Salle.***

ABox deviendraient :

A *= {dc : Donner_Conférence, cc : Commencer_La_Conférence, a_Commencer_La_Conférence (dc ; cc), p : Parler, a_Parler (dc ; p), fc : Fermer_La_Conférence, a_Fermer_La_Conférence(dc ; fc),* ***ds :****Déverouiller_Salle, pc : Processus_Conférence , a_Déverouiller_Salle (pc ; ds)}.*

Ici *dc; cc, pc, ds ,fc,* sont des individus des concepts : *Donner_Conférence, Commencer_La_Conférence, Processus_conférence, Déverouiller_Salle* et *Fermer_La_Conférence* respectivement.

La base de connaissances devrait être testée pour assurer sa cohérence et, par conséquent le choix de l'utilisateur sera validé.

Notre base de connaissances manque encore de dépendances entre les variantes. Comme nous pouvons voir dans le modèle des caractéristiques Figure. 5.2, le choix de variante *Donner_Conférence* nécessite de choisir la variante *Déverouiller_Salle*. Pour tenir une telle contrainte dans notre base de connaissances nous avons besoin de règles.

L'un des formalismes basés sur les règles, qui sont décidables et souvent utilisées, sont les règles de Horn à libres fonctions[MOTIK05].

Donc, nous allons ajouter des règles dans notre base de connaissances comme suit :

- La règle selon laquelle le choix de variante *Donner_Conférence* nécessite le choix de variante *Déverouiller_Salle,* Peut être écrite comme suit *:*

  a_ Donner_Conférence (pc; dc) → a_Déverouiller_Salle (pc; ds) (01)

- La règle selon laquelle le choix de variante *Parler* nécessite le choix de variante *Commencer_La_Conférence, peut être écrite comme suit :*

  a_ Parler (dc; p) → a_Commencer_La_Conférence (dc; cc) (02)

- La règle selon laquelle le choix de variante *Commencer_La_Conférence* nécessite le choix de variante *Déverouiller_Salle, peut être écrite comme suit :*

  a_Commencer_La_Conférence (dc; cc) → a_Déverouiller_Salle (pc; ds) (03)

- La règle selon laquelle le choix de variante *Fermer_La_Conférence* nécessite le choix de variante *Commencer_La_Conférence, peut être écrite comme suit :*

a_Fermer_La_Conférence (dc; fc) → a_Commencer_La_Conférence(dc; cc) (04)

- La règle selon laquelle le choix de variante Répondre_à_La_Question nécessite le choix de variante *Poser_La_Question, peut être écrite comme suit :*

a_ Répondre_à_La_Question (dc; rq) → a_ Poser_La_Question (dc; pq) (05)

Ici *dc; cc, pc, ds ,fc, pq et rq* sont des individus des concepts : *Donner_Conférence, Commencer_La_Conférence, Processus_conférence, Déverouiller_Salle, Fermer_La_Conférence, Poser_La_Question et Répondre_à_La_Question* respectivement. La présence des règles dans la base de connaissances signifie que le raisonneur doit être capable de les comprendre et les exécuter au besoin. La Figure 5.11 (Décrite en utilisant UML) représente un exemlpe du processus de Conférence après la configuration.

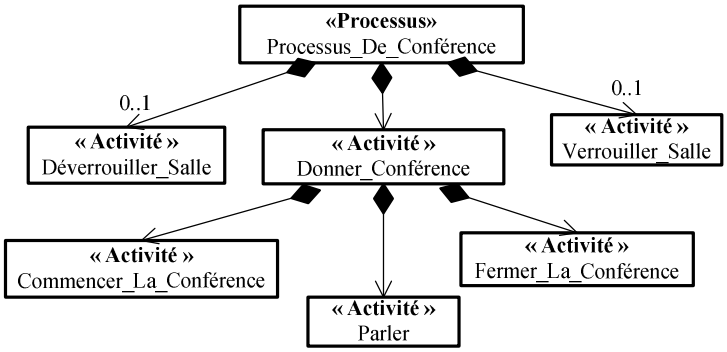

Figure 5.11. La configuration du processus de Conférence

Nous remarquons que pendant cette étape, certains points de variation, peuvent être restés non résolus, la résolution ou la prise de décisions sur ces points de variations sera durant la phase de l'exécution de processus.

Le processus d'affaires configuré et l'ontologie du domaine d'application sont des entrées pour la prochaine étape, qui est l'attribution des rôles.

2.2.3. L'attribution des rôles

L'attribution des rôles est réalisée en correspondant les requises aux acquises, selon l'algorithme présenté dans le chapitre précédent, qui est proposé dans [ČIUK07a] et enrichie par les concepts qu'nous avons ajoutés.

Donc et d'après cet algorithme, le concept ***Acquises_De_Déverouillage_et_Donner_Un_Cours*** englobant les deux requises correspondants aux concepts ***Requises_De_Déverouillage*** et ***Requises_Pour_Donner_Une_Conférence.***

Alors cette entité qui a comme Acquises ce concept *Acquises_De_Déverouillage_et_Donner_Un_Cours,* doit être reconstruite et divisée en

plusieurs entités spécifiques (la spécification des Acquises). Nous avons donc les deux entités *Chargé_de_Cours* et *Veilleur_de_Nuit* qui ont comme acquises *Acquises_Pour_Donner_Un_Cours* et *Acquises_De_Déverouillage* respectivement.

Donc, l'ontologie du domaine d'application reconstruite est illustrée dans la Figure 5.12.

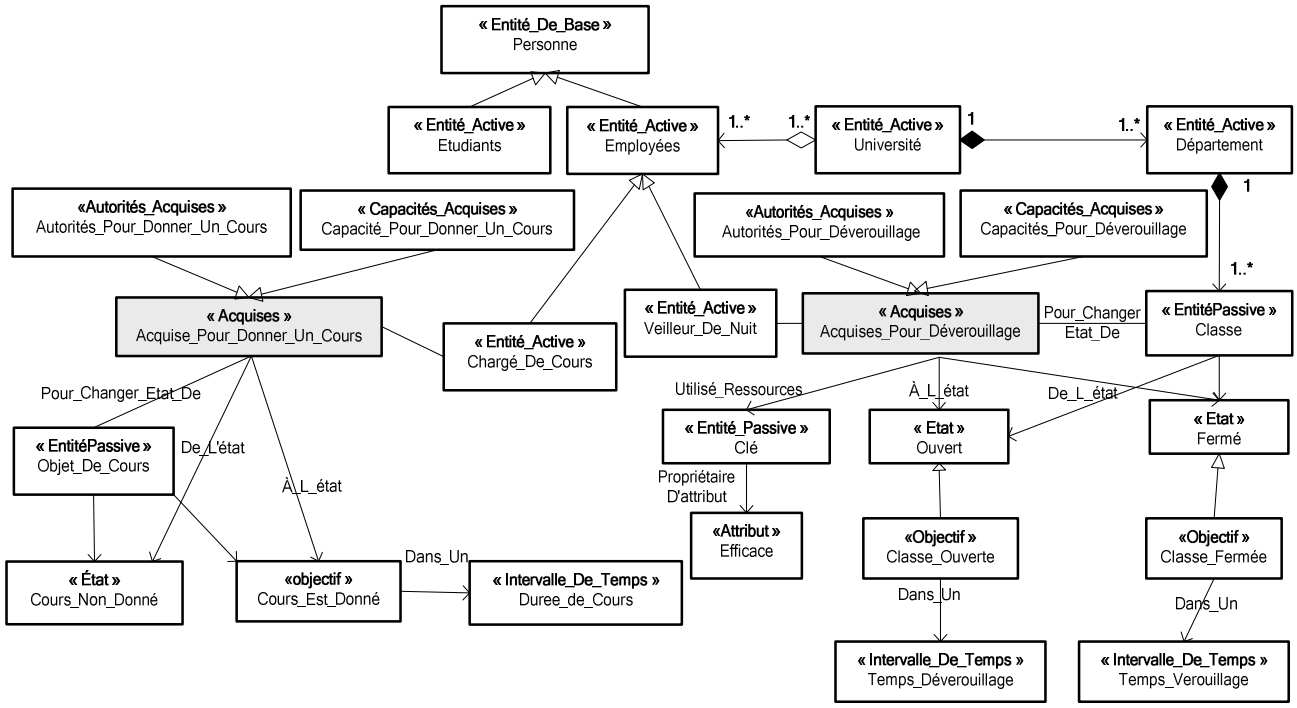

Figure 5.12. Un fragment d'ontologie du domaine d'application reconstruite

En commençant une autre comparaison on trouve que les acquises *Acquises_Pour_Donner_Un_Cours* ***englobant*** les trois requises correspondant aux concepts *Requises_Pour_Commencer_Conférence, Requises_Pour_Fermer_Conférence* et *Requises_Pour_Parler.* Alors cette entité qui a comme Acquises ce concept *Acquises_Pour_Donner_Un_Cours*, doit être reconstruite et divisée en plusieurs entités spécifiques (la spécification des Acquises).

Nous avons donc les deux entités : *Chargé_de_Cours* qui a comme acquises *Acquises_Pour_Expliquer_Un_Cours* et l'entité *Technicien* qui a les deux acquises *Acquises_Pour_Commencer_Le_Cours* et *Acquises_Pour_Terminer_Le_Cours* respectivement.

De ce fait, l'ontologie du domaine de l'application reconstruite , une fois encore, est illustrée dans la Figure 5.13.

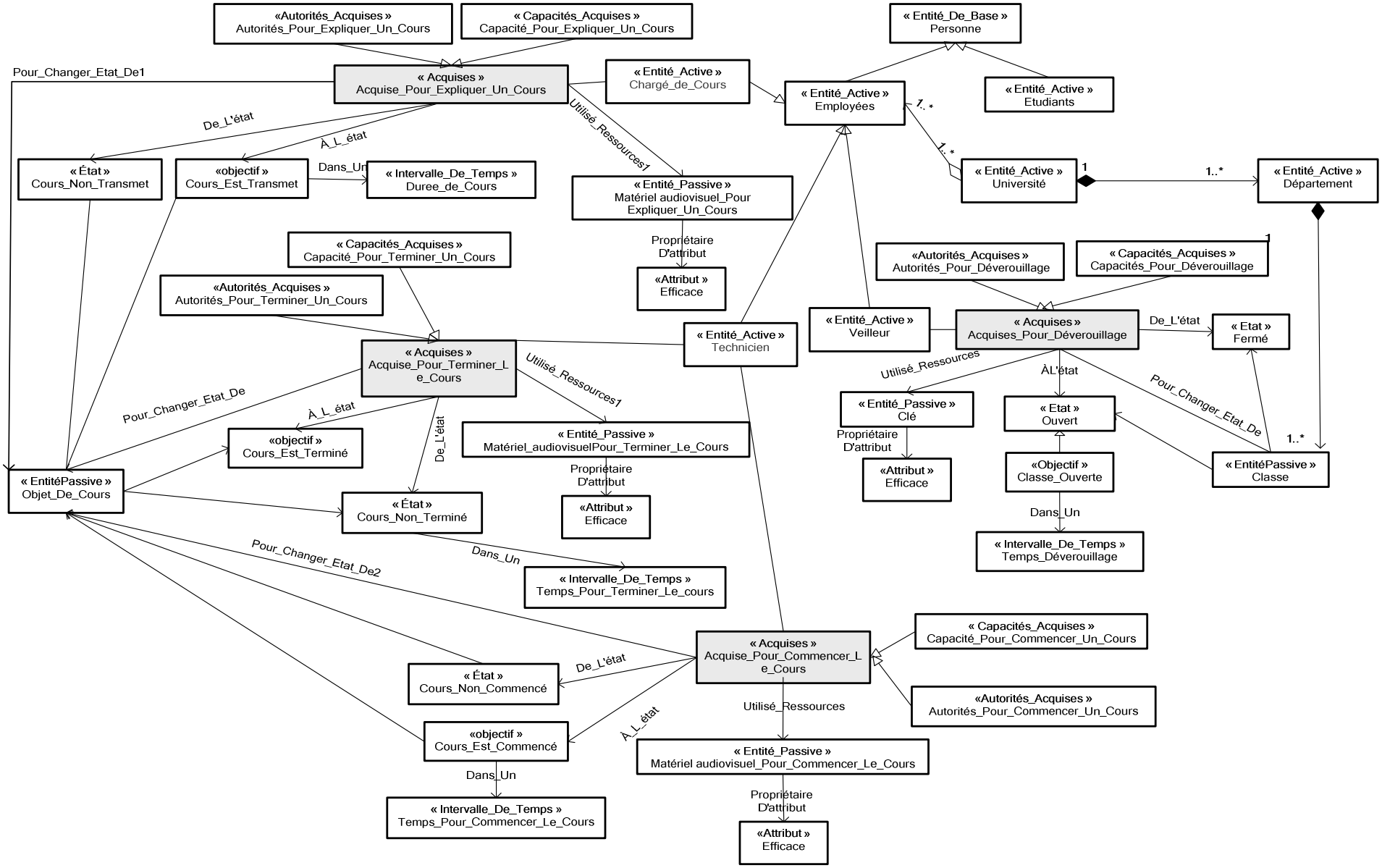

Figure 5.13. Un fragment d'ontologie finale du domaine d'application reconstruite

Dans ce cas nous avons les attributions Entité/ Rôle, qui sont illustrés dans ce Tableau 5.2:

Rôle		**Entité**	
Acteur	*Responsable_Des_Clés*	**Entité Active**	*Veilleur_de_Nuit*
	Locuteur		*Chargé_de_Cours*
	Intervenant		*Technicien*
Entrée	*Salle Vérrouiller*	**Entité Passive correspondant à la relation Pour_Changer _État_de**	*Classe*
	Objet_De_Parole		*Objet_De_Cours*
	Objet_De_Parole		*Objet_De_Cours*
	Conférence Donnée		*Objet_De_ Cours*
Sortie	*Salle Déverrouiller*	**Objectif associé à l'Entité Passive correspondant à la relation Pour_Changer _État_de**	*Classe_Fermée*
	Objet_De_Parole_Donné		*Cours_Est_Transmet*
	Conférence_Commencée		*Cours_Est_Commencé*
	Conférence_Fermée		*Cours_Est_Fermé*
Ressource	*Instrument_Déverrouillage*	**Entité Passive correspondant à la relation Utilisé_Ressource**	*Clé*
	Instrument_Pour_ Commencer_Conférence		*Matériel_AudioVisuel_Pour _Commencer_Cours*
	Instrument_Pour_ Terminer_Conférence		*Matériel_AudioVisuel_Pour _Terminer_Cours*
	Instrument_Pour_ Expliquer_Conférence		*Matériel_AudioVisuel_Pour _Expliquer_Cours*

Tableau 5.2 Attributions d'Entités/ Rôles

Le résultat de cette phase est un modèle du processus d'affaires localisé dans un domaine d'application choisi et peut avoir des points de variation qui serons résolus durant l'exécution de ce processus, nous les appelons des points de variation dynamiques.

D'autre part et durant la phase de l'ingénierie d'application du processus, nous pouvons aussi définir d'autres points de variation dynamiques qui seront aussi résolus durant la phase d'exécution de ce processus. La définition de ces points de variation peut être faite durant l'implémentation de ce processus, par un langage dédié.

2.2.4. La définition du flow de contrôle et la description du modèle exécutable de processus cours

Cette activité nous permet d'ajouter les connaissances de contrôle qui définissent l'ordre d'exécution de ces activités [ČIUK07a]. Par conséquence, un modèle exécutable de processus cours est produit. Nous pouvons utiliser un langage d'exécution de processus tel que WSBPEL [BPEL07], qui va exécuter par un système de gestion de

Workfows. Comme déjà mentionné dans le chapitre 4, les connaissances de contrôle (Figure 5.14 (Décrite en utilisant UML)) pour le processus générique de conférence sont définies séparément.

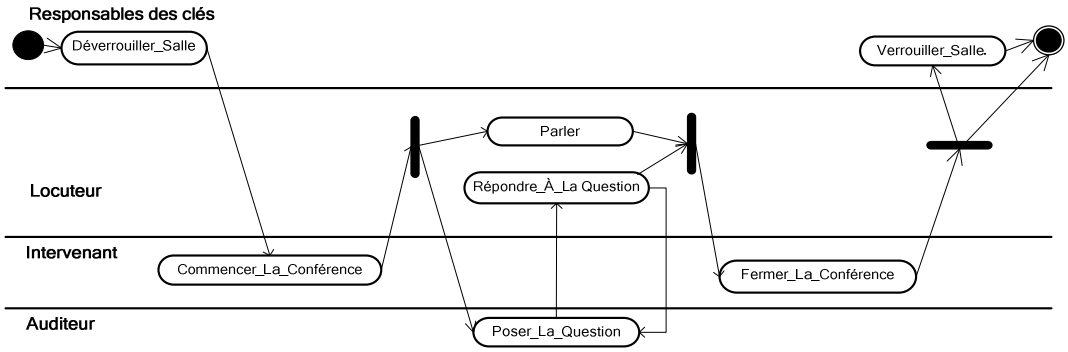

Figure 5.14. Connaissances de contrôle pour le processus générique de conférence

Ces connaissances de contrôle sont ajoutées au processus de cours pour avoir un modèle exécutable de ce processus (Figure 5.15 (Décrite en utilisant UML)).

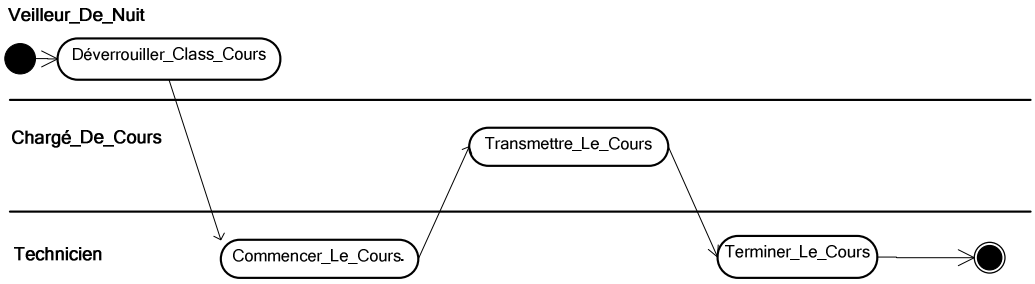

Figure 5.15. Diagramme d'activités représentant le processus de cours

2.3. La phase d'exécution du processus cours

La troisième phase de notre approche à pour objectif de gérer l'exécution de processus de Cours défini précédemment par un langage tel que WSBEL. Cela par la résolution des variabilités qui existent encore dans ce modèle exécutable du processus de Cours. Dans cette étape, il faut prendre les décisions qui correspondent au choix d'activités à exécuter, comme elles sont définies dans les phases précédentes et selon la disponibilité des services exigés pour l'exécution de chaque activité.

Ces services peuvent être fournis par d'autres activités de processus lui-même ou disponible sur l'environnement d'exécution.

Ainsi, pendant l'engénierie applicative, on peut ajouter d'autres points de variation. Par exemple, dans le pocessus cours, on peut donner un cours que ce soit Localement ou bien à distance. Donc, nous avons le diagramme des caractéristique (Figure 5.16) suivant :

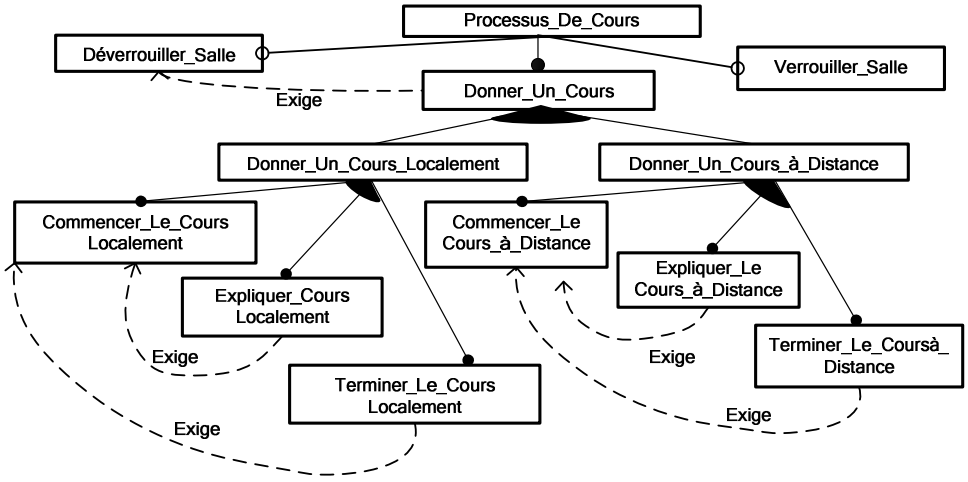

Figure 5.16. Modèle des caractéristiques de processus cours

Dans ce cas là, nous avons le point de variation , entre les deux activités génériques *Donner_Un_cours_Localement* et *Donner_Un_cours_À_Distance,* qui va être résolu durant l'exécution du ce processus. Pour cela, par exemple, si nous n'avons pas un protocole ou un service pour faire un cours à distance, nous pouvons le faire localement. Alors, nous allons choisir l'exécution de l'activité générique *Donner_Un_Cours_Localement* ou lieu de l'activité générique *Donner_Un_cours_À_distance.* Cette variation, illustré dans ce modèle des caractéristiques, peut avoir lieu dans l'ingénierie du domaine de processus Conférence.

Nous pouvons synthétiser d'après l'exemple montré précédement, pour assurer la fléxibilité et la dynamique d'exécution d'un processus d'affaires, nous devons avoir des points de variation qui vont être résolus durant l'exécution de ce processus. Ce choix est assuré par le langage de programmation que nous utiliserons pour coder ce processus et le système de gestion de Workflow.

Pour ce qui concerne l'aspect d'implémentation de notre proposition, nous allons le présénter dans la section suivante.

3. L'aspect d'implémentation

Nous proposons l'utilisation de l'outil Protégé, qui est fondé sur le modèle de la logique de description, Protégé version 3.4.6. Cet outil fournit une interface modulaire,

permettant à l'utilisateur d'éditer, de visualiser et d'enregistrer l'ontologie au format OWL en générant deux types de fichiers : Un fichier projet Protégé (.pprj) et un fichier OWL. Protégé est un outil développé en Java et qui nécessite pour fonctionner la machine virtuelle Java (environnement de Java).

