Université de Montréal

# L'ingénierie de documents d'affaires dans le cadre du web sémantique

par
Jamel Eddine Jridi

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Thèse présentée à la Faculté des études supérieures
en vue de l'obtention du grade de Philosophiæ Doctor (Ph.D.)
en informatique

Juin, 2014

## RÉSUMÉ

Dans cette thèse, nous présentons les problèmes d'échange de documents et proposons une méthode pour y remédier.

XML a été utilisé pour définir les formats d'échange entre les applications de e-business. Les formats d'échange ebXML, xCBL, RosettaNet et cXML ont été développés pour la gestion des documents d'entreprise en utilisant des technologies XML. XML fournit la flexibilité nécessaire pour créer, manipuler, modéliser et organiser l'information d'un point de vue syntaxique, mais ne traite pas la sémantique des données. Nous discutons certaines limites de XML ainsi que l'apport du web sémantique pour y remédier.

Dans cette thèse, nous proposons une méthodologie pour adapter les standards d'affaire basés sur XML aux technologies du Web sémantique en utilisant la transformation des documents définis en DTD ou XML Schema vers une représentation ontologique en OWL 2. L'objectif est de rendre le transfert plus transparent et de récolter les fruits des efforts mis dans le développement de normes d'affaires basées sur XML.

Les ontologies générées par le processus de tranformation ne profitent pas de certaines relations sémantiques importantes en OWL comme `subClassOf`. Nous proposons une approche basée sur l'analyse formelle de concepts pour améliorer l'organisation de l'arbre d'héritage d'une ontologie OWL 2 en regroupant les classes ayant un comportement similaire et partageant des propriétés pour améliorer la qualité et la lisibilité d'une ontologie.

Les ontologies représentent les connaissances et les relations entre les concepts d'un domaine pour faciliter l'interopérabilité sémantique entre les partenaires, ce qui est plus simple lorsque les partenaires partagent la même ontologie. Mais il semble presque impossible de convaincre toutes les entreprises de changer leur modèle de document pour un modèle commun étant donnés l'effort et le coût dépensés dans leur développement.

Ces standards ayant été développés avec des vocabulaires différents, les ontologies générées par le processus de transformation ne sont pas nécessairement compatibles même s'ils traitent du même domaine d'intérêt. L'interopérabilité et la communication

entre les partenaires d'affaires se compliquent à cause de différents types d'hétérogénéité entre les ontologies des standards d'affaires.

L'alignement d'ontologies permet dans une certaine mesure de surmonter le problème de l'hétérogénéité sémantique en essayant de trouver des correspondances entre les entités (classes, propriétés, instances, etc ) de deux ontologies. Nous tirons avantage des recherches dans ce domaine et montrons comment ces techniques d'alignement aident les entreprises à mieux communiquer en trouvant des correspondances entre leurs ontologies d'affaires respectives. À titre d'exemple, nous appliquons trois techniques d'alignement sur des ontologies d'affaires qui traitent la gestion de bons de commande dans le cadre des standards cXML, xCBL et RosettaNet afin de déterminer des liens sémantiques entre elles. Nous avons effectué deux expériences avec plusieurs méthodes d'alignement (`Levenshtein`, `TFIDF` et l'algorithme $\texttt{OLA}_2$) qui se basent sur la structure, les noms et les commentaires des entités. Les résultats d'alignement sont comparés avec des alignements produits manuellement en utilisant les métriques de précision, rappel et F-score.

À partir des résultats d'alignement obtenus par la technique `TFIDF`, nous constatons que ces ontologies sont similaires en partageant les termes les plus fréquents et discriminants, ce qui est logique puisqu'elles traitent du même domaine d'intérêt. Les résultats d'alignement de cXML et xCBL avec RosettaNet PIP3A4 sont prometteurs et l'algorithme $\texttt{OLA}_2$ a donné la meilleure performance avec la plus grande valeur de F-score par rapport aux méthodes terminologiques.

**Mots clés: ingénierie de documents, web sémantique, standards d'affaire, alignement d'ontologies, analyse formelle de concepts.**

## ABSTRACT

In this thesis, we present the document exchange problems and the need for methods for document engineering.

XML which identifies data by tagging it has been used for defining formats for data exchange between e-business applications. The exchange formats such as ebXML, xCBL, RosettaNet and cXML have been developed for the management of business documents using XML technologies. XML provides the flexibility to create, manipulate, model and organize information, but only from a syntactical point of view and does not deal with the semantics of the data. We discuss some limits of XML and present how semantic web technologies can overcome some of them.

In this thesis, we propose a methodology for adapting B2B standards to semantic web technologies based on an automatic mapping of their documents written in DTD or XML Schema to an OWL 2 ontological representation. The goal is to make the transfer more transparent and reap the benefits of the effort put in the development of XML business standards.

As the generated ontologies do not benefit from important OWL semantic relations like `subClassOf`, we propose an approach based on Formal Concept Analysis to regroup ontology classes sharing a set of properties (Data and Object Property) to improve the quality, readability and the inheritance richness of a flat ontology.

Ontologies represent knowledge and relationships between concepts of a domain to facilitate semantic interoperability between partners, which is simpler when partners share the same ontology. But it seems close to impossible to convince all organisations to change their document for a common model given the effort and cost spent in the developement of their actual model.

These standards having been developed independently with their own vocabulary, ontologies generated by the transformation process are not necessarily compatible even if they deal with the same field of interest. Interoperability between businesses becomes more difficult due to different types of heterogeneity between ontologies.

Ontology alignment helps overcome the problem of semantic heterogeneity by match-

ing entities (classes, properties, instances, etc.) of two ontologies. We benfit from research in this area and show how these alignment techniques can help businesses to communicate fruitfully and overcome the hetereogeneity problem by finding correspondences between their ontologies. A case study will be presented using three techniques to align business ontologies describing purchase orders within cXML, xCBL and RosettaNet standards. We conducted two experiments with several alignment methods (`Levenshtein`, `TFIDF` and $\texttt{OLA}_2$ algorithm) based on the structure, names and comments of entities. The alignment results are compared with alignments generated manually using precision, recall and F-score.

Using `TFIDF`, we find that these ontologies are similar by sharing the most common and discriminant terms, which makes sense as these ontologies deal with the same area of interest. The result of business standards alignement are promising and the $\texttt{OLA}_2$ algorithm gives the best performance for all ontologies with a highest F-score compared to terminological methods.

**Keywords: Document Engineering, Semantic web, business standard, Ontology matching, Formal Concept Analysis.**

## TABLE DES MATIÈRES

## LISTE DES FIGURES

## LISTE DES TABLEAUX

## LISTE DES ANNEXES

## LISTE DES SIGLES

| | | Page |
|---|---|---|
| ASMOV | Automated Semantic Mapping of Ontologies with Validation | 101 |
| BP | Business Process | 43 |
| CDC | Core Data Component | 43 |
| CPA | Collaboration Protocol Agreement | 43 |
| CPP | Collaboration Protocol Profile | 43 |
| cXML | commerce eXtensible Markup Language | 41 |
| DTD | Document Type Definition | 23 |
| EDI | Electronic Data Interchange | 35 |
| ebRIM | electronic business Registry Information Model | 44 |
| ebRS | electronic business Registry Specification | 44 |
| ebXML | electronic business XML | 45 |
| FCA | Formal Concept Analysis | 75 |
| IDF | Inverse Document Frequency | 96 |
| MIME | Multipurpose Internet Mail Extensions | 41 |
| OLA | Ontology Lite Alignement | 99 |
| OAEI | Ontology Alignment Evaluation Initiative | 98 |
| OASIS | Organization for the Advancement of Structured Information Standards | 98 |
| OWL | Web Ontology Language | 30 |
| PIP | Partner Interface Process | 39 |
| RDF | Resource Document Format | 27 |
| RNIF | RosettaNet Implementation Framework | 41 |
| SGML | Standard Generalized Markup Language | 19 |
| TDCC | US Transport Data Coordinating Committee | 36 |
| TF | Term Frequency | 96 |
| TFIDF | Term Frequency Inverse Document Frequency | 96 |

| | | |
|---|---|---|
| UML | Unified Modeling Language | 2 |
| UN/CEFACT | United Nations/Centre for Trade Facilitation and Electronic Business | 36 |
| URI | Uniform Resource Identifier | 27 |
| WSML | Web Service Modeling Language | 3 |
| WSMO | Web Service Modeling Ontology | 53 |
| W3C | World Wide Web Consortium | 3 |
| XML | eXtensible Markup Language | 19 |
| XSLT | eXtensible Stylesheet Language Transformations | 22 |
| xCBL | XML Common Business Library | 38 |

# CHAPITRE 1

# INTRODUCTION

Dans ce chapitre, nous présentons le contexte général de notre travail. Nous évoquons les problématiques au niveau de l'échange de documents et l'importance d'avoir des méthodes pour aider à la gestion de documents. Ensuite, nous définissons l'objectif de notre thèse ainsi que les différentes contributions. Enfin, nous présentons la structure de notre thèse.

## 1.1 Contexte général

De nos jours, il existe plusieurs types de documents ayant différents contextes et structures. La circulation et l'échange de documents sont très importants, ce qui facilite la communication entre les parties d'un processus dans une entreprise, dans un organisme ou même sur internet. Dans le cadre d'une entreprise, nous pouvons échanger des documents de différentes natures, par exemple ; des relevés de transactions, des formulaires, des contrats, etc. L'apparition et l'évolution des entreprises virtuelles est remarquable sans avoir besoin de personnel ni d'espace physique. Une grande partie des affaires traitées sur le Web aujourd'hui a lieu à travers des échanges d'information rendus possibles par l'utilisation de documents comme interface. Cette évolution est due à l'évolution des techniques offertes par les services web. Mais l'échange de ces documents peut causer des problèmes liés à la compréhension du sens des mots dans les documents échangés. Le sens d'un mot pourrait changer selon les situations ou contextes. Par exemple, le mot `adresse` peut désigner l'adresse du vendeur dans un document et l'adresse de l'acheteur dans un autre. La figure 1.1 montre l'ambiguïté au niveau des champs `Reference`, dans les documents échangés entre des partenaires du processus de vente de livres. Nous remarquons que dans les différents services qui gèrent la vente en ligne sur un site fictif GMBooks.com : le champ référence de la commande est décrit par des termes ayant des sémantiques différentes :

- `Order Reference` signifie le numéro de la commande.
- `Customer Reference` désigne l'identifiant du client sur le site (p.e. nom d'utilisateur, etc), par contre la valeur de ce champ fait référence au numéro de la commande, et
- `Details` peut contenir une description textuelle de la commande mais pas forcément son numéro.

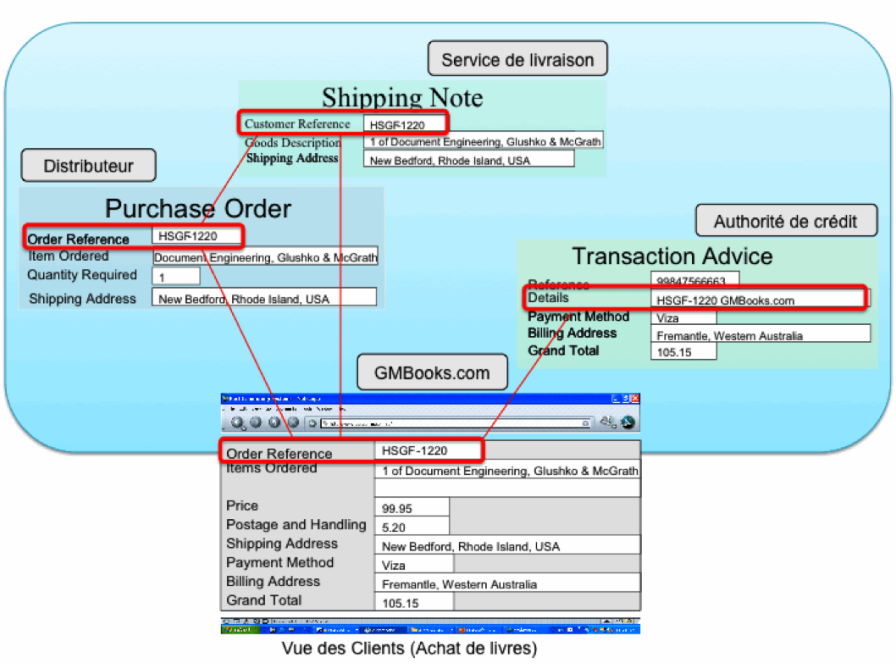



Figure 1.1 – L'ambiguïté entre les partenaires d'un processus de vente des livres [18].

Pour y remédier, il faut que chaque partenaire utilise les mêmes définitions des informations dans leurs documents, ce qui est difficile. Il est plus pratique que chacun utilise un système permettant de transformer la représentation de l'information reçue dans une autre représentation compréhensible par l'entreprise. Donc, il faut avoir une façon plus précise et adéquate au moment de la définition de contenu des documents. Le but est de déterminer comment l'information et les données sont définies et utilisées, afin de concevoir un modèle décrivant la structure et la sémantique des concepts utilisés. Parmi les solutions proposées, on retrouve : un dictionnaire de données, une base de données, un diagramme de classe UML, ou un schéma de documents en XML, etc.

Toutes ces recherches font partie du domaine de l'ingénierie des documents, nécessaire pour analyser, concevoir et mettre en œuvre les échanges d'information sur Internet. Elle est basée sur les méthodes d'analyse et de conception de données permettant d'avoir un modèle précis décrivant les informations.

XML et les services Web fournissent une grande flexibilité pour créer, manipuler, modéliser, organiser et partager les informations. Ces techniques ne nous donnent toutefois pas un point de vue sur la sémantique des documents. Il nous faut donc penser aux documents d'une façon plus abstraite et utiliser d'autres techniques, par exemple basées sur le web sémantique.

Dans cette thèse, nous nous intéressons à l'ingénierie des documents dans le cadre du web sémantique. Ces documents sont de diverses natures et présentés différemment. Nous concevrons donc une méthode permettant de gérer ces échanges de documents, en utilisant les techniques du web sémantique.

## 1.2 Contribution

L'objectif de notre thèse est d'intégrer la sémantique et de réduire l'hétérogénéité dans les échanges électroniques dans le domaine des affaires ainsi que de donner à la machine la possibilité d'interpréter les données. L'objectif de l'ingénierie de documents dans le domaine des affaires est d'avoir un modèle de documents générique qui gère les affaires inter-entreprises. Actuellement, il existe plusieurs standards d'affaire, basés sur XML, qui ont été développés indépendamment par différents organismes, p.e. OASIS.

Les limites du langage XML ouvrent la voie à plusieurs recherches d'intégration de la sémantique dans ces standards. À notre connaissance, il n'existe pas, à date, d'ontologie modélisant ces standards. Il y a des intitiatives qui se basent sur des langages non recommandés par le W3C, p.e. WSML, ou bien des ontologies construites manuellement sans tirer parti des efforts mis dans la modélisation XML de ces standards.

Notre première contribution est la proposition d'une feuille de transformation XSLT, nommée XSD2OWL, qui transforme les documents des standards d'affaire définis en XML Schema vers une représentation ontologique en tenant compte des efforts mis

dans la modélisation XML. Le résultat de l'application des règles de transformation de notre système XSD2OWL est une ontologie OWL permettant de rendre la sémantique du schéma XML plus explicite.

Notre deuxième contribution est basée sur l'utilisation de l'analyse formelle de concepts pour la détection des groupes de classes partageant un certain nombre de propriétés afin d'améliorer la représentation de la hiérarchie de classes des ontologies générées par notre processus de transformation.

L'utilisation des ontologies ne signifie pas que le problème d'hétérogénéité a été résolu puisque les ontologies ne sont pas nécessairement compatibles avec les mêmes définitions de concepts. Notre dernière contribution consiste à prendre avantage des recherches effectuées dans le domaine d'alignement des ontologies. Nous utilisons différentes méthodes d'alignement, terminologique, structurelle et hybride, afin de voir jusqu'à quel point nous pouvons réduire l'hétérogéneité entre les ontologies des standards d'affaire. Nous avons appliqué cette méthodologie à l'ensemble des schémas des documents qui gèrent les bons de commandes des standards xCBL, cXML et RosettaNet, décrit au chapitre 4. Cette tâche est considérée comme l'activité la plus importante dans la chaîne d'approvisionnement. Les résultats obtenus sont prometteurs et ouvrent la voie à d'autres recherches pour l'intégration de la sémantique dans le domaine des affaires.

## 1.3 Structure de la thèse

Cette thèse est organisée comme suit. Au chapitre 2, nous définissons la discipline de l'ingénierie de documents avec quelques domaines d'application pour en présenter les avantages surtout au niveau de l'organisation des documents. Enfin, nous en étudions le processus ainsi que les différentes méthodes d'analyse et de conception.

Le chapitre 3 introduit le formalisme XML et ses avantages, le domaine du web sémantique ainsi que ses avantages dans le domaine de la gestion des processus d'affaires.

Au chapitre 4, nous présentons un bref historique de l'évolution des techniques d'échange électronique d'information entre les entreprises jusqu'à l'arrivée du langage XML qui a aidé à la structuration des documents d'affaire avec plusieurs normes d'af-

faire, p.e. xCBL [1], cXML [2] et RosettaNet [41]. Ensuite, nous citons les avantages et les limites de ces normes d'affaire qui sont basées sur XML pour l'organisation des affaires inter-entreprises. Par la suite, nous présentons l'approche globale de cette thèse et les solutions proposées pour remédier le probléme de ces standards lié à l'hétérogénéité dans les échanges de documents entre les entreprises.

Au chapitre 5, nous détaillons notre approche d'intégration de la sémantique dans les standards d'affaire. Cette approche permet de transformer ces standards implantés en XML Schema et en DTD vers une représentation ontologique [27]. En plus, nous avons étudié les approches proposées pour l'intégration de la sémantique ainsi que leurs limites. Une étude de cas est réalisée pour illustrer le processus de transformation.

Malgré l'utilisation des ontologies, le problème d'hétérogénéité dans les échanges n'a pas été résolu. Au chapitre 6, nous étudions les types d'hétérogénéité existant entre les ontologies des standards d'affaire au niveau de la gestion de bons de commande dans la chaîne d'approvisonnement ainsi que les solutions apportées par l'alignement d'ontologies, dont nous détaillons quelques techniques ainsi que son apport pour réduire l'hétérogénéité dans les échanges. Nous n'avons pas développé une nouvelle approche d'alignement, nous cherchons plutôt à tirer avantage des recherches dans ce domaine afin d'identifier des correspondances entre plusieurs ontologies d'affaires hétérogènes et ainsi aider les entreprises à mieux communiquer même si leurs modèles de documents diffèrent.

Nous récapitulons, au chapitre 7, les principales idées introduites dans notre thèse et nous proposons certaines perspectives. Enfin, nous avons défini les concepts de base étudiés dans cette thèse dans l'annexe I du manuscrit.

# CHAPITRE 2

# L'INGÉNIERIE DE DOCUMENTS

## 2.1 Introduction

Dans ce chapitre, nous définissons la discipline de l'ingénierie de documents ainsi que les domaines d'application. Ensuite, nous citons les avantages et les solutions apportées par cette discipline. Par la suite, nous étudions les différents types d'analyse formant les phases du processus d'ingénierie de documents. Nous utilisons un exemple fictif mais réaliste pour décrire le déroulement de l'approche. Cette partie est inspirée du livre de Glushko et al. [18].

## 2.2 Définition et domaine d'application

L'ingénierie de documents est une nouvelle discipline pour la spécification, la conception et la mise en œuvre des documents échangés électroniquement [18]. Cette discipline est appliquée dans plusieurs domaines afin de faciliter la communication et gérer la grande masse d'informations qui circulent dans/et en dehors d'un organisme. Elle est utilisée dans plusieurs secteurs dont celui de la gestion des affaires ou du domaine médical, etc. Dans le domaine des affaires, le but est d'une part, de gérer le processus des échanges et d'autre part, d'identifier et de classifier les types de documents, p.e. les bons de livraison, les contrats, les rapports des employés, les ventes, etc. Par contre dans le domaine médical, les documents peuvent être des rapports de médecins, des ordonnances médicales, des résultats d'analyses, etc.

Pour ce faire, l'ingénierie de documents se base sur des méthodes d'analyse et de conception permettant de construire des modèles formels pour décrire l'information selon un domaine d'application particulier. Les méthodes d'analyse de documents sont effectuées dans le but de trouver un modèle abstrait et logique à partir des instances existantes d'un seul type de document, et ensuite d'encoder le modèle en utilisant des techniques comme XML. Un meilleur modèle, décrivant une classe de documents, est

celui qui répond le mieux aux exigences des utilisateurs actuels et potentiels pour la réalisation de tâches spécifiques avec de nouvelles instances.

Selon Robert J.Glushko et Tim Mc Grath, l'utilisation des patterns réutilisables pour la gestion de processus des affaires rend la sémantique plus précise. Ceci n'est pas le cas dans le cadre d'échange des documents entre différentes entreprises, puisque chaque entreprise peut avoir un modèle personnel pour gérer ces documents. Par contre pour remédier à ce problème, il faut avoir un modèle d'échange de documents génériques qui gère la communication entre les entreprises afin de conserver la sémantique des documents, ce qui est très difficile.

## 2.3 Les avantages de l'ingénierie de documents

Dans les organisations, une grande variété de documents circulent à l'intérieur ou à l'extérieur p.e. les feuilles de temps, les bilans de dépenses, les rapports, les bons de commande, les catalogues, et d'autres types de documents. Ces documents sont de différentes natures : narratives comme les manuels d'instructions, ou transactionnelles comme les formulaires ou les relevés de transactions.

Le processus des affaires est géré par l'échange des documents qui sont utilisés comme des interfaces de communication. Il est possible d'avoir des chevauchements dans la sémantique des composants formant ces documents. Ce qui provoque une perte de temps et des dépenses afin de comprendre ces informations.

L'ingénierie de documents est une solution efficace. Elle trouve une façon pour modéliser, décrire l'information, et résoudre les problèmes liés à la sémantique. Dans un processus d'affaire, il y a au moins deux côtés à chaque échange de documents, et toutes les parties doivent s'assurer qu'ils comprennent les documents de la même manière.

L'ingénierie de documents construit un modèle de processus des affaires et de documents échangés. Cette approche est essentielle parce que les documents narratifs et transactionnels sont souvent étroitement liés, par exemple, la relation entre les formulaires d'immigration et les instructions pour les remplir, ou entre les brochures et les bons de commande.

## 2.4 Les phases de l'approche

L'ingénierie de documents s'appuie sur des méthodes d'analyse et de conception permettant de construire des modèles formels pour décrire l'information selon un domaine d'application particulier. Ces types d'analyse sont :

- L'analyse du processus d'affaire décrit le contexte général du processus, pour comprendre la sémantique des informations échangées ainsi que leurs relations.
- L'analyse des tâches identifie les différentes étapes et les informations réalisant une tâche donnée.
- L'analyse des documents commence par l'analyse des instances de documents. Cette technique extrait l'information structurelle, la représentation des données, le contenu de document, etc.
- L'analyse des données traite de façon abstraite les composantes des données, révélées par l'analyse de documents, et présente le domaine d'un point de vue conceptuel.

Ces méthodes d'analyse de documents sont effectuées dans le but de trouver un modèle abstrait et logique à partir des instances existantes d'un seul type de document, et ensuite d'encoder le modèle en utilisant des techniques comme XML. Un meilleur modèle, décrivant une classe de documents, est celui qui répond le mieux aux exigences des utilisateurs actuels et potentiels pour la réalisation de tâches spécifiques avec de nouvelles instances. Le processus d'ingénierie de documents est composé de plusieurs phases :

1. L'analyse du contexte d'utilisation.
2. L'extraction des patterns par l'analyse du processus d'affaires.
3. L'analyse des documents et de leurs composants.
4. La conception des modèles de documents (Phase d'assemblage).
5. L'implémentation des modèles de processus et de documents (encodage en XML).

Pour illustrer le déroulement du processus, nous utilisons un scénario d'une vente en ligne illustré dans la figure 2.1. Une entreprise envoie ses promotions à différents sites web qui gèrent la vente de ses produits en ligne. Ces sites reçoivent plusieurs types de produits, de marques différentes. Ces sites web sont utilisés comme une vitrine pour afficher les promotions reçues. Cette stratégie est utilisée pour améliorer la visibilité des produits et augmenter le chiffre d'affaire. Le gestionnaire des sites web gagne un pourcentage des ventes p.e. l'entreprise donne au site un produit à la valeur de 20$ et ce dernier le revend à 25$. Les clients commandent les produits à partir de ces sites web. Le gestionnaire du site traite la commande, vérifie les fonds du client et prend une pré-autorisation sur le montant de la commande. Ensuite, il envoie un bon de commande à l'entreprise qui prend en charge la préparation et la livraison des produits. Au moment de la livraison, cette entreprise informe le gestionnaire du site des détails de la livraison et lui autorise le retrait d'argent. Plusieurs organismes participent au processus de vente des produits matérialisé par un échange de documents entre les gestionnaires des sites et l'entreprise : bons de commande, des relevés de transactions, etc. Le problème majeur est lié à la différence de structure, de contenu et de sémantique des éléments dans les documents échangés, d'où l'importance d'une stratégie de gestion de ces documents. Dans le reste de ce chapitre, nous illustrons le processus de gestion de documents avec cet exemple.

### 2.4.1 L'analyse du contexte d'utilisation

Il faut comprendre les activités principales du processus d'affaire ainsi que les membres participant à ces activités, analyser le contexte d'utilisation et identifier les exigences [18]. L'identification des exigences dans les phases initiales d'un projet est très importante. Elle réduit la probabilité de refaire l'analyse, la conception et la mise en œuvre des étapes. Les exigences pour les documents et les processus d'affaires doivent être précises et vérifiables pour être utiles. Une façon d'y arriver est de les exprimer comme des règles (ou contraintes). Ces exigences seront exprimées avec des règles sur le contenu, la structure et la présentation des documents et de leurs composants. Ces règles formalisent les définitions du contexte dans lequel les documents sont utilisés, et les patterns décrivant

les activités principales dans un processus d'affaire. Un pattern est un modèle, abstrait générique et réutilisable, qui peut être adapté à un ensemble d'instances ou contextes. La figure 2.1 décrit le contexte général d'un processus de vente en ligne, dans lequel, nous avons différents participants : le client, le détaillant (le site web), le livreur, le distributeur (l'entreprise), etc.

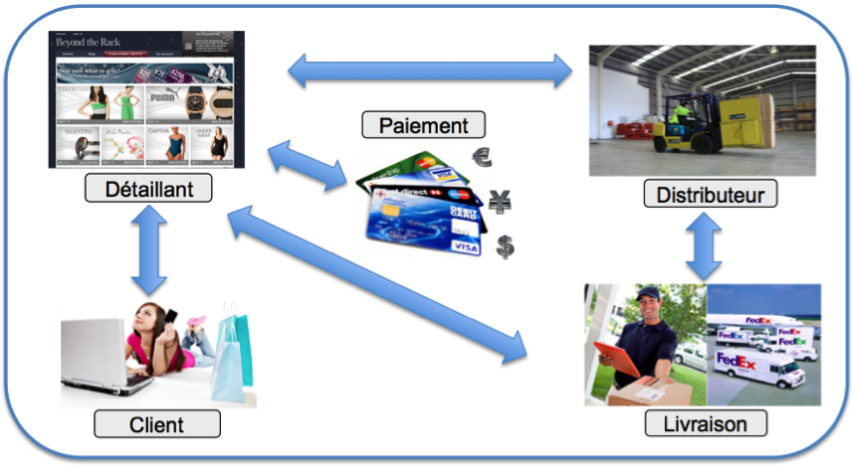



Figure 2.1 – Exemple des sites web qui gèrent la vente et les promotions de produits. Ainsi que la communication entre les différentes parties de ce processus.

### 2.4.2 L'extraction des patterns par l'analyse du processus d'affaire

Cette étape modélise les processus d'affaire dans une entreprise, à plusieurs niveaux d'abstraction et de granularité. Un processus d'affaire est une chaîne d'activités ou d'événements à réaliser dans un ordre chronologique. Cette chaîne peut produire des résultats qui peuvent être l'entrée d'un autre processus d'affaire. Il existe deux types de processus d'affaire : les processus internes (ou privés) réalisés entièrement dans une même entreprise, et les processus externes (de collaboration, ou d'affaires publiques) sont effectués entre deux ou plusieurs parties d'affaire. La compréhension d'un processus se base sur un ensemble de questions touchant le fonctionnement d'un processus, son but, etc. Par souci de cohérence, nous répondons aux mêmes questions pour chaque processus. Le tableau 2.I montre les questions à poser ainsi que les réponses possibles dans le cadre de la vente en ligne. Ces questions et leurs réponses forment une base d'information sur les processus interconnectés à partir de laquelle nous pouvons développer

des modèles.

Ces données nous aident à construire le modèle du processus d'affaire pour la vente en ligne. Ce processus est composé de deux parties : la première partie concerne la réalisation de la commande, par contre la deuxième est destinée à la livraison du produit. Il est possible d'avoir un modèle pour la réalisation de la commande, et un autre pour la livraison. Ces modèles peuvent être décrits par les diagrammes UML, comme le diagramme de séquence pour montrer le scénario d'exécution d'une transaction particulière.

Tableau 2.I – Description du processus de vente en ligne.

| **Questions** | **Réponses** |
|---|---|
| Quel est le nom du processus ? | Le processus de vente en ligne. |
| Quel est l'objectif de ce processus ? | L'objectif est de gérer la vente de plusieurs types de produits. Ce processus est composé de plusieurs sous étapes :<br>• Le choix des produits et paiements.<br>• Traitement de la commande.<br>• La livraison de la commande. |
| Quelles sont les industries, les organisations impliquées dans le processus ? | • Les services de cartes de crédit.<br>• Le fournisseur.<br>• Les sociétés de livraison. |
| Quelles sont les intervenants ou les participants au processus ? | • Le client comme acteur principal.<br>• Le gestionnaire du site.<br>• Le distributeur (le fournisseur de l'entreprise)<br>• Le livreur (FedEx, UPS, etc.) |

L'échange ou la réalisation de ces processus se fait en utilisant des documents qui sont utilisés comme des interfaces ou des conteneurs d'information [18]. L'analyse de ces processus permet une connaissance commune et partagée de leur fonctionnement entre les différents partenaires d'affaire. Le plus important est d'avoir un modèle unique

qui gère l'échange de documents entre les partenaires, c.-à-d. il est possible d'avoir des changements au niveau de processus interne mais c'est plus difficile pour ceux qui sont externes.

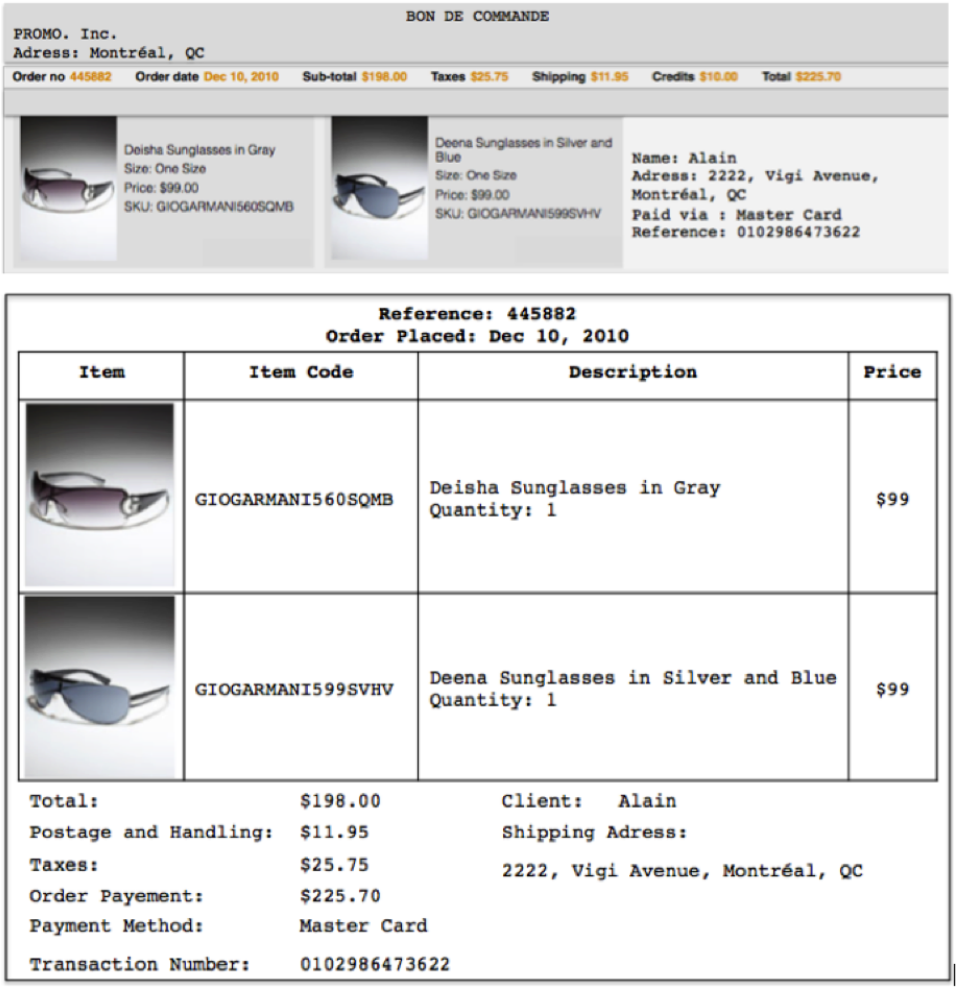
### 2.4.3 L'analyse de documents et de ces composants

Les documents sont les parties les plus visibles d'un processus d'affaire, nous devons donc les identifier et les comprendre. Nous voulons étudier les rôles joués par les documents utilisés comme des interfaces et des supports de communication entre une ou plusieurs parties. L'analyse des documents se base sur plusieurs critères : le contenu, la structure, la présentation, la syntaxe et la sémantique des informations.

Cette phase identifie les composantes sémantiques ou concepts contenus dans les documents sélectionnés, afin de créer une vue conceptuelle cohérente nommée `document component model`. Cette analyse est faite en utilisant des techniques d'analyse de données qui normalisent les composantes dans des structures en fonction de leur dépendance fonctionnelle (sémantique commune). Dans l'exemple du processus de vente en ligne, plusieurs documents sont échangés, dont différents types de relevé de transaction d'achat, des bons de livraison et des bons de commande. Dans notre exemple du processus de vente, l'entreprise reçoit des bons de commande de différents sites web qui gèrent les promotions et la vente de ses produits. La figure 2.2 montre deux exemples de bons de commande envoyés depuis les organismes des sites à l'entreprise. Nous remarquons l'existence de composants d'information ayant une sémantique commune même si les documents viennent de différentes sources. Mais il existe aussi un chevauchement dans la sémantique de certains composants d'information existants, citons par exemple :

- `Reference` dans le premier document représente le numéro de transaction d'achat, mais celui de la commande dans le deuxième document.
- `Total` dans le premier document représente le montant total de la transaction après l'ajout des taxes, par contre dans le deuxième document, il signifie le total avant l'ajout des taxes.

Les bonnes règles de nommage sont importantes dans notre modélisation. Des noms significatifs aident non seulement à trouver une sémantique commune entre les composants, mais aident également dans l'analyse et la conception de composants de documents réutilisables. C'est le cas des champs `Shipping` et `Postage and Handling` qui signifient à la fois le montant de livraison à payer. Aussi les champs `Order date` et `Order Placed` signifiant la date de la réalisation de la commande.



