

LABORATOIRE



INFORMATIQUE, SIGNAUX ET SYSTÈMES
DE SOPHIA ANTIPOLIS
UMR 6070

PROBLÉMATIQUE DE L'EBXML ET LOGIQUE DE DESCRIPTION

Chan LE DUC, Nhan LE THANH

Projet MECOSI

Rapport de recherche
I3S/RR-2002-11-FR

mars 2002

RÉSUMÉ : Ce rapport étudie les problèmes de représentation des ontologies de l'ebXML. Ces problèmes sont identifiés et analysés au fur et à mesure de la présentation des fonctionnalités sémantiques du langage. Dans la spécification des ontologies de l'ebXML, la sémantique est encapsulée par les formes de représentation plus ou moins informelles : schémas d'UML, lexique de descriptions et règles de production. L'introduction d'un formalisme de représentation de ontologies de l'ebXML est donc indispensable. Cette représentation formelle permettra également de traiter le problème de transparence sémantique entre deux ontologies locales. Nous proposons donc dans ce rapport, un système hybride de représentation basé sur la logique de description. Cette logique est considérée dans la littérature comme un formalisme expressif pour la représentation sémantique. Cependant, certains développements de cette logique sont identifiés afin de la rendre plus expressive par rapports aux besoins sémantiques de l'ebXML.

MOTS CLÉS : ebXML, UML, sémantique du commerce électronique; ontologie, règle de production, représentation hybride, transparence sémantique, logique de description

ABSTRACT: This report introduces semantic representation of ebXML and its problems. These problems are identified and analysed when we attempt to investigate the functionality specifications of ebXML. Our study about the way of representing the knowledge allows to identify used formalisms in ebXML. They are UML, lexicon and production rules. Some of them possess informal semantic. A formalization of the semantics is necessary because the functionality specifications of ebXML require manipulations on the captured semantic. Therefore, we propose a hybrid formal representation in which the semantics of these formalisms are encapsulated. In addition, data interchanges in ebXML between heterogeneous partners are based on the different lexicons. Hence, a reorganization of the lexicons forming the relevant ontologies and an integration of them are required. Description Logic is considered as expressive formalism for semantic representation. It is introduced on behalf of the unified formalism for the formalization of the semantics. Some extensions, however, are added in order to increase its expressiveness for requirements of ebXML. Finally, that formalization implies a mapping between the semantics of the formalisms used by ebXML and the extension of description logic.

KEY WORDS : ebXML, UML, e-commerce semantic, ontology, production rules, hybrid representation, semantic transparency, description logic

Problématique de l'ebXML et logique de description

Chan Le Duc
cleduc@i3s.unice.fr

Nhan Le Thanh
Nhan.Le-Thanh@unice.fr

Rapport de recherche
Laboratoire I3S - Université de Nice

Résumé

This report introduces semantic representation of ebXML and its problems. These problems are identified and analysed when we attempt to investigate the functionality specifications of ebXML. Our study about the way of representing the knowledge allows to identify used formalisms in ebXML. They are UML, lexicon and production rules. Some of them possess informal semantic. A formalization of the semantics is necessary because the functionality specifications of ebXML require manipulations on the captured semantic. Therefore, we propose a hybrid formal representation in which the semantics of these formalisms are encapsulated. In addition, data interchanges in ebXML between heterogeneous partners are based on the different lexicons. Hence, a reorganization of the lexicons forming the relevant ontologies and an integration of them are required.

Description Logic is considered as expressive formalism for semantic representation. It is introduced on behalf of the unified formalism for the formalization of the semantics. Some extensions, however, are added in order to increase its expressiveness for requirements of ebXML. Finally, that formalization implies a mapping between the semantics of the formalisms used by ebXML and the extension of description logic.

Keywords : ebXML, UML, commerce semantic, production rules, hybrid representation, semantic transparency, description logic.

1 Introduction

Le développement du commerce électronique exige une plate-forme flexible, dynamique et ouverte facilitant l'échange de données entre les partenaires en tenant compte de l'hétérogénéité de leurs activités et de leur taille. L'échange de données de façon traditionnelle avec l'EDI¹ montre des difficultés infran-

¹Electronic Data Interchange

chissables pour les acteurs dont les moyens sont modérés [1]. De plus, l'apparition de XML avec la capacité de structurer des données de ce langage nous permet de penser à des échanges de données plus riches sur Internet. Dans ce contexte, l'ebXML² est créé et il semble qu'il répondrait à ces besoins. Et pourtant, nous n'avons pas tous les outils nécessaires pour réaliser les spécifications de l'ebXML. L'un des outils manquants est un mécanisme capable de représenter la sémantique commerciale impliquée par les formalismes utilisés. Un tel mécanisme devrait faciliter la réalisation de ces spécifications, par exemple la composition des documents commerciaux, des manipulations sur les processus commerciaux, l'échange de données transparent sémantique, etc.

En effet, les connaissances commerciales sont constituées de facteurs divers que nous ne réussissons pas toujours à capturer par les outils actuels. Typiquement, l'UML³ est considéré comme un outil puissant pour la modélisation mais il ne peut que modéliser les connaissances des processus prévisibles, déterministes et sans répudiation. Pour faciliter la capture et la modélisation des connaissances commerciales, l'UN/CEFACT propose un outil, appelé l'UMM⁴, qui est considéré comme une extension de l'UML pour le commerce. Puisque l'ebXML utilise cet outil pour la modélisation des processus commerciaux et de la composition des documents d'échange, la sémantique capturée est caractérisée par la prévisibilité, c'est-à-dire que toutes les transactions ont des rôles, des bornes de temps, des succès et échec bien déterminés, par la capacité de créer des connexions légales, et par la non-répudiation, c'est-à-dire que les conduites commerciales légales s'imposent. Bien que cette limite facilite déjà la représentation de connaissances, la sémantique informelle de l'UML dont l'ebXML hérite nous empêche d'échanger des données de façon transparente sémantique. En outre, l'ebXML se sert également du formalisme des règles de production pour représenter une partie des connaissances, une définition de la sémantique formelle, commune et compatible à celles de formalismes utilisés n'est pas établie automatiquement. Il existe des efforts de l'ebXML pour surmonter ces difficultés, par exemple le mécanisme de contexte, mais ils ne peuvent que répondre partiellement au problème. Par ailleurs, si l'on rend le scénario d'échange plus flexible en acceptant des connaissances sans partager, les difficultés se multiplieront.

Il existe déjà des manières de représenter la sémantique de langage, par exemple, le réseau sémantique, le graphe conceptuel, etc. mais aucune parmi elles ne donne des services d'inférence efficaces sur la sémantique capturée. La logique de description (DL), qui est inspirée du réseau sémantique et considérée comme un fragment décidable de la FOL⁵, montre une capacité expressive suffisante (moyennant une extension) par le fait qu'elle représente

²Electronic Business XML

³Unified Modeling Language

⁴UN/CEFACT Modeling Methodology

⁵First Order Logic

la sémantique du modèle entité - relation [2]. Ainsi, une extension de la DL pour la sémantique du modèle orienté - objet est en cours de recherche. Cela nous permet de penser à développer des ontologies de l'ebXML en utilisant la DL pour la formalisation de la sémantique. Ces ontologies avec un mécanisme d'intégration établissent une base sur laquelle l'échange de données de façon transparente sémantique se fonde.

En sachant que nous n'arrivons jamais à capturer *totalemment* une sémantique impliquée par un langage (langage naturel, langage de modélisation, etc.), la formalisation d'une sémantique vise toujours à un ensemble de services d'inférences dont les spécifications ebXML ont besoin. Cependant, notre approche se concentre sur les formalismes utilisés (UML, règles de production, etc.), la formalisation obtenue est encore utilisée par d'autres applications dans lesquelles ces formalismes et services d'inférences sont exigés.