L'ensemble des fonctionnalités de l'éditeur étant regroupées en plusieurs plugins, Nous avons accès à ces fonctionnalités à travers les ongles qui doivent apparaître dans l'éditeur selon le besoin. Le premier onglet présente les classes de l'ontologie. Il est possible de créer, modifier, supprimer une classe, et de lui attacher des propriétés. Ces propriétés peuvent elles-mêmes être caractérisées. Nous présentons dans ce qui suit les principales actions à suivre pour réaliser le fichier OWL-DL. Pour vérifier la consistance et le classement des concepts, nous choisissons Racer (efficace et gratuit) comme un raisonneur. Pour la configuration, nous allons utiliser la logique de description, pour définir les différents concepts et relations de l'ontologie du domaine. Pour définir les règles de dépendance, nous allons utiliser le plugin *SWRL Rules*. Pour l'attribution des rôles, nous allons utiliser la fonction *Map* de plugin *Prompt* de cet outil *Protégé.*

Création d'un nouveau projet OWL

a. Lorsque nous lançons Protégé, une fenêtre s'ouvre et nous demande de choisir entre l'ouverture d'un projet existant et la création d'un nouveau projet. Dans notre cas, nous allons créer un nouveau projet que s'appellera *Ontologie_de_Haut_Niveau.*

b. Cliquons sur "Créer un nouveau projet" (Interface Anglaise «New Project »).

c. Une nouvelle fenêtre s'ouvre et nous demande de sélectionner le type de projet que nous voulons créer. Notre but étant de construire une ontologie OWL, nous allons sélectionner l'option OWL/RDF Files et faire suivant.

d. Le logiciel demande ensuite de nommer et de donner l'URI de l'ontologie. Nous remarquerons qu'une URI est proposée par défaut mais nous allons définir une URI dédiée à notre ontologie *Ontologie_de_Haut_Niveau*. Notons l'URI de l'ontologie comme suit : *http : //localhost8080/Ontologie_de_Haut_Niveau.* et faire suivant.

e. Dans la fenêtre suivante, Protégé nous demande de choisir le profil du langage qui sera utilisé. Choisir le profil OWL DL et faire suivant.

f. Le logiciel nous demande ensuite quelle vue nous souhaitons. Laisser logic view et faire terminer.

g. Naturellement, il faut sauvegarder notre ontologie sur notre machine, pour cela faire Fichier→Sauvegarder le Projet→Nom de Fichier OWL = Ontologie_de_Haut_Niveau.

h. Pour l'instant, elle ne contient aucun concept, n'est pas peuplée et seul l'espace de nom a été défini.

Import d'une ontologie existante

a. Il faut se positionner dans l'onglet Meta data, au niveau de la fenêtre Ontology Browser et cliquer sur importer une ontologie.

b. Une nouvelle fenêtre s'ouvre, choisir, importer depuis un fichier local et faire suivant.

c. Donner le chemin où se trouve l'ontologie à importer et faire suivant puis terminer.

Plus de détails sur l'utilisation de Protégé, il se trouve au niveau de leur site web officiel.

Durant l'implémentation, nous étions confrontés à quatre problèmes principaux à implémenter :

3.1. La construction de différentes ontologies citées dans l'approche

Après avoir choisi l'outil Protégé pour l'implémentation des ontologies, nous commençons la construction des ontologies comme suit :

a. Un nouveau projet OWL pour construire l'ontologie de haut niveau, avec ses différents concepts et divers relations et dépendances entre ces concepts *"Concepts", "Entité", "Relation", " Rôle" et "État_D_Affaire", "Temps"* (Figure 5.17, Figure 5.18).

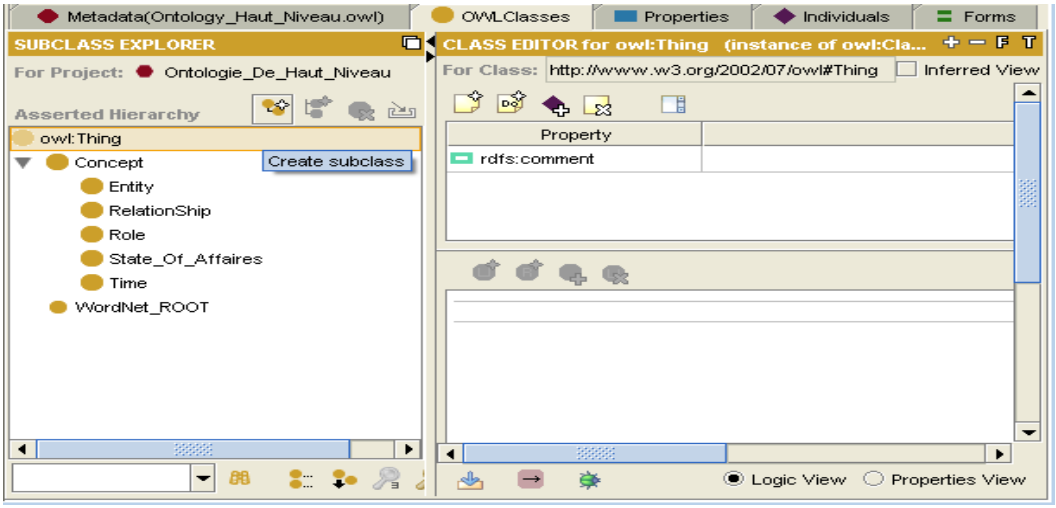

Figure 5.17. Une partie de vue logique de l'ontologie de haut niveau avec l'outil Protégé

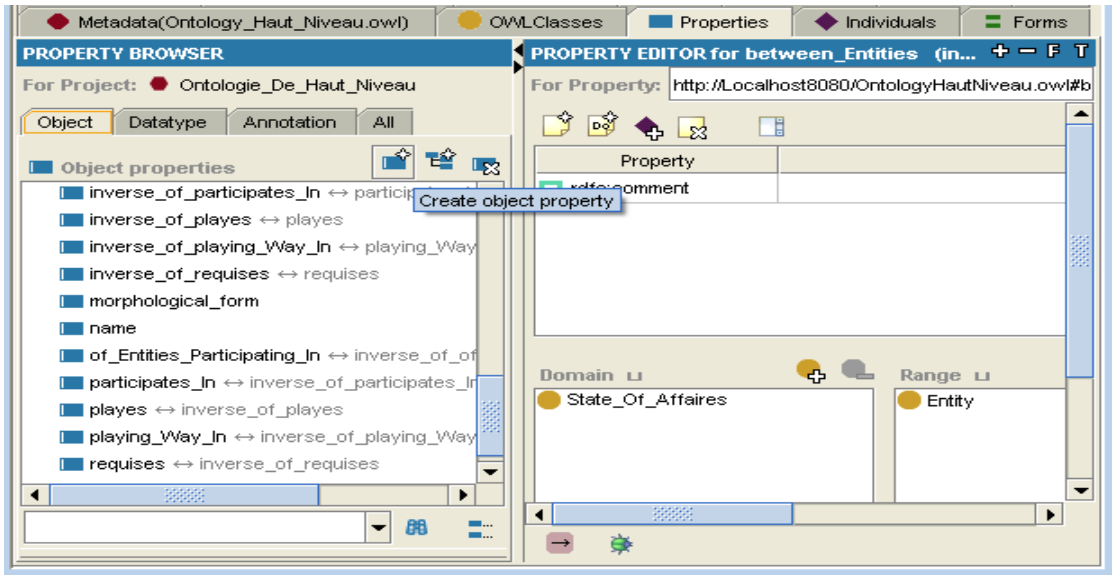

Figure 5.18. Une partie des propriétés d'ontologie de haut niveau avec l'outil Protégé

b. Ensuite, nous créons un nouveau projet OWL pour construire l'ontologie du processus d'affaires de haut niveau, avec ses différents concepts et diverses relations et dépendance entre ces concepts tels que : Acteurs, entrées, sorties et ressources, etc. Puisque l'ontologie de haut niveau est considérée comme un méta-concept pour les deux autres types d'ontologies déjà cités, nous l'importons puis nous construisons l'ontologie du processus d'affaires de haut niveau par l'instanciation des concepts de *l'ontologie de haut niveau* (Figure 5.19, Figure 5.20).

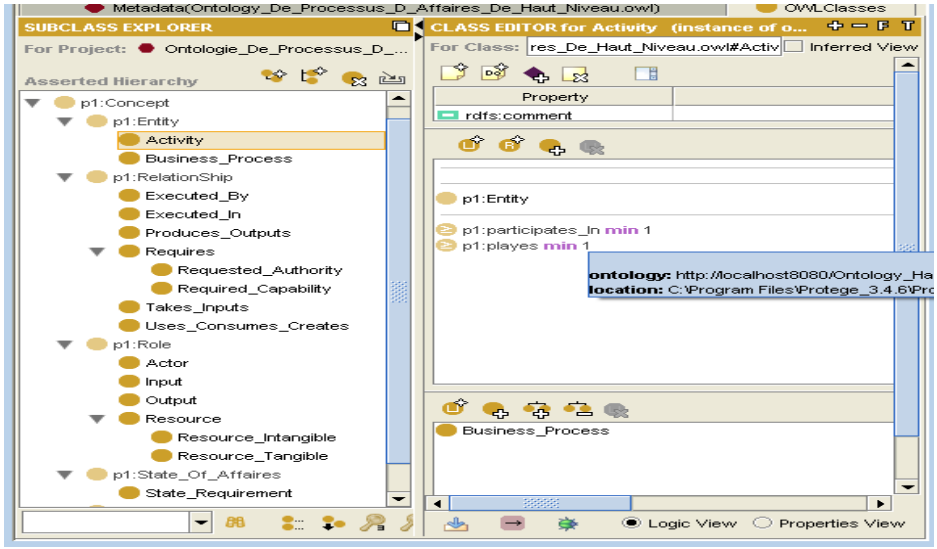

Figure 5.19. Une partie de vue logique de l'ontologie de processus d'affaires de haut niveau

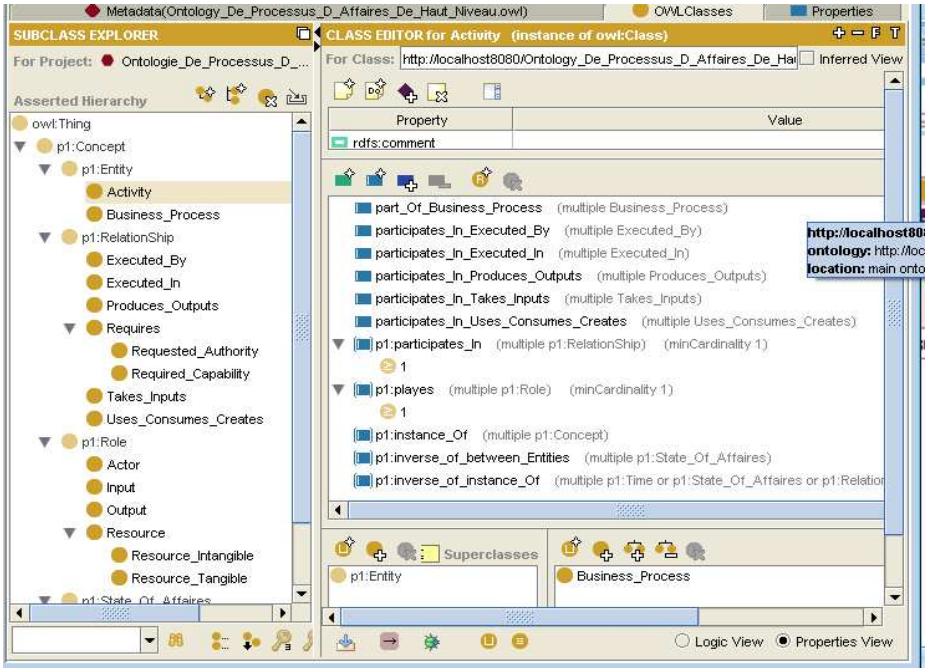

Figure 5.20. Une partie de vue des propriétés d'ontologie du processus d'affaires de haut niveau

c. Dans un nouveau projet, nous importons l'ontologie du processus d'affaires de haut niveau pour construire l'ontologie du processus de conférence (Figure 5.21, Figure 5.22) qui est un processus générique.

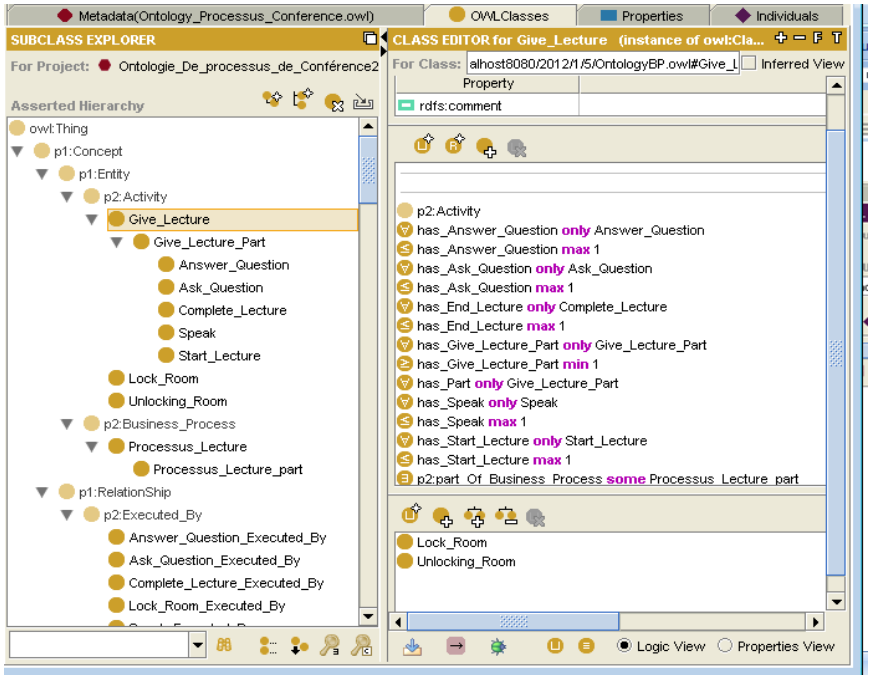

Figure 5.21. Une partie de vue logique de l'ontologie de processus de conférence avec l'outil Protégé

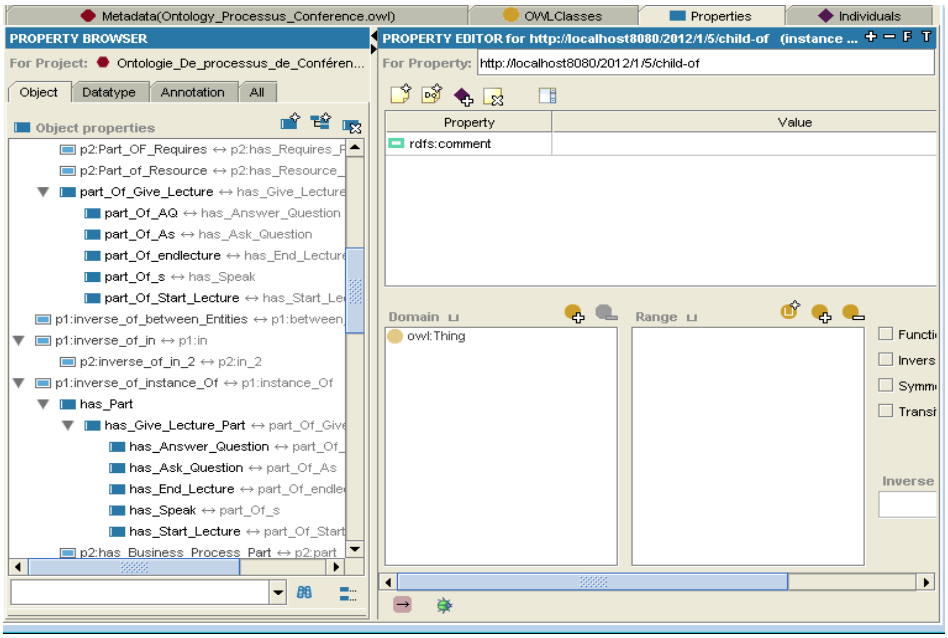

Figure 5.22. Une partie des propriétés d'ontologie du processus de conférence

d. Durant l'ingénierie du domaine applicative de processus, nous créons un troisième projet OWL, de la même façon que l'ontologie du processus d'affaires de haut niveau. Nous construisons *l'ontologie du domaine d'application de haut niveau* par l'instanciation des concepts de *L'ontologie de haut niveau.* De ce fait, dans un nouveau projet nous importons l'ontologie de haut niveau pour construire l'ontologie du domaine d'application de haut niveau. Elle est aussi implémentée avec ses différents concepts et diverses relations et dépendances entre ces concepts *entités actives* (Postes *de travail, Systèmes d'application et Unités organisationnelles) et entités passives, Intervalle de temps, "Objectif", Acquises, Autorités, Capacités, etc.* (Figure 5.23, Figure 5.24).

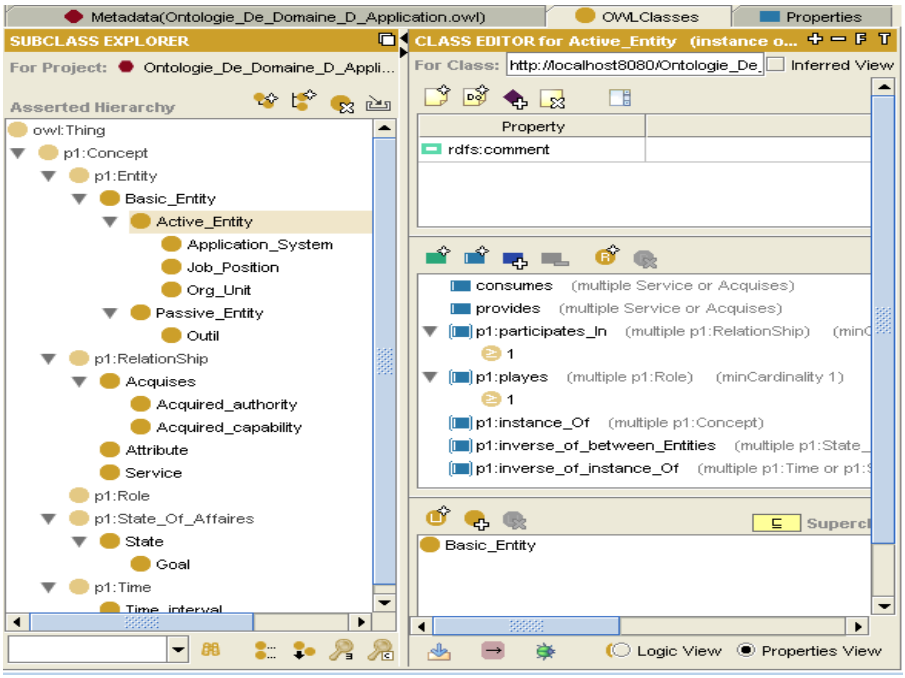

Figure 5.23. Une partie de vue des propriétés d'ontologie *du domaine d'application de haut niveau*

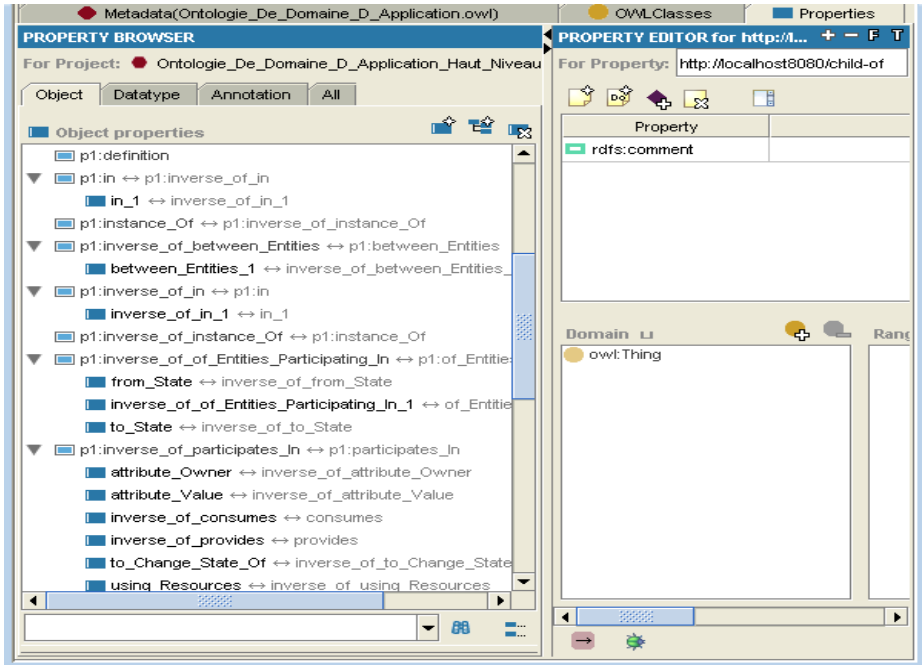

Figure 5.24. Une partie des propriétés d'ontologie du domaine d'application de haut niveau

Dans un autre nouveau projet, nous importons l'ontologie du domaine d'application de haut niveau, pour construire l'ontologie du processus de Cours que nous avons choisi comme un domaine d'application (Figure 5.25, Figure 5.26, Figure 5.27).