BON DE COMMANDE

PROMO. Inc.
Adress: Montréal, QC

Order no 445882 | Order date Dec 10, 2010 | Sub-total $198.00 | Taxes $25.75 | Shipping $11.95 | Credits $10.00 | Total $225.70

Deisha Sunglasses in Gray
Size: One Size
Price: $99.00
SKU: GIOGARMANI560SQMB

Deena Sunglasses in Silver and Blue
Size: One Size
Price: $99.00
SKU: GIOGARMANI599SVHV

Name: Alain
Adress: 2222, Vigi Avenue, Montréal, QC
Paid via : Master Card
Reference: 0102986473622

**Reference: 445882**
**Order Placed: Dec 10, 2010**

| Item | Item Code | Description | Price |
|---|---|---|---|
| | GIOGARMANI560SQMB | Deisha Sunglasses in Gray<br>Quantity: 1 | $99 |
| | GIOGARMANI599SVHV | Deena Sunglasses in Silver and Blue<br>Quantity: 1 | $99 |

| | | | |
|---|---|---|---|
| Total: | $198.00 | Client: | Alain |
| Postage and Handling: | $11.95 | Shipping Adress: | |
| Taxes: | $25.75 | 2222, Vigi Avenue, Montréal, QC | |
| Order Payement: | $225.70 | | |
| Payment Method: | Master Card | | |
| Transaction Number: | 0102986473622 | | |

Figure 2.2 – Deux exemples de bons de commande envoyés de deux sites différents. Chaque site a sa façon de représenter les informations.

L'analyse de l'information dans l'ingénierie des documents crée un modèle conceptuel généralisé exprimant les règles de gestion des affaires pour tous types de documents requis dans le contexte d'utilisation. Nous appelons cet artefact un modèle de

composants de documents définissant tous les éléments nécessaires pour optimiser la réutilisation et minimiser la redondance lors de la conception des nouveaux modèles de documents.

### 2.4.4 La conception des modèles de documents (Phase d'assemblage)

Nous passons ensuite de l'analyse à la conception. Nous créons des modèles pour les nouveaux types de documents basés sur les composantes, les structures et associations (par exemple, cardinalités en UML) satisfaisant aux exigences de notre contexte d'utilisation (voir la figure 2.3).

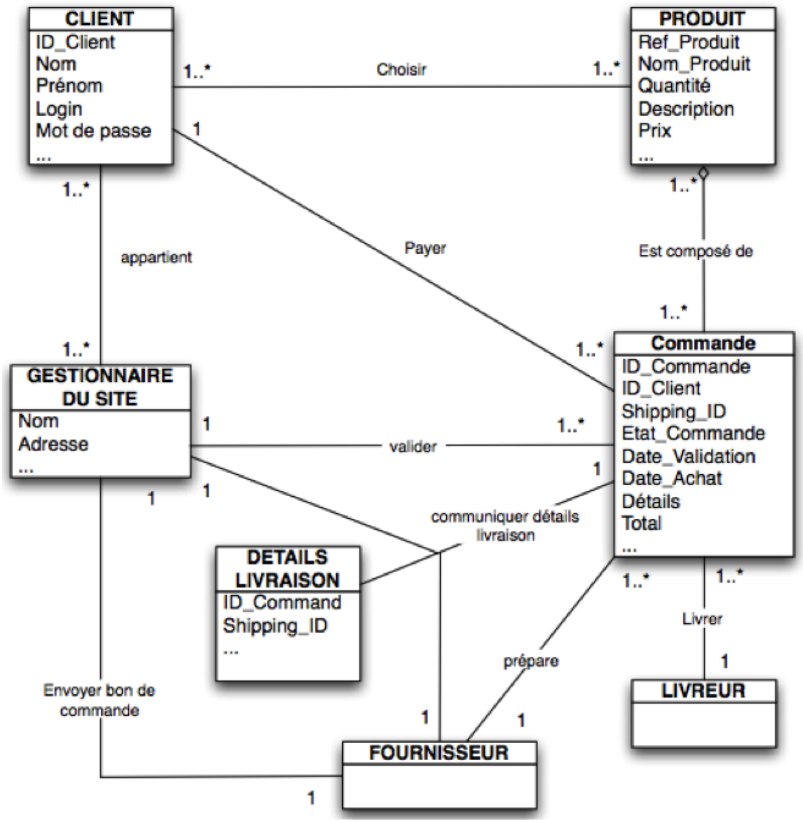



Figure 2.3 – Diagramme de classes de processus de vente en ligne.

Nous avons plusieurs composants participant à la réalisation de notre processus de vente en ligne. La figure 2.3 résume les informations ainsi que les relations entre les composantes du processus. Ces nouveaux modèles s'appellent modèles d'assemblage

de documents. L'assemblage de documents est basé sur les modèles de composantes d'information. Une conception efficace des modèles d'assemblage de documents nous oblige aussi à savoir quand les modèles peuvent être réutilisés, quand un nouveau modèle doit être créé, et ce qui distingue un modèle d'un autre.

### 2.4.5 L'implémentation des modèles (Encodage en XML)

Cette étape transforme le modèle conceptuel, qui met en évidence les relations sémantiques entre les classes (p.e. diagramme de classes), en un modèle physique qui décrit l'ensemble de ces classes (p.e. une base de données, un XML Schema, etc).

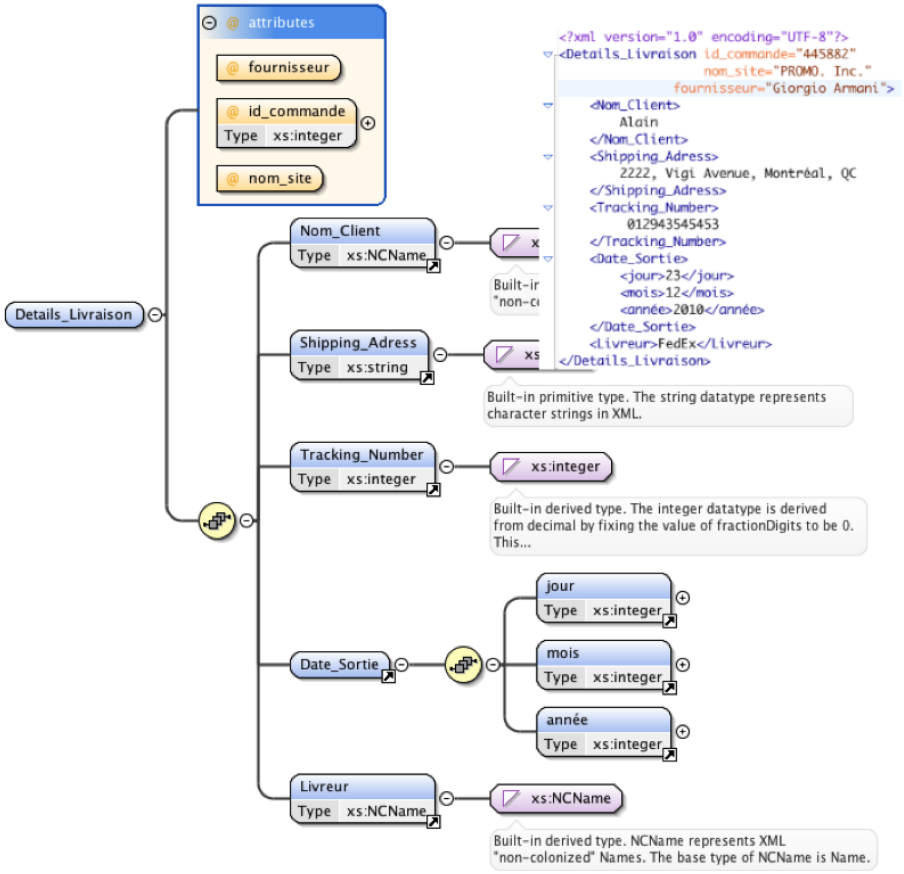



Figure 2.4 – Un modèle de document implanté avec XML Schema en lui associant une instance d'un document XML (en haut à droite). C'est un exemple de documents échangés entre le fournisseur et le gestionnaire du site qui contient les détails de livraison de la commande.

Le modèle d'implémentation de documents est réalisé dans un langage de balisage

comme XML Schema. Le choix est fait grâce aux avantages de XML Schema : la simplicité, la puissance, l'expressivité, et la maintenabilité [18]. La figure 2.4 décrit un modèle d'un document échangé entre le fournisseur et le gestionnaire du site. Ce document informe le gestionnaire du site de la sortie de la commande ainsi que les détails de la livraison.

Pour les modèles physiques des processus d'affaires, la réalisation est faite en adoptant un méta-modèle approprié pour coder les règles et les exigences du contexte d'utilisation. Cette adaptation permet de créer le modèle de mise en œuvre des processus d'affaires. Parmi les exemples de modèles de processus d'affaire, nous trouvons RosettaNet PIPs [41] encodée en XML.

## 2.5 Conclusion

Dans ce chapitre, nous avons présenté les différentes phases du processus d'ingénierie de documents. Pour chacune, nous avons donné un scénario d'exécution pour en illustrer le déroulement. La phase d'analyse de contexte d'utilisation ainsi que l'analyse de processus d'affaire sont importantes pour comprendre le déroulement du processus et dégager les participants et les types de documents échangés. L'analyse de ces documents donne une idée sur les composants d'information existants pour en extraire des patterns d'utilisation. Le résultat de ces analyses ainsi que les différentes phases dans le processus d'ingénierie de documents mènent à des modèles de collaboration intra et inter-entreprises. Ces derniers gèrent le processus d'affaire entre partenaires dans ce processus.

Au chapitre suivant, nous définissons le formalisme XML et ses avantages, le web sémantique, et leurs différences. Nous argumentons le besoin d'ontologie dans le domaine des affaires à la section 3.3.1. En plus, nous présentons les langages XML Schema et OWL 2 et nous comparons leur expressivité.

# CHAPITRE 3

# XML VS. WEB SÉMANTIQUE

## 3.1 Introduction

Dans ce chapitre, nous définissons le formalisme XML et ses avantages, le domaine du web sémantique et son utilité. Nous les comparons et soulignons les avantages apportés par le web sémantique dans le domaine de la gestion des processus d'affaires.

## 3.2 XML et ses avantages

XML (eXtensible Markup Language, ou langage de balisage extensible), est un "métalangage", mais aussi une organisation des informations. XML a été recommandé par le W3C en 1998, comme une version simplifiée de SGML (Standard Generalized Markup Language), dont il retient plusieurs principes. Le plus important point commun avec le SGML est le fait que tout document XML peut être basé sur une validation basée sur une grammaire. Les origines du SGML remontent à la fin des années 1960 avec le langage GML (Generalized Markup Language). Le SGML est devenu par la suite une norme ISO, et il a été adopté comme standard en 1986. Il fournit différentes syntaxes de balisage qui peuvent être utilisées pour définir des langages de balisage plus ou moins spécialisés. Il a servi de base pour la définition de XML et HTML.

Un document XML est structuré sous forme d'éléments imbriqués qui correspondent à une structure arborescente (voir à la figure 3.1). Cette arborescence possède différents types de nœuds :

- La racine est le nœud de base d'un document XML. Il est unique et englobe tous les autres éléments (nœuds).
- Un élément est un nœud possédant un nom. Les éléments sont encodés à l'aide de balises, qui sont déclarées entre < et >. Le début d'un élément est délimité par une balise ouvrante `<nom_balise>` et sa fin par une balise fermante `</nom_balise>`.

Si un élément est vide, on peut le définir avec la syntaxe `<nom_balise/>`. Ces éléments forment la structure du document : ce sont les branches (les balises) et les feuilles (le contenu textuel des balises) de l'arborescence. Ils peuvent contenir :

- du texte : le contenu d'un élément `auteur`,
- des attributs : l'élément `livre` contient un attribut `annee`,
- des éléments imbriqués : l'élément `livre` englobe les deux sous-éléments `titre` et `auteurs`.
- les éléments peuvent avoir un contenu mixte c.à.d. des données textuelles mélangées avec d'autres éléments, p.e. la balise `description` contient du texte ainsi que la balise `isbn`.

- Les commentaires sont considérés comme des nœuds à part entière dans l'arbre XML.

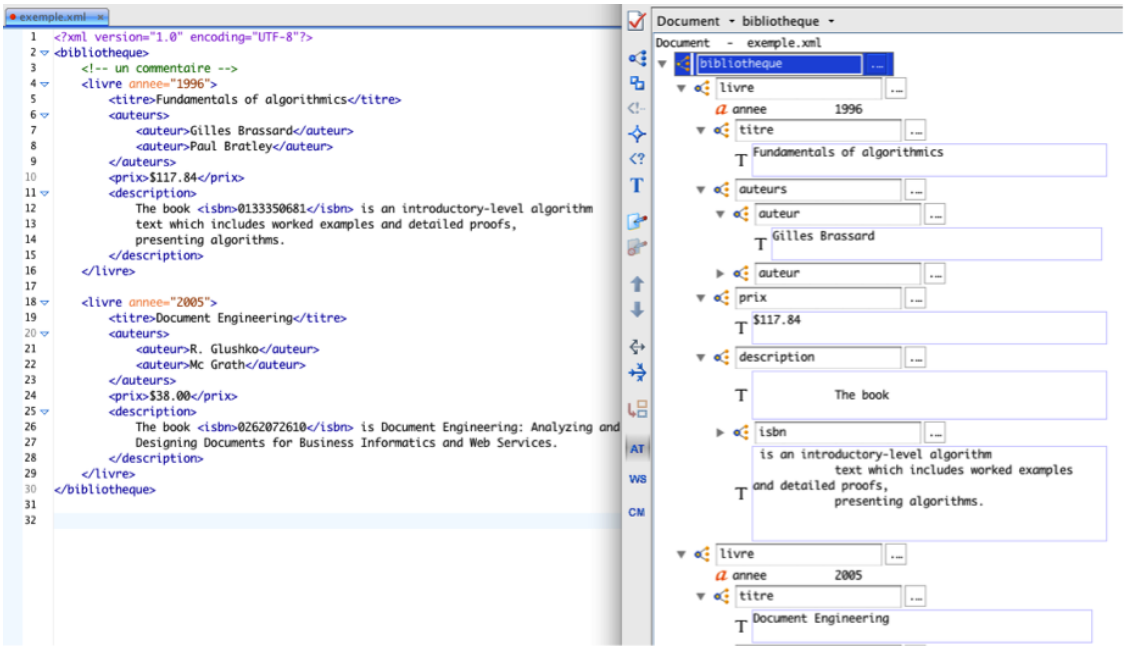



Figure 3.1 – Exemple d'un document XML (à gauche) avec sa représentation sous forme d'une arborescence (à droite). Ce document décrit l'ensemble des livres contenus dans une bibliothèque.

Le point fort de ce métalangage réside dans sa flexibilité et son interopérabilité entre diverses plates-formes de programmation et systèmes d'exploitation. La plus grande dif-

férence entre le HTML et le XML est que, contrairement au XML, le HTML contient des attributs et des éléments avec une sémantique prédéfinie. Par contre pour le XML, les auteurs d'un document peuvent créer des vocabulaires spécifiques à leur application ou à leur besoin. Ces balises sont des identificateurs arbitraires choisis comme étant évocateurs du type de leur contenu.

Un document XML est un document auto descriptif et extensible. Sa structure arborescente permet de modéliser une grande quantité de données informatisées. Il facilite l'échange de contenus entre des systèmes d'information hétérogènes, notamment, sur internet. Nous distinguons deux sortes de contenus : orientés données et orientés document.

Les contenus orientés données utilisent XML en tant que vecteur de données. Ils sont conçus pour être exploités par une machine. XML est généralement utilisé, comme une base de données, pour le stockage et l'échange de l'information structurées. Le document XML est l'unité fondamentale de stockage dans une base de données XML, l'équivalent d'une ligne d'une table dans une base de données relationnelle. Les contenus orientés données sont caractérisés par une structure assez régulière, des données qui présentent une granularité fine, c'est-à-dire que la plus petite unité de données est située au niveau d'un élément de type chaine de caractères ou d'un attribut, il est rare d'avoir du contenu mixte. Il existe plusieurs exemples de contenus orientés données : des relevés des transactions, des factures, des horaire des vols, etc.

La deuxième catégorie d'utilisation XML, généralement connu sous le nom "contenus orientés document", utilise XML pour indiquer la structuration d'un texte de son contenu, p.e. indiquer que telle partie d'un texte est un titre, un paragraphe, une liste énumérée, etc. Les contenus orientés document sont habituellement conçus pour être utilisés par des humains, comme des livres, des messages électroniques, des annonces, des manuels d'instruction, etc. Ils sont caractérisés par une structure irrégulière, des données qui présentent une granularité plus grande, et beaucoup de contenus mixtes.

Cette structuration permet de séparer l'information (le contenu) de son apparence (le contenant), et donc de fournir plusieurs types de sorties pour un même fichier de données, en fonction de l'utilisateur ou de l'application (un autre document XML, un

fichier HTML, un fichier PDF, etc). XML fait partie d'une famille de technologies :

- XPath : un langage de requête pour sélectionner des parties dans un document XML. En outre, XPath peut être utilisé pour calculer des valeurs (par exemple, des chaînes, des nombres ou des valeurs booléennes) à partir du contenu d'un document XML. Le langage XPath est basé sur la représentation arborescente du document XML. Il offre la possibilité de naviguer dans l'arborescence, et de sélectionner des nœuds en se basant sur une variété de critères.
- XQuery : un langage de requête pour sélectionner et extraire des éléments et des attributs à partir d'un ou plusieurs documents XML. Il offre aussi un ensemble de fonctionnalités permettant d'effectuer de nombreux types d'opérations sur des données et des documents XML, y compris :
  - La sélection de l'information fondée sur des critères spécifiques.
  - Le filtrage des informations non désirées.
  - Le tri, le groupement et l'agrégation des données.
  - La transformation et la structuration des données XML dans un autre vocabulaire ou structure XML.
  - La transformation des données XML en XHTML.

  Il est souvent qualifié de "SQL du XML", puisqu'il est semblable à SQL pour la base de données. Il permet de faire des requêtes complexes sur la structure ou encore les contenus en se basant sur des expressions XPath. Voici un exemple d'une question que XQuery pourrait résoudre en utilisant l'exemple illustré dans la figure 3.1 : sélectionner tous les livres ayant un prix inférieur à 100$ de la collection des livres stockée dans le document XML appelé `exemple.xml`.

```
for $book in doc("exemple.xml")bibliotheque/livre
where $book/prix < 100
order by $book/prix
return $book/titre
```

- XSLT : (Extensible Stylesheet Language Transformation) un langage de programmation qui sert à transformer des documents XML dans divers formats comme

HTML, XML, PDF, texte, etc. Ce langage sera utilisé au prochain chapitre 5 pour transformer les standards d'affaire.

- DTD (Document Type Definition), XML Schema, Relax NG, Schematron : langages de validation de documents. La définition d'un langage ou d'un type de document déclare un vocabulaire et des règles de validation : un ensemble d'éléments et d'attributs, un ensemble de valeurs prédéfinies pour certains de ces éléments, une séquence dans laquelle ces éléments peuvent apparaître, la définition des éléments obligatoires, facultatifs, répétitifs, etc. Un document XML, dit bien formé, signifie qu'il respecte les règles syntaxiques de XML garantissant une structure arborescente ; un fichier est valide s'il est bien formé et conforme à un schéma qui lui est associé. Un document XML bien formé est manipulable par un analyseur XML, indépendamment du type de données qu'il contient.

XML Schema est le plus utilisé pour la définition de la structure et la syntaxe des documents XML. En plus d'être exprimé en XML, il est plus riche que les DTDs qui ne permettent pas de valider le contenu texte des éléments et des attributs. Par exemple, les documents en XML Schema peuvent décrire des types de données qui ne peuvent pas être exprimés avec DTD.

XML Schema a été utilisé par la plupart des standards d'affaire étudiés dans notre thèse (définis au chapitre 4) pour encoder les modèles conceptuels des documents. Cette étape représente la dernière dans le processus d'ingénierie de documents (chapitre 2).

Dans la suite de cette section, nous présentons la spécification XML Schema et les informations de base nécessaires dans d'autres parties de la thèse.

### 3.2.1 XML Schema

XML Schema permet de définir que la structure et le type de contenu d'un document XML. Cette définition permet notamment de vérifier que la validité syntaxique de ce document. Le but d'un schéma est de définir une classe de documents XML. Il permet de décrire les informations syntaxiques comme l'ordre d'imbrication et d'apparition

des éléments et de leurs attributs dans un document XML. XML Schema comprend un ensemble de composantes appartenant à trois groupes principaux : les composants primaires, secondaires et auxiliaires.

Tout d'abord, les composants primaires comprennent les définitions des types simples (`xs:simpleType`) et complexes (`xs:complexType`), des éléments (`xs:element`) et d'attributs (`xs:attribute`). Dans XML Schema, il y a une différence de base entre les éléments de type complexe qui peuvent contenir des sous-éléments, avec un ordre prédéfini ou arbitraire, et être qualifiés par des attributs qui décrivent leurs caractéristiques et ceux de type simple qui ne peuvent contenir ni sous-éléments ni être qualifiés par des attributs. Les types simples sont définis comme des restrictions des valeurs, une liste de types de données de base fournies par le schéma XML, p.e. `string` et `boolean`, ou d'un type personnalisé.

Ensuite, les composants secondaires incluent les définitions des groupes d'attributs (`xs:attributeGroup`), les définitions de contraintes d'identité et les définitions de groupes modèles (`xs:group`). Un groupe d'attributs permet de nommer un groupe constitué de déclarations d'attributs et d'un attribut générique afin d'être ensuite utilisé par référence dans la représentation XML des définitions de types complexes et d'autres définitions de groupes d'attributs. Les définitions des groupes modèles sont fournies pour pouvoir y faire référence à dans les définitions de types complexes.

Les éléments peuvent avoir un ordre prédéfini représenté par l'élément `xs:sequence` dans le schéma XML ou arbitraire en utilisant l'élément de type `xs:choice`. Ainsi, il est possible de spécifier le nombre d'occurrences des éléments par les attributs de type `minOccurs` et `maxOccurs`.

Les constructions en XML Schema de type ; attribut, élément, type simple et complexe, groupe d'attributs et groupe modèle, ont des noms uniques indiqués par leur attribut `name`, en plus de pouvoir être référencées par d'autres constructions à l'aide de l'attribut `ref`. Ainsi, toutes les constructions en XML Schema peuvent avoir des identificateurs uniques indiqués dans leur attribut `id`).

Enfin, les composants auxiliaires représentent les annotations, les assertions et la gestion des révisions et du versionnement des schémas. D'une part, les annotations

(`xs:annotation`) sont utilisées pour documenter les éléments du schéma. D'autre part, les assertions (`xs:assert`) offrent une solution flexible pour contrôler les valeurs des éléments et d'attributs en associant un ensemble de contraintes sur ces valeurs. L'élément de type `xs:redefine` permet de redéfinir dans le schéma actuel des types simples et complexes, des groupes et des groupes d'attributs obtenus à partir de fichiers de schéma externes. Enfin, l'élément `xs:override` est utilisé pour réaliser des corrections dans la définition de n'importe quels composants du schéma.

## 3.3 Web sémantique

L'idée d'un Web sémantique a été introduite par Tim Berners-Lee dans les années quatre-vingt-dix [21]. Le contenu des documents HTML est aujourd'hui conçu pour être lu et interprété par les êtres humains. Ce contenu peut difficilement être manipulé et analysé, en tenant compte du sens, par des programmes informatiques. Les ordinateurs peuvent traiter de manière experte les pages Web pour leur mise en forme, mais en général, ils n'ont pas de moyen fiable de traiter la sémantique. Le défi du Web sémantique est donc de fournir un langage qui exprime des données et des règles de raisonnement sur ces données. L'idée principale du web sémantique est de rendre le contenu accessible et compréhensible par la machine, et de déduire du contenu implicite. Ceci ouvre la voie au développement d'outils sophistiqués susceptibles d'apporter un niveau supérieur de fonctionnalité pour assister les activités humaines sur le web.

Dans l'élaboration du web sémantique, XML fournit la couche de base de la manipulation syntaxique. Bien que XML soit un langage universel pour définir des balises, il ne fournit aucun moyen pour approcher la sémantique des données. Cette sémantique est traitée par les humains et non par les machines. Les ontologies sont devenues vitales pour beaucoup d'applications telles que des systèmes de gestion d'information et le commerce électronique. Selon Tim Berners-Lee, le web peut atteindre son plein potentiel s'il devient un lieu où les données peuvent être partagées et traitées par des outils automatisés et par les gens. Demain, les programmes devront partager des données et des processus, même si ces programmes ont été conçus de manière indépendante.

### 3.3.1 Besoin d'ontologie dans le domaine d'affaire

Le partage d'information est un problème dans le processus d'affaire surtout au niveau de documents échangés entre les parties. Pour remédier à ce problème, l'ordinateur doit avoir un meilleur accès à la sémantique de l'information [18]. Ainsi, pour un document, nous n'avons pas seulement besoin de stocker ses métadonnées (p.e. son auteur, titre, date de création, etc.), mais nous devons aussi rendre accessibles aux machines les concepts les plus importants abordés dans le document, les relations de ces concepts avec ceux d'autres documents, etc. L'utilisation des ontologies rend l'information interprétable par les machines.

Une représentation efficace d'un domaine organise les différents types de documents et dégage les relations inter-documents et inter-domaines. Cette organisation permet d'avoir une recherche plus efficace. Prenons l'exemple du domaine d'affaire, l'automatisation de la gestion de processus d'affaire est encore très limitée [20]. Il n'existe ni représentation uniforme de l'ensemble de documents dans un processus d'affaire, ni organisation selon un point de vue sémantique. Hepp et al. [20] ont montré que les entreprises ont besoin d'avoir :

- une vue unifiée sur les processus d'affaire.
- une modélisation efficace de leurs processus.
- une représentation de ces processus sous un format lisible par les machines, ce qui permet d'interroger les processus par des requêtes logiques.
- une vue sémantique de l'ensemble de documents.

Une des solutions proposées est l'utilisation des technologies offertes par le web sémantique. Les ontologies codifient les concepts de base d'un domaine avec un vocabulaire commun. La représentation à l'aide des ontologies facilite le partage d'information entre les parties et les agents logiciels ainsi que la réutilisabilité des connaissances. Durant notre étude, nous utiliserons les ontologies pour la modélisation de documents dans les processus d'affaire.

Le consortium W3C a publié deux standards du Web sémantique ; RDF/RDFS (Resource Description Framework) et OWL (Web Ontology Language), pour la description des ontologies. Selon Tim Berners-Lee, RDF et OWL constituent le fondement pour les applications du Web sémantique offrant différents niveaux d'expressivité. Ces formats fournissent un cadre de travail pour la gestion des ressources, l'intégration, le partage et la réutilisation des données sur le Web. Avant de les appliquer dans un cadre sémantique, il nous faut d'abord apprendre à connaître ces deux spécifications.

### 3.3.2 RDF/RDFS

L'aspect sémantique est exprimé par le langage RDF et encodé par un ensemble de triplets. RDF est un moyen d'identifier les relations entre les ressources sur le web. Il repose sur un modèle de triplet (sujet, propriété, objet) ou (sujet, verbe, complément) [21] où :

- le sujet est la ressource à décrire.
- le prédicat est un type de propriété applicable au sujet. Le prédicat lui-même est considéré comme une ressource, et identifié par une URI. Un URI est une chaine de caractères utilisée pour identifier un nom ou une ressource sur Internet.
- L'objet est une donnée ou une autre ressource.

Le tableau 3.I représente des triplets RDF sous différentes formes correspondant au texte suivant :

**`Jamel rédige un rapport sur l'ingénierie de documents.`**

Allant de haut en bas, nous voyons : le format en triplet dans lequel chaque élément est une URI ou une donnée dans le cas d'un objet, le format relationnel et le format d'échange RDF/XML. Il existe aussi une notation graphique dont le sujet et l'objet sont liés par une flèche étiquetée.

Dans la perspective du web sémantique, RDF définit une couche au-dessus de XML. De ce fait, RDF a été doté d'une syntaxe XML. Toutefois, RDF ne fournit aucun mécanisme pour décrire des propriétés ni pour décrire les relations entre ces propriétés et

Tableau 3.I – Représentation des triplets RDF sous différentes formes.

| | |
|---|---|
| **Triplets** | • Jamel {**est_rédacteur_de**} l'ingénierie de documents.<br>• L'ingénierie de documents {**est_un**} rapport. |
| **RDF/XML** | `<rdf:RDF xmlns:rdf=".../22-rdf-syntax-ns#"`<br>`xmlns:phd="http://www.iro.umontreal.ca/jridijam#">`<br>`  <rdf:Description rdf:about="phd:jamel">`<br>`  <phd:est_redacteur_de>`<br>`  <rdf:Description rdf:about="phd:ingenierie_de_documents">`<br>`  <phd:est_un>`<br>`  <rdf:Description rdf:about="phd:rapport"/>`<br>`  </phd:est_un>`<br>`  </rdf:Description>`<br>`  </phd:est_redacteur_de>`<br>`  </rdf:Description>`<br>`</rdf:RDF>` |
| **Turtle** | `@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns> .`<br>`@prefix phd: <http://www.iro.umontreal.ca/jridijam#> .`<br>`<phd:jamel> phd:est_redacteur_de <phd:ingenierie_de_documents>.`<br>`<phd:ingenierie_de_documents> phd:est_un <phd:rapport>.` |

d'autres ressources. C'est le rôle du langage de description de vocabulaire RDF "RDF Schema". RDF Schema est une extension sémantique à RDF. Un schéma est un méta-modèle qui spécifie quels types de ressources peuvent être utilisés dans les triplets RDF. Ces schémas introduisent de nouvelles primitives pour la définition de classes, de propriétés, dont la sémantique formelle s'appuie sur la logique du premier ordre. La définition de classes et propriétés du langage RDFS est similaire aux systèmes de types des langages de programmation orienté objet tels que Java. Par exemple, nous pourrions définir la propriété `author` ayant un domaine `Document` et un type `Person`, alors qu'un système orienté objet classique définirait typiquement une classe `Document` avec un attribut `author` de type `Person`. Les ressources peuvent être divisées en groupes appelés des classes (`rdfs:Class`). Les membres d'une classe sont dits des instances de la classe. Les classes sont elles-mêmes des ressources. Elles sont souvent identifiées par

des URI et peuvent être décrites par des propriétés RDF. Il existe plusieurs propriétés qui peuvent exister entre les classes ou ressources ; p.e. les classes `Shoe` et `Cloth` sont des sous-classes de la classe `Product`, alors toutes les instances de `Shoe` et `Cloth` seront également des instances de `Product`. La propriété `rdfs:subClassOf` peut être utilisée pour déclarer qu'une classe est une sous-classe d'une autre.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/.../22-rdf-syntax-ns"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema" >
  <rdfs:Class rdf:ID="Product"/>
  <rdfs:Class rdf:ID="Shoe">
    <rdfs:subClassOf rdf:resource="Product"/>
  </rdfs:Class>
  <rdfs:Class rdf:ID="Cloth">
      <rdfs:subClassOf rdf:resource="Product"/>
  </rdfs:Class>
</rdf:RDF>
```

La classe `rdf:Property` est la classe des propriétés RDF. Elle permet d'associer une propriété à une ressource ou classes en utilisant :

- La propriété `rdfs:range` est une instance de la classe `rdf:Property`, utilisée pour déclarer l'ensemble des valeurs possibles d'une propriété (une donnée ou bien une autre ressource),
- La propriété `rdfs:domain` est une instance de la classe `rdf:Property`, utilisée pour déclarer qu'une propriété s'applique à une classe de ressources. Par exemple, la propriété `referencedBy` sera appliquée sur la classe `Product` avec une valeur de type `xsd:string`. Par contre dans le deuxième exemple, nous avons la propriété `composedOf` qui s'applique sur la classe `Product`.

