N° d'ordre

Résumé

Afin d'améliorer la pertinence des décisions prises, les organisations ont tendance à fonctionner selon les règles métier, sur lesquelles sont basées la majorité des exigences logiciel. Étant donné que les experts métier ne sont généralement pas familiarisés avec les langages (semi) formels, le langage naturel reste le format le plus approprié pour représenter ces règles métier.

Dans cette thèse, nous présentons des approches automatisées pour extraire à la fois le dictionnaire terminologique ainsi que la sémantique des règles métier à partir des spécifications écrites en langage naturel, en utilisant le traitement du langage naturel (Natural Language Processing). Nos contributions se distinguent des travaux existants en ce qu'elles adoptent une méthode qui traitent les règles textuelles d'une manière approfondit pour identifier automatiquement les informations implicites cachées dans les structures de phrases complexe. Par conséquence, on a pu extraire non seulement une liste de termes-relations, mais aussi les faits implicites, ainsi que des spécifications supplémentaires telles que les définitions et les synonymes pour donner une signification complète à chaque concept identifié. Le fameux standard SBVR a été adopté comme format final pour encourager en plus de l'interopérabilité, l'intégration des non experts IT dans la validation des spécifications.

Les performances de nos contributions ont été évaluées à travers leurs applications sur une dizaines de règles métier textuelles. Avec des meilleures métriques, nos contributions ont prouvé leur capacité à générer automatiquement le vocabulaire métier ainsi que la sémantique des règles métier en plus des spécifications supplémentaires.

Mots-clés : Règles métier, Vocabulaire métier, Exigences, MDA, SBVR, NLP

Abdellatif HAJ

La Spécification Formelle de la Sémantique des Règles Métier Textuelles
Une Approche par le Traitement du Langage Naturel

2021, MIA

Université Hassan 1er
Centre d'Études Doctorales

Faculté des Sciences et Techniques
Settat

THÈSE DE DOCTORAT

Pour l'obtention de grade de Docteur en ***Mathématiques, Informatique et Applications***

Formation Doctorale : **Mathématiques, Informatique et Applications**

Spécialité : **Informatique**

Sous le thème

La Spécification Formelle de la Sémantique des Règles Métier Textuelles

Une Approche par le Traitement du Langage Naturel

Présentée par :

Abdellatif HAJ

Soutenue le : *le 07 octobre 2021.*

A la Faculté des Sciences et Techniques de Settat devant le jury composé de :

Pr. El Hassan ESSOUFI	**PES**	**FST, Settat**	Président
Pr. Mahmoud NASSAR	**PES**	**ENSIAS, Rabat**	Rapporteur
Pr. Jalal LAASSIRI	**PES**	**FS, Kénitra**	Rapporteur
Pr. Abderrahim MARZOUK	**PES**	**FST, Settat**	Rapporteur
Pr. Sara AREZKI	**PH**	**FST, Settat**	Examinatrice
Pr. Youssef BALOUKI	**PH**	**FST, Settat**	Co-Directeur de thèse
Pr. Taoufiq GADI	**PES**	**FST, Settat**	Directeur de thèse

Année Universitaire : 2020/2021

Cette page a été intentionnellement laissée vierge

“You have to learn the rules of the game, and then you have to play better than anyone else”
- Albert Einstein –

Résumé

Afin d'améliorer la pertinence des décisions prises, les organisations ont tendance à fonctionner selon les règles métier, sur lesquelles sont basées la majorité des exigences logiciel. Étant donné que les experts métier ne sont généralement pas familiarisés avec les langages (semi) formels, le langage naturel reste le format le plus approprié pour représenter ces règles métier.

Comme mentionné dans le manifeste des règles métier proposé par le Groupe des règles métier BRG (Business Rules Group) : « les règles sont construites en se basant sur des faits, qui à leur tour sont construits en se basant sur des termes », par conséquence, il serait judicieux de standardiser la terminologie utilisée dans le domaine métier, ce qui va éviter non seulement les malentendus entre les différentes parties prenantes, mais cela va augmenter aussi la précision du traitement automatique du langage naturel avec lequel sont écrites les spécifications métier. La création manuelle du domaine métier demande du temps, et sera certainement source d'erreurs, ce qui va couter chère aux entreprises surtout lorsqu'on a un grand nombre de spécifications textuelles en constante changement.

D'autre part, dans les premières phases du processus de développement logiciel, les spécifications sont principalement écrites dans un langage naturel plutôt que présentées sous formes de modèles formels, ce qui n'est pas pris en charge par les méthodes de développement des logiciels comme l'architecture dirigée par les modèles MDA (Model Driven Architecture). Pour cette raison, le OMG (Object Management Group) a proposé en 2008 le standard SBVR (Semantic of Business Vocabulary and Rules) qui permet d'exprimer à la fois le vocabulaire métier ainsi que les spécifications textuelles dans un langage compréhensible à la fois par les différentes parties prenantes ainsi que les machines, afin de faciliter son intégration dans le cycle de vie MDA.

Dans cette thèse, nous présentons des approches automatisées pour extraire à la fois le dictionnaire terminologique ainsi que la sémantique des règles métier à partir des spécifications écrites en langage naturel, en utilisant le traitement du langage naturel (Natural Language Processing). Nos contributions se distinguent des travaux existants en ce qu'elles adoptent une méthode qui traitent les règles textuelles d'une manière approfondit pour identifier automatiquement les informations implicites cachées dans les structures de phrases complexe. Par conséquence, on a pu extraire non seulement une liste de termes-relations, mais aussi les faits implicites, ainsi que des spécifications supplémentaires telles que les définitions et les synonymes pour donner une signification complète à chaque concept identifier. Le fameux standard SBVR a été adopté comme format finale pour encourager en plus de l'interopérabilité, l'intégration des non experts IT dans la validation des spécifications.

Les performances de nos contributions ont été évaluées à travers leurs applications sur des dizaines de règles métier textuelles. Avec des meilleures métriques, nos contributions ont prouvé leur capacité à générer automatiquement le vocabulaire métier ainsi que la sémantique des règles métier en plus des spécifications supplémentaires.

Mots clés : Règles métier ; Vocabulaire métier ; Exigences ; MDA ; SBVR ; NLP

Abstract

To improve the relevance of decisions made within organizations, the latter tend to operate according to business rules, upon which most software requirements are based. Since business experts are generally not familiar with (semi) formal languages, natural language remains the most suitable format to represent these business rules.

As mentioned in the business rules manifesto proposed by the Business Rules Group (BRG): “Rules are built on facts, which in turn are built on terms”, therefore, it would be wise to pay great attention to standardize the terminology used in the business domain, which will not only avoid misunderstandings between different stakeholders, but it will certainly increase the precision of the natural language processing with which are written the business specifications. Manual creation of the business domain is time consuming and error prone, especially when you have a large number of constantly changing textual specifications.

In the other hand, in the early stages of the software development process, specifications are mostly written in natural language rather than presented in formal models, which is not supported by software development methods such as Model Driven Architecture (MDA). For this reason, the OMG (Object Management Group) has proposed in 2008 the SBVR standard (Semantic of Business Vocabulary and Rules) which makes it possible to express both the business vocabulary as well as the textual specifications in a language understandable both by stakeholders as well as machines, in order to facilitate its integration into the MDA life cycle.

In this thesis, we present automated approaches to extract both the terminology dictionary as well as the semantics of business rules from specifications written in natural language, using Natural Language Processing (NLP). Our contributions differ from existing work in that they adopt a method that processes textual statements in a deep way to automatically identify implicit information hidden in complex sentence structures. As a result, we could extract not only a flat list of terms and relationships, but also implicit facts, as well as additional specifications such as definitions and synonyms to give full meaning to each identified concept. The famous SBVR standard was adopted as the final format to encourage, in addition to interoperability, the integration of non-IT people in the validation of specifications.

The performance of our contributions has been evaluated through their applications on dozens of textual business rules. With better metrics, our contributions have proven their ability to automatically generate the business vocabulary as well as the semantics of business rules beside additional specifications.

Keywords: Business Vocabulary ; Business Rules ; Requirements ; MDA ; SBVR ; NLP

ملخص

من أجل تحسين جودة القرارات المتخذة داخل المؤسسات، تميل هذه الاخيرة إلى العمل وفقًا لقواعد العمل، والتي تستند إليها غالبية متطلبات البرامج. نظرًا لكون خبراء الأعمال ليسوا على دراية بلغات الرياضيات وما شابهها، تظل اللغة الطبيعية هي الشكل الأنسب للتعبير عن قواعد العمل. حسب ما هو مذكور في بيان قواعد العمل الذي اقترحته مجموعة "قواعد العمل" (Business Rules Group) : القواعد مبنية على الحقائق، والتي بدورها مبنية على المفردات"، لذلك، سيكون من المفيد الاهتمام بتوحيد المصطلحات المستخدمة في مجال الأعمال، والذي لن يؤدي فقط إلى تجنب سوء الفهم بين مختلف الأطراف المعنية ، ولكنه سيساهم بالتأكيد في تحسين دقة المعالجة الالية للغة الطبيعية التي بها تتم كتابة مواصفات العمل. التعبير عن مجال الاعمال يدويا يستغرق وقتًا كبيرا وسيكون بلا شك عرضة للأخطاء، خاصة عندما يكون لديك عدد كبير من الجمل التي تعبر عن قواعد العمل. من ناحية أخرى، في المراحل الأولى من عملية تطوير البرمجيات، تتم كتابة المواصفات في الغالب بلغة طبيعية بدلاً من تقديمها كنماذج رياضياتية، والتي لا تدعمها طرق تطوير البرامج مثل البنية المبنية على النماذج(MDA) ، لهذا السبب، اقترحت OMG عام 2008 معيار SBVR(دلالة المفردات وقواعد الأعمال) والذي يجعل من الممكن التعبير عن كل من مفردات الأعمال وكذلك المواصفات النصية بلغة مفهومة من قبل كل المتدخلين وكذلك الآلات، من أجل تسهيل اندماجها في دورة حياة MDA. في هذه الأطروحة، نقدم مناهج آلية لاستخراج كل من قاموس المصطلحات بالإضافة إلى دلالات قواعد العمل من المواصفات المكتوبة باللغة الطبيعية، باستخدام معالجة اللغة الطبيعية. تختلف مساهماتنا عن بقية الاعمال الاخرى من حيث انها تتبنى طريقة تعالج العبارات النصية بطريقة عميقة لتحديد المعلومات الضمنية المخبأة في الجمل المعقدة. ونتيجة لذلك، لم نتمكن من استخراج قائمة كاملة من المصطلحات وكذا العلاقات بينها فحسب، بل أيضًا الحقائق الضمنية، بالإضافة إلى المواصفات الإضافية مثل تعاريف المصطلحات والمرادفات لإعطاء معنى شامل لكل مصطلح تم استخراجه. تم اعتماد معيار SBVR الشهير كصيغة نهائية لتشجيع ادماج غير المتخصصين في تكنولوجيا المعلوميات في التحقق من المواصفات.

تم تقييم أداء مناهجنا من خلال تطبيقها على العشرات من جمل قواعد العمل. من خلال النتائج الكبيرة، أثبتت مناهجنا قدرتها على استخراج مفردات الأعمال تلقائيًا بالإضافة إلى دلالات قواعد العمل والمواصفات الإضافية لكل مصطلح.

كلمات مفتاح: متطلبات البرامج، معجم الاعمال، قواعد العمل , تحليل اللغة الطبيعية ، دلالات مفردات و قواعد العمل , البنية المبنية على النماذج

Dédicace

À mes chers parents

À mon beau-père et ma belle-mère

À ma chère épouse et mes enfants

À mes sœurs, frère et beaux frères

À mes professeurs

À tous mes amis

À tous ceux et celles qui me sont chers

Je dédie ce modeste travail.

Abdellatif

Remerciements

Je tiens en tout premier lieu à remercier les nombreuses personnes qui m'ont aidées à mener à bien cette thèse, sur le plan professionnel comme sur le plan personnel.

J'adresse mes premiers remerciements à Monsieur Taoufiq GADI et Monsieur Youssef BALOUKI, qui furent des excellents encadrants de thèse. Merci de m'avoir écouté, conseillé, supporté durant ces années. C'est un plaisir de travailler avec vous et j'ai eu la chance de beaucoup apprendre à vos côtés.

Je remercie ensuite les membres de mon jury pour avoir dirigé ma thèse, et pour avoir accepté de rapporter mes travaux. J'adresse mes remerciements les plus sincères à Monsieur le président pour avoir accepté de présider mon jury. Mes remerciements vont aussi aux personnes, ayant accepté d'être examinatrice de cette thèse.

Je voudrais aussi remercier les personnes avec qui j'ai travaillé autour de mon sujet de thèse, de mes débuts en Master à Settat jusqu'à ma soutenance : Zakaria Bousalem et Issam Errahmani,

Je tiens à remercier aussi les membres du Laboratoire de Mathématiques, Informatique et Sciences de l'Ingénieur (MISI) de m'avoir accueilli, et pour tous les conseils qu'ils ont pu me donner.

D'un point de vue plus personnel, je remercie ma famille, notamment mes parents, mes beaux-parents et ma femme Dr. Lamia Aznague pour le soutien constant qu'ils m'ont apporté durant toutes mes études – et même avant. Sans eux, tout cela n'aurait pas été possible.

En dernier lieu, je remercie en vrac toutes ces personnes auprès desquelles j'ai été moins présent durant ces années de thèse, notamment Mohammed, Abdelmounaim et Abdellah et toutes mes excuses pour ceux que j'ai oublié !

Table des Matières

Liste des Tableaux

Liste des Figures

Acronymes

Abréviation	Désignation
BPMN	Business Process Model and Notation
BR	Business Rules
BRG	Business Rules Group
CIM	Computation Independent Model
DSI	Domain Specific Information
IDM	L'ingénierie Dirigée par les Modèles
MDA	Model Driven Architecture
MISI	Laboratoire de Mathématiques Informatique et Sciences de l'Ingénieur
NER	Named Entity Recognition
NLP	Natural Language Processing
OCL	Object Constraint Language
OMG	Object Management Group
PDM	Plateform Dependant Model
PIM	Platform Independent Model
POS	Part of Speech
PSM	Platform Specific Model
SRS	Software Requirement System
UCM	Use Case Model
UI	Interface Utilisateur
UML	Unified Modeling Language
XMI	XML Metadata Interchange

Cette page a été intentionnellement laissée vierge

Introduction Générale

Sommaire :

▪ Contexte général

Après la mise en place d'une solution logiciel finale, les experts métier cherchent à être indépendant le maximum possible envers les développeurs, qui - eux même - pourraient éventuellement être frustrés de recevoir des demandes de changement 'critiques' ou d'autres exigences après avoir implémenter le système. Il est également pénible pour un développeur que son projet soit interrompu par des demandes de modifications d'un système qui fonctionne exactement comme convenu à l'avance par les différentes parties prenantes. Ce qui est tout à fait conforme avec le fait que les lacunes dans la spécification et la gestion des exigences sont les principales sources d'échec des solutions informatiques.

Le Groupe Standish[1] dans son rapport CHAOS Report (2011-2015) [1], déclare que 29 % des projets informatiques réussissent (fonctionnalités, temps et budget respectés) contre 19% considérés comme des échecs (annulés ou jamais utilisés) (voir Figure 1: Le pourcentage des projets qui ont respecté le budget, le temp et les fonctionnalités, selon la base de données CHAOS 2011–2015.

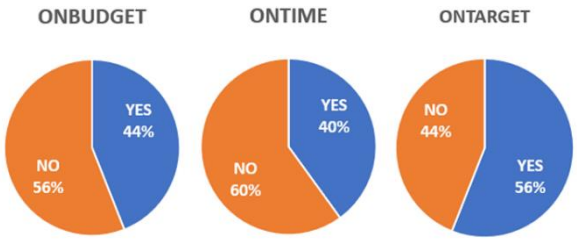


Figure 1: Le pourcentage des projets qui ont respecté le budget, le temp et les fonctionnalités, selon la base de données CHAOS 2011–2015.

Les trois principaux facteurs identifiés dans les projets réussis sont :

1. L'implication des utilisateurs.
2. Le soutien de la direction.
3. Les exigences claires.

Tandis que les trois principaux indicateurs trouvés dans les projets ayant des difficultés sont :

1. Faible engagement des utilisateurs.
2. Exigences et spécifications incomplètes.
3. Modification des exigences et des spécifications.