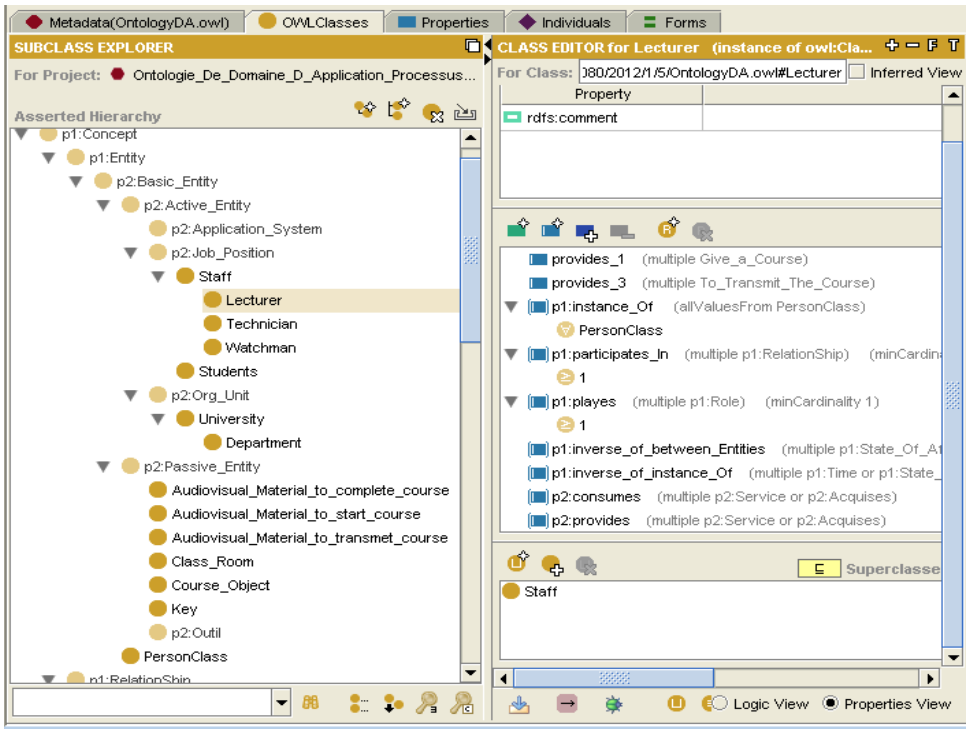

Figure 5.25. Une partie de vue des propriétés d'ontologie du processus Cours

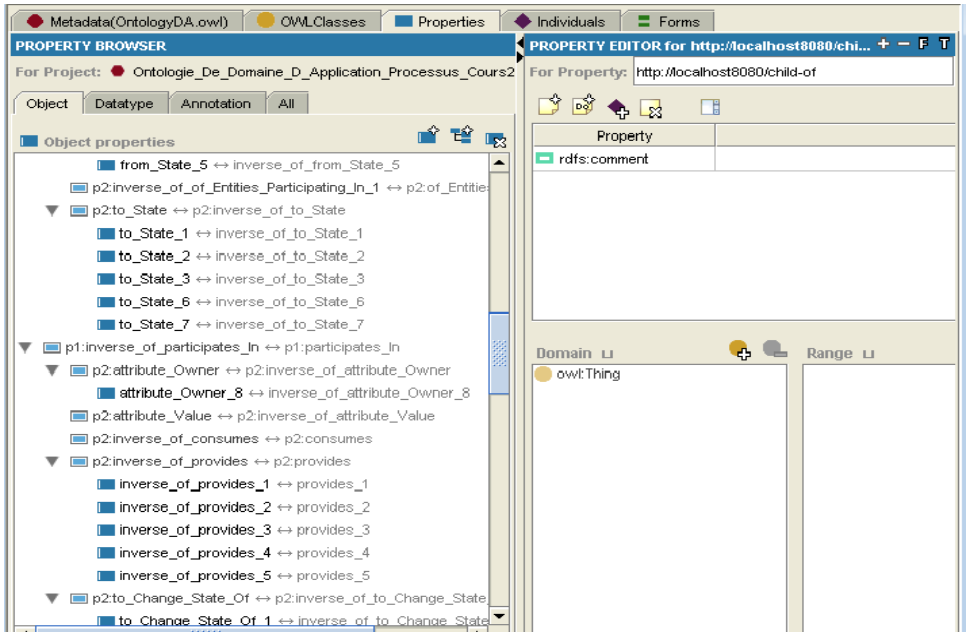

Figure 5.26. Une partie des propriétés d'ontologie du processus Cours

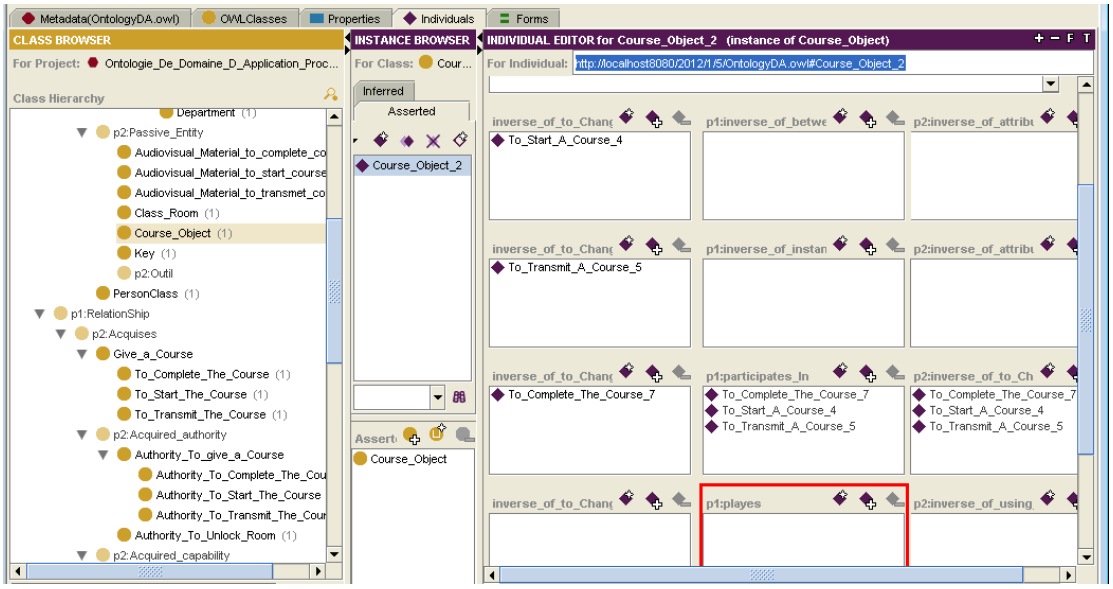

Figure 5.27. Une partie d'ontologie du processus cours avec leur différentes relations et dépendances

Pour vérifier la consistance et le classement des concepts dans toutes les ontologies que nous avons construites, nous utiliserons le raisonneur Racer. Pour cela, nous exécutons le serveur du raisonneur Racer, puis avec les instructions de protégé nous invoquons ce serveur pour faire les vérifications nécessaires.

Pour chaque ontologie construite, nous avons un extrait du fichier OWL dans l'annexe A.

3.2. La configuration du processus de conférence correspondant au processus de cours

Pour l'implémentation de cette partie, l'outil Protégé permet de définir l'espace de configuration (TBox) pour un problème de configuration en ajoutant des conditions sur les différents concepts appartenant au problème de configuration, ici, le problème de configuration est dans l'activité présenté par le concept *Donner_Conférence* de modèle des caractéristiques du processus de conférence. Les figures suivantes : Figure 5.28, Figure 5.29, Figure 5.30, Figure 5.31 et Figure 5.32, Figure 5.33 et Figure 5.34, représentent les conditions ajoutées sur les concepts : *Donner_Conférence*, *Partie_De_Donner_Conférence*, *Commencer_La_Conférence*, *Parler*, *Fermer_La_Conférence*, *Poser_La_Question* et *Répondre_à_La_Question*, respectivement.

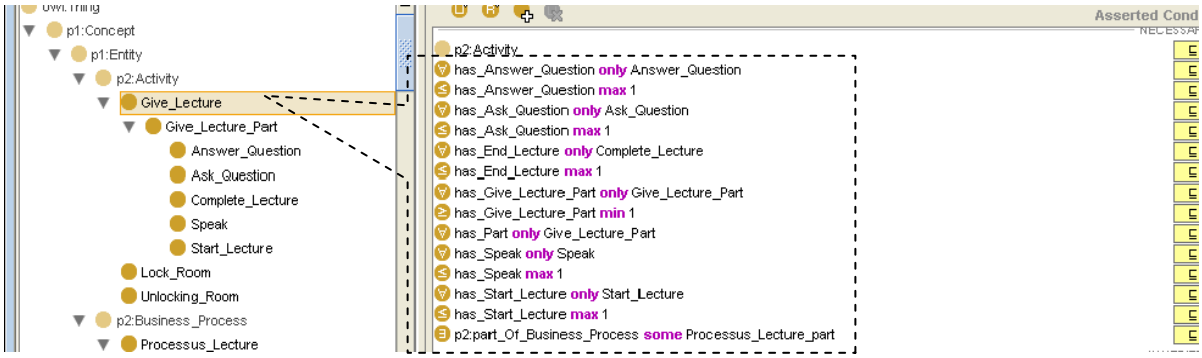

Figure 5.28. Conditions ajoutées sur le concept *Donner_Conférence (Give_Lecture)* avec l'outil Protégé

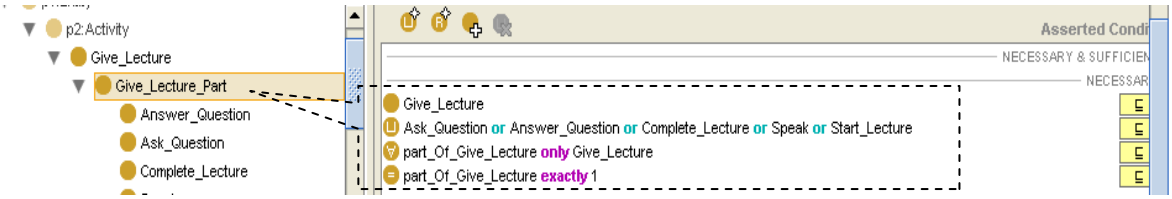

Figure 5.29. Conditions ajoutées sur le concept *Partie_De_Donner_Conférence (Give_Lecture_Part)* avec l'outil Protégé

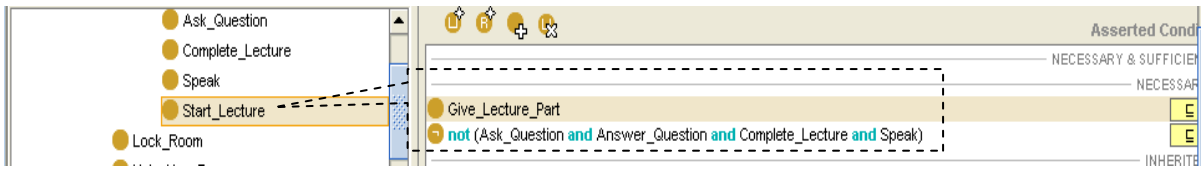

Figure 5.30. Conditions ajoutées sur le concept *Commencer_La_Conférence (Start_Lecture)* avec l'outil Protégé

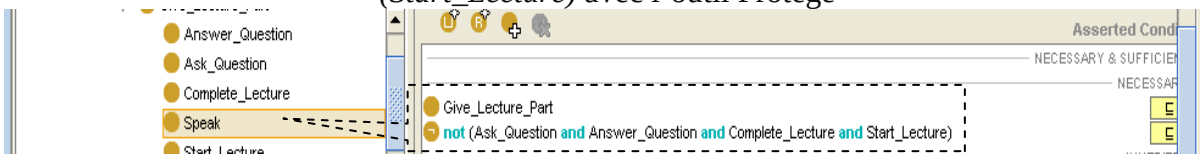

Figure 5.31. Conditions ajoutées sur le concept *Parler(Speak)* avec l'outil Protégé

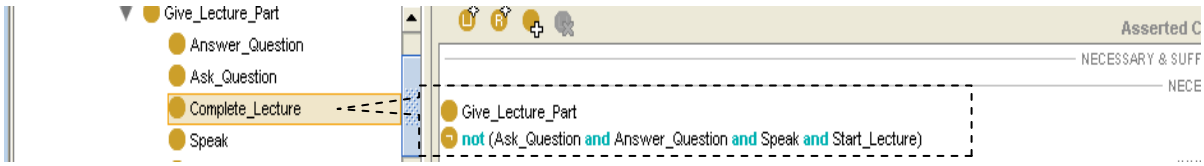

Figure 5.32. Conditions ajoutées sur le concept *Fermer_La_Conférence (Complete_Lecture)* avec l'outil Protégé

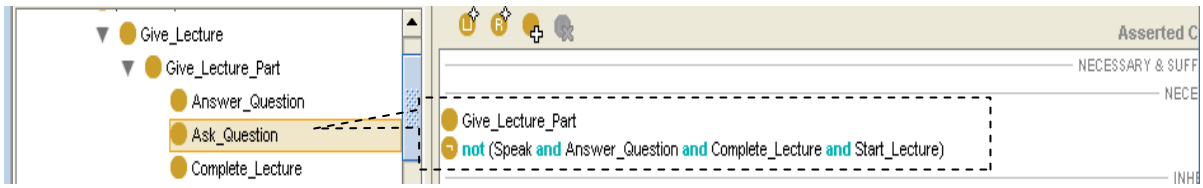

Figure 5.33. Conditions ajoutées sur le concept *Poser_La_Question (Ask_Question)* avec l'outil Protégé

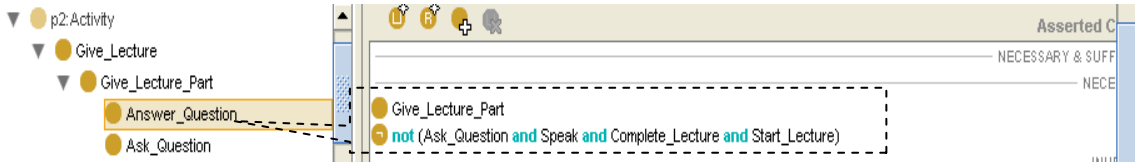

Figure 5.34. Conditions ajoutées sur le concept *Répondre_à_La_Question (Answer_Question)* avec l'outil Protégé

Pour l'implémentation d'ABox, il inclut exclusivement les variantes obligatoires. C'est-à-dire on doit instancier seulement ces variantes, ensuite et comme nous l'avons mentionné auparavant, le choix des variantes optionnelles ou alternatives nécessite que ABox soit augmenté avec les individus correspondants et les relations existantes entre-elles.

Cela veut dire, qu'il faut instancier aussi et seulement les variantes optionnelles ou alternatives choisies durant la résolution de variabilités. Sans omettre de relier ces variantes instanciées avec les relations (Rôles) qui conviennent. La Figure 5.35, représente le choix des variantes *Commencer_La_Conférence, Parler et Fermer_La_Conférence* dans le problème de configuration *Donner_Conférence* avec l'outil Protégé. Cela par l'instanciation de ces variantes, ensuite on les relie avec les relations (Rôles) qui conviennent. Par contre, les variantes *Poser_La_Question* et *Répondre_à_la_Question* non ne sont pas choisies. Par conséquent, elles ne sont pas instanciées.

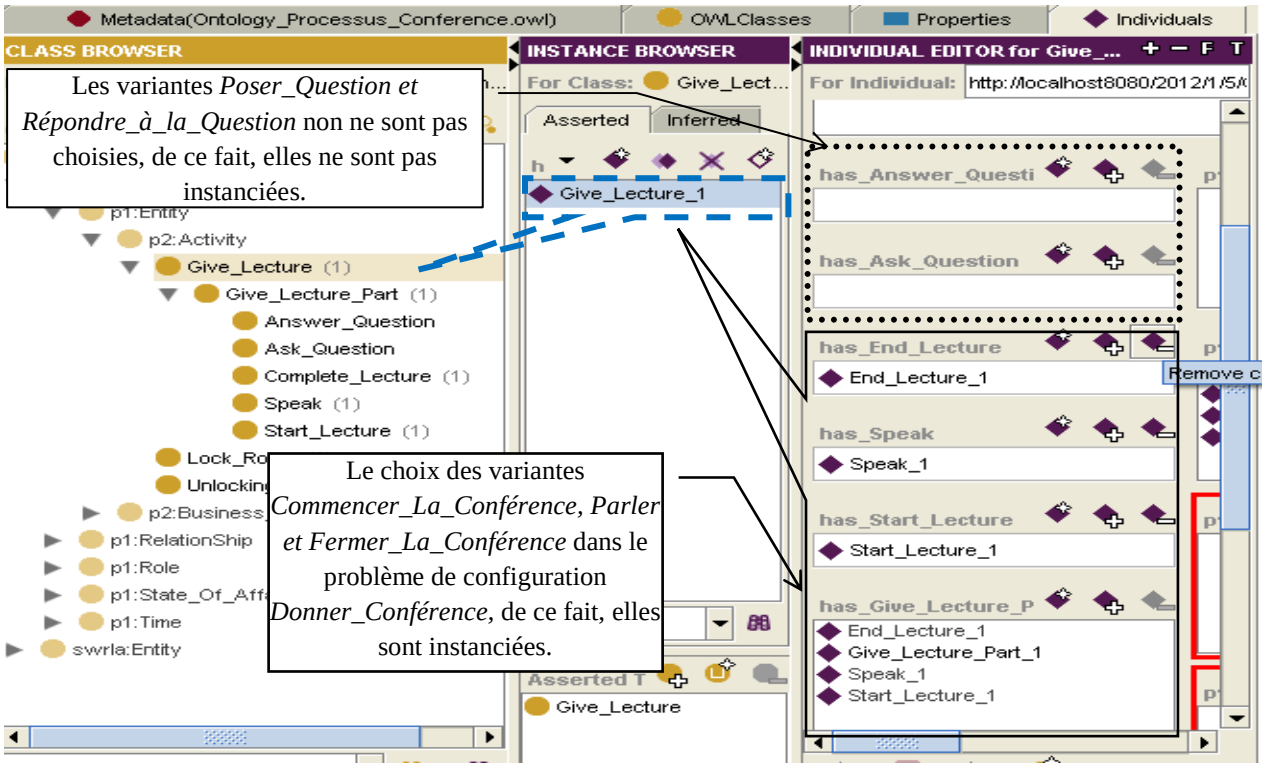

Figure 5.35. Choix des variantes selon le domaine d'application

Pour les règles qui assurent les dépendances entre les variantes, protégé offre aussi un plugin ***SWRL Rules*** pour construire ces règles. Les règles *(01), (02), (03), (04) et (05)* sont implémentées comme elles sont illustrées dans la Figure 5.36.

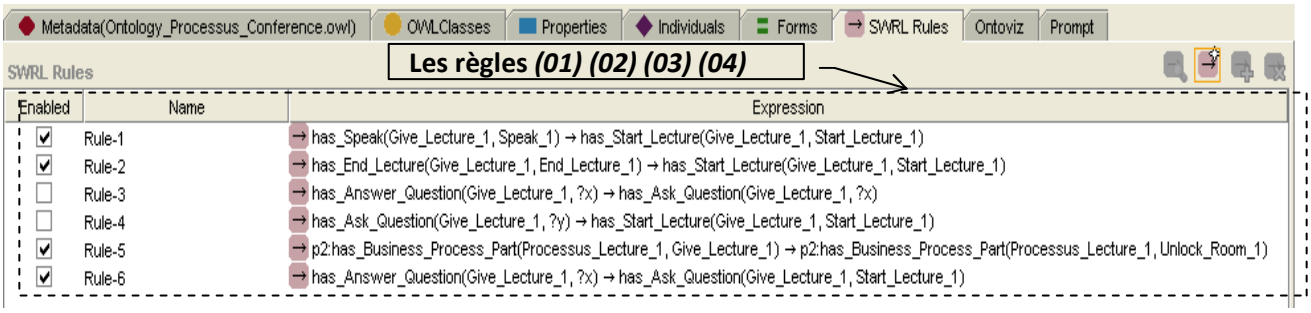

Figure 5.36. Les règles *(01), (02), (03), (04), (05) sont* implémentées avec l'onglet SWRL Rules d'outil Protégé

Nous remarquons que pendant cette étape, certains points de variation, peuvent être restés non résolus, la résolution ou la prise de décision sur ces points de variations se réalisera durant la phase d'exécution du processus.

Le processus de conférence configuré et l'ontologie du domaine d'application choisi, c'est-à-dire l'ontologie du processus de cours, sont des entrées pour la prochaine étape qui concerne l'attribution des rôles.

3.3. L'attribution des rôles

a. Pour l'implémentation de cette étape, au début nous avons intégré l'outil protégé avec Eclipse l'éditeur de Java pour faire le matching (Correspondance) entre les concepts des deux types d'ontologies déjà définis. Après, nous avons commencé à faire une recherche sur les méthodes et les techniques qui permettent de réaliser cette activité. Il y a beaucoup de travaux, traitant ce problème, que ce soit le matching selon la structure ou selon la sémantique ou bien la combinaison des deux, nous avons déjà introduit le résumé de ces techniques dans le premier chapitre. Chacune de ces techniques a ses avantages et ses inconvénients, même durant leur combinaison, il y'avait eu des inconvénients. Durant cette étape nous avons trouvé que l'outil protégé avait un plugin appelé Prompt qui constituait un ensemble de fonctions. Parmi elles la fonction Map qui peut faire le mapping ou la correspondance entre deux ontologies d'une manière interactive et itérative, qui satisfait nos besoins. Par conséquence, au lieu d'implémenter notre propre fonction, actuellement nous choisissons d'utiliser cette fonction « Map »pour gagner du temps.

b. Donc, pour réaliser l'attribution des rôles, tout abord nous devons ouvrir le projet de l'ontologie du processus conférence dans l'outil Protégé. Ensuite nous devons choisir l'onglet Prompt, nous utilisons la fonction « Map » qui propose quatre algorithmes pour faire la comparaison initiale : *Lexical Matching, FOAM Plugin for prompt, Lexical Matching with synonyms, Using UMLS Concepts identifiers for Matching.* Nous choisissons aussi l'un des ces algorithmes après l'insertion de projet cible, dans notre cas le projet de l'ontologie de processus cours.

c. la Figure 5.37 représente le choix de la fonction *« Map »* de plugins *« Prompt »*, le choix de l'algorithme de comparaison initiale ainsi que l'insertion de fichier qui représente le projet de l'ontologie de processus cours.

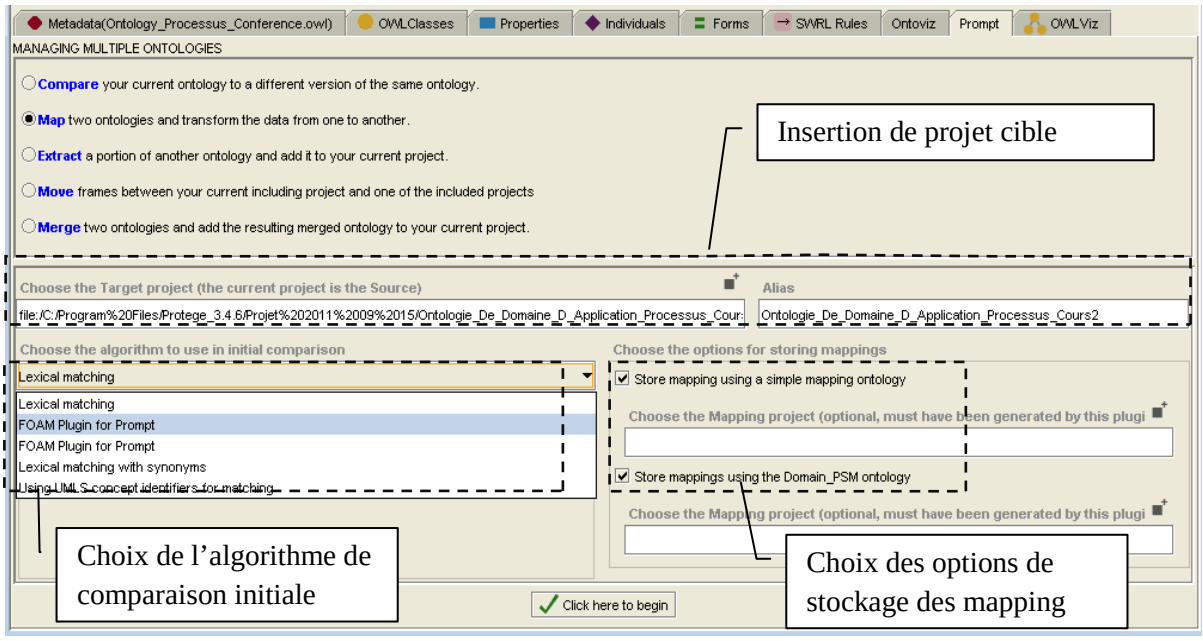

Figure 5.37. La fonction Map de plugins Prompt dans l'outil Protégé

d. Après le choix de l'algorithme de comparaison initiale, on click sur le bouton « *Click Here To Begin* », pour commencer le traitement de matching (correspondance).

e. Une autre fenêtre apparait, qui nous illustre les correspondances proposées entre les concepts des deux ontologies (selon l'algorithme choisi). La Figure 5.38, représente la console de commande qui illustre la fonction de mapping. La Figure 5.39, représente les correspondances proposées.