```
<rdf:Property rdf:ID="referencedBy">
  <rdfs:domain rdf:resource="#Product"/
  <rdfs:range rdf:resource="&xsd;string"/>
</rdf:Property>

<rdf:Property rdf:ID="composedOf">
  <rdfs:domain rdf:resource="#Order"/>
  <rdfs:range rdf:resource="#Product"/>
</rdf:Property>
```

La puissance d'expression de RDF et de RDFS est très limitée. Il est impossible de préciser la nature des relations entre ressources (réflexivité, etc.) Ainsi qu'il n'y a pas de contraintes d'existence et de cardinalité.

- RDFS ne permet pas d'exprimer que 2 classes sont disjointes :
  - Exemple : les classes `Shoe` et `Cloth` sont disjointes.
- RDFS ne permet pas de créer des classes par combinaison ensembliste d'autres classes (intersection, union, complément) :
  - Exemple : on veut construire la classe `Product` comme l'union disjointe des classes des `Shoe` et `Cloth`.
- RDFS ne permet pas de définir de restriction sur le nombre d'occurrences de valeurs que peut prendre une propriété :
  - Exemple : une commande appartient à un et un seul client.
- RDFS ne permet pas de caractériser des propriétés notamment :
  - Transitivité :`isMoreExpensiveThan`, un produit est plus cher qu'un autre.
  - Unicité : `hasPartner`, une commande appartient à un seul client.
  - Propriété inverse : la propriété `shipTo` est l'inverse de `shippedBy`.

Ces limites ont donné naissance à un nouveau langage de description d'ontologies, nommé OWL (Web Ontology Language), qui a été recommandé par le W3C début 2004, plus expressif que RDF et RDFS, jusqu'à l'arrivée de OWL 2.

Le but dans le développement d'OWL 2 est de fournir un support efficace de modélisation d'ontologies et une plateforme robuste pour le développement futur [38]. Nous utilisons OWL 2 dans le cadre de cette thèse dont la syntaxe est décrite dans le document W3C [37].

### 3.3.3 OWL 2

OWL 2, recommandé par le W3C en 2012, est un langage de description d'ontologies conçu pour la publication et le partage d'ontologies sur le Web sémantique. Une ontologie est un ensemble structuré des termes et concepts utilisés pour décrire et représenter un domaine de connaissance. Les ontologies sont utilisées par des personnes, des bases de données et des applications, qui ont besoin de partager les informations d'un domaine p.e. la médecine, les processus d'affaire, la gestion financière, etc. Les ontologies incluent les définitions des concepts de base dans un domaine, leurs relations et la description des connaissances.

OWL 2 permet de définir des classes (concepts) à travers l'élément `owl:Class` qui représentent des ensembles d'individus (objets, instances) et des propriétés qui définissent les caractéristiques de ces individus et des relations (binaires) possibles entre eux. OWL distingue deux grandes catégories de propriétés celles reliant deux concepts (`owl:ObjectProperty`) ou les propriétés de type de données reliant un concept à un littéral (`owl:DataProperty`). Aussi, les classes peuvent être définies comme des combinaisons logiques (intersection, union ou complément) d'autres classes. Une ontologie OWL 2 est composée de trois catégories syntaxiques différentes qui forment la base de la logique de description utilisée comme modèle formel :

- Entités (les classes, propriétés et individus) : elles constituent les éléments de base d'une ontologie. Par exemple, une classe `Product` représente l'ensemble des produits. De même, la propriété `composedOf` représente la relation entre `Product` et `Order`. Enfin, l'individu `GIOARMANI560SQMB` représente une instance particulière de la classe `Sunglass` (sous classe de `Accessory`).

- Expressions : représente une combinaison d'entités pour former des descriptions complexes à partir de formes des entités et d'opérateurs.

- Les axiomes sont l'ensemble des énoncés supposés vrais. Par exemple, en utilisant un axiome `subClassOf`, on peut affirmer que la classe `Shoe` est une sous-classe de la classe `Product`.

Ces trois catégories syntaxiques expriment la partie logique d'une ontologie. Elles sont interprétées avec une sémantique précise qui permet de tirer des inférences utiles. Par exemple, si l'individu `GIOARMANI560SQMB` est une instance de la classe `Sunglass`, qui est une sous-classe de `Accessory` (sous-classe de `Product`), on peut dire que `GIOARMANI560SQMB` est aussi une instance de la classe `Product`.

OWL 2 définit plusieurs profils offrant différents compromis en terme d'expressivité et de complexité. Il repose sur deux principales couches : une couche syntaxique et une sémantique. En pratique, une syntaxe est nécessaire pour le stockage et l'échange des ontologies OWL 2. La syntaxe d'échange primaire pour OWL 2 est RDF/XML, d'autres syntaxes peuvent aussi être utilisés. Il s'agit notamment de Turtle, OWL/XML et la syntaxe de Manchester.

OWL 2 introduit `owl:AllDisjointClasses` comme un raccourci qui nous permet de déclarer plusieurs classes disjointes. Concernant les propriétés et les relations entre les classes, OWL 2 permet de déclarer des relations de différentes natures : asymétrique via `owl:AsymetricProperty`, réflexive via `owl:ReflexiveProperty` et non réflexive via `owl:IrreflexiveProperty`.

OWL 2 supporte la plupart des types de données à l'exception de quelques types de données, p.e `xsd:time`, `xsd:date`, `xsd:gYear`, `xsd:gMonth`, `xsd:gMonthDay`, et `xsd:gYearMonth`. Par ailleurs, il propose des nouveaux types de données, p.e. `owl:real` qui représente les nombres réels, et `xsd:dateTimeStamp` qui se base sur `xsd:dateTime` mais il nécessite la spécification de fuseau horaire. En plus des types de base, OWL 2 supporte l'utilisation des restrictions sur les valeurs de données inspirées à partir de XML Schema [21]. Les restrictions sur les données numériques sont `xsd:maxInclusive` et `xsd:MinExclusive` pour définir un intervalle de valeurs, pour les chaines de caractères sont `xsd:minLength`, `xsd:maxLength` et `xsd:pattern` qui permet de sélectionner des sous-chaînes de caractères en se basant sur des expressions régulières.

En pratique, une syntaxe est nécessaire pour le stockage et l'échange des ontologies OWL 2. La syntaxe d'échange primaire pour OWL 2 est RDF/XML [36], d'autres syntaxes peuvent aussi être utilisés comme OWL/XML [38] et la syntaxe de Manchester [54].

Actuellement, il existe plusieurs outils de construction et de vérification d'ontologies dont *Protégé*[1], un éditeur d'ontologies open source. Il offre un support d'interrogation et utilise aussi des moteurs d'inférences qui vérifient la consistance de l'ontologie crée c.à.d. tester l'absence de définitions contradictoires. Il existe plusieurs moteurs d'inférences sur les logiques de description tels que *Pellet* [48] qui accepte en entrée des fichiers OWL.

## 3.4 XML vs. OWL

Dans cette section, nous distinguons XML Schema et OWL 2. Les schémas XML et les ontologies OWL 2 offrent chacun un vocabulaire et une structure décrivant les données.

XML Schema décrirt la structure d'un document XML, plus précisément ; les éléments et leur ordre d'imbrication, les attributs XML et les types de données. XML fournit un ensemble de règles pour la création de vocabulaires qui structurent à la fois les documents et les données. Il est une syntaxe puissante pour représenter des documents structurés, mais n'impose aucune contrainte sémantique sur la signification de ces documents. XML décrit un arbre de nœuds, par contre OWL repose sur une représentation sous forme de graphe à base de triplets RDF (sujet-prédicat-objet).

OWL 2 est un langage plus expressif que XML Schema utilisé pour composer des vocabulaires XML. En effet, il existe une certaine équivalence sémantique entre les constructions en XML Schema et celles en OWL. Par exemple, les types complexes du XML Schema représentent des classes d'instances XML ayant des caractéristiques communes, tout comme les classes OWL représentent des ensembles d'individus ayant des propriétés communes. D'autre part, les éléments de type simple en XML Schema représentent des caractéristiques d'un type complexe, semblables aux propriétés en OWL 2 qui représentent le comportement d'une classe.

Par contre, OWL 2 permet des constructions qui ne peuvent pas être exprimées en XML Schema, p.ex. l'union des classes, les classes disjointes, les propriétés asymé-

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

triques, réflexives, et disjointes, et la représentation de la hiérarchie de classes.

Certaines définitions en XML Schema ne peuvent pas être directement exprimés en OWL 2, p.ex. les `xs:assert`, `xs:redefine` et `xs:override`. Suite à notre analyse, faite à la section 5.3.1, des documents d'affaire définis en XML Schema, ces constructions n'ont pas été utilisées dans la définition de ces documents ce qui nous pose pas des problèmes lors de la transformation. D'autre part, l'ordre d'apparition des éléments en XML Schema définis par `xs:sequence` et `xs:choice` ne changent pas la sémantique du document et ne sont donc pas exprimés en OWL 2 qui ne traite pas l'ordre d'apparition des éléments, mais qui repose plutôt sur une représentation sous forme de graphe.

En conclusion, OWL 2 est plus adéquat pour la représentation de la connaissance que XML Schema qui valide les documents XML syntaxiquement sans avoir recours à leur sémantique. En plus d'être plus expressif que XML Schema, OWL 2 est doté d'une capacité d'inférence et de raisonnement déductif sur les concepts de l'ontologie.

## 3.5 Conclusion

Dans ce chapitre, nous avons présenté le langage XML, le web sémantique ainsi que leur différence. XML est un langage de balisage extensible indépendant du langage de programmation ou du système d'exploitation. Il donne accès à une panoplie de technologies pour le traitement, la structuration, la transformation et l'accès de données. Son point fort réside dans sa lisibilité et son interopérabilité. Il permet de structurer un ensemble de documents.

Dans le domaine d'affaire, des standards ont été proposés pour la gestion de documents circulant dans un processus d'affaire en utilisant les technologies XML. Au chapitre suivant, nous citons ces standards d'affaire en citant leurs avantages et leurs limites.

# CHAPITRE 4

# ÉTUDE DES STANDARDS D'AFFAIRE

## 4.1 Introduction

L'ingénierie de documents est une nouvelle discipline pour la spécification, la conception et la mise en œuvre des documents échangés électroniquement. Cette discipline est appliquée dans plusieurs domaines afin de faciliter la communication et gérer la grande masse d'informations qui circulent dans/en dehors d'un organisme.

Aujourd'hui, il existe une grande variété de modèles de documents d'affaire qui gèrent l'échange entre les partenaires commerciaux. Ainsi, il devient inévitable de traiter les différents documents dans le processus d'affaire afin de concevoir des normes ou des conventions communes pour l'échange électronique de données.

L'idée d'échange électronique des données (EDI) entre les entreprises n'est pas nouvelle, elle est mise en œuvre depuis les années 1960. La gestion des échanges électroniques de documents dans les affaires inter-entreprises a été dominée pendant longtemps par des standards d'échange traditionnels, comme UN/EDIFACT [25]. L'EDI a offert aux entreprises la perspective, de réduire les coûts, d'éliminer l'échange de documents papier, et d'améliorer leur productivité en utilisant l'échange électronique d'information. Cependant, seules les grandes compagnies ont eu les moyens de développer des services EDI, et elles imposent à leurs partenaires commerciaux de suivre leur modèle de document. Si un partenaire change son système d'échange, alors tous ses partenaires doivent également changer de leur côté. Plusieurs travaux et standards ont été proposés pour la gestion de documents d'affaire. Ces travaux ont essayé d'utiliser plusieurs techniques (p.e. XML, le web sémantique, etc.) pour remédier aux limites des EDI traditionnels. Mais, les grandes entreprises n'ont pas accepté ce changement car le coût elevé de l'implantation de l'EDI montrent qu'elles ne sont pas prêtes à réinvestir dans une autre solution.

Avant d'entrer dans les détails de l'utilisation des modèles de processus d'affaire

pour soutenir les solutions B2B qui désigne une relation commerciale entre entreprises basée sur l'utilisation d'un support numérique pour les échanges d'information. Il est intéressant de regarder l'histoire des initiatives en B2B pour les classer et voir leur évolution au fur et à mesure de l'apparition de nouvelles technologies.

## 4.2 Description des standards d'affaire basés sur XML

L'échange informatisé de données permet au système informatique d'une entreprise de communiquer et d'échanger des données numériques avec des systèmes informatiques d'autres sociétés. Ce qui permet aux entreprises l'automatisation des échanges électroniques des documents commerciaux tels que les bons de commande et les factures. Environ 90% des grandes entreprises ont investi dans l'EDI [25] et la gestion de documents. Le concept d'échange électronique des documents d'entreprises a conduit à l'élaboration de plusieurs standards, par exemple aux États-Unis le US Transport Data Coordinating Committee (TDCC) en 1968 et GTDI en Europe en 1983. En raison de la mondialisation des échanges, le UN/ECE/WP.4 (un prédécesseur de l'UN/CEFACT) a lancé, au milieu des années 1980, une initiative menant à l'UN/EDIFACT, devenue une norme ISO en 1987. UN/CEFACT est un organisme des Nations unies dont l'appellation en français est *Facilitation des Procédures Commerciales et du Commerce Électronique*, ce qui correspond en anglais à *United Nations Centre for Trade Facilitation and Electronic Business*. Cet organisme encourage la collaboration entre les gouvernements et les entreprises ainsi qu'entre les entreprises, afin d'assurer l'interopérabilité des échanges d'information.

Le langage XML s'est rapidement imposé comme la solution pour la définition des formats d'échange sur internet entre les applications du commerce électronique. Il a aidé à remédier aux limites des normes traditionnelles, car il utilise un mécanisme d'identification des données par le marquage explicite de l'information. Plusieurs vocabulaires d'affaires basés sur XML ont été développés, nous en présentons ici quatre basés sur

XML dans l'ordre de leur apparition : xCBL[1], RosettaNet [2], cXML[3] et ebXML[4]. Ces standards d'affaire serviront d'exemples aux chapitres 5 et 6. L'effort de standardisation xCBL a commencé en 1997, avant la recommandation W3C du langage XML en 1998. RosettaNet a été lancé en 1998. cXML (commerce eXtensible Markup Language) est un protocole de communication basé sur le langage XML, créé par la société Ariba en 1999. Le projet ebXML a débuté dans la fin de l'année 1999 pour fournir une infrastructure complète et de répondre aux besoins liés à l'utilisation de XML dans l'e-business. L'e-business, dans le sens d'*affaires électroniques*, recouvre les applications faisant appel aux technologies d'information et de communication pour traiter les relations entre une entreprise et ses partenaires. Il utilise des moyens électroniques pour réaliser des affaires, comme le web et l'internet. L'e-business comprend plusieurs processus :

- Les processus de production, dont l'approvisionnement, la commande et la gestion des stocks, le traitement des paiements, la relation électronique avec les fournisseurs et les processus de contrôle de la production.

- La gestion des clients, par exemple les promotions et les efforts de commercialisation, la vente sur internet, le traitement des bons d'achat et des paiements.

- La gestion des processus internes, comme les services aux employés, la formation, le partage d'informations internes et le recrutement.

Il ne faut surtout pas confondre e-commerce et e-business. Le e-commerce, ou bien *commerce électronique*, représente un cas particulier et ne couvre qu'une partie du e-business. Il utilise les supports électroniques pour tout ou partie des activités commerciales entre une entreprise et les particuliers (la publicité, les commandes en ligne, le paiement électronique, la livraison).

[1] http ://xcbl.org/
[2] http ://www.rosettanet.org/
[3] http ://cxml.org/
[4] http ://ebxml.org/

### 4.2.1 xCBL

Créé en 1997, xCBL a été initialement appelé Common Business Library (CBL) et développé dans un projet de recherche pour tester les limites de XML pour le commerce électronique. En Janvier 1999, CommerceOne acquis la technologie CBL. Cela a conduit à la création de xCBL 2.0, la première version accessible au public.

xCBL est une collection de spécifications XML de documents d'affaires utilisés dans l'e-business. Il ne définit pas les processus d'affaires ou de messagerie d'une façon détaillée. xCBL est un ensemble de blocs XML et un framework de documents qui permet la création de documents XML solides et réutilisables, pour faciliter le commerce mondial [1]. Il est utilisé comme langage partagé entre les participants dans le domaine du commerce électronique. Cette interopérabilité permet aux entreprises d'échanger facilement des documents d'affaire, en donnant un accès global aux acheteurs, aux fournisseurs et aux intervenants dans un processus d'affaire. xCBL 4.0, la dernière version validée en 2003, supporte tous les documents essentiels et transactions dans le secteur du commerce électronique, y compris l'automatisation de la chaîne d'approvisionnement multi-entreprises, les achats directs et indirects, la planification, les ventes aux enchères, la facturation et le paiement dans un contexte international (paiement en devises). xCBL est disponible dans de nombreux formats : xCBL 4.0 est disponible en tant que W3C XML Schema seulement. Mais les versions précédentes sont fournies comme un ensemble de schémas de SOX (SOX est le langage de schéma créé par Commerce One et utilisées dans ses propres produits). Techniquement, xCBL 4.0 contient 44 documents et utilise plusieurs espaces de noms, où chaque espace de noms représente un domaine fonctionnel, par exemple :

**Functional area** : `ordermanagement`
**Namespace** : `rrn:.../xcbl/v4_0/.../v1_0/ordermanagement.xsd`
**Documents** : `Order, ChangeOrder, OrderResponse, OrderRequest, OrderStatusRequest, OrderStatusResult, OrderConfirmation, OrderConfirmationResponse.`
**Description** : Il contient les documents xCBL associés à la création de bons d'achat et la gestion des commandes.

### 4.2.2 RosettaNet

Fondé en 1998, RosettaNet est un organisme indépendant, autofinancé, à but non lucratif regroupant plus de 500 entreprises de haute technologie. Parmi les entreprises qui soutiennent et utilisent les normes RosettaNet, Intel, IBM, Oracle, Netscape, Cisco Systems, Microsoft, et Texas Instruments. Ces membres créent et implémentent des standards pour le traitement des transactions du B2B en particulier au niveau de la chaine d'approvisionnement. Rosettanet se focalise sur l'industrie informatique et développe des standards verticaux efficaces. Ces standards définissent des schémas XML décrivant une centaine de processus d'échange d'information entre les partenaires en s'appuyant sur Partner Interface Processes (PIP) [41][31]. Les PIPs sont organisés en sept groupes correspondant à un processus d'affaire particulier (Gestion des profils des partenaires, information des produits, gestion des commandes, gestion des stocks, marketing, service après-vente et support technique, et la fabrication). Ces groupes sont décomposés en segments représentant le point central du réseau commercial. Chaque segment aura un identifiant dans le système de RosettaNet. Nous prenons le cas du groupe 3 (Gestion des commandes) qui est composé de plusieurs segments, par exemple :

- PIP 3A2 : Demande de prix et disponibilité.
- PIP 3A3 : Transfert du panier d'achat.
- PIP 3A4 : Gestion des bons de commande.
- PIP 3A6 : Demande d'état des commandes distribuées.
- PIP 3A7 : Notification d'acceptation du bon de commande.
- PIP 3B4 : Demande d'état de la livraison.

Pour réaliser des affaires électroniques dans le cadre de RosettaNet, les partenaires doivent passer par certaines étapes. Premièrement, les partenaires de la chaîne d'approvisionnement se réunissent et analysent leurs scénarios d'affaires inter-entreprises (par exemple, comment ils interagissent pour faire des affaires les uns avec les autres, quels sont les documents qu'ils échangent et dans quel ordre). Puis ils définissent les processus

électroniques mis en œuvre dans le cadre du RosettaNet. La figure 4.1, adaptée de Dogac [10], présente un scénario d'achat entre deux entreprises dans le cadre de RosettaNet.

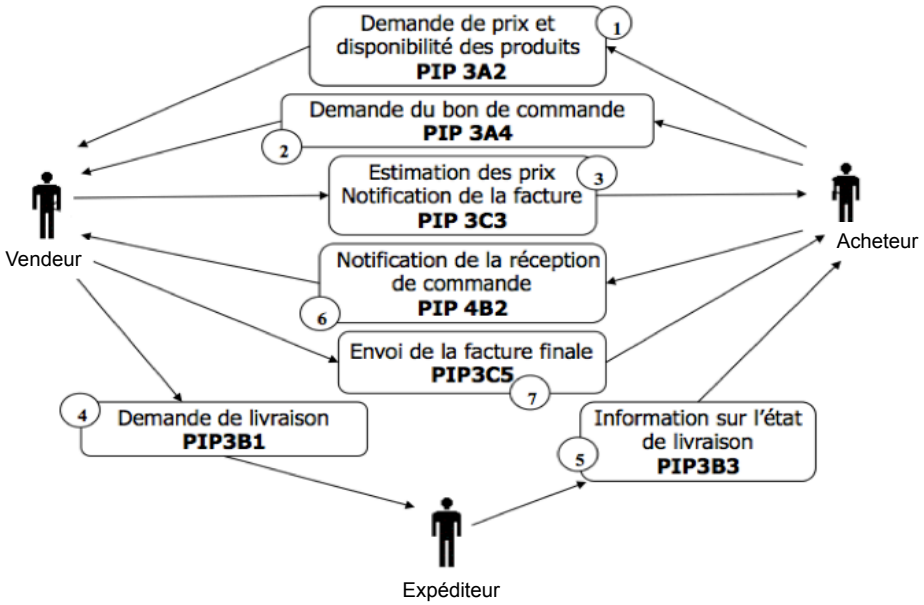



Figure 4.1 – Scénario d'achat entre deux entreprises dans le cadre de RosettaNet inspiré de Dogac [10].

Dans ce scénario, un acheteur demande le prix et la disponibilité de certains produits d'un vendeur (PIP3A2). Après avoir reçu la réponse, l'acheteur initie une demande de commande (PIP3A4) et un document XML sera communiqué, validé par le schéma `PurchaseOrderRequest_02_05.xsd`. Le vendeur, d'autre part, traite le bon de commande et envoie une notification d'une facture (PIP3C3) à l'acheteur. Le vendeur envoie une demande de livraison (PIP3B1) au livreur (il y a un tiers dans ce scénario, qui est un livreur). Après la livraison des marchandises, le livreur envoie le statut de la livraison (PIP3B3). Lorsque l'acheteur reçoit la livraison, il envoie une notification de la réception de la commande (PIP4B2) au vendeur. Enfin, le vendeur prépare une facture finale et en avise l'acheteur (PIP3C5).

Les PIPs incluent la spécification des partenaires dans un processus d'affaires (l'acheteur, le vendeur, le livreur, etc.), les activités commerciales entre eux, le type, le contenu, et la séquence des documents commerciaux échangés par les interactions en exécutant ces activités. Ils précisent également le temps, la sécurité, l'authentification et des

contraintes de performance de ces interactions. La structure et le contenu des documents d'affaires échangés sont spécifiés grâce à une DTD [6]. Les partenaires commerciaux qui utilisent les règles de RosettaNet doivent échanger des documents conformes aux DTD décrits dans PIPs. En plus des PIPs, RosettaNet comprend deux dictionnaires, qui fournissent un ensemble de propriétés pour ces PIPs. Ces dictionnaires sont `RosettaNet Technical Dictionary` et `RosettaNet Business Dictionary`. L'échange de documents dans le cadre de RosettaNet est géré par la mise en œuvre de la norme RNIF qui concerne le routage, l'emballage et la sécurité. La norme RNIF définit la structure du message, le cryptage du message, et la signature numérique. Elle est basée sur les technologies HTTP, MIME, et les normes XML.

Même si RosettaNet est plus abouti que xCBL puisqu'il offre une définition détaillée des processus d'affaire, des documents commerciaux et de la messagerie au niveau de la chaîne d'approvisionnement. Il souffre de limites en ce qui a trait au problème d'alignement de processus d'affaire et la définition ambigüe des messages. Premièrement, le processus interne de chaque acheteur doit être ajusté à chaque vendeur, c.à.d. que tous les processus doivent être conformes avec les PIPs [31]. La définition de schémas contient un ensemble d'éléments optionnels qui ne sont pas implémentés par chaque compagnie. Dans ce cas, un travail manuel est nécessaire pour interpréter les nouvelles informations envoyées par un nouveau partenaire. Lorsque le nombre de partenaires augmente, l'échange devient de plus en plus difficile [31].

### 4.2.3 cXML

Le standard cXML est un protocole basé sur le langage XML, créé par la société Ariba en 1999 et destiné à échanger des documents entre les systèmes d'entreprises, les acheteurs et les fournisseurs. Les modèles des documents sont définis en DTD. Ces DTD sont des fichiers texte décrivant la syntaxe et l'ordre des éléments d'un document cXML. Les applications cXML utilisent ces DTDs pour valider tous les documents entrants et sortants d'une entreprise [2]. Actuellement, ce protocole est supporté par plus de 50 entreprises internationales à travers le monde y compris Visa, Cisco Systems et GM [51].

Les documents cXML ont été définis pour décrire les détails des transactions, le contenu de catalogues, les bons de commande et la confirmation des commandes, etc. `Catalogs` et `Purchase Orders` sont les types de documents les plus utilisés [2].

- `Catalogs` : les catalogues sont des fichiers qui véhiculent les informations des produits et le contenu des services aux acheteurs. Ils décrivent les produits et les services offerts par un fournisseur, et ils sont la principale voie de communication entre les fournisseurs et leurs clients.

- `Purchase Orders` : un bon de commande est une demande formelle envoyée par un acheteur à un fournisseur pour l'exécution d'un contrat d'achat. Le processus de réalisation d'une commande commence lorsque le client envoie le document `OrderRequest`, une version électronique d'un bon de commande. Une fois, la commande reçue dans le système de gestion de commande, le fournisseur renvoie un accusé de réception avec le document `OrderResponse` qui indique si la demande a été reçue avec succès.

### 4.2.4 ebXML

ebXML est une suite de spécifications qui permettent aux entreprises de toute taille et dans n'importe quel lieu géographique de mener des affaires sur internet. Il est basé sur des normes comme XML et UML. En utilisant ebXML, les entreprises ont maintenant une méthode standard pour échanger des données, définir et enregistrer les processus d'affaires. ebXML a commencé en 1999 comme une initiative d'OASIS[5] et CEFACT/UN. Fondée en 1993, OASIS est un organisme international sans but lucratif qui s'intéresse au développement et l'adoption de standards dans l'e-business. OASIS compte plus de 5000 participants représentant plus de 600 organisations et membres dans 100 pays.

Le but d'ebXML est de créer un seul marché électronique mondial où les entreprises peuvent communiquer, échanger des données, devenir des partenaires commerciaux et réaliser des affaires [25]. Toutes ces opérations seront effectuées automatiquement en

[5]http ://www.oasis-open.org/org

échangeant des documents XML. En conséquence, ebXML est basé sur un ensemble de concepts comme :

- Les processus d'affaires (BP) : la définition d'un processus d'affaire consiste à étudier son scénario de réalisation ainsi que la séquence des messages qui aide à gérer ce processus. Ces processus seront modélisés sous forme des diagrammes UML (diagramme de cas d'utilisation et de séquences). Le modèle d'information et de processus d'affaires ebXML permet aux partenaires commerciaux de saisir les détails de leurs scénarios d'affaires. Un processus d'affaires décrit la relation entre des partenaires commerciaux, leurs rôles et les transactions d'affaire effectuées avec d'autres partenaires.
- Profil de collaboration (CPP) : il consiste à déterminer les processus d'affaire qu'un partenaire commercial peut supporter. Ce document contient les informations d'un partenaire, les processus supportés, les besoins, etc.
- Accord de collaboration (CPA) : il représente un agrément mutuel entre deux partenaires commerciaux sur le processus d'échanges électroniques de données dans le cadre d'ebXML. Ce document décrit l'interaction entre deux CPP et les besoins des partenaires. Il contient les informations sur la définition des interactions entre les deux parties qui s'engagent sur le type de collaboration. Dans cet exemple, nous montrons le processus de construction d'un CPA. Soit un acheteur A qui interroge le registre ebXML et découvre un partenaire commercial B comme vendeur. Tout d'abord, il télécharge le CPP du vendeur B dans son serveur. Ensuite, il crée un CPA(A,B) qui sera communiqué avec le vendeur B. Enfin, une fois qu'ils sont d'accord, les deux partenaires A et B s'engagent pour l'exécution de la collaboration.
- Les composants de données communs (CDC) : ils représentent l'ensemble des informations sémantiques que les partenaires partagent. Ces informations seront communiquées dans les documents échangés au cours d'un processus d'affaires. Un processus d'affaires décrit la relation entre des partenaires commerciaux, leurs rôles et les transactions d'affaires effectuées avec d'autres partenaires. Chaque

transaction d'affaires est réalisée par un échange électronique de documents d'affaires contenant un ensemble de composants communs.

- Un registre des données : le registre ebXML est un entrepôt des informations soumises par un organisme. Le contenu de l'information peut être un schéma XML et des documents, des descriptions de processus, des modèles UML, etc. Il fournit un ensemble de services qui permettent le partage d'informations entre les parties intéressées. Ces services sont définis dans le registre de spécification de services (ebRS) [5], en ce qui concerne l'accès au registre et la recherche des informations des partenaires à travers des requêtes soumises au registre. Les informations des partenaires seront identifiées par des identificateurs uniques générés automatiquement par le registre. En plus de ebRS, ebRIM fournit un schéma de haut niveau pour le registre ebXML. Il fournit des informations sur le type de métadonnées qui sont stockées dans le registre ainsi que les relations entre les classes de métadonnées [15].
- Une couche de transport et de routage de données : l'envoi, la réception, la notification et l'interrogation des registres sont faits par un service de messagerie ebXML incluant des règles de sécurité.

La meilleure façon de décrire la technologie ebXML est de proposer le scénario d'un cas d'utilisation entre deux partenaires commerciaux. Ce cas d'utilisation commence par la configuration des interfaces ebXML et aboutit à l'exécution d'une transaction d'affaire simple. Avant de passer à la phase d'utilisation d'ebXML, il existe une démarche d'intégration du standard dans les entreprises voulant communiquer. Cette phase est nommée phase de conception passant par la modélisation et le profilage. D'une part, la modélisation consiste à définir les processus, les messages ainsi que les données à communiquer. Ces modèles de processus métiers (BP) seront modélisés, en adoptant la norme UML, sous forme des diagrammes de cas d'utilisation et de séquences. Ces derniers vont être stockés dans les registres ebXML. D'autre part, le profilage consiste à déterminer les processus supportés par l'entreprise comme l'achat et la vente.

La figure 4.2 donne une vision de l'architecture technique d'un système ebXML. Elle

montre un exemple d'échange utilisant la plateforme ebXML et les étapes nécessaires pour configurer puis déployer des applications ebXML.

1. Une compagnie A est informée qu'un registre ebXML est accessible sur Internet.
2. Cette compagnie, après avoir consulté les contenus du registre ebXML, décide de construire et de déployer son propre système ebXML.

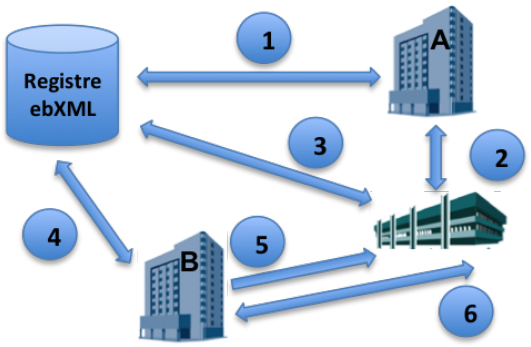



Figure 4.2 – L'approche ebXML- Automatisation des échanges interentreprises.

3. La compagnie A soumet les informations de son entreprise, appelée Profil d'affaires, au registre ebXML. Le profil soumis à la base de registre ebXML décrit les capacités ebXML de l'entreprise, ainsi que ses scénarios d'affaires supportés. Les scénarios d'affaires supportés par la compagnie A sont déclarés sous forme de versions des documents XML et blocs d'information qui leur sont associés (p.e les règles de calcul d'une taxe sur les ventes). Après vérification de la conformité des formats et des descriptions des règles d'utilisation du scénario d'affaires, un accusé de réception est envoyé par le registre à la compagnie A.
4. La compagnie B découvre les scénarios d'affaires supportés par la compagnie A en consultant le registre ebXML. La compagnie B envoie à la compagnie A une notification pour établir une relation d'échange électronique ebXML en utilisant un scénario supporté par la compagnie A.
5. Avant de s'engager dans l'exécution du scénario, la compagnie B envoie à l'interface ebXML de la compagnie A une proposition d'accord d'échange. L'accord

d'échange proposé montre que les deux compagnies s'engagent mutuellement sur les scénarios d'affaires.

6. La compagnie A accepte l'échange. Les compagnies A et B sont alors prêtes à réaliser des échanges électroniques utilisant ebXML.

Le cadre ebXML est en avance, à certains égards. Il a recueilli le soutien de consortiums de l'industrie et les fournisseurs par exemple UN/CEFACT et OASIS, mais les implementations semblent rares. `freebXML`[6] est une initiative qui vise à favoriser le développement et l'adoption de la technologie ebXML. Elle essaie de créer un site centralisé pour le partage du code ebXML «libre» et les applications de mise en oeuvre de son architecture. Une solution ebXML est l'initiative `Oracle WebLogic` [7] qui échange des messages d'affaires avec les partenaires commerciaux en utilisant le protocole d'affaires ebXML. Il existe aussi `IBM WebSphere Partner Gateway V6.1`[8] qui a ajouté des nouvelles fonctionnalités, des performances et des améliorations d'utilisabilité. Il s'agit notamment de l'intégration de l'architecture ebXML Messaging Service (ebMS) et CPA utilisé au Canada, en Europe et en Asie dans le domaine de la santé et l'industrie des automobiles. `IBM WebSphere Partner Gateway` est un produit entièrement conçu pour l'échange de documents entre les partenaires d'affaires. Ici au Canada, il y a le projet en cours du gouvernement d'Ontario qui tente d'intégrer l'architecture ebXML dans le e-governement[9]. En plus, il y a la société torontoise Semantion[10] et la société GetMate[11] qui s'intéressent au développement des outils qui supporte ebXML.

Actuellement, le marché n'est probablement pas prêt encore pour les registres ebXML et la communication automatisée entre les partenaires commerciaux, car une intégration entre les partenaires est encore difficile à atteindre. Les descriptions des processus d'affaire existantes dans le cadre d'ebXML semblent en effet assez généraux et ne permettent pas de réduire l'incertitude et la confusion. Plutôt, ils n'offrent pas une défi-

[6] http ://www.freebxml.org/
[7] http ://docs.oracle.com/cd/E14981-01/wli/docs1031/tpintro/ebxml.html
[8] http ://www-01.ibm.com/software/integration/wspartnergateway/whatsnew/details.html
[9] http ://www.ebxml.org/case_studies/
[10] http ://www.ebxmlsoft.com/
[11] http ://xeni.co.kr/GetMate_info/GetMate_is.html

nition spécifique des processus d'affaires, mais fournissent simplement aux partenaires commerciaux, les moyens et les outils pour réaliser les processus d'affaires. Alors que, RosettaNet définit les processus, les documents commerciaux et de la messagerie d'une façon détaillée. Les spécifications de RosettaNet sont plus concrètes par la description détaillée des processus d'affaire.

## 4.3 Présentation de notre approche

La multitude des standards d'affaire basés sur XML avec chacun leur modèle de document, définis en XML Schema ou en DTD, complique leur intégration. XML a représenté une bonne solution aux entreprises pour la modélisation et la structuration de leurs documents d'affaire mais au niveau syntaxique. En effet, les standards utilisant le langage XML pour l'implémentation de leurs modèles de documents se concentrent sur leur cohérence syntaxique plutôt que sur leur contenu sémantique. Il est en effet difficile de représenter la sémantique et les relations entre les concepts dans les documents échangés.

Le but de notre travail est de réduire l'hétérogénéité dans les documents échangés et de rendre la sémantique des documents exploitable par les systèmes des entreprises. L'approche proposée comprend deux principales étapes : (1) l'ontologisation des documents définis par les standards d'affaire et (2) l'application de l'alignement entre les ontologies des documents des standards d'affaire.

La première étape consiste à intégrer la sémantique au sein des standards d'affaire. Dans cette étape, nous nous intéressons principalement à l'étape d'encodage des modèles de documents qui représente la dernière étape du processus d'ingénierie de documents. Cette étape consiste à transformer le modèle conceptuel, qui met en évidence les relations sémantiques entre les classes (p.e. diagramme de classes), en un modèle physique qui décrit l'ensemble de ces classes (p.e. une base de données, un XML Schema, etc.). Les modèles physiques définis en XML Schema ou DTD ne mettent pas en évidence les relations sémantiques entre les classes représentées dans le modèle conceptuel. XML Schema ne permet de définir que la structure et le type de contenu d'un document XML.

Cette définition permet notamment de vérifier que la validité syntaxique de ce document. Par conséquent, il est nécessaire d'utiliser un langage plus expressif (p.e. OWL) permettant d'encoder toutes les relations définies dans le modèle conceptuel d'un document. OWL supporte la représentation d'un domaine de connaissance en classes, propriétés et instances pour l'utilisation dans un environnement distribué comme le web.

L'approche proposée consiste à traduire les modèles de documents de chaque standard d'affaire encodés par des schémas XML ou DTD en des ontologies OWL. La transformation des modèles de documents décrits en XML Schema vers des ontologies OWL via un processus automatisé offre un avantage important qui peut réduire l'effort humain nécessaire lors de la conception d'une ontologie en plus d'avoir une représentation sémantique des concepts d'un domaine. Nous proposons une feuille de transformation XSLT, présentée à la section 5.3, basée sur un ensemble de règles de transformation des constructions définies en XML Schema vers des constructions en OWL sémantiquement équivalentes. Certaines règles sont inspirées de la littérature.

XML ne définit aucune contrainte sémantique et ne permet pas d'exprimer toutes les contraintes pour la modélisation de toute la connaissance. Il offre un vocabulaire pour représenter la structure, p.e. l'ordre des éléments dans le document, et les types de données, mais ne permet pas de modéliser certaines relations sémantiques de type `subClassOf`. Notre processus de transformation ne traite pas certaines relations sémantiques importantes, comme `subClassOf`, c'est pourquoi nous effectuons un regroupement des classes partageant certaines propriétés pour améliorer la qualité, la lisibilité et la représentation des ontologies [28]. Nous utilisons l'analyse formelle de concepts (FCA) [16] pour classifier au sein d'un ensemble d'objets ceux partageant un certain nombre de propriétés [29]. L'utilisation de FCA sera décrite à la section 5.5.

Dans notre cas, le résultat de notre transformation est un ensemble d'ontologies qui représentent chacune un document d'affaire avec leur propre structure et vocabulaire. Il n'y a pas d'ontologie commune ou de vocabulaire commun (thésaurus distributionnel) pour gérer la communication entre les entreprises et aider à éliminer les problèmes d'hétérogénéité. L'hétérogénéité provient du fait que les ontologies sont créées de manière indépendante avec chacune leur vocabulaire même si elles traitent du même domaine

d'intérêt. Ce problème complique l'interopérabilité et la communication entre les partenaires d'affaire. Comme il nous semble impossible de convaincre toutes les entreprises de modifier leur modèle conceptuel de documents pour avoir un modèle commun, il faut développer une solution qui utilise les modèles existants et qui cherche un lien entre eux.

L'alignement des ontologies permet dans une certaine mesure de surmonter le problème de l'hétérogénéité sémantique [46]. L'alignement est un processus qui prend en entrée deux ontologies et trouve des liens sémantiques ou un ensemble de correspondances entre les entités de ces ontologies (des classes, des propriétés, des instances, etc.).

Dans la deuxième étape, nous tirons avantage des recherches dans ce domaine et montrons comment ces techniques d'alignement aident les entreprises à mieux communiquer en trouvant des correspondances entre leurs ontologies d'affaires respectives. Nous appliquons des techniques d'alignement définies dans la littérature sur des ontologies modélisant les documents d'affaires qui traitent la gestion de bons de commande dans le cadre des standards cXML, xCBL et RosettaNet afin de déterminer des liens sémantiques entre elles. L'absence d'un thésaurus distributionnel contenant la description des termes du domaine ainsi que l'absence des instances de documents nous obligent à nous limiter à des expérimentations basées sur des méthodes terminologiques et structurelles. Nous utilisons deux méthodes terminologiques, une à base de caractères comme la distance Levenshtein appliquée sur les noms des entités, et une autre à base de tokens comme la technique `TFIDF` utilisant les noms et les commentaires des entités des ontologies. Ces techniques sont les plus utilisées par les systèmes d'alignement présentés dans la littérature [14].

Ensuite, nous évaluons la performance d'un système d'alignement basé sur la structure, nommé $\text{OLA}_2$, afin de voir s'il y a une amélioration apportée par la structure dans la performance par rapport aux méthodes terminologiques utilisées. Ce système est considéré parmi les meilleurs algorithmes d'alignement sur les benchmarks.

Les résultats d'alignement sont comparés avec des alignements produits manuellement en utilisant les métriques de précision, rappel et F-score. Les résultats de notre expérimentation sont détaillés à la section 6.7.

## 4.4 Conclusion

Dans ce chapitre, nous avons présenté les travaux d'ingénierie de documents basés sur XML. La structuration, l'interopérabilité, et le traitement de la sémantique de données des processus d'affaire étaient les buts principaux de l'ensemble des travaux étudiés dans ce chapitre. Les entreprises et les organismes ont souffert des problèmes des EDI traditionnels surtout au niveau des processus d'échange. L'apparition de XML était une solution partielle. Son apparition a poussé de grandes entreprises à s'investir pour créer des standards d'échange basés sur XML, comme RosettaNet et ebXML. L'étude faite sur ces travaux a démontré qu'ils souffrent de certaines limites surtout la sémantique et l'hétérogénéité des échanges. Au chapitre suivant, nous décrivons les solutions apportées par la notion d'ontologie et les propriétés du langage OWL par rapport à XML.

# CHAPITRE 5

# INTÉGRATION DE LA SÉMANTIQUE DANS LES STANDARDS D'AFFAIRE

## 5.1 Introduction

L'hétérogénéité dans l'échange de documents d'affaire présente un problème majeur dans la communication entre les entreprises. Une solution est de donner à l'ordinateur un meilleur accès à la sémantique de l'information. En se basant sur XML, des standards comme xCBL, ebXML et RosettaNet vus au chapitre 4, ont été développés pour l'organisation des affaires inter-entreprises et la gestion de documents circulant dans les processus d'affaires. Malgré les pistes que le langage XML a offert au niveau de l'organisation et de la structuration, la représentation de la sémantique de données n'est pas encore possible et il réside dans le niveau syntaxique. Actuellement, la plupart des standards d'affaire existants, telles que RosettaNet, ebXML, xCBL et cXML, sont toujours basés sur XML.

Certaines recherches ont envisagé l'adoption des technologies du web sémantique dans le domaine des affaires.

Lytras et al. [32] suggèrent d'adopter les techniques du web sémantique dans le domaine des affaires afin d'améliorer l'interopérabilité et la représentation des connaissances. L'utilisation des ontologies dans la représentation de la sémantique aide à surmonter le problème de l'hétéréogénéité entre les standards d'affaire grâce aux techniques d'alignement d'ontologies. En plus, l'utilisation d'ontologies fournit un support d'interrogation et de raisonnement pour extraire de nouvelles connaissances à partir des données.

GoodRelations[1] et eCl@ssOWL[2] sont deux ontologies OWL pour le commerce électronique décrivant les types et les propriétés des produits. eCl@ssOWL est utilisé avec GoodRelations pour couvrir les aspects commerciaux de l'offre et de la demande, p.e. les prix, le paiement, ou la livraison. Vu l'importance de la sémantique dans le domaine

[1] http ://www.heppnetz.de/projects/goodrelations/

[2] http ://www.heppnetz.de/projects/eclassowl/

des affaires, des travaux ont été effectués sur les standards d'affaire pour l'intégration de la sémantique. À notre connaissance, il n'existe pas, à date, d'ontologie modélisant les documents de ces standards. Il y a des initiatives qui se basent sur des langages non recommandés par l'organisme W3C ou bien des ontologies construites manuellement sans tirer profits des efforts consacrés à la modélisation XML de ces standards.

Dans ce chapitre, nous proposons un outil automatique de transformation des documents d'affaire définis dans les standards d'affaire existants basés sur XML Schema et DTD vers une représentation ontologique [27] dont le processus sera détaillé à la section 5.3 en utilisant une étude de cas du standard d'affaire RosettaNet. Notre objectif est de rendre le transfert plus transparent et de récolter les fruits des efforts mis dans le développement de ces normes d'affaires définis au chapitre 4.

Notre approche s'intéresse à l'étape d'encodage des modèles de documents, la dernière du processus d'ingénierie de documents. Elle consiste à transformer le modèle conceptuel, qui met en évidence les relations sémantiques entre les classes (p.e. diagramme de classes), en un modèle physique qui décrit l'ensemble de ces classes (p.e. une base de données, un XML Schema, etc.). Les modèles physiques définis en XML Schema ou DTD ne mettent pas en évidence les relations sémantiques entre les classes représentées dans le modèle conceptuel. XML Schema permet de définir que la structure et le type de contenu d'un document XML et non pas sa sémantique. Cette définition permet de vérifier que la validité syntaxique de ce document. Par conséquent, il est nécessaire d'utiliser un langage plus expressif (p.e. OWL 2) permettant d'encoder toutes les relations définies dans le modèle conceptuel d'un document. OWL supporte la représentation d'un domaine de connaissance en classes, propriétés et instances pour l'utilisation dans un environnement distribué comme le web. Les différences entre OWL 2 et XML Schema ont été décrites à la section 3.4.

XML offre un vocabulaire pour représenter la structure, p.e. l'ordre des éléments dans le document, et les types de données. Il est limité syntaxiquement puisqu'il ne définit aucune contrainte sémantique et ne permet pas de modéliser certaines relations sémantiques de type `subClassOf`. Par conséquent, notre processus de transformation ne traite pas certaines relations sémantiques, comme `subClassOf`. Dans ce cas, nous

utilisons l'analyse formelle de concepts (FCA) [16] pour classifier au sein d'un ensemble d'objets ceux partageant un certain nombre de propriétés [29]. Cette approche améliorera la qualité, la lisibilité et la représentation des ontologies [28]. L'utilisation de FCA sera décrite à la section 5.5. La section 5.7 conclut le chapitre où nous présentons quelques perspectives.

## 5.2 État de l'art sur l'intégration de la sémantique

Nous avons étudié différentes approches utilisant le langage XML dans le domaine d'ingénierie de documents et la représentation des processus d'affaire. Mais ces standards, définis en XML Schema et DTD, ne traitent pas la sémantique de l'information. Pour cela, nous devons non seulement stocker les métadonnées du document (par exemple, auteur, titre, etc), mais nous devons aussi mettre à la disposition des machines les concepts plus importants, les relations entre eux ou avec d'autres documents, etc.

Dans le cadre de ses travaux sur le web sémantique, le W3C a publié en 2004 une recommandation définissant le langage OWL, afin de modéliser des ontologies utilisables et échangeables sur le web. Une ontologie est la définition formelle de la description d'un domaine de connaissance. OWL décrit la structure d'un domaine en termes de classes et de propriétés. Les classes peuvent être des noms (URI) ou des expressions et supporte un ensemble de propriétés.

En utilisant les modèles sémantiques basés sur les ontologies, les entreprises acquièrent plusieurs avantages tels que la possibilité d'interroger leurs bases de connaissances et de faciliter le partage d'information [3]. L'intégration de la sémantique est une des directions de recherche prometteuses pour améliorer l'intégration des applications d'échange, se basant sur la sémantique des données, au sein des entreprises. Plusieurs standards et langages ont été proposés, tels que WSMO [40] utilisé par [31], et OWL-S [53], définissant des frameworks et des langages formels pour les services web basés sur la sémantique. Ces langages ne sont pas recommandés par l'organisme W3C, ce qui complique leur adoption par les systèmes des entreprises.

Les standards d'affaire ont fait l'objet de nombreuses études pour intégrer la séman-

tique. Dogac et al. [11] [12] et Schulte et al. [44] ont examiné la façon d'enrichir les registres ebXML par des ontologies pour en décrire la sémantique. Dans le cadre du travail de Dogac et al. [11], la représentation des ontologies n'est pas faite en utilisant la syntaxe OWL, ce qui complique les raisonnements sur l'ontologie ou la vérification de sa consistance. Il reste beaucoup à faire d'autant plus que les modèles ne sont pas rédigés en utilisant la syntaxe OWL. Il laisse l'application des raisonnements et la vérification de l'ontologie, une étape difficile qui nécessite une migration vers la syntaxe OWL. La réalisation des raisonnements sur l'ontologie aide à déduire des nouvelles informations cachées et des relations entre les concepts dans les documents. À mon avis, la modification du registre ebXML pour soutenir OWL est nécessaire afin d'avoir une capacité de raisonnement. Dogac et al. [12] ont déclaré que cette approche nécessite des changements considérables dans l'architecture de registre. Ils ont tenté d'intégrer les ontologies OWL-S dans le registre ebXML, mais la capacité de raisonnement en utilisant les OWL reasonners (p.e. Racer et Pellet) n'est pas offerte jusqu'à date. En dépit de l'effort d'adaptation, beaucoup de travail reste à faire, en particulier sur les raisonneurs.

La transition vers les ontologies a été l'objet d'études effectuées sur RosettaNet. Kotinurmi et al. [31] ont essayé d'intégrer la notion d'ontologie au niveau de processus d'échange en se basant sur le langage WSML [40] qui reste toujours au niveau du prototypage. Cette approche utilise une couche d'adaptation qui transforme des documents XML en WSML et vice versa, entre deux partenaires dans le cadre RosettaNet. Haller et al. [19] ont proposé une méthodologie de transformation des PIPs de RosettaNet définis en XML Schema vers WSML. Sans donner de détails de mise en œuvre, ils ne portaient que sur 50 PIPs définis en XML Schema, et pas avec ceux en DTD. En plus, WSML est limité aux services Web et n'a jamais passé le stade de la soumission à l'organisme W3C depuis 2005, par contre OWL, un langage d'ontologie plus générique recommandé par le W3C, fournit une meilleure interopérabilité et offre la possibilité d'exploiter plusieurs techniques d'alignement.

Parmi les initiatives basées sur les langages recommandés par l'organisme W3C. RosettaNet Ontology [3] est une représentation OWL du PIP3A4 de RosettaNet construite

[3] http ://lsdis.cs.uga.edu/projects/meteor-s/wsdls/ontologies/rosetta.owl

manuellement qui n'a pas beaucoup de détails et qui s'appuie sur des propriétés non définies dans la représentation XML, p.e. `has_price_n_availability_line_item`. Cette ontologie ne résoud pas le problème de l'incertitude et de l'ambiguïté. Ainsi, les entreprises qui utilisent RosettaNet ont besoin de plus d'efforts pour adopter les nouveaux termes définis dans cette ontologie.

Pour cette raison, il est crucial de débuter avec des documents existants définis en XML Schema afin de conserver la même liste de concepts et de termes définis. Nous proposons une transformation automatique de représentations DTD et XML Schema des standards d'affaire vers des ontologies OWL. Certaines approches similaires à la nôtre ont été proposées : ReDeFer[4], XS2OWL[5] et JXML2OWL [39].

García et al. [17] proposent une approche automatique de transformation de XML Schema vers OWL, nommée ReDeFer, pour transformer ebXML, l'une des principales normes de B2B, vue à la section 4.2.4, qui modélise les processus d'affaire [17]. Tsinaraki et al. [52] propose une approche similaire de transformation des standards multimédia comme MPEG-7 et MPEG-21, nommée XS2OWL. ReDeFer et XS2OWL sont deux approches similaires qui reposent sur des feuilles de transformation XSLT, chacune étant basée sur un ensemble de règles de transformation des constructions en XML Schema vers des constructions en OWL équivalentes sémantiquement.

Toutefois, certains aspects ne sont pas traités, par exemple les propriétés ayant des valeurs énumérées et les annotations. Le portail OntologyDesignPatterns.org[6] décrit les expériences et les problèmes rencontrés avec l'outil ReDeFer.

D'une part, les énumérations en XML Schema définissent les valeurs possibles d'une propriété qui doivent être prises en compte lors du processus de transformation puisqu'elles représentent une information importante dans les processus d'affaire, p.e. les conventions d'écriture des codes des monnaies mondiales utilisées par les entreprises.

En plus, les annotations permettent la documentation et la définition des éléments. Elles sont utilisées par les standards d'affaire pour décrire la sémantique des éléments contenus dans les documents d'affaire ce qui nous semble une information importante

[4] ReDeFer, http ://rhizomik.net/redefer
[5] http ://www.music.tuc.gr/projects/sw/xs2owl/
[6] http ://ontologydesignpatterns.org/ont/fao/figis/speciesNTK.owl

surtout pour des raisons de maintenance ou de compréhension.

D'autre part, les documents des standards d'affaire reposent sur un ensemble d'éléments appartenant à différents espaces de noms ; leur transformation nécessite de tenir compte les préfixes associés à ces espaces de noms, ce qui n'est pas fait par ces autres systèmes. Comme les standards d'affaire comportent un nombre important d'`import` qui sont utilisés pour inclure plusieurs schémas externes avec différents espaces de noms qui doivent être pris en compte lors de la transformation.

Nous avons testé ces outils sur les standards d'affaire xCBL, cXML et RosettaNet, mais le résultat obtenu n'était pas valide ni syntaxiquement ni sémantiquement dû à la complexité des schémas. La consistance a été évaluée en utilisant le raisonneur Pellet [48].

Avec ReDeFer [17] et XS2OWL [52], les `xs:sequence` et `xs:choice` sont représentés en OWL respectivement par `xs:intersectionOf` et `xs:unionOf` et des classes anonymes qui sont produites pour les représenter malgré qu'une classe en OWL est identifiée par son URI en concaténant l'URI de l'ontologie avec le nom de la classe. Cette approche ne nous semble pas appropriée pour notre application même si ce type de transformation est "standard" dans la théorie classique des types. En outre, l'une des limites majeures de ces systèmes est qu'ils ne prennent pas en compte qu'un élément peut être défini dans un fichier externe ayant un espace de noms différent de celui en cours de transformation.

Rodrigues et al. [39] ont développé JXML2OWL, un outil de transformation manuel et interactif permettant à l'utilisateur de trouver des correspondances entre les éléments d'un document défini en XML Schema et DTD avec des concepts (classes et propriétés) d'une ontologie existante définie en OWL dans le but d'automatiser le processus d'intégration des données sémantiques. Selon l'alignement réalisé, l'outil génère des règles de transformation en XSLT qui permettent la transformation automatique de toutes les instances de ce document XML vers l'ontologie. Cet outil est utilisé pour l'intégration de la sémantique pour le standard d'affaire cXML [51]. Les problèmes de cette approche sont liés aux règles de projection qui sont créées manuellement par un expert de l'entreprise sans détailler les critères utilisés pour les construire. Dans le cas du PIP3A4 de

RosettaNet décrivant un bon de commande, la projection est coûteuse en temps puisque le document contient beaucoup d'éléments. Il n'y a pas de processus de validation des règles créées par l'expert, on peut donc s'interroger sur leur pertinence.

## 5.3 Processus de transformation

Notre approche est basée sur différentes règles de projection pour transformer les éléments définis dans un document en XML Schema vers OWL 2. Ces règles de transformation ont été inspirées des travaux de García et al. [17] et Tsinaraki et al. [52]. Notre approche permet d'obtenir une transformation des concepts définis dans chaque document de standards d'affaire en incluant le détails manquants dans les approches de transformation vues à la section 5.2. Cette approche a été appliquée sur :

- RosettaNet : 112 PIPs, où 90 sont définis en XML Schema et 22 en DTD, disponibles pour téléchargement parmi les 132 PIPs dans le répertoire des PIPs.Ils sont transformés en 112 ontologies, chacune représentant un PIP.

- xCBL : 44 documents en XML Schema transformés en 44 ontologies décrivant chacune un segment de xCBL.

- cXML : 2 documents définis en DTD qui sont destinés à la gestion de bons de commande (demande et confirmation).

- ebXML : 2 documents définis en XML Schema (ebBP et ebCPPA). À titre de rappel, ebCPPA représente l'accord entre deux partenaires commerciaux contenant les informations de deux partenaires avec ceux des processus supportés. Par contre, ebBP contient la définition d'un processus d'affaire et son scénario de réalisation.

Une vue d'ensemble de l'architecture de notre système est esquissé à la figure 5.1. Le cœur de notre système est composé de deux parties : DTD2XSD et XSD2OWL. Le système prend en entrée un document défini en DTD ou XML Schema qui va être transformé en une ontologie OWL à l'aide du feuille de transformation XSLT 2.0. XSLT (Ex-

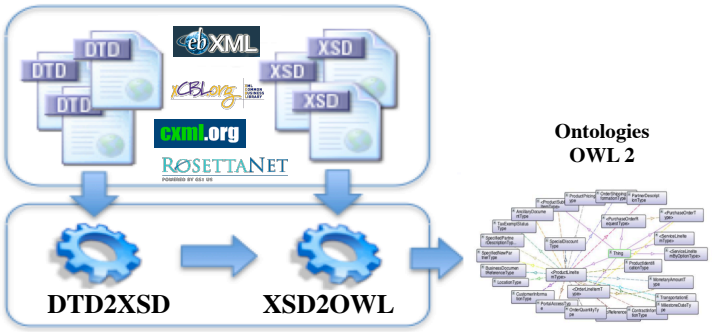



Figure 5.1 – Approche de transformation.

tensible Stylesheet Language Transformations) est un langage de transformation d'un document XML vers un autre document XML ou un autre format comme HTML.

### 5.3.1 XSD2OWL

Le processus de transformation de XSD2OWL est basé sur la traduction des constructions définies en XML Schema vers celles en OWL sémantiquement équivalentes, présentée au tableau 5.I. Ces règles de transformation ont été inspirées des travaux de García et al. [17] et Tsinaraki et al. [52].

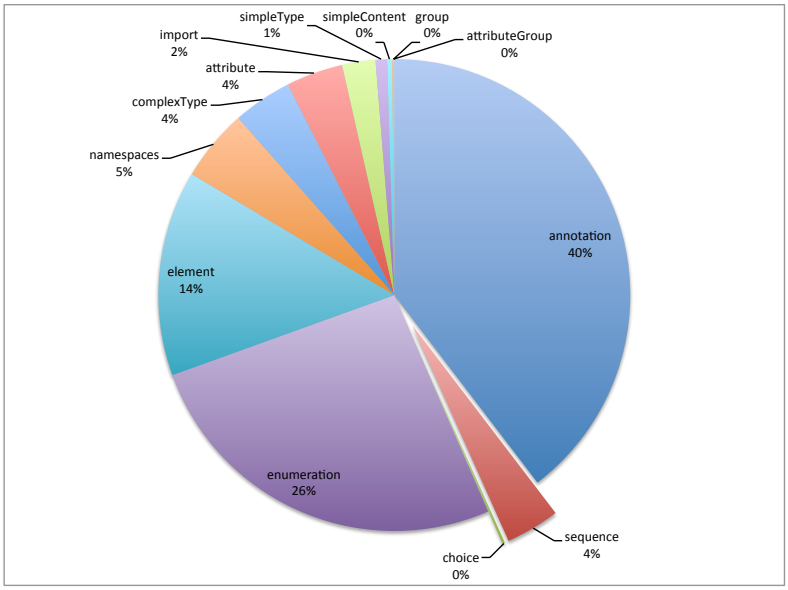



Figure 5.2 – Vue globale des composants XML Schema utilisés dans la définition des documents des standards d'affaire.

La figure 5.2 représente une analyse statistique des composants XML Schema utilisés dans la définition des documents des standards d'affaire. Elle donne une liste des composants qui devraient être pris en compte lors du processus de transformation, d'où la liste des règles proposées au tableau 5.I.

Tableau 5.I – La représentation des requêtes XPath de chaque constructeur XML Schema et leur traduction en OWL 2.

| | XPath XML Schema | OWL 2 |
|---|---|---|
| **R1** | `complexType`\|`attributeGroup`\|`group`\|`element//complexType` | `owl:Class` |
| **R2** | `complexType//element` | `owl:DataProperty` `owl:ObjectProperty` |
| **R3** | `element/complexType/simpleContent` | `owl:DataProperty` |
| **R4** | `element@type` | `owl:DataPropertyRange` `owl:ObjectPropertyRange` |
| **R5** | `complexType//attribute` | `owl:DataProperty` |
| **R6** | `attribute@type` | `owl:DataPropertyRange` |
| **R7** | `enumeration` | `owl:DataOneOf` |
| **R8** | `@minOccurs` | `owl:AnnotationAssertion` (`owl:minCardinality`) |
| **R9** | `@maxOccurs` | `owl:AnnotationAssertion` (`owl:maxCardinality`) |
| **R10** | `annotation` | `owl:AnnotationAssertion` (`rdfs:comment`) |
| **R11** | `import` | `owl:Import` |
| **R12** | `namespace` | `owl:Prefix` |

Le moteur de transformation, XSD2OWL, est basé sur XSLT et XPath, des langages conçus pour naviguer dans un document XML. Chaque expression XPath fait référence à un élément XML dans le document. Les noms d'éléments et les espaces de noms définis dans le schéma XML sont conservés dans la représentation OWL de ce document.

Chaque ligne du tableau 5.I contient la traduction de chaque construction en XML Schema vers l'équivalente en OWL, où la première colonne représente le numéro de la règle de transformation associée à chaque traduction, la deuxième colonne représente la requête XPath identifiant l'ensemble de noeuds définissant un élément dans le XML

Schema, p.e la requête `complexType//attribute` renvoie les attributs définis dans un `complexType`, et la troisième colonne contient sa représentation en OWL.

Selon la nature et l'organisation des documents étudiés, nous avons inclus les règles **R7**, **R10**, **R11** et **R12** qui n'ont pas était définis dans les travaux de la littérature. Les règles définies au tableau 5.I traitent les constructions XML Schema comprises dans les schémas XML des documents étudiés. Les énumérations en XML Schema définissent les valeurs possibles d'une propriété qui doivent être prises en compte lors du processus de transformation puisqu'elles représentent une information importante dans les processus d'affaire, p.ex. les conventions d'écriture des codes de la monnaie mondiale utilisés par les entreprises. Selon la figure 5.2, elles sont largement utilisées dans la définition des documents étudiés avec un pourcentage de 26%.

Les `namespace` sont pris en compte, car les documents des standards d'affaire reposent sur un ensemble d'éléments appartenant à différents espaces de noms, d'où leur transformation nécessite donc la conservation des préfixes associés à ces espaces de noms. Ainsi, ces documents comportent un nombre important des `import` qui sont utilisés pour ajouter plusieurs schémas externes avec différents espaces de noms qui doivent être pris en compte lors de la transformation.

Dans les schémas XML, nous distinguons les types simples et complexes. Les types simples ne décrivent que des contenus textuels et utilisables pour les éléments comme pour les attributs. Alors que les types complexes sont constitués d'éléments ou d'un contenu mixte avec du texte et des éléments.

Comme le montre le tableau 5.I, les types complexes dans le XML Schema, définis par des `xs:complexType`, des `xs:attributeGroup` et des `xs:group`, sont transformés en classes OWL définies par l'expression `owl:Class`. Les `xs:attribute` sont représentés par des `owl:DataProperty`, tandis que les `xs:element` deviennent des `owl:DataProperty` ou bien des `owl:ObjectProperty` dans la représentation ontologique.

Les notations utilisées dans la description des règles de tranformation sont décrites dans le tableau 5.II, où chaque ligne représente une notation ainsi que sa signification.

Dans la suite, nous détaillons la transformation de ces artefacts avec leur représenta-

Tableau 5.II – Lettres utilisées dans les règles de XSD2OWL ainsi que leur signification.

| | **représente** |
|---|---|
| **C** | le nom d'un type complexe (`xs:complexType`, `xs:attributeGroup` ou `xs:group`). |
| **E** | le nom d'un élément (`xs:element`). |
| **A** | le nom d'un attribut (`xs:attribute`). |
| **S** | le nom d'un type simple (`xs:simpleType`). |
| **T** | un type prédéfini comme `xs:string`. |
| $\mathbf{t}_i$ | une valeur de type **T**. |
| **N** | un nombre. |

tion en XML Schema (à gauche) et leur transformation en OWL (à droite). Dans cette section, les règles sont illustrées avec la syntaxe OWL/XML [38] mais que la feuille de transformation produit aussi du RDF/XML.

**Transformation des types complexes.** Les types complexes en XML Schema sont définis par des `xs:attributeGroup` , des `xs:group` et des `xs:complexType`.

L'application de **R1** transforme tout élément de type `xs:group` et `xs:attributeGroup` en une classe en OWL introduite par le constructeur `owl:Class`, comme le montre le modèle ci-dessous, où le nom de la classe (`@IRI`) est la valeur contenue dans l'attribut `@name` de l'élément en question.

| | | |
|---|---|---|
| **R1** | `<xs:attributeGroup name="C">`<br>`...`<br>`</xs:attributeGroup>` | `<owl:Declaration>`<br>`<owl:Class IRI="C"/>`<br>`</owl:Declaration>` |

| | | |
|---|---|---|
| **R1** | `<xs:group name="C">`<br>`<xs:sequence/>`<br>`</xs:group>` | `<owl:Declaration>`<br>`<owl:Class IRI="C"/>`<br>`</owl:Declaration>` |

De même, les éléments de type `xs:complexType` sont représentés par des classes dans l'ontologie OWL. L'exemple ci dessous d'application de **R1** sur les `xs:complexType` montre deux manières de définir un type complexe en XML Schema. Dans ce cas, le nom

de la classe, représenté par l'attribut `@IRI`, aura la valeur de l'attribut `@name` contenu dans l'élément `xs:complexType`, si c'est le cas, sinon celui de son parent de type `xs:element`.

| | | |
|---|---|---|
| **R1** | `<xs:complexType name="C">`<br>`  <xs:sequence/>`<br>`</xs:complexType>` | `<owl:Declaration>`<br>`  <owl:Class IRI="C"/>`<br>`</owl:Declaration>` |

| | | |
|---|---|---|
| **R1** | `<xs:element name="E">`<br>`  <xs:complexType>`<br>`  <xs:sequence/>`<br>`  </xs:complexType>`<br>`</xs:element>` | `<owl:Declaration>`<br>`  <owl:Class IRI="E"/>`<br>`</owl:Declaration>` |

**Transformation des `xs:element`.**

La déclaration d'éléments en XML Schema est faite par `xs:element` avec une portée globale, c.à.d un enfant direct de la racine `xs:schema` du document, ou locale c.à.d au sein de la définition d'un type complexe. Un type complexe est défini à l'aide de l'élément `xs:complexType` qui pourra contenir une séquence d'éléments, une série d'attributs, etc. Dans le cas des types complexes, un élément peut être déclaré localement, s'il contient l'attribut `@name`, ou bien en faisant référence, avec l'attribut `@ref`, à un élément de portée globale. En OWL, les éléments seront représentés par des `DataProperty` ou des `ObjectProperty`.

La valeur de l'attribut `@type` détermine le type de la propriété selon les deux cas suivants, comme le montre la règle **R2** :

- `owl:ObjectProperty` : si cet attribut fait référence à un type complexe (**C**) avec un contenu non mixte, c.à.d ne contenant que des éléments (`xs:element` ou `xs:attribute`).

| | | |
|---|---|---|
| **R2** | `<xs:element name="E" type="C"/>` | `<owl:Declaration>`<br>`  <owl:ObjectProperty IRI="E"/>`<br>`</owl:Declaration>` |

- `owl:DataProperty` : s'il réfère à un type simple (**S**) ou prédéfini (**T**).

| | | |
|---|---|---|
| **R2** | `<xs:element name="E" type="S\|T"/>` | `<owl:Declaration>`<br>`  <owl:DataProperty IRI="E"/>`<br>`</owl:Declaration>` |

Alors qu'il peut désigner un élément avec un contenu simple c-à-d avec du texte et/ou des attributs, l'élément `xs:complexType` contient, dans ce cas, un élément de type `xs:simpleContent` qui permet de restreindre la valeur du contenu de l'élément ou d'étendre l'élément avec des attributs. Il existe deux façons pour représenter un élément avec un contenu simple en XML Schema.

En premier lieu, un `xs:element` déclaré à l'aide d'un `xs:complexType` anonyme (sans l'attribut `@name`) avec un contenu de type prédéfini et un attribut. La règle **R3** le transforme en `owl:DataProperty` ayant comme `owl:DataPropertyRange`, le type défini par l'attribut `@base` de l'élément `xs:extension`. Dans le cas où le `xs:complexType` est anonyme, une `owl:Class` sera créée ayant comme nom la concaténation du nom de l'élément défini par l'attribut `@name` et la chaîne de caractère `Type`. Cette classe représentera le `owl:DataPropertyDomain` de cette propriété.

| | | |
|---|---|---|
| **R3** | `<xs:element name="E">`<br>`  <xs:complexType>`<br>`  <xs:simpleContent>`<br>`  <xs:extension base="T">`<br>`  <xs:attribute/>`<br>`  </xs:extension>`<br>`  </xs:simpleContent>`<br>`  </xs:complexType>`<br>`</xs:element>` | `<owl:Declaration>`<br>`  <owl:Class IRI="EType"/>`<br>`</owl:Declaration>`<br><br>`<owl:Declaration>`<br>`  <owl:DataProperty IRI="E"/>`<br>`</owl:Declaration>`<br><br>`<owl:DataPropertyRange>`<br>`  <owl:DataProperty IRI="E"/>`<br>`  <owl:Datatype IRI="T"/>`<br>`</owl:DataPropertyRange>`<br><br>`<owl:DataPropertyDomain>`<br>`  <owl:DataProperty IRI="E"/>`<br>`  <owl:Class IRI="EType"/>`<br>`</owl:DataPropertyDomain>` |

En deuxième lieu, cet élément contient un attribut `@type` qui désigne un élément de type `xs:complexType` avec un attribut `@name` et contenant un `xs:simpleContent`. De manière analogue, la règle **R3** le transforme en `owl:DataProperty`.

| | | |
|---|---|---|
| **R3** | `<xs:element name="E" type="C"/>`<br>`<xs:complexType name="C">`<br>`  <xs:simpleContent>`<br>`  <xs:extension base="T">`<br>`  <xs:attribute/>`<br>`  </xs:extension>`<br>`  </xs:simpleContent>`<br>`</xs:complexType>` | `<owl:Declaration>`<br>`  <owl:DataProperty IRI="E"/>`<br>`</owl:Declaration>`<br><br>`<owl:DataPropertyRange>`<br>`  <owl:DataProperty IRI="E"/>`<br>`  <owl:Datatype IRI="T"/>`<br>`</owl:DataPropertyRange>`<br><br>`<owl:DataPropertyDomain>`<br>`  <owl:DataProperty IRI="E"/>`<br>`  <owl:Class IRI="C"/>`<br>`</owl:DataPropertyDomain>` |

Maintenant, nous discutons le cas où l'attribut `@type` désigne un type personnalisé défini par un `xs:simpleType` ou un type prédéfini dérivant de `anySpecialType` du XML Schema, l'élément sera transformé en `owl:DataProperty`.

Le cas des `xs:simpleType` est plus compliqué que celui des types prédéfinis. Nous rappelons qu'un `xs:simpleType` détermine les contraintes sur les valeurs que les attributs ou les éléments peuvent prendre. Les enumérations est l'une des contraintes servant à limiter le contenu d'un élément XML à un ensemble de valeurs possibles. En OWL, le `owl:DataPropertyRange` est représenté par le constructeur `owl:DataOneOf` contenant un certain nombre de `owl:Literal` pour définir les valeurs possibles de la propriété, comme le montre la règle **R7**. Dans le cas contraire, le `owl:DataPropertyRange` prendra la valeur de l'attribut `xs:simpleType/xs:restriction/@base`, la valeur de cet attribut est un type prédéfini p.e. `xs:string`.

| | | |
|---|---|---|
| **R7** | <xs:element name="**E**" type="**S**"/><br><xs:simpleType name="**S**"><br><xs:restriction base="**T**"><br><xs:enumeration value="$\mathbf{t}_1$"/><br><xs:enumeration value="$\mathbf{t}_2$"/><br></xs:restriction><br></xs:simpleType> | <owl:Declaration><br><owl:DataProperty IRI="**E**"/><br></owl:Declaration><br><br><owl:ObjectPropertyRange><br><owl:DataProperty IRI="**E**"/><br><owl:DataUnionOf><br><owl:DataOneOf><br><owl:Literal ...>$\mathbf{t}_1$<br></owl:Literal><br><owl:Literal ...>$\mathbf{t}_2$<br></owl:Literal><br></owl:DataOneOf><br></owl:DataUnionOf><br></owl:ObjectPropertyRange> |

Avec ReDeFer[7], tous les éléments de type `xs:simpleType` définis par l'utilisateur sont mappés à des `xs:string` afin de garder leurs valeurs lexicales intactes, selon Garcìa et al. [17], bien que cela provoque une perte d'information dans l'ontologie OWL résultante.

Dans le cas des types prédéfinis, le `owl:DataPropertyRange` prendra la valeur de l'attribut `@type` comme le montre la règle **R4**.

| | | |
|---|---|---|
| **R4** | <xs:element name="**E**" type="**T**"/> | <owl:Declaration><br><owl:DataProperty IRI="**E**"/><br></owl:Declaration><br><br><owl:DataPropertyRange><br><owl:DataProperty IRI="**E**"/><br><owl:Datatype IRI="**T**"/><br></owl:DataPropertyRange> |

En XML Schema, il est possible de préciser le nombre minimal et maximal d'occurences d'un élément avec les attributs `@maxOccurs` et `@minOccurs`. La valeur peut être un entier supérieur ou égal à 0 ou la chaîne `unbounded` pour ne pas définir de nombre maximal. En OWL, ils seront définis par un `owl:AnnotationAssertion`

[7] http ://rhizomik.net/html/redefer/

ayant `owl:maxCardinality` et `owl:minCardinality` comme type d'annotation. La syntaxe de déclaration en OWL est représentée dans l'exemple d'application de règles **R8** et **R9**, ci-dessous.

| | | |
|---|---|---|
| **R8** | `<xs:element name="E" minOccurs="N"/>` | `<owl:AnnotationAssertion>`<br>`  <owl:AnnotationProperty`<br>`  abbreviatedIRI="owl:minCardinality"/>`<br>`  <owl:IRI>E</owl:IRI>`<br>`  <owl:Literal ...>N</owl:Literal>`<br>`</owl:AnnotationAssertion>` |

| | | |
|---|---|---|
| **R9** | `<xs:element name="E" maxOccurs="N"/>` | `<owl:AnnotationAssertion>`<br>`  <owl:AnnotationProperty`<br>`  abbreviatedIRI="owl:maxCardinality"/>`<br>`  <owl:IRI>E</owl:IRI>`<br>`  <owl:Literal ...>N</owl:Literal>`<br>`</owl:AnnotationAssertion>` |

**Transformation des `xs:attribute`.** Tous les `xs:attribute` sont transformés en `owl:DataProperty` à l'aide de la règle **R5**. Un attribut est identifié par son nom, défini par l'attribut `@name` ainsi que le type de données supporté à l'aide de l'attribut `@type` qui représente le `owl:DataPropertyRange` par la règle **R6**. Le processus de transformation est identique à celui du `xs:element` lorsque l'attribut `@type` désigne un type personnalisé défini par un `xs:simpleType` ou un type prédéfini dérivant de `anySpecialType` du XML Schema.

| | | |
|---|---|---|
| **R5-R6** | `<xs:attribute name="A" type="T"/>` | `<owl:Declaration>`<br>`  <owl:DataProperty IRI="A"/>`<br>`</owl:Declaration>`<br><br>`<owl:DataPropertyRange>`<br>`  <owl:DataProperty IRI="A"/>`<br>`  <owl:Datatype IRI="T"/>`<br>`</owl:DataPropertyRange>` |

Il est possible que les attributs `@ref` ou `@type` désignent des entités appartenant à des espaces de noms différents de celui du fichier en cours de transformation, défini par l'attribut `@targetNameSpace`.