De ce fait, le succès des projets informatiques est étroitement lié non seulement aux compétences des membres de l'équipe IT, mais principalement à trois facteurs majeurs :

- Le degré d'implication des autres intervenants :

La divergence des points d'intérêt des différentes parties prenantes d'un projet informatique rend l'implication de ces intervenants limitée principalement dans les premières phases, où le langage naturel prédomine.

[1] Cabinet de conseil indépendant fondé en 1985, analyse les méthodes de conception d'une dizaine de milliers de projets informatiques depuis 1994 et publie des rapports (faiblesses & améliorations)

- La qualité avec laquelle les exigences sont écrites :

Et là encore, les dirigeants des organisations ne sont pas toujours familiarisés avec les diagrammes et les langages formels, et c'est la raison pour laquelle les exigences sont souvent exprimées en langage naturel non structuré (dans plus de 70% des cas[2]). Ainsi, l'ambiguïté du langage naturel compliquera certainement la communication entre les analystes et les ingénieurs.

- La capacité de s'adapter aux changements :

Le développement des applications qui repose principalement sur le code source trouve des difficultés majeures à adapter les solutions aux technologies en constante évolution, ce qui coûte cher aux entreprises au niveau du temps qu'au niveau du budget.

De tout ce qui précède, on ne peut qu'avouer que les conclusions de CHAOS Report sont totalement justifiées. Pour cette raison, les organisations continuent à s'efforcer de réduire le temp et le coût de développement et de maintenance des applications informatiques, à travers l'adoption de nouvelles méthodes de développement des solutions informatiques, et plus particulièrement :

- L'Architecture Dirigée par les Modèles ou MDA (Modèle Driven Architecture) [3], qui s'impose comme la solution tendance permettant la séparation des préoccupations, en se servant des modèles de réutilisation, des flux d'automatisation de processus et de génération de code.
- L'approche de règles métier [4] pour le développement de systèmes comme un moyen pratique pour construire de meilleurs systèmes rapides et facilement modifiables.

Généralement, la modélisation des logiciels commence par l'identification des exigences en passant par l'analyse et la conception, pour aboutir à la production du code source comme étape finale. Malheureusement, les langages formels utilisés pour favoriser l'automatisation de ce processus n'encouragent pas l'implication des experts métier dans la réalisation du projet. En même temps, la transformation des spécifications textuelles en langages (semi) formels compréhensibles par la machine n'est pas intégré dans le cycle de vie de MDA. C'est pour cela que l'effort requis à cette étape est considéré toujours comme source d'erreurs, demande du temps, et par la suite affectera certainement la rapidité de répondre aux changements. Partant de ce fait, toute tentative d'automatisation de la transformation Texte-to-Model dans les premières phases du processus de développement des applications aura certainement des effets positifs sur le temps, le coût et la fiabilité des modèles générés.

Motivation

Dans la littérature, y compris les travaux proposés par notre laboratoire « (Laboratoire de Mathématiques Informatique et Sciences de l'Ingénieur) » (MISI) [5] [6] [7] [8] [9] [10] [11] [12] [13], plusieurs approches ont été proposées dans le but d'automatiser les transformations M2M (Modèle-to-modèle) entre les différente couches MDA. A ce niveau-là, la fiabilité des modèles sources et cibles est étroitement liée au niveau de fiabilité des modèles qui les précèdent, jusqu'à ce qu'on arrive aux premières phases dans lesquelles le langage naturel non structuré joue un rôle primordial, ce qui justifie l'aspect critique des phases initiales. Cette situation nous a interpellé et a piqué notre intérêt face au traitement automatique du langage naturel avec lequel les

spécifications textuelles sont exprimées avant aucune transformation, dans le but de supporter et améliorer les approches déjà proposées.

▪ Problématique

Les erreurs introduites au cours des activités liées aux spécification des exigences représentent jusqu'à 50% de toutes les erreurs trouvées lors de la réalisation d'un produit logiciel[14]. Il existe différents types d'informations sur les exigences pour lesquels un ensemble d'adjectifs cohérent est utilisés pour les catégoriser. Chaque type d'exigence est influencé par d'autres types, en partant des règles métier jusqu'à ce qu'on arrive aux spécifications des exigences logicielles. Là encore, toute inconsistance au niveau des règles métier aura un effet négatif qui se propage dans toutes les autres types des exigences, et par la suite dans toutes les étapes du processus de développement des applications.

Problématique : La transformation directe des exigences textuelles vers un autre langage suffisamment formel est considérée comme la solutions idéale pour diminuer le temp dans les premières phases ainsi que pour automatiser l'exploitation de ces règles (filtrage et manipulation). Or, cela entre en conflit avec le fait que les experts métier ne sont généralement pas familiarisés avec les langages formels. Ainsi, toute mal compréhension (d'une part ou d'autre) peut avoir un impact négatif sur la représentation cible, et bien sûr, sur tout le reste du processus.

Objectif : Vue que les règles métier sont l'origine de plusieurs types d'exigences logicielle, notre objectif est d'améliorer leur fiabilité à travers leur transformation automatique vers le format le moins ambiguë et le plus compréhensible à la fois par la machine ainsi que par tout intervenant non-spécialiste. Ainsi, le langage cible peut être validé par les experts métier, et puis directement exploité par la machine, comme ça on fournira une base suffisamment fiable, sur laquelle les développeurs peuvent s'appuyer pour générer d'autre modèles aussi fiables qu'elle.

Question de recherche : Comment peut-on automatiser la transformation des règles métier depuis le langage naturel vers un langage formel, tout en impliquant les experts métier dans cette opération ?

Pour répondre à cette question tout en respectant l'architecture MDA, plusieurs sous-questions se posent :

- **Q1.** Quel est le modèle (cible) le plus approprié auquel on peut transformer nos règles métier afin qu'elles soient compréhensibles par les différents intervenants y compris la machine ?
- **Q2.** Quelle est le modèle (source) générique qui peut représenter toutes (ou la majorité) des structures linguistiques possible avec lesquelles une règle métier peut être formulée, pour qu'on puisse le transformer à notre modèle cible ?
- **Q3.** Quelles sont les éléments (reflétant la sémantique) à extraire depuis nos règles métier textuelle, nécessaire pour la formulation de notre modèle source ?
- **Q4.** A quel point peut-on automatiser l'identification de la sémantique de nos règles métier exploitée dans la transformation depuis le modèle source de la question Q2 vers le modèle cible de la question Q1 ?

▪ Contributions de la thèse

L'objectif principal de cette thèse est de contribuer à combler le fossé entre les experts métier et les responsables IT à travers l'identification automatique de la sémantique des règles métier écrites en langage naturel et la représenter dans un format compréhensible par toutes les parties prenantes y compris les machines. Pour cela, nous proposons une transformation automatique T2M (Texte to Modèle) pour générer un modèle des règles métier textuelles directement exploitable par la machine.

Cette thèse devrait intéresser tous les informaticiens qui travaillent dans le domaine du traitement du langage naturel, en particulier, ceux qui souhaitent transformer les règles métier textuelles en langage formel.

Cette thèse apporte les contributions suivantes :

1. Améliorer les méthodes de transformation Texte-to-modèle.
2. Aider les experts métier à bien structurer et décrire les règles métier de manière précise et sans ambiguïté.
3. Simplifiez la communication entre les experts métier et les experts IT.
4. Démarrez l'approche MDA dès les premières phases.

Pour cela, nous verrons dans un premier temps comment pourrait-on automatiquement extraire le vocabulaire utilisé pour exprimer les règles métier. Nous devrons également identifier les termes qui ont le même sens mais différentes expressions pour éviter tout mal compréhension, avant de générer le modèle final représentant la sémantique de chaque règle métier et le présenter dans un format accessible par les différents intervenants.

Les contributions proposées dans le cadre de ce travail répondent aux problématiques précédemment décrites en proposant :

- La génération automatique du dictionnaire terminologique à partir des règles métier textuelles

Dans ce travail nous présentons une approche automatique pour générer le vocabulaire métier depuis des règles métier écrites en langage naturel. Comme résultat, un dictionnaire terminologique a été généré tel qu'il est décrit par le standard SBVR (Semantic of Business Vocabulary and Rules) [15] pour donner une signification plus complète à chaque concept identifié. Pour cela, on a utilisé un traitement approfondi du langage naturel (NLP ou Natural Language processing) pour extraire non seulement une liste de termes-relations, mais également les informations implicites ainsi que des spécifications supplémentaires comme les définitions des concepts, les synonymes et les concepts généraux.

– Identification automatique de la similarité sémantique entre les concepts de règles métier textuelles

Dans ce travail, nous présentons une approche automatique pour identifier les termes ayant la même sémantique utilisée pour formuler les règles métier textuel, en utilisant à nouveau le traitement du langage naturel (NLP) renforcé par un algorithme, basé sur des heuristiques sur le contexte dans lequel les termes ont été utilisés.

– Génération automatique de la sémantique du vocabulaire et des règles métier à partir des règles métier textuelles

Dans ce travail, nous avons prolongé les deux travaux déjà proposés pour qu'on puisse transformer les règles métier textuelles vers un modèle SBVR plus complet qui peut couvrir la sémantique de toute la règle métier. Le modèle SBVR généré comprend en plus du dictionnaire terminologique, la formulation sémantique de chaque énoncé de règles métier.

▪ Organisation du manuscrite

Le reste du présent manuscrit contient deux grandes parties à savoir « L'état de l'art » et « Les contributions ». Chaque partie est organisée en chapitres, avant de clôture par une conclusion générale.

o État de l'Art

Cette partie est composée de trois chapitres introduisant les principaux concepts que nous aborderons le long de notre manuscrit.

- **Chapitre 1** – « *Exigences ou règles métier* » : Ce chapitre présente une vue globale sur les différents types d'exigences qui peuvent être utilisés pour déterminer le contexte dans lequel un système sera utilisé. Le focus sera plus sur les règles métier, sur lesquelles on a travaillé dans cette thèse.
- **Chapitre 2** – « Le SBVR et *l'architecture MDA* » *:* Ce chapitre met l'accent sur l'architecture ADM, et plus précisément la partie CIM vers laquelle on veut migrer nos spécifications textuelles. Le chapitre présente également une description générale du standard SBVR (notre modèle cible). La description couvre principalement le métamodèle SBVR utilisé dans nos contributions pour présenter à la fois le vocabulaire ainsi que les règles métier.
- **Chapitre 3** – « *Les règles métier : de l'informel au formel* » : Ce chapitre présente l'état de l'art des approches qui traitent les transformations des spécifications textuelles en langages formels, afin de positionner notre travail dans la littérature.

o Contributions

Cette partie est consacrée à la présentation des différentes contributions de cette thèse, elle comporte trois chapitres :

- **Chapitre 5 -** « *La génération automatique du vocabulaire métier* » : Ce chapitre présente une approche automatique pour générer le vocabulaire métier depuis des règles métier écrites en langage textuel, et le présenter sous forme d'un dictionnaire terminologique qui respect le standard SBVR.
- **Chapitre 6 -** « *L'identification automatique des synonymes* » : Ce chapitre aborde notre deuxième contribution, dans laquelle nous présentons une approche pour identifier automatiquement les termes ayant différentes expressions qui reflètent le même sens, en utilisant le NLP avec des heuristiques.
- **Chapitre 7 -** « *Les règles métier : de l'informel à SBVR* » : Ce chapitre présente notre méthode pour générer un modèle SBVR plus complet reflétant en plus du dictionnaire terminologique, la sémantique de toute la règle métier.

Pour conclure, ce manuscrit se termine par une synthèse générale des différents axes abordés dans cette thèse. Ensuite, nous proposons des perspectives et pistes d'évolution que nous envisageons pour compléter et améliorer ce travail.

▪ Productions Scientifiques

– Conférences Internationales publiées

Abdellatif Haj, Youssef Balouki, and Taoufiq Gadi, "*Automated Checking of Conformance to SBVR Structured English Notation*," in Advanced Intelligent Systems for Sustainable Development (AI2SD'2018), vol. 915, M. Ezziyyani, Ed. Cham: Springer International Publishing, 2019, pp. 697–706. DOI: 10.1007/978-3-030-11928-7_63

Abdellatif Haj, Youssef Balouki, and Taoufiq Gadi, "*Automatic Extraction of SBVR Based Business Vocabulary from Natural Language Business Rules*," in Xenopus, vol. 1865, K. Vleminckx, Ed. New York, NY: Springer New York, 2019, pp. 170–182. DOI: 10.1007/978-3-030-11914-0_19

– Articles publiés

Abdellatif HAJ, Youssef Balouki, Taoufiq Gadi "*Automated Identification of Semantic Similarity between Concepts of Textual Business Rules*", International Journal of Intelligent Engineering and Systems (IJIES), DOI: 10.22266/ijies2021.0228.15

Abdellatif HAJ, Youssef Balouki, Taoufiq Gadi "*Automated generation of Terminological Dictionary from textual business rules*", Journal of Software: Evolution and Process, John Wiley & Sons Ltd, ISSN 2047-7481, DOI : 10.1002/smr.2339

A. Haj, A. Jarrar, Y. Balouki, and T. Gadir, "The Semantic of Business Vocabulary and Business Rules: an Automatic Generation from Textual Statements," IEEE Access, pp. 1–1, 2021, doi: 10.1109/ACCESS.2021.3071623.

Etat de l'art

Sommaire :

1. Exigences ou règles métier ?

La contribution principale de cette thèse est de transformer les règles métier textuelles - l'origine de plusieurs types d'exigences logicielles - en langage formel. Ce chapitre présente une vue globale sur les règles métiers sur lesquelles on a travaillé dans cette thèse, ainsi que leur position par rapport aux différents types d'exigences existants.

Sommaire

Introduction

Durant la spécification des exigences, les intérêts des différentes parties prenantes se croisent plus que nulle part dans les phases d'un projet. Bien gérée, cette intersection peut conduire à des intervenants ravis et épanouis. Pour cette raison, les développeurs, les analystes métier, les utilisateurs, les clients, et bien d'autres, doivent respecter les bonnes pratiques des exigences pour avoir un produit de qualité supérieure, étant donné que les exigences sont à la base non seulement du développement des logiciels mais aussi des activités de gestion de projet. Notre contribution générale est de porter des améliorations à ce contexte, à travers l'amélioration des règles métier textuelles, l'origine de la majorité de ces exigences.

Dans le but de clarifier plus le contexte dans lequel on travaille, nous présentons dans ce chapitre le domaine des exigences et sa relation avec les règles métier (notre cible). Ce chapitre commence par définir ce qui est une exigence avant de présenter leur classification et leur hiérarchie qui reflète le positionnement dans lequel sont situées nos règles métier. De même, la définition ainsi que la classification des règles métier sont présentés, pour encadrer les limites de noter contribution.

1.1. Les exigences

Avant d'aller plus loin dans ce chapitre, il vaut mieux commencer par la terminologie.

1.1.1. Qu'est-ce qu'une « Exigence » ?

La notion d'exigence se diffère d'un contexte à l'autre. Par exemple, si on se met dans la peau d'un développeur, une exigence peut être un produit de qualité pour le client. Une fois dans la peau d'un client, une exigence peut être une description détaillée de l'interface utilisateur pour le développeur. Le terme « exigence » n'a jamais eu une définition. Dans ce qui suit, nous présentons quelques définitions que nous avons trouvées utiles ainsi que notre définition adoptée.

En commençant par jeter un coup d'œil dans le dictionnaire, on trouvera une définition de très haut niveau comme celle proposée par le dictionnaire de Cambridge « *quelque chose que vous devez faire ou quelque chose dont vous avez besoin* » [16].

Selon le consultant Brian Lawrence, une exigence est simplement «*tout ce qui détermine les choix de la conception*» [17]. D'après cette courte définition, l'essentiel est de faire des choix de conception appropriés qui répondront aux besoins du client. Mais il manque beaucoup de détails.

Une définition plus complexe vient de Ian Sommerville et Pete Sawyer : « *Les exigences sont une spécification de ce qui doit être mis en œuvre. Ce sont des descriptions de la façon dont le système doit se comporter, ou d'une propriété ou d'un attribut système. Ils peuvent constituer une contrainte sur le processus de développement du système* » [18]. D'après cette définition, les exigences présentent à la fois le comportement du système externe du point de vue de l'utilisateur,

ainsi que les caractéristiques internes du point de vue du développeur. Autrement dit, ils incluent les comportements du système dans des conditions spécifiques, ainsi que les propriétés qui rendent ce système approprié.