Le reste de ce rapport sera organisé en sections. Dans la deuxième section, nous commençons par présenter la représentation des connaissances commerciales de l'ebXML et un exemple illustrant comment découvrir, définir un processus commercial à partir des modèles de données UMM et comment composer un document commercial à partir de composants de base et des informations contextuelles. Une analyse sur les formalismes utilisés et sur la relation entre eux achève la première sous-section. La deuxième sous-section présente un scénario d'échange avec l'ontologie partagée ebXML dans lequel le premier problème se pose : la nécessité d'une représentation hybride pour la sémantique ebXML. Le deuxième scénario d'échange qui introduit des ontologies locales dans le système fait apparaître le deuxième problème : échanger de façon transparente sémantique pour les acteurs. Une petite synthèse sur les problèmes cités termine la section.

La troisième section commence par la présentation de la DL. Nous proposons également quelques facteurs ajoutés à la DL afin de répondre à la représentation de la sémantique de diagrammes UML. Le reste de la section se concentre sur la transformation de la sémantique UML vers des expressions DLs. Une brève discussion sur la transformation des règles de production vers des expressions DLs achève la section.

2 Problématique de l'ebXML

2.1 Sémantique de l'ebXML

2.1.1 Composants de l'ebXML portant la sémantique commerciale

Une description brève des composants de l'ebXML portant la sémantique commerciale est présentée dans [1]. La description en détail sur ces composants se trouve dans [10], [12] et [13]. Puisque notre objectif est de découvrir les formalismes utilisés par ebXML, nous ne fournissons qu'une vision sur la

relation au niveau de la représentation de connaissances entre l'UML, UMM et ebXML. Une version de cette vision est présentée par la figure *Fig. 1*.

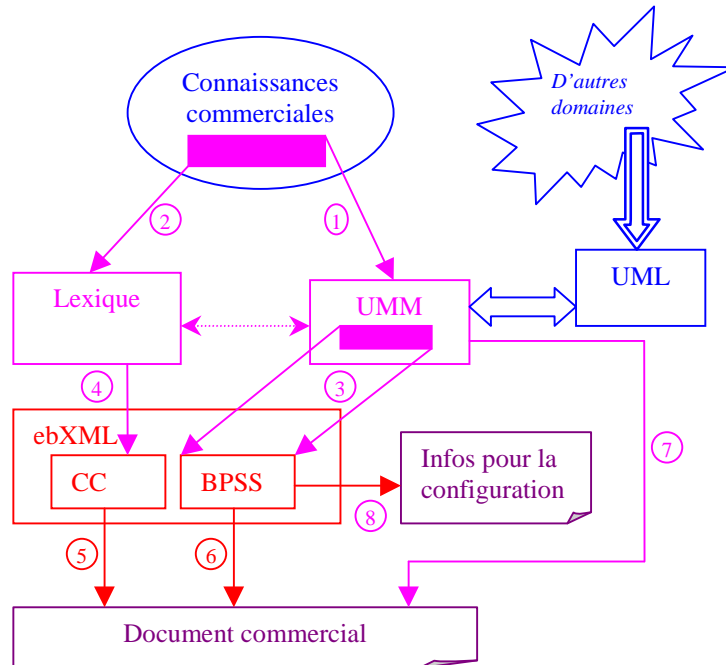


FIG. 1 – Relation entre UML, UMM et ebXML

L'ebXML propose un ensemble de fiches descriptives (*worksheet*), les diagrammes d'état et d'activité conformément à l'UMM permettant de représenter plus pratiquement les connaissances commerciales (*flèche 1*). A partir de ces fiches, l'utilisateur peut définir ses processus commerciaux décrivant ses affaires. L'ebXML construit deux parties importantes portant la sémantique. La première partie, appelée *Business Process Specification Schema* (BPSS), décrit un sous-ensemble de la sémantique (*flèche 3*). BPSS a pour objectif de fournir des informations de configurations du système d'échange en durée d'exécution (*flèche 8*). L'utilisateur peut extraire des fiches descriptives des éléments et relations (processus commerciaux, modèle d'informations) conformément à BPSS afin de les introduire aux *Business Process Specification* (BSP). La deuxième partie, appelée *composants de base* (*Core Component* CC), décrit les composants libres de contexte permettant la réutilisation pour la composition de documents (*flèche 5*). Ces composants sont dérivés du lexique et du méta-modèle de l'UMM (*flèche 4*). Le document commercial se compose de composants de base et d'informations contextuelles extraites de fiches descriptives (*flèche 5,6,7*). Les connaissances suivant les *flèches 6,7* sont représentées par les règles de production. Une autre version plus lisible au niveau de la participation des formalismes

est présentée par la figure *Fig.2*.

Le formalisme UML ne peut que capturer une partie des connaissances commerciales. Il existe des travaux, [7] et [8], qui proposent un formalisme ayant l'ambition de représenter la totalité des connaissances commerciales mais ils sont confrontés à plusieurs problèmes à franchir au niveau de la théorie. Une contribution de l'ebXML est de définir BPSS, un sous-ensemble sémantique de l'UMM, permettant aux utilisateurs à la fois de configurer des interfaces, logiciels du système d'échange en durée d'exécution (chorégraphie des transactions commerciales, etc.) et de définir BPS conformément à ses affaires. BPSS est décrit par deux versions : l'une en diagrammes de classes UML, l'autre en XML. En outre, une autre contribution de l'ebXML est de construire les composants de base réutilisables CC. Ces composants de base avec les règles de production portant les informations contextuelles permettent de composer les documents commerciaux. Ces règles résultent du modèle d'information UMM (fiches descriptives).

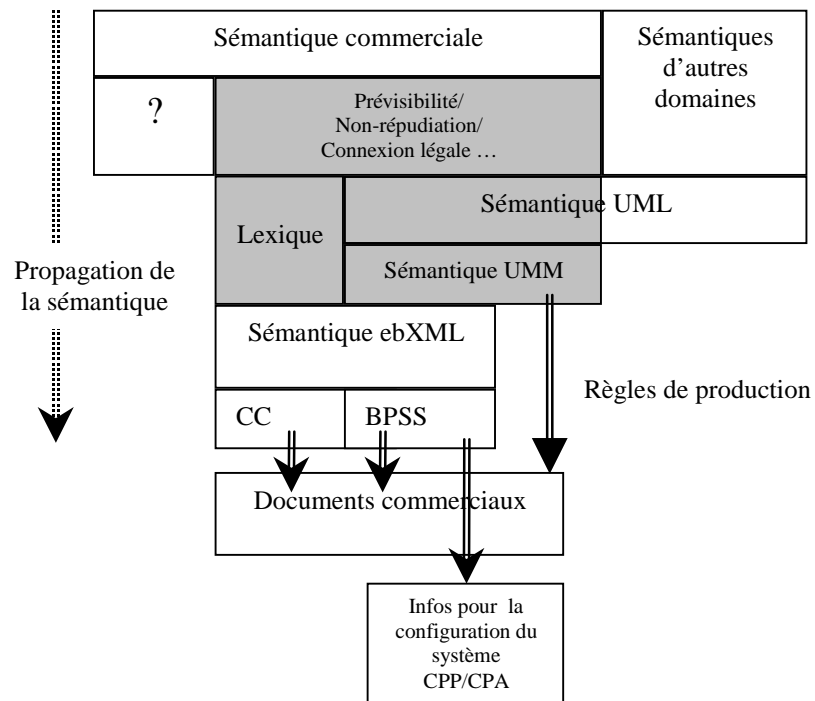


FIG. 2 – *Formalismes utilisés par ebXML*

La composition/interprétation des documents commerciaux à l'aide du mécanisme de contexte est capable de répondre à la différence de vocabulaires des acteurs. Et pourtant, cette différence se limite par la spécification ebXML aux catégories contextuelles autorisées. Rien n'assure que ces caté-

gories soient exhaustives.

2.1.2 Exemple sur la capture sémantique commerciale

Nous considérons un exemple sur le modèle commercial : *Direct-To-Customer-Drop-Ship-Retail-Model* montrant comment l'ebXML capture et représente la sémantique d'un processus commercial (*voir l'annexe*). Cet exemple illustrera également l'utilisation des informations extraites du processus commercial dans la composition/interprétation du document commercial. Pour éviter des détails lourds, nous ne présentons que des fiches simplifiées.