```
<xsd:element ref="dp:ContractInformationType" ... />
<xsd:element name="CountryOfOrigin" type="uc:CountryType" ... />
```

Cet exemple représente un extrait du `PurchaseOrderRequest_02_05.xsd` du RosettaNet, où l'attribut `@ref` de l'élément `dp:ContractInformationType` désigne un élément externe défini dans le document `Procurement_02_27.xsd`, tandis que l'attribut `@type` de l'élément `CountryOfOrigin` fait référence à un `xs:simpleType` introduit dans le document `ISO_Country_01_02.xsd`. Alors, nous récupérons les chemins d'accès à ces fichiers contenant les entités `dp:ContractInformationType` et `uc:CountryType`, et par la suite déterminons le type (data ou object) des propriétés `CountryOfOrigin` et `dp:ContractInformationType`.

Techniquement, nous retrouvons l'espace de noms, défini dans la racine `xs:schema` du document, associé aux éléments `dp:ContractInformationType` et `uc:CountryType` en fonction de leur préfixe (`xmlns:dp` et `xmlns:uc`), comme le montre la figure 5.3 où le texte en gras correspond à l'espace de nom de l'élément `uc:CountryType` qui est différent à celui du fichier en cours de transformation définis dans l'attribut `@targetNameSpace`. Cet élément était défini dans le fichier `ISO_Country_01_02.xsd`. Ensuite, nous accédons à l'attribut `@schemaLocation` appartenant à l'élément de type `xs:import` où la valeur de son attribut `@namespace` est égale aux espaces de noms de ces éléments.