Tandis que notre définition préférée est suggérée par le Conseil International de l'Ingénierie des Systèmes (INCOSE) : « *Une déclaration qui identifie une caractéristique ou une contrainte de système, de produit ou de processus, qui est non ambiguë, claire, unique, cohérente, autonome (non groupée) et vérifiable, et est jugée nécessaire pour l'acceptabilité des parties prenantes* » [19].

Cette définition indique que :

- Une exigence a la forme d'une « déclaration », et c'est bien le format sur lequel on travaille dans cette thèse.
- Une exigence est une caractéristique ou contrainte, et ce sont bien les deux grands types de déclarations sur lesquels on travaille dans cette thèse.
- Une exigence doit être non ambigüe, claire, unique, cohérente, autonome et vérifiable. Ces caractéristiques sont bien ceux exigées par nos contributions.

Après ce tour des différentes définitions liées au terme « exigence », ça serait intéressant de présenter aussi la classification de ses types pour montrer le niveau hiérarchique dans lequel se situent les règles métier, et c'est bien ce qu'on essaie de faire dans la section suivante.

1.1.2. La classification des exigences

Selon Karl Wiegers [20], une seule déclaration peut être décrite comme étant une exigence métier, exigence utilisateur, exigence fonctionnelle, exigence système, attribut de qualité, une fonctionnalité, une contrainte ou une règle métier. Les noms utilisés pour les différents livrables des exigences varient également.

- *Exigence métier* :

Les exigences métier sont les objectifs métier de haut niveau de l'organisation qui font référence à un ensemble d'informations décrivant un besoin qui conduit à un ou plusieurs projets pour fournir des solutions souhaités (e.g. Les opportunités, les objectifs et les mesures de réussite).

- *Exigence utilisateur :*

Dans un sens plus large, ils sont appelés aussi « exigences des parties prenantes ». Ce type d'exigences décrit ce que l'utilisateur doit être en mesure de faire avec le système ou un attribut de produit qui apportera de la valeur à quelqu'un. Il comprend également les descriptions des attributs ou des caractéristiques du produit qui sont importants pour la satisfaction de l'utilisateur. Les cas d'utilisation, les user stories sont les moyens les plus utilisés pour présenter les besoins des utilisateurs (e.g. "En tant que passager, je souhaite m'enregistrer pour un vol afin de pouvoir embarquer dans mon avion.").

- *Exigence fonctionnelle :*

Ils décrivent le comportement que le système présentera dans des conditions spécifiques pour permettre aux utilisateurs d'accomplir leurs tâches. (e.g. "*If the profile of the passenger does not have a seating preference, the reservation system shall assign a seat*").

- *Exigence du système :*

Une exigence de niveau supérieur pour un produit composé de plusieurs composants ou sous-systèmes, qui peuvent être des logiciels ou/et matériels, y compris les personnes et les processus de ces systèmes (e.g. poste de travail du caissier dans un supermarché constitué de lecteur de codes à barres, écran et tiroir-caisse qui interagissent sous contrôle de logiciel).

- *Exigences non fonctionnelles*
  - *Attribut de qualité :*

Appelés aussi 'facteurs de qualité' ou 'capacités'. Elles sont des exigences non fonctionnelles qui décrivent un service ou une caractéristique de performance d'un produit dans diverses dimensions (e.g. performances, sécurité, disponibilité et portabilité).

  - *Exigence d'interfaces externes :*

Elles décrivent la connexion entre un système logiciel et le reste de l'univers qui peuvent être des utilisateurs, autres systèmes ou matériel (e.g. 'le système doit être capable de lire des fichiers au format XML').

  - *Contrainte :*

Les contraintes de conception et de mise en œuvre imposée aux options disponibles pour le développeur (e.g. 'La solution doit être implémentée en langage Java').

- *Règle métier :*

En gardant tous simplement le plus important pour la fin, les règles métier existent au-delà des limites d'une application logicielle spécifique, en dictant souvent que le système doit contenir des fonctionnalités pour se conformer aux règles pertinentes. C'est pour cela qu'elles ne sont pas des exigences logicielles en soi, mais l'origine de plusieurs types d'exigences logicielles. Par conséquent, on peut retracer certaines exigences fonctionnelles jusqu'à une règle métier particulière. Et c'est pour ce rôle 'pivot' que nous les avons ciblés.

Dans l'ensemble, les exigences logicielles comprennent trois niveaux distincts : les exigences métier, les exigences utilisateur et les exigences fonctionnelles. Cette hiérarchie est illustrée dans la Figure 2 Les types des exigencesd'une manière incomplète mais utile comme schéma organisationnelle montrant la position de nos règles métier cibles.

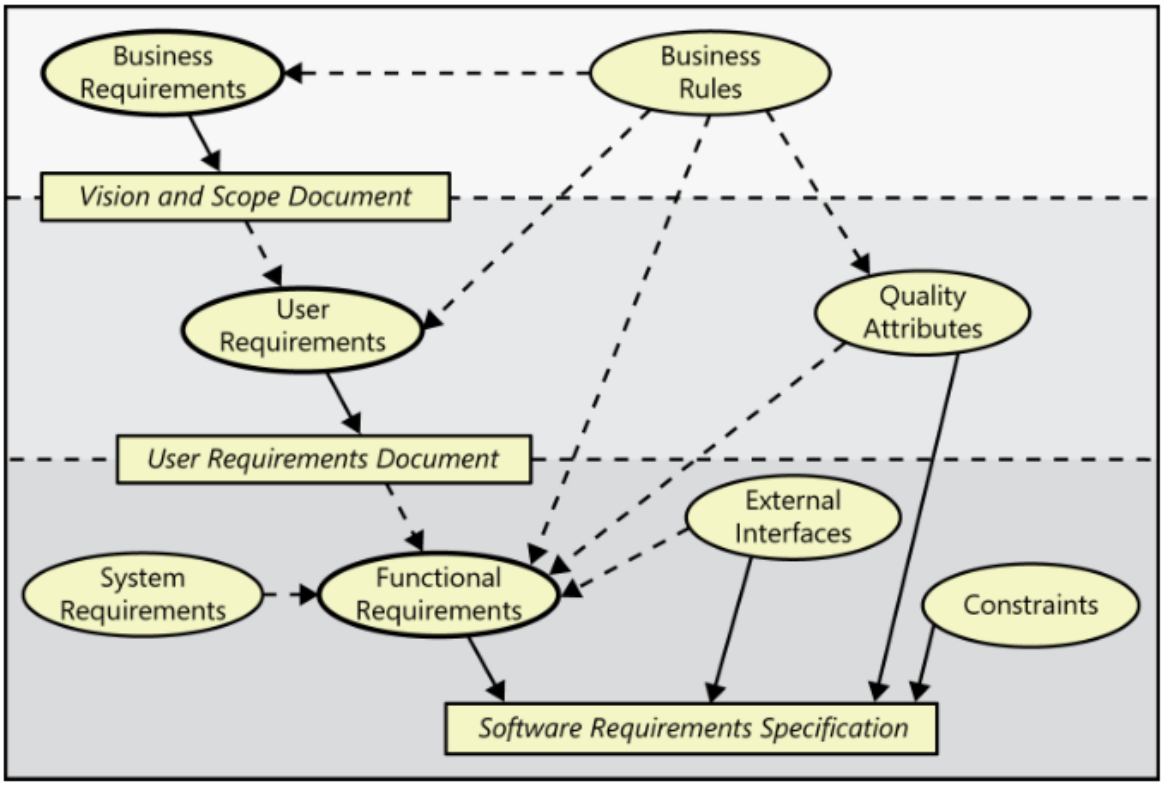


Figure 2 Les types des exigences et la relations entre eux

Les formes ovales de la figure représentent les types des exigences, tandis que les rectangles représentent les documents dans lesquels ces informations sont stockées. Les flèches pleines signifient « sont stockés dans ». Alors que les flèches en pointillés signifient « est à l'origine de ».

Les règles métier et les exigences système sont stockées séparément des exigences logicielles, au niveau le plus haut de cette hiérarchie, ce qui reflète leur importance par rapport aux autres types d'exigences. D'autant plus, les règles métier sont à l'origine des différents types d'exigence situés dans les différents niveaux de cette hiérarchie, à savoir les exigences métier, les exigences utilisateur, les exigences fonctionnelles. Cette situation a piqué notre intérêt à transformer ces règles métier textuelles en un format plus formel dans le but de pouvoir en bénéficier pour supporter les autres types d'exigences d'une part, et pour encourager l'automatisation du processus de développement des logiciels dans cette phase d'autre part.

La section suivante porte plus de détails sur les règles métier, leur définition et leur classification.

1.2. Les règles métier

Pour prendre des décisions, nos comportements diffèrent d'une personne à l'autre en fonction de la logique acquise durant les expériences passées. De même, les entreprises prennent des

décisions basées sur des faits et des règles connues sous le nom de « règle métier » ou Business Rules (BR) en anglais.

Comme pour les stratégies de sécurité de l'entreprise, les règles métier sont à l'origine d'attributs de qualité spécifiques qui sont ensuite implémentés dans les fonctionnalités. Elles comprennent les politiques de l'entreprise, les réglementations gouvernementales, les normes de l'industrie et les algorithmes de calcul qui définissent ou contraignent certains aspects de l'entreprise.

Pour tracer les limites de notre contribution, il est judicieux de commencer tous d'abord pas déterminer le concept « règle métier » ainsi que ses différents types.

1.2.1. Qu'est-ce qu'une « Règle métier »

Plusieurs définitions ont été produites pour le terme règle métier.

Une définition utile et assez largement citée proposée par Tony Morgan dans son livre [21] « *La règle peut être exprimée en termes directement liés à l'entreprise, en utilisant un langage simple et sans ambiguïté accessible à toutes les parties prenantes : propriétaire d'entreprise, analyste métier , architecte technique, etc.* ».

Cette définition indique trois grandes idées :

- Les règles métier sont exprimées en termes liés à l'entreprise.
- Les règles métier utilisent un langage simple et sans ambiguïté.
- Les règles métier sont accessible pour toutes les parties prenantes.

En fait, nos contributions vont dans le même sens que ces trois points. Notre première et deuxième contribution concernent la standardisation de la terminologie utilisée au sein des organisations. Alors que notre troisième contribution vise la simplification du langage avec lequel les règles métier sont exprimées, en prenant en considération l'accessibilité du format généré pour toutes les intervenant.

Une autre définition est proposée par l'équipe du projet « Guide Business Rule Project» [22] (groupe de personnes qui se consacrent à la normalisation des termes entourant le concept de règles métier). La définition qui figure dans le rapport final du Business Rules Group de juillet 2000 est « *Une règle métier est une déclaration qui définit ou contraint certains aspects de l'entreprise. Il vise à affirmer la structure de l'entreprise ou à contrôler ou influencer le comportement de l'entreprise*. ».

D'après cette deuxième définition, les règles métier sont des définitions ou des contraintes qui visent la structure ainsi que le comportement de l'organisation, ce qui reflète - d'une part - l'importance de ces déclarations pour les organisations, et c'est la raison pour laquelle on les a visées dans nos contributions comme nous l'avons déjà mentionné. D'autre part, les différentes classes des règles métier doivent être présentées pour déterminer le cadre dans lequel nos contributions se situent. C'est ce qu'est présenté dans la section suivante.

1.2.2. La classification des règles métier

Différents types de déclarations peuvent être considérées comme des règles métier, cependant il n'existe pas de classification universelle pour ces règles. Différents schémas de classification ont été proposées pour les règles métier, selon leurs objectifs et leur public visé. Citer au moins un schéma de classification va nous permettre de connaitre la gamme complète de ces règles et de mieux déterminer le cadre dans lequel on va travailler. Nous avons adopté le schéma de classification des règles métier de Von Halle décrit dans son livre [23] (voir Figure 3 Schéma de classification des règles métier).

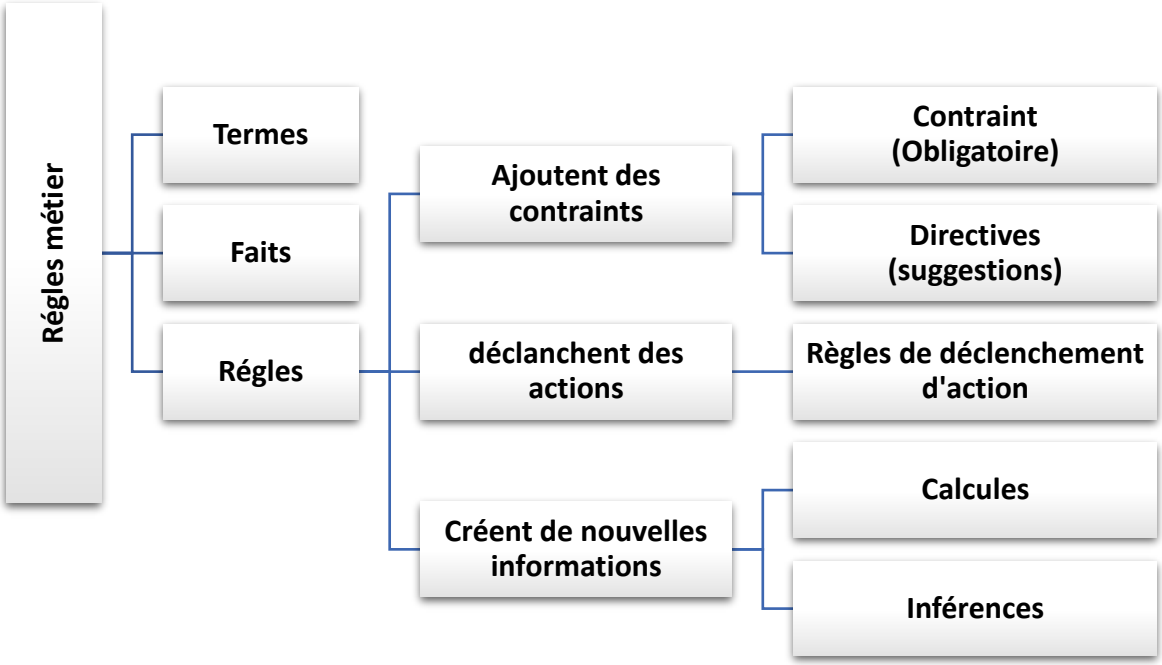


Figure 3 Schéma de classification des règles métier

Tableau 1 Classification des règles métier : descriptions et exemples

	Description	**Exemple**
Terme	Un nom ou une phrase nominale avec une définition convenue, qui peut définir un concept, une propriété, une valeur ou un ensemble de valeurs	Rental Car Days of the Moroccan Work
Fait	Un énoncé reliant les termes à des observations pertinentes pour l'organisation	Customer places order.
Règle	Un énoncé déclaratif qui utilise des informations en entrée et crée en sortie, soit des informations, soit une action en appliquant une logique exécutable	
Contrainte	Une déclaration complète exprimant les circonstances qui doivent être vraies pour les informations d'entrée afin de garder l'intégrité des événements.	A customer MUST not have more than 10 open orders.
Directive	Une déclaration complète qui exprime un avertissement sur une circonstance qui devrait être vraie.	A customer SHOULD not have more than 10 open orders.
Déclenchement d'action	Une déclaration complète qui teste les conditions avant de lance un autre événement, un message ou une autre activité.	If a customer is high risk, then notify the customer services.
Calcule	Une déclaration complète qui fournit un algorithme pour arriver à la valeur d'un terme.	The total-amount-due for an order is computed as the sum of the line-item -amount for the order plus tax.
Inférence	Une déclaration complète qui teste des conditions avant d'établit la vérité d'un nouveau fait.	If a customer has no outstanding invoices, then the customer is of preferred status.

En générale, il existe trois grandes catégories des règles métier : i) les règles exprimant des faits qui doivent ou devraient être vraies ou non, ii) les règles vérifiant une condition avant d'activer d'autres actions, iii) et les règles générant de nouvelles informations (e.g. calculs).