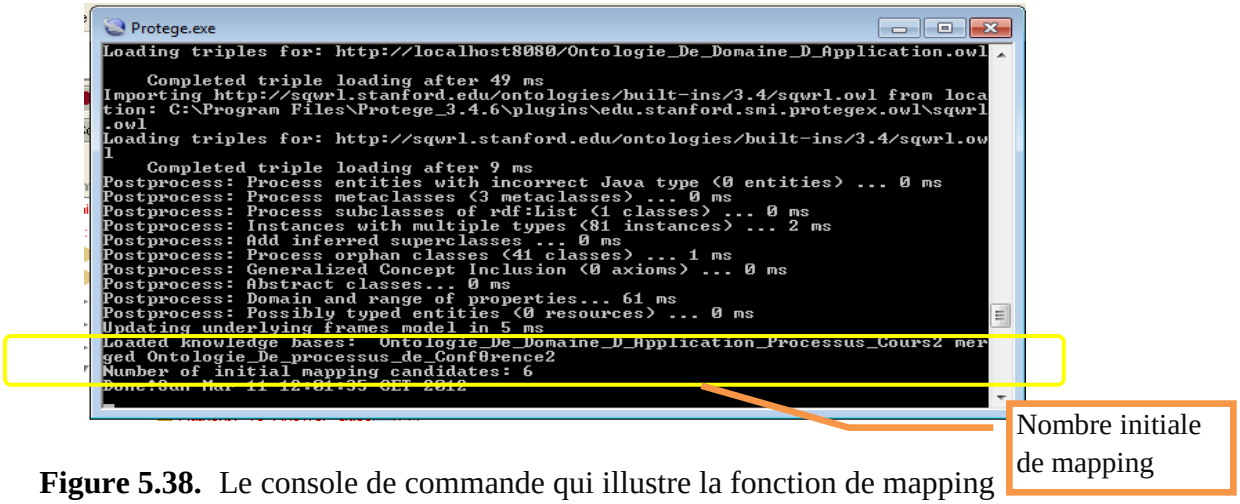

Figure 5.38. Le console de commande qui illustre la fonction de mapping

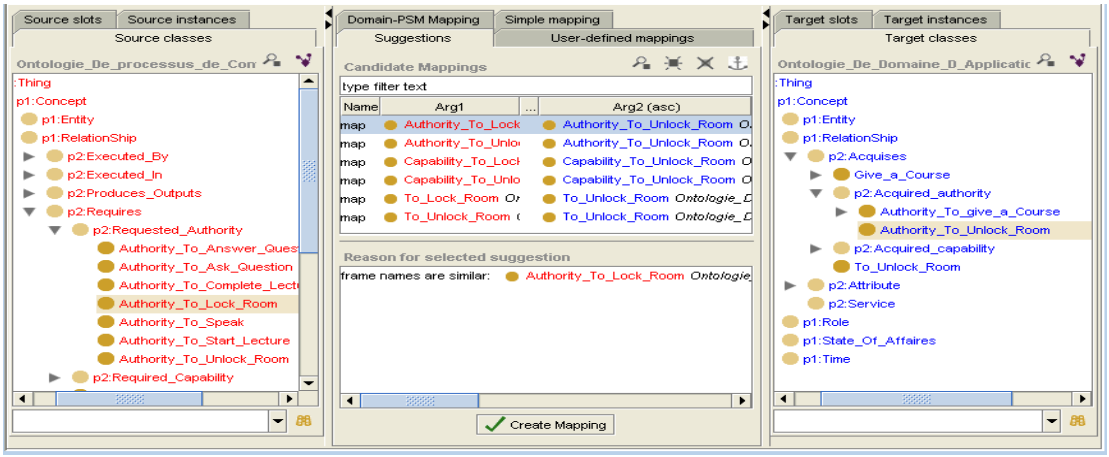

Figure 5.39. Les correspondances proposées

Si nous voulons faire notre propre matching on choisi l'onglet « *User defined mappings* ».

Pour faciliter la manipulation de cette fonction, il est possible de choisir une autre interface, nommé "CogZ", à partir de la liste Prompt de la barre des menus puis dans le menu secondaire Perspectives nous choisissons l'interface CogZ.

Nous avons donc l'interface présentée dans la Figure 5.40, avec les matchings proposées par l'algorithme de mapping déjà choisi :

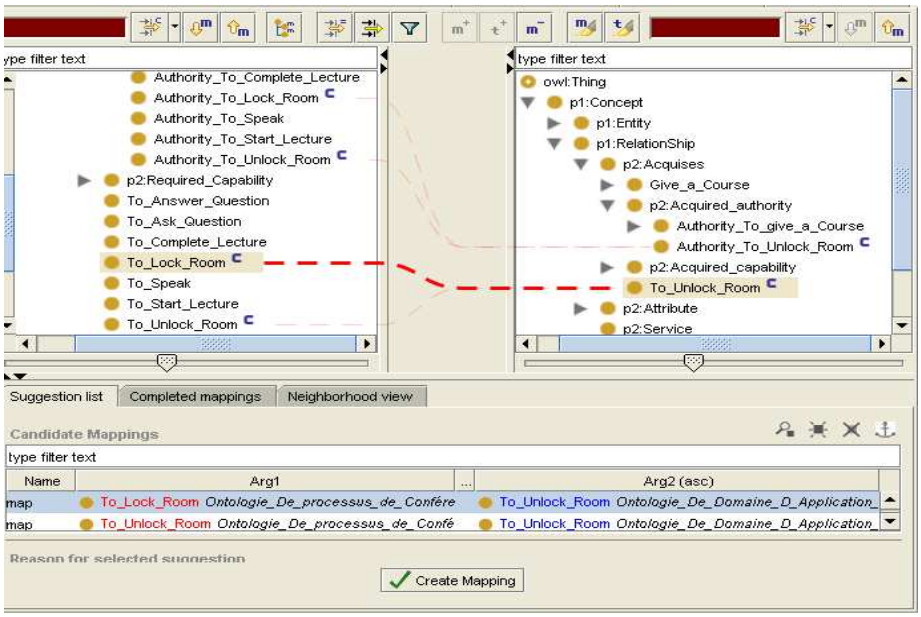

Figure 5.40. L'interface "CogZ" avec les matchings proposées

Si une proposition n'est pas satisfaisante au domaine d'application, la fonction Map nous permet de la supprimer. Donc, pour supprimer une proposition, nous sélectionnons le mapping à supprimer puis nous appuyons sur le bouton (m-). Nous aurons une boite de dialogue pour confirmer la suppression.

Si nous voulons ajouter des nouvelles correspondances, cette fonction nous permet aussi de les créer (correspondances des classes, des instances ou des slots (relations)). Donc, pour ajouter d'autres mappings, nous sélectionnons les concepts concernés des deux cotés puis nous appuyons sur le bouton (m+). Nous aurons une boite de dialogue (Figure 5.41), pour créer ce mapping, nous appuyons sur le bouton « *Create mapping* ».

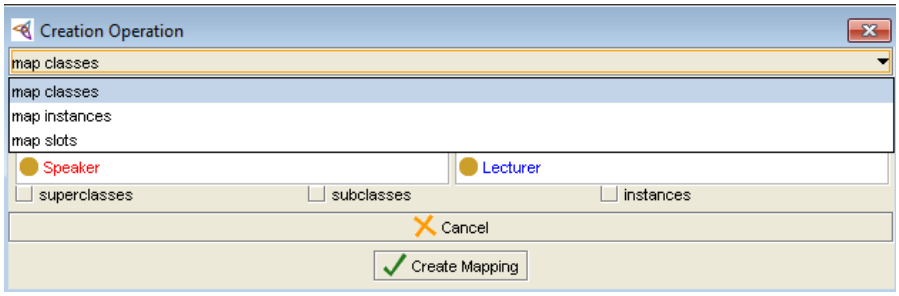

Figure 5.41. Boite de dialogue pour créer un mapping

Même chose pour les autres mapping jusqu'à ce que tous les mapping possibles soient réalisés, une partie de mapping finale est présentée dans la Figure 5.42.

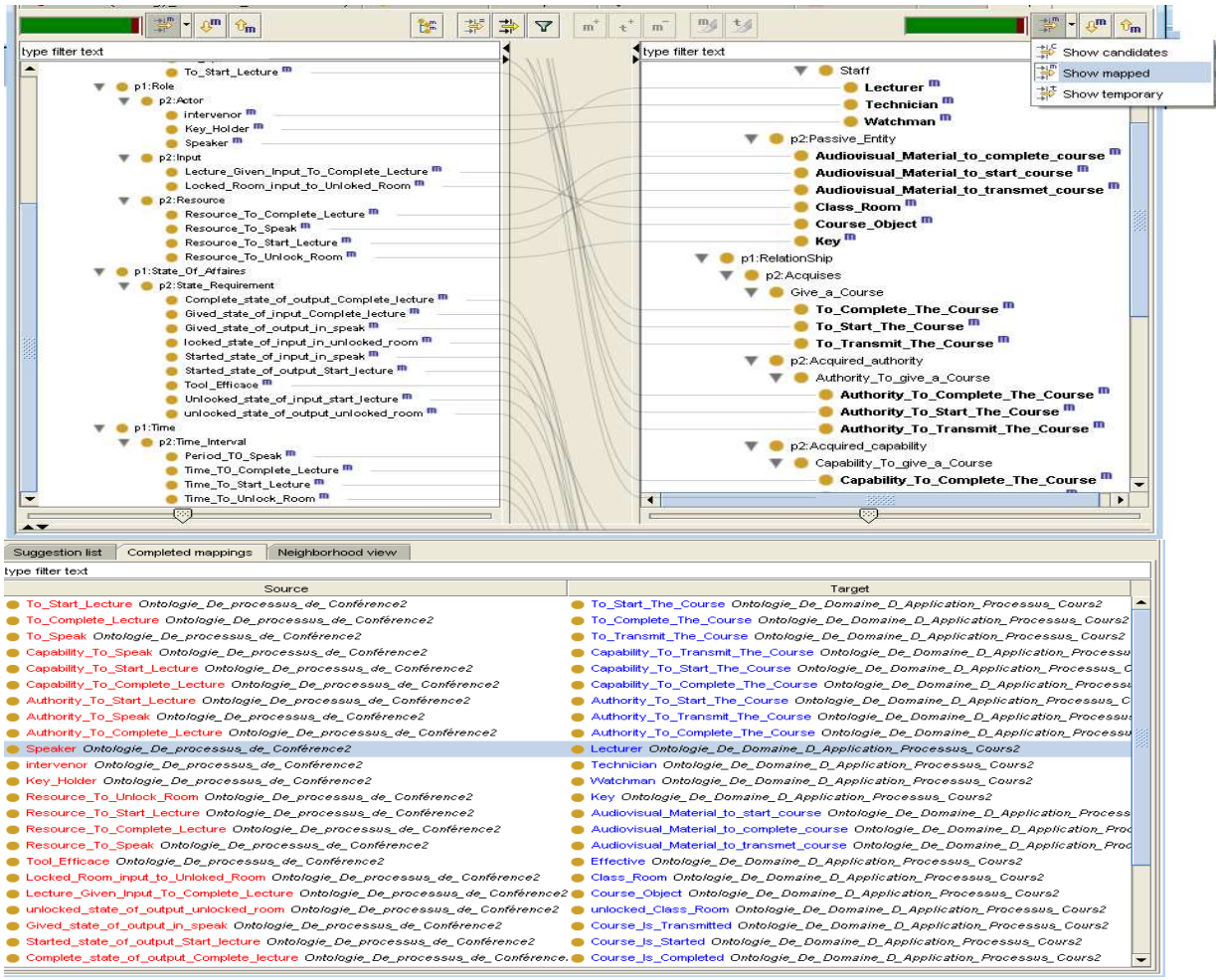

Figure 5.42. Les mappings possibles entre les deux ontologies

Par la suite, pour que ce mapping soit concret (la création des fichiers de mapping), nous devons exécuter la fonction de mapping dans le menu *CogZ* avec l'instruction « *Perform all exact name matches* ». Une boite de dialogue apparait pour confirmer ce mapping.

Selon les paramètres de l'algorithme de mapping déjà choisi, nous avons choisi deux types de sauvegarde des fichiers de mapping, donc nous avons comme résultats :

1. Une sauvegarde de mapping selon l'ontologie de mapping simple, nous avons trois types de fichiers qui sont créés:

- *Ontologie_De_Processus_Conference2-Mappings-Simple.Pprj :* qui stocke les informations des formes.
- *Ontologie_De_Processus_Conference2-Mappings-Simple.Pont :* qui stocke les classes de l'ontologie.
- *Ontologie_De_Processus_Conference2-Mappings-Simple.Pins :* qui stocke les instances de l'ontologie.

2. Une sauvegarde de mapping selon l'ontologie du mapping domaine PSM, nous avons trois types de fichiers qui sont créés:

- *Ontologie_De_Processus_Conference2-Mappings-dpsm.Pprj* qui stocke les informations des formes.
- *Ontologie_De_Processus_Conference2 -Mappings- dpsm.Pont :* qui stocke les classes de l'ontologie.
- *Ontologie_De_Processus_Conference2-Mappings-dpsm.Pins :* qui stocke les instances de l'ontologie.

Un extrait de chaque fichiers (*dpsm*) est figuré dans **l'annexe B.**

- Les deux fichiers « *Ontologie_De_Processus_Conference2-Mappings-Simple.Pprj* » et « *Ontologie_De_Processus_Conference2-Mappings-dpsm.Pprj* » permettent d'exploiter ce mapping, par exemple nous pouvons ajouter d'autre mapping ou en supprimer d'autres (Figure 5.43).

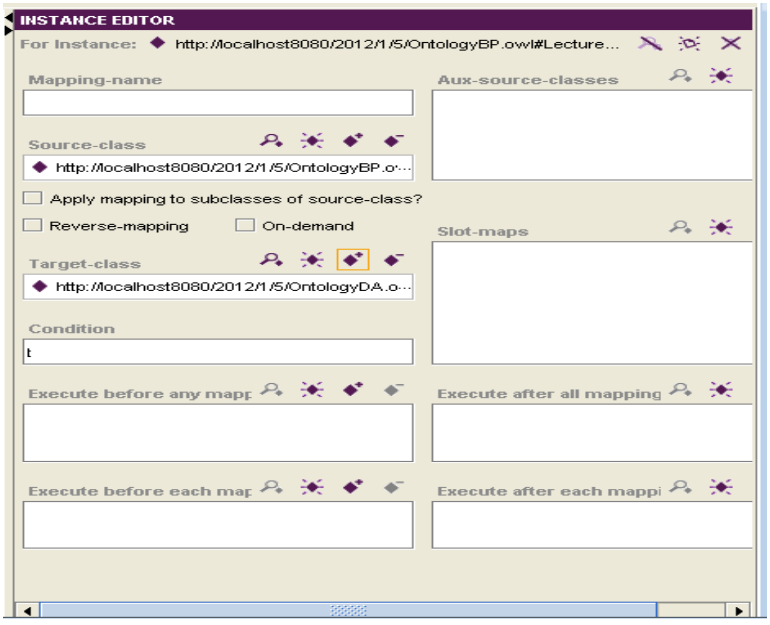

Figure 5.43. Le fichier *Ontologie_De_Processus_Conference2-Mappings-dpsm.Pprj* pour ajouter d'autres mappings

Dans le menu CogZ, nous choisissons le menu secondaire Raports, pour générer les rapports de mapping: un rapport sur l'état de mapping est présenté dans le Tableau 5.3, les autres rapports sur l'hiérarchie source de mapping et l'hiérarchie destinataire sont figurés dans l'**annexe C**.

Mapping Status Report

Generation Date: 2012-01-31 19:01:34

Generated By: Moufida

Table 1: Verified mappings

SOURCE TERM	TARGET TERM	REASON
Capability_To_Start_Lecture	Capability_To_Start_The_Course	**Capability_To_Start_Lecture** mapped to **Capability_To_Start_The_Course**
Capability_To_Complete_Lecture	Capability_To_Complete_The_Course	**Capability_To_Complete_Lecture** mapped to **Capability_To_Complete_The_Course**
Authority_To_Start_Lecture	Authority_To_Start_The_Course	**Authority_To_Start_Lecture** mapped to **Authority_To_Start_The_Course**
Authority_To_Speak	Authority_To_Transmit_The_Course	**Authority_To_Speak** mapped to **Authority_To_Transmit_The_Course**
Authority_To_Complete_Lecture	Authority_To_Complete_The_Course	**Authority_To_Complete_Lecture** mapped to **Authority_To_Complete_The_Course**
Speaker	Lecturer	**Speaker** mapped to **Lecturer**
intervenor	Technician	**intervenor** mapped to **Technician**
Key_Holder	Watchman	**Key_Holder** mapped to **Watchman**
Resource_To_Unlock_Room	Key	**Resource_To_Unlock_Room** mapped to **Key**
Resource_To_Start_Lecture	Audiovisual_Material_to_start_course	**Resource_To_Start_Lecture** mapped to **Audiovisual_Material_to_start_course**
Resource_To_Complete_Lecture	Audiovisual_Material_to_complete_course	**Resource_To_Complete_Lecture** mapped to **Audiovisual_Material_to_complete_course**
Resource_To_Speak	Audiovisual_Material_to_transmet_course	**Resource_To_Speak** mapped to **Audiovisual_Material_to_transmet_course**
Tool_Efficace	Effective	**Tool_Efficace** mapped to **Effective**
Locked_Room_input_to_Unloked_Room	Class_Room	**Locked_Room_input_to_Unloked_Room** mapped to **Class_Room**
Lecture_Given_Input_To_Complete_Lecture	Course_Object	**Lecture_Given_Input_To_Complete_Lecture** mapped to **Course_Object**
unlocked_state_of_output_unlocked_room	unlocked_Class_Room	**unlocked_state_of_output_unlocked_room** mapped to **unlocked_Class_Room**
Gived_state_of_output_in_speak	Course_Is_Transmitted	**Gived_state_of_output_in_speak** mapped to **Course_Is_Transmitted**
Started_state_of_output_Start_lecture	Course_Is_Started	**Started_state_of_output_Start_lecture** mapped to **Course_Is_Started**
Complete_state_of_output_Complete_lecture	Course_Is_Completed	**Complete_state_of_output_Complete_lecture** mapped to **Course_Is_Completed**
Started_state_of_input_in_speak	Course_Is_Not_Transmitted	**Started_state_of_input_in_speak** mapped to **Course_Is_Not_Transmitted**
Unlocked_state_of_input_start_lecture	Course_is_Not_Started	**Unlocked_state_of_input_start_lecture** mapped to **Course_is_Not_Started**
Gived_state_of_input_Complete_lecture	Course_is_not_Completed	**Gived_state_of_input_Complete_lecture** mapped to **Course_is_not_Completed**
locked_state_of_input_in_unlocked_room	Class_Room_Is_Locked	**locked_state_of_input_in_unlocked_room** mapped to **Class_Room_Is_Locked**
Period_TO_Speak	Period_To_Transmit_The_Course	**Period_TO_Speak** mapped to **Period_To_Transmit_The_Course**
Time_TO_Complete_Lecture	Time_To_Complete_The_Course	**Time_TO_Complete_Lecture** mapped to **Time_To_Complete_The_Course**

Time_To_Start_Lecture	Time_To_Start_The_Course	**Time_To_Start_Lecture** mapped to **Time_To_Start_The_Course**
Time_To_Unlock_Room	Time_To_Unlock_Class_Room	**Time_To_Unlock_Room** mapped to **Time_To_Unlock_Class_Room**
To_Unlock_Room	To_Unlock_Room	**To_Unlock_Room** mapped to **To_Unlock_Room**
Authority_To_Unlock_Room	Authority_To_Unlock_Room	**Authority_To_Unlock_Room** mapped to **Authority_To_Unlock_Room**
Capability_To_Unlock_Room	Capability_To_Unlock_Room	**Capability_To_Unlock_Room** mapped to **Capability_To_Unlock_Room**
To_Start_Lecture	To_Start_The_Course	**To_Start_Lecture** mapped to **To_Start_The_Course**
To_Speak	To_Transmit_The_Course	**To_Speak** mapped to **To_Transmit_The_Course**
Capability_To_Speak	Capability_To_Transmit_The_Course	**Capability_To_Speak** mapped to **Capability_To_Transmit_The_Course**
To_Complete_Lecture	To_Complete_The_Course	**To_Complete_Lecture** mapped to **To_Complete_The_Course**

Tableau 5.3 Rapport sur l'état de mapping

Le résultat de cette phase est un modèle du processus conférence localisé dans un domaine d'application choisi (devient un processus de cours) et peut avoir des points de variation qui seront résolus durant l'exécution de ce processus, nous les appelons des points de variation dynamiques. La définition de ces points de variation peut être faite durant l'implémentation de ce processus, par un langage dédié.

3.4. La définition du flux de contrôle

Après avoir attribué les entités aux rôles qui conviennent, nous avons un ensembles d'activités avec les différentes entités contribuent dans ces activités, l'exécution de ces dernières restent encore reliée à l'environnement d'exécution. C'est-à-dire, qu'il peut y en avoir d'autres points de variation qui ne sont pas résolus , nous les avons appelés points de variation dynamiques, ou bien nous pouvons (durant la phase d'ingénierie applicative du processus) en créer d'autres selon l'environnement d'exécution de ce processus.

Dans cette étape nous devons définir l'ordre d'exécution des activités de ce processus, de ce fait, nous ajouterons les connaissances de contrôle qui définissent l'ordre d'exécution de ces activités, nous prenons en compte leur environnement d'exécution. Nous pouvons utiliser un langage d'exécution des processus tel que WSBPEL.

Nous prenons par exemple le point de variation, illustré par le modèle des caractéristiques du processus de cours proposé dans la section 2.3. Nous pouvons présenter ce modèle par le diagramme d'activités illustré dans la Figure 5.44 (Décrite en

utilisant UML). Donc, nous pouvons donner un cours que ce soit Localement ou bien à distance.