```
<xs:schema targetNamespace=
"urn:rosettanet:specification:interchange:PurchaseOrderRequest:xsd:schema:02.05"
xmlns:dp="urn:rosettanet:specification:domain:Procurement:xsd:schema:02.29"
xmlns:uc="urn:rosettanet:specification:universal:Country:xsd:codelist:01.02" >
<xs:import
namespace="urn:rosettanet:specification:domain:Procurement:xsd:schema:02.29"
schemaLocation="../Domain/Procurement/Procurement_02_29.xsd" />
<xs:import
namespace="urn:rosettanet:specification:universal:Country:xsd:codelist:01.02"
schemaLocation="../Universal/CodeList/ISO_Country_01_02.xsd" />
```

Figure 5.3 – Exemple de récupération des chemins d'accès aux fichiers correspondant aux éléments appartenant à un espace de noms externe. L'espace de nom du fichier en cours de transformation est identifié par l'attribut `@targetNameSpace`.

Dans la représentation ontologique, nous conservons les mêmes noms des éléments ainsi que les types de données définis dans le document XML. Un préfixe est ajouté au nom d'une propriété, dans le cas de `owl:DataProperty` et `owl:ObjectProperty`, pour améliorer la lisibilité et respecter les conventions de nommage du W3C pour que le prédicat du triplet RDF prenne la forme d'un verbe, p.e. `cityName` deviendra `has_cityName`. Par contre, aucun ne sera ajouté dans le cas des noms de propriétés contenant l'une de ces prépositions (`Is`, `To`, `By` et `At`) comme token, p.e. `ShipToPartnerSubLine` ou `InstallAt`. L'espace de noms de l'ontologie prendra celui défini dans le XML Schema par l'attribut `@targetNameSpace`.

| | | |
|---|---|---|
| **R10** | `<xs:element name="E">`<br>`  <xs:annotation>`<br>`  <xs:appinfo>`<br>`  <urss:Definition>`<br>`  description`<br>`  </urss:Definition>`<br>`  </xs:appinfo>`<br>`  </xs:annotation/>`<br>`</xs:element/>` | `<owl:AnnotationAssertion>`<br>`  <owl:AnnotationProperty`<br>`  abbreviatedIRI="&rdfs:comment"/>`<br>`  <owl:IRI>E</owl:IRI>`<br>`  <owl:Literal datatypeIRI="...">`<br>`  description`<br>`  </owl:Literal>`<br>`</owl:AnnotationAssertion>` |

La transformation des `xs:annotation` est prise en compte par XSD2OWL, ce qui n'est pas le cas de ReDeFer. Les annotations sont des commentaires ajoutés par les utilisateurs incluant la définition des éléments (`xs:complexType`, `xs:element`, `xs:attribute` ou `xs:enumeration`), la version, etc. Tous les `xs:annotation` sont représentés par des `owl:AnnotationAssertion` ayant `rdf:comment` comme un `owl:AnnotationProperty`. La syntaxe est présentée dans l'exemple d'application de la règle **`R10`**.

Le résultat de l'application des règles de tranformation de notre système XSD2OWL est une ontologie OWL permettant de rendre la sémantique du schéma XML plus explicite. Pour valider nos résultats, nous appliquons deux types de validation (syntaxiques et sémantiques). La validation syntaxique consiste à vérifier la conformité de nos résultats avec le schéma XML pour la sérialisation OWL/XML[8]. Tandis que la validation

[8]http ://www.w3.org/TR/2012/REC-owl2-xml-serialization-20121211/

sémantique consiste à vérifier la cohérence des définitions des entités dans les ontologies résultantes. La cohérence a été vérifiée en utilisant l'un des validateurs OWL, p.e. Pellet.

L'utilisation d'ontologies est non seulement utile pour la communication entre les différentes applications mais leur soutien de raisonnement est important pour l'extration des connaissances et l'interrogation de l'ontologie.

### 5.3.2 DTD2XSD

Parmi les 122 documents des PIPs de RosettaNet, 22 sont définis en DTD ainsi que les documents du standard cXML. Nous utilisons Trang[9] pour les transformer de DTD vers XML Schema avant d'appliquer la feuille de style XSD2OWL définie à la section précédente. Trang est un programme pour la conversion entre différents langages de schéma, par exemple de RELAX NG vers XML Schema, ou de DTD vers XML Schema. L'un des avantages de Trang est sa portabilité, en plus qu'il est gratuit et facile à utiliser. Il a des caractéristiques uniques par rapport aux autres systèmes puisqu'il prend en charge les espaces de noms, y compris ceux qui mélangent plusieurs espaces de noms. Ensuite, les règles de transformation de XSD2OWL présentées à la section précédente sont appliquées à la sortie de Trang.

Comme indiqué à la section 5.3.1, les annotations permettent la documentation et la définition des éléments. RosettaNet et xCBL annotent les éléments définis dans leur document. Dans le cas de PIPs définis en DTD, RosettaNet associe, à chaque PIP, un document HTML contenant la description de ces éléments. Un traitement spécifique à ces PIPs a été réalisé afin d'extraire la définition, à partir du document HMTL, de chaque élément rencontré dans le document DTD. Avant l'extraction des descriptions, JTidy[10] est utilisé pour transformer les documents HTML en XML bien formés.

### 5.3.3 Évaluation de l'expressivité

La figure 5.2 montre que nous traitons plus que 90% des composants définis dans les schémas des documents d'affaire. Par contre, les éléments de type `xs:sequence` et

[9] http ://www.thaiopensource.com/relaxng/trang.html
[10] http ://jtidy.sourceforge.net/

`xs:choice` ne sont pas pris en compte lors de la transformation. `xs:sequence` précise que les éléments enfants doivent apparaître dans une séquence précise avec un nombre quelconque de fois ou dans n'importe quel ordre pour les `xs:choice`. En plus, il y a certaines propriétés qui n'ont pas été prises en compte lors du processus de transformation que ce soit à cause des limitations syntaxiques du XML Schema ou bien celles de OWL2.0.

Nous commençons par celles du XML Schema. XML Schema est limité syntaxiquement puisqu'il ne définit aucune contrainte sémantique et ne permet pas de modéliser certaines relations sémantiques par exemple de type `subClassOf` ou bien celles des classes disjointes.

Par contre, certaines définitions de concepts exprimées en XML Schema sont sous-déterminées en OWL 2, comme discuté à la section 3.4. Les informations sur la séquence d'apparition des éléments ne peuvent pas être représentées en OWL et ne contribuent pas à la sémantique mais plutôt à la structure du document. Avec ReDeFer [17] et XS2OWL [52], les `xs:sequence` et `xs:choice` sont représentés en OWL respectivement par `xs:intersectionOf` et `xs:unionOf` et des classes anonymes sont produites pour représenter les `sequence` et les `choice`, ce qui ne nous semble pas approprié pour notre application même si ce type de transformation est "standard" dans la théorie classique des types.

Ainsi, nous n'avons pas pu capturer certaines informations décrites en XML Schema dans l'ontologie principale en raison des limitations d'expressivité de la syntaxe OWL 2. La syntaxe OWL 2 ne fournit pas de constructions avec la sémantique équivalentes à celle de `xs:redefine`, `xs:override` et `xs:assert`, définis à la section 3.2. Ces constructions ne représentent que des informations liées à la structure des éléments dans le document et non pas leur sémantique. En plus qu'elles ne sont pas utilisées dans les schémas des documents d'affaire suite à notre analyse présentée à la figure 5.2.

Ces informations représentent une perte et une limite de notre approche de transformation même si les documents des standards d'affaire ne reposent pas sur ce type des composants syntaxiques.

## 5.4 Étude de cas

Cette section illustre le processus de transformation décrit précédemment par une étude de cas réalisée sur le document `PurchaseOrderRequest_02_05.xsd` du PIP3A4 de RosettaNet. Nous discutons des principes de PIP3A4 et nous décrivons l'ontologie résultante du processus de transformation XSD2OWL.

Le PIP3A4, nommé `Request Purchase Order`, permet à un acheteur d'émettre un bon de commande et d'obtenir par la suite une confirmation du vendeur si la commande a été acceptée, rejetée ou retardée. Ce document, défini en XML Schema, décrit les concepts tels que : l'élément `DocumentHeader` qui décrit les informations du document (date de création, version, etc) et l'élément `PurchaseOrder` qui présente la spécification de la commande (description des produits, l'identification des partenaires, etc). Le tableau 5.III présente quelques statistiques sur l'ampleur du document.

Tableau 5.III – Statistiques sur le document `PurchaseOrderRequest_02_05.xsd` du PIP3A4 de RosettaNet.

| | **Nombre** |
|---|---|
| `xs:complexType` | 7 |
| `xs:attribute` | 1 |
| `xs:element/@ref` | 23 |
| `xs:element/@name` | 37 |
| lignes dans le fichier XSD | 782 |
| lignes dans le fichier OWL généré | 5565 |

L'ontologie résultante contient 7 `Class`, 19 `DataProperty` et 41 `ObjectProperty`. La figure 5.4 présente un extrait de l'ontologie, générée par XSD2OWL, du document PIP3A4 de RosettaNet.

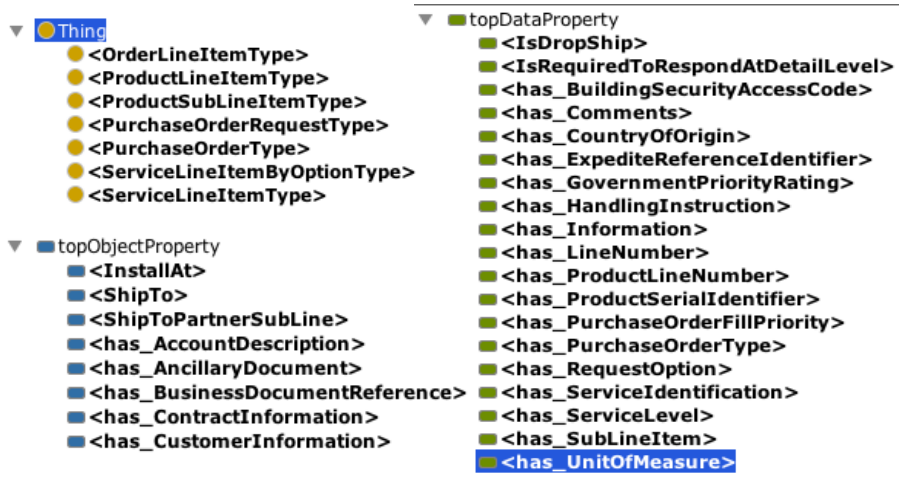



Figure 5.4 – Capture d'écran des onglets de Protégé qui présentent les artefacts OWL de l'ontologie du PIP3A4 générée par XSD2OWL.

L'`IRI` de l'ontologie est identifié par la chaîne "`urn:rosettanet:specification:interchange:PurchaseOrderRequest:xs:schema:02.05.`" qui correspond à l'espace de noms du document `PurchaseOrderRequest_02_05.xsd` du PIP3A4. Dans le reste de cette section, nous donnons un exemple de chaque artefact OWL (`Class`, `DataProperty` et `ObjectProperty`).

**Class.** `<ProductLineItemType>` est une classe dans l'ontologie du PIP3A4. Elle définit l'ensemble de propriétés décrivant un produit. Cette définition est tirée de la documentation de cet élément qui représente un `xs:complexType` dans le document XML Schema du PIP3A4 et transformé en `owl:Class` par la règle **`R1`**.

| | |
|---|---|
| **XSD** | `<xs:complexType name="`**`ProductLineItemType`**`">`<br>`  <xs:sequence>`<br>`  ....`<br>`  </xs:sequence>`<br>`</xs:complexType>` |
| **OWL** | `<owl:Declaration>`<br>`  <owl:Class IRI="`**`ProductLineItemType`**`"/>`<br>`</owl:Declaration>` |

**DataProperty.** L'élément `Comments` est transformé en `owl:DataProperty`, nommée `<has_Comments>`, par l'application de la règle **R2**. Il a été déclaré dans un `xs:complexType`, nommé `ProductLineItemType` dans le document `PurchaseOrderRequest` du PIP3A4.

**XSD**

```
<xs:complexType name="ProductLineItemType">
  <xs:sequence>
    <xs:element name="Comments" type="xs:string"/>
    ....
  </xs:sequence>
</xs:complexType>
```

**OWL**

```
<owl:Declaration>
  <owl:DataProperty IRI="has_Comments"/>
</owl:Declaration>

<owl:DataPropertyRange>
  <owl:DataProperty IRI="has_Comments"/>
  <owl:Datatype IRI="xs:string"/>
</owl:DataPropertyRange>

<owl:DataPropertyDomain>
  <owl:DataProperty IRI="has_Comments"/>
  <owl:Class IRI="ProductLineItemType"/>
</owl:DataPropertyDomain>
```

Alors, le `owl:DataPropertyDomain` sera la classe `<ProductLineItemType>` et le `owl:DataPropertyRange` prendra `xs:string`, la valeur de l'attribut `@type`, par l'application de la règle **R4**.

**ObjectProperty.** L'élément `shipTo` est transformé en `owl:ObjectProperty` à l'aide de la règle **R2**.

Cet élément a été déclaré dans un élément, nommé `ProductLineItemType`, de type `xs:complexType` défini dans le document `PurchaseOrderRequest` du PIP3A4. Alors que son attribut `@type` fait référence à un élément de type `xs:complexType`, nommé `SpecifiedPartnerDescriptionType`, introduit dans le document `PartnerIdentification_01_16.xsd`.

Cette propriété relie la classe `<SpecifiedPartnerDescriptionType>` , qui représente le `owl:DataPropertyRange` défini par l'application de la règle **R4**, avec la classe `<ProductLineItemType>` comme `owl:DataPropertyDomain`.

**XSD**

```
<xs:complexType name="ProductLineItemType">
  <xs:sequence>
    <xs:element name="shipTo"
                type="upi:SpecifiedPartnerDescriptionType"/>
    ....
  </xs:sequence>
</xs:complexType>
```

**OWL**

```
<owl:Declaration>
  <owl:ObjectProperty IRI="shipTo"/>
</owl:Declaration>

<owl:ObjectPropertyRange>
  <owl:ObjectProperty IRI="shipTo"/>
  <owl:Class IRI="&upi;SpecifiedPartnerDescriptionType"/>
</owl:DataPropertyRange>