La différence entre ces types de règles métier nécessite une différence dans les structures des phrases utilisées pour exprimer ces règles. Vue le nombre de patterns pouvant être utilisés pour exprimer ces règles, nous nous sommes concentrés dans cette thèse sur les règles de la première et de la deuxième catégorie ; à savoir, les règles métier exprimant des contraints et des directives, en plus des règles lançant des actions après un test de condition (les règles ayant des antécédents et des conséquences). Ces types de règle métier jouent un rôle important dans la génération

automatique du dictionnaire terminologique que nous essayons de générer à travers nos contributions. Les autres types de règles métier peuvent être facilement intégrés par la suite, en ajoutant leurs patterns associés (et lexique si nécessaire). Puisque cela se fera au détriment de la clarté de nos approches, ces types de règles ignorées seront prises en compte dans nos futurs travaux.

Conclusion

Dans ce chapitre nous avons essayé de refléter la complexité de la phase d'Inception, durant laquelle différents types d'exigences sont traitées, et qui sont souvent basées sur des règles métier. Ces dernières qui ne sont pas moins complexe que les exigences. Malheureusement il n'y a pas de solutions magiques pour traiter et gérer toutes ces déclarations, surtout qu'elles sont souvent écrites en langage naturel non formel. Et c'est bien la raison pour laquelle les nouvelles méthodes de développement basées sur les modèles n'intègrent pas cette phase dans leur processus.

L'architecture dirigée par les modèles (MDA : Model Driven Architecture) proposée par l'OMG (Object Mangement Group) [24], et adoptée par notre laboratoire de recherche afin d'automatiser le processus de développement, est l'une des méthodes qui ne prend pas en considération cette phase. Cependant, L'OMG a proposé en 2008 le standard SBVR (Semantic of Business Vocabulary and Rules) [25] qui permet de documenter la sémantique du vocabulaire métier et des règles métier dans un langage naturel et les exprimer dans une logique formelle afin qu'elles puissent être comprises à la fois par les utilisateurs et les machines.

Dans cette thèse nous essayons de combler l'écart entre la phase d'Inception dans laquelle le langage naturel domine, et les différentes couche MDA dans lesquelles des modèles formels sont utilisés, en utilisant le standard SBVR. Avant de discuter les différentes approches proposées dans la littérature pour résoudre ce problème, nous présentons tout d'abord dans la section suivante l'architecture MDA, ainsi que le standard SBVR, notre modèle cible.

CHAPITRE 2

2. Le SBVR et l'architecture MDA

Ce chapitre met l'accent de façon générale sur l'architecture MDA (Model Driven Architecture), et plus précisément la partie CIM (Computation Independent Model) vers laquelle on veut transformer nos règles métier textuelles. Le chapitre présente également une description générale du standard SBVR (notre modèle cible). La description couvre principalement le métamodèle SBVR utilisé dans nos contributions pour présenter à la fois le vocabulaire ainsi que les règles métier.

Sommaire

Introduction

En 2001, l'OMG a adopté l'architecture MDA comme une approche de développement de logiciels basée sur les modèles. Cette architecture est basée sur des spécifications existantes proposées par l'OMG telles que l'UML (Unified Modeling Language). Cette approche est largement adoptée dans différent projets y compris notre laboratoire de recherche (MISI) [5] [6] [7] [8] [9] [10] [11] [12] [13].

Les spécifications textuelles comme les exigences et les règles métier sont souvent à l'origine des différents modèles de la MDA, et qui ne sont souvent pas prises en compte par cette architecture. C'est pour cela que l'OMG a proposé en 2008 le standard SBVR (Semantic of Business Vocabulary and Rules) pour représenter les spécifications textuelles en langage compréhensible à la fois par les utilisateurs ainsi que les machines. Cependant, les différentes parties prenantes d'un projet ne sont souvent pas familiarisées avec le standard SBVR.

La contribution de cette thèse est de combler le fossé entre les spécifications textuelles de la phase d'Inception et l'architecture MDA, à travers la transformation automatique des règles métier textuelles vers le standard SBVR en utilisant la transformation Text-to-Model. La notion de la 'transformation' est l'un des concepts fondamentaux de la MDA, ainsi que d'autres concepts qu'on va découvrir dans ce chapitre, en plus du standard SBVR (notre modèle cible).

2.1. MDA : Model Driven Architecture

Avec ces technologies et ces nouvelles plates-formes en constante évolution, la MDA devient de plus en plus la meilleure solution pour la rationalisation du processus de leur intégration ainsi que pour le développement rapide de nouvelles spécifications.

L'utilisation des modèles dans le processus de développement logiciel encourage la séparation des préoccupations, ce qui diminue le temps requis par ce processus et favorise la réutilisabilité de ces modèles (dans d'autres projets par exemple).

L'aspect fondamental de la MDA est sa capacité de couvrir tout le cycle de vie du processus de développement, en partant de l'analyse et la conception, passant par la programmation et les tests, en arrivant au déploiement et la maintenance.

Dans la suite de cette section, nous allons présenter les concepts fondamentaux de la MDA, et plus précisément, les modèles et la transformation entre eux.

2.1.1. Les concepts fondamentaux de la MDA

La MDA ou l'architecture dirigée par les modèles est une approche de développement logiciel dans laquelle les modèles sont utilisés pour analyser, concevoir, déployer, maintenir et documenter un système, et qui inclut les concepts suivants :

- *L'architecture*

L'architecture d'un système est la spécification des composant de ce système ainsi que leurs connecteurs et les règles d'interaction entre eux. Dans le contexte de la MDA, tous ces éléments sont exprimés en utilisant un ensemble de modèles interdépendants.

- *Modèle*

Un modèle est une spécification formelle de la structure d'un système ou son comportement, d'un point de vue spécifique et dans un contexte donné. Les modèles sont souvent représentés par des diagrammes utilisant généralement une notation formelle (e.g. UML), mélangés avec des expressions en langage naturel. Une spécification est dite formelle lorsqu'elle est basée sur un langage qui a une signification sémantique bien définie associée à chacun de ses composants. Ce formalisme permet aux modèles de s'exprimer selon un schéma bien défini tel que XMI.

Les modèles sources et cibles utilisés dans cette thèse sont respectivement le langage naturel et le standard SBVR, qu'on va détailler dans les sous sections suivantes.

- *Le Point de vue*

Un point de vue est une technique d'abstraction permettant de se concentrer sur un ensemble particulier de préoccupations au sein d'un système, représenté en utilisant des modèles.

La MDA spécifie trois points de vue par défaut sur un système : i) le point de vue indépendant de la programmation qui décrit les exigences du système, sans tenir compte de sa structure ou de son traitement ; ii) le point de vue indépendant de la plate-forme qui décrit les capacités opérationnelles d'un système en dehors du contexte d'une plate-forme spécifique ; iii) et le point de vue spécifique à une plate-forme qui ajoute au point de vue précèdent, les détails relatifs à l'utilisation d'une plate-forme spécifique (les langages de programmation, les bases de données, les systèmes d'exploitation, etc.).

La contribution de cette thèse couvre le premier point de vue, qui est le point de vue indépendant de la programmation, et plus précisément, les règles métier textuelles.

- *Transformation de modèles*

La transformation de modèle est le processus de conversion d'un modèle en un autre dans le même système. Cette transformation peut - par exemple - combiner le modèle indépendant de la plate-forme avec d'autres informations pour générer un modèle spécifique à une plate-forme. C'est à travers l'application des règles de transformation sur des modèles sources qu'on obtient d'autres modèles cibles. Ce modèle peut être appliqué à plusieurs reprises à des modèles successifs.

Les deux sous sections suivantes ajoutent plus de détails sur les deux concepts clés de la MDA qui sont les modèles et les transformations.

2.1.2. Les modèles MDA

Pour représenter les trois points de vue définis ci-dessus, la MDA définie trois couches d'abstraction par défaut. Dans chacune de ces trois couches, un ensemble de modèles peut être créer, pour représenter un point de vue plus spécifique du système.

- *Modèle des exigences (CIM : Computation Independent Model)*

Ce modèle présente ce que le système est censé faire indépendamment de la technologie de mis en œuvre, en utilisant un vocabulaire familier aux experts métier, ce qui joue un rôle important à combler le fossé qui existe généralement entre les experts métier et les experts IT, responsables de la mise en œuvre du système. Pour cette raison, ce domaine est également appelé « modèle métier » ou « modèle de domaine ».

Dans la spécification MDA, les modèles CIM doivent être traçables aux modèles PIM et PSM qui les implémentent (et inversement). Vue l'aspect informel ainsi que l'ambiguïté qui caractérise le langage naturel utilisé généralement à ce niveau, l'OMG a proposé en 2008 le standard SBVR pour représenter la sémantique du langage naturel dans un format compréhensible par la machine. Mais, pour bénéficier de ce standard, les utilisateurs ont besoin de se former sur le métamodèle SBVR, ainsi que l'annotation SBVR-SE utilisée généralement pour exprimer les spécifications SBVR, ou bien utiliser un outil qui aide à la rédaction de ces spécifications selon ce standard.

La génération automatique de la sémantique des spécifications écrites en langage naturel tout en respectant le standard SBVR sera très bénéfique en termes de temp, comme en termes de compatibilité avec le principe de la MDA. Et c'est bien là où se situe la contribution de cette thèse.

- *Modèle indépendant de plate-forme (PIM : Plateforme Independent model)*

Un modèle PIM définit généralement les connaissances sans détails techniques, ce qui le rend suffisamment indépendant pour permettre sa mise en correspondance avec plusieurs plates-formes.

- *Modèle spécifique à la plate-forme (PSM : Plateforme Specific model)*

Un modèle PSM ajoute aux spécifications du modèle PIM les détails nécessaires pour décrire comment un système utilise une plate-forme spécifique.

2.1.3. La transformation entre modèles (Mapping)

Le concept de la transformation joue un rôle clé dans l'approche MDA. Il s'agit de la conversion d'un modèle (source) en un autre (cible) en utilisant des règles de transformation.

Dans l'ensemble global des modèles de l'architecture MDA, des transformations horizontales et verticales peuvent se produire ; c'est-à-dire, à l'intérieur d'une même couche d'abstraction et entre les couches aussi.

Le mappage dans l'architecture MDA fournit des spécifications sur la façon de transformer un modèle source en un autre bien spécifique. Bien que la transformation puisse être manuelle à un certain moment, l'objectif final est d'automatiser autant de processus que le permet le jeu d'outils utilisé.

Le cycle de vie MDA commence à partir du modèle d'analyse et se termine par la génération du code source, en utilisant une série de transformations PIM vers PSM et PSM vers le code source. Cependant, la transformation CIM vers PIM ne fait pas partie de ce cycle. Pour cette raison, plusieurs approches ont été proposées pour convertir les modèles CIM en PIM, mais peu de travaux ont couvert la phase pré-CIM dans laquelle le langage naturel domine. La phase pré-CIM est la première et principale source de communication entre les experts métier et les experts IT. Ainsi, il a un rôle spécial dans le développement de logiciels qui est négligé dans les approches pratiques de la MDA.

Les recommandations de l'OMG dans le cadre de la MDA sont :

- La transformation de modèles est le processus de convertir un modèle en un autre du même système.
- Un mappage de type de modèle spécifie un mappage de tout modèle construit à l'aide d'un langage de modèle source à des modèles exprimés à l'aide d'un langage de modèle cible.
- La transformation peut être manuelle, avec une assistance informatique, ou automatiquement.

Finalement, une transformation de modèle apparaît comme une fonction qui prend en entrée un ensemble de modèles et qui produit en sortie un autre ensemble de modèles. Les modèles d'entrée et de sortie, dans la transformation de modèle MDA, sont structurés par des métamodèles.

Les modèles sources et cibles utilisés dans cette thèse sont respectivement le langage naturel et le standard SBVR. Ce dernier possède un métamodèle qui peut être formalisé selon un schéma XMI proposé dans la documentation officielle. Alors que le langage naturel peut être utilisé dans différents format non standardisés, ce qui nous a obligé de lui créer un métamodèle générique englobant les différents éléments souvent existants dans les déclarations textuelles utilisées pour exprimer les règles métier, avant de les représenter selon le standard SBVR. Ce qui nous a permis de respecter l'architecture dirigée par les modèles de la MDA.

Notre métamodèle source utilisé pour les besoins de nos contributions est présenté dans la partie « contributions » ; alors que le métamodèle du standard SBVR (notre cible) est présenté dans la section qui suit.

2.2. SBVR: The Semantic of Business Vocabulary and Rules

« Semantic of usiness Vocabulary and rules » (SBVR) [15] est un standard de l'OMG publié en 2008, et partie intégrante de la MDA pour la modélisation du langage naturel. SBVR permet de formaliser à la fois le vocabulaire métier et les règles métier en utilisant le langage naturel et les représenter dans une logique formelle compréhensible par la machine.

Ce standard est caractérisé par :

- Il prend en charge le développement multilingue, car il est centré sur les concepts et pas sur les termes, ce qui permet de séparer les expressions de leur signification. Cela permet aussi au vocabulaire d'être partagé par diverses communautés et domaines.
- Il prend en charge la modélisation orientée fait précise dans la logique formelle.
- Il prend en charge le schéma XMI permettant au vocabulaire et règles métier d'être interchangés entre les organisations et outils logiciels.
- Il prend en charge les notations textuelles telles que SBVR-SE et RuleSpeak [26], permettant d'être mappée selon le métamodèle SBVR.
- Il est basé sur la logique formelle avec une interface en langage naturel.
- Il peut être transformé en d'autres modèles en utilisant la transformation de modèle.

Sur la base de ces caractéristiques, nous avons choisi le standard SBVR comme modèle pivot pour représenter les spécifications textuelles en langage formel.

Nos contributions traitent les deux éléments clés standardisés par SBVR, à savoir : le vocabulaire métier et les règles métier. La **Erreur ! Source du renvoi introuvable.** présente une v ue simplifiée sur le métamodèle SBVR qui décrit ces deux éléments. Un métamodèle plus détaillé est discuté dans la partie 'Contributions'.

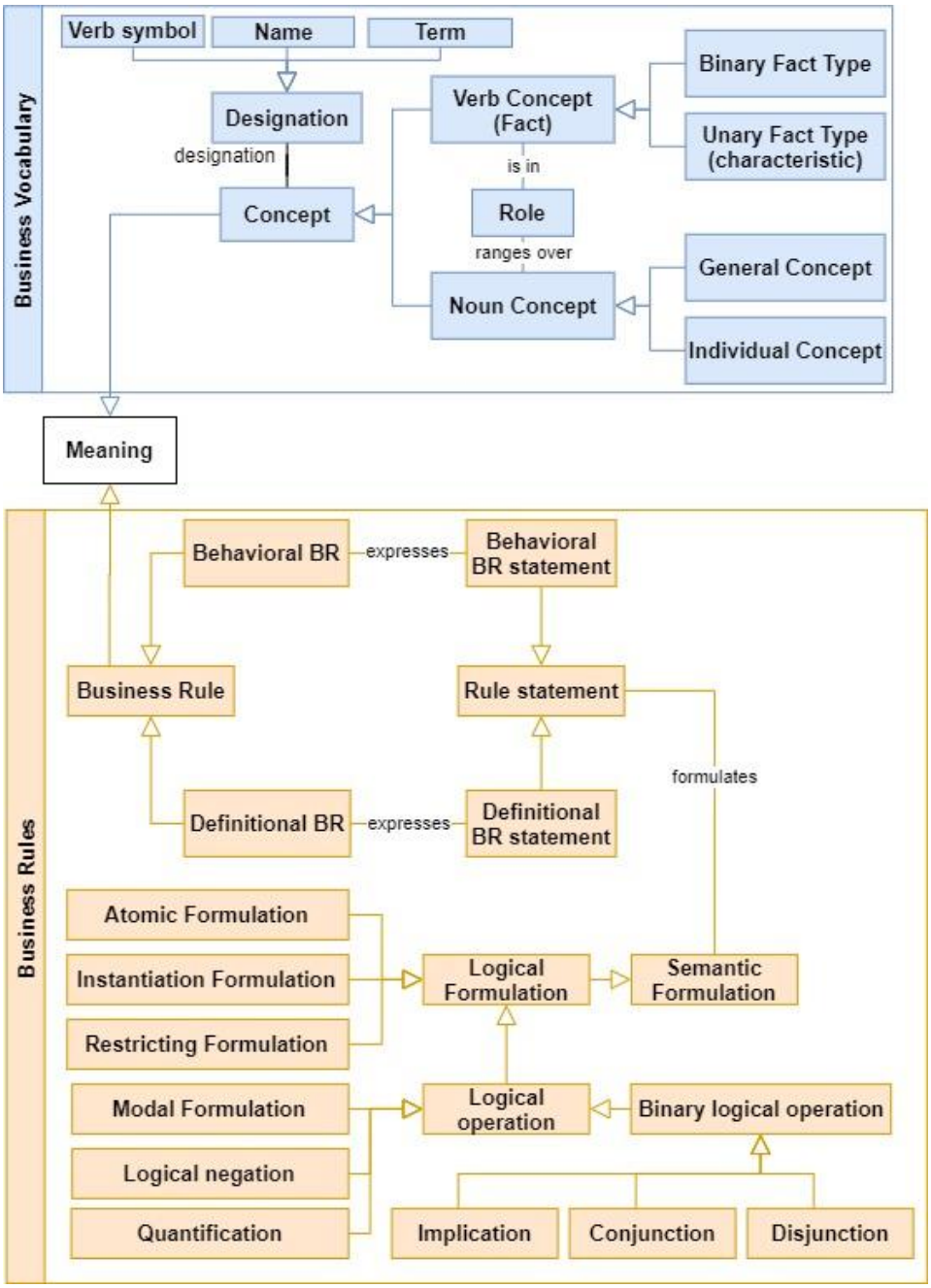


Figure 4 Le métamodèle SBVR simplifié

Selon le standard SBVR, la communauté du discours (speech community) utilise le vocabulaire métier pour construire le dictionnaire terminologique spécifique à une organisation. Ce dictionnaire contient généralement les expressions (désignations) exprimant des concepts, ainsi que des déclarations (statements) exprimant des règles métier.