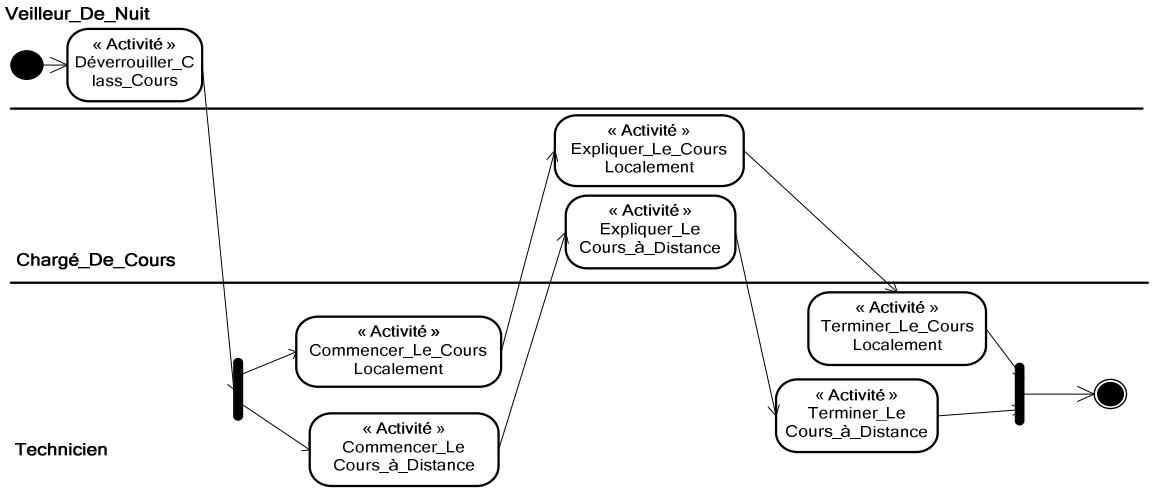

Figure 5.44. La définition de l'ordre d'exécution d'activités de processus cours (avec des points de variation dynamiques)

Comme nous l'avons déjà souligné, nous avons le point de variation entre les deux activités génériques *Donner_Un_cours_Localement et Donner_Un_cours_À_Distance*, qui va être résolu durant l'exécution de ce processus. Le choix sera fait selon la disponibilité des protocole ou services pour faire un cours à distance, ou bien celles qui nous permettent de le faire localement. C'est par exemlpe nous n'avons pas les protolcoles qui nous permettent de faire la connexion à distance, on sera obligé de choisir l'exécution de l'activité générique *Donner_Un_Cours_Localement* ou lieu de *l'activité générique Donner_Un_cours_À_distance.*

Tant que la résolution de ce point de variation sera durant l'exécution de ce processus, on a la possibilité de programmer ce choix durant la construction du processus (par un langage dédié tel que WSBPEL) et bien sûr exécuter ce processus dans un système de gestion des workflows permet ce choix.

3.5. La description du processus d'affaires exécutable (la construction du modèle exécutable de processus cours)

Nous utilisons un langage tel que WS BPEL [BPEL07] pour décrire le processus de cours exécutable, qui va être exécuté par un système de gestion de Workfows. Ce système illustre l'exécution des activités de processus cours. Le choix des variantes dynamiques cité précédemment est assuré par ce langage de programmation, qui décrit ce processus, ainsi que le système de gestion de Workflow qui exécute ce processus.

La définition d'un processus BPEL décrit des liens et la logique du processus BPEL entre les différents partenaires. Le langage BPEL supporte deux types d'activités :

- *Les activités simples* ou primitives ne comportant pas d'autres activités comparables aux tâches décrites dans les processus métiers avec leur description.

Le langage BPEL défini huit éléments de description d'activités simples :

- L'élément *receive* de réception d'un message.
- L'élément *reply* de renvoi d'une réponse au partenaire.
- L'élément *invoke* d'invocation d'une opération d'un partenaire.
- L'élément *assign* d'affectation des variables.
- L'élément *throw* de déclenchement d'une exception.
- L'élément *wait* d'attente d'un délai.
- L'élément *empty* sans activité déterminée.
- L'élément *terminate* d'arrêt de l'exécution d'un processus BPEL.

- *Les activités structurées* d'ordonnancement de l'exécution d'un ensemble d'activités simples ou structurées comme les structures de contrôle, les flux de données, la gestion des exceptions, la gestion des événements externes et la coordination de l'échange de messages entre les instances de processus BPEL.

Le langage BPEL ne propose pas d'éléments génériques de définition d'une activité. Il définit un ensemble d'éléments déterminés exprimant des comportements spécifiques. Ces éléments d'activités partagent un ensemble d'attributs standards optionnels :

- L'attribut *name* est le nom donné à l'activité.
- L'attribut *joinCondition* est une expression binaire de spécification des conditions de jointures des chemins d'exécution concurrents sur cette activité.

Il est nécessaire d'ajouter certains éléments de gestion des signaux de synchronisation durant l'exécution simultanée de plusieurs activités du processus BPEL. L'élément <*link*> représente le lien que traverse le signal de synchronisation entre deux activités. Chaque activité est marquée dans sa relation de synchronisation en spécifiant son lien d'origine ou de destination :

- L'élément source <*source*> indique l'émission d'un signal de synchronisation vers une autre activité.
- L'élément cible <*target*> identifie l'activité comme la destination d'un signal de synchronisation.

Le langage BPEL définit sept éléments de structuration d'activités :

- L'élément *sequence* d'exécution sérialisé d'activités.
- L'élément *switch* de sélection d'une activité alternative.
- L'élément *while* de définition d'un ensemble d'activités répétées.
- L'élément *pick* de contrôle d'arrivée de messages.
- L'élément *flow* d'exécution d'activités simultanées.
- L'élément *scope* de définition d'un nouveau périmètre d'utilisation de variables dans un sous groupe d'activités.
- L'élément *compensate* de définition d'un périmètre de compensation.

Le code source WSBPEL est exécuté dans un système de gestion de processus métiers BPMS. Nous pouvons aussi employer une technologie adéquate supportant la traduction automatique en WSBPEL et son exécution.

3.6. L'exécution du processus cours

Cette phase à comme but de gérer l'exécution de processus de Cours définit précédemment par un langage tel que WSBEL. Dans cette étape, il faut prendre les décisions qui correspondent au choix d'activités à exécuter, comme elles sont définies dans les phases précédentes et selon la disponibilité des services exigés pour l'exécution de chaque activité. Ces services peuvent être fournis par d'autres activités de processus lui-même ou disponible sur l'environnement d'exécution. Dans notre cas, durant l'exécution nous devons vérifier la diponibilité des services qui nous permettent la connextion à distance, pour faire ou choisir de faire un cours à disance. En cas d'absence nous allons choisir automatiquement de faire un cours localement. Les éléments définit ci-dessus sont suffisants pour faire ce choix.

Donc, comme nous l'avons déjà cité, le code source WSBPEL de ce processus sera exécuté dans un système de gestion de processus métiers BPMS tels que BONITA, Intalio|BPMS, de ce fait :

- Ce système orchestre l'exécution des activités de processus d'affaires et à certains moments (comme défini dans le modèle de processus d'affaires) demande des services fournis par les entités actives appropriées.
- Les capacités et les autorités fournies par les systèmes d'application sont demandées à travers des interfaces web services.
- Les fournisseurs des services humains sont demandés à travers des interfaces utilisateurs spéciaux. Ils sont informés d'activités suspendus qui doivent être exécutées.

4. Conclusion

Dans ce chapitre, nous avons appliqué l'approche proposée dans le Chapitre 04, la réutilisation des connaissances ontologiques dans le processus d'affaires, sur le processus générique de Conférence.

Premièrement, nous avons donné une description ou une conceptualisation détaillée de l'application de cette approche sur le processus de conférence, qui est considéré comme un processus d'affaires générique. Cela durant les trois phases proposées : l'ingénierie du domaine de processus, l'ingénierie d'applicative du processus ainsi que la phase de l'exécution de processus. Nous avons décrit principalement les points suivants :

- ✓ Nous avons construit un modèle des caractéristiques du processus de conférence.
- ✓ Nous avons défini la logique du déroulement des activités de processus Conférence.
- ✓ Nous avons exposé une conceptualisation de toutes les activités de processus conférence (Donner conférence, Déverrouiller la salle, Verrouiller la salle, Poser la question, Répondre à la question, Commencer La conférence et Fermer La conférence), sous forme d'ontologies du processus à partir de méta-concepts *«Ontologie du processus d'affaires de haut niveau» mentionné* dans le Chapitre 4.
- ✓ Nous avons aussi défini une ontologie, à partir de méta-concepts « *Ontologie de domaine d'application de haut niveau »* mentionné dans le chapitre 4, elle représente l'*Ontologie du domaine d'application du processus de cours* dans une université particulière.
- ✓ Pour résoudre le problème de la configuration de processus, nous avons suivi l'approche à base de la Logique de Description (DL) [Baade03] décrite dans [ČIUK07a], parce que nous avons défini toutes les ontologies proposées utilisant l'Ontology Web Langage (OWL) à base de la Logique de Description (DL).
- ✓ Nous avons remarqué que pendant cette étape, certains points de variation, peuvent être restés non résolus, la résolution ou la prise de décisions sur ces points de variations sera durant la phase de l'exécution de processus.
- ✓ Le processus de conférence configuré et l'ontologie du domaine d'application (correspondant au processus cours) sont des entrées pour la prochaine étape, qui est l'attribution des rôles.

- ✓ L'attribution des rôles est réalisée en correspondant les requises aux acquises, selon l'algorithme qui était proposée dans [ČIUK07a] et que nous avons enrichi par les concepts qu'nous avons ajoutés.
- ✓ Nous avons ajouté des entités à l'ontologie de domaine d'application selon l'algorithme d'attribution des rôles, dans ce cas la nous avons arrivé à construire l'ontologie du domaine d'application finale de processus qui correspond à l'ontologie du processus de conférence configurée.
- ✓ Nous avons défini l'ordre d'exécution des activités de ce processus, nous avons donné un diagramme qui représente cet ordre.
- ✓ À la fin de cette description et durant la phase de l'exécution du processus cours, nous avons donné un modèle des caractéristiques qui présente des points de variation dynamiques qui peuvent être résolus durant l'exécution du processus.

Deuxièmement, nous avons donné un aperçu schématisé sur l'aspect d'implémentation de cette approche :

- ✓ Nous avons utilisé l'outil Protégé pour implémenter toutes les ontologies déjà mentionnées dans cette approche, avec les différentes relations, restreintes et règles de dépendances entre les différents concepts. Le raisonneur *Racer* a permis de tester la cohérence et la consistance de ces ontologies.
- ✓ L'outil Protégé avec le raisonneur *Racer*, nous a permis aussi d'utiliser la logique de description pour résoudre le problème de la configuration de processus. Cela est réalisé en utilisant les différentes relations, restreintes et règles de dépendances entre les différents concepts, notant que le plugin *SWRL RULES* nous a permis de définir ces règles de dépendances.
- ✓ Pour l'attribution des rôles, nous avons utilisé la fonction *Map* de plugin *Prompt* de l'outil Protégé.
- ✓ L'implémentation de modèle exécutable de processus cours est possible via un langage d'exécution des processus tel que WSBPEL, ce dernier doit être exécuté par un système de gestion des Workflow tel que Intalio|BPMS ou BONITA.

Dans ce qui suit nous présentons une conclusion générale de notre travail ainsi que les perspectives attendues pour des travaux future.

Conclusion Générale et Perspectives

“Learn from yesterday, live for today, hope for tomorrow.
The important thing is not to stop questioning.”
(Albert Einstein)

La plupart des techniques et méthodes utilisées durant le développement des différents systèmes d’information sont réutilisées. Ainsi, ces systèmes ont besoin de techniques de réutilisation pour remédier à la complexité des systèmes économiques et stratégiques. De l’autre côté, les enjeux du processus d’affaires occupent une partie importante dans les méthodologies avancées de l’ingénierie d’entreprise. Tel que la représentation des connaissances sur les familles des processus d'affaires similaires et leur réutilisation dans des projets de l’ingénierie d'entreprise. Dans ce cas, nous avons constaté que les travaux de Caplinskas et ses coéquipiers sont très intéressants dans la mesure où ils ont proposé une approche à base d’ontologies qui permet la réutilisation des connaissances au niveau du processus d’affaires.

La réutilisation présentée dans leur approche permet la résolution de variabilité d’une manière statique c’est à dire durant la construction du processus d’affaires localisé dans un domaine d’application choisi, sans tenir en compte de la variabilité qui peut avoir lieu durant l’exécution de ce processus.

Pour cette raison, dans ce mémoire nous avons proposé une nouvelle approche, semi automatique, qui se base sur les ontologies pour permettre la réutilisation des connaissances du processus d’affaires durant l’ingénierie de domaine d’application puis dans l’environnement d’exécution approprié. C'est-à-dire nous avons distingué deux types de points de variation, statiques et dynamiques. Les points de variation statiques sont résolus durant la construction du processus dans le domaine d’application choisi et les autres points de variations dits dynamiques qui sont résolus durant l’exécution de ce processus dans l’environnement d’exécution choisi aussi. Cela en tenant compte à la variabilité qui peut avoir lieu durant l’exécution de ce processus.

Donc, notre contribution majeure est la proposition d’une conceptualisation de variabilité qui supporte ces deux types de points de variation ainsi que l’algorithme de la résolution de cette variabilité. Par conséquence, l’architecture de l'approche que nous avons proposée comporte trois phases : la phase d’ingénierie du domaine de processus (A), la phase d’ingénierie applicative du processus (B) et la phase d’exécution du processus (C).

Pour soutenir et consolider l'architecture déjà citée, nous avons proposé d'enrichir deux types d'ontologies réutilisables de haut niveau : Celle qui représente les connaissances d'un processus d'affaires générique pendant la phase d'ingénierie du domaine (nous avons mis beaucoup plus l'accent sur la conceptualisation de variabilité) et l'autre ontologie qui a été désignée à la représentation des connaissances d'un domaine d'application pendant la phase d'ingénierie applicative du processus. Les deux ontologies ont été basées sur un système de méta-concepts, qui constitue une ontologie de haut niveau, que nous avons également traité.

Les principaux concepts que nous avons ajoutés sont : le concept de Temps (il est important surtout durant l'exécution du processus d'affaires) qui est considéré, dans l'ontologie de haut niveau comme instance de concept ¨Concept¨, et qui avait une relation entre le concept Relation. Puisque les deux autres ontologies sont considérées comme instanciations de l'ontologie de haut niveau, le concept de temps est apparu dans les deux ontologies sous le nom d'Intervalle de Temps. Il avait une relation avec l'entité Activité dans l'ontologie de processus et deux relations dans l'ontologie de domaine d'application, l'une avec la relation Acquises et l'autre avec l'état d'affaires État.

Nous avons distingué deux types de points de variation : statiques et dynamiques. Les points de variation statiques sont résolus durant la configuration de processus d'affaires générique lors de phase (B), par contre les points de variation dynamiques sont résolus lors de la phase (C). Nous avons aussi distingué deux états pour un point de variation, fermé ou ouvert : Le point de variation est « ouvert » dans ce cas, il est possible pendant l'ingénierie applicative d'ajouter une nouvelle variante ou de modifier des variantes existantes. Le point de variation est « fermé» dans ce cas, il n'est pas possible de sortir des choix proposés par le point de variation.

Par conséquent, nous avons deux types de configuration, celui de domaine applicative du processus et l'autre durant la phase d'exécution du ce processus, donc notre contribution dans ces deux types de configuration était comme suit :

- Le configurateur devrait être en mesure de distinguer les points de variation statiques et dynamiques et de résoudre les variabilités dont les points de variation sont statiques dans la phase de l'ingénierie applicative du processus, et celles dont les points de variation sont dynamiques dans la phase d'exécution du ce processus (c'est-à-dire choisir une variante pour chaque variabilité selon les contraintes).
- Le configurateur devrait être en mesure aussi d'intégrer les variantes choisies dans le processus en utilisant les mécanismes de l'implémentation appropriés à la variabilité.

De ce fait, une description du processus configuré (c'est-à-dire sans variabilités statiques dans la première configuration de l'approche proposée ou sans variabilités dynamiques dans la deuxième configuration) est entièrement produite et enregistrée dans le Profil de Dérivation.

Pour bien assimiler l'approche proposée, nous l'avons appliquée sur le processus générique de Conférence. Tout d'abord, nous avons donné une description ou une conceptualisation détaillée de l'application de cette approche sur ce processus. Ensuite, nous avons donné un aperçu schématisé sur l'aspect d'implémentation de cette approche :

- Nous avons utilisé l'outil Protégé pour implémenter toutes les ontologies déjà mentionnées dans cette approche, avec les différentes relations, restreintes et règles de dépendances entre les différents concepts. Le raisonneur *Racer* a permis de tester la cohérence et la consistance de ces ontologies.
- L'outil Protégé avec le raisonneur *Racer*, nous a permis aussi d'utiliser la logique de description pour résoudre le problème de la configuration de processus. Cela est réalisé en utilisant les différentes relations, restreintes et règles de dépendances entre les différents concepts, notant que le plugin *SWRL RULES* nous a permis de définir ces règles de dépendances.
- Pour l'attribution des rôles, nous avons utilisé la fonction *Map* de plugin *Prompt* de l'outil *Protégé*.
- L'implémentation de modèle exécutable de processus cours est possible via un langage d'exécution des processus tel que WSBPEL, ce dernier doit être exécuté par un système de gestion des Workflow tel que Intalio|BPMS ou BONITA.

Ce qui distingue cette approche des autres (par exemple les approches à base de Enterprise Resource Planning (ERP)) est que les processus d'affaires sont adoptés pour les domaines d'application, et non pas inversement.

Perspectives

De fait que les domaines divers utilisés dans ce mémoire. Principalement l'ingénierie d'entreprise, l'ingénierie de domaine, ingénierie des connaissances, l'ingénierie des systèmes basés sur les ontologies. Ainsi que la diversité des problèmes envisagés dans cette approche, nous décrivons dans cette section de nombreuses perspectives sur les aspects ci-dessous :

- *La construction des différentes ontologies :* Tout d'abord, appliquer cette approche sur le terrain avec des données réelles, ce que va améliorer les ontologies proposées.
- *La configuration :* améliorer la configuration selon le besoin des ontologies proposées.
- *L'attribution des rôles :* proposer et implémenter une fonction propre qui permet l'attribution des rôles d'une façon efficace et optimale.
- *Automatiser* toutes les étapes de l'approche proposée qui ne sont pas déjà automatisées c'est-à-dire :
    - *La définition de l'ordre d'exécution des activités de processus et la description du processus par un langage dédié tel que WS BPEL.*
    - *L'exécution de ce processus par un système de gestion des Workflows,* si les systèmes de gestion de workflows existent, ne prend pas en considération nous besoin dans cette approche, nous pouvons créer un outil qui permet surtout la résolution des points de variation dynamiques durant l'exécution du processus d'affaires. Nous prendrons en compte leur environnement d'exécution.

Références

[AALS02] Aalst.Van der and Kees Max van Hee, Workflow Management: Models, Methods, and Systems, Wil, The MIT Press Cambridge, Massachusetts London, England, 2002.

[ABOU10] Mohammed Faical Abouzaid, Analyse Formelle D'orchestrations De Services Web, Thèse Présentée En Vue De L'obtention Du Diplôme De Philosophie Doctor (Génie Informatique), Département De Génie Informatique Et Génie Logiciel École Polytechnique De Montréal, Décembre 2010.

[ADRI07] Adrien Coulet, Ontologies et alignement d'ontologies: notes de cours, (inspiré des cours de R. Dieng, J. Euzenat, A. U6mez-Pérez, J. Lieber et A. Napoli), dernière version: 19/11/07.

[BAAD91] F. Baader and P. Hanschke. A schema for integrating concrete domains into concept languages. In Proc. Of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI'91), pages 452-457, 1991.

[BAAD03] Baader, F., D. Calvanese, D.L. Mcguinness, D. Nardi and P.F. Patel-Schneider.The description logic handbook ; Theory, implementation, and applications. Cam- bridge University Press. 574pp, 2003.

[BECK03] Becker, M. Towards a General Model of Variability in Product Families. In J. van Gurp, J. Bosch (Eds.), Proceedings of the 1st Workshop on Software Variability Management, Groningen, The Netherlands. pp. 19–27, 2003.

[BORS97] Borst, W. N. Construction of Engineering Ontologies. Center for Telematica and Information Technology, University of Tweenty, Enschede, NL, 1997.

[BPDM08] Business Process Definition MetaModel (BPDM) (2008). Object Management Group, Inc. Access through Internet: http://www.omg.org/cgi-bin/ doc?bmi/ , 2008.

[BPEL07] Web Services Business Process Execution Language v2.0 (2007). OASIS standard. Access through Internet: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-S.pdf, 2007.

[BRAC79] Brachman, R. J. On the Epistemological Status of Semantic Networks. In N.V. Findler (ed.) Associative Networks: Representation and Use of Knowledge by Computers. Academic Press: 3-50, 1979.

[BRIO08] Briol Patrice, Ingénierie des processus métiers, de l'élaboration à l'exploitation, ISBN 978-1-4092-0040-6, 2008.

[CAPL98] Caplinskas A, Vasilecas 0, Registers and Register - Based Information Systems. Proc of the Third International Baltic Workshop "BalticDB&IS'98". Riga Latvia April 15-17 1998, 1998.

[CAPL02] Caplinskas, A., A. Lupeikiene, O. Vasilecas. Unified enterprise engineering environment: ontological point of view. In H.-M. Haav, A. Kalja (Eds.), Proceedings of the Baltic Conference, BalticDB&IS 2002, vol. 1. Institute of Cybernetics at Tallinn Technical University. pp. 30–50, 2002.

[CAPL03] Albertas CAPLINSKAS, Audron eLUPEIKIEN, E Olegas VASILECAS. The Role of ontologies in reusing domain and enterprise engineering Assets; INFORMATICA, 2003, Vol.14, No.4, 455–470 Institute of Mathematics and Informatics, Vilnius, 2003.

[CAPL03a] Caplinskas A, An Ontology-based Approach to Enterprise Engineering. Computer Science Reports, vol 14/ 03, pp 22-26, 2003.

[CAPL04] Caplinskas, A., and D. Ciuksys. Ontologies knowledge reuse and domain engineering techniques in information system engineering. In O. Vasilecas, A. Caplinskas, W. Wojtkowski, W.G. Wojtkowski, J. Zupancic, S. Wrycza (Eds.), Proceedings of the Thirteenth International Conference on Information Systems Development, Vol. 1. Technika, Vilnius. pp. 264–270, 2004.

[CHAA09] Mohamed Chaabani Mohamed Mezghiche Martin Strecker, Formalisation de la logique de description ALC dans l'assistant de preuve Coq, 2009.

[CHAN86] CHANDRASEKARAN, B. Generic tasks in knowledge-based reasoning: High-level building blocks for expert system design. IEEE Expert 1(3), p. 23–30, 1986.

[ČIUK06] Ciuksys, D., and A. Caplinskas. Modelling of reusable business processes: an ontology-based approach. In A.G. Nilsson, R. Gustas, W. Wojtkowski, W.G. Wojtkowski, S. Wrycza, and J. Zupancic (Eds.), Advances in Information Systems Development, Vol. 1. Springer. pp. 71–82, 2006.