Dans un premier temps, on utilise la *fiche 1, Fig. 6* pour décrire le type de processus. Chaque type de processus est une série de processus formant un secteur commercial. L'identification de ces processus s'effectue dans les fiches : *Gestion de commandes d'achat, Commande de vente, Livraison de marchandises*, etc. Il y a 5 acteurs participant aux processus d'échange : *Détaillant, DSVendeur, MétierTransport, AutoritéCrédit* et *Consommateur*. La figure *Fig. 9* indique les processus d'échange entre les acteurs et leur rôle. Nous allons examiner le processus *Gestion de commandes d'achat* effectué par *Détaillant* et *DSVendeur* (*Fig. 8*). S'il existe une commande de vente effectuée par un consommateur, le processus sera déclenché. Un processus commercial peut se constituer de plusieurs collaborations. Mais le processus *Gestion de commandes d'achat* ne comprend qu'une collaboration *Create-Vendor-Purchase-Order*. La *fiche 2, la Fig. 10* et la *Fig. 11* s'utilisent pour décrire la collaboration. Dans cette phase de la capture de connaissances, l'aspect de temps entre en compte. Cela entraînerait à un ajout de l'aspect de temps au formalisme utilisé. Cet aspect est présent dans les diagrammes d'états et d'activités d'UML.

La collaboration possède la transaction *Create-Vendor-Purchase-Order*. La *fiche 3, la Fig. 12* et la *Fig. 13* s'utilisent pour décrire la transaction *Create-Vendor-Purchase-Order*. Chaque transaction correspond toujours à un document échangé. Dans ce cas, les deux acteurs qui participent à la transaction sont : *Détaillant* et *DSVendeur*. Le document qui est envoyé du détaillant au DSVendeur décrit les informations sur l'acheteur, le vendeur, les produits demandés par l'acheteur, etc.

A partir des connaissances commerciales acquises qui sont contenues dans les fiches, la composition du document commence par une proposition d'un squelette du document (*fiche 5, Fig. 15, Fig. 16*) et par la recherche des composants de base appropriés. Les informations contextuelles sont également déterminées (*fiche 4, Fig. 14*). Ces informations sont représentées par des règles de production, par exemple :

- i) Si un acheteur est de la région U.S., les descriptions des produits n'incluront pas les lignes d'articles dans la commande.

- ii) Si un vendeur est en France, la description des produits sera incluse dans la commande.
- iii) Si l'industrie d'un acheteur est l'automobile, la catégorie des produits sera ajoutée aux lignes d'articles de la commande.
- iv) Si une entité d'informations catégorique de produits existe et l'industrie d'un vendeur est chimique, un attribut sera ajouté à la catégorie du produit pour indiquer la toxicité du produit dans cette catégorie.

Après avoir appliqué ces règles sur les composants de base, on obtiendra les composants avec la sémantique déterminée. Le document final se compose de ces composants.

L'application des règles contextuelles sur les composants de base trouvés peut poser des problèmes, par exemple, un conflit sur le résultat (si l'on applique les règles i et ii sur une ligne de produit dans le document) ou le résultat obtenu dépend de l'ordre appliqué des règles (si l'on applique les règles iii et iv sur une ligne de produit dans le document). Si la règle iv est appliquée avant, l'attribut de toxicité ne sera pas introduit car la catégorie de produit n'existe pas encore.

2.1.3 Synthèse sur les formalismes utilisés

A travers la présentation des composants de l'ebXML dans les sous-sections précédentes, nous nous rendons compte que la sémantique commerciale de l'ebXML se compose de celle de diagrammes UML, d'une ontologie incluant le CC. Autrement dit, le formalisme UML avec les explications des termes situés dans l'ontologie est capable de représenter toutes les connaissances dont l'ebXML a besoin. En effet, un diagramme de classes UML s'utilise pour décrire le méta-modèle du CC permettant d'y introduire les informations contextuelles. Les composants de base avec la sémantique sont instanciés à partir de ces diagrammes. BPSS est également représenté par un diagramme de classes dans lequel la sémantique des collaborations, des transactions d'un processus commercial est capturée. Les diagrammes d'états et d'activités s'utilisent pour exprimer les comportements d'objets et les interactions entre eux. Une ontologie définit tous les termes utilisés dans ces diagrammes. En outre, la composition des documents commerciaux se basant sur ces connaissances représentées et sur le mécanisme de contexte exigerait sans doute un autre formalisme pour représenter les règles de production. Chaque règle se représente en deux parties dont la conclusion inclut une *action*. Un formalisme souhaité pour ces règles serait capable de franchir les problèmes cités (conflit de résultats, dépendance de l'ordre appliqué) dans l'exemple.

Ces formalismes se montrent de quelques avantages en permettant de capturer la majorité des connaissances. Néanmoins, ils nous mettent face à une grande difficulté : la sémantique décrite par le langage naturel. La formalisation de cette sémantique est indispensable car les spécifications de fonction-

nalités de l'ebXML nécessitent des manipulations sur la sémantique, c'est-à-dire les services d'inférences, par exemple, gestion des processus commerciaux, composition des documents commerciaux, intégration des lexiques, etc. De plus, une sémantique formelle partagée et compatible à celles de formalismes utilisés est exigée pour répondre à l'hétérogénéité de ces formalismes. Il est évident que l'on ne peut pas capturer une sémantique en totalité, la formalisation de la sémantique abordée doit viser toujours aux fonctionnalités concrètes et bien définies, par exemple, les spécifications de fonctionnalités de l'ebXML mentionnées.

Dans la sous-section suivante, nous essayons d'analyser encore plus les problèmes cités en mettant l'accent sur la transparence sémantique dans le contexte d'échange de données ebXML entre deux acteurs.

2.2 Scénario d'échange avec l'ontologie partagée : problème de la représentation hybride

Nous présentons un scénario simple d'échange de documents utilisés dans un système supportant plusieurs acteurs. Ce scénario correspond à la sémantique statique qui ne change pas tout au long du processus d'échange. Les deux composants de l'ebXML : les processus commerciaux et composants de base couvrant la sémantique commerciale s'appuient sur une ontologie partagée (*Fig. 3*).

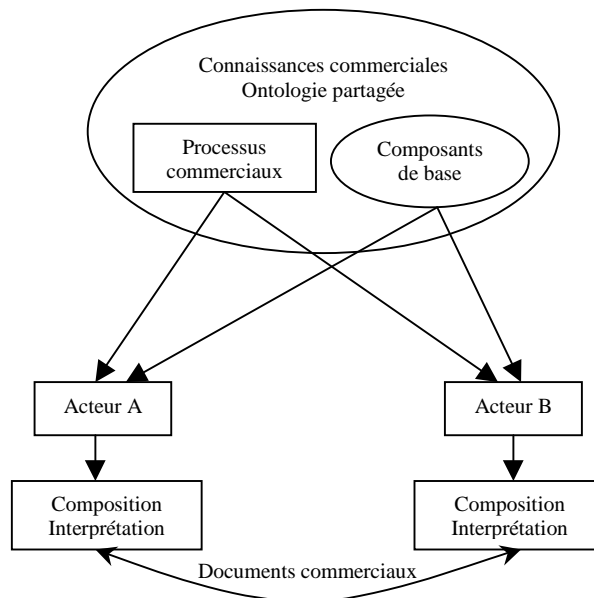


FIG. 3 – Scénario d'échange avec une sémantique identique et statique

L'ebXML utilise ce scénario et propose un mécanisme contextuel pour ré-

soudre partiellement le problème des différences sémantiques impliquées par plusieurs vocabulaires : un vocabulaire neutre sous forme des composants de base engendre différents vocabulaires en y réunissant des informations contextuelles. L'essentiel de ce scénario est d'atteindre les mêmes connaissances ou la même sémantique sur les processus concernés avant le premier échange et cette sémantique ne change pas tout au long de l'échange. Par exemple, l'acteur A envoie à l'acteur B une commande d'achat qui contient un produit *Ordinateur-Superbe-Qualité*. Dans ce scénario ce terme doit être défini dans l'ontologie partagée (composants de base avec des informations contextuelles) pour que les deux acteurs se comprennent.