<owl:ObjectPropertyDomain>
  <owl:ObjectProperty IRI="shipTo"/>
  <owl:Class IRI="ProductLineItemType"/>
</owl:DataPropertyDomain>
```

## 5.5 Application du Formal Concept Analysis (FCA)

L'évaluation de la qualité est une tâche fondamentale dans l'ingénierie des ontologies pour détecter les défauts de conception et identifier les parties qui causent problème.

Tartir et al. [50] et Sicilia et al. [47] ont proposé des métriques pour analyser la qualité des ontologies. Une d'entre elles, `Inheritance Richness`, représente la distribution de l'informations entre les différents niveaux de l'arbre d'héritage d'une ontologie, définie comme le nombre moyen de sous-classes par classe. Quand la métrique est proche de

zéro, l'ontologie est "aplatie" et décrit une connaissance d'une façon générique, tandis qu'elle représente une organisation plus hiérarchique d'un domaine dans le cas contraire.

Les ontologies générées par le processus de tranformation, décrit à la section 5.3, sont horizontales car elles ne profitent pas de certaines relations sémantiques importantes en OWL comme `subClassOf`.

Dans cette section, nous essayons d'améliorer l'organisation de l'arbre d'héritage d'une ontologie OWL 2 en proposant une méthode pour regrouper les classes ayant un comportement similaire et partageant des propriétés pour améliorer la qualité et la lisibilité d'une ontologie. Nous utilisons l'analyse formelle de concepts, nommée en anglais Formal Concept Analysis [16], dans la détection des groupes de concepts en utilisant les artefacts d'une ontologie (`Class`, `DataProperty` et `ObjectProperty`).

### 5.5.1 Définition de l'analyse formelle de concepts

"L'analyse formelle de concepts est une méthode d'analyse de données basée sur la théorie des treillis et spécialisée dans l'extraction d'un ensemble ordonné de concepts au sein d'un ensemble de données, appelé un contexte formel, qui est composé d'objets décrits par des attributs." [24]

Les données d'entrée à FCA ont la forme d'un tableau croisé qui décrit les relations entre les objets définis par les lignes du tableau et les attributs représentés par les colonnes. Un exemple d'un tableau croisé est présenté au tableau 5.IV. Une entrée <*i*,*j*> du tableau est marquée d'un "*x*" signifie qu'il y a une relation entre l'objet $x_i$ et l'attribut $y_j$.

Tableau 5.IV – Exemple de la représentation de données sou forme d'un tableau croisé.

| *R* | $y_1$ | $y_2$ |
|---|---|---|
| $x_1$ | x | |
| $x_2$ | x | x |
| $x_3$ | | x |

Un contexte formel est un triplet (*X*, *Y*, *R*) où *X* et *Y* représentent respectivement un ensemble non vide d'objets et d'attributs ; *R* décrit la relation binaire entre *X* et *Y*, e.g.

$R \subseteq X$ x $Y$. Pour tout $x \in X$ et $y \in Y$, $(x,y) \in R$ signifie que l'objet $x$ possède l'attribut $y$.

À partir du contexte formel ($X$, $Y$, $R$), une collection de concepts formels hiérarchiquement ordonnés par une relation de sous-concept et super-concept, appelée un treillis de concepts, sera produite. Un concept formel est une paire ($E$, $I$) avec $E$ et $I$ sont respectivement appelés l'*extension* et l'*intension* du concept :

- $E = \{y \in Y \mid \forall\ x \in A \subseteq X\}$ représente tous les attributs $y$ de $Y$ partagés par un sous-ensemble d'objets $A$ de $X$.

- $I = \{x \in X \mid \forall\ y \in B \subseteq Y\}$ représente tous les objets $x$ de $X$ ayant tous les attributs du sous-ensemble $B$ de $Y$.

Étant donné deux concepts formels $C_1 = (E_1, I_1)$ et $C_2 = (E_2, I_2)$, ils seront ordonnés par l'opérateur $\leq$ comme suit : $C_1 \leq C_2$ si et seulement si $I_2 \subseteq I_1$ et $E_1 \subseteq E_2$. Alors, $C_1$ est un sous-concept de $C_2$ et vice versa $C_2$ est un super-concept de $C_1$. Par exemple à partir du tableau 5.IV, le concept ($\{x_2\},\{y_1,y_2\}$) est un sous-concept de ($\{x_1,x_2\},\{y_1\}$).

### 5.5.2 Approche de détection des concepts basée sur FCA

De nombreuses approches basées sur FCA ont été proposées dans le domaine d'ingénierie d'ontologie. Cimiano et al. [4] ont discuté les services que FCA peut offrir pour l'ingénierie des ontologies comme la construction et l'analyse des ontologies.

Stumme [49] présente une approche de fusion d'ontologies basée sur FCA, nommée FCA-MERGE. De leur côté, Obitko et al. [33] ont proposé une approche pour améliorer la conception des ontologies en ajoutant si nécessaire des nouvelles classes ou propriétés. Ils proposent d'améliorer la description de concepts et les relations d'une ontologie plutôt que de chercher à les hiérarchiser.

Nous utilisons plutôt l'analyse formelle de concepts pour tenter de regrouper les concepts d'une ontologie *aplatie* et ainsi améliorer la représentation des connaissances [28]. Le processus de regroupement est illustré à la figure 5.5 et comporte quatre principales étapes : le traitement de l'ontologie à l'aide de OWL API[11], la construction du

[11] http ://owlapi.sourceforge.net/

treillis, l'extraction des groupes de concepts et la validation des résultats de regroupement.

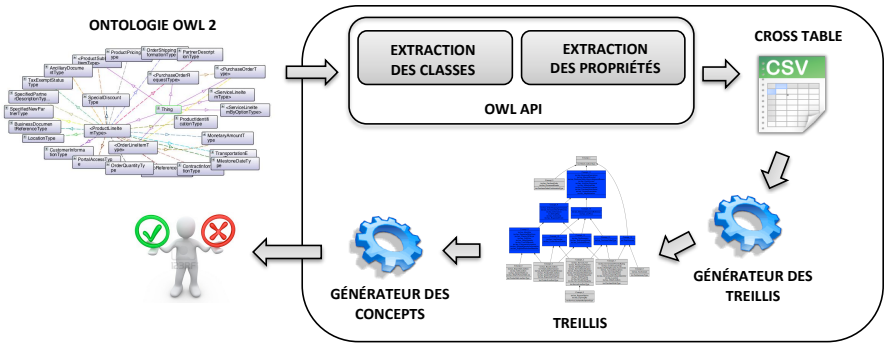



Figure 5.5 – Processus de regroupement des concepts.

**Étape 1.** Le traitement de l'ontologie consiste à naviguer dans l'ontologie pour extraire des éléments (`Class`, `DataProperty` et `ObjectProperty`) et construire le tableau croisé où les lignes représentent les classes de l'ontologie et les colonnes, les propriétés. Une entrée <*i*,*j*> du tableau est marquée d'un "*x*" si le `domain` de la propriété, définie par la colonne *j*, est la classe présentée par la *i* ème ligne.

**Étape 2.** La génération du treillis a été décrite à la section 5.5.1. La sortie de cette étape est une collection de concepts formels, appelée treillis de concepts représentés sous forme d'un arbre d'héritage qui fournit un regroupement des classes de l'ontologie source. Un exemple de treillis de concepts est présenté à la figure 5.6, correspond à l'étude de cas défini à la section 5.4. Tous les concepts intermédiaires représentent l'ensemble de propriétés partagées et ceux non numérotées représentent les classes.

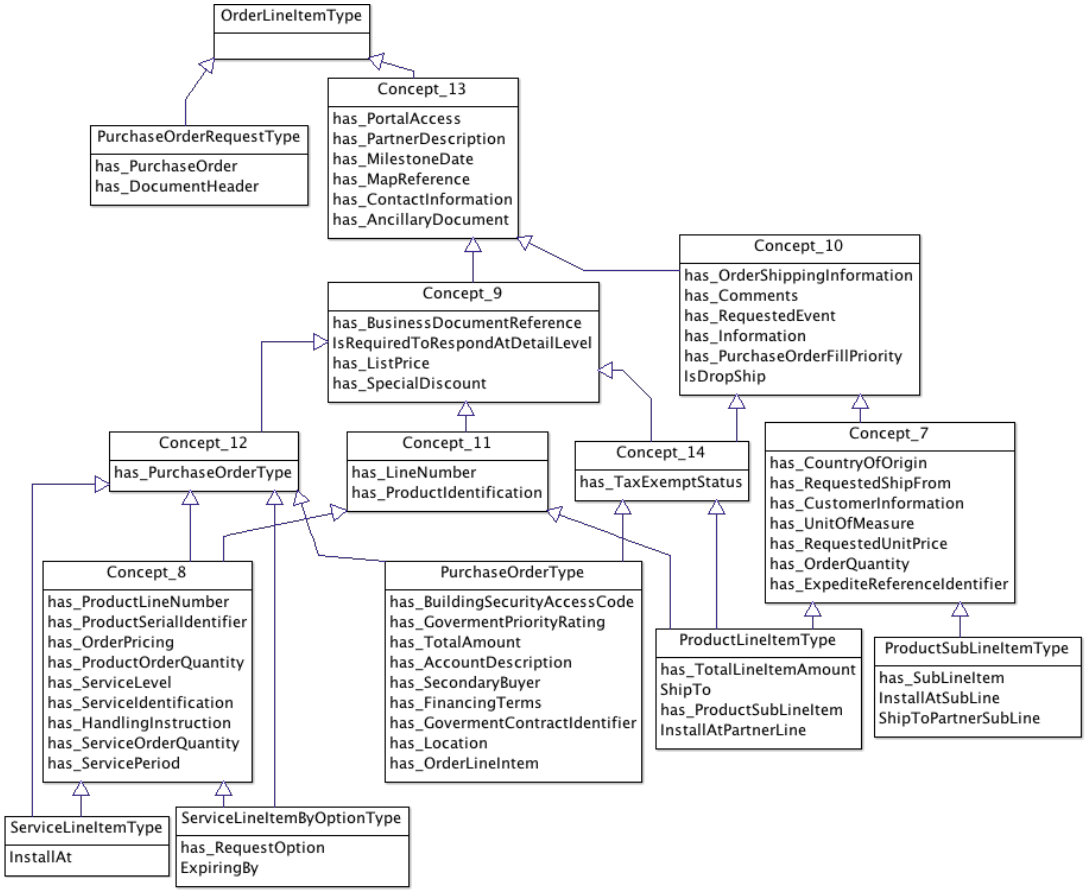



Figure 5.6 – Treillis de concepts décrivant le document `Purchase Order Request` du PIP3A4 de RosettaNet.

Dans le cas où les classes d'une ontologie ne partagent aucune propriété. Un treillis de concepts avec un seul niveau d'héritage sera généré où chaque concept représente une classe de l'ontologie ainsi que l'inexistence des concepts intermédiaires définissant les propriétés partagées entre les classes. Donc, nous ne pouvons pas parler d'un groupe de classes qui partagent les mêmes propriétés et FCA n'a donc pas aidé dans ce cas.

**Étape 3.** Nous regroupons les classes où chacune sera associée au concept ayant le plus grand nombre de propriétés communes en naviguant dans l'arborescence du treillis.

Dans la figure 5.7, nous notons que le `Concept_7` combine les deux classes `ProductLineItemType` et `ProdutSubLineItemType` avec 19 propriétés communes.

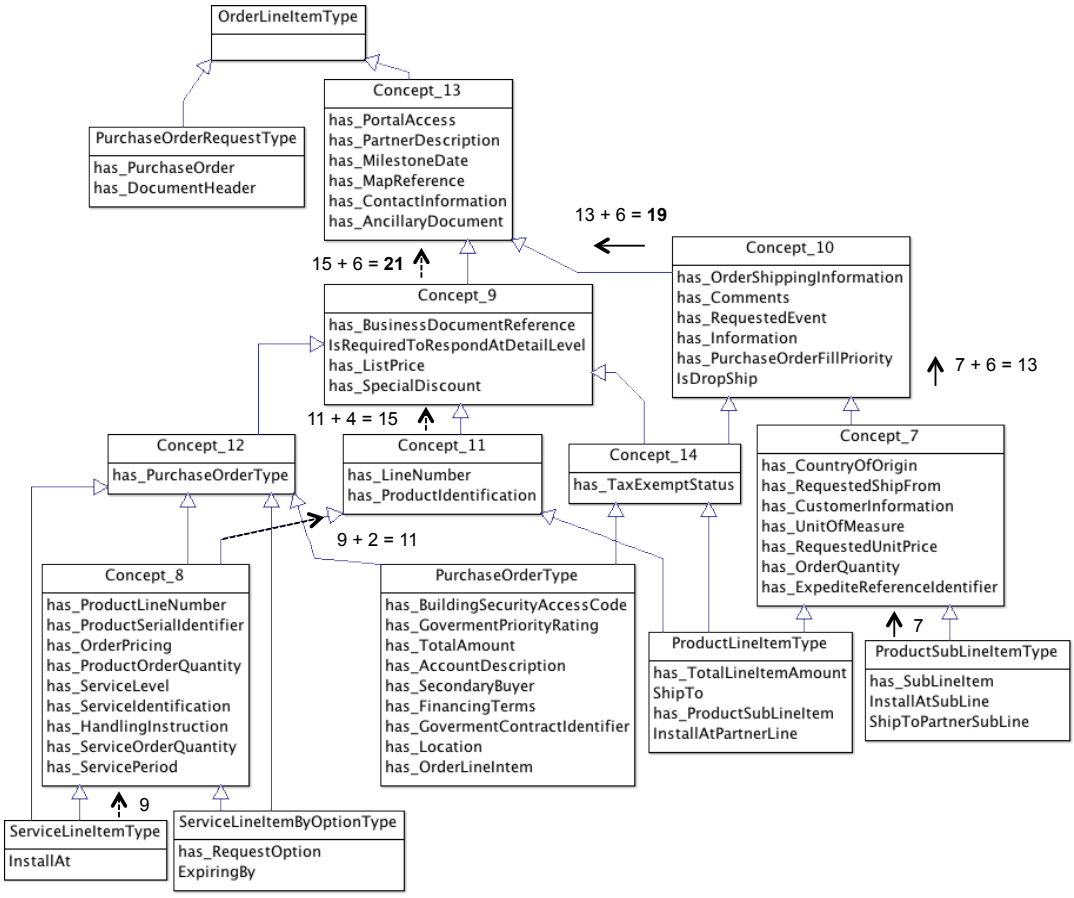



Figure 5.7 – Navigation dans l'arborescence du treillis de concepts de la figure 5.6. Les flèches, en trait pointillé, représentent les chemins donnant le plus grand nombre de propriétés pour le regroupement des concepts `ServiceLineItemType` et `ServiceLineItemByOptionType`. Les flèches, en trait dur, représentent les concepts `ProductLineItemType` et `ProductSubLineItemType`.

Ainsi, le `Concept_8` joint `ServiceLineItemType` et `ServiceLineItemByOptionType` avec 21 propriétés partagées.

Nous avons utilisé FCA pour créer une hiérarchie de concepts dans une ontologie ayant des classes avec un comportement commun. Mais, nous ne considérons que deux niveaux dans l'arbre d'héritage pour en limiter le profondeur comme le suggère Sheldon et al. [45] qui montrent qu'une hiérarchie peu profonde améliore la maintenabilité, la lisibilité et la compréhension.

**Étape 4.** Nous validons les résultats obtenus en analysant les classes regroupées, si elles possèdent une sémantique commune ou non. Cette analyse se base sur la documentation et la définition des classes.

Dans l'exemple de la figure 5.6, les classes regroupées partagent des tokens dans leurs noms, p.e. `Service` ou `Item`, qui vont être utilisés pour changer manuellement les noms de concepts générés par FCA. Donc, `Concept_7` devient `ProductLineItem` et `Concept_8`, `ServiceLineItemType`.

## 5.6 Expérimentation

À la section 5.5.2, nous avons utilisé le document du PIP3A4 de RosettaNet pour expliquer le processus de fonctionnement de notre approche de détection de concepts dans une ontologie OWL. Dans cette section, nous présentons des statistiques globales de l'application de cette approche sur l'ensemble des 122 PIPs de RosettaNet. Afin de tester notre méthodologie, nous utilisons les ontologies de ces PIPs générées par le processus de transformation détaillé à la section 5.3. Au tableau 5.V, nous évaluons ces ontologies en utilisant des métriques.

Tableau 5.V – Analyse empirique des ontologies des 122 PIPs de RosettaNet par des métriques : avant et après l'application de notre approche de détection de concepts basé sur FCA.

| | **Avant** | **Après** | **Définition de la métrique** |
|---|---|---|---|
| **noc** | 1252 | 1252 | Nombre de classes. |
| **nodp** | 3045 | 3045 | Nombre de Data Property. |
| **noop** | 2607 | 2607 | Nombre d'Object Property. |
| **nop** | 5652 | 5652 | Somme de **nodp** et **noop**. |
| **nosc** | 0 | 384 | Nombre de sous classes. |
| **norc** | 1252 | 1050 | Nombre de classes sans superclasses (`root` classes). |
| **nolc** | 1252 | 868 | Nombre de classes (sans sousclasses). |
| **rr** | 1 | 0.93 | **nop**/(**nop**+**nosc**) |
| **ir** | 0 | 0.31 | Moyenne du nombre de sous classes par classe. |

Au tableau 5.V, nous constatons que les valeurs originales des métriques `ir` et `nosc` sont nulles et les valeurs de `norc` et `nolc` sont égales au nombre de classes dans l'ontologie. Les valeurs de ces métriques indiquent que ces ontologies sont plates ou horizontales même si RosettaNet représente un domaine avec de nombreux éléments partageant une sémantique commune.

Nous avons extrait 182 concepts comprenant 384 classes parmi les 1252 existantes dans les ontologies de tous les PIPs, ce qui augmente de la métrique `ir` de 0 à 0.31 et diminue de la métrique `rr` de 1 à 0.93. En moyenne, chaque concept contient 2 classes.

Il y a 90 concepts distincts détectés puisque plusieurs concepts sont partagés entre les ontologies des PIPs, p.e. le concept, combinant les deux classes définissant le type de taxe `RegionalBusinessTaxIdentifier` et `NationalBusinessTaxIdentifier`, a été détecté dans 3 PIPs (`PIP3A5`, `PIP3A11` et `PIP3B6`).

Nous ne possèdons ni le nombre, ni la liste des bons concepts pour réaliser une vérification automatique de tous les concepts détectés par notre algorithme et par la suite évaluer sa performance (Rappel et Précision). Alors, nous avons procédé à une validation manuelle de tous les concepts identifiés par l'algorithme. Nous avons remarqué que 79 concepts ont effectivement une sémantique commune parmi les 90 concepts détectés alors que les 11 concepts restants représentent le nombre des mauvais concepts identifiés comme bons. Par conséquent, nous avons eu une précision de 87% sur l'ensemble des ontologies sur tous les PIPs de RosettaNet.

## 5.7 Conclusion

Nous avons justifié le besoin de l'intégration de la sémantique dans les standards d'affaire. Nous avons proposé un processus de transformation de ces standards de la représentation XML et DTD vers une représentation ontologique. Nous avons étudié différentes règles de transformation qui ont été illustrées avec un exemple du document `PurchaseOrderRequest_02_05.xsd` du PIP3A4 de RosettaNet.

Comme les ontologies générées ne profitent pas de certaines relations sémantiques en OWL, p.e `subClassOf`, à ce problème, nous avons mis en place une approche de re-

groupement de classes basée sur l'analyse formelle de concepts et évalué sa performance sur l'ensemble des PIPs de RosettaNet.

Même si l'adoption des ontologies aide à organiser la connaissance et améliore l'interprétation de l'information par les machines. Les ontologies ne sont pas nécessairement compatibles ayant la même structure et la même défintion des concepts, ce qui complique l'interopérabilité et provoque l'hétéréogénéité dans les échanges. Au chapitre suivant, nous discutons les types d'hétéréogénéité entre les ontologies des standards d'affaire et les solutions pour les surmonter.

# CHAPITRE 6

# ALIGNEMENT DES ONTOLOGIES DANS LE DOMAINE DES AFFAIRES

## 6.1 Introduction

Avant d'entrer dans les détails d'utilisation des ontologies dans l'organisation et la gestion de documents d'affaires et de montrer comment l'alignement entre les ontologies peut aider les entreprises à mieux communiquer, il est utile de présenter les problèmes qui se présentent lors d'un échange de documents dans un processus d'affaire et les solutions proposées pour y remédier.

Nous avons vu au chapitre 1 que l'échange de documents peut causer des problèmes liés à la compréhension du sens des mots dans les documents échangés, le sens d'un mot pouvant changer selon les situations ou les contextes. Ce problème est dû au fait que chaque partenaire commercial utilise son propre modèle d'affaire avec une définition spécifique de documents. De plus, le partage d'information est un problème dans le processus d'affaire surtout au niveau des documents échangés entre les parties. Une solution à ce problème est de permettre à l'ordinateur d'avoir un meilleur accès à la sémantique de l'information par le biais du Document Engineering qui cherche à organiser, modéliser et décrire l'information contenue dans les documents ainsi que les processus d'affaire.

En se basant sur XML, des standards comme xCBL, cXML, ebXML et RosettaNet, ont été développés pour l'organisation des affaires inter-entreprises et la gestion de documents circulant dans les processus d'affaires. Malgré les pistes que le langage XML a offertes au niveau de l'organisation et de la structuration, la représentation de la sémantique de données n'est pas encore possible. Chaque standard a été élaboré séparément par différents organismes pour répondre aux besoins d'un groupe particulier d'entreprises qui communiquent.

L'utilisation des ontologies est importante et prometteuse. Le partage des modèles de documents permet de communiquer la sémantique entre les différentes parties d'un

processus d'affaire et de rendre l'information interprétable par les machines. L'utilisation des ontologies identifie les concepts les plus significatifs et les relations utilisées dans un domaine de connaissance. Les technologies comme RDF et RDFS permettent la construction de représentations complexes de la connaissance, en plus d'offrir un ensemble de primitives et d'éléments pour les différents types de ressources. Cette constatation nous a amené à élaborer un processus automatique de transformation des documents de ces standards, décrit au chapitre 5, de DTD et XML Schema vers une représentation ontologique OWL [27].

Il est utile d'utiliser l'ontologie pour représenter la connaissance ainsi que la relation entre les concepts d'un domaine. Les ontologies sont destinées à faciliter l'interopérabilité sémantique entre les parties ce qui est relativement simple lorsque toutes les parties utilisent la même ontologie [35]. Par contre, les ontologies ne sont pas nécessairement compatibles. Dans notre cas, le résultat de notre transformation est un ensemble d'ontologies qui représentent chacune un document d'affaire avec leur propre structure et vocabulaire. Il n'y a pas d'ontologie commune ou de vocabulaire commun (thésaurus distributionnel) pour gérer la communication entre les entreprises et aider à éliminer les problèmes d'hétérogénéité. L'hétérogénéité provient du fait que les ontologies sont créées de manière indépendante avec chacune leur vocabulaire même si elles traitent du même domaine d'intérêt. Ce problème complique l'interopérabilité et la communication entre les partenaires d'affaire. Comme il nous semble impossible de convaincre toutes les entreprises de modifier leur modèle de documents pour avoir un modèle commun, il faut développer une solution qui utilise les modèles existants et qui cherche un lien entre eux.

L'alignement des ontologies permet dans une certaine mesure de surmonter le problème de l'hétérogénéité sémantique [46]. L'alignement est un processus qui prend en entrée deux ontologies et trouve des liens sémantiques ou un ensemble de correspondances entre les entités de ces ontologies (des classes, des propriétés, des instances, etc.).

Notre étude porte sur le domaine du e-business, plus spécifiquement sur les standards d'affaires. Plusieurs études d'alignement ont été proposées dans ce domaine : processus

d'affaire [30][17], ou web services [56] mais jamais sur les documents qui sont le principal support de communication entre les partenaires d'affaires.

Notre contribution ne porte pas sur une nouvelle approche d'alignement, nous cherchons plutôt à tirer avantage des recherches dans ce domaine afin de trouver des correspondances entre plusieurs ontologies d'affaires hétérogènes et aider les entreprises à mieux communiquer même si leurs modèles de documents diffèrent. Généralement, la plupart des systèmes d'alignement ont été évalués sur un ensemble de jeu de test (benchmark) durant les OAEIs ou appliqués à des ontologies portant sur le domaine de la bioinformatique comme ASMOV [26].

Au chapitre 4, nous avons vu les principales normes basées sur XML pour l'organisation des affaires inter-entreprises comme ebXML, RosettaNet, cXML et xCBL. Nous avons présenté leur différences surtout au niveau des lignes directrices.

Nous nous intéressons aux standards offrant une description détaillée des documents d'affaire cXML, xCBL et RosettaNet. Ces standards partagent le même domaine d'intérêt et traitent de la chaîne d'approvisionnement plus spécifiquement la gestion des bons de commande. La représentation de ces standards réside encore sous la forme DTD et XML, nous avons développé un outil de transformation automatique de ces documents vers une représentation ontologique. RosettaNet semble le plus abouti puisqu'il est soutenu par plusieurs grandes entreprises comme Intel, Microsoft, etc. Avec l'alignement d'ontologies, nous donnons la chance aux petites entreprises qui utilisent les autres standards (e.g. cXML et xCBL) de communiquer avec ceux qui supportent RosettaNet.

Avant d'entamer cette contribution, il est raisonnable de considérer plusieurs questions : ces ontologies sont-elles similaires ? À quel point les résultats d'alignement aident-ils à réaliser l'objectif de l'ingénierie de documents dans le domaine d'affaire ?

Dans ce chapitre, nous nous intéressons à la gestion des bons de commande (niveau demande et confirmation de bon de commande dans le cadre de cXML, xCBL et RosettaNet) ; cet aspect est considéré comme un axe très important de la chaîne d'approvisionnement. Nous décrivons brièvement les standards étudiés ainsi que les approches d'alignement proposées dans le domaine d'affaire. Ensuite, nous allons étudier les ontologies en utilisant des métriques afin de choisir les critères d'alignement adéquats et pour

en déterminer le degré de similarité. Nous analysons ensuite les résultats d'alignement de cXML avec RosettaNet PIP3A4 ainsi que xCBL avec RosettaNet PIP3A4 (au niveau du Request et Confirm).

## 6.2 Description de RosettaNet PIP3A4, xCBL et cXML

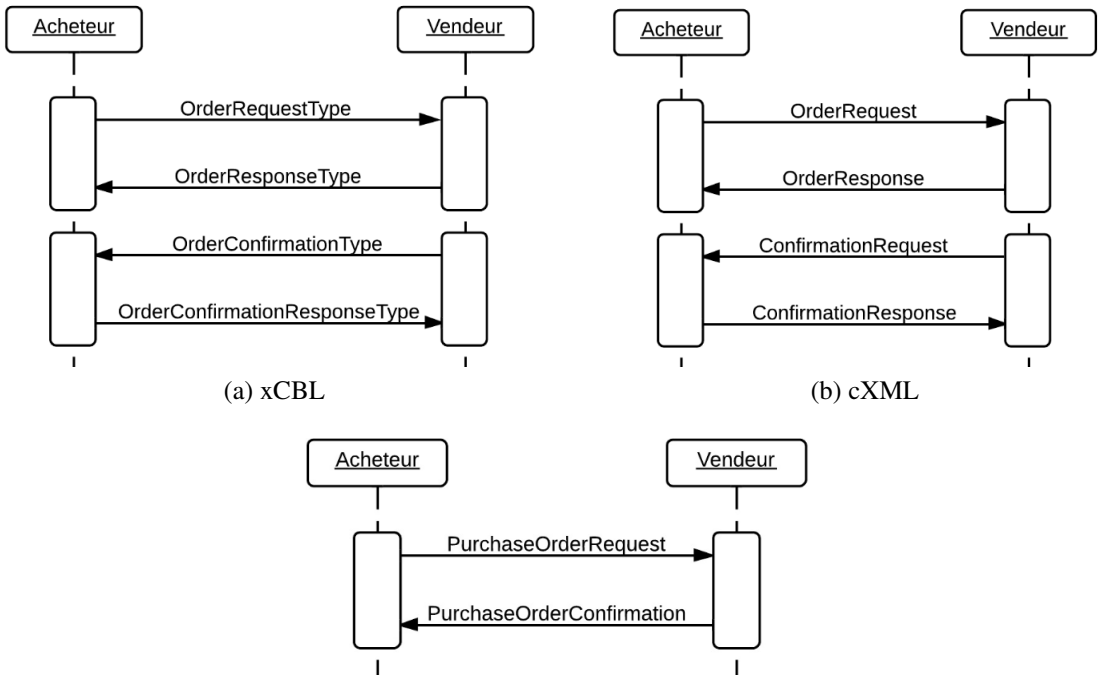
RosettaNet, xCBL et cXML sont des normes basées sur XML pour l'organisation des affaires inter-entreprises et la gestion de documents circulant dans les processus d'affaires. Ils partagent un but principal et commun qui consiste à offrir des normes gérant les affaires entre les entreprises dans le monde entier et surtout répondre aux besoins spécifiques de l'industrie dans la chaîne d'approvisionnement.

Une chaîne d'approvisionnement est l'ensemble des étapes que prend une entreprise pour transformer ses matières premières en produit fini, la distribution des produits, la gestion des commandes, la livraison, etc. Les processus d'affaire dans une chaîne d'approvisionnement sont gérés entre les systèmes d'entreprises par un échange de données informatisées (EDI), généralement sous la forme de documents transactionnels tels que les bons de commande.

Notre étude se focalise sur la gestion des bons de commande. Généralement, ce processus d'affaire respecte le même diagramme de séquence pour les 3 standards. La figure 6.1 montre le déroulement de ce processus pour chaque standard d'affaire avec chaque message communiqué entre les partenaires représente un document transactionnel (voir au tableau 6.I).

Tableau 6.I – La liste des documents utilisés dans la gestion des bons de commande. La réalisation de ce processus est gérée par l'envoi de cette liste de documents, qui comprend les documents pour la demande et la confirmation de bons de commande, entre l'acheteur et le vendeur.

| | **Request** | **Confirm** |
|---|---|---|
| **RosettaNet** (`.xsd`) | `PurchaseOrderRequest` | `PurchaseOrderConfirmation` |
| **xCBL** (`.xsd`) | `OrderRequestType` | `OrderConfirmationType` |
| **cXML** (`.dtd`) | `OrderRequest` | `ConfirmationRequest` |




(a) xCBL

(b) cXML

(c) RosettaNet PIP3A4

Figure 6.1 – Diagramme de séquence pour la réalisation de commande dans xCBL (a), cXML (b), et RosettaNet PIP3A4 (c).

Au chapitre 4, nous avons décrit ces standards d'affaires en détail (le principe général, les composantes, les fonctionnalités et leur différences). Par contre, nous détaillons les parties traitant de la gestion des bons de commande.

### 6.2.1 RosettaNet (`PIP3A4`)

RosettaNet, vu à la section 4.2.2, définit des schémas XML décrivant une centaine de processus d'échange d'information entre les partenaires en s'appuyant sur des Partner Interface Processes (PIP). Le PIP3A4, nommé `Request Purchase Order`, appartient au 3ème cluster `Order Managament` et le segment `Quote and Order Entry`. Ce PIP permet à un acheteur d'émettre un bon de commande et d'obtenir une confirmation d'un vendeur comme le montre la figure 6.1c [41]. Dans le cadre du PIP3A4, le processus débutera suite à un document `PurchaseOrderRequest` envoyé de l'acheteur au vendeur. Ensuite, un `PurchaseOrderConfirmation` sera envoyé contenant la

confirmation de la commande du vendeur au client.

### 6.2.2 xCBL (`ordermanagement`)

xCBL, vu à la section 4.2.1, est une collection de spécifications XML de documents d'affaires utilisés dans l'e-business. Techniquement, xCBL contient 44 documents et utilise plusieurs espaces de noms, où chaque espace de noms représente un domaine fonctionnel [1]. `ordermanagement` est l'un des domaines fonctionnels de xCBL contenant les documents xCBL associés à la création de bons d'achat et la gestion des commandes, par exemple : `OrderRequestType` et `OrderResponseType`. En xCBL, le processus commence quand l'acheteur envoie un `OrderRequestType` au vendeur (voir figure 6.1a). De son côté, le vendeur confirme la réception de la demande en utilisant le document `OrderResponseType` envoyé au client. Ensuite, le vendeur demande à l'acheteur de confirmer sa commande. Cette communication sera faite par le document `OrderConfirmationType`. Enfin, le client, de sa part, doit accepter ou refuser les détails de sa commande en envoyant un message sous forme d'un document `OrderConfirmationResponseType`.

### 6.2.3 cXML

cXML, vu à la section 4.2.3, offre des spécifications de documents d'affaires définies par une DTD [2]. Les partenaires commerciaux utilisent cXML pour communiquer des bons de commandes. Le processus de réalisation d'une commande est décrit à la figure 6.1b. Le client envoie le document `OrderRequest`. Le document `OrderRequest` est une version électronique d'un bon de commande qui contient les informations du client, de la facturation, la livraison etc. Une fois, la commande reçue dans le système de gestion de commande, le fournisseur renvoie un accusé de réception avec le document `OrderResponse` qui indique si la demande a été reçue avec succès. La confirmation de la commande sera faite par des documents `ConfirmationRequest` envoyés par le fournisseur au client qui, de son côté, confirme la commande via le document `ConfirmationResponse`.

## 6.3 Application de la transformation

Nous appliquons toutes les règles de transformation, définies à la section 5.3, sur des documents traitant de la gestion des bons de commandes, des standards xCBL, cXML et RosettaNet. Au chapitre 5, nous avons détaillé le processus de transformation basé sur deux moteurs de mapping : DTD2XSD (section 5.3.2) et XSD2OWL (section 5.3.1).

La sortie du processus de transformation est une ontologie OWL 2. Nous avons appliqué deux types de validation :

- syntaxique qui en vérifie la conformité avec le schéma XML pour la sérialisation OWL/XML.
- sémantique qui teste la cohérence et la consistance des définitions dans l'ontologie.

Une étude quantitative des ontologies résultantes (nombre de classes, de propriétés, etc.) est illustrée au tableau 6.II. Leur qualité est mesurée à l'aide de métriques, définies au tableau 5.V, qui ont été développées pour des raisons de compréhension et de maintenance.

Tableau 6.II – Analyse empirique des ontologies issues de la transformation sur des exemples des documents d'affaires pour les standards xCBL, RosettaNet et cXML pour deux types de documents : la demande (`Request`) et la confirmation (`Confirm`). L'évaluation est faite en utilisant des métriques définies ci-dessous.

| | **Les standards d'affaire** | | | | | |
|---|---|---|---|---|---|---|
| | **RosettaNet** | | **xCBL** | | **cXML** | |
| | **`Request`** | **`Confirm`** | **`Request`** | **`Confirm`** | **`Request`** | **`Confirm`** |
| **`noc`** | 151 | 143 | 66 | 33 | 17 | 27 |
| **`nodp`** | 264 | 252 | 174 | 68 | 34 | 36 |
| **`noop`** | 126 | 130 | 91 | 35 | 18 | 25 |
| **`nop`** | 372 | 382 | 265 | 103 | 52 | 61 |
| **`nosc`** | 0 | 0 | 0 | 0 | 0 | 0 |
| **`norc`** | 151 | 143 | 66 | 33 | 17 | 27 |
| **`nolc`** | 151 | 143 | 66 | 33 | 17 | 27 |
| **`rr`** | 1 | 1 | 1 | 1 | 1 | 1 |
| **`ir`** | 0 | 0 | 0 | 0 | 0 | 0 |

Tableau 6.III – Analyse empirique des ontologies après l'utilisation du FCA pour le regroupement des classes ayant un comportement commun. Cette analyse se concentre sur les métriques changées suite à l'application de FCA.

| REQUEST | RosettaNet | | xCBL | | cXML | |
|---|---|---|---|---|---|---|
| | **Avant** | **Après** | **Avant** | **Après** | **Avant** | **Après** |
| **nosc** | 0 | 60 | 0 | 0 | 0 | 0 |
| **norc** | 151 | 91 | 66 | 66 | 17 | 17 |
| **nolc** | 151 | 133 | 66 | 66 | 17 | 17 |
| **rr** | 1 | 0.86 | 1 | 1 | 1 | 1 |
| **ir** | 0 | 0.40 | 0 | 0 | 0 | 0 |
| **Précision** | | **72%** | | **0%** | | **0%** |

| CONFIRM | RosettaNet | | xCBL | | cXML | |
|---|---|---|---|---|---|---|
| | **Avant** | **Après** | **Avant** | **Après** | **Avant** | **Après** |
| **nosc** | 0 | 58 | 0 | 0 | 0 | 0 |
| **norc** | 143 | 85 | 33 | 33 | 27 | 27 |
| **nolc** | 143 | 126 | 33 | 33 | 27 | 27 |
| **rr** | 1 | 0.86 | 1 | 1 | 1 | 1 |
| **ir** | 0 | 0.40 | 0 | 0 | 0 | 0 |
| **Précision** | | **71%** | | **0%** | | **0%** |

Pour améliorer la structure de ces ontologies, nous appliquons la technique du `Formal Concept Analysis`, vue à la section 5.5.2, pour regrouper des classes ayant un comportement similaire. L'idée de cette méthode est de structurer l'ontologie en classes et sous-classes, mesurée par la métrique `ir`.

Le tableau 6.III présente les résultats de notre analyse empirique réalisée sur les ontologies des standards d'affaire après l'utilisation du FCA. Les résultats montrent que cette méthode est plus performante dans le cas de RosettaNet que dans le cas de xCBL et cXML. La précision est le rapport entre le nombre des bons concepts détectés et le nombre total de concepts détectés. Pour déterminer les bons de concepts, nous avons utilisé la définition des classes appartenant à chaque concept et par conséquent voir s'ils partagent une certaine sémantique. Ces définitions sont offertes dans la documentation de ces standards.

Dans le cas de RosettaNet, nous avons obtenu `60` sous-classes (`nosc`) ; la métrique `ir` a donc augmenté de `0` à `0.4`, et la métrique `rr` est passée de `1` à `0.86`.

Dans le cas de xCBL et cXML, ces valeurs sont inchangées parce que leurs classes ne partagent aucune propriété même pour ceux avec une sémantique commune. Techniquement, les applications de FCA sur xCBL et cXML génèrent des treillis à un seul niveau. Chaque concept représente une classe de l'ontologie d'où l'inexistence des concepts intermédiaires définissant les propriétés partagées entre les classes. Donc, nous ne pouvons pas parler d'un groupe de classes qui partagent les mêmes propriétés, FCA n'a donc pas aidé dans ce cas.

### 6.3.1 Types d'hétérogénéité

Les ontologies représentent des descriptions formelles d'un domaine. Elles sont destinées à faciliter l'intéropérabilité sémantique entre plusieurs applications distantes. Plusieurs ontologies ont été développées de façon indépendante par différentes organisations ce qui rend la communication et l'interopérabilité entre leur application de plus en plus difficiles.

Dans le cadre des standards d'affaire, les modèles de documents ont été développés d'une façon indépendante par différents organismes (e.g. `OASIS`) pour gérer les affaires entre des groupes d'entreprises. Même si, ces ontologies traitent du même domaine d'intérêt, la définition et la représentation des concepts diffèrent.

Cette hétérogénéité ne réside pas uniquement au niveau des objectifs d'applications ou des langages de définition des ontologies. Il existe plusieurs types d'hétérogénéité dont les plus importants, selon Euzenat et al. [14], sont :

**Syntaxique** : elle apparaît quand deux ontologies ne sont pas exprimées avec le même langage ou utilisent des formalismes différents de modélisation de connaissance. Par exemple, RosettaNet et xCBL utilisent XML Schema pour la modélisation des documents, alors que cXML adopte Document Type Definition (DTD). Nous l'avons résolue grâce au processus de transformation de DTD et XML Schema de ces standards vers un formalisme unique, le langage OWL.

**Terminologique** : ce problème est dû à l'utilisation de langues naturelles différentes ou à l'existence de la synonymie, c.à.d l'utilisation de termes différents pour désigner des objets sémantiquement similaires. Par exemple, les concepts `From` de cXML et `SenderType` de RosettaNet sont semblables, de même pour `ContactInformationType` de RosettaNet et `PartyType` de xCBL.

**Conceptuelle** : la différence de modélisation des ontologies d'un même domaine d'intérêt, la façon dont les concepts ont été conçus et surtout le niveau de détail dans la représentation de connaissance. Par exemple, RosettaNet offre plus de détails dans la description des bons de commandes par rapport à xCBL et cXML même s'ils ont la même perspective, mais avec des niveaux de détails différents. Par exemple, le concept `PaymentMethodType` dans xCBL représente la manière d'effectuer un paiement. Par contre, RosettaNet décrit en détail cet aspect par des sous concepts (`PartPaymentType` et `PrePaymentDetailType`) représentant chacun une méthode de paiement.

**Pragmatique** : représente l'interprétation d'une entité selon le contexte et l'objectif de l'ontologie. Ce genre d'hétérogénéité est difficile à gérer par les ordinateurs [14].

Nous essayons de réduire l'hétérogénéité terminologique et conceptuelle entre les ontologies des standards d'affaire. L'hétérogénéité syntaxique est résolue grâce à notre processus de tranformation puisque ces ontologies utilisent le même langage de définition (OWL). Par contre, L'hétérogénéité pragmatique dépend du sujet traité. Selon notre analyse, ce genre d'hétérogénéité ne se présente pas dans le cas des ontologies générées pour les standards d'affaire puisqu'elles traitent le même domaine pragmatique et partagent les même termes techniques dans la définition des concepts. à la section suivante, nous proposons l'alignement d'ontologie comme solution aux problèmes d'hétérogénéité ainsi que les techniques et les approches existantes.

## 6.4 Définition d'alignement d'ontologies

L'alignement des ontologies permet d'améliorer l'intéropérabilité entre des applications distantes et de surmonter dans une certaine mesure le problème de l'hétérogénéité sémantique [46]. L'alignement est un processus qui prend en entrée deux ontologies et

trouve des liens sémantiques ou un ensemble de correspondances entre les entités de ces ontologies (des classes, des propriétés, des instances, etc.).

La figure 6.2 représente le schéma fonctionnel du processus d'alignement. Selon Euzenat et al. [14], il peut être vu formellement comme une fonction $f$ à partir d'une paire d'ontologies à aligner $O_1$ et $O_2$, un alignement de départ $A$ qui doit être complété par le processus d'alignement, un ensemble des paramètres $p$ comme des poids ou des seuils, et des resources externes $r$ (WordNet, thésaurus d'un domaine, etc.), retourne un alignement $A'$ entre ces deux ontologies :

$$A' = f(O_1, O_2, A, p, r)$$

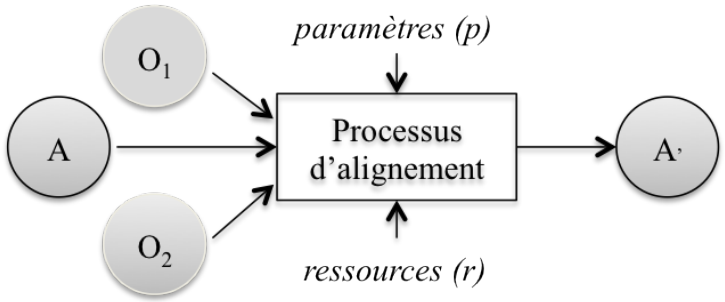



Figure 6.2 – Définition du processus d'alignement [14].

Le processus d'alignement consiste à définir le type de la relation entre les entités ainsi que leur degré de similarité (ou bien degré de confiance). Généralement, les algorithmes d'alignement utilisent une relation d'équivalence (=) signifiant que les deux entités sont identiques ou équivalentes.

### 6.4.1 Classification des techniques d'alignement

Selon la complexité des ontologies étudiées, il existe une variété de techniques d'alignement à utiliser. Euzenat et al. [14] présentent une étude approfondie de ces techniques. Ils les ont classifiées selon la méthode utilisée et le niveau de granularité adopté : terminologique, structurelle, extensionnelle et sémantique.

La méthode extensionnelle se base sur des instances des ontologies. Elle permet un meilleur alignement quand les deux ontologies partagent les mêmes individus. Quant

à la méthode sémantique, elle nécessite d'avoir une ontologie intermédiaire qui définit la connaissance partagée entre les deux ontologies à aligner ou d'un thésaurus contenant une organisation et description des termes du domaine d'affaire. Même si elles représentent de belles opportunités, nous nous limitons à présenter dans cette section les méthodes terminologiques et structurelles en fontion de nos resources disponibles car on ne dispose ni d'instances, ni d'un thésaurus distributionnel . Plusieurs systèmes d'alignement se basent sur WordNet comme une resource externe contenant la description du vocabulaire de la langue. Or, les ontologies que nous étudions portent sur le domaine des affaires avec des termes du domaine et des concepts définis par des termes qui n'apparaissent pas dans WordNet, par exemple `PurchaseOrderRequestType`.

#### 6.4.1.1 Méthode terminologique

Cette méthode s'intéresse au niveau élémentaire. Elle analyse les entités ou les instances sans tenir compte de leur relation avec d'autres entités ou instances. Il existe deux grandes catégories de méthodes, celles basées sur des caractères ou des tokens.

**Méthodes à base de caractères :** considèrent les noms des entités comme une séquence de caractères afin de trouver des similarités entre eux. Dans ce cas, la mesure de similarité se base sur la distance entre ces chaînes et diffère selon la mesure de calcul comme la distance de Hamming ou de Levenshtein.

La distance de Hamming consiste à compter le nombre de positions où les deux chaînes diffèrent. Elle est souvent utilisée Étant donné deux chaînes de caractères $s$ et $t$, la distance de Hamming entre $s$ et $t$ sera calculée comme suit :

$$Hamming(s,t) = \left( \sum_{i=1}^{min(|s|,|t|)} s[i] \neq t[i] \right) + abs(|s| - |t|)$$

Néanmoins, pour comparer des chaînes de caractères pouvant subir non seulement des substitutions, mais aussi des insertions ou des suppressions, des métriques plus sophistiquées comme la distance de Levenshtein sont plus adaptées qui prennent en considération le décalage dans les positions de caractères entre les deux chaînes à comparer.