[ČIUK07] Donatas Čiukšys, albertas Čaplinskas, ontology-based approach to reuse of business process knowledge, ISSN 392-056. INFORMACIJOS MOKSLAI, 2007.

[ČIUK07a] Donatas CIUKSYS, Albertas CAPLINSKAS, Reusing ontological knowledge about business processes in is engineering: process configuration problem, INFORMATICA, 2007 Vol. 99, No. 99, 1–18, 2007.

[CURT92] Curtis, B., Kellner, M.I. and Over, J.: "Process modeling", Communications ACM, 35, (9), pp. 75- 90, 1992.

[CZAR00] CzARNECKI, K.; EISENECKER, U. Generative Programming: Methods, Tools, and Applications. Addison-Wesley Professional, 2000.

[DAVI06] DAVIES, J.; STUDER, R.; WARREN, P. Semantic Web Technologies: Trends and Research in Ontology-based Systems. Wiley, 2006.

[DEMA97] DEMAR Claudine Workflow Management Systems**,** site web: http://perso.telecom-paristech.fr/~saglio/bdas/97/Demar.html **,** Mémos 97 consultables**.**

[ebXML03] Voir : http://www.ebxml.org, 2003.

[ELBY09] M Abdeltif Elbyed, ROMIE, une approche d'alignement d'ontologies à base d'instances, Thèse présentée pour l'obtention du grade de Docteur De L'institut National Des Télécommunications Dans Le Cadre De L'école Doctorale S&I En Co-accréditation avec L'universite D'EVRY-VAL D'ESSONNE. Soutenue le 16 Octobre 2009.

[ERL05] ERL, T. Service-Oriented Architecture: Concepts, Technology, and Design. Upper Saddle River: Prentice Hall PTR, 2005.

[FOX92] Fox, M.S. The TOVE project: a common-sense model of the enterprise, industrial and engineering applications of artificial intelligence and expert systems. In F. Belli and F.J. Radermacher (Eds.), Lecture Notes in Artificial Intelligence, 604. Springer–Verlag. pp. 25–34. 1992.

[FRAN01] Françoise TESSIER, Influence de l'approche processus de la norme ISO 9001 version 2000 sur l'organisation actuelle d'une unité de production, 19 mars 2001 ;

[FVHD04] Frank van Harmelen and Deborah L. McGuinness. OWL web ontology language overview. W3C recommendation, W3C, 10 February 2004. http://www.w3.org/TR/2004/REC-owl-features-20040210/

[GÓME99] Gómez-Pérez, A. Ontological Engineering: A state of the Art. Expert Update. British Computer Society. Autumn, 1999.

[GRUB93] Gruber, T. R. Toward principles for the design of ontologies used for knowledge sharing. In Formal Ontology in Conceptual Analysis and Knowledge Representation. Kluwer Academic Publishers, Deventer, 1993.

[GRUB93a] Gruber, T.. A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition, 5(2), 199-220, 1993.

[GRÜN00] Grüninger, M., K. Atefi, M.S. Fox.Ontologies to support process integration in enterprise engineering. Computational & Mathematical Organization Theory, 6, 381–394, 2000.

[GUAR95] Guarino, N, R. Poli (1995). Formal Ontology, Conceptual Analysis and Knowledge Representation. Special issue of the International Journal of Human and Computer Studies. 43(5/6):625-640. 1995.

[GUAR95a] Guarino, N., & Giaretta, P. Ontologies and Knowledge Bases. Towards a Terminological Clarification. In N. Mars (Ed.), Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing (pp. 25–32). Amsterdam: IOS Press, 1995.

[GUAR98] Guarino, N. Formal Ontology and Information Systems. In Proc. of Formal Ontology and Information Systems, Trento, Italy. IOS Press, 1998. http://www.loa-cnr.it/Papers/FOIS98.pdf.

[GURP01] Van Gurp, J., J. Bosch, M. Svahnberg. On the Notion of Variability in Software Product Lines. In R. Kazman, P. Kruchten, C. Verhoef, and H. van Vliet (Eds.), Proceedings of the Working IEEE/IFIP Conference on Software Architecture. IEEE Computer Society Press. pp. 45–54, 2001.

[HAFE04] Hafedh Milli, Guitta Bou Jaoude, Éric Lefebvre, Guy Tremblay, Alex Petrenko, Business Process Modeling Languages: Sorting Through the Alphabet Soup, Laboratoire de Recherche sur les Technologies du Commerce Électronique, Centre de Recherche Informatique de Montréal, Montréal, Canada, 2004.

[HENR93] Henry J. Johansson et al. Business Process Reengineering: BreakPoint Strategies for Market Dominance. John Wiley & Sons, 1993.

[HERN05] Nathalie Hernandez, Ontologies de domaine pour la modélisation du contexte en recherche d'information, Laboratoire IRIT – Pôle SIG-EVI, Thèse de doctorat Présentée devant L'Université Paul Sabatier de Toulouse, Soutenue le mardi 06 décembre 2005.

[HOLL99] Hollingsworth David, ICL A&TC, The Workflow Management Coalition Specification, Workflow Management Coalition, Terminology & Glossary, Document Number WFMC-TC-1011 Document Status - Issue 3.0 Feb 99,

[HORR04] Horrocks, I., and P.F. Patel-Schneider. A proposal for an owl rules language. In Proceedings of the 13th international conference on World Wide Web, ACM, NewYork. pp. 723–731, 2004.

[JAYC02] Jay Cousins and Tony Stewart, RivCom Ltd, What is Business Process Design and Why Should I Care? R I V C 0 M? 04/09/2002;

[JEAN08] Jean-Noël Gillot, La gestion des processus métiers : l'alignement des objectifs stratégiques de l'entreprise et du Système d'information par les processus, publier 2008 ;

[JENN00] Jennings, N. R., Norman, T. J., Faratin, P., Autonomous Agents For Business Process Management, Dept. Electronic Engineering, Queen Mary & Westfield College, University of London, London, UK P. O'BRIEN and B. ODGERS BT Research Labs, Ipswich, Suffolk, UK Applied Artificial Intelligence, 14 :145- 189, Copyright 2000 Taylor & Francis, 2000.

[JIAN10] JIANQI yu, ligne de produits dynamique pour les applications a services, Thèse préparée au sein du Laboratoire d'Informatique de Grenoble (LIG), soutenue le 16 JUIN 2010.

[JOHA97] Johannes Sametinger, Software Engineering with Reusable Components, Springer-Verlag, Berlin Heidelberg NewYork, London Paris Takyo, Hang kong Barcelona, Budapest, March. 03. 1997.

[KANG90] Kang, K.; Cohen, S.; Hess, J.; Novak, W.; Peterson, A. Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report CMU/SEI-90-TR-21, SEI. Carnegie Mellon University, Pittsburgh, Pennsylvania, 1990.

[KLEI01] Klein, M., Gomez-Pérez, A., Gruninger, M., & Stuckenschmidt, H. Combining and relating ontologies : an analysis of problems and solutions. Workshop on Ontologies and Information Sharing (IJCAI-01), August 4-5, 2001.

[LAME02] G. Lame, Construction d'ontologie à partir de texte, une ontologie du droit dédiée à la recherche d'information sur le Web, Thèse de doctorat, Ecole des Mines de Paris, 2002.

[MARI06] Marine Terezinha da Silva Bello Flores, Interfaces culturelles pour environnements d'apprentissage en ligne – Cas du système Virtuale du Centre Universitaire Feevale au Brésil , Septembre 2006.

[MEDI93] Medina Mora, R., T. Winograd, and R. Flores. Action work ow as the enterprise integration technology. Bulletin of the Technical Committee on Data Engineering. IEEE Computer Society 16(2), 1993.

[MICH93] Michael Hammer and James Champy. Reengineering the Corporation: A Manifesto for Business Revolution, Harper Business, 1993.

[MINS75] Minsky, M. A Framework for Representing Knowledge. In Edited by P. Winston. The Psychology of Computer Vision. New-York : Mc-Graw-Hill, p 211-281, 1975.

[MOF06] Meta Object Facility (MOF) Core Specification, version 2.0 (2006). Object Management Group, Inc, 2006. Access through Internet: http://www.omg.org/cgi-bin/doc?formal/[2006-01-01] .

[MOTI05] Motik B., Sattler U. and Studer R. Query Answering for OWL-DL with Rules. Journal of Web Semantics: Science, Services and Agents on the World Wide Web, 3(1), 41–60, 2005.

[NARD03] Nardi, D., Brachman, R.J.: An Introduction to Description Logics. In: The Description Logic Handbook. Baader et al. Cambridge University Press. Cambridge, England, 2003.

[OLIV03] Olivier Dameron, Modélisation, représentation et partage de connaissances anatomiques sur le cortex cérébral, Thèse Soutenue le 17 décembre 2003, Université de rennes I.

[OMG01] OMG, EDOC: UML Profile for Enterprise Distributed Object Computing, Document ptc/2001-12-04, December 2001.

[RAPH03] Raphaël SIBLER, L'approche Processus, Une méthode de lecture de l'organisation, Dossier du mois de janvier 2003, Créé le 28/12/02

[RUMM95] Rummler & Brache, Improving Performance: How to manage the white space on the organizational chart. Jossey-Bass, San Francisco, 1995.

[SCHM06] D. Schmidt, "Model-Driven Engineering", IEEE Computer, Vol.39, 2006.

[SMIT03] SMITH, H.; FINGAR, P. Business Process Management (BPM): The Third Wave. Meghan Kiffer Press, 2003.

[SOWA84] Sowa, J. Conceptual Structures: information processing in mind and machine. Reading: Addison Wesley publishing Company, 1984. 481 pages. (The System Programming Series), 1984.

[SPAR04] Sparx Systems UML Tutorials, THE BUSINESS PROCESS MODEL, 2004, www.sparxsystems.com/downloads/whitepapers/The_Business_Process_Model.pdf

[THOM93] Thomas Davenport, Process Innovation: Reengineering work through information technology. Harvard Business School Press, Boston, 1993.

[USCH96] Uschold, M., M. King, S. Moralee, Y. Zorgios. The enterprise ontology. Knowledge Engineering Review, 13(1), 31–90, 1996.

[USCH98] USCHOLD, M.; KING, M.; MORALEE, S.; zORGIOS, Y. The Enterprise Ontology. In The Knowledge Engineering Review, vol. 13, Special Issue on Putting Ontologies to Use, 1998.

[USCH99] Jasper, R.and Uschold, M. A framework for understanding and classifying ontology applications. IJCAI-99, Ontology Workshop, Stockholm, 1999.

[VOLK94] Volker Gruhn, Software Process Management and Business Process (Re-) Engineering, LIVRE “Software process technology: Third European Workshop EWSPT'94, Villard de Lans” Par Brian C. Warboys(Ed.). Fraunhofer Institute for Software and Systems Engineering, Baroper Strabe 301? 44227 Dortmund, Germany, 1994.

[WfMC02] Layna Fischer**,** An introduction to workflow, in The Workflow Handbook 2002, the Workflow Management Coalition (2002), ISBN 0-9703509-2-9, 2002.

[XAVI05] Xavier Lacot, Introduction à OWL, un langage XML d'ontologies Web. Juin 2005.

[XAVI09] Xavier Godefroy, Le BPM, CNAM cours NFE107 : Urbanisation et architecture des SI, Rapport sur le BPM, mai 2009.

ANNEXE A

UN EXTRAIT DU FICHIER : Ontologie_Haut_Niveau.Owl

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:xsp="http://www.owl-
ontologies.com/2005/08/07/xsp.owl#"
   xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"

xmlns="http://Localhost8080/OntologyHautNiveau.owl#
"
   xmlns:swrl="http://www.w3.org/2003/11/swrl#"

xmlns:protege="http://protege.stanford.edu/plugins/ow
l/protege#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
syntax-ns#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-
schema#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"

xml:base="http://localhost8080/Ontology_Haut_Niveau.
owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class
rdf:about="http://Localhost8080/OntologyHautNiveau.o
wl#Entity">
    <rdfs:subClassOf>
     <owl:Restriction>
      <owl:onProperty>
       <owl:ObjectProperty
rdf:about="http://Localhost8080/OntologyHautNiveau.o
wl#participates_In"/>
      </owl:onProperty>
      <owl:minCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#in
t"
      >1</owl:minCardinality>
     </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
     <owl:Restriction>
      <owl:onProperty>
       <owl:ObjectProperty
rdf:about="http://Localhost8080/OntologyHautNiveau.o
wl#playes"/>
      </owl:onProperty>
      <owl:minCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#in
t"
      >1</owl:minCardinality>
     </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
     <owl:Class
rdf:about="http://Localhost8080/OntologyHautNiveau.o
wl#Concept"/>
    </rdfs:subClassOf>
    <owl:disjointWith>
     <owl:Class
rdf:about="http://Localhost8080/OntologyHautNiveau.o
wl#Role"/>
    </owl:disjointWith>
    <owl:disjointWith>
     <owl:Class
rdf:about="http://Localhost8080/OntologyHautNiveau.o
wl#State_Of_Affaires"/>
    </owl:disjointWith>
    <owl:disjointWith>
     <owl:Class
rdf:about="http://Localhost8080/OntologyHautNiveau.o
wl#Time"/>
    </owl:disjointWith>
    <owl:disjointWith>
     <owl:Class
rdf:about="http://Localhost8080/OntologyHautNiveau.o
wl#RelationShip"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class
rdf:about="http://Localhost8080/OntologyHautNiveau.o
wl#Time">
    <owl:disjointWith>
     <owl:Class
rdf:about="http://Localhost8080/OntologyHautNiveau.o
wl#State_Of_Affaires"/>
    </owl:disjointWith>
    <owl:disjointWith>
     <owl:Class
rdf:about="http://Localhost8080/OntologyHautNiveau.o
wl#Role"/>
    </owl:disjointWith>
    <owl:disjointWith>
     <owl:Class
rdf:about="http://Localhost8080/OntologyHautNiveau.o
wl#RelationShip"/>
    </owl:disjointWith>
    <owl:disjointWith
rdf:resource="http://Localhost8080/OntologyHautNivea
u.owl#Entity"/>
    <rdfs:subClassOf
rdf:resource="http://Localhost8080/OntologyHautNivea
u.owl#Concept"/>
  </owl:Class>
  <owl:Class
rdf:about="http://Localhost8080/OntologyHautNiveau.o
wl#RelationShip">
```

```
   <owl:disjointWith
rdf:resource="http://Localhost8080/OntologyHautNivea
u.owl#Time"/>
   <owl:disjointWith>
    <owl:Class
rdf:about="http://Localhost8080/OntologyHautNiveau.o
wl#State_Of_Affaires"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class
rdf:about="http://Localhost8080/OntologyHautNiveau.o
wl#Role"/>
   </owl:disjointWith>
   <owl:disjointWith
rdf:resource="http://Localhost8080/OntologyHautNivea
u.owl#Entity"/>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:minCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#in
t"
      >2</owl:minCardinality>
      <owl:onProperty>
       <owl:ObjectProperty
rdf:about="http://Localhost8080/OntologyHautNiveau.o
wl#inverse_of_participates_In"/>
      </owl:onProperty>
    </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
      <owl:ObjectProperty
rdf:about="http://Localhost8080/OntologyHautNiveau.o
wl#inverse_of_playing_Way_In"/>
      </owl:onProperty>
      <owl:minCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#in
t"
      >2</owl:minCardinality>
    </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf
rdf:resource="http://Localhost8080/OntologyHautNivea
u.owl#Concept"/>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
      <owl:ObjectProperty
rdf:about="http://Localhost8080/OntologyHautNiveau.o
wl#in"/>
      </owl:onProperty>
      <owl:minCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#in
t"
      >1</owl:minCardinality>
    </owl:Restriction>
   </rdfs:subClassOf>
 </owl:Class>
 <owl:Class
rdf:about="http://Localhost8080/OntologyHautNiveau.o
wl#Role">
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
      <owl:ObjectProperty
rdf:about="http://Localhost8080/OntologyHautNiveau.o
wl#playing_Way_In"/>
      </owl:onProperty>
      <owl:minCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#in
t"
      >1</owl:minCardinality>
    </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf
rdf:resource="http://Localhost8080/OntologyHautNivea
u.owl#Concept"/>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
      <owl:ObjectProperty
rdf:about="http://Localhost8080/OntologyHautNiveau.o
wl#requises"/>
      </owl:onProperty>
      <owl:minCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#in
t"
      >1</owl:minCardinality>
    </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:maxCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#in
t"
      >1</owl:maxCardinality>
      <owl:onProperty>
       <owl:ObjectProperty
rdf:about="http://Localhost8080/OntologyHautNiveau.o
wl#requises"/>
      </owl:onProperty>
    </owl:Restriction>
   </rdfs:subClassOf>
   <owl:disjointWith
rdf:resource="http://Localhost8080/OntologyHautNivea
u.owl#Time"/>
   <owl:disjointWith>…ETC.
```

UN EXTRAIT DU FICHIER: Ontologie_Du_Processus_D_Affaires_De_Haut_Niveau.Owl

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:xsp="http://www.owl-
ontologies.com/2005/08/07/xsp.owl#"
   xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
```

```
xmlns:p1="http://Localhost8080/OntologyHautNiveau.owl#"
   xmlns:swrl="http://www.w3.org/2003/11/swrl#"

xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"

xmlns="http://localhost8080/Ontology_De_Processus_D_Affaires_De_Haut_Niveau.owl#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"

xml:base="http://localhost8080/Ontology_De_Processus_D_Affaires_De_Haut_Niveau.owl">
 <owl:Ontology rdf:about="">
  <owl:imports rdf:resource="http://localhost8080/Ontology_Haut_Niveau.owl"/>
 </owl:Ontology>
 <owl:Class>
  <owl:unionOf rdf:parseType="Collection">
   <rdf:Description rdf:about="http://Localhost8080/OntologyHautNiveau.owl#Time"/>
   <rdf:Description rdf:about="http://Localhost8080/OntologyHautNiveau.owl#State_Of_Affaires"/>
   <rdf:Description rdf:about="http://Localhost8080/OntologyHautNiveau.owl#RelationShip"/>
   <rdf:Description rdf:about="http://Localhost8080/OntologyHautNiveau.owl#Entity"/>
   <rdf:Description rdf:about="http://Localhost8080/OntologyHautNiveau.owl#Role"/>
  </owl:unionOf>
 </owl:Class>
 <owl:Class rdf:ID="Resource">
  <owl:disjointWith>
   <owl:Class rdf:ID="Input"/>
  </owl:disjointWith>
  <owl:disjointWith>
   <owl:Class rdf:ID="Actor"/>
  </owl:disjointWith>
  <owl:disjointWith>
   <owl:Class rdf:ID="Output"/>
  </owl:disjointWith>
  <rdfs:subClassOf rdf:resource="http://Localhost8080/OntologyHautNiveau.owl#Role"/>
 </owl:Class>
 <owl:Class rdf:ID="Time_Interval">
  <rdfs:subClassOf rdf:resource="http://Localhost8080/OntologyHautNiveau.owl#Time"/>
 </owl:Class>
 <owl:Class rdf:ID="Requested_Authority">
  <rdfs:subClassOf>
   <owl:Class rdf:ID="Requires"/>
  </rdfs:subClassOf>
  <owl:disjointWith>
   <owl:Class rdf:ID="Required_Capability"/>
  </owl:disjointWith>
 </owl:Class>
 <owl:Class rdf:ID="Resource_Tangible">
  <owl:disjointWith>
   <owl:Class rdf:ID="Resource_Intangible"/>
  </owl:disjointWith>
  <rdfs:subClassOf rdf:resource="#Resource"/>
 </owl:Class>
 <owl:Class rdf:about="#Resource_Intangible">
  <owl:disjointWith rdf:resource="#Resource_Tangible"/>
  <rdfs:subClassOf rdf:resource="#Resource"/>
 </owl:Class>
 <owl:Class rdf:about="#Input">
  <rdfs:subClassOf rdf:resource="http://Localhost8080/OntologyHautNiveau.owl#Role"/>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
     >1</owl:cardinality>
    <owl:onProperty>
     <owl:ObjectProperty rdf:ID="input_has_State"/>
    </owl:onProperty>
   </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#Resource"/>
  <owl:disjointWith>
   <owl:Class rdf:about="#Actor"/>
  </owl:disjointWith>
  <owl:disjointWith>
   <owl:Class rdf:about="#Output"/>
  </owl:disjointWith>
 </owl:Class>
 <owl:Class rdf:ID="Produces_Outputs">
  <owl:disjointWith>
   <owl:Class rdf:ID="Takes_Inputs"/>
  </owl:disjointWith>
  <owl:disjointWith>
   <owl:Class rdf:ID="Executed_By"/>
  </owl:disjointWith>
  <owl:disjointWith>
   <owl:Class rdf:about="#Requires"/>
  </owl:disjointWith>
  <owl:disjointWith>
   <owl:Class rdf:ID="Uses_Consumes_Creates"/>
  </owl:disjointWith>
  <owl:disjointWith>
```

```
    <owl:Class rdf:ID="Executed_In"/>
   </owl:disjointWith>
   <rdfs:subClassOf
rdf:resource="http://Localhost8080/OntologyHautNivea
u.owl#RelationShip"/>
  </owl:Class>
  <owl:Class rdf:ID="Business_Process">
   <rdfs:subClassOf
rdf:resource="http://Localhost8080/OntologyHautNivea
u.owl#Entity"/>
   <owl:disjointWith>
    <owl:Class rdf:ID="Activity"/>
   </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:about="#Activity">
   <owl:disjointWith rdf:resource="#Business_Process"/>
   <rdfs:subClassOf
rdf:resource="http://Localhost8080/OntologyHautNivea
u.owl#Entity"/>
  </owl:Class>
  <owl:Class rdf:about="#Required_Capability">
   <owl:disjointWith
rdf:resource="#Requested_Authority"/>
   <rdfs:subClassOf>
    <owl:Class rdf:about="#Requires"/>
   </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#Takes_Inputs">
   <rdfs:subClassOf
rdf:resource="http://Localhost8080/OntologyHautNivea
u.owl#RelationShip"/>
   <owl:disjointWith
rdf:resource="#Produces_Outputs"/>
   <owl:disjointWith>
    <owl:Class rdf:about="#Executed_By"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#Requires"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#Uses_Consumes_Creates"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#Executed_In"/>
   </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:about="#Actor">
   <rdfs:subClassOf
rdf:resource="http://Localhost8080/OntologyHautNivea
u.owl#Role"/>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:minCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#in
t"
      >1</owl:minCardinality>
     <owl:onProperty>
      <owl:ObjectProperty
rdf:ID="playing_Way_In_Execution_By"/>
     </owl:onProperty>
    </owl:Restriction>
   </rdfs:subClassOf>
   <owl:disjointWith rdf:resource="#Resource"/>
   <owl:disjointWith rdf:resource="#Input"/>
   <owl:disjointWith>
    <owl:Class rdf:about="#Output"/>
   </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:about="#Output">
   <rdfs:subClassOf
rdf:resource="http://Localhost8080/OntologyHautNivea
u.owl#Role"/>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
      <owl:ObjectProperty rdf:ID="Output_has_State"/>
     </owl:onProperty>
     <owl:cardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#in
t"
      >1</owl:cardinality>
    </owl:Restriction>
   </rdfs:subClassOf>
   <owl:disjointWith rdf:resource="#Resource"/>
   <owl:disjointWith rdf:resource="#Input"/>
   <owl:disjointWith rdf:resource="#Actor"/>
  </owl:Class>
  <owl:Class rdf:about="#Executed_By">
   <rdfs:subClassOf
rdf:resource="http://Localhost8080/OntologyHautNivea
u.owl#RelationShip"/>…ETC.
```