Afin de supporter différents vocabulaires, l'ebXML construit un diagramme de classes UML sous la forme de structure hiérarchique. Les composants de base qui sont instanciés à partir de la racine de ce diagramme sont les définitions des vocabulaires sans information contextuelle, appelés *Core Component Type*. Chaque fois le processus d'échange est défini avec la disponibilité des informations contextuelles, les descendants *Basic Information Entity* et *Aggregate Basic Information Entity* sont déterminés par ces informations contextuelles. Ainsi, les vocabulaires appropriés à ce contexte se produisent. Par exemple, un *Core Component Type* : *adresse* est défini sans champs *etat*. Une fois qu'un partenaire d'échange est déterminé comme étant des Etat-Unis, le champs *etat* y est rajouté. L'ebXML propose d'utiliser 5 types de contextes : *Processus*, *Produit*, *Géographie*, *Industrie*, *Culture*, c'est-à-dire que les modifications possibles sur les vocabulaires doivent se limiter à ces catégories.

Par ailleurs, les spécifications ebXML demandent un mécanisme capable de gérer les processus commerciaux qui sont définis comme l'ensemble des diagrammes UML et le lexique des termes. En particulier, le système devrait permettre de vérifier si un processus existe dans le système, un processus est défini à partir des processus existants, etc. C'est pourquoi la réalisation de ces spécifications exige de manipuler la sémantique des formalismes. Pour répondre à cet exigence, l'ontologie partagée doit se fonder sur un formalisme générique unifiant l'UML, le lexique, les règles de production ensemble. Autrement dit, une représentation hybride sera la solution à tel système. Et pourtant, une telle représentation implique un problème important : la définition de la sémantique commune et cohérente, qui est partagée par les formalismes utilisés.

En bref, cette analyse nous amène à tenir compte des tâches suivantes pour le système ebXML : formaliser la sémantique ebXML qui vient de celles des formalismes : UML, règles de production ; introduire des services d'inférences sur la sémantique représentée supportant les fonctionnalités attendues. Une approche proposée pour ces tâches est de choisir un formalisme générique dans lequel le formalisme UML s'immerge. D'autres formalismes s'y rajoutent au fur et à mesure. Nous revenons à ce sujet dans la section suivante.

2.3 Scénario d'échange avec l'ontologie locale : problème de la transparence sémantique

L'eCommerce est considéré comme i) un marché énorme et ouvert (plusieurs entreprises sont capables d'y entrer et d'en sortir librement) ; ii) un marché avec des relations dynamiques (des associations sont formées et annulées facilement et fréquemment) ; iii) un marché avec l'hétérogénéité des activités de partenaires. L'ambition de l'ebXML est de répondre aux besoins du marché à tel niveau. Il sera infaisable si tous les acteurs se restreignent à utiliser le même vocabulaire avec des compréhensions identiques sur les termes dans ce vocabulaire pour échanger des documents. De plus, il est très fréquent que la résolution des différences sémantiques entre les partenaires hétérogènes s'effectue dynamiquement pendant l'échange, c'est - à - dire de façon "en ligne". C'est pourquoi le fait de supporter un scénario d'échange avec la sémantique dynamique est indispensable pour l'ebXML (Fig. 4).

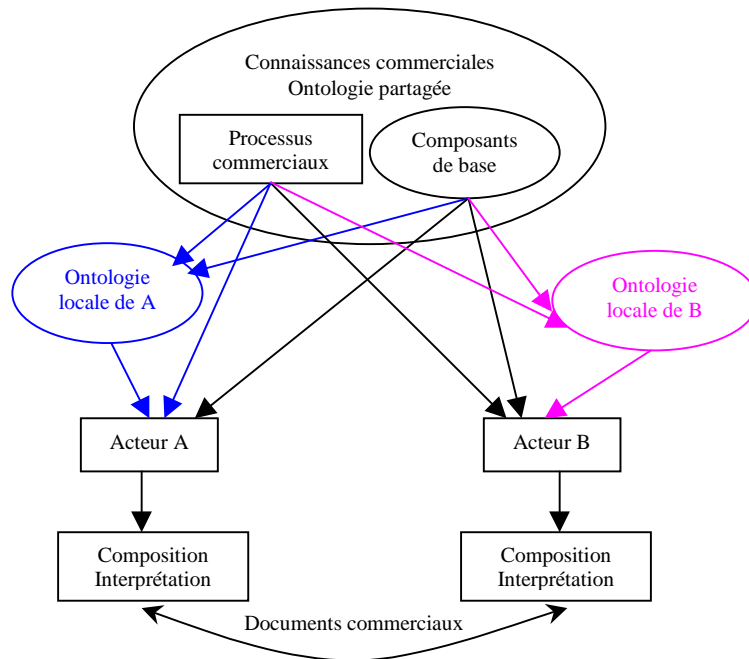


FIG. 4 – Scénario d'échange avec une sémantique dynamique

L'essentiel de ce scénario est d'introduire une ontologie locale pour chaque acteur permettant de définir les termes propres. La relation entre les ontologies locales et l'ontologie partagée (elle contient les termes de base) pose des problèmes essentiels de ce scénario. On reprend l'exemple ci-dessus. Le produit *Ordinateur-Superbe-Qualité* est défini dans l'ontologie OA en XML comme suit (OP : Ontologie Partagée, OA : Ontologie de A ; OB : Ontologie de B)

```

<OA : Ordinateur-Superbe-Qualité>
  <OP : Processeur> " OA : Processeur-Performant "
  </OP : Processeur>
  <OP : Mémoire> > 256Mo </OP : Mémoire>
  <OP : Disque> SCSI </OP : Disque>
  <OA : Ecran> " OA : Résolution- Elevée " </OA : Ecran>
</OA : Ordinateur-Superbe-Qualité>

```

Quand l'acteur B reçoit le terme *Ordinateur-Superbe-Qualité*, B ne le comprend pas car ce terme n'est défini ni dans l'OB ni dans l'OP (B ne peut comprendre " OA : *Processeur-Performant*" ou " OA : *Résolution-Elevée*"). Un appariement entre l'OA, l'OB avec la référence vers l'OP est exigé pour déterminer ce terme dans l'OB. Ce fait implique deux phases : construction du terme (à partir des informations externes : OA, OP) et appariement entre cette construction et l'OB pour obtenir la définition du terme dans l'OB. Il est probable que les deux définitions du terme sont différentes au niveau de la structure mais elles doivent être identiques ou ne pas être conflictuelles au niveau de la sémantique. Par exemple, l'OA définit le terme *Processeur-Performant* comme suit :

```

<OA : Processeur-Performant>
  <OP : Marque> Intel-Pentium </OP : Marque>
  <OP : Vitesse> > 1GHz </OP : Vitesse>
  <OP : Cache> > 1Mo </OP : Cache>
  <OP : NbdePro> > 1 </OP : NbdePro>
</OA : Processeur-Performant>

```

OB définit les termes *Processeur-De-Qualité-Elevée*, *Processeur-Multi*, *Processeur-rapide* comme suit :

```

< OB : Processeur-De-Qualité-Elevée>
  <OP : Marque> Intel-Pentium </OP : Marque>
  <OP : Vitesse> >1GHz </OP : Vitesse>
  <OP : Cache> >1Mo </OP : Cache>
</OB : Processeur-De-Qualité-Elevée>

```

```

<OB : Processeur-Multi>
  <OP : Marque> Intel-Pentium </OP : Marque>
  <OP : NbdePro> >1 </OP : NbdePro>
</OB : Processeur-Multi>

```

```

<OB : Processeur-Rapide>
  <OP : Vitesse> >1GHz </OP : Vitesse>
</OB : Processeur-Rapide>

```

L'établissement de la définition de *Processeur-Performant* dans l'OB nécessite des inférences sur les connaissances de l'OB, par exemple, la subsomption, la conjonction (basé sur l'OP partagée),etc. Ces services d'inférences permettent de découvrir les mêmes concepts portant des noms différents et

de définir un nouveau concept à partir de concepts plus génériques. Avec ces services, il existe deux candidats dans l'OB pour *Processeur-Performant* : *Processeur-De-Qualité-Elevée* et un nouveau concept défini par l'intersection entre *Processeur-Multi* et *Processeur-Rapide*.