La distance Levenshtein considère les noms des entités comme une séquence de caractères. Cette distance ajoute par rapport à la distance de Hamming, la possibilité de prendre en compte, lors de la comparaison de deux chaînes de caractères des insertions et des suppressions en plus des substitutions. Elle donne la similarité entre deux chaînes de caractères en se basant sur le nombre minimal des opérations élémentaires (substitution, insertion et suppression) à effectuer pour passer d'une chaîne à une autre. Quand la distance de Levenshtein est proche de 0, les deux chaînes à comparer sont assez similaires. Par contre, si cette distance tend vers la longueur de la plus grande chaîne alors les deux chaînes sont très différentes.

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Chaîne$_1$ : | d | i | s | c | o | u | n | t | | s | T | y | p | e |
| Chaîne$_2$ : | | | | c | o | u | n | t | r | y | T | y | p | e |
| Opérations : | INS | INS | INS | c | o | u | n | t | SUP | SUB | T | y | p | e |

L'exemple, ci-dessus, montre la distance de Levenshtein entre les deux chaînes de caractères `discountsType` et `countryType` égale à 5 puisqu'il y a 5 opérations nécessaires pour passer de la première chaîne à la deuxième ; trois insertions (INS) des caractères *d*, *i* et *s* suivit d'une suppression (SUP) du caractère *r* et enfin une substitution (SUB) est faite entre *s* et *y*. Par contre, la distance de Hamming entre ces deux chaînes est égale à 12.

L'algorithme de calcul de la distance de Hamming entre deux chaînes de caractère compare les caractères position par position par contre l'algorithme de la distance d'édition est meilleur, car il prend en considération lors de la comparaison des caractères le décalage dans leurs positions, ce qui nous semble plus adéquat pour notre cas de figure. Par exemple, les deux chaînes de caractères `PurchaseOrderRequestType` et `OrderRequestType` sont similaires puisqu'elles partagent les tokens `Order`, `Request` et `Type` et il suffit juste d'insérer le mot `Purchase` pour passer de la première chaîne à la deuxième. Donc, la distance Levenshtein est égale à 8 alors que la distance de Hamming est égale à 24 qui est beaucoup plus grande que la distance Levenshtein malgré que les chaînes sont similaires.

**Méthodes à base de tokens :** considérent le nom d'une entité comme une suite de mots

(tokens) en le décomposant, par exemple `WarrantyType` devient `Warranty Type`. La mesure de similarité peut être la métrique de Jaccard ou la mesure cosinus.

Cette approche est appropriée dans le cas des standards d'affaire, car d'une façon conventionnelle les noms des entités sont souvent des syntagmes nominaux formés d'une suite des noms anglais concaténés, chacun débutant par une majuscule.

La métrique de Jaccard détermine, entre deux suites de mots, le rapport entre le nombre de tokens communs et le nombre total de tokens distincts. Cette métrique trouve des similarités entre des chaînes ayant des tokens en commun, par exemple `NameAddress` et `PhysicalAddress`, mais pas entre `From` et `SenderType`.

Par contre, la mesure cosinus est fréquemment utilisée en tant que mesure de ressemblance entre deux documents et calculée selon la formule suivante :

$$cos(A,B) = \frac{\vec{A} * \vec{B}}{|\vec{A}| * |\vec{B}|} = \frac{\sum_{i=1}^{n} A_i * B_i}{\sqrt{\sum_{i=1}^{m} A_i^2} * \sqrt{\sum_{i=1}^{n} B_i^2}}$$

où $A_i$ et $B_i$ sont respectivement les poids du terme $i$ dans les documents $A$ et $B$, $m$ et $n$ sont les nombres de termes respectifs des documents $A$ et $B$.

En effet, une entité dans une ontologie est décrite par un ensemble de mots-clés représentés sous forme de vecteur où chaque entrée définit le poids du terme. Un terme d'indexation est un élément dont la sémantique décrit (partiellement) le contenu d'une entité dans une ontologie (le nom de l'entité, les commentaires, etc.). Un poids est généralement associé à chaque terme d'indexation pour refléter son importance dans la représentation du texte et son pouvoir discriminant. Cette étape consiste à pondérer l'ensemble des termes dans un document (entité). L'une des techniques de pondération connues en recherche d'information est le `TFIDF` [42].

`TFIDF` (Term Frequency Inverse Document Frequency) permet d'évaluer l'importance d'un terme dans un document ainsi que toute la collection. `TF` est une mesure de l'importance d'un terme dans un document. En général, cette valeur est déterminée par la fréquence du terme dans le document. Par contre, `IDF` est une mesure permettant de déterminer l'aspect discriminatoire d'un terme. Un terme qui a une valeur de `TFIDF` élevée apparaît souvent dans ce document, et rarement dans les autres. Le poids d'un terme

dans un document sera calculé comme suit :

$$\texttt{TFIDF}(t_i, d_j) = \texttt{TF}(t_i, d_j) * \texttt{IDF}(t_i)$$

Avec :

$$\texttt{IDF}(t_i) = \log \frac{|D|}{|d_j : t_i \in d_j|}$$

TF($t_i$, $d_j$) est le nombre d'apparitions du terme $t_i$ dans le document $d_j$. |D| est le nombre total de documents dans le corpus et | $d_j : t_i \in$d$_j$ | est le nombre de documents contenant le terme $t_i$.

#### 6.4.1.2 Méthode structurelle

Cette méthode prend en considération deux types de comparaison de structure des entités (interne et relationnelle). La méthode basée sur la structure interne des entités prend en considération un ensemble de critères comme le type des propriétés (`range`), les cardinalités, etc. Deux propriétés ayant un `domain` et un `range` similaires seront alignées. Par exemple, les propriétés `StreetSupplement2` et `AddressLine2`, dont le `range` est `string`, peuvent être alignées. Cette technique est limitée et doit être combinée avec d'autres comme les méthodes terminologiques afin de réduire le nombre de correspondances possibles [14]. En effet, une entité dans une ontologie peut avoir plusieurs propriétés partageant le même type de données.

Le deuxième type de comparaison s'intéresse aux relations entre les entités. Cette méthode consiste à représenter les ontologies comme des graphes conceptuels étiquetés. La similarité entre deux noeuds sera calculée en fonction de leur position dans le graphe. En d'autres termes, deux nœuds sont similaires si leurs voisins sont similaires.

### 6.4.2 Synthèse

Nous avons vu différentes techniques d'alignement à base de noms d'entités et de structure. La méthode terminologique est une étape préalable et incontournable dans le processus d'alignement d'ontologies. Ces mesures sont utilisées le plus souvent pour détecter si deux chaînes sont similaires. Cette méthode est performante dans le cas où

les concepteurs d'ontologie utilisent des chaînes semblables pour désigner les mêmes concepts. Plusieurs mesures de similarité ont été proposées comme la distance entre les chaînes ou la mesure de similarité cosinus. Ces techniques sont utilisées dans la plupart des systèmes d'alignement, p.e. OLA [13], avant d'appliquer des techniques structurelles.

Par contre, il est nécessaire d'utiliser d'autre sources d'information plus fiables comme la structure de l'ontologie soit interne ou relationelle. Ces techniques sont complémentaires afin d'obtenir un meilleur alignement.

Shvaiko et al. [14] ont proposé une revue des plus récents systèmes d'alignement d'ontologies. Dans la plupart de ces systèmes, les techniques élémentaires et structurelles ont été combinées pour rendre les aligneurs de plus en plus efficaces. Nous donnons des exemples de ces systèmes à la section 6.5.

## 6.5 Outils d'alignement existants

Dans cette section, nous décrivons quelques systèmes d'alignement considérés par l'OAEI (Ontology Alignment Evaluation Initiative) parmi les meilleurs. L'objectif des systèmes d'alignement d'ontologies est de réduire l'hétérogénéité entre les applications. Même s'ils ont un objectif commun, ils diffèrent au niveau de leur fonctionnement puisque chaque système traite ce problème à sa façon en fonction des techniques utilisées (e.g linguistique, structurelle), de la stratégie de combinaison de ces techniques, des mesures de calcul de similarité adoptées ainsi de l'ensemble de ressources externes utilisées (WordNet, thésaurus, etc.). Plusieurs systèmes d'alignement se basent sur WordNet comme resource externe. Or, les ontologies que nous étudions portent sur le domaine des affaires avec des termes du domaine et des concepts définis par des termes qui n'apparaissent pas dans WordNet, par exemple `PurchaseOrderRequestType`.

Nous pouvons classifier les systèmes d'alignement selon différentes dimensions :

- données d'entrée qui dépendent du modèle conceptuel avec lequel une ontologie est exprimée (schéma relationnel, orienté objets, RDF, OWL) et du niveau de granularité de données (schéma ou instance) ;

- processus d'alignement qui dépend du type de méthode (terminologique, structurelle, sémantique ou hybride) pour traiter les données et de la méthode de calcul de similarité entre les entités ;

- sortie de l'algorithme représentée par un ensemble de correspondances sous forme des quadruplets <$e_1$, $e_2$, r, d>, où $e_1$ et $e_2$ sont deux entités de deux ontologies $O_1$ et $O_2$ reliées par r avec un degré de confiance d. L'ensemble des correspondances peut être utilisé pour l'alignement des ontologies, la fusion, la migration des données entre les ontologies ou la traduction d'une ontologie vers une autre via des requêtes.

L'objectif des systèmes d'alignement d'ontologies est de réduire l'hétérogénéité entre les applications, mais elles diffèrent dans leurs méthodes d'appariement en fonction du type de données sur lesquelles les algorithmes travaillent : des chaînes de caractères (terminologique), la structure (structurel), des instances de données (extensionnel) ou des modèles (sémantique). Donc, le choix du système d'alignement adéquat se base sur les ressources disponibles ainsi que sur ses performances. Comme les ontologies étudiées sont écrites en OWL pour lesquelles nous ne possédons ni instances ni thésaurus, nous détaillons ces trois systèmes (OLA [13], Falcon [23] et ASMOV [26]) qui traitent ces critères et qui ont donné une bonne performance dans les OAEI.

### 6.5.1 OLA (OWL Lite Alignment)

OLA [13] est un système automatique d'alignement des ontologies décrites en OWL Lite et OWL DL. Il prend en entrée deux ontologies OWL et propose un ensemble de correspondances entre les entités de ces ontologies.

Djoufak et al. [9] présentent $OLA_2$ [1], une reformulation de l'algorithme OLA [13], qui a démontré des performances supérieures à la version précédente dans les exercices de l'OAEI [8].

$OLA_2$ représente une ontologie avec un graphe orienté et étiqueté dans lequel chaque sommet correspond à une entité de l'ontologie et chaque arc est une relation entre deux

[1] $OLA_2$ : https ://gforge.inria.fr/frs/ ?groupid=271

entités étiquetées par le nom du lien sémantique. Ces graphes d'ontologie seront utilisés dans la construction d'un graphe de similarité considéré comme le coeur de l'alignement en ajoutant la notion de pondération. Le calcul de similarité de tous les noeuds est itératif. La valeur de similarité d'un noeud à une itération `i+1` est recalculée en fonction des valeurs des nœuds adjacents obtenues à l'itération précédente, c-à-d l'ensemble des nœuds qui sont la source des arcs ayant ce nœud pour cible. Quand, les valeurs des nœuds du graphe de similarité ne varient plus entre deux itérations successives, l'algorithme a atteint un point fixe, dit point de convergence [9].

### 6.5.2 Falcon

Falcon est une approche automatique d'alignement d'ontologie en RDFS et OWL basée sur le principe diviser-pour-régner [23]. Il a été conçu pour gérer des ontologies avec un très grand nombre d'entités. L'approche fonctionne en trois phases : partitionner les ontologies en un ensemble de blocs, trouver des correspondances entre ces blocs, et trouver un alignement.

La première phase partitionne les entités (classes et propriétés) d'une ontologie donnée en un ensemble de clusters disjoints, en utilisant un regroupement structurel. Ensuite, Falcon construit des blocs à partir de ces clusters en attribuant à chaque bloc l'ensemble des entités formant un triplet RDF. Un triplet RDF est une association (sujet, prédicat, objet) où le sujet est la ressource à décrire, le prédicat définit le type de propriété applicable à cette ressource alors que l'objet représente la valeur de la propriété (une donnée ou une autre ressource).

La deuxième phase trouve des similarités entre chaque bloc des deux ontologies. Avant d'aligner les blocs, une mesure de similarité des chaînes de caractères est appliquée pour trouver les entités fortement liées, appelées ancres. Les blocs les plus similaires sont ceux possédant le plus d'ancres communes. Les paires de blocs ayant des valeurs de similitude élevées sont sélectionnées en fonction d'un seuil. Enfin, dans la troisième phase, les résultats des deux techniques linguistique et structurelle itérative sont combinées pour découvrir les alignements entre les paires de blocs appariés [46].

### 6.5.3 ASMOV (Automated Semantic Mapping of Ontologies with Validation)

ASMOV est une approche automatique pour l'alignement d'ontologies qui vise l'intégration de l'information pour la bioinformatique [26]. Cette approche est résumée en deux étapes : calcul de similarité et vérification sémantique. La première étape consiste à calculer la similarité entre les entités de deux ontologies comme la moyenne pondérée de plusieurs mesures de similarité ; lexicale par exemple avec la distance de Levenshtein, structurelle (hiérarchie des classes et type de propriétés) et extensionnelle (instances de classes et valeurs des propriétés). Elle utilise aussi plusieurs types de sources de connaissances externes telles que WordNet et UMLS. Ensuite, le système s'assure qu'il n'y a aucune inconsistance ou incohérence sémantique dans l'alignement obtenu. Ce processus est itéré jusqu'à ce qu'il ne détecte aucune nouvelle correspondance.

### 6.5.4 Discussion

Ces systèmes d'alignement diffèrent au niveau de la stratégie d'alignement, des mesures de similarité entre deux entités (linguistique, structurelle) ainsi que de la combinaison de ces mesures. Comme nous l'avons constaté à la section 6.4.2, les systèmes combinant les techniques terminologiques et structurelles donnent de meilleurs résultats.

La qualité des résultats de chaque système est évaluée en fonction de la métrique F-score dans les compétitions de OAEI [2]. Shvaiko et al. [46] ont réalisé une étude comparative entre ces systèmes en fonction de la mesure F-score de chaque système, calculée selon l'équation (6.1). Le meilleur F-score en 2009 a été obtenu par ASMOV (0,63), qui est de loin supérieur au 0,49 de OAEI-2008 et au 0.43 de Falcon en 2006. Toutefois, 0.71 est le meilleur F-score obtenu par $OLA_2$ lors de OAEI-2007 [46][8].

## 6.6 État de l'art sur l'application d'alignement dans le domaine d'affaire

Au sein d'un processus d'affaire, l'échange informatisé de données permet au système informatique d'une entreprise de communiquer et d'échanger électroniquement des documents commerciaux, tels que les bons de commande et les factures, avec d'autres

[2] OAEI : http ://oaei.ontologymatching.org/

systèmes informatiques. Un processus d'affaire est un ensemble d'activités à réaliser selon un ordre chronologique pour réaliser une tâche comme la gestion des commandes. La réalisation d'un processus se base sur trois niveaux de détails : la modélisation du processus d'affaire pour comprendre les activités principales du processus d'affaire, son déroulement ainsi que les participants, la gestion de la messagerie (les entrées/sorties) et la gestion des données définies dans des documents d'affaire.

Généralement, les systèmes d'entreprise sont développés indépendamment par diverses organisations et non nécessairement avec les mêmes structures et vocabulaires. Cela conduit à une hétérogénéité dans la syntaxe, la structure et la sémantique révélée lorsque ces systèmes interagissent. Plusieurs langages, tels que OWL-S [53] et WSML [34], ont été proposés pour l'intégration de la sémantique dans la modélisation des services web. L'utilisation des formalismes recommandés par le W3C, par exemple OWL, facilite l'adoption par les entreprises.

Malgré l'intégration de la sémantique par l'adoption des langages sémantiques, l'hétéréogénéité persiste. Le problème dans l'échange inter-entreprises réside dans l'existence de plusieurs modèles pour un même processus d'affaire, de différents moyens de communication ou même de différents modèles de documents. Pour garantir l'intéropérabilité dans les échanges, il est nécessaire de trouver des correspondances entre les différentes spécifications d'un processus d'affaire.

Au cours des dernières années, plusieurs travaux ont porté sur ce problème mais avec différents points de vue. Certains identifient des correspondances entre les activités représentant le même comportement dans les modèles d'affaire de deux entreprises [56][43][7], alors que d'autres s'intéressent aux documents [3].

Schubert et al. [43] ont étudié l'état actuel des techniques gérant la collaboration électronique entre les entreprises dans la chaîne d'approvisionnement. Leur projet cherche à élaborer un framework qui facilite la collaboration dans le B2B. À partir d'un ensemble de scénarios de processus d'affaire collectés de plusieurs entreprises parlant la langue allemande, ils prévoient construire un modèle d'affaire de référence décrivant les patterns partagés entre les entreprises avec le même contexte. Toutefois, il est difficile de convaincre les entreprises à changer leur scénario d'affaire vu les coûts et les efforts im-

pliqués. Techniquement, cette étude porte sur les scénarios d'affaire comme le niveau le plus abstrait dans le processus d'affaire sans discuter les causes de l'hétéreogénité dans les échanges entre les entreprises. Nous constatons que l'adoption des normes d'affaires par les entreprises qui collaborent a résolu d'une certaine façon le problème traité par cette étude. Dans notre étude, nous nous intéressons aux types d'hétéréogénéités pouvant exister dans un processus d'échange et leurs principales causes. Nous étudions les documents d'affaire comme un support de communication et source d'hétéréogénéité entre les entreprises.

Pour les services web, Zhu et al. [56] et Kim et al. [30] pensent que les problèmes majeurs dans la gestion des processus d'affaires sont liés aux difficultés rencontrées dans la recherche, la réutilisation des processus d'affaires, et la récupération des services web pertinents. Par contre, le problème majeur réside dans les types d'hétéréogenité dans les échanges entre les entreprises. Ils affirment que l'évaluation de la proximité entre ces services est une tâche toujours cruciale dans la réutilisation et dans le processus d'intégration. Ils présentent deux approches qui adaptent la notion d'alignement pour la recherche des services web similaires. La plupart de ces travaux donnent des propositions d'alignement abstraites sans nécessairement les appliquer à des standards existants afin de tester leur efficacité dans un environnement pratique.

D'autres approches s'intéressent principalement à la représentation des données définies dans des documents d'affaire. Nous détaillons ces approches puisque nous partageons la même problématique liée à l'hétéreogénité dans les documents d'affaire.

Cardoso et al. [3] ont développé un système semi-automatique, nommée B2BISS, permettant de trouver des correspondances entre différentes représentations de données avec un niveau d'expressivité distinct ; syntaxique basée sur XML ou sémantique par le langage OWL. Lorsqu'une entreprise reçoit une transaction définie en XML, le système trouve des correspondances entre les éléments de cette transaction et les concepts ontologiques présentés dans l'ontologie interne de l'entreprise en se basant sur des règles de transformation stockées dans une base des règles. Il existe trois cas de figure pris en compte par B2BISS : (a) pas de correspondance, (b) une correspondance partielle, et (c) une correspondance complète.

Le premier cas apparaît quand il n'y a aucune règle de transformation dans la base de B2BISS permettant la transformation des éléments XML de cette transaction en un concept de l'ontologie. Par conséquent, de nouvelles règles doivent être créées manuellement en utilisant l'outil JXML2OWL [51]. Par contre, si les règles de transformation peuvent partiellement transformer la transaction, alors l'intervention de l'utilisateur sera nécessaire pour créer manuellement les règles manquantes. Une fois créées, elles seront stockées dans la base des règles afin d'être réutilisées sans donner des détails sur la façon adoptée pour la réutilisation. Dans le troisième cas, B2BISS a trouvé dans sa base un sous-ensemble de règles qui peuvent transformer tout les éléments de document XML correspondant à la transaction reçue.

Cette approche souffre de certaines limites : les règles de transformation sont créées manuellement par un expert de l'entreprise sans détailler les critères utilisées pour construire ces règles. Dans le cas du PIP3A4 de RosettaNet décrivant un bon de commande, la projection est coûteuse en temps puisque le document contient plusieurs éléments. Il n'y a pas de processus de validation des régles créées par l'expert. De plus, il faut mettre à jour toutes les règles dans la base lors d'un changement dans l'ontologie de l'entreprise. Enfin, le processus de B2BISS ne possède pas d'étape de validation pour vérifier la consistance des résultats obtenus.

Nous avons discutés, à la section 5.2, le travail de García et al. [17] qui proposent une approche automatique de transformation de XML Schema vers OWL, nommée ReDeFer utilisé pour transformer ebXML, l'une des principales normes de B2B qui modélise les processus d'affaire [17] et non les documents qui sont la source des échanges entre les entreprises. Nous avons essayé d'utiliser ReDeFer pour transformer les standards que nous avons étudiés, mais le résultat obtenu n'était pas valide ni syntaxiquement ni sémantiquement.

Ils ont également proposé un alignement entre les processus d'affaire définis par ebXML BP et BPEL-WS en utilisant l'outil OWL Ontology Aligner [55]. Le choix de ce système d'alignement n'est pas très détaillé et la performance de l'alignement n'a pas été évaluée. Nous aurions voulu tester leur outil d'alignement sur ces standards d'affaire afin de comparer ces résultats d'alignement avec les nôtres, mais ce système n'était plus

disponible en ligne.

## 6.7 Expérimentation

L'alignement des ontologies présente un grand intérêt pour plusieurs domaines d'application qui manipulent des connaissances hétérogènes. En e-business, deux entreprises doivent partager leurs connaissances, plus spécifiquement leurs modèles de documents. Actuellement, plusieurs ontologies ou standards d'affaire ont été développés indépendamment par différents organismes en fonction des besoins de leur application. Donc, le passage d'une ontologie à une autre devient de plus en plus nécessaire. Ce passage sera possible à l'aide des techniques d'alignement d'ontologie qui aident à trouver des correspondances entre les entités formant deux documents d'affaire modèlisés en ontologies.

La plupart des systèmes d'alignement ont été évalués sur un ensemble de jeu de test (benchmark) durant les OAEIs ou appliqués à des ontologies portant sur le domaine de la bio-informatique comme ASMOV [26]. Dans cette section, nous sortons de ce cadre expérimental pour traiter d'autres ontologies portant sur le domaine des affaires en particulier la gestion des bons de commande. Dans cette section, nous appliquons des techniques d'alignement sur les ontologies des documents d'affaire de la gestion des bons de commande définis par les standards xCBL, cXML et RosettaNet, discutées à la section 6.3.

La plupart des systèmes d'alignement ne se basent que sur les techniques terminologiques et structurelles et n'utilisent que rarement les méthodes extensionnelles ou sémantiques [46]. Cette limite est due aux manques des ressources disponibles : que ce soit l'absence d'instances pour appliquer les techniques extensionnelles ou bien d'un thésaurus définissant les termes du domaine [46]. Dans notre cas, l'absence de ressources disponibles nous oblige à nous contenter d'expérimentations basées sur des méthodes terminologiques et structurelles.

Nous avons effectué deux expérimentations. Dans la première, nous utilisons deux méthodes terminologiques, une à base de caractères comme la distance Levenshtein ap-

pliquée sur les noms des entités, et une autre à base de tokens comme la technique TFIDF utilisant les noms et les commentaires des entités des ontologies. Ces techniques sont les plus utilisées par les systèmes d'alignement présentés dans la littérature [14] y compris ceux présentés à la section 6.5. Nous voulons tester la performance de ces techniques terminologiques sur les ontologies des documents d'affaire puisqu'elles appartiennent au même domaine d'intérêt qui est la gestion des bons de commande. En plus, nous vérifions s'il y a une amélioration apportée par la structure de ces ontologies dans les résultats d'alignement par rapport aux méthodes terminologiques utilisées.

Le choix de la distance Levenshtein par rapport aux autres métriques comme la distance de Hamming a été justifié dans la section 6.4.1.1. Ce choix réside dans la nature des noms des entités des ontologies des documents d'affaire étudiés qui partagent un ensemble de tokens.

Il est utile de tirer avantage d'autres sources d'information liées aux entités de l'ontologie, p.e. les annotations contenant la définition et la description des entités. Dans ce cas, l'algorithme de la distance Levenshtein n'est pas adéquat puisqu'il se base sur des chaînes de caractères courtes (les noms des entités). Pour des chaînes de caractères plus longues (plusieurs mots), il faut utiliser les algorithmes de Jaccard ou la mesure cosinus utilisant TFIDF comme méthode de pondération des termes. La mesure de Jaccard est utile surtout pour étudier la similarité entre des ensembles binaires puisqu'elle ne considère pas la fréquence des termes dans un document ou bien dans la collection pour déterminer leur importance. Alors que, TFIDF, une méthode de recherche d'information utilisée par plusieurs systèmes d'alignement, est la plus adéquate puisqu'elle permet d'évaluer l'importance d'un terme dans un document ainsi que toute la collection.

Dans la deuxième expérimentation, nous utilisons l'algorithme $OLA_2$ qui a donné les meilleures performances dans les OAEIs comme discuté à la section 6.5.4. Nous appliquons cette méthode structurelle afin de voir s'il y a une amélioration apportée par la structure dans la performance par rapport aux méthodes terminologiques utilisées. Notre but est d'obtenir le meilleur alignement entre les standards d'affaire quelque soit la technique.

L'évaluation de la performance sera faite à l'aide des mesures classique de la re-

cherche d'information : Précision ($P$), Rappel ($R$) et F-score ($F$). Nous comparons les résultats de l'aligneur avec un alignement que nous avons fait manuellement en fonction des définitions d'entités offertes dans la documentation de chaque standard. Ces métriques seront calculées comme suit :

$$P = \frac{TP}{TP+FP} \qquad R = \frac{TP}{TP+FN} \qquad F = 2 * \frac{P*R}{P+R} \tag{6.1}$$

où $TP$ est le nombre de bonnes correspondances correctement identifiés, $FP$ est le nombre de mauvaises correspondances identifiées comme bonnes, et $FN$ est le nombre de bonnes correspondances non identifiées par l'algorithme.

### 6.7.1 Alignement basé sur deux méthodes terminologiques

Les ontologies étudiées s'intéressent à la chaîne d'approvisionnement plus spécifiquement à la gestion des bons de commande. Avant d'entamer l'alignement avec $\text{OLA}_2$, nous essayons d'aligner chaque ontologie $O_1$ et $O_2$ en utilisant une méthode terminologique à base de la technique `TFIDF` et de la distance Levenshtein, car les ontologies ont le même domaine d'intérêt et partagent donc les mêmes termes techniques dans la définition des concepts. Ces techniques ont été détaillées à la section 6.4.1.

Nous avons défini la distance Levenshtein comme le nombre minimal d'opérations d'édition (substitution, ajout et suppression) à effectuer pour passer d'une chaîne à une autre. La mesure de similarité entre deux chaînes de caractères $\texttt{ch}_1$ et $\texttt{ch}_2$ est résumée dans l'équation 6.2. Nous présentons ici la version normalisée de la distance Levenshtein divisée par la longueur de la plus longue chaîne entre $\texttt{ch}_1$ et $\texttt{ch}_2$. Plus que, cette mesure est proche de 1 plus les deux chaînes sont similaires.

$$similarite(ch_1, ch_2) = 1 - \frac{Levenshtein(ch_1, ch_2)}{max(|ch_1|, |ch_2|)} \tag{6.2}$$

Par contre avec la technique `TFIDF`, chaque entité dans l'ontologie (classes ou propriétés) est considérée comme un document où les termes sont les noms des entités (classes et propriétés) ainsi que le contenu des commentaires, s'ils existent.

Une entité de l'ontologie $O_1$ est alignée avec une entité, de même type (classe ou

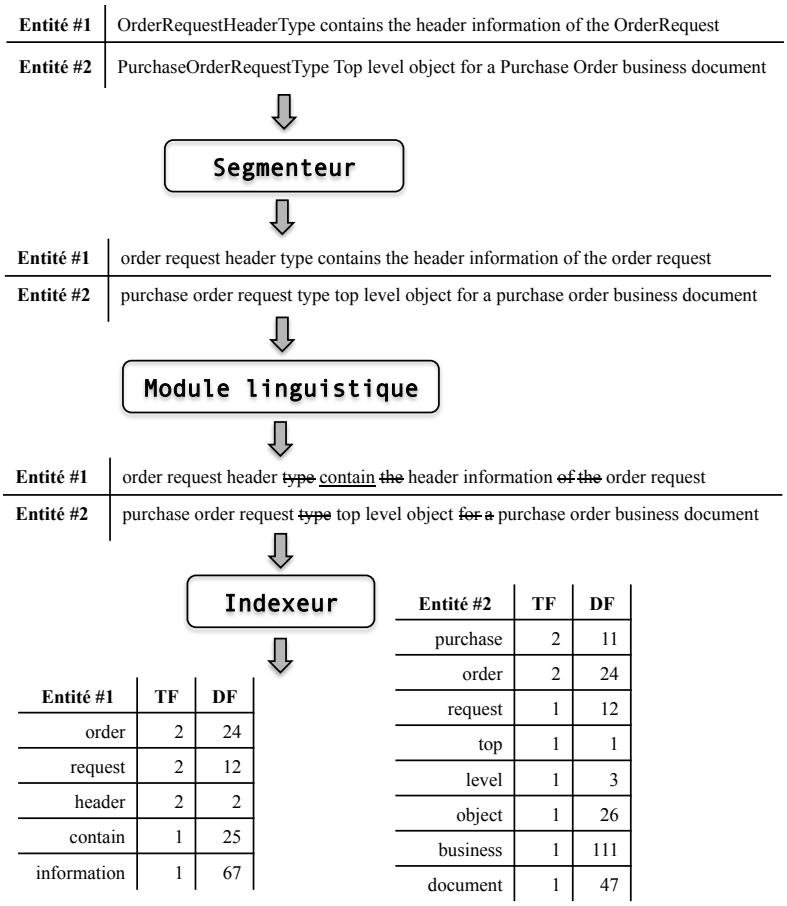



Figure 6.3 – Application de la méthode `TFIDF` sur deux classes `OrderRequestHeaderType` (**Entité #1**) et `PurchaseOrderRequestType` (**Entité #2**) des ontologies des standards xCBL et RosettaNet.

propriété), de l'ontologie $O_2$ la plus similaire en fonction de la mesure cosinus calculée entre les deux vecteurs de termes représentant ces entités.

La figure 6.3 montre les étapes du processus d'indexation en utilisant la définition (nom et commentaire) des classes `OrderRequestHeaderType` (**Entité #1**) et `PurchaseOrderRequestType` (**Entité #2**) appartenant respectivement aux ontologies des standards xCBL et RosettaNet.

Le processus d'indexation est composé de trois modules : le segmenteur, le module linguistique et l'indexeur. Tout d'abord, le segmenteur décompose les mots complexes en unités, par exemple, le mot `OrderRequest` sera décomposé en deux tokens `order` et `request`. Ensuite, le module linguistique contient deux étapes principales : l'élimina-

tion des mots fréquents à ignorer (stopwords) représentés par des mots barrés à la figure 6.3, et la lemmatisation, c.à.d ne garder que la forme canonique d'un mot appelé lemme, par exemple le mot souligné `contain` à la figure 6.3. Enfin, le résultat de l'indexeur est un vecteur de termes $\vec{s}$, appelé `index`, où chaque entrée représente un terme `t` défini par sa fréquence (`TF`) dans l'entité ainsi que le nombre d'entités (`DF`) contenant ce terme dans leurs noms ou leurs commentaires. Par exemple, le terme `order` apparaît deux fois dans la définition de la classe `OrderRequestHeaderType` par contre il apparaît dans 24 classes parmi l'ensemble des classes de xCBL et RosettaNet.

Tableau 6.IV – Les résultats de l'alignement effectué sur les ontologies des standards xCBL et cXML avec RosettaNet. On y trouve les mesures de Précision, Rappel et F-score de l'utilisation de la méthode terminologique `TFIDF` (**`T`**) et la distance Levenshtein (**`L`**) sur ces standards. Avec la technique `TFIDF`, la mesure de similarité adoptée est la mesure cosinus calculée entre les deux vecteurs de termes représentant les entités de deux ontologies. Les valeurs en gras représentent les F-score de la meilleure technique.

| | **xCBL** | | | | **cXML** | | | |
|---|---|---|---|---|---|---|---|---|
| | **`Request`** | | **`Confirm`** | | **`Request`** | | **`Confirm`** | |
| | **`L`** | **`T`** | **`L`** | **`T`** | **`L`** | **`T`** | **`L`** | **`T`** |
| **Nombre d'entités** | 331 | 331 | 136 | 136 | 69 | 69 | 88 | 88 |
| **TP** | 107 | 176 | 60 | 68 | 34 | 31 | 34 | 47 |
| **FP** | 224 | 155 | 76 | 68 | 69 | 69 | 54 | 41 |
| **TP+FN** | 298 | 298 | 122 | 122 | 62 | 62 | 81 | 81 |
| **Précision** | 32% | 53% | 44% | 50% | 51% | 55% | 39% | 53% |
| **Rappel** | 36% | 59% | 49% | 56% | 56% | 61% | 42% | 58% |
| **F-score** | 34% | **56%** | 46% | **53%** | 54% | **58%** | 40% | **56%** |

La performance de cette technique est illustrée au tableau 6.IV. À partir de ce tableau, nous remarquons que la technique `TFIDF` est meilleure que la distance Levenshtein. Cette technique a donné une performance moyenne de 56% pour la plupart des standards étudiés contre 43% pour Levenshtein. Dans le cas des documents, `OrderRequestType` du xCBL avec `PurchaseOrderRequest` celui de RosettaNet (bons de commande contenant les informations du client, de la facturation, la livraison, etc), la mesure F-score est de 56% pour `TFIDF` et de 34% pour la distance Levenshtein.

Selon la figure 6.4, les résultats obtenus avec la technique `TFIDF` sont prometteurs

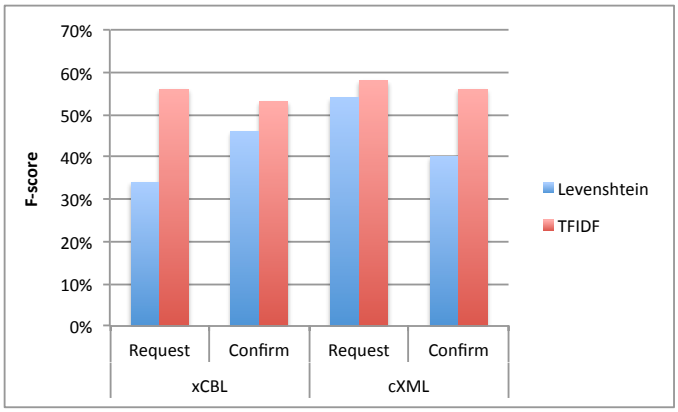



Figure 6.4 – Représentation de la mesure F-score calculée pour chaque standard d'affaire (xCBL et cXML) dans le cadre du `request` et du `confirm`. L'axe vertical représente les valeurs de cette mesure pour les techniques `Levenshtein` et `TFIDF`.

par rapport à ceux de `Levenshtein`. Nous concluons que ces ontologies partagent les termes les plus fréquents et les plus discriminants. Cette constation paraît logique puisque ces ontologies traitent du même domaine.

Après une analyse manuelle des résultats, nous avons remarqué que, même si la technique `TFIDF` est meilleure dans l'ensemble, certaines correspondances ont été bien identifiées par la distance Levenshtein et non pas avec `TFIDF`. Nous rappellons qu'une correspondance représente la relation entre deux entités $e_1$ et $e_2$ appartenant respectivement aux ontologies $O_1$ et $O_2$ à aligner.

Tableau 6.V – Alignement du concept `OrderRequest` de cXML avec ceux de RosettaNet en utilisant les deux méthodes terminologiques `TFIDF` et `Levenshtein`. Les valeurs en gras représentent les meilleures valeurs d'alignement obtenues.

| | **TFIDF** | **Levenshtein** |
|---|---|---|
| **PurchaseOrderRequestType** | 0.49 | **0.50** |
| **RequestingOrderLineItemReferenceType** | **0.59** | 0.25 |

L'exemple du tableau 6.V montre l'alignement de la classe `OrderRequest` du document `OrderRequest` de cXML avec celui de RosettaNet en utilisant les deux techniques `TFIDF` et Levenshtein. Avec `TFIDF`, cette classe a été mal alignée avec la classe `RequestingOrderLineItemReferenceType` du RosettaNet. La mesure cosinus calculée est 0.59, meilleure que celle avec l'entité `PurchaseOrderRequestType` de 0.49.

Par contre, à l'aide de la distance Levenshtein, elle a été bien alignée avec l'entité `PurchaseOrderRequestType` avec une mesure de similarité égale à 0.5. La qualité des commentaires utilisés pour définir les concepts de chaque ontologie est la cause du problème de la technique `TFIDF` dont l'impact sur la qualité du résultat d'alignement est justifiée par le fait que les deux entités partagent les termes les plus fréquents.

Beaucoup de faux positifs ont été identifiées par les méthodes terminologiques qui se basent sur les noms des entités et non pas sur leur structure. Avec la distance Levenshtein, deux chaînes peuvent être très similaires tout en désignant des concepts différents, par exemple dans le cas des deux chaînes `ProductReferenceType` et `TaxReferenceType`. De même, nous pouvons avoir deux chaînes qui ne sont pas similaires avec la distance Levenshtein mais qui le sont sémantiquement. Donc, il est nécessaire d'utiliser d'autre sources d'information pour éliminer l'hétéréogéneité.

L'information structurelle de l'ontologie peut aider à réduire les problèmes des méthodes terminologiques mentionnés ci-haut. À l'aide de $OLA_2$, une technique combinant la méthode terminologique et structurelle, nous allons voir s'il y a une amélioration au niveau des résultats de l'alignement. À notre connaissance, il n'y a pas encore de travaux qui ont abordé l'alignement de standards d'affaire, dus à l'absence d'ontologies qui les définissent.

### 6.7.2 Alignement basé sur l'algorithme $OLA_2$

Comme nous en avons discuté à la section 6.5.1, le système $OLA_2$ prend en entrée deux ontologies OWL. Comme le format pris en compte par $OLA_2$ est RDF/XML, nous avons choisi ce format de sortie pour notre processus de transformation. Nous avons dû toutefois effectuer une étape de pré-traitement. Comme les ontologies étudiées possèdent plusieurs `owl:import` qui ne sont pas pris en compte par $OLA_2$, nous avons dû adapter nos ontologies au format accepté par $OLA_2$, et intégrer récursivement l'ensemble des entités des fichiers importés. Pour ce faire, nous avons développé un programme Java qui utilise OWL API [3] ,une API Java pour créer et manipuler des ontologies OWL.

[3] owlapi.sourceforge.net

**Algorithm 1** : $OLA_2$

**Entrée** : $\mathcal{O}_1$ et $\mathcal{O}_2$, deux ontologies décrites en OWL
**Sortie** : $\mathcal{C}$, un ensemble de correspondances entre entités de $\mathcal{O}_1$ et $\mathcal{O}_2$