UN EXTRAIT DU FICHIER: Ontologie_Du_Processus_Conference2.Owl

```
<?xml version="1.0"?>
<rdf:RDF

xmlns:swrla="http://swrl.stanford.edu/ontologies/3.3/s
wrla.owl#"

xmlns:p1="http://Localhost8080/OntologyHautNiveau.o
wl#"
   xmlns:swrl="http://www.w3.org/2003/11/swrl#"

xmlns:p2="http://localhost8080/Ontology_De_Processus
_D_Affaires_De_Haut_Niveau.owl#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-
schema#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"

xmlns="http://localhost8080/2012/1/5/OntologyBP.owl
#"
   xmlns:xsp="http://www.owl-
ontologies.com/2005/08/07/xsp.owl#"
   xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"

xmlns:protege="http://protege.stanford.edu/plugins/ow
l/protege#"
```

```
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
syntax-ns#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"

xmlns:sqwrl="http://sqwrl.stanford.edu/ontologies/built
-ins/3.4/sqwrl.owl#"

xml:base="http://localhost8080/2012/1/5/Ontology_Pro
cessus_Conference.owl">
 <owl:Ontology rdf:about="">
   <owl:imports
rdf:resource="http://sqwrl.stanford.edu/ontologies/built
-ins/3.4/sqwrl.owl"/>
   <owl:imports
rdf:resource="http://swrl.stanford.edu/ontologies/3.3/s
wrla.owl"/>
   <owl:imports
rdf:resource="http://localhost8080/Ontology_De_Proces
sus_D_Affaires_De_Haut_Niveau.owl"/>
 </owl:Ontology>
 <owl:Class
rdf:about="http://localhost8080/2012/1/5/OntologyBP.
owl#Processus_Lecture_part">
   <rdfs:subClassOf>
    <owl:Class
rdf:about="http://localhost8080/2012/1/5/OntologyBP.
owl#Processus_Lecture"/>
   </rdfs:subClassOf>
 </owl:Class>
 <owl:Class
rdf:about="http://localhost8080/2012/1/5/OntologyBP.
owl#Period_to_Ask_Question">
   <rdfs:subClassOf
rdf:resource="http://localhost8080/Ontology_De_Proces
sus_D_Affaires_De_Haut_Niveau.owl#Time_Interval"/>
   <owl:disjointWith>
    <owl:Class
rdf:about="http://localhost8080/2012/1/5/OntologyBP.
owl#Period_TO_Speak"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class
rdf:about="http://localhost8080/2012/1/5/OntologyBP.
owl#Time_TO_Complete_Lecture"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class
rdf:about="http://localhost8080/2012/1/5/OntologyBP.
owl#Period_to_Answer_Question"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class
rdf:about="http://localhost8080/2012/1/5/OntologyBP.
owl#Time_To_Unlock_Room"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class
rdf:about="http://localhost8080/2012/1/5/OntologyBP.
owl#Time_To_Start_Lecture"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class
rdf:about="http://localhost8080/2012/1/5/OntologyBP.
owl#Time_To_Lock_Room"/>
   </owl:disjointWith>
 </owl:Class>
 <owl:Class
rdf:about="http://localhost8080/2012/1/5/OntologyBP.
owl#Complete_Lecture_Produces_Outputs">
   <owl:disjointWith>
    <owl:Class
rdf:about="http://localhost8080/2012/1/5/OntologyBP.
owl#Speak_Produces_Outputs"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class
rdf:about="http://localhost8080/2012/1/5/OntologyBP.
owl#Answer_Question_Produces_Outputs"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class
rdf:about="http://localhost8080/2012/1/5/OntologyBP.
owl#Lock_Room_Produces_Outputs"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class
rdf:about="http://localhost8080/2012/1/5/OntologyBP.
owl#Start_LectureProduces_Outputs"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class
rdf:about="http://localhost8080/2012/1/5/OntologyBP.
owl#Unlocking_Room_Produces_Outputs"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class
rdf:about="http://localhost8080/2012/1/5/OntologyBP.
owl#Ask_Question_Produces_Outputs"/>
   </owl:disjointWith>
   <rdfs:subClassOf
rdf:resource="http://localhost8080/Ontology_De_Proces
sus_D_Affaires_De_Haut_Niveau.owl#Produces_Outputs
"/>
 </owl:Class>
 <owl:Class
rdf:about="http://localhost8080/2012/1/5/OntologyBP.
owl#Resource_Intangible_To_Speak">
   <owl:disjointWith>
    <owl:Class
rdf:about="http://localhost8080/2012/1/5/OntologyBP.
owl#Resource_Intangible_To_Start_Lecture"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class
rdf:about="http://localhost8080/2012/1/5/OntologyBP.
owl#Resource_Intangible_To_Complete_Lecture"/>
   </owl:disjointWith>
   <owl:disjointWith>
```

```
    <owl:Class
rdf:about="http://localhost8080/2012/1/5/OntologyBP.
owl#Resource_Intangible_To_Lock_Room"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class
rdf:about="http://localhost8080/2012/1/5/OntologyBP.
owl#Resource_Intangible_To_Unlock_Room"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class
rdf:about="http://localhost8080/2012/1/5/OntologyBP.
owl#Resource_Intangible_To_Ask_Question"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class
rdf:about="http://localhost8080/2012/1/5/OntologyBP.
owl#Resource_Intangible_To_Answer_Question"/>
   </owl:disjointWith>
   <rdfs:subClassOf>
    <rdf:Description
rdf:about="http://localhost8080/Ontology_De_Processu
s_D_Affaires_De_Haut_Niveau.owl#Resource_Intangible
">
     <owl:disjointWith>
      <owl:Class
rdf:about="http://localhost8080/2012/1/5/OntologyBP.
owl#Resource_To_Start_Lecture"/>
     </owl:disjointWith>
     <owl:disjointWith>
      <owl:Class
rdf:about="http://localhost8080/2012/1/5/OntologyBP.
owl#Resource_To_Speak"/>
     </owl:disjointWith>
     <owl:disjointWith>
      <owl:Class
rdf:about="http://localhost8080/2012/1/5/OntologyBP.
owl#Resource_To_Complete_Lecture"/>
     </owl:disjointWith>
     <owl:disjointWith>
      <owl:Class
rdf:about="http://localhost8080/2012/1/5/OntologyBP.
owl#Resource_To_Lock_Room"/>
     </owl:disjointWith>
     <owl:disjointWith>
      <owl:Class
rdf:about="http://localhost8080/2012/1/5/OntologyBP.
owl#Resource_To_Unlock_Room"/>
     </owl:disjointWith>
     <owl:disjointWith>
      <owl:Class
rdf:about="http://localhost8080/2012/1/5/OntologyBP.
owl#Resource_To_Ask_Question"/>
     </owl:disjointWith>
     <owl:disjointWith>
      <owl:Class
rdf:about="http://localhost8080/2012/1/5/OntologyBP.
owl#Resource_To_Answer_Question"/>
     </owl:disjointWith>
    </rdf:Description>
   </rdfs:subClassOf>
  </owl:Class>
  <owl:Class
rdf:about="http://localhost8080/2012/1/5/OntologyBP.
owl#Answered_Question">
   <rdfs:subClassOf
rdf:resource="http://localhost8080/Ontology_De_Proces
sus_D_Affaires_De_Haut_Niveau.owl#Output"/>
   <owl:disjointWith>
….ETC
```

UN EXTRAIT DU FICHIER: Ontologie_De_Domaine_D_Application_De_Haut_Niveau.owl

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:xsp="http://www.owl-
ontologies.com/2005/08/07/xsp.owl#"
   xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"

xmlns:p1="http://Localhost8080/OntologyHautNiveau.o
wl#"
   xmlns:swrl="http://www.w3.org/2003/11/swrl#"

xmlns:protege="http://protege.stanford.edu/plugins/ow
l/protege#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
syntax-ns#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-
schema#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"

xmlns="http://localhost8080/Ontologie_De_Domaine_D
_Application.owl#"

xml:base="http://localhost8080/Ontologie_De_Domaine
_D_Application.owl">
 <owl:Ontology rdf:about="">
  <owl:imports
rdf:resource="http://localhost8080/Ontology_Haut_Nive
au.owl"/>
 </owl:Ontology>
 <owl:Class rdf:ID="Acquired_authority">
  <owl:disjointWith>
   <owl:Class rdf:ID="Acquired_capability"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
   <owl:Class rdf:ID="Acquises"/>
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:ID="Basic_Entity">
  <rdfs:subClassOf
rdf:resource="http://Localhost8080/OntologyHautNivea
u.owl#Entity"/>
 </owl:Class>
 <owl:Class rdf:ID="Application_System">
  <rdfs:subClassOf>
```

```
    <owl:Class rdf:ID="Active_Entity"/>
   </rdfs:subClassOf>
   <owl:disjointWith>
    <owl:Class rdf:ID="Org_Unit"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:ID="Job_Position"/>
   </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:about="#Acquises">
   <rdfs:subClassOf
rdf:resource="http://Localhost8080/OntologyHautNivea
u.owl#RelationShip"/>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:maxCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#in
t"
      >1</owl:maxCardinality>
     <owl:onProperty>
      <owl:ObjectProperty rdf:ID="from_State"/>
     </owl:onProperty>
    </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
      <owl:ObjectProperty rdf:ID="to_State"/>
     </owl:onProperty>
     <owl:maxCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#in
t"
      >1</owl:maxCardinality>
    </owl:Restriction>
   </rdfs:subClassOf>
   <owl:disjointWith>
    <owl:Class rdf:ID="Attribute"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:ID="Service"/>
   </owl:disjointWith>
  </owl:Class>
  <owl:Class>
   <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Acquises"/>
    <owl:Class rdf:ID="State"/>
   </owl:unionOf>
  </owl:Class>
  <owl:Class rdf:ID="Passive_Entity">
   <rdfs:subClassOf rdf:resource="#Basic_Entity"/>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
      <owl:ObjectProperty
rdf:ID="between_Entities_1"/>
     </owl:onProperty>
     <owl:minCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#in
t"
      >1</owl:minCardinality>
    </owl:Restriction>
   </rdfs:subClassOf>
   <owl:disjointWith>
    <owl:Class rdf:about="#Active_Entity"/>
   </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:about="#Service">
   <owl:disjointWith>
    <owl:Class rdf:about="#Attribute"/>
   </owl:disjointWith>
   <owl:disjointWith rdf:resource="#Acquises"/>
   <rdfs:subClassOf
rdf:resource="http://Localhost8080/OntologyHautNivea
u.owl#RelationShip"/>
  </owl:Class>
  <owl:Class rdf:about="#Attribute">
   <owl:disjointWith rdf:resource="#Service"/>
   <owl:disjointWith rdf:resource="#Acquises"/>
   <rdfs:subClassOf
rdf:resource="http://Localhost8080/OntologyHautNivea
u.owl#RelationShip"/>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty>
      <owl:ObjectProperty rdf:ID="attribute_Owner"/>
     </owl:onProperty>
     <owl:cardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#in
t"
      >1</owl:cardinality>
    </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
    <owl:Restriction>
     <owl:cardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#in
t"
      >1</owl:cardinality>
     <owl:onProperty>
      <owl:ObjectProperty rdf:ID="attribute_Value"/>
     </owl:onProperty>
    </owl:Restriction>
   </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Goal">
   <rdfs:subClassOf>
    <owl:Class rdf:about="#State"/>
   </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#Acquired_capability">
   <owl:disjointWith
rdf:resource="#Acquired_authority"/>
   <rdfs:subClassOf rdf:resource="#Acquises"/>
  </owl:Class>
  <owl:Class rdf:about="#Job_Position">
   <rdfs:subClassOf>
    <owl:Class rdf:about="#Active_Entity"/>
   </rdfs:subClassOf>
   <owl:disjointWith>
    <owl:Class rdf:about="#Org_Unit"/>
```

```
   </owl:disjointWith>
   <owl:disjointWith
rdf:resource="#Application_System"/>
 </owl:Class>
 <owl:Class rdf:about="#Active_Entity">
   <owl:disjointWith rdf:resource="#Passive_Entity"/>
   <rdfs:subClassOf rdf:resource="#Basic_Entity"/>
 </owl:Class>
 <owl:Class rdf:ID="Outil">
   <rdfs:subClassOf rdf:resource="#Passive_Entity"/>
 </owl:Class>
 <owl:Class rdf:about="#Org_Unit">
   <owl:disjointWith rdf:resource="#Job_Position"/>
   <owl:disjointWith
rdf:resource="#Application_System"/>
   <rdfs:subClassOf rdf:resource="#Active_Entity"/>
 </owl:Class>
 <owl:Class>
   <owl:unionOf rdf:parseType="Collection">
     <owl:Class rdf:about="#Service"/>
     <owl:Class rdf:about="#Acquises"/>
   </owl:unionOf>
 </owl:Class>
 <owl:Class rdf:ID="Time_interval">
   <rdfs:subClassOf
rdf:resource="http://Localhost8080/OntologyHautNivea
u.owl#Time"/>
 </owl:Class>
 <owl:Class rdf:about="#State">
   <rdfs:subClassOf
rdf:resource="http://Localhost8080/OntologyHautNivea
u.owl#State_Of_Affaires"/>
   <rdfs:subClassOf>
     <owl:Restriction>
       <owl:onProperty>
         <owl:ObjectProperty
rdf:ID="of_Entities_Participating_In_1"/>
       </owl:onProperty>
       <owl:minCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#in
t"
       >1</owl:minCardinality>
     </owl:Restriction>
   </rdfs:subClassOf>
 </owl:Class>
 <owl:ObjectProperty rdf:ID="provides">
   <rdfs:subPropertyOf
rdf:resource="http://Localhost8080/OntologyHautNivea
u.owl#participates_In"/>
   <rdfs:range>
     <owl:Class>
       <owl:unionOf rdf:parseType="Collection">
…ETC.
```

UN EXTRAIT DU FICHIER: Ontologie_Du_Processus_Cours2.Owl

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:xsp="http://www.owl-
ontologies.com/2005/08/07/xsp.owl#"
   xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
   xmlns:swrl="http://www.w3.org/2003/11/swrl#"

xmlns:protege="http://protege.stanford.edu/plugins/ow
l/protege#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
syntax-ns#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-
schema#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"

xmlns:p2="http://localhost8080/Ontologie_De_Domaine
_D_Application.owl#"

xmlns="http://localhost8080/2012/1/5/OntologyDA.owl
#"

xml:base="http://localhost8080/2012/1/5/OntologyDA.o
wl">
 <owl:Ontology rdf:about="">
   <owl:imports
rdf:resource="http://localhost8080/Ontologie_De_Doma
ine_D_Application.owl"/>
 </owl:Ontology>
 <owl:Class rdf:ID="Authority_To_give_a_Course">
   <rdfs:subClassOf>
     <rdf:Description
rdf:about="http://localhost8080/Ontologie_De_Domain
e_D_Application.owl#Acquired_authority">
       <owl:disjointWith>
         <owl:Class rdf:ID="To_Unlock_Room"/>
       </owl:disjointWith>
       <owl:disjointWith>
         <owl:Class rdf:ID="Give_a_Course"/>
       </owl:disjointWith>
     </rdf:Description>
   </rdfs:subClassOf>
   <owl:disjointWith>
     <owl:Class rdf:ID="Authority_To_Unlock_Room"/>
   </owl:disjointWith>
 </owl:Class>
 <owl:Class rdf:ID="Students">
   <rdfs:subClassOf
rdf:resource="http://localhost8080/Ontologie_De_Doma
ine_D_Application.owl#Job_Position"/>
   <rdfs:subClassOf>
     <owl:Restriction>
       <owl:allValuesFrom>
         <owl:Class rdf:ID="PersonClass"/>
       </owl:allValuesFrom>
       <owl:onProperty
rdf:resource="http://Localhost8080/OntologyHautNivea
u.owl#instance_Of"/>
     </owl:Restriction>
   </rdfs:subClassOf>
   <owl:disjointWith>
     <owl:Class rdf:ID="Staff"/>
```

```
   </owl:disjointWith>
  </owl:Class>
  <owl:Class
rdf:ID="Authority_To_Complete_The_Course">
   <rdfs:subClassOf
rdf:resource="#Authority_To_give_a_Course"/>
   <owl:disjointWith>
    <owl:Class
rdf:ID="Authority_To_Transmit_The_Course"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class
rdf:ID="Authority_To_Start_The_Course"/>
   </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:ID="Time_To_Unlock_Class_Room">
   <rdfs:subClassOf
rdf:resource="http://localhost8080/Ontologie_De_Doma
ine_D_Application.owl#Time_interval"/>
   <owl:disjointWith>
    <owl:Class rdf:ID="Time_To_Start_The_Course"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class
rdf:ID="Time_To_Complete_The_Course"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class
rdf:ID="Period_To_Transmit_The_Course"/>
   </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:ID="Lecturer">
   <rdfs:subClassOf>
    <owl:Class rdf:about="#Staff"/>
   </rdfs:subClassOf>
   <owl:disjointWith>
    <owl:Class rdf:ID="Technician"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:ID="Watchman"/>
   </owl:disjointWith>
  </owl:Class>
  <owl:Class
rdf:ID="Audiovisual_Material_to_start_course">
   <rdfs:subClassOf>
    <rdf:Description
rdf:about="http://localhost8080/Ontologie_De_Domain
e_D_Application.owl#Passive_Entity">
     <owl:disjointWith>
      <owl:Class rdf:about="#PersonClass"/>
     </owl:disjointWith>
    </rdf:Description>
   </rdfs:subClassOf>
   <owl:disjointWith>
    <rdf:Description
rdf:about="http://localhost8080/Ontologie_De_Domain
e_D_Application.owl#Outil">
     <owl:disjointWith>
      <owl:Class rdf:ID="Class_Room"/>
     </owl:disjointWith>
     <owl:disjointWith
rdf:resource="#Audiovisual_Material_to_start_course"/>
     <owl:disjointWith>
      <owl:Class rdf:ID="Key"/>
     </owl:disjointWith>
     <owl:disjointWith>
      <owl:Class
rdf:ID="Audiovisual_Material_to_complete_course"/>
     </owl:disjointWith>
     <owl:disjointWith>
      <owl:Class
rdf:ID="Audiovisual_Material_to_transmet_course"/>
     </owl:disjointWith>
     <owl:disjointWith>
      <owl:Class rdf:ID="Course_Object"/>
     </owl:disjointWith>
    </rdf:Description>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#Key"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#Class_Room"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class
rdf:about="#Audiovisual_Material_to_transmet_course"
/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#Course_Object"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class
rdf:about="#Audiovisual_Material_to_complete_course"
/>
   </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:ID="Course_Is_Transmitted">
   <rdfs:subClassOf>
    <rdf:Description
rdf:about="http://localhost8080/Ontologie_De_Domain
e_D_Application.owl#Goal">
     <owl:disjointWith>
      <owl:Class rdf:ID="Course_is_Not_Started"/>
     </owl:disjointWith>
     <owl:disjointWith>
      <owl:Class rdf:ID="Course_is_not_Completed"/>
     </owl:disjointWith>
     <owl:disjointWith>
      <owl:Class rdf:ID="Class_Room_Is_Locked"/>
     </owl:disjointWith>
     <owl:disjointWith>
      <owl:Class rdf:ID="Class_Room_Is_Unlocked"/>
     </owl:disjointWith>
     <owl:disjointWith>
      <owl:Class rdf:ID="Course_Is_Not_Transmitted"/>
     </owl:disjointWith>
    </rdf:Description>
   </rdfs:subClassOf>
```

```
   <owl:disjointWith>
    <owl:Class rdf:ID="Course_Is_Completed"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:ID="unlocked_Class_Room"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:ID="Course_Is_Started"/>
   </owl:disjointWith>
  </owl:Class>
  <owl:Class
rdf:about="#Audiovisual_Material_to_complete_course"
>
   <owl:disjointWith
rdf:resource="http://localhost8080/Ontologie_De_Doma
ine_D_Application.owl#Outil"/>
   <owl:disjointWith>
    <owl:Class rdf:about="#Key"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#Class_Room"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class
rdf:about="#Audiovisual_Material_to_transmet_course"
/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#Course_Object"/>
   </owl:disjointWith>
   <owl:disjointWith
rdf:resource="#Audiovisual_Material_to_start_course"/>
    …ETC.
```