Une solution plus générique est l'intégration des deux ontologies l'OA et l'OB en utilisant l'approche *assertionnelle*. Cette approche s'appuie sur les informations structurelles, par exemple, l'équivalence ou la subsomption des concepts, la transformation entre des concepts avec des niveaux abstraits différents. Il existe sur l'ontologie de la logique de description des algorithmes développés très récemment permettant des inférences non-standard. Ces inférences facilitent l'intégration assertionnelle de deux ontologies mais il nous reste beaucoup de travail à faire pour que les problèmes essentiels soient vraiment résolus.

Nous nous rendons compte facilement que ce scénario implique des échanges des données de façon transparente sémantique pour les acteurs. La nécessité de la sémantique formelle qui est affirmée par le premier scénario se justifie également dans le deuxième scénario. En effet, l'intégration d'ontologies exige de manipuler la sémantique formalisée pour découvrir la subsomption, l'équivalence entre les concepts ou transformer l'un en l'autre. Il existe actuellement deux approches pour le problème d'échange transparent sémantique. La première approche se fonde sur un vocabulaire unique avec la sémantique associée bien déterminée comme dans [7] et [8]. Le travail restant à faire est de développer un langage formel, qui porte cette sémantique, permettant de représenter presque toutes les connaissances commerciales. Alors tous les acteurs parlent la même langue avec la même base de connaissances pour décrire le monde, donc ils se comprennent. Cette approche fait face aux difficultés suivantes : formalisation des connaissances sur le nécessaire, le probable et le contingent ; adaptation du formalisme choisi à la nature de la représentation hybride de connaissances en conservant cohérence et efficacité ; la construction d'un vocabulaire unique à partir de vocabulaires divers existants, etc. La deuxième approche ne traite que les connaissances prévisibles, sans répudiation dans les transactions commerciales. Elle utilise plusieurs formalismes avec une sémantique formelle commune. Des services d'inférences doivent se développer pour manipuler les connaissances représentées et apparier des ontologies différents. Cette approche est confrontée également aux problèmes suivants : transformation de plusieurs formalismes vers un formalisme unifié portant la sémantique commune ; intégration des ontologies, etc.

En bref, nous avons identifié et analysé les problèmes posés par les spécifications ebXML pour arriver à proposer les directions de recherche appropriées. Nous mettons l'accent sur les idées principales par le résumé suivant :

- L'ebXML utilise l'UML (diagrammes de classes, d'états et d'activité), le lexique et l'ensemble de règles de production pour représenter des connaissances commerciales. Les spécifications ebXML ont proposé les

fonctionnalités exigeant des manipulations sur la sémantique de ces formalismes. Cela nous entraîne à procéder aux tâches suivantes :

- choisir et développer un formalisme générique, dont la sémantique est formelle, capable d'encapsuler les formalismes utilisés par l'ebXML.
- transformer la sémantique des diagrammes de classes UML vers celle de formalisme choisi.
- transformer la sémantique des diagrammes d'états et d'activités UML vers celle de formalisme choisi.
- concevoir l'ontologie correspond au lexique en utilisant le formalisme choisi.
- transformer la sémantique de l'ensemble de règles de production vers celle de formalisme choisi.

Ce problème et ces tâches sont inspirés du premier scénario d'échange.

- Le deuxième scénario implique le problème d'échange transparent sémantique. Cela nous entraîne à développer des techniques permettant d'intégrer les ontologies. Il est évident que les ontologies sont conçues par le formalisme choisi dans l'étape précédente.

3 Logique de description pour la sémantique de l'ebXML

Une description générale sur la DL est présentée dans [1]. Dans ce rapport nous allons aborder une extension existante de la DL, *ALCQI*, sur laquelle le formalisme proposé, *DLX*, sera développé. Ce formalisme est présenté dans l'espoir de pouvoir simplifier la formalisation de la sémantique hétérogène de l'ebXML en conservant l'essentiel de chaque formalisme utilisé. Nous allons également expliquer comment ce formalisme générique proposé avec sa sémantique formelle répond à la réalisation des fonctionnalités décrites par ebXML. Puisque le langage correspondant à la logique de description a une sémantique définie formellement, la formalisation d'une sémantique d'un formalisme utilisé par ebXML implique la représentation de cette sémantique en ce langage. Les sous-sections suivantes seront donc consacrées à décrire respectivement, *DLX* en tant que le formalisme générique, la transformation de diagrammes UML vers *DLX*, et finalement la transformation de règles de production vers *DLX*.

3.1 Syntaxe et sémantique de *ALCQI* - Formalisme *DLX*

Si l'on dénote *A*, *P*, *C*, et *R* respectivement, à un concept atomique, un rôle atomique, un concept arbitraire et un rôle arbitraire, le langage correspondant à *ALCQI* est défini comme suit :

$$\begin{aligned}
C, C' &\longrightarrow A \mid \neg C \mid C \sqcap C' \mid C \sqcup C' \mid \\
&\quad \forall R.C \mid \exists R.C \mid \exists^{\geq n} R.C \mid \exists^{\leq n} R.C \\
R &\longrightarrow P \mid P^-
\end{aligned}$$

Nous utilisons également les abréviations suivantes :

$$\begin{aligned}
\perp &\text{ pour } A \sqcap \neg A \\
\top &\text{ pour } A \sqcup \neg A \\
\exists^{\leq n} R.C &\text{ pour } \exists^{\geq n} R.C \sqcap \exists^{\leq n} R.C \\
\exists^{\geq n} R &\text{ pour } \exists^{\geq n} R.\top
\end{aligned}$$

La sémantique de \mathcal{ALCQI} est définie par l'interprétation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ où $\Delta^{\mathcal{I}}$ est un ensemble fini, non vide (domaine de \mathcal{I}) et $\cdot^{\mathcal{I}}$ est une fonction (fonction d'interprétation de \mathcal{I}). Si les ensembles des concepts et celui des rôles atomiques sont respectivement dénotés par \mathcal{A} et \mathcal{P} , les propriétés suivantes de l'interprétation \mathcal{I} sont vérifiées :

$$\begin{aligned}
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C_1 \sqcap C_2)^{\mathcal{I}} &= C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}} \\
(C_1 \sqcup C_2)^{\mathcal{I}} &= C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}} \\
(\forall R.C)^{\mathcal{I}} &= \{o \in \Delta^{\mathcal{I}} \mid \forall o'.(o, o') \in R^{\mathcal{I}} \rightarrow o' \in C^{\mathcal{I}}\} \\
(\exists R.C)^{\mathcal{I}} &= \{o \in \Delta^{\mathcal{I}} \mid \exists o'.(o, o') \in R^{\mathcal{I}} \wedge o' \in C^{\mathcal{I}}\} \\
(\exists^{\geq n} R.C)^{\mathcal{I}} &= \{o \in \Delta^{\mathcal{I}} \mid \#\{o' \mid (o, o') \in R^{\mathcal{I}} \wedge o' \in C^{\mathcal{I}}\} \geq n\} \\
(\exists^{\leq n} R.C)^{\mathcal{I}} &= \{o \in \Delta^{\mathcal{I}} \mid \#\{o' \mid (o, o') \in R^{\mathcal{I}} \wedge o' \in C^{\mathcal{I}}\} \leq n\} \\
(P^-)^{\mathcal{I}} &= \{(o, o') \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (o', o) \in P^{\mathcal{I}}\}
\end{aligned}$$

Le formalisme \mathcal{DLX} est défini en rajoutant les facteurs suivants à \mathcal{ALCQI} :

$$\begin{aligned}
(i : C)^{\mathcal{I}} &= \{t \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid t[i] \in C^{\mathcal{I}}\} \\
(\leq k[i]R)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \#\{t \in R^{\mathcal{I}} \mid t[i] = a\} \leq k\}
\end{aligned}$$