```
1 :   G1 = OWL2OntologyGraph(O1) ;
2 :   G2 = OWL2OntologyGraph(O2) ;
3 :   S  = ComputeSimilarityGraph(G1, G2) ;
4 :   V  = ComputeInitialCoefficients(S) ;
4 :   M = ComputeAdjacencyMatrix(S) ;
5 :   repéter
6 :           M' = UpdateAdjacencyMatrix(M,V) ;
7 :           V  = ComputationalComponentIteration(M',V) ;
8 :   jusqu'à convergence
9 :   C = HungarianExtraction(V) ;
```

Figure 6.5 – Le fonctionnement de $OLA_2$ adapté de Djoufak et al. [9].

$OLA_2$ est une séquence de cinq étapes présentée à la figure 6.5. Les deux ontologies à aligner sont représentées sous forme des graphes par la fonction `OWL2OntologyGraph` (lignes 1 et 2), où chaque sommet correspond à une entité de l'ontologie et chaque arc est une relation entre deux entités. Un graphe de similarité est obtenu à partir des graphes d'ontologie via la fonction `ComputeSimilarityGraph` (ligne 3). Le vecteur de similarité et la matrice d'adjacence sont obtenus, à partir du graphe de similarité, avec les fonctions `ComputeInitialCoefficients` et `ComputeAdjacencyMatrix` [9] (ligne 4).

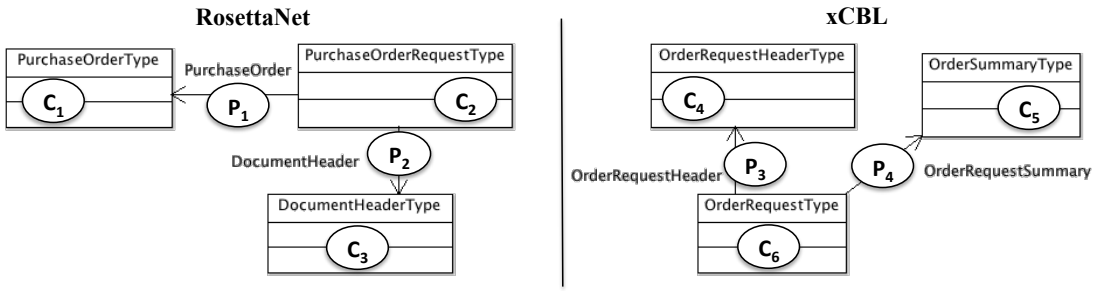



Figure 6.6 – Représentation des extraits d'ontologies de xCBL et RosettaNet en diagramme de classes. Les classes (**C**) sont numérotées de $\mathbf{C}_1$ à $\mathbf{C}_6$ et les propriétés (**P**) de $\mathbf{P}_1$ à $\mathbf{P}_4$.

Nous illustrons le fonctionnement de l'algorithme de $OLA_2$ avec un exemple des ontologies de xCBL et RosettaNet. Les ontologies de la figure 6.6 sont représentées par des diagrammes de classes UML, où les concepts ontologiques correspondent à des classes UML, denotées par $\mathbf{C}_1$ à $\mathbf{C}_6$, et les propriétés ontologiques à des variables de

classe ou à des associations reliant deux classes, denotées par $\mathbf{P}_1$ à $\mathbf{P}_4$.

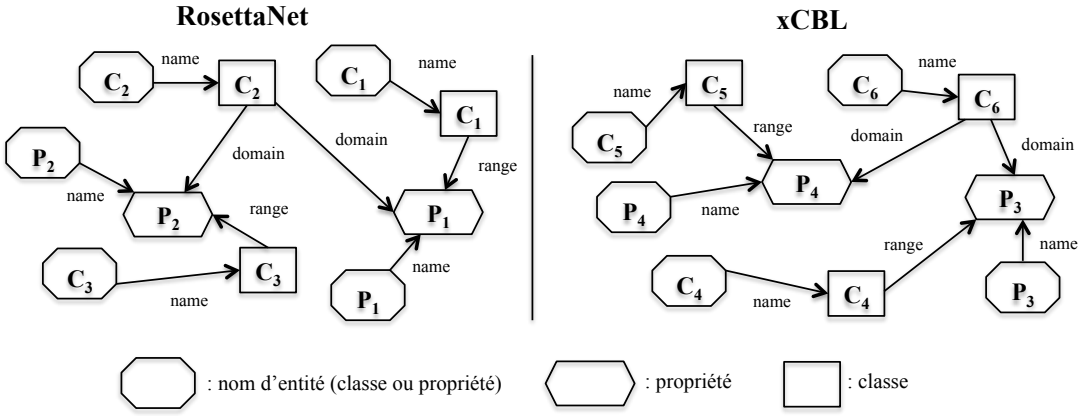



Figure 6.7 – Représentation de deux graphes d'ontologies de xCBL et RosettaNet correspondant aux diagrammes de classes de la figure 6.6 selon la notation présentée par Djoufak et al [9].

Chaque ontologie est représentée par $OLA_2$ sous forme d'un graphe, comme le montre la figure 6.7, où les nœuds représentent les entités de ces ontologies, les arcs sont les liens reliant ces noeuds (name, range, domain, etc.) et la forme dénote la catégorie de l'entité représentée par le nœud. Par exemple, la classe `PurchaseOrderType` de RosettaNet ($\mathbf{C}_1$) est le `range` de la propriété `PurchaseOrder` ($\mathbf{P}_1$) qui correspond au lien de type `range` reliant le rectangle de la classe $\mathbf{C}_1$ et l'hexagone de la propriété $\mathbf{P}_1$ dans le graphe d'ontologie de RosettaNet.

Ces graphes d'ontologie seront utilisées dans la construction du graphe de similarité de la figure 6.8 entre les deux ontologies à aligner. Les noeuds d'un graphe de similarité sont des triplets ($e_1$, $e_2$, $s$) avec $e_1$ et $e_2$ sont deux entités de la même catégorie (class, data property ou object property) appartenant aux graphes des ontologies $O_1$ et $O_2$ et $s$ est le degré de similarité calculé au moyen d'une distance d'édition entre les deux étiquettes représentant les entités de chaque noeud. Les relations entre les noeuds d'un graphe de similarité sont représentées par des arcs orientés et pondérés ayant comme étiquette le type du lien (range, domain, name, etc). Ces arcs existent si et seulement s'il y a simultanément un lien entre les entités formant les deux noeuds du graphe de similarité [9]. Par exemple, les deux noeuds, encerclés dans le graphe de similarité de la figure 6.8, sont connectés puisqu'il y a un lien de type `domain`, à partir du graphe d'ontologie, qui

relie en même temps l'entité `OrderRequestType` ($\mathbf{C}_6$) avec `OrderRequestHeader` ($\mathbf{P}_3$) et l'entité `PurchaseOrderRequestType` ($\mathbf{C}_2$) avec `DocumentHeader` ($\mathbf{P}_2$). Les poids des arcs sont introduits dans un fichier des paramètres à l'algorithme $OLA_2$. Selon Djoufak et al. [9], ces poids sont calculés par un processus externe à $OLA_2$ qui s'assure que les résultats fournis par ce dernier sur le jeu de test benchmarks d'OAEI sont les meilleurs mais sans donner de détail sur ce processus. Nous avons utilisé le fichier des paramètres offert par la version en ligne de $OLA_2$ pour réaliser cette expérience.

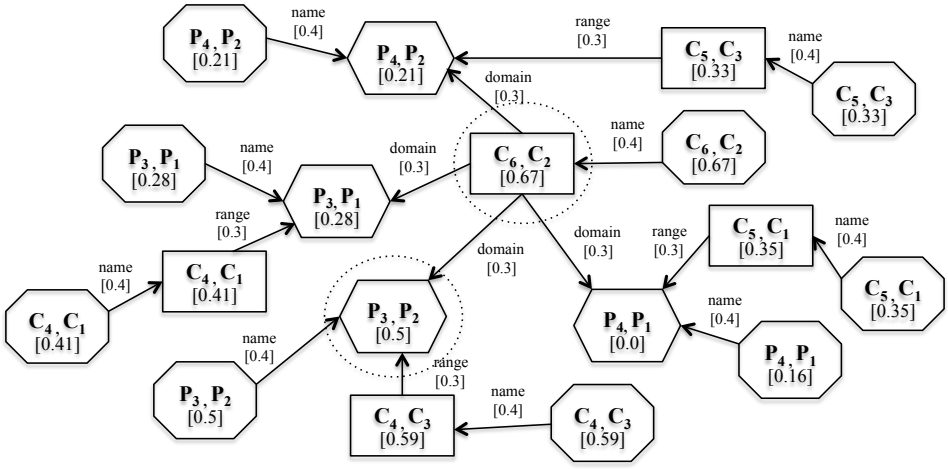



Figure 6.8 – Représentation du graphe de similarité à partir de deux graphes d'ontologies de xCBL et RosettaNet de la figure 6.7.

La matrice d'adjacence `M` et le vecteur de similarité des noeuds `V` seront produits à partir du graphe de similarité. Le vecteur de similarité $V_0$ représente les poids initiaux des nœuds du graphe de similarité. Alors que, la matrice d'adjacence représente l'influence de la similarité de chaque nœud incident à la similarité du nœud cible. La figure 6.9 présente un extrait de la matrice d'adjacence initiale correspondante au graphe de similarité de la figure 6.8. Les valeurs des cases $M_{i,j}$ de la matrice initiale $M_0$ sont les valeurs figurant sur les arcs du graphe de similarité reliant deux noeuds représentés par les indices `i` et `j` dans la matrice, sinon l'absence d'arc est noté par 0 [9]. Par exemple, la valeur de la case $M_{2,3}$ est 0.3 puisqu'il y a un lien de type `domain` de poids 0.3 reliant les noeuds **<**$\mathbf{C}_6$,$\mathbf{C}_2$**>** et **<**$\mathbf{P}_3$,$\mathbf{P}_2$**>** encerclés dans le graphe de similarité de la figure 6.8.

À chaque itération `i` de l'algorithme, le calcul de similarité est défini par le produit

| i \ j | $C_4$ , $C_1$ | $C_6$ , $C_2$ | $P_3$ , $P_2$ | …. |
|---|---|---|---|---|
| $C_4$ , $C_1$ | 0 | 0 | 0 | …. |
| $C_6$ , $C_2$ | 0 | 0 | 0.3 | …. |
| $P_3$ , $P_2$ | 0 | 0 | 0 | …. |
| …. | …. | …. | …. | …. |

(j=3 ⇩ sur la colonne $P_3$ , $P_2$ ; i=2 ⇨ sur la ligne $C_6$ , $C_2$)

Figure 6.9 – Représentation d'un extrait de la matrice d'adjacence initiale correspondant aux noeuds encerclés du graphe de similarité de la figure 6.8.

du vecteur de similarité $V_i$ par la matrice d'adjacence $M_i$ qui sera mise à jour à chaque itération.

Avant chaque itération, $OLA_2$ sélectionne, au moyen de l'algorithme d'appariement de Hopcroft et Karp [22], les arcs qui seront utilisés pour calculer la similarité. À une itération `i+1`, la valeur de similarité d'un noeud <$e_1$,$e_2$> dépend principalement des similarités, calculées à une itération `i`, de noeuds sélectionnés par $OLA_2$ ayant ce noeud pour cible. Les poids des arcs reliant le noeud <$e_1$,$e_2$> avec ceux non sélectionnés prennent la valeur 0 en mettant à jour les entrées correspondantes dans la matrice d'adjacence $M_i$.

Quand les valeurs de similarités restent constantes après deux itérations successives, l'algorithme produit les alignements en choisissant les couplages maximaux du vecteur de similarités.

La performance de cet algorithme dans le cas des standards d'affaires est illustrée au tableau 6.VI. Nous remarquons qu'il y a une amélioration dans le résultat d'alignement de l'algorithme $OLA_2$ par rapport aux méthodes terminologiques. Des faux positifs identifiés par `TFIDF` et `Levenshtein` sont devenus des vrais positifs dans le cas de $OLA_2$.

Le tableau 6.VII montre un exemple d'alignement de `StreetSupplement2`, une propriété de xCBL, avec les différentes techniques d'alignement. Ni `TFIDF`, ni `Levenshtein` n'avait aligné ce concept, mais que $OLA_2$ a réussi à le faire.

Tableau 6.VI – Les résultats de l'alignement effectué sur les ontologies des standards xCBL et cXML avec RosettaNet. Ces résultats montrent les mesures de Précision, Rappel et F-score de l'utilisation de l'algorithme $OLA_2$ sur ces standards.

| | **Alignement avec $OLA_2$** | | | |
|---|---|---|---|---|
| | **xCBL** | | **cXML** | |
| | **`Request`** | **`Confirm`** | **`Request`** | **`Confirm`** |
| **Nombre d'entités** | 331 | 136 | 69 | 88 |
| **TP** | 193 | 86 | 37 | 50 |
| **FP** | 138 | 49 | 21 | 24 |
| **TP+FN** | 298 | 122 | 62 | 81 |
| **Précision** | 58% | 64% | 64% | 67% |
| **Rappel** | 65% | 72% | 60% | 62% |
| **F-score** | 61% | 68% | 62% | 64% |

Tableau 6.VII – Alignement du concept xCBL `StreetSupplement2` en utilisant `TFIDF` et `Levenshtein` sur les caractères ainsi que $OLA_2$.

| | **Alignement de `StreetSupplement2`** | **Vrai/Faux** |
|---|---|---|
| **TFIDF** | Information | Faux |
| **Levenshtein** | TransportEvent | Faux |
| **$OLA_2$** | AddressLine2 | Vrai |

Nous avons rencontré des difficultés dans l'exécution de $OLA_2$ sur une machine avec de performance moyenne surtout dans le cas de xCBL où l'algorithme s'est planté dû au manque d'espace mémoire et à la taille des ontologies de plus en plus grande. Dans ce cas, l'exécution de $OLA_2$ a nécessité l'utilisation d'un ordinateur d'expérimentation de 64bits. La figure 6.10 montre la variation du temps d'exécution en fonction de l'espace mémoire réservé pour $OLA_2$ dans l'alignement de xCBL avec RosettaNet au niveau du `request` et du `confirm`.

À partir de la figure 6.10, nous constatons que l'alignement de xCBL avec RosettaNet au niveau du `request` a pris plus de temps que celui du `confirm` quelque soit l'espace mémoire réservé. au tableau 6.VI, nous remarquons que l'ontologie de xCBL au niveau de `request` est plus grande que celle du `confirm` puisqu'elle contient 331 entités entre classes et propriétés par rapport à 136 entités pour le `confirm`, ce qui en-

traîne un temps d'exécution élévé par rapport au `confirm`. Le temps d'exécution de $OLA_2$ dépend principalement de la taille de l'ontologie.

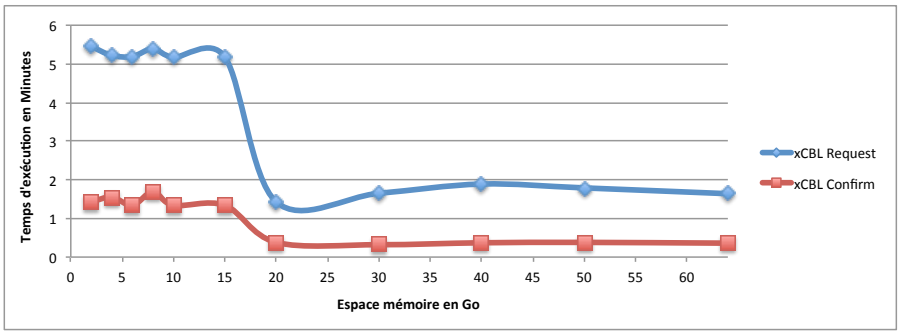



Figure 6.10 – Variation du temps d'exécution en fonction de l'espace mémoire réservé pour $OLA_2$ dans l'alignement de xCBL avec RosettaNet au niveau du `request` et du `confirm`.

### 6.7.3 Comparaison des résultats

La qualité des résultats de chaque système est évaluée en fonction de la métrique F-score dans les compétitions de OAEI. La figure 6.11 représente les mesures F-score obtenues en utilisant les techniques `Levenshtein`, `TFIDF` et l'algorithme $OLA_2$.

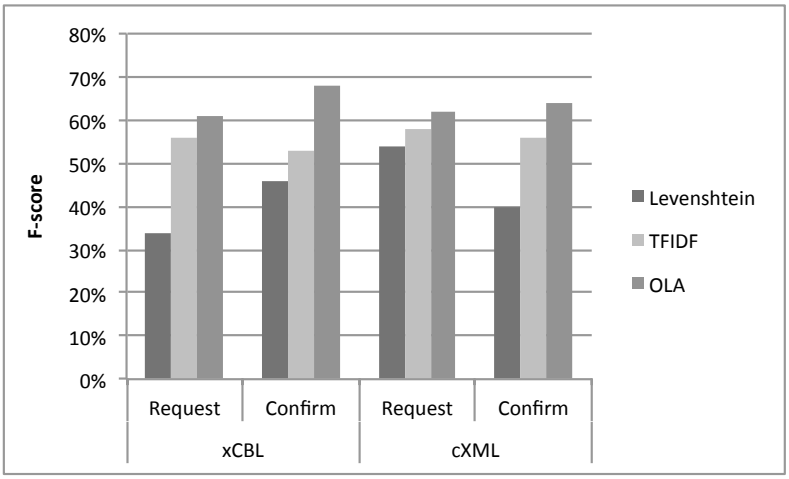



Figure 6.11 – Représentation de la mesure F-score calculée pour chaque standard d'affaire (xCBL et cXML) dans le cadre du `request` et du `confirm`. L'axe vertical représente les valeurs de cette mesure pour les techniques `Levenshtein`, `TFIDF` et l'algorithme $OLA_2$.

À partir de ce graphique, nous constatons que l'algorithme $OLA_2$ a donné les meilleures performances pour toutes les ontologies étudiées avec un F-score moyen de 64% par

rapport à `TFIDF` de 58% et `Levenshtein` de 43%. Par contre, la méthode `TFIDF` est meilleure que `Levenshtein`. La distance Levenshtein est limitée au noms d'entités. Elle est performante dans le cas où les concepteurs d'ontologie utilisent des chaînes semblables pour désigner les mêmes concepts. Alors que la méthode `TFIDF` en plus des noms des entités, elle prend en considération leurs commentaires, s'ils existent.

En conclusion, l'algorithme $OLA_2$ est une méthode générique qui reste la plus performante sur des ontologies qui ne possèdent ni commentaires, ni les mêmes termes dans la définition des concepts. Par contre, l'inconvénient majeur de l'algorithme $OLA_2$ était le temps de calcul et l'espace mémoire nécessaire par rapport au techniques terminologiques utilisées qui n'ont pas nécessité le recours à un ordinateur d'expérimentation. En effet, cet inconvénient, selon Djoufak et al. [9], est dû aux produits matriciels qui coûtent cher en temps et en mémoire surtout quand il s'agit d'une ontologie de grande taille.

## 6.8 Conclusion

Dans ce chapitre, nous avons présenté les types d'hétéreogéneité rencontrés (syntaxique, terminologique, et conceptuelle) lors d'échange de documents et du partage d'information dans le domaine d'affaire. Ces problèmes sont dus à l'existence de plusieurs modèles de documents élaborés indépendamment, sous forme de standards d'affaire implémentés dans plusieurs langages comme XML et DTD, par différents organismes pour répondre aux besoins spécifiques d'entreprises qui collaborent.

Lorsque les ontologies ne partagent pas le même vocabulaire pour la définition des concepts, l'hétérogéneité persiste même s'ils ont le même domaine d'intérêt. Nous avons présenté l'alignement d'ontologies comme une piste prometteuse pour la réduire. Nous avons présenté les travaux précédents portant sur la notion d'alignement dans le domaine d'affaire. La plupart d'entre eux sont restés au niveau abstrait du processus d'affaire et ne traitent pas des documents qui représentent pourtant la source principale de l'hétérogéneité due à la sémantique des termes utilisés.

Ensuite, nous avons décrit des techniques d'alignement, des mesures de similarité ainsi que le fonctionnement de quelques outils d'alignement. Nous avons expérimenté

différentes méthodes d'alignement terminologiques (`Levenshtein` et `TFIDF`) et une autre à base de structure ($OLA_2$), pour déterminer les correspondances entre les ontologies des standards cXML et xCBL avec celle de RosettaNet. Le choix de l'algorithme $OLA_2$ se base, d'une part sur ces performances dans les compétitions OAEI, en plus d'être disponible en libre accès.

Avant l'alignement, nous avions soulevé deux questions : ces ontologies sont-elles similaires ? À quel point les résultats d'alignement aident-ils à réaliser l'objectif de l'ingénierie de documents dans le domaine d'affaire ? Les résultats que nous avons obtenus avec la technique `TFIDF` montrent que ces ontologies sont similaires et qu'elles partagent les termes les plus fréquents et les plus discriminants. Cette constatation paraît logique puisque ces ontologies traitent le même domaine d'intérêt. À la section 6.7.3, nous avons montré que la performance de l'algorithme $OLA_2$ est meilleure par rapport aux techniques terminologiques utilisées, selon la mesure F-score.

# CHAPITRE 7

# CONCLUSION

Dans notre thèse, nous nous sommes intéressés à la gestion des documents d'affaires dans le cadre du web sémantique. Nous avons identifié les problématiques liées à l'échange des documents dans le domaine des affaires. La structuration, l'interopérabilité, et le traitement de la sémantique de données dans les processus d'affaires étaient les défis majeurs dans les processus d'échange. Nous avons proposé comme solution l'utilisation de l'ingénierie de documents pour la spécification, la conception et la mise en œuvre des documents échangés électroniquement. Au chapitre 2, nous avons défini cette discipline ainsi que des étapes de réalisation.

L'apparition de nouvelles technologies de modélisation de processus d'affaires ainsi que des langages comme XML et le web sémantique ont aidé à réduire les limites des EDI traditionnels. L'apparition de XML était une solution partielle. Son apparition a poussé des grandes entreprises à créer des standards d'échange basés sur XML. Au chapitre 4, nous avons vu les principaux standards basés sur XML comme xCBL, ebXML et RosettaNet. Nous avons décrit le principe de fonctionnement et les composantes de chaque standard. Malgré leur impact sur la réalisation des affaires, ces standards souffrent de certaines limites surtout au niveau du traitement de la sémantique des échanges, de la possibilité d'effectuer des raisonnements sur les données et de l'hétérogénéité des échanges.

Nous avons proposé une feuille de transformation XSLT qui transforme les documents des standards d'affaire définis en DTD et XML Schema vers une représentation ontologique en tenant compte des efforts mis dans la modélisation XML.

Puisque les normes d'affaire ont été établies indépendamment avec leur propre vocabulaire, les ontologies générées par le processus de transformation ne sont pas nécessairement compatibles ce qui complique l'interopérabilité et la communication entre les partenaires à cause de différents types d'hétérogénéité.

Pour remédier à ce problème, nous nous sommes basés également sur les recherches

effectuées dans le domaine de l'alignement d'ontologies pour déterminer les liens sémantiques entre des ontologies hétérogènes générés par notre processus de transformation et pour tester leur apport à aider les entreprises à communiquer de manière fructueuse. Nous nous sommes concentrés sur les normes comme cXML, RosettaNet et xCBL qui partagent le même domaine d'intérêt lié à la gestion des bons de commande.

Nous avons appliqué différentes méthodes d'alignement (`Levenshtein`, `TFIDF` et l'algorithme `OLA2`) basées sur la structure, les noms et les commentaires des entités. Les résultats d'alignement de cXML et xCBL avec RosettaNet PIP3A4 sont prometteurs et l'algorithme $OLA_2$ a donné la meilleure performance avec la plus grande valeur de F-score par rapport aux méthodes terminologiques.

L'utilisation de `OLA2` permet d'améliorer significativement la qualité des résultats d'alignement par rapport aux méthodes terminologiques. Ces résultats prometteurs nous permettent de conclure à l'utilité des approches d'alignement d'ontologies pour la réduction du degré d'hétérogénéité entre les applications d'affaires ainsi que la possibilité de mise à l'échelle.

Nos travaux évoquent également un ensemble de questions non résolues ouvrant la porte à des travaux futurs.

Tel que mentionné, le processus de transformation souffre de certaines limites surtout au niveau de la représentation de certaines relations sémantiques en OWL. Nous avons essayé d'améliorer la représentation de la hiérarchie des classes dans une ontologie par l'application de l'analyse formelle de concepts pour introduire la relation `subClassOf`. Mais il reste beaucoup de travail à faire surtout au niveau des classes disjointes et des propriétés inverses.

Dans notre cas, nous ne disposons pas d'instances, or nous croyons que la disponibilité d'instances et de leurs relations permettaient un enrichissement sémantique entre les concepts. Nous pourrions, d'une part, déduire certaines relations à partir des valeurs des éléments. Pour déduire des classes disjointes, nous pourrions comparer les instances de chacune des classes d'un concept deux à deux et vérifier la présence d'instances partagées entre deux classes, en leur absence on pourrait supposer que ces classes sont disjointes.

Les instances pourraient jouer un rôle important dans le processus d'alignement d'ontologies. Ce processus compare les extensions des concepts, c'est-à-dire l'ensemble des instances reflètant ce que ces concepts désignent dans la pratique, au lieu de leur interprétation. Lorsque les deux ontologies à aligner partagent le même ensemble d'individus, l'alignement en est facilité [14]. Pour calculer les similarités entre les concepts des ontologies à aligner, nous pouvons utiliser la métrique de Jaccard souvent utilisée pour déterminer la similarité entre échantillons en statistiques. Étant donné deux classes de deux ontologies à aligner, cette valeur tend vers 1 quand les classes partagent les mêmes instances et 0 sinon. Nous pouvons aussi appliquer l'analyse formelle de concepts pour aligner les classes d'ontologies ayant un nombre d'instances partagées. Les classes des ontologies à aligner seront organisées dans un treillis de concepts où chaque concept est identifié par ses propriétés (les intentions) représentées par les classes ; leurs valeurs (les extensions) étant les instances associées à ces classes.

Les résultats d'alignement pourraient également être améliorés avec un thésaurus décrivant les termes du domaine d'affaire. Il serait intéressant de concevoir un thésaurus, une base de données lexicales, pour répertorier, classifier et relier les différents termes techniques liés au domaine d'affaires, ce qui manque à date. Cette structure hiérarchique, du général au plus spécifique, montre les relations entre les différents sujets, où chaque entrée représente un ensemble de termes avec une sémantique commune. Pour créer un tel thésaurus, il faudrait constituer un vocabulaire en analysant par exemple les documents des standards d'affaire pour en extraire les concepts de base ainsi que les relations d'équivalence linguistique (p.ex. la synonymie), d'hyponymie ou d'hyperonymie .

Pour la construction de la taxonomie de notre thésaurus, nous pouvons nous inspirer de l'organisation des PIPs de RosettaNet, qui représentent la plupart des tâches de la chaîne d'approvisionnement, afin d'établir une liste des grands thèmes.

La terminologie du domaine des affaires est complexe et ambigüe, un mot pouvant désigner deux concepts différents, ce qui complique la tâche du construction de vocabulaire qui consiste à détecter les termes liés sémantiquement. Des techniques, p.ex. la distance de Levenshtein, donnent une mesure de similarité entre deux termes représentés par des chaînes de caractères, mais elle n'est efficace que lorsqu'on utilise des chaînes

semblables pour désigner les mêmes concepts. Dans ce cas, il faut utiliser d'autres informations comme les relations entre les concepts d'un document ainsi que leur contexte d'utilisation. Nous pouvons utiliser les résultats préliminaires obtenus dans l'alignement des ontologies d'affaire pour construire des listes des termes, les noms de classes alignées qui représentent un même concept. Dans un processus d'alignement d'ontologies, un concept aligné à un ou plusieurs concepts indique une relation sémantique entre eux, alors leurs noms représentent une entrée dans notre thésaurus. Leur définition offerte dans la documentation de ces standards pourrait être utilisée dans la description de chaque entrée du lexique.

Cette thèse s'est intéressée aux échanges dans l'e-business. Nous avons montré l'apport de l'alignement d'ontologies dans les échanges entre les entreprises ; par contre, nous nous sommes limités aux documents d'affaires liés à la gestion de bons de commande. Il serait intéressant d'étudier d'autres problèmes dans l'e-commerce pour étendre notre travail à d'autres ontologies destinées à la description des produits, p. ex. GoodRelations[1]. Il serait alors possible d'appliquer l'alignement d'ontologies pour aider les clients à trouver des produits similaires selon leurs critères de recherche même si les fournisseurs utilisent des ontologies différentes pour la description de leurs produits. Cette liste de produits similaires pourrait être établie en fonction du degré de similitude entre les ontologies décrivant les catalogues de ces produits.

Notre travail a montré l'intérêt de la migration des systèmes d'entreprises d'une représentation syntaxique vers un modèle sémantique pour faciliter l'accès et l'interprétation de l'information. L'alignement d'ontologies permet de réduire l'hétérogénéité pour améliorer la communication entre les applications d'affaires. Cette recherche ouvre la voie à d'autres approches d'intégration de la sémantique dans les standards d'affaire pour améliorer l'interopérabilité et l'échange de documents.

[1] http ://www.heppnetz.de/projects/goodrelations/

## BIBLIOGRAPHIE


[1] xCBL User's Guide Version 4.0, 2003.

[2] Ariba. cXML User's Guide Version 1.2.025, April 2014.

[3] Jorge Cardoso et Christoph Bussler. Mapping between heterogeneous XML and OWL transaction representations in B2B integration. *Data & Knowledge Engineering*, 70(12):1046–1069, 2011.

[4] Philipp Cimiano, Andreas Hotho, Gerd Stumme et Julien Tane. Conceptual knowledge processing with formal concept analysis and ontologies. Dans *Concept Lattices*, pages 189–207. Springer, 2004.

[5] OASIS/ebXML Registry Technical Committee et al. OASIS/ebXML Registry Services Specification v2.0 (2002). URL `http://www.oasis-open.org/committees/regrep/documents/2.0/specs/ebrs.pdf`. consulté le 20 Février 2012.

[6] Suresh Damodaran. B2B integration over the internet with XML : RosettaNet successes and challenges. Dans *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 188–195. ACM, 2004.

[7] Roberto Pérez López de Castro, Inty Sáez Mosquera et Jorge Marx Gómez. Towards semantic support for business process integration. Dans *Engineering and Management of IT-based Service Systems*, pages 83–99. Springer, 2014.

[8] Jean-François Djoufak-Kengue, Jérôme Euzenat, Petko Valtchev et al. OLA in the OAEI 2007 evaluation contest. Dans *Proc. 2nd ISWC 2007 workshop on ontology matching (OM)*, pages 188–195, 2007.

[9] Jean-François Djoufak-Kengue, Jérôme Euzenat, Petko Valtchev et al. Alignement d'ontologies dirigé par la structure. Dans *Actes 14e journées nationales sur langages et modèles à objets (LMO)*, pages 43–57, 2008.

[10] Asuman Dogac. RosettaNet. Grenoble Ecole de Management MEDFORIST Workshop, 2012.

[11] Asuman Dogac, Yildiray Kabak et Gokce B Laleci. Enriching ebXML registries with OWL ontologies for efficient service discovery. Dans *Research Issues on Data Engineering : Web Services for e-Commerce and e-Government Applications, 2004. Proceedings. 14th International Workshop on*, pages 69–76. IEEE, 2004.

[12] Asuman Dogac, Yildiray Kabak, Gokce B Laleci, Carl Mattocks, Farrukh Najmi et Jeff Pollock. Enhancing ebXML registries to make them OWL aware. *Distributed and Parallel Databases*, 18(1):9–36, 2005.

[13] Jérôme Euzenat, David Loup, Mohamed Touzani, Petko Valtchev et al. Ontology alignment with OLA. Dans *Proc. 3rd ISWC2004 workshop on Evaluation of Ontology-based tools (EON)*, pages 59–68, 2004.

[14] Jérôme Euzenat et Pavel Shvaiko. *Ontology Matching*, volume 18. Springer Heidelberg, 2007.

[15] Sally Fuger, Farrukh Najmi et Nicola Stojanovic. ebXML Registry Information Model version 3.0 (2005). URL `http://docs.oasis-open.org/regrep/v3.0/specs/regrep-rim-3.0-os.pdf`. consulté le 20 Février 2012.

[16] Bernhard Ganter et Rudolf Wille. Formal concept analysis : mathematical foundations. *Springer*, 1999.

[17] Roberto García et Rosa Gil. Facilitating business interoperability from the semantic web. Dans *Business Information Systems*, pages 220–232. Springer, 2007.

[18] Robert J. Glushko et Tim McGrath. *Document Engineering : Analyzing and Designing Documents for Business Informatics and Web Services*. MIT Press, 2005.

[19] Armin Haller, Jedrzej Gontarczyk et Paavo Kotinurmi. Towards a complete SCM ontology : the case of ontologising RosettaNet. Dans *Proceedings of the 2008 ACM symposium on Applied computing*, pages 1467–1473. ACM, 2008.

[20] Martin Hepp et Dumitru Roman. An ontology framework for semantic business process management. *Proceedings of Wirtschaftsinformatik*, March 2 2007.

[21] Pascal Hitzler, Markus Krötzsch et Sebastian Rudolph. *Foundations of semantic web technologies*. Chapman & Hall/CRC, 2009.

[22] John E Hopcroft et Richard M Karp. An nˆ5/2 algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing*, 2(4):225–231, 1973.

[23] Wei Hu, Yuzhong Qu et Gong Cheng. Matching large ontologies : A divide-and-conquer approach. *Data & Knowledge Engineering*, 67(1):140–160, 2008.

[24] Marianne Huchard. Analyse formelle de concepts : Une approche pour fouiller des ensembles de données multi-relationnels, et quelques applications au génie logiciel. 2012.

[25] Christian Huemer. ebXML : An Emerging B2B Framework. 2002.

[26] Yves R Jean-Mary, E Patrick Shironoshita et Mansur R Kabuka. Ontology matching with semantic verification. *Web Semantics : Science, Services and Agents on the World Wide Web*, 7(3):235–251, 2009.

[27] Jamel Eddine Jridi et Guy Lapalme. Adapting RosettaNet B2B standard to Semantic Web Technologies. Dans *International Conference on Entreprise Information Systems (ICEIS)*, pages 443–450, 2013.

[28] Jamel Eddine Jridi et Guy Lapalme. FCA-Based Concept Detection in a RosettaNet PIP Ontology. Dans *FCA4AI International Workshop What can FCA do for Artificial Intelligence ? (IJCAI)*, page 25, 2013.

[29] Mehdi Kaytoue, Amedeo Napoli et al. Classification de données numériques par treillis de concepts et structures de patrons. Dans *Journées Nationales de l'Intelligence Artificielle Fondamentale*, 2009.

[30] Gunwoo Kim et Yongmoo Suhh. Ontology-based semantic matching for business process management. *ACM SIGMIS Database*, 41(4):98–118, 2010.

[31] Paavo Kotinurmi, Armin Haller et Eyal Oren. Ontologically Enhanced RosettaNet B2B Integration. *Semantic Web for Business : Cases and Applications*, 2008.

[32] Miltiadis D. Lytras et Roberto García. Semantic Web applications : a framework for industry and business exploitation–what is needed for the adoption of the semantic web from the market and industry. *International Journal of Knowledge and Learning*, 4(1):93–108, 2008.

[33] Marek Obitko, Vaclav Snasel, Jan Smid et V Snasel. Ontology design with formal concept analysis. *Concept Lattices and their Applications, Ostrava : Czech Republic*, pages 111–119, 2004.

[34] Luca Panziera, Matteo Palmonari, Marco Comerio et Flavio De Paoli. WSML or OWL ? a lesson learned by addressing NFP-based selection of semantic Web services. Dans *Non Functional Properties and SLA Management (NFPSLAM 2010) workshop.*, 2010.

[35] Elie Raad, Joerg Evermann et al. Is Ontology Alignment like analogy ?–Knowledge Integration with LISA. Dans *29th Symposium On Applied Computing (SAC)*, pages 294–301, 2014.

[36] W3C Recommendation. RDF/XML Syntax Specification (Revised), 2004. URL `http://www.w3.org/TR/REC-rdf-syntax/`.

[37] W3C Recommendation. OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition), 2012. URL `http://www.w3.org/TR/owl2-syntax/`.

[38] W3C Recommendation. OWL 2 Web Ontology Language XML Serialization (Second Edition), 2012. URL http://www.w3.org/TR/owl2-xml-serialization/.

[39] Toni Rodrigues, Pedro Rosa et Jorge Cardoso. Moving from syntactic to semantic organizations using JXML2OWL. *Computers in Industry*, 59(8):808–819, 2008.

[40] Dumitru Roman, Holger Lausen et Uwe Keller. Web Service Modeling Ontology standard. *WSMO Working Draft v1.0*, 2005.

[41] RosettaNet Program Office. *Overview Clusters, Segments, and PIPs*, December 2011.

[42] Gerard Salton et Michael J McGill. Introduction to modern information retrieval. 1983.

[43] Petra Schubert et Christine Legner. B2B integration in global supply chains : An identification of technical integration scenarios. *The Journal of Strategic Information Systems*, 20(3):250–267, 2011.

[44] Stefan Schulte, Melanie Siebenhaar et Ralf Steinmetz. Integrating semantic web services and matchmaking into ebXML registry. Dans *Proceedings of the Fourth International Workshop SMR2 2010 on Service Matchmaking and Resource Retrieval in the Semantic Web*, pages 69–83, 2010.

[45] Frederick T Sheldon, Kshamta Jerath et Hong Chung. Metrics for maintainability of class inheritance hierarchies. *Journal of Software Maintenance and Evolution : Research and Practice*, 14(3):147–160, 2002.

[46] Pavel Shvaiko et Jérôme Euzenat. Ontology matching : state of the art and future challenges. *Knowledge and Data Engineering, IEEE Transactions on*, 25(1):158–176, 2013.

[47] Miguel-Ángel Sicilia, Daniel Rodríguez, Elena García Barriocanal et Salvador Sánchez Alonso. Empirical findings on ontology metrics. *Expert Systems with Applications*, 39(8):6706–6711, 2012.

[48] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur et Yarden Katz. Pellet : A practical OWL-DL reasoner. *Web Semantics : science, services and agents on the World Wide Web*, 5(2):51–53, 2007.

[49] Gerd Stumme. Using ontologies and formal concept analysis for organizing business knowledge. Dans *In Proc. Referenzmodellierung 2001*. Citeseer, 2001.

[50] Samir Tartir et Budak Arpinar. Ontology evaluation and ranking using OntoQA. Dans *Semantic Computing, 2007. ICSC 2007. International Conference on*, pages 185–192. IEEE, 2007.

[51] Daniel Teixeira. Converting cXML business transactions to a semantic data model. Rapport technique, Department de Matemática e Engenharias, Universidade da Madeira, Funchal, Portugal, 2007.

[52] Chrisa Tsinaraki et Stavros Christodoulakis. XS2OWL : a formal model and a system for enabling XML schema applications to interoperate with OWL-DL domain knowledge and semantic web tools. *Digital Libraries : Research and Development*, pages 124–136, 2007.

[53] A W3C Member Submission. OWL-S : Semantic Markup for Web Services. http ://www.w3.org/submission/OWL-S/, 2004.

[54] W3C Working Group Note. OWL 2 Web Ontology Language XML Manchester Syntax (Second Edition), 2012. URL `http://www.w3.org/TR/owl2-manchester-syntax/`.

[55] Anna V Zhdanova. Towards a community-driven ontology matching. Dans *Proceedings of the 3rd international conference on Knowledge capture*, pages 221–222. ACM, 2005.

[56] Jian Zhu et Hung Keng Pung. Process matching : A structural approach for business process search. Dans *Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns, 2009. COMPUTATIONWORLD'09. Computation World :*, pages 227–232. IEEE, 2009.

# Annexe I

## Définition des concepts de base

**Ingénierie de documents :** discipline pour la spécification, la conception et la mise en oeuvre des documents échangés électroniquement [19]. Elle facilite la communication et gérer la grande masse d'informations qui circulent dans/et en dehors d'un organisme. Elle est utilisée dans plusieurs secteurs dont celui de la gestion des affaires ou du domaine médical, etc. (*Section 2.2*)

**XSLT :** (Extensible Stylesheet Language Transformations) langage de transformation d'un document XML vers un autre document XML ou un autre format comme HTML. (*Section 3.2*)

**XML Schema :** langage de définition de documents XML. Il permet de définir que la structure et le type de contenu d'un document XML. Cette définition permet notamment de vérifier que la validité syntaxique de ce document. Le but d'un schéma est de définir une classe de documents XML. Il décrit les informations syntaxiques comme l'ordre d'imbrication et d'apparition des éléments et de leurs attributs dans un document XML. (*Section 3.2.1*)

**Ontologie :** un ensemble structuré des termes et concepts utilisés pour décrire et représenter un domaine de connaissance. Elle permet aux utilisateurs d'organiser l'information dans une hiérarchie de concepts, de décrire les relations entre eux, de rendre le contenu accessible et compréhensible par la machine, et de déduire du contenu implicite. (*Section 3.3*)

**OWL :** (Ontology Web Language) langage de description d'ontologies conçu pour la publication et le partage d'ontologies sur le Web sémantique. (*Section 3.3.3*)

**Business to business (B2B) :** ensemble des activités dans la réalisation des affaires entre les entreprises. (*Section 4.1*)

**Échange de données :** équivalent d'un échange des documents papier, p.e. un bon de commande ou une facture. Les normes EDI sont des normes qui régissent la façon dont ces documents sont structurés et définissent leurs règles d'utilisation. Ce type d'échange

représente un échange automatique d'informations, contenues dans les documents, entre deux systèmes d'entreprises. (*Section 4.2*)

**E-business :** utilisation de technologies de l'information et de la communication dans la réalisation des affaires entre les entreprises. (*Section 4.2*)

**E-commerce :** cas particulier de l'e-business qui utilise les supports électroniques pour tout ou une partie des activités commerciales entre une entreprise et les particuliers (la publicité, les commandes en ligne, le paiement électronique, la livraison). (*Section 4.2*)

**Processus d'affaire (BP) :** scénario de réalisation d'une activité d'affaire entre les entreprises ainsi que la séquence des messages qui aide à gérer ce processus. Il décrit la relation entre des partenaires commerciaux, leurs rôles et les transactions d'affaire effectuées avec d'autres partenaires. (*Section 4.2.4*)

**Analyse formelle des concepts :** méthode d'analyse de données basée sur la théorie des treillis et spécialisée dans l'extraction d'un ensemble ordonné de concepts au sein d'un ensemble de données, appelé un contexte formel, qui est composé d'objets décrits par des attributs [24]. (*Section 5.5.1*)

**Alignement d'ontologies :** processus qui prend en entrée deux ontologies et trouve des liens sémantiques ou un ensemble de correspondances entre les entités de ces ontologies (des classes, des propriétés, des instances, etc.). Il permet d'améliorer l'interopérabilité entre des applications distantes et de surmonter dans une certaine mesure le problème de l'hétérogénéité sémantique [46]. (*Section 6.4*)