ANNEXE B

UN EXTRAIT DU FICHIER Ontologie_De_Processus_Conference2-Mappings-dpsm.Pprj

```
; Tue Jan 31 20:33:49 CET 2012
;
;+ (version "3.4.6")
;+ (build "Build 613")
([ANNOTATED_INSTANCE_WIDGET] of  Widget
        (name ":ANNOTATED-INSTANCE"))
([ANNOTATION_TEXT_WIDGET] of  Widget
        (height 100)
        (is_hidden FALSE)
        (name ":ANNOTATION-TEXT")
        (widget_class_name
"edu.stanford.smi.protege.widget.YellowStickyWidget")
        (width 200)
        (x 0)
        (y 0))
([BROWSER_SLOT_NAMES] of  Property_List
        (properties
        [Ontologie_De_processus_de_Conférence2-
mappings-dpsm_ProjectKB_Class10167]
        [Ontologie_De_processus_de_Conférence2-
mappings-dpsm_ProjectKB_Class10168]
        [Ontologie_De_processus_de_Conférence2-
mappings-dpsm_ProjectKB_Class10169]
        [Ontologie_De_processus_de_Conférence2-
mappings-dpsm_ProjectKB_Class10170]
        [Ontologie_De_processus_de_Conférence2-
mappings-dpsm_ProjectKB_Class10171]
        [Ontologie_De_processus_de_Conférence2-
mappings-dpsm_ProjectKB_Class10172]
        [Ontologie_De_processus_de_Conférence2-
mappings-dpsm_ProjectKB_Class10173]
        [Ontologie_De_processus_de_Conférence2-
mappings-dpsm_ProjectKB_Class10174]
        [Ontologie_De_processus_de_Conférence2-
mappings-dpsm_ProjectKB_Class10175]
        [Ontologie_De_processus_de_Conférence2-
mappings-dpsm_ProjectKB_Class10176]
        [Ontologie_De_processus_de_Conférence2-
mappings-dpsm_ProjectKB_Class10177]
        [Ontologie_De_processus_de_Conférence2-
mappings-dpsm_ProjectKB_Class10178]))
([CLSES_TAB] of  Widget
        (widget_class_name
"edu.stanford.smi.protege.widget.ClsesTab"))
([CREATION_TIMESTAMP_WIDGET] of  Widget
        (name ":CREATION-TIMESTAMP"))
([CREATOR_WIDGET] of  Widget
        (name ":CREATOR"))
([FORMS_TAB] of  Widget
        (widget_class_name
"edu.stanford.smi.protege.widget.FormsTab"))
([INSTANCE_ANNOTATION_FORM_WIDGET] of  Widget
        (height 476)
        (is_hidden FALSE)
        (name ":INSTANCE-ANNOTATION")
        (property_list
[INSTANCE_ANNOTATION_PROPERTY_LIST])
        (widget_class_name
"edu.stanford.smi.protege.widget.FormWidget")
        (width 603)
        (x 0)
        (y 0))
([INSTANCE_ANNOTATION_PROPERTY_LIST] of
Property_List
        (properties
                [ANNOTATED_INSTANCE_WIDGET]
                [CREATOR_WIDGET]
                [CREATION_TIMESTAMP_WIDGET]
                [ANNOTATION_TEXT_WIDGET]))
([INSTANCES_TAB] of  Widget
        (widget_class_name
"edu.stanford.smi.protege.widget.InstancesTab"))
([KB_656429_Class2] of  Map
)
([KB_685273_Class0] of  String
        (name "factory_class_name")
```

```
        (string_value
"edu.stanford.smi.protege.storage.clips.ClipsKnowledge
BaseFactory"))

([KB_685273_Class1] of  Map
        (entries
[Ontologie_De_processus_de_Conférence2-mappings-
dpsm_ProjectKB_Class10180])
        (referenced_maps
[Ontologie_De_processus_de_Conférence2-mappings-
dpsm_ProjectKB_Class10179]))
([LAYOUT_PROPERTIES] of  Property_List
        (name "layout properties")
        (properties [VERTICAL_STRETCHER]))
([Ontologie_De_processus_de_Conférence2-mappings-
dpsm_ProjectKB_Class10] of  Widget
        (height 150)
        (is_hidden FALSE)
        (label "Slot-maps")
        (name "slot-maps")
        (property_list
[Ontologie_De_processus_de_Conférence2-mappings-
dpsm_ProjectKB_Class11])
        (widget_class_name
"edu.stanford.smi.protege.widget.InstanceListWidget")
        (width 250)
        (x 250)
        (y 140))
([Ontologie_De_processus_de_Conférence2-mappings-
dpsm_ProjectKB_Class100] of  String
        (name "ButtonDescription-Remove")
        (string_value "Remove Selected Instances"))
([Ontologie_De_processus_de_Conférence2-mappings-
dpsm_ProjectKB_Class10003] of  Widget
        (is_hidden TRUE)
        (property_list
[Ontologie_De_processus_de_Conférence2-mappings-
dpsm_ProjectKB_Class10004])
        (widget_class_name "TGViztab.TGVizTab"))
([Ontologie_De_processus_de_Conférence2-mappings-
dpsm_ProjectKB_Class10004] of  Property_List
)
([Ontologie_De_processus_de_Conférence2-mappings-
dpsm_ProjectKB_Class10005] of  Widget
        (is_hidden TRUE)
        (property_list
[Ontologie_De_processus_de_Conférence2-mappings-
dpsm_ProjectKB_Class10006])
        (widget_class_name
"dfki.protege.ontoviz_tab.OntovizTab"))
([Ontologie_De_processus_de_Conférence2-mappings-
dpsm_ProjectKB_Class10006] of  Property_List
)
([Ontologie_De_processus_de_Conférence2-mappings-
dpsm_ProjectKB_Class10007] of  Widget
        (is_hidden TRUE)
        (property_list
[Ontologie_De_processus_de_Conférence2-mappings-
dpsm_ProjectKB_Class10008])
        (widget_class_name
"edu.stanford.smi.protege.widget.ClsesAndInstancesTab
"))
([Ontologie_De_processus_de_Conférence2-mappings-
dpsm_ProjectKB_Class10008] of  Property_List
)
([Ontologie_De_processus_de_Conférence2-mappings-
dpsm_ProjectKB_Class10009] of  Widget
        (is_hidden TRUE)
        (property_list
[Ontologie_De_processus_de_Conférence2-mappings-
dpsm_ProjectKB_Class10010])
        (widget_class_name
"edu.stanford.smi.RemoteKBTab.WordNetTab"))
([Ontologie_De_processus_de_Conférence2-mappings-
dpsm_ProjectKB_Class10010] of  Property_List
)
([Ontologie_De_processus_de_Conférence2-mappings-
dpsm_ProjectKB_Class10011] of  Widget
        (is_hidden TRUE)
        (property_list
[Ontologie_De_processus_de_Conférence2-mappings-
dpsm_ProjectKB_Class10012])
        (widget_class_name
"edu.stanford.smi.protegex.owl.ui.metadatatab.OWLMe
tadataTab"))
([Ontologie_De_processus_de_Conférence2-mappings-
dpsm_ProjectKB_Class10012] of  Property_List
)
([Ontologie_De_processus_de_Conférence2-mappings-
dpsm_ProjectKB_Class10013] of  Widget
        (is_hidden TRUE)
        (property_list
[Ontologie_De_processus_de_Conférence2-mappings-
dpsm_ProjectKB_Class10014])
        (widget_class_name
"edu.stanford.smi.protegex.owl.swrl.ui.tab.SWRLTab"))
([Ontologie_De_processus_de_Conférence2-mappings-
dpsm_ProjectKB_Class10014] of  Property_List
)
([Ontologie_De_processus_de_Conférence2-mappings-
dpsm_ProjectKB_Class10015] of  Widget
        (is_hidden TRUE)
        (property_list
[Ontologie_De_processus_de_Conférence2-mappings-
dpsm_ProjectKB_Class10016])
        (widget_class_name
"edu.stanford.smi.protegex.fctab.FacetConstraintsTab"))
([Ontologie_De_processus_de_Conférence2-mappings-
dpsm_ProjectKB_Class10016] of  Property_List
)
([Ontologie_De_processus_de_Conférence2-mappings-
dpsm_ProjectKB_Class10017] of  Widget
        (is_hidden TRUE)
        (property_list
[Ontologie_De_processus_de_Conférence2-mappings-
dpsm_ProjectKB_Class10018])
        (widget_class_name
"edu.stanford.smi.protegex.xml.tab.XMLTab"))
```

```
([Ontologie_De_processus_de_Conférence2-mappings-dpsm_ProjectKB_Class10018] of  Property_List
)
([Ontologie_De_processus_de_Conférence2-mappings-dpsm_ProjectKB_Class10019] of  Widget
        (is_hidden TRUE)
…ETC.
```

UN EXTRAIT DU FICHIER: Ontologie_De_Processus_Conference2 - Mappings- dpsm.Pont

```
; Tue Jan 31 20:33:49 CET 2012
;
;+ (version "3.4.6")
;+ (build "Build 613")
(defclass %3ACLIPS_TOP_LEVEL_SLOT_CLASS "Fake class to save top-level slot information"
        (is-a USER)
        (role abstract)
        (single-slot on-demand
                (type SYMBOL)
                (allowed-values FALSE TRUE)
                (default FALSE)
;+              (cardinality 0 1)
                (create-accessor read-write))
        (single-slot append-slotmap-values%3F
                (type SYMBOL)
                (allowed-values FALSE TRUE)
                (default FALSE)
;+              (cardinality 0 1)
                (create-accessor read-write))
        (multislot pre-execute-code
                (type INSTANCE)
;+              (allowed-classes executable-code)
                (create-accessor read-write))
        (single-slot apply-to-subclass-instances%3F
                (type SYMBOL)
                (allowed-values FALSE TRUE)
                (default FALSE)
;+              (cardinality 0 1)
                (create-accessor read-write))
        (single-slot global-target-slot
                (type STRING)
;+              (cardinality 0 1)
                (create-accessor read-write))
        (multislot post-execute-code
                (type INSTANCE)
;+              (allowed-classes executable-code)
                (create-accessor read-write))
        (single-slot name_
                (type STRING)
;+              (cardinality 0 1)
                (create-accessor read-write))
        (multislot slot-maps
                (type INSTANCE)
;+              (allowed-classes slot-mapping)
                (create-accessor read-write))
        (multislot per-instance-pre-execute-code
;+              (comment "ZP: Code is executed only if condition evaluates to true! ")
                (type INSTANCE)
;+              (allowed-classes executable-code)
                (create-accessor read-write))
        (single-slot source-class-desc
                (type INSTANCE)
;+              (allowed-classes source-class-description)
;+              (cardinality 0 1)
                (create-accessor read-write))
        (single-slot source-slot-composition
                (type STRING)
;+              (cardinality 0 1)
                (create-accessor read-write))
        (single-slot source-slot
                (type INSTANCE)
;+              (allowed-classes source-slot-description)
;+              (cardinality 0 1)
                (create-accessor read-write))
        (single-slot mapping-name
                (type STRING)
;+              (cardinality 0 1)
                (create-accessor read-write))
        (single-slot subclasses-accepted%3F
                (type SYMBOL)
                (allowed-values FALSE TRUE)
                (default TRUE)
;+              (cardinality 0 1)
                (create-accessor read-write))
        (single-slot condition
;+              (comment "value consists of an optional language spec prefix, followed by the expr to eval.\nE.g.: \"<LANG:TCL>(< *<.size>* 5)\"")
                (type STRING)
                (default "t")
;+              (cardinality 0 1)
                (create-accessor read-write))
        (single-slot code-name
                (type STRING)
;+              (cardinality 0 1)
                (create-accessor read-write))
        (single-slot language
                (type SYMBOL)
                (allowed-values TCL Python)
                (default Python)
;+              (cardinality 0 1)
                (create-accessor read-write))
        (multislot aux-source-classes-desc
                (type INSTANCE)
;+              (allowed-classes source-class-description)
                (create-accessor read-write))
        (multislot mappings
                (type INSTANCE)
;+              (allowed-classes instance-mapping)
                (create-accessor read-write))
        (single-slot code
                (type STRING)
```

```
;+              (cardinality 0 1)
                (create-accessor read-write))
        (multislot per-instance-post-execute-code
;+              (comment "ZP: Code is executed only if
condition evaluates to true! ")
                (type INSTANCE)
;+              (allowed-classes executable-code)
                (create-accessor read-write))
        (single-slot reverse-mapping
                (type SYMBOL)
                (allowed-values FALSE TRUE)
                (default FALSE)
;+              (cardinality 0 1)
                (create-accessor read-write))
        (single-slot target-slot
                (type INSTANCE)
;+              (allowed-classes target-slot-
description)
;+              (cardinality 0 1)
                (create-accessor read-write))
        (single-slot slot-map-name
                (type STRING)
;+              (cardinality 0 1)
                (create-accessor read-write))
        (single-slot const-val
                (type STRING)
;+              (cardinality 0 1)
                (create-accessor read-write))
        (single-slot target-class
                (type INSTANCE)
;+              (allowed-classes target-class-
description)
;+              (cardinality 0 1)
                (create-accessor read-write))
        (single-slot global-source-slot
                (type STRING)
;+              (cardinality 0 1)
                (create-accessor read-write)))
(defclass GLOBAL-MAPPING-METACLASS
        (is-a %3ASTANDARD-CLASS)
        (role concrete)
        (single-slot %3ADIRECT-INSTANCES
                (type INSTANCE)
;+              (allowed-classes %3ATHING)
;+              (cardinality 0 1)
                (create-accessor read-write)))
(defclass mapping
        (is-a USER)
        (role abstract)
        (multislot pre-execute-code
                (type INSTANCE)
;+              (allowed-classes executable-code)
                (create-accessor read-write))
        (multislot post-execute-code
                (type INSTANCE)
;+              (allowed-classes executable-code)
                (create-accessor read-write)))
(defclass slot-mapping "MC: Should we factor source-slot
here, declined to have 0, 1 or multiple values depending
on subclasses?\nActually, should be called \"source-
slots\"."
        (is-a mapping)
        (role abstract)
        (single-slot target-slot
                (type INSTANCE)
;+              (allowed-classes target-slot-
description)
;+              (cardinality 0 1)
                (create-accessor read-write))
        (single-slot append-slotmap-values%3F
                (type SYMBOL)
                (allowed-values FALSE TRUE)
                (default FALSE)
;+              (cardinality 0 1)
                (create-accessor read-write))
        (single-slot slot-map-name
                (type STRING)
;+              (cardinality 0 1)
                (create-accessor read-write)))
(defclass renaming-slot-mapping
        (is-a slot-mapping)
        (role concrete)
        (single-slot source-slot
                (type INSTANCE)
;+              (allowed-classes source-slot-
description)
;+              (cardinality 0 1)
                (create-accessor read-write)))
(defclass constant-slot-mapping
        (is-a slot-mapping)
        (role concrete)
        (single-slot const-val
                (type STRING)
;+              (cardinality 0 1)
                (create-accessor read-write)))
(defclass lexical-slot-mapping
        (is-a slot-mapping)
        (role concrete)
        (single-slot source-slot-composition
                (type STRING)
;+              (cardinality 0 1)
                (create-accessor read-write)))
(defclass functional-slot-mapping
        (is-a slot-mapping)
        (role concrete)
        (single-slot source-slot-composition
                (type STRING)
;+              (cardinality 0 1)
                (create-accessor read-write))
        (single-slot language
                (type SYMBOL)     …ETC.
```

UN EXTRAIT DU FICHIER : Ontologie_De_Processus_Conference2-Mappings-dpsm.Pins

```
; Tue Jan 31 20:33:49 CET 2012
;
```

```
;+ (version "3.4.6")
;+ (build "Build 613")
([global-mapping] of GLOBAL-MAPPING-METACLASS
)
([Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class0] of  instance-mapping
        (apply-to-subclass-instances%3F FALSE)
        (condition "t")
        (on-demand FALSE)
        (reverse-mapping FALSE)
        (source-class-desc
[Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class1])
        (target-class
[Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class2]))
([Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class1] of  source-class-description
        (name
"http://localhost8080/2012/1/5/OntologyBP.owl#To_Start_Lecture"))
([Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class10] of  source-class-description
        (name
"http://localhost8080/2012/1/5/OntologyBP.owl#Capability_To_Speak"))
([Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class100] of  source-class-description
        (name
"http://localhost8080/2012/1/5/OntologyBP.owl#Authority_To_Unlock_Room"))
([Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class10000] of  instance-mapping
        (apply-to-subclass-instances%3F FALSE)
        (condition "t")
        (on-demand FALSE)
        (reverse-mapping FALSE)
        (source-class-desc
[Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class61])
        (target-class
[Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class62]))
([Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class10001] of  instance-mapping
        (apply-to-subclass-instances%3F FALSE)
        (condition "t")
        (on-demand FALSE)
        (reverse-mapping FALSE)
        (source-class-desc
[Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class55])
        (target-class
[Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class56]))
([Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class10002] of  source-class-description
        (name
"http://localhost8080/2012/1/5/OntologyBP.owl#Unlocked_Room_input_to_Loked_Room"))
([Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class10003] of  instance-mapping
        (apply-to-subclass-instances%3F FALSE)
        (condition "t")
        (on-demand FALSE)
        (reverse-mapping FALSE)
        (source-class-desc
[Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class10002])
        (target-class
[Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class53]))
([Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class10004] of  source-class-description
        (name
"http://localhost8080/2012/1/5/OntologyBP.owl#Lecture_started_Input_to_Speak"))
([Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class10005] of  instance-mapping
        (apply-to-subclass-instances%3F FALSE)
        (condition "t")
        (on-demand FALSE)
        (reverse-mapping FALSE)
        (source-class-desc
[Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class10004])
        (target-class
[Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class53]))
([Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class10006] of  source-class-description
        (name
"http://localhost8080/2012/1/5/OntologyBP.owl#Lecture_completed_Output_of_complete_lecture"))
([Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class10007] of  source-class-description
        (name
"http://localhost8080/2012/1/5/OntologyBP.owl#Speak_Output_of_speak"))
([Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class10009] of  source-class-description
        (name
"http://localhost8080/2012/1/5/OntologyBP.owl#Lecture_started_Output_of_start_lecture"))
([Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class10010] of  source-class-description
        (name
"http://localhost8080/2012/1/5/OntologyBP.owl#Unlocked_Room_Output_of_unlocked_room"))
([Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class10012] of  instance-mapping
        (apply-to-subclass-instances%3F FALSE)
        (condition "t")
        (on-demand FALSE)
        (reverse-mapping FALSE)
        (source-class-desc
[Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class10010])
```

(target-class
[Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class56]))
([Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class10013] of instance-mapping
(apply-to-subclass-instances%3F FALSE)
(condition "t")
(on-demand FALSE)
(reverse-mapping FALSE)
(source-class-desc
[Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class10006])
(target-class
[Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class59]))
([Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class10014] of instance-mapping
(apply-to-subclass-instances%3F FALSE)
(condition "t")
(on-demand FALSE)
(reverse-mapping FALSE)
(source-class-desc
[Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class10009])
(target-class
[Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class59]))
([Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class10015] of instance-mapping
(apply-to-subclass-instances%3F FALSE)
(condition "t")
(on-demand FALSE)
(reverse-mapping FALSE)
(source-class-desc
[Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class10007])
(target-class
[Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class59]))
([Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class101] of target-class-description
(name
"http://localhost8080/2012/1/5/OntologyDA.owl#Authority_To_Unlock_Room"))
([Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class102] of instance-mapping
(apply-to-subclass-instances%3F FALSE)
(condition "t")
(on-demand FALSE)
(reverse-mapping FALSE)
(source-class-desc
[Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class103])
(target-class
[Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class104]))
([Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class103] of source-class-description
(name
"http://localhost8080/2012/1/5/OntologyBP.owl#Capability_To_Unlock_Room"))
([Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class104] of target-class-description
(name
"http://localhost8080/2012/1/5/OntologyDA.owl#Capability_To_Unlock_Room"))
([Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class11] of target-class-description
(name
"http://localhost8080/2012/1/5/OntologyDA.owl#Capability_To_Transmit_The_Course"))
([Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class12] of instance-mapping
(apply-to-subclass-instances%3F FALSE)
(condition "t")
(on-demand FALSE)
(reverse-mapping FALSE)
(source-class-desc
[Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class13])
(target-class
[Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class14]))
([Ontologie_De_processus_de_Conférence2-mappings-dpsm_Class13] of source-class-description
(name
"http://localhost8080/2012/1/5/OntologyBP.owl#Capability_To_Start_Lecture"))
…ETC.

ANNEXE C

Rapport sur l'hiérarchie source de Mapping

Mapping Hierarchy Report - Source

Generation Date: 2012-01-31 19:01:47
Generated By: Moufida

Ontologie_De_processus_de_Conférence2 - mapped item hierarchy

- owl:Thing
  - p1:Concept
    - p1:Time
      - p2:Time_Interval
        - *Period_TO_Speak*
        - *Time_TO_Complete_Lecture*
        - *Time_To_Start_Lecture*
        - *Time_To_Unlock_Room*
    - p1:RelationShip
      - p2:Requires
        - p2:Requested_Authority
          - *Authority_To_Complete_Lecture*
          - *Authority_To_Speak*
          - *Authority_To_Unlock_Room*
          - *Authority_To_Start_Lecture*
        - p2:Required_Capability
          - *Capability_To_Unlock_Room*
          - *Capability_To_Start_Lecture*
          - *Capability_To_Complete_Lecture*
          - *Capability_To_Speak*
        - *To_Speak*
        - *To_Start_Lecture*
        - *To_Complete_Lecture*
        - *To_Unlock_Room*
    - p1:Role
      - p2:Resource
        - *Resource_To_Speak*
        - *Resource_To_Start_Lecture*
        - *Resource_To_Unlock_Room*
        - *Resource_To_Complete_Lecture*
      - p2:Input
        - *Lecture_Given_Input_To_Complete_Lecture*
        - *Locked_Room_input_to_Unloked_Room*
      - p2:Actor
        - *intervenor*
        - *Key_Holder*
        - *Speaker*
    - p1:State_Of_Affaires
      - p2:State_Requirement
        - *Started_state_of_output_Start_lecture*
        - *locked_state_of_input_in_unlocked_room*
        - *unlocked_state_of_output_unlocked_room*
        - *Complete_state_of_output_Complete_lecture*
        - *Gived_state_of_output_in_speak*
        - *Unlocked_state_of_input_start_lecture*
        - *Started_state_of_input_in_speak*
        - *Tool_Efficace*
        - *Gived_state_of_input_Complete_lecture*

Rapport sur l’hiérarchie destinataire de Mapping

Mapping Hierarchy Report - Target

Generation Date: 2012-01-31 19:01:55
Generated By: Moufida

Ontologie_De_Domaine_D_Application_Processus_Cours2 - mapped item hierarchy

- owl:Thing
  - p1:Concept
    - p1:Entity
      - p2:Basic_Entity
        - p2:Passive_Entity
          - *Audiovisual_Material_to_start_course*
          - *Audiovisual_Material_to_complete_course*
          - *Course_Object*
          - *Audiovisual_Material_to_transmet_course*
          - *Class_Room*
          - *Key*
        - p2:Active_Entity
          - p2:Job_Position
            - Staff
              - *Lecturer*
              - *Technician*
              - *Watchman*
    - p1:Time
      - p2:Time_interval
        - *Time_To_Unlock_Class_Room*
        - *Time_To_Complete_The_Course*
        - *Time_To_Start_The_Course*
        - *Period_To_Transmit_The_Course*
    - p1:RelationShip
      - p2:Acquises
        - p2:Acquired_authority
          - Authority_To_give_a_Course
            - *Authority_To_Complete_The_Course*
            - *Authority_To_Start_The_Course*
            - *Authority_To_Transmit_The_Course*
          - *Authority_To_Unlock_Room*
        - p2:Acquired_capability
          - Capability_To_give_a_Course
            - *Capability_To_Complete_The_Course*
            - *Capability_To_Start_The_Course*
            - *Capability_To_Transmit_The_Course*
          - *Capability_To_Unlock_Room*
        - *To_Unlock_Room*
        - Give_a_Course
          - *To_Complete_The_Course*
          - *To_Start_The_Course*
          - *To_Transmit_The_Course*
      - p2:Attribute
        - *Effective*
    - p1:State_Of_Affaires
      - p2:State
        - p2:Goal
          - *Course_Is_Transmitted*
          - *Course_Is_Completed*
          - *unlocked_Class_Room*
          - *Course_Is_Started*
        - *Class_Room_Is_Locked*
        - *Course_is_not_Completed*
        - *Course_Is_Not_Transmitted*
        - *Course_is_Not_Started*