Un petit exemple a pour but d'expliquer quelques propriétés ci-dessus : supposons que le concept *Homme* et le rôle *enfant* sont définis. $\forall \text{enfant.Homme}$ signifie l'ensemble d'instances *Homme* dont tous les enfants sont les garçons. $\exists \text{enfant.Homme}$ signifie l'ensemble d'instances qui a au moins un garçon. $\exists^{\leq 2} \text{enfant}^-$ signifie quelqu'un qui a au plus deux parents. Une *base de connaissances* \mathcal{DLX} est un ensemble fini d'*assertions* sous la forme suivante :

$$\begin{aligned}
C_1 &\sqsubseteq C_2 \\
R_1 &\sqsubseteq R_2
\end{aligned}$$

Il existe les services d'inférences principaux sur la base \mathcal{DLX} : *validabilité de la base de connaissances* s'il existe un modèle (interprétation) pour la base de connaissances \mathcal{K} , *validabilité de concept* si \mathcal{K} a un modèle \mathcal{I} tel que $C^{\mathcal{I}} \neq \emptyset$, et *subsumption* si \mathcal{K} implique une assertion d'inclusion $C_1 \sqsubseteq C_2$ et pour toutes les interprétations $\mathcal{I} : C_1^{\mathcal{I}} \sqsubseteq C_2^{\mathcal{I}}$.

Exemple de la base \mathcal{K} pour *Core Component*

Dans CC, *produit* est défini comme une *Aggregate Information Entity* constituée de *Basic Information Entity* : *quantité* et *prix* comme ses attributs. Supposons que les concepts *Produit*, *Description*, *Quantité*, *Prix* existent dans la base \mathcal{K} partagée. Les relations (rôles) *attribut* entre *Produit* et *Quantité* : R_1 , *Produit* et *Prix* : R_2 sont définies en utilisant la notation présentée ci-dessus :

$$\begin{aligned} R_1 &\sqsubseteq (1 : \textit{Produit}) \sqcap (2 : \textit{Quantité}) \\ \textit{Produit} &\sqsubseteq ((\leq 1[1]R_1) \sqcap (\leq 1[1]R_1)) \end{aligned}$$

La dernière assure que pour chaque instance *Produit* il existe uniquement une instance *Quantité* par le biais R_1 .

$$\begin{aligned} R_2 &\sqsubseteq (1 : \textit{Produit}) \sqcap (2 : \textit{Prix}) \\ \textit{Produit} &\sqsubseteq ((\leq 1[1]R_2) \sqcap (\leq 1[1]R_2)) \end{aligned}$$

La dernière assure que pour chaque instance *Produit* il existe uniquement une instance *Prix* par le biais R_2 .

Maintenant, l'acteur A souhaite composer un document *commande* pour envoyer à l'acteur B. Ce document-là contient un *ProduitA* qui est *localement* défini sur A en rajoutant l'attribut *Description* pour décrire, par exemple la fragilité du produit :

$$\begin{aligned} \textit{ProduitA} &\sqsubseteq \textit{Produit} \\ R_3 &\sqsubseteq (1 : \textit{ProduitA}) \sqcap (2 : \textit{Description}) \\ \textit{ProduitA} &\sqsubseteq ((\leq 1[1]R_3) \sqcap (\leq 1[1]R_3)) \end{aligned}$$

Pour des raisons de simplicité, supposons qu'il n'existe pas d'ontologie locale sur l'acteur B. Même si le concept *ProduitA* n'est pas défini dans la base partagée, l'acteur B peut interpréter ce concept en s'appuyant sur les connaissances existantes. Effectivement, le service d'inférence *subsumption* de la base \mathcal{K} permettra de découvrir que le concept *ProduitA* est défini du *Produit*, donc il interprétera *ProduitA* comme un *Produit*. En outre, puisque le concept *Description* existe dans la base partagée, l'acteur B peut le traiter si nécessaire.

3.2 \mathcal{DLX} et diagrammes UML

Une classe de l'UML constituée de trois composants : identification (nom de classe), les attributs et les signatures des méthodes est naturellement représentée par un concept de \mathcal{DLX} . Dans ce rapport, nous ne traitons pas la transformation de ces signatures en \mathcal{DLX} . Nous définissons les assertions de même façon pour les *attributs* et les relations *agrégation* entre des classes

dans un diagramme de classes :

$$R \sqsubseteq (1 : C_1) \sqcap (2 : C_2)$$

$$C_1 \sqsubseteq ((\leq n_u[1]R) \sqcap (\leq n_v[1]R))$$

où C_1 contient C_2 ; n_u, n_v sont la multiplicité de l'agrégation R .

La transformation d'une association entre deux classes C_1, C_2 vers \mathcal{DLX} est plus sophistiquée car il nous faut capturer ses significations suivantes :

- i) définir un concept A pour l'association A et deux rôles r_1, r_2 pour les composants de A . Chaque rôle r_i a A comme le premier composant et C_i comme le deuxième.
- ii) assurer que le concept A possède exactement un composant de chaque r_i .
- iii) assurer que chaque instance du concept A détermine uniquement l'association correspondante.

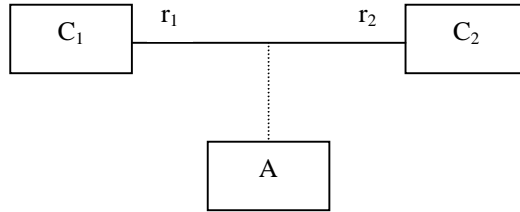


FIG. 5 – Association dans UML

Il faudra probablement rajouter de nouveaux facteurs à \mathcal{DLX} pour capturer toutes ces significations.

Pour ce qui concerne la généralisation, on peut le transformer en l'assertion suivante :

$$C_i \sqsubseteq C$$

Cependant, comment capturer la sémantique du mécanisme de l'héritage et du polymorphisme dans la relation de généralisation ? Nous ne traitons pas ce problème dans ce rapport !

Quant aux diagramme d'états UML, les spécifications ebXML exigent les manipulations suivantes sur les processus commerciaux : vérifier si un processus est équivalent à l'autre ou vérifier si l'un est inclus dans l'autre. La question sur la représentation sémantique des processus nous fait penser qu'il faudra améliorer l'expressivité de \mathcal{DLX} . En effet, en sachant que le diagramme d'états décrit des comportements d'une instance d'un concept, les états et les transitions du diagramme peuvent être définis respectivement comme un concept *état* et les *fonctions* d'un ensemble de concepts dans un autre concept. Ces fonctions permettent de déterminer la relation entre deux états consécutifs. Dans cette hypothèse, un diagramme d'états d'un concept C peut se traduire en un *transducteur* dans la théorie des automates.

Il est évident qu'une extension de \mathcal{DLX} en y rajoutant des définitions sur la *relation n-arité* et sur la *fonction* sera utile. Dans ce cas-là, nous espérons que des algorithmes développés sur les transducteurs permettent d'arriver aux spécifications mentionnées.

3.3 \mathcal{DLX} et règles de production

Les règles de production est un formalisme important pour la représentation de la sémantique ebXML car les informations contextuelles se traduisent sous la forme de ces règles. En plus, l'assemblage des documents commerciaux subit également une application de règles sur les composants de base. Une règle de production peut être représentée comme suit :

$$\begin{aligned}
 \text{regle} & ::= \langle \text{condition} \rangle \Rightarrow \langle \text{conclusion} \rangle \\
 \langle \text{condition} \rangle & ::= \langle \text{predicat} \rangle (\langle \text{variable} \rangle, \langle \text{constant} \rangle) \\
 \langle \text{conclusion} \rangle & ::= \langle \text{action} \rangle (\langle \text{variable} \rangle, \langle \text{constant} \rangle) \\
 \langle \text{predicat} \rangle & ::= \text{same} \mid \text{notsame} \mid \dots \\
 \langle \text{action} \rangle & ::= \text{add} \mid \text{remove} \mid \dots
 \end{aligned}$$

On rend compte facilement que la sémantique de cette représentation est capturée en \mathcal{DLX} sans difficulté sauf *action*. En effet, on peut considérer $\langle \text{condition} \rangle$, $\langle \text{conclusion} \rangle$, $\langle \text{variable} \rangle$ comme les *concepts*, et *predicat*, *same*, *notsame* comme les *rôles*. En outre, si l'on considère l'action *add* comme l'ajout d'un attribut C' à un concept C , cela est spécifié par l'assertion suivante :

$$R \sqsubseteq (1 : C) \sqcap (2 : C')$$

Cependant, on ne peut pas transformer de la même façon l'action *remove* en \mathcal{DLX} . Il existe des travaux, par exemple [17], qui traite de la transformation des règles Horn vers DL mais celles-ci ne contiennent pas le concept *action*. La formalisation de ce concept en DL exige donc plus de travail.