Un concept peut être - d'une part - un concept de nom (Noun Concept), qui peut être à son tour un concept général (General Concept), ou un concept individuel (Individual Concept). Un

concept peut être aussi un concept de verbe (Verb Concept) qui peut être à son tour un concept de verbe binaire, ou un concept de verbe unaire (représentant une caractéristique).

En ce qui concerne les règles métier, elles sont exprimées en utilisant des déclarations (statements) basées sur les concepts. Les règles métier peuvent être soit une règle comportementale, ou définitionnelle, qui sont formulées en utilisant des formulations sémantiques à savoir : les formulations atomiques, les instanciations, les restrictions, les modalités, les quantifications, la négation, l'implication, les conjonctions etc.

Une description générale des principaux éléments constituant le métamodèle SBVR est présentée dans les deux sous-sections suivantes.

2.2.1. Le vocabulaire SBVR

Selon la documentation SBVR, le vocabulaire métier est définie par :

Definition 1: "*Business Vocabulary is a set of designations (such as terms and names) and verb concept wordings primarily drawn from a single language to express concepts within a body of shared meanings*".

Autrement dit, notre communication implique des expressions qui peuvent avoir plusieurs significations ; ainsi, la norme SBVR a introduit la notion de '*Concept*' et '*Designation*', pour séparer les significations de leurs expressions. Chaque désignation a un signifiant (expression) représentant un concept (signification).

Ce qui suit est une description des principaux types de concepts utilisés par SBVR :

- *Noun concept :*

Un concept de nom est défini comme '*concept qui est la signification d'un nom ou d'une phrase nominale*'. Il peut être un concept général ou individuel.

– General Concept :

Le concept général (ou concept de nom général) catégorise les choses sur la base de leurs propriétés communes. Il est défini dans la documentation par : '*concept de nom qui classe les choses en se basant sur leurs propriétés communes*'.

Ce type de concept incorpore un ensemble de caractéristiques comme combinaison unique qui distingue ce concept de tous les autres concepts. Autrement dit, si un concept général A a les mêmes caractéristiques qu'un autre concept général B, cela signifie qu'ils sont logiquement équivalents et qu'ils dénotent les mêmes choses dans tous les mondes possibles.

En langages naturels, les noms communs (e.g. '*voiture de location*' correspondant aux voitures louées) peuvent être représentés comme des noms généraux. Alors que dans les modèles de classe UML, un nom général peut être mappé à une classe UML.

– Individual Concept :

Le concept individuel est défini dans la documentation par : '*le concept de nom qui correspond à au plus une chose dans tous les mondes possibles*'. C'est à dire que ce type de concept correspond à un seul objet. En même temps, il peut y avoir plusieurs concepts de noms individuels qui correspondent à la même chose.

En langage naturel, les noms propres sont classés comme des concepts individuels (e.g. Le concept de nom individuel '*Marrakech*' ou '*Elbahja*' dont l'unique exemple est une ville individuelle au Maroc). Alors que dans les modèles de classe UML, un concept individuel peut être mappé à un objet.

- *Verb concept (Fact Type):*

Également appelé '*Type de fait*', le '*verb concept*' est défini dans la documentation par '*la signification d'une phrase verbale qui implique un ou plusieurs rôles*'. Il peut être de plusieurs types, à savoir :

– Binary Fact Type:

Le type '*Binary Fact Type*' est défini dans la documentation par '*le concept de verbe qui comprend exactement deux rôles*'. Généralement, dans les '*Binary Fact Type*', il existe une phrase verbale qui relie les deux rôles. L'exemple, '*the customer enters the card*' est une phrase verbale qui relie les deux rôles '*customer*' et '*card*'.

En langage naturel, la combinaison du sujet, du verbe et de l'objet forme un '*Binary Fact Type*'. Alors que dans les modèles de classes UML, un '*Binary Fact Type*' peut être mappé à une association entre deux classes.

– Unary Fact Type (Characteristic):

'*Characteristic*' est un type de '*Verb Concept*'. Il a toujours exactement un seul rôle (e.g '*the password is valid*').

En langage naturel, une '*characteristic*' peut être un adjectif ou un nom associatif. Alors que dans les modèles de classe UML, une '*characteristic*' peut être mappée à un attribut d'une classe.

A la fin, notre première contribution concerne la génération automatique d'un dictionnaire terminologique qui respect le standard SBVR et qui comporte les différents types de concepts. Pour cette raison, nous avons adopté un métamodèle (basé sur le métamodèle SBVR) comme métamodèle cible que notre dictionnaire doit respecter. Plus de détails sur le métamodèle SBVR adopté se trouve dans la partie 'Contributions'.

2.2.1. Les règles métier SBVR

Selon la documentation, une règle SBVR peut être formellement définie comme '*un élément d'orientation qui introduit une obligation ou une nécessité'*.

Sur la base des informations représentées dans une règle SBVR, les règles peuvent être classées en deux types ; règle de définition et règle de comportement :

- *Règles de définition*

Également appelée règle de structure, une règle de définition est utilisée pour définir la configuration d'une organisation ainsi que pour représenter sa structure. Par exemple, la règle suivante : '*It is necessary that each customer has at least one account*' explique qu'un client doit avoir au moins un compte, même si elle peut avoir plus d'un compte.

- *Règles de comportement*

Également appelée règle d'opération, une règle de comportement exprime la conduite ou le comportement d'une entité. Généralement, une règle de comportement est une revendication d'obligation. Par exemple, dans la règle suivante : '*It is obligatory that each customer can withdraw a maximum of GBP 200 per day*', le comportement du compte d'un client s'explique en disant qu'un client ne peut pas retirer plus de 200 GBP de son compte.

En utilisant le vocabulaire métier, la sémantique de ces règles est représentée sous forme de structures logiques appelées '*formulations logiques*'.

2.2.2. Formulation sémantique SBVR

Une formulation sémantique qui façonne une règle métier est appelée une formulation logique. Voici une brève description des principales formulations sémantiques utilisées par SBVR :

- *Formulation modale :*

Une formulation modale est une formulation logique utilisée pour refléter que le sens d'une autre formulation logique a une relation particulière avec les mondes possibles ou avec les mondes acceptables.

Il existe quatre types de base de formulations modales.

– Formulation de nécessité :

Si une formulation logique est représentée comme une formulation de nécessité, c'est qu'elle est vraie dans tous les mondes possibles. En anglais, une phrase contenant des mots comme '*need*' peut être mappée en formulation de nécessité. Dans SBVR une formulation de nécessité est représentée par la phrase '*It is necessary that*'.

– Formulation d'obligation :

Si une formulation logique est vraie dans tous les mondes acceptables, elle est représentée comme une formulation d'obligation. En anglais, une phrase contenant des mots comme '*should*', '*must*', et '*have to*' peut être mappée en formulation d'obligation. Dans SBVR une formulation d'obligation est représentée en utilisant la phrase '*It is obligatory that*'.

– Formulation de permission :

Si une formulation logique est vraie dans certains mondes acceptables, elle est représentée comme une formulation de permission. En anglais, une phrase contenant des mots comme '*may*' peut être mappée en formulation de permission. Dans SBVR, une formulation de permission est représentée par l'expression '*It is permitted that*'.

– Formulation de possibilité :

Si une formulation logique est vraie dans certains mondes possibles, elle est représentée comme une formulation de possibilité. En anglais, une phrase contenant des mots tels que '*can*' et '*could*' peut être mappée en formulation de possibilité. Dans SBVR, une formulation de possibilité est représentée par la phrase '*It is possible that*'.

- *Opérations logiques :*

Les opérations logiques sont utilisées pour combiner une ou plusieurs expressions, pour produire des expressions plus complexes.

Nos contributions prennent en charge les types d'expressions logiques suivants :

– Conjonction :

Dans SBVR, une conjonction est une opération logique binaire utilisée pour formuler la signification d'une décision logique de deux opérandes que chacun d'eux est vrai (i.e. p ET q).

– Disjonction :

Dans SBVR, une disjonction est une opération logique binaire pour formuler la signification d'une décision logique de deux opérandes, qu'au moins l'un d'eux est vrai (i.e. p OU q).

– Équivalence :

L'équivalence dans SBVR est une autre opération logique binaire qui comprend une décision logique que parmi deux opérandes, le premier opérande est égal au second opérande (i.e. p est égal à q ; ou p est q).

– Implication :

Dans SBVR, une implication est une opération logique binaire utilisée pour formuler la signification d'une décision logique de deux opérandes, que le second opérande est vrai si le premier opérande est vrai (i.e. p alors q).

– Négation :

Dans SBVR, la négation est une opération unaire pour la décision logique d'un opérande qui formule le sens que l'opérande est faux, (i.e. PAS p).

- *Quantification :*

La quantification est une formulation logique qui utilise une variable pour spécifier la portée d'un concept. Il existe six types de quantifications de base définis dans SBVR, brièvement décrits ci-dessous :

– Quantification universelle :

Une quantification universelle est définie comme une référence pour chaque élément d'un domaine (i.e. "*each*").

– Quantification existentielle :

Dans SBVR une quantification existentielle représente une cardinalité minimale représentée par un élément, (i.e. 'at least').

– Au plus-n-Quantification :

Au plus-n-quantification montre la cardinalité maximale représentée par le nombre n représenté par un élément chose, (i.e. '*at most*').

– Au moins-n-Quantification :

Au moins-n-quantification montre une cardinalité minimale représentée par le nombre n représenté par un élément (i.e. 'at least').

– Quantification d'une plage numérique :

Une quantification de plage numérique dans SBVR présente une cardinalité minimale et maximale représentée par un élément (i.e. "*3 to 5 things*").

– Exactement n :

Cette quantification montre que la cardinalité exacte est représentée exactement comme n quantification représentée par un élément (i.e.'exactly').

Conclusion

Les différents éléments discutés ci-dessus jouent le rôle de briques permettant la représentation de la sémantique du langage naturel utilisé pour exprimer les règles métier. Si cette sémantique est facilement compréhensible par la machine, sa rédaction reste liée au niveau de maitrise du standard SBVR par les rédacteurs métier, ou de l'un des outils dédiés. Pour remédier à ce problème, nous avons proposé une méthode pour la génération automatique de cette sémantique selon le standard SBVR.

Le chapitre suivant présente une vue générale sur les différentes approches trouvées dans la littérature traitant la transformation des spécifications textuel en langage formel ou semi-formel, afin de positionner notre travail dans cette littérature.

3. Les règles métier : de l'informel au formel

Ce chapitre présente l'état de l'art des approches de transformation des spécifications textuelles en langages formels, afin de positionner notre travail dans la littérature.

Sommaire

Introduction

Diverses approches ont été proposées pour résoudre le problème de la transformation des spécifications écrites en langage naturel vers un langage plus formel. Ces approches sont presque toutes basées sur les techniques du traitement du langage naturel NLP (*Natural Language Processing*) [27]. Ce chapitre présente les principales méthodes trouvées dans la littérature, permettant la transformation des spécifications textuelles en langage formel d'une part, et ceux permettant la génération d'un modèle SBVR depuis des règles métiers écrites en langage naturel.

3.1. Les spécifications textuelles : du langage naturel vers le langage formel

La transformation des spécifications textuelles écrites en langage naturel vers un langage plus formel a été largement abordée dans la littérature. M.G. Ilieva [28] utilise des heuristiques pour extraire les classes candidates et leurs relations, en identifiant automatiquement le sujet, le prédicat et l'objet depuis des exigences textuelles, avant de les mapper à un modèle d'analyse orienté objet. C. Arora [29] génère le modèle de domaine à partir des exigences textuelles. H.M. Harmain [30], Sagar [31], Sharma [32] et B. A. Karaa [33] analysent les spécifications textuelles et génèrent un modèle de classe UML [34]. A. Al-Hroob [35] et S. Tiwari [36] extraient automatiquement des éléments des cas d'utilisation depuis des spécifications textuelles. F. Friedrich [37] propose une approche pour générer automatiquement des modèles BPMN à partir de texte écrit en langage naturel basé sur une combinaison d'outils NLP existants.

D'autres approches ajoutent un corpus à des fins d'apprentissage comme D. Sadoun [38] qui modélise le modèle de domaine à travers un processus de population d'ontologie en utilisant un corpus pour l'apprentissage, afin d'identifier les triplets dans les spécifications textuelles. Une autre méthode proposée par R. J. Kate [39] qui induit les règles de transformation des phrases écrites en langage naturel vers un langage formel spécifique, à travers l'association des pattern à des Template. L'approche est basée sur l'apprentissage des règles de transformation à partir de corpus de phrases associés à leurs équivalents en langage formel. Ils supposent qu'une grammaire analysable de manière déterministe pour le langage formel cible est disponible.

L'un des inconvénients majeurs de l'utilisation de ces méthodes est le fait que le corpus de formation n'est pas toujours disponible, et que les responsables métier ne sont pas toujours familiarisés avec les modèles (semi) formels ce qui limite leur intégration à la validation des modèles générés. Et c'est la raison pour laquelle que nous avons adopté le standard SBVR comme représentation intermédiaire entre les spécifications textuelles et n'importe quelle autre transformation. Cette transformation intermédiaire va nous permettre d'écrire les spécifications en un langage naturel compréhensible à la fois par les utilisateurs et les machines. Ainsi, les responsable métier peuvent participer à la validation du modèle généré avant toute transformation vers un autre modèle formel.

3.2. Les règles métier : du langage naturel au standard SBVR

M. Fernández [40] propose une approche pour identifier les déclarations des règles métier candidates à partir des descriptions textuelles libres écrites en espagnol; puis l'analyste métier décide leur validité, avant d'écrire manuellement les expressions SBVR valides.

M. Selway [41] présente une méthode pour générer des modèles formels de SBVR à partir des spécifications métier écrites en langage naturel, à travers l'apprentissage du vocabulaire depuis un glossaire écrit en SBVR-SE [15] (Structured English), avant d'analyser ces spécification en se basant sur le vocabulaire appris.

S. Roychoudhury [42] génère un modèle SBVR à partir du langage naturel. Les phrases et le modèle de domaine source préalablement créés sont utilisés pour apprendre les patterns des templates avant d'extraire les relations. De plus, les experts du domaine sont impliqués dans des séries de transformations, ce qui le rend semi-automatique. D'autre part, l'éditeur SBVR a besoin d'une connaissance préalable de la notation SBVR-SE pour modifier les suggestions ou ajouter des informations supplémentaires telles que les synonymes et les définitions des concepts.

I. S. Bajwa [43] présente une méthode pour générer les contraintes OCL à partir des spécifications écrites en langage naturel en utilisant le standard SBVR comme étape intermédiaire. Un modèle de classe UML préexistant est requis, et censé être cohérent avec le vocabulaire utilisé dans les contraintes textuelles.