4 Conclusion

Ce rapport examine la capture et la représentation sémantique commerciale de l'ebXML. A travers cette étude, les formalismes avec ses problèmes sont identifiés . Ces problèmes se posent après avoir analysé les spécifications ebXML qui exigent des manipulations sur cette sémantique. Nous essayons de proposer un formalisme, nommé \mathcal{DLX} pour la formaliser. Ces premiers efforts ouvrent une nouvelle direction de recherche pour des problèmes traditionnels. Cependant, il nous reste beaucoup de travail à faire pour obtenir les résultats attendus. Pour cette raison, notre travail suivant va consacrer aux tâches principales comme suit : compléter la transformation des diagrammes de classes UML, représenter des règles de production en \mathcal{DLX} . Une extension

de \mathcal{DLX} sera également étudiée pour les diagrammes d'états UML.

Références

- [1] Chan Leduc. Modèles d'échange de données dans le commerce. *Rapport de recherche*, Laboratoire I3S, Université de Nice, 12-2001.
- [2] Diego Calvanese, Maurizio Lenzerini, Daniele Nardi. Description logics for conceptual data modeling. Dans *Logics for Databases and Information Systems*, Kluwer Academic Publisher, 1998.
- [3] James Rumbaugh, Ivar Jacobson, Grady Booch. *The Unified Modeling Language Reference Manual*, Addison Wesley Publ, 1999.
- [4] Peter Lucas, Linda Van Der Gaag. *Principles of Expert Systemes*, Addison Wesley Publ, 1991.
- [5] Bernhard Nebel. *Reasoning and Revision in Hybrid Representation Systems*, Springer-Verlag Publ, 1990.
- [6] Ralf Küsters. *Non-Standard Inferences in Description Logics*, Springer Publ, 2001.
- [7] Steven O.Kimbrough. *EDI, XML, and the Transparency Problem in Electronic Commerce*, University of Pennsylvania, 2000.
- [8] Steven O.Kimbrough. *Sketch of basic Theory for a formal Language for business communication*, University of Pennsylvania, 1998.
- [9] ebXML Technical Architecture Project Team. *ebXML Technical Architecture Specification v1.0.4*, <http://www.ebxml.org>, 16 February 2001.
- [10] ebXML Business Process Project Team. *ebXML Business Process Specification Schema, Version 1.01*, <http://www.ebxml.org>, 11 May 2001.
- [11] Business Process Team. *Business Process and Business Information Analysis Overview v1.0*, <http://www.ebxml.org>, 11 May 2001.
- [12] Business Process Team. *Business Process Analysis Worksheets and Guidelines v1.0*, <http://www.ebxml.org>, 11 May 2001.
- [13] ebXML Core Components. *Core Component Overview Version 1.05*, <http://www.ebxml.org>, 10 May 2001.
- [14] ebXML Core Components. *Core Component Discovery and Analysis*, <http://www.ebxml.org>, 10 May 2001.
- [15] ebXML Core Components. *Document Assembly and Context Rules*, <http://www.ebxml.org>, 10 May 2001.
- [16] ebXML Core Components. *Context and Re-Usability of Core Components*, <http://www.ebxml.org>, 10 May 2001.
- [17] Alon Y.Levy et Marie-Christine Rousset. *CARIN : A Representation Language Combining Horn rules and Description Logics*, Dept. of Computer Science of Paris Sud, 1996.

5 Annexe

| Form: Business Process Area | |
|------------------------------------|--|
| Forme Id | PA-3.2-Customer-Order-Fulfillment |
| Nom du type processus | Customer Order Fulfillment (gestion optimale des commandes) |
| | |
| Objectifs | Permettre au détaillant de conduire un vendeur direct pour délivrer (dans une durée spécifique) un produit à un consommateur spécifique |
| Contraintes | <ul style="list-style-type: none"> • Être prêt de distribuer immédiatement les produits à la commande de consommateur par un vendeur après le traitement de la commande d'achat d'un détaillant. • Livraison à temps des produits du vendeur au consommateur • Immédiat avis par un vendeur pour un détaillant sur la livraison des produits au consommateur; avec un détail de service de consommateurs. |
| Partenaires (Stakeholders) | <ul style="list-style-type: none"> • Détaillant • Vendeur direct (Direct Supply Retail Vendor: DSVendor) • Transport Métier • Consommateur |
| Processus commerciaux | <ul style="list-style-type: none"> • Gestion des commandes d'achat • Livraison de biens (shipment) = envoi |

FIG. 6 – Fiche 1 : Décrire le type de processus

| Form: Business Process Area | |
|------------------------------------|--|
| Forme Id | PA-3.1-Customer-Order-Management |
| Nom du type processus | Customer Order Management |
| Objectifs | <ul style="list-style-type: none"> • Prendre une commande d'un consommateur sur Internet • Valider la capacité d'un consommateur de payer pour un produit livré • Prendre le paiement de la carte crédite du consommateur après qu'un produit est été directement livré à un consommateur |
| Contraintes | <ul style="list-style-type: none"> • Promesse à un consommateur de la disponibilité des produits à l'emplacement du vendeur dès que la commande du consommateur est acceptée par le détaillant • Le consommateur doit avoir suffisamment du crédit permettant de payer pour le produit après que ce produit aura être livré. |