T. Skersys [44] propose une approche hybride pour extraire les concepts généraux, les concepts verbaux et les déclarations SBVR d'un cas d'utilisation formel, améliorée plus tard par P.Danenas [45] en utilisant une approche basée sur la NLP au lieu de règles purement basées sur des modèles.

P. K. Chittimalli [46] propose une approche pour extraire le vocabulaire et les règles métier SBVR à partir des documents métier spécifiques à un domaine. Ils essaient d'identifier automatiquement les entités et les relations entre les concepts de nom à l'aide des heuristiques, avant de générer les règles métier. Cependant, les heuristiques définies sont en quelque sorte trop attachées au lexique comme dans l'exemple suivant « *if the statement contains 'if'* », alors que de nombreuses autres expressions peuvent être utilisées dans les expressions textuelles comme «*when*» ou même en utilisant uniquement des marques de ponctuation. De plus, autres éléments SBVR non pris en compte par l'approche proposée devraient être identifiés pour exprimer un modèle SBVR plus complet, comme la modalité, les quantifications et les formulations logiques, ainsi que d'autres spécifications sur les concepts extraits (e.g. les synonymes).

Le reste des approches trouvées dans la littérature procèdent de la même manière que celles décrites ci-dessus, avec quelques différences mineures. La **Table 1** présente un comparatif entre les différentes approches permettant la génération du modèle SBVR depuis des règles métier textuelles.

Table 1 Comparatif des différentes approches traitant les règles métier textuelles

Approche	Année	Entrée	Sortie	Transformation
JL.M-Fernández [40]	2008	free text descriptions	BR statements	Manual
M. Selway [41]	2015	Textual specification	BR statements	semi-automatic
R. Suman [42]	2017	legal English text	SBVR model	semi-automatic
I. S. Bajwa [43]	2017	Textual specification	BR statements, then OCL constraints	semi-automatic
T. Skersys [44]	2018	Use Case Diagram	General concept, verb concepts, and BR statements	semi-automatic
P. Danenas [45]	2020			
P. K. Chittimalli [46]	2020	Textual business rules	Entities, facts, and BR statements	Automatic
Our approach		Textual business rules	SBVR model: – **Terminological Dictionary**: noun concepts and verb concepts with their specifications such as definitions, synonyms, and generalization relations. – **BR statements**: semantic formulation such as modal formulation, atomic formulation, instantiation formulation, negation, quantification, implication, and conjunctions.	Automatic

La **Table 2** comporte la comparaison de ces approches au niveau de la façon avec laquelle le vocabulaire métier a été extrait depuis les règles métier textuelles.

Table 2 Comparatif des approches traitant le vocabulaire métier SBVR

Critères	**Chittimalli** [46]	**Suman** [42]	**Selway** [41]	**Bajwa** [43]	**Notre Approche**
Sortie	Vocabulaire et règles métier	Vocabulaire et règles métier	Règles métier	Vocabulaire et règles métier + OCL	Dictionnaire terminologique et Règles métier
Méthode	NLP	Apprentissage des patterns	NLP	NLP	NLP
Requis	_	Dictionnaire	Vocabulaire	Diagramme de classe	_
Identification des concepts	Auto	Semi	Semi	Semi	Auto
Synonymes	Non	Manuel	Manuel	Non	Auto
Définitions	Non	Manuel	Manuel	Non	Auto
Relation de Spécialisation	Non	Manuel	Manuel	Non	Auto

Les approches discutées ci-dessus se focalisent sur l'extraction de la sémantique à partir du texte, mais ignorent d'autres problèmes conduisant à des incohérences, résultant notamment du fait que les règles métier sont généralement écrites par différentes parties prenantes. L'utilisation de plus d'une expression pour désigner un même concept est l'un des problèmes qu'on a essayé de traiter dans cette thèse, et qui caractérise notre contribution par rapport aux autres méthodes proposées.

La section suivante présente et compare les différentes approches existantes pour l'identification des synonymes à partir des spécifications textuelles.

3.3. La similarité sémantique entre les concepts métier

Malgré les problèmes qui peuvent résulter de l'existence de plusieurs expressions ayant la même signification, l'identification des synonymes est négligée dans la plupart des approches traitant les spécifications textuelles. Dans cette section, nous donnerons un aperçu des approches existantes traitant l'identification des synonymes à partir des spécifications écrites en langage naturel.

Malgré l'énorme effort requis, Selway [41] marque manuellement les termes synonymes lorsque il essaie de transformer les spécifications métier textuelles en modèles formels. De la même manière, Suman [42] transforme le texte juridique écrit en anglais vers le standard SBVR, et Bajwa [43] génère les contraintes OCL à partir du langage naturel. Alors que Chittimalli [46]

néglige l'extraction de la relation de similitude sémantique entre les termes quand il essaye d'identifier automatiquement les entités et les relations à partir du règles textuelles.

Dans d'autres approches, il existe une différence importante dans la manière dont les synonymes sont identifiés. Gruhn [47] a créé un catalogue de synonymes et d'antonymes pour encourager la détection des patterns d'erreur qui peuvent être trouvés dans les chaînes de processus événementielles. Awad [48] a proposé une approche automatisée pour interroger un référentiel de modèles de processus métier, dans lequel le sens le plus commun a été sélectionné depuis le dictionnaire WordNet [49].

Quelques approches ont tenté d'extraire les synonymes en utilisant un dictionnaire (comme WordNet [49]) sans explication ni quel algorithme a été utilisé, ni si le contexte a été pris en considération dans la détection des synonymes ou non, comme Danenas [45] et Skersys [44] qui ont extraient le vocabulaire et les règles métier SBVR depuis des diagrammes de cas d'utilisation UML; ou Friedrich [37] qui a présenté une approche automatique pour générer des modèles BPMN (Business Process Model and Notation) à partir du texte écrit en langage naturel; ou Zhang [50] qui a combiné diverses fonctionnalités pour récupérer les liens exigence-code.

Ehrig [51] a proposé une approche pour la détection (semi-) automatique de synonymes et d'homonymes de noms depuis des éléments de processus en utilisant un vocabulaire préexistant basé sur une ontologie obtenue à partir d'un profil UML. Il a combiné trois mesures de similitude : syntaxique (compare le nombre de caractères en communs), linguistique (le nombre de sens obtenus de WordNet) et structurelle (l'emplacement des concepts).

Pittke [52] a proposé une technique pour détecter et résoudre les ambiguïtés lexicales dans les modèles de processus textuels causés par des homonymes et des synonymes en combinant des scores calculés à partir des traductions dans différentes langues. Pour trouver des termes avec des sens similaires, il a utilisé une approche multilingue en se servant des graphes basés sur la base de connaissances multilingue BabelNet [53] avec XMeans clustering [54].

Dalpiaz [55] a tenté de détecter l'ambiguïté terminologique dans les 'user stories' en calculant la similitude entre les termes puis entre leur contexte. Pour cette raison, Cortical.io [56] a été utilisé, qui est considéré comme un outil très puissant basé sur l'intelligence artificielle, avec une vitesse de traitement élevée. Cortical.io s'appuie sur des matrices créées à partir d'une grande collection de sites web pour calculer la similitude sémantique.

La **Table 3** présente un comparatif entre les approches traitant les synonymes dans les spécifications textuelles.

Table 3: Tableau comparatif des approches traitant les synonymes dans les spécifications textuelles

Approche	Entrée	Sortie	Identification des Synonymes
P. K. Chittimalli [46]	Règles métier textuelles	SBVR: Entités, faits, et règles	Non
R. Suman [42]	Texte en anglais	Modèle SBVR	Manuel
I. S. Bajwa [43]	Spécifications textuelles	Contraintes OCL	Manuel
M. Selway [41]	Spécifications métier textuelles	Model formel	Manuel
V. Gruhn [47]	Event-Driven Process Chains	Patterns erronées	Manuel
P. Danenas [45]	Diagramme des cas d'utilisations	SBVR : vocabulaire et règles métier	Entièrement basé sur les résultats du dictionnaire (méthode n'est pas citée)
T. Skersys [44]	Diagramme des cas d'utilisations	SBVR : vocabulaire et règles métier	Entièrement basé sur les résultats du dictionnaire (méthode n'est pas citée)
F. Friedrich [37]	Texte en langage naturel	Modèle BPMN	Entièrement basé sur les résultats du dictionnaire (méthode n'est pas citée)
Y. Zhang [50]	Exigences textuelles	Liaisons exigences-code	Entièrement basé sur les résultats du dictionnaire (méthode n'est pas citée)
A. Awad [48]	Business Process model query	List de modèles de process	Entièrement basé sur les résultats du dictionnaire (le sens le plus en commun)
M. Ehrig [51]	Business Process models	Similarité entre modèles de processus métier	Semi-automatique : exige un vocabulaire pré exidtant basé sur l'ontologie
F. Pittke [52]	Model de processus textuels	Les termes homonymes et synonymes	Automatique : combine les scores calculés depuis la traduction en diffèrent langages en utilisant BabelNet
F. Dalpiaz [55]	User stories	Les termes synonymes proches	Automatique : calcule la similarité entre les termes puis entre leurs contextes en utilisant Cortical.io
Notre méthode	Règles métier en langage naturel	Les termes Synonymes & abréviation/expansion enregistrés selon le standard SBVR.	Automatique : utilise le NLP et un knowledge-based algorithme raffiné en utilisant des heuristiques.

On peut conclure qu'il n'existe pas de méthode unique convenue pour extraire les synonymes du texte qui est valable pour tous les objectifs ; mais nous sommes confrontés à une tâche non triviale qui diffère selon l'objectif et le contexte.

Notre objectif est d'extraire les termes ayant la même signification, mais avec des expressions différentes. Pour cette raison, des heuristiques ont été utilisées avec le NLP sans intervention humaine. Les résultats obtenus ont été stockés dans un fichier XMI selon le standard SBVR pour faciliter son intégration dans les approches déjà existantes.

Conclusion

Malgré l’intérêt donné à la transformation des spécifications textuelles vers la spécification SBVR, les approches ont tendance à se concentrer sur l'identification des objets et leurs relations explicites. Alors que les informations implicites cachées dans les spécifications complexes sont généralement négligées. De plus, d'autres informations devraient être identifiées pour augmenter la compréhension et l'utilité du vocabulaire extrait, en particulier : les définitions et les synonymes [57]. Cependant, à notre connaissance, aucune approche n'a généré un modèle SBVR plus complet à partir d'énoncés textuelles des règles métier sans se servir des ressources externes comme on le fait dans nos contributions.

Nos contributions ciblent à la fois la génération du dictionnaire terminologique enrichi avec des spécifications supplémentaires pour chaque concept identifié d’une part, et les règles métier avec leurs formulations logiques d’autre part, en utilisant le traitement du langage naturel. Notre modèle cible fournit une signification plus complète à chaque règle métier avant de présenter le résultat dans un format MS-Word compréhensible par les utilisateurs, avec leur sémantique sauvegardée dans un fichier XMI compréhensible par les machines. Ces deux types de fichiers sont considérés comme une bonne alternative pour combler les lacunes entre les analystes métier et les experts métier dans les entreprises.

Contributions

Sommaire :

4. La génération automatique du vocabulaire métier

Dans ce chapitre, nous présentons une approche automatisée pour générer le vocabulaire métier à partir des règles métier textuelles. Notre approche se distingue des travaux existants dans la mesure où elle permet l'extraction du dictionnaire terminologique comme décrit par le standard SBVR, afin de fournir une signification standardisée et plus complète pour chaque concept. Pour atteindre cet objectif, un traitement approfondi du langage naturel a été utilisé afin d'extraire non seulement une liste simple de termes et de relations, mais également des spécifications supplémentaires et des informations implicites.

Avec un résultat satisfaisant, notre approche a prouvé sa capacité à générer automatiquement le dictionnaire terminologique SBVR à partir d'un grand nombre de règles métier écrites en langage nature.

Sommaire

Introduction

Selon le Manifeste des règles métier lancé par le Business Rules Group [3] (article 3.1): "*Les règles reposent sur des faits, et les faits reposent sur des concepts exprimés par des termes*". Autrement dit, les termes agissent comme des blocs de construction pour créer des déclarations textuelles de règles métier [4]. Ce manifeste insiste également sur le fait que les règles sont atomiques, bien formées et rédigées en utilisant le vocabulaire métier. Pour cette raison, de nombreuses approches font appellent à des documents préexistants pour traiter les spécifications écrites en langage naturel. Ce genre de documents est généralement appelé «Domain Specific Information» (DSI) [27]. Il est souvent sollicité lors du traitement du langage naturel avec lequel sont écrites les spécification textuelles comme dans [41] [58] [59]. Vue le grand nombre de règles métier utilisées au sein des entreprises, la génération manuelle (voire la maintenance) du vocabulaire métier prend du temps et souvent sujette aux erreurs. Et c'est la raison pour laquelle, un grand intérêt a été accordé au cours de la dernière décennie pour rendre cette génération automatique.

Dans la littérature, l'extraction automatique de la terminologie depuis les spécifications textuelles repose généralement sur l'identification des termes et des faits en générant souvent une signification superficielle. La terminologie métier est généralement présentée comme un modèle de glossaire, de définition ou de domaine [27]. Elle est souvent difficilement manipulée par les non-informaticiens vue son format formel, ou impossible de l'intégrer dans des systèmes hétérogènes en raison de son format non standardisé. Cependant, le vocabulaire spécifique à un domaine - fréquemment appelé « modèle conceptuel» [60], ou «vocabulaire métier» [25] - devrait normalement être formellement défini et accompagné de descriptions textuelles et de synonymes [57].

Dans ce chapitre, nous présentons une approche [61] [62] pour générer automatiquement le vocabulaire métier à partir d'énoncés de règles métier écrites en langage naturel, avec une signification plus complète pour chaque concept extrait, tout en gardant à l'esprit cinq objectifs :

1. Aider la '*speech Community*' à adopter un vocabulaire métier bien formé.
2. Normaliser le vocabulaire métier pour encourager la réutilisation et l'interopérabilité.
3. Augmenter l'utilité du vocabulaire métier en fournissant un dictionnaire terminologique plus complet.
4. Encourager l'automatisation dans les premières phases du processus de développement en générant le vocabulaire métier en tant qu'élément atomique sur lequel sont fondées les règles métier.
5. Combler le fossé entre les experts métier/IT et la machine.

Pour tous ce qui précède, nous proposons une méthode pour la génération automatiquement du dictionnaire terminologique conformément au standard SBVR. Ce dernier a été choisi comme modèle cible, étant donné sa richesse en termes d'expressivité sémantique, sans parler de sa compréhensibilité à la fois par les intervenants non spécialistes, ainsi que les machines. La notation

SBVR-Structured English (SBVR-SE) a été adoptée pour présenter ce dictionnaire terminologique pour sa compréhensibilité et sa représentation formelle par rapport aux autres langages contrôlés [63].

Les contributions de ce chapitre sont les suivantes :

1. La génération automatique d'un modèle cible conforme au méta-modèle du dictionnaire terminologique décrit par le standard SBVR en utilisant la transformation entre modèles.
2. Un traitement du langage naturel approfondi pour extraire automatiquement la sémantique des règles métier textuelles.
3. L'identification automatique des concepts et leurs spécifications supplémentaires telles que : les définitions (intentionnelles et extensionnelles), synonymes et concepts généraux.

L'approche a été appliquée sur plus de 150 déclarations de règles métier et a prouvé - avec une précision élevée - sa capacité à extraire des faits (implicites et explicites) et des noms (généraux ou propres, simples ou composés) associés avec des spécifications supplémentaires.

En général, les règles métier sont classées en trois grands types : (i) les règles métier exprimant des contraintes ; (ii) les règles métier testant une condition avant de déclencher un autre événement, (iii) et les règles métier générant de nouvelles informations, comme l'exécution de calculs [23].

Dans cette contribution, notre objectif est d'extraire le vocabulaire métier avec des spécifications supplémentaires. Ainsi, nous nous sommes concentrés plus sur les règles métier fournissant des informations pertinentes sur les concepts et leurs relations (règles métier de type (i) et (ii)). Et c'est la raison pour laquelle, les règles métier exprimant des calculs ont été ignorées dans cette approche, en raison de sa faible contribution à l'enrichissement du dictionnaire. Les règles métier contenant des contraintes de temps sont également ignorées dans notre approche, car elles demandent un traitement spécifique. Ces types de règles ignorées peuvent être facilement intégrées dans notre approche, mais cela se fera au détriment de la clarté de l'approche, et elles seront prises en compte dans nos futurs travaux.