FIG. 7 – Fiche 1 bis : Décrire le type de processus

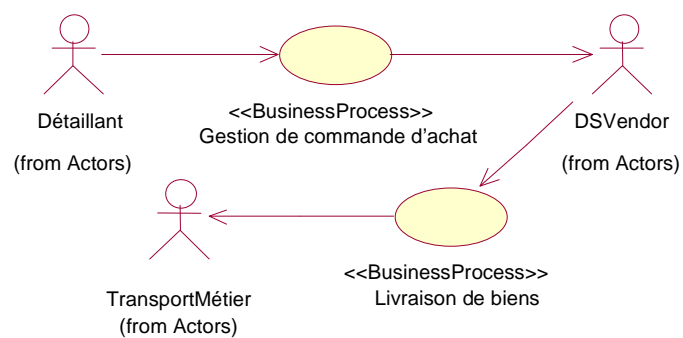


FIG. 8 – Diagramme de cas pour la collaboration Détaillant-DSVendeur

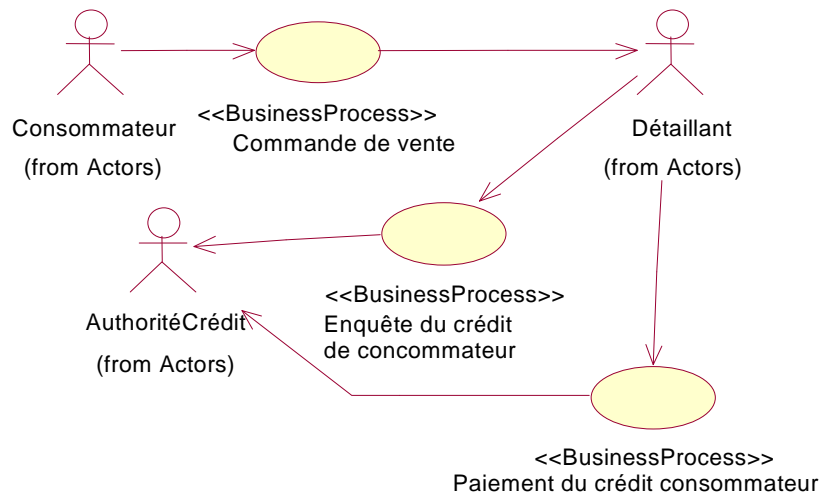


FIG. 9 – Diagramme de cas pour la collaboration des acteurs

| Form: Business Collaboration | |
|-------------------------------------|---|
| Forme Id | BC-6.4-Create-Vendor-Purchase-Order |
| Types Partenaires | <ul style="list-style-type: none"> • Détaillant • DSVendeur |
| Rôles autorisés | <ul style="list-style-type: none"> • Détaillant.InventaireAcheteur • DSVendeur. ServiceConsommateur |
| Pas légaux / exigences | Confirmer que PO reconnaissance implique un agrément de connexion entre un détaillant et DSVendeur, sur conditions d'une relation commerciale existante et une commande ouverte spécifique. |
| Etendue (scope) | Tester DSVendeur inventaire dans la main pour déterminer si une commande d'achat peut être acceptée ou rejetée. |
| Initial/terminal événements | <ul style="list-style-type: none"> • une commande de vente valide existe • une réponse d'une commande d'achat |
| Contraintes | La réponse doit être disponible dans 4 heures depuis la soumission de la requête PO |

FIG. 10 – Fiche 2 : Décrire la collaboration du processus commercial : Create-Vendor-Purchase-Order

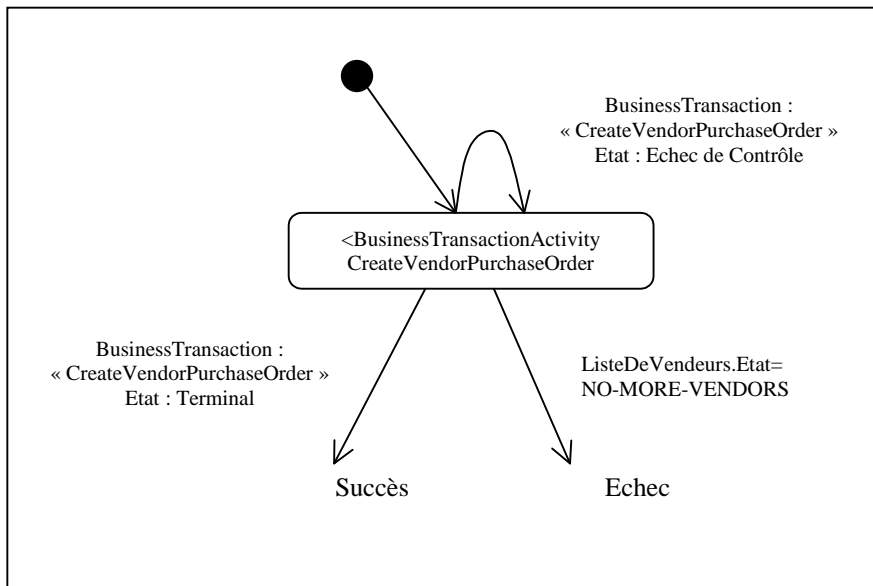


FIG. 11 – Diagramme d'état pour la collaboration

| Form: Business Transaction | |
|---------------------------------------|--|
| Forme Id | BT-8.4-Create-Vendor-Purchase-Order |
| Description | Une relation de produits de multi-vendeur / single, le détaillant a besoin d'envoyer au DSVendeur une commande d'achat REQUEST, qui aura besoin d'une réponse (avec des produits confirmés pour couvrir la PO) du DSVendeur. |
| Contraintes | Commande de vente valide Vérification crédit valide du consommateur |
| Type partenaire de requête | Détaillant |
| Rôle d'activité de requête | Acheteur inventaire (inventory buyer) |
| Document d'activité de requête | Requête de commande d'achat |
| Type partenaire de réponse | DSVendeur |
| Rôle d'activité de réponse | Vendeur |
| Document d'activité de requête | Reconnaissance de commande d'achat |

FIG. 12 – *Fiche 3 : Décrire la transaction : Create-Vendor-Purchase-Order*

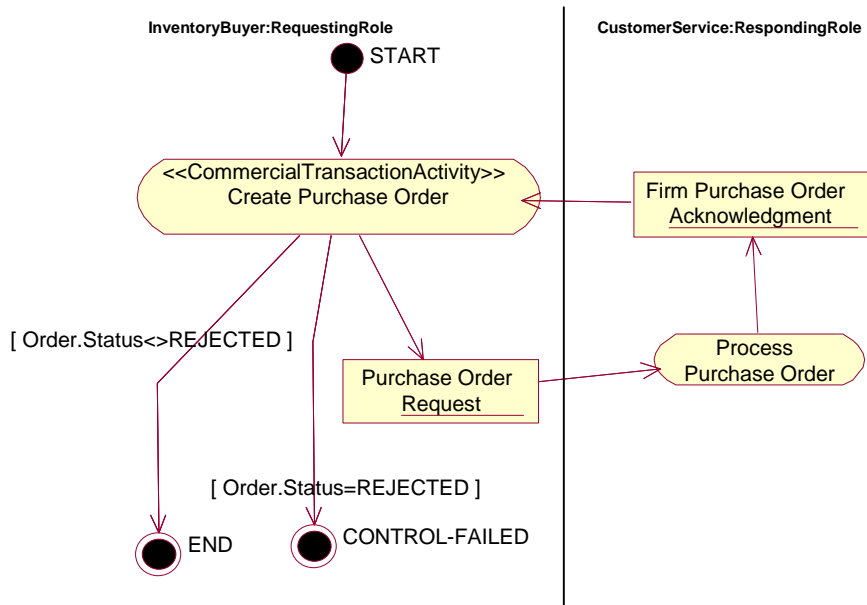


FIG. 13 – Diagramme d'activité pour la transaction

| Form: Business Information Context | |
|--|---|
| Forme Id: | BIC-10.1-Purchase-Order |
| Segment industriel | Vente au détail |
| Processus commerciaux | BPUC-5.4-Purchase-Order-Management BC-8.4-Create-Vendor-Purchase-Order |
| Physique Géographie /Conditions /Région | Amérique du nord |
| Rôle de partenaire | Acheteur inventaire Service de consommateurs |

FIG. 14 – Fiche 4 : Décrire le contexte du processus : PurchaseOrder

| Forme: Content Description | | | | | |
|----------------------------|--------------|-----------------|------------------|---|-------|
| Forme Id: | CD-9.1-Order | | | | |
| Element/Nom de Composant | Occurrence | Type de données | Largeur de champ | Description sémantique | Notes |
| Entête de commande | 1 | | N/A | L'entête de commande contient les informations d'entête de la commande | |
| Détail de commande | 0..1 | | N/A | Détail de commande contient la ligne d'articles et détails de la commande. | |
| Sommaire de commande | 0..1 | | N/A | Sommaire de commande contient le sommaire des informations de la commande, e.g la totalité de champs numériques | |

FIG. 15 – Fiche 5 : Description du document commercial

| Forme: Content Description | | | | | |
|-----------------------------------|----------------------|------------------------|-------------------------|--|--------------|
| Forme Id: | CD-9.2-Order-Summary | | | | |
| Élément/Nom de Composant | Occurrence | Type de données | Largeur de champ | Description Sémantique | Notes |
| Nombre de lignes | 0..1 | Entier | | Nombre de lignes identifie le nombre de lignes d'articles | |
| Totale Taxe | 0..1 | Valeur monétaire | N/A | Totale Taxe contient la quantité de taxe totale pour la commande. | |
| Totale quantité | 0..1 | Valeur monétaire | N/A | Totale quantité contient le prix total pour la commande entière. | |
| Transport Packaging Totalités | 0..1 | | | Transport Packaging Totalités est un sommaire de transport et package des informations si inclus dans la commande | |
| Sommaire Note | 0..1 | String | | Sommaire Note contient une forme de texte libre pour le sommaire de commande. Cet élément peut contenir les notes ou d'autres informations qui ne sont pas explicitement dans une autre structure. | |

FIG. 16 – *Fiche 5 bis : Description du document commercial*