Le reste de ce chapitre est structuré comme suit : premièrement, nous décrivons en détail notre approche pour la génération automatique du dictionnaire terminologique selon le standard SBVR. Puis, nous présentons la méthode d'évaluation ainsi que les résultats obtenus en termes d'identification des éléments du dictionnaire terminologique. Ensuite, nous examinons les résultats obtenus, avant de dessine les limites de l'approche proposée ainsi que les orientations pour les futurs travaux. En fin, nous terminons par une conclusion.

4.1. Présentation de l'approche

Notre objectif est d'automatiser la génération du dictionnaire terminologique SBVR à partir des déclarations de règles métier textuelles, en plus des spécifications supplémentaires pour chaque concept identifié. Pour cette raison, nous avons utilisé à la fois une transformation Text-2-Model (T2M) et une autre transformation Model-2-Model (M2M) comme suit :

1. *La transformation T2M* : premièrement, nous avons analysé la structure grammaticale de plusieurs déclarations des règles métier textuelles, avant de définir un métamodèle générique pour les clauses composant ces déclarations. Ensuite, chaque clause respectant ce métamodèle est automatiquement identifiée, en utilisant le traitement du langage naturel.
2. *La transformation M2M* : les clauses (composant les déclarations des règles métier) respectant notre métamodèle générique sont utilisé pour produire le dictionnaire terminologique respectant le métamodèle SBVR.

La **Figure 5** Notre modèle de transformation montre les étapes de transformation des instructions textuelles (source) au métamodèle SBVR (cible) en utilisant (comme intermédiaire) notre métamodèle générique de clauses.

.

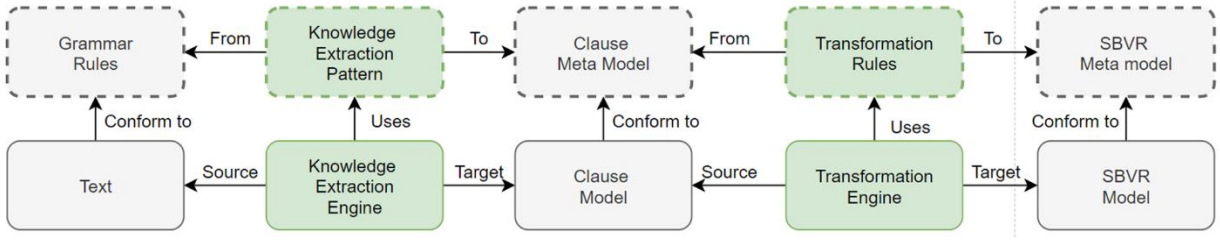


Figure 5 Notre modèle de transformation pour la génération du dictionnaire terminologique SBVR

Notre approche commence par un pipeline NLP pour marquer les parties du discours (Part of Speech (POS) tagging), et pour générer l'arbre des dépendances grammaticales de chaque règle. Ensuite, nos algorithmes de recherche de patterns sont utilisés pour extraire notre modèle générique de clauses composant chaque règle. Puis, les clauses sont disséquées pour enlever les conjonctions, avant d'identifier les noms et les verbes avec leurs spécifications selon la norme SBVR. Enfin, notre modèle cible représentant le dictionnaire terminologique de SBVR (**Figure 6**) est généré.

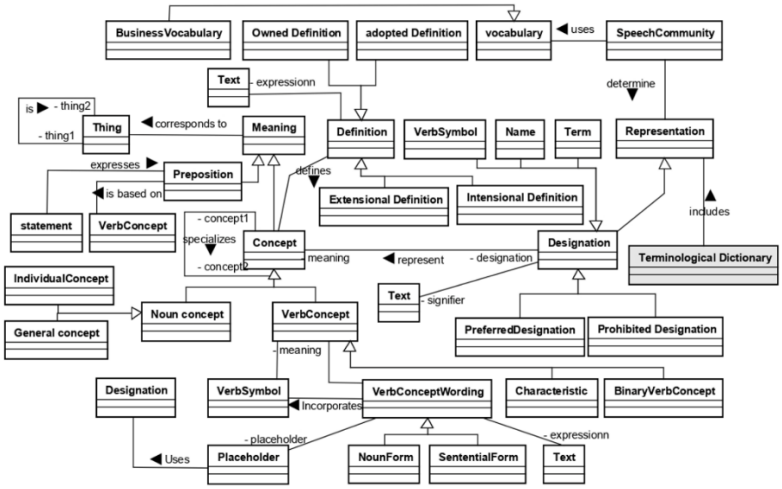


Figure 6: Le métamodèle cible représentant le dictionnaire Terminologique SBVR

Comme décrit dans la Figure 7 Notre approche pour l'identification du dictionnaire terminologique SBVR, notre approche commence par insérer les règles une par une dans le pipeline NLP pour la tokenisation, l'étiquetage (POS), et la génération de l'arbre des dépendances grammaticales. Les clauses sont ensuite extraites et disséquées pour enlever les conjonctions et identifier les éléments (noms et verbes) les plus atomiques ou implicites. Ensuite, les clauses respectant notre métamodèle sont identifiées. Et finalement, chaque concept est extrait et épicé avec des spécifications supplémentaires telle que les définitions, la désignation préférée, le concept général, etc. Une fois ces étapes terminées, les éléments extraits sont mappés aux éléments du dictionnaire terminologique SBVR, et présentés en utilisant la notation SBVR-SE.

Pour des raisons d'espace, nous avons choisi de regrouper nos algorithmes d'extraction de patterns en 3 organigrammes, à la place de les représenter en pseudo-code.

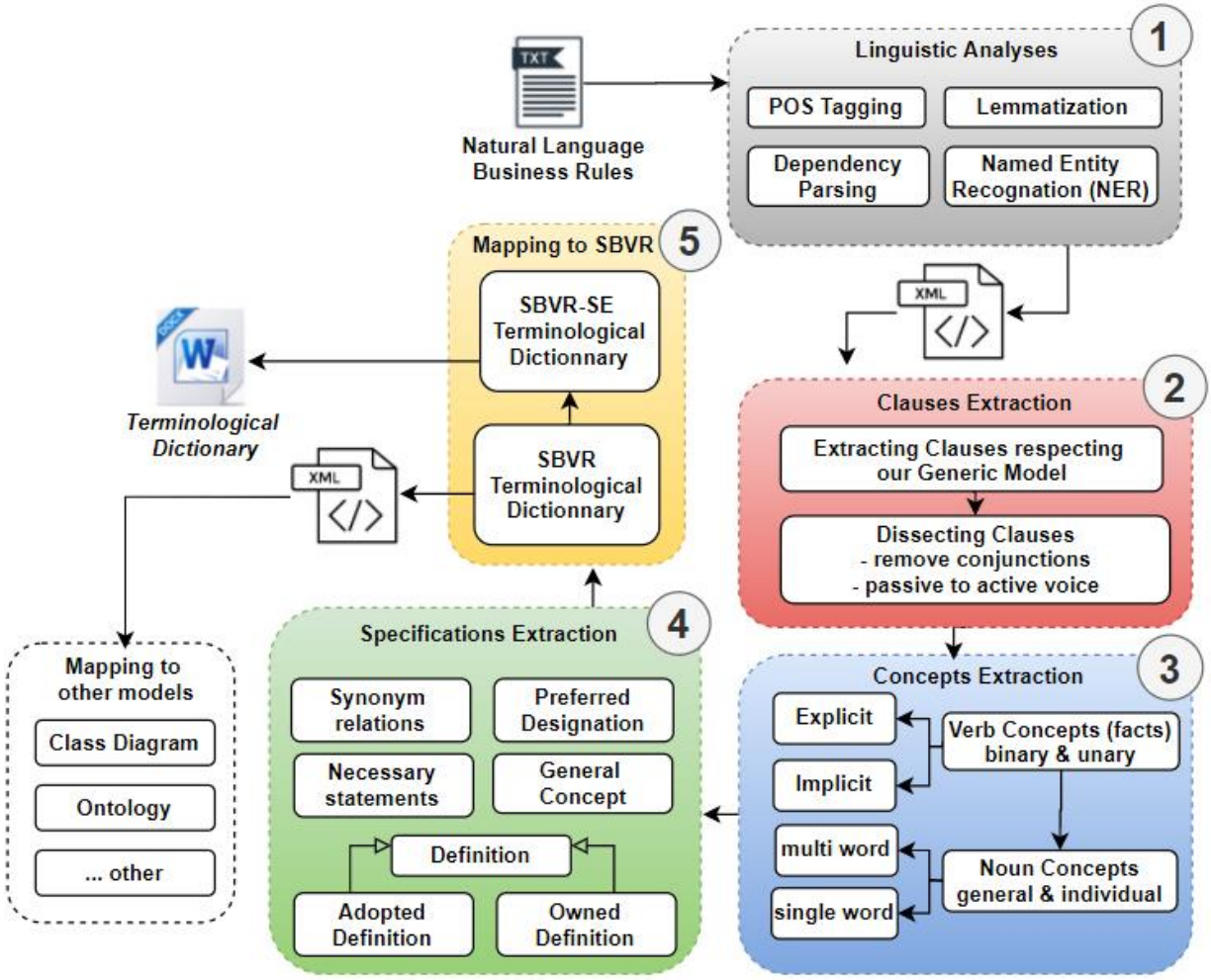


Figure 7 Notre approche pour l'identification du dictionnaire terminologique SBVR

4.1.1. L'analyse l'linguistique

Les règles métier sont insérées dans le pipeline de notre approche sous forme de document textuel. Ensuite, chaque règle est traitée automatiquement et individuellement. Pour les analyses syntaxiques et lexicales, nous avons adopté l'outil Stanford CoreNLP [64] comme l'un des outils modernes, régulièrement mis à jour avec une analyse de haute qualité. Malgré que cet outil nous ait permis de marquer et d'analyser les règles textuelles ; nous avons remarqué que la précision de cet outil diminue en fonction du niveau de complexité de la structure grammaticale de la phrase utilisée. Pour cela, nous avons commencé par réduire cette complexité et enlever la première partie de chaque déclaration si elle commence par «It is <modality> that…», comme montré dans la Figure 8 Exemple de réduction de la complexité grammaticale d'une .

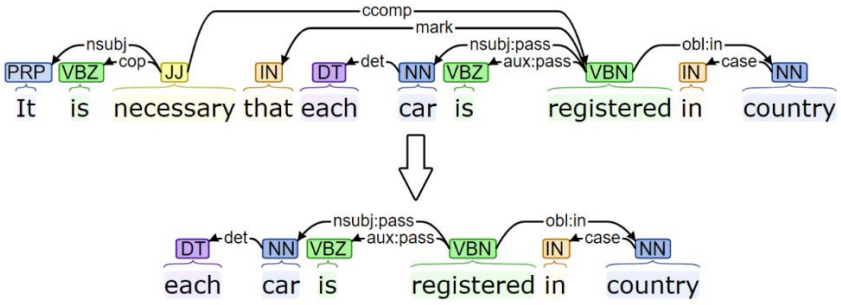


Figure 8 Exemple de réduction de la complexité grammaticale d'une règle métier textuelles

4.1.2. L'extraction des clauses

Les phrases sont des constructions grammaticales composées d'au moins une clause [65] contenant des concepts. Ainsi, après le POS-tagging et la génération de l'arbre des dépendances grammaticales, chaque règle est analysée séparément pour extraire les clauses explicites et implicites respectant notre métamodèle générique pour les clauses. Ensuite, la complexité de chaque règle est réduite à travers leur division en sous clauses pour permettre l'extraction des concepts.

Les algorithmes et les patterns utilisés pour identifier les clauses explicites et implicites sont détaillés ci-dessous.

- *Les clauses explicites*

Pour extraire les clauses explicites composant les règles métier textuelles, nous avons décidé de créer un métamodèle générique (Figure 9) pour ces clauses. En suit, les clauses respectant ce métamodèle vont être identifier en cherchant des patterns via un algorithme.

Avant de définir notre métamodèle générique, nous avons analysé la structure des différentes formes dans lesquelles les règles métier textuelles peuvent être exprimées, discutée dans différentes sources spécialisées dans ce domaine [4] [23] [66] [67].

.

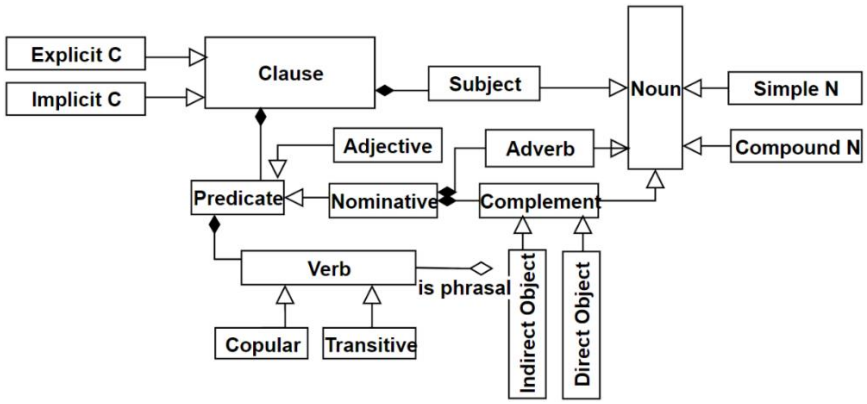


Figure 9 Notre méta-modèle générique des clauses composant les règles métier textuelles pour la génération du dictionnaire terminologique

Une clause est composée d'au moins un sujet et d'un prédicat. Les prédicats sont composés d'un verbe et d'un adjectif ou d'un complément sous forme d'objet direct (et indirect). Les verbes peuvent être des verbes transitifs s'ils ont un sujet et un objet direct, ou bien des verbes copulaires (verbe « être »). De plus, les verbes sont marqués comme des verbes prépositionnels s'ils sont liés à une préposition. D'autres part, les clauses sont classées en deux catégories : explicites ou implicites. Dans le reste de cette section, nous allons essayer de clarifier nos algorithmes pour l'extraction de notre métamodèle générique des clauses depuis les règles métier textuelles.

Notre algorithme de la Figure 10 Notre algorithme pour extraire les clauses explicites des parcours l'arbre des dépendances grammaticales de chaque règle pour identifier les patterns exprimant des clauses qui respectent notre métamodèle générique. Tous les patterns sont détaillés par des exemples dans la **Table 4**.

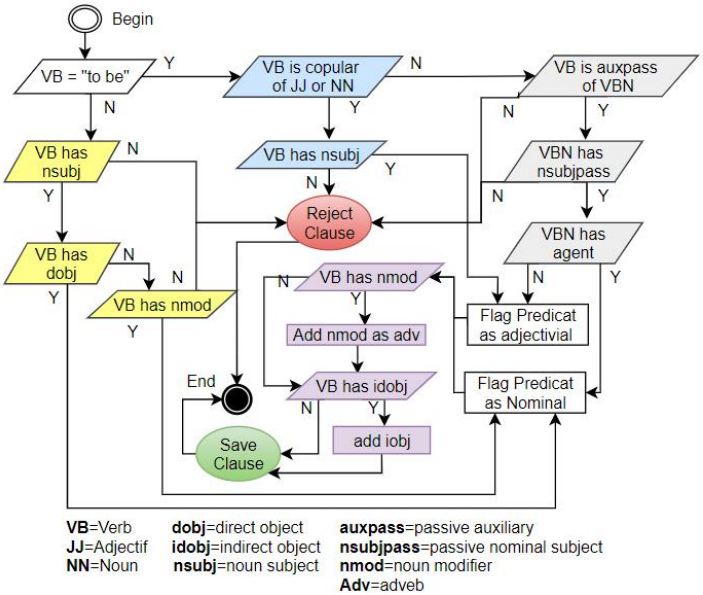


Figure 10 Notre algorithme pour extraire les clauses explicites des règles métier textuelles

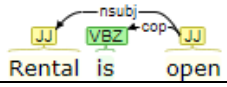
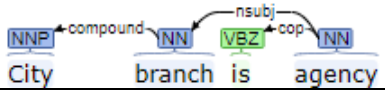
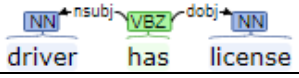
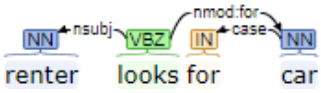
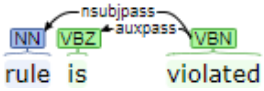
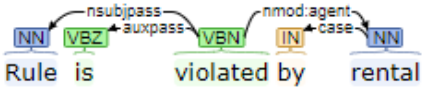
Premièrement, notre algorithme vérifie si le verbe principal est un verbe copulaire (verbe être) d'un adjectif (JJ) ou d'un nom (NN). Ensuite, si le verbe a un sujet (nsubj), la clause est ajoutée en tant que nouvelle clause avec un prédicat adjectival, sinon elle est considérée comme une clause mal formée et doit être rejetée. En revanche, si le verbe n'est pas un verbe copulaire, alors il est considéré comme un verbe transitif, pour lequel il faut trouver un sujet et un objet direct pour créer une nouvelle clause avec un prédicat nominal, sinon la clause est rejetée.

L'outil d'analyse linguistique utilisé (Stanford CoreNLP) a un cas particulier : si le verbe est un verbe prépositionnel (lié à une préposition), l'objet direct est marqué comme un modificateur de nom (le pattern P4 de la Table 4 est un exemple). Dans ce cas, pour les verbes prépositionnels, notre algorithme recherche un modificateur de nom au lieu d'un objet direct.

En ce qui concerne le traitement de la voix passive, notre algorithme cherche l'auxiliaire utilisé pour exprimer la voix passive (auxpass) avec son sujet nominal (nsubjpass) pour les verbes trouvés en '*past participle*'. Si un agent a été identifié, alors une nouvelle clause est ajoutée avec un prédicat nominal, sinon un prédicat adjectival sera affecté à la nouvelle clause.

Enfin, d'autre éléments sont ajoutés (s'ils en existent) avant de sauvegarder la clause, comme les adverbes et les objets indirects.

Table 4 Nos patterns pour identifier les clauses explicites

	Pattern	**Exemple**
La voix active		
P1	Un adjective (JJ) ayant un sujet (subj) et un verb copulaire (cop)	
P2	Un nom (NN) ayant un subjet (subj) et un verb copulaire (cop)	
P3	Un verbe (VBZ) ayant un sujet (subj) et un complément d'objet direct (dobj)	
P4	Un verbe prépositionel (VBZ) ayant un sujet (subj) sans complément d'obet dirrect, relié à un modificateur de nom (NN) en utilisant une préposition (IN)	
P5	Un verbe en 'Past Participle' (VBN) ayant un sujet (nsubjpass) et un auxilaire (auxpass) concernant la voix passive sans agent	
La voix passive		
P6	Un verbe en 'Past Participle' (VBN) ayant un sujet (nsubjpass) et un auxilaire (auxpass) concernant la voix passive en plus d'un agent. Il sera converti à: Agent + VBN + nsubjpass	 Il sera concerti à: "*Rental Violate Rule*"

- *Les clauses implicites*

Pour identifier les clauses implicites, nous avons essayé d'une part, de disséquer les clauses composées, en clauses plus simples contenant un sujet et un complément d'objet direct ou un adjectif. Pour cette raison, on a essayé de casser les conjonctions pour chercher les faits les plus atomiques (cachés) de chaque clause (voir un exemple dans la Figure 11 Exemple de dissection des clauses composées).

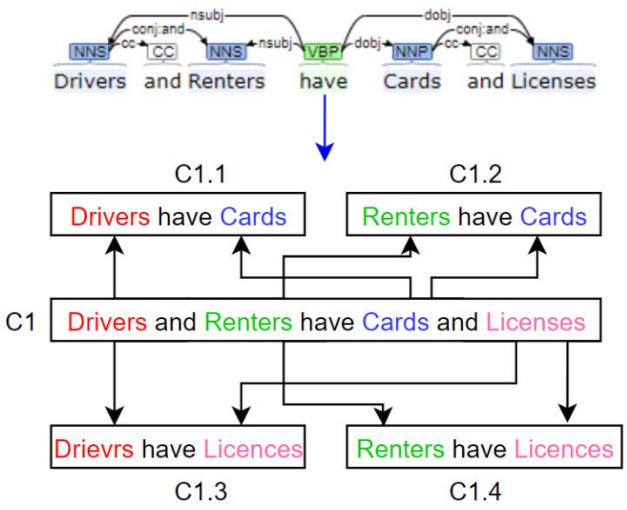


Figure 11 Exemple de dissection des clauses composées

D'autre clauses implicites sont extraites depuis les noms composés selon l'algorithme de la Figure 12 Algorithme d'extraction des clauses implicites depuis les noms composés, en cherchant les patterns de la Table 5 dans chaque règle métier. Par exemple, '*customer has age*' est généré depuis le nom composé '*customer age*'.

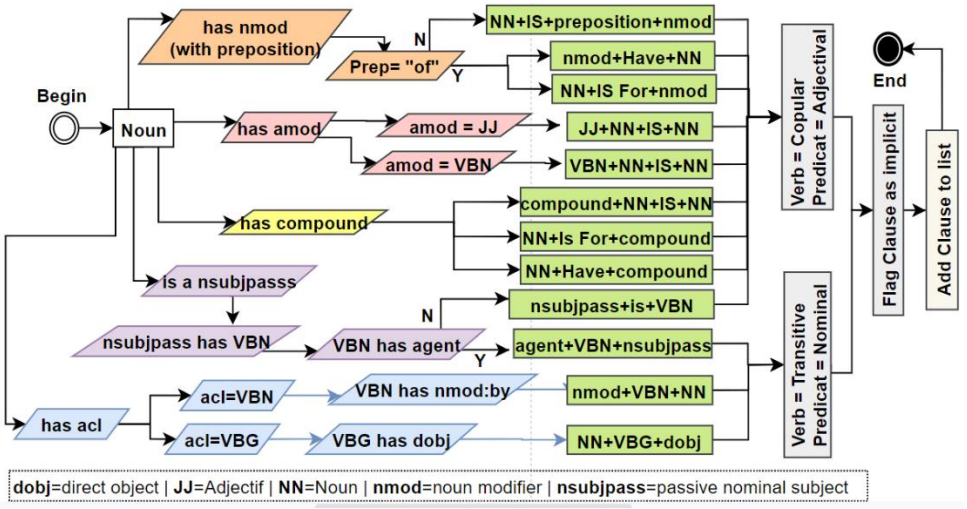
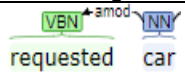
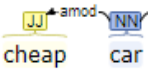
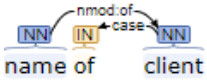
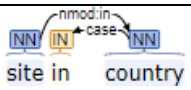
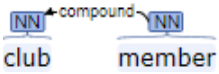
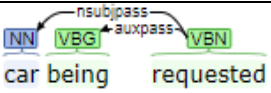
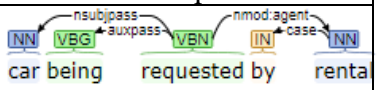
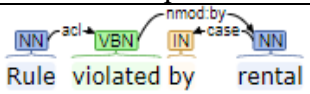
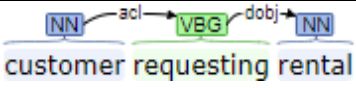


Figure 12 Algorithme d'extraction des clauses implicites depuis les noms composés

Cet algorithme est utilisé pour identifier les patterns des clauses implicites, depuis les noms composés exactement de deux mots, vue la difficulté de décider quelle clause implicite est cachée dans les noms composés de plus de deux mots. Pour illustrer cela, à partir du mot composé '*round-trip car movement*', nous pouvons générer deux clauses implicites '*round-trip is a car movement*' et '*round-trip car is a movement*'. Nous ne pouvons sélectionner la bonne proposition que si on les compare avec toutes les déclarations qui existent (son contexte). Pour cette raison, nous avons ajusté notre algorithme pour qu'il puisse prendre en charge aussi l'extraction de clauses implicites depuis les mots composés de plus de deux mots, en identifiant le chevauchement entre les noms de plusieurs mots. Par exemple, nous avons un chevauchement dans les deux noms composés '*location penalty charge*' et '*penalty charge*', nous pouvons donc en déduire que le premier est une spécification du deuxième. Ainsi, la clause '*location penalty charge is penalty charge*' sera extraite au lieu de '*location penalty is charge*' ou bien '*location is penalty charge'*.

Table 5 Nos patterns représentants les clauses implicites à extraire depuis les noms composés

	Noms composés	Exemple
P7	Nom (NN) ayant un verbe en '*past participle*' (VBN) comme modifier **La clause extraite:** NN + "is" + VBN	 "Car is requested"
P8	Nom (NN) ayant un adjectif (JJ) comme modifier **La clause extraite:** JJ + NN + "is" + NN	 "Cheap car is car"
P9	Nom (NN1) relié à un modifier (NN2) en utilisant la proposition "of" **La clause extraite:** NN1 + "is for" + NN2 NN2 + "has" + NN1	 "Name is for client" "Client have Name"
	Nom (NN1) relié à un modifier (NN2) en utilisant autre prépostion que "of" **La clause extraite:** NN1 + "is" + Prep + NN2	 "Site is in country"
P10	Noms en relation de composition **La clause extraite:** NN1 + NN2 + "is" + NN2 NN2 + "is for" + NN1 NN1 + "has" + NN2	 "club member is member" "Member is for Club" "Club have Member"
P11	Verbe en "past participle" (VBN) ayant un sujet (nsubjpass) et un auxilaire (Auxpass) exprimant la viox passive sans agent. **La clause extraite:** nsubjpass + auxpass + VBN	 "Car is requested"
P12	Verbe en "past participle" (VBN) ayant un sujet (nsubjpass) et un auxilaire (Auxpass) exprimant la viox passive avec un agent. **La clause extraite:** Agent + VBN + nsubjpass	 "Rental request car"
P13	Nom (NN1) ayant une clause adjectival composée d'un verbe en "past participial" (VBN) relié à un modifier (NN2) en utilisant la preposition "by" **La clause extraite:** NN2 + VBN + NN1	 "Rental violate rule"
P14	Nom (NN1) ayant une clause adjectival composée d'un verbe en « present participle » (VBG) ayant un complément d'objet direct (dobj) **La clause extraite:** NN1 + VBG + Dobj	 "Customer request rental"

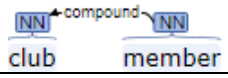
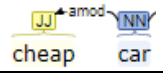
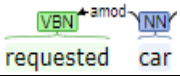
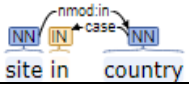
4.1.3. L'extraction des concepts

Selon le métamodèle du dictionnaire terminologique défini par le standard SBVR, il existe deux types de concepts : concept de nom (général ou individuel) ou concept de verbe (unaire ou binaire). L'extraction de ces concepts se fera comme suit :

- *Concepts de noms*

Les noms jouant le rôle d'un sujet d'un complément d'objet direct sont d'abord identifiés et marqués comme des concepts constitués d'un seul mot. Si ces noms ont des modificateurs, alors ils sont ajustés en ajoutant ces modificateurs pour créer des noms composés. Pour cette raison, nous avons défini quatre patterns (voir Table 6) à être identifiés en utilisant l'algorithme de la Figure 13 Algorithme d'identification des patterns des noms simples et composés.

Table 6 Nos patterns exprimant les noms composés

	Pattern	Exemple
P15	Noms en relation de composition.	
P16	Nom (NN) ayant un adjectif (JJ) comme modificateur adjectial (amod).	
P17	Nom (NN) ayant un verbe en '*past participle*' (VBN) comme modificateur adjectivial (amod).	
P18	Nom NN1 ayant un autre nom NN2 comme modificateur nominal (nmod) qui utilise une proposition (IN).	

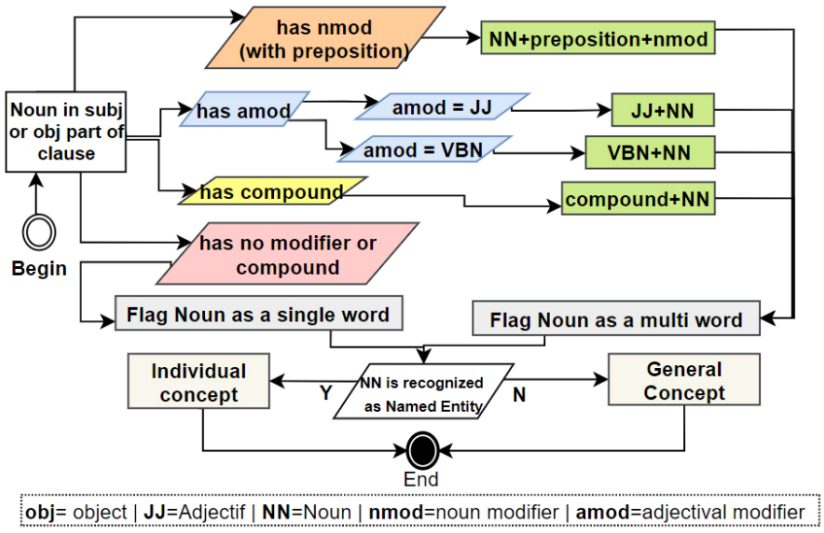


Figure 13 Algorithme d'identification des patterns des noms simples et composés

Comme on peut le constater, notre algorithme vérifie si le nom identifié a un modificateur nominal (nmod) qui utilise une préposition (IN) (e.g. '*site in country*'), ou bien si le nom est en relation de composition avec un autre nom (e.g. '*credit card*'). De plus, l'algorithme cherche les modificateurs adjectivaux (amod) ayant un format adjectival (JJ) (e.g. '*cheap car*') ou en forme de '*past participle*' (VBN) (e.g. '*requested car'*). Comme dernière étape de l'identification des noms composés, si le concept de nom est reconnu par l'outil Stanford CoreNLP comme une entité nommée (NER or Named Entity Recognition), alors ce nom est enregistré en tant que concept individuel, sinon il est enregistré en tant que concept général.

- *Concepts de verbe (faits)*

Comme décrit dans la documentation SBVR [15], les concepts verbaux peuvent être trouvés sous leur forme '*Sentential form*' (e.g. '*driver has name*') ou sous forme de '*Nom form*' (e.g. '*driver name*'). Puisque les clauses explicites et implicites qui composent chaque règle sont déjà extraites, nous pouvons facilement extraire les concepts verbaux dans ses deux formats comme suit: les clauses explicites sont des concepts verbaux en '*Sentential form*', tandis que les clauses implicites seront des concepts verbaux en '*noun form*'.

Voici un exemple des concepts composant une seule règle métier, identifiés par notre approche (17 concepts verbaux (implicites et explicites) ont été extraits englobant 14 concepts nominaux) :

- **Règle métier** : "*It is necessary that the international car movement is a one-way car movement that includes a sending branch that is managed by a local area that is located at a EU-Rent site that is included in an operating country that is not the operating country that contains the EU-Rent site that is location of the local area that manages the receiving branch of the car movement*".
- **Concepts de verbe**: "location is of local area ; sending branch is branch ; operating country include EU-Rent site ; receiving branch is of car movement ; EU-Rent site is location ; international car movement is car movement; one-way car movement is car movement; local area is area ; local area manage receiving branch ; local area is located at EU-Rent site ; local area manage sending branch ;operating country contain EU-Rent site ; one-way car movement include sending branch ; international car movement is one-way car movement ; operating country is country ; EU-Rent site is site ; receiving branch is branch"
- **Concepts de nom**: "location ; local area ; area ; sending branch ; branch ; operating country ; country ; EU-Rent site ; site ; receiving branch ; car movement ; movement ; one-way car movement ; international car movement".

4.1.4. Extraction des spécifications supplémentaires

Dans cette section, nous allons présenter notre proposition pour enrichir les concepts extraits par des spécifications supplémentaires telles que leurs définitions et leurs synonymes.

- *Les Définitions*

– Définitions propres

La définition propre (*owned definition*) est affectée aux concepts en interne par le '*speech community*' en utilisant le vocabulaire métier. Ce qui suit présente notre préposition pour automatiser l'identification et l'affectation d'une définition en utilisant la seule ressource dont nous disposons, à savoir les règles métiers.

Les règles ayant une clause indépendante (clause sans condition) avec un prédicat adjectival seront affectées en tant que définition aux concepts de nom situés dans la partie 'sujet' de la clause, selon les patterns de la Table 7 (Les définitions propres pour les concepts verbaux ne sont pas prises en charge par notre approche).

Table 7 Nos patterns concernant les 'Owend Definition' des concepts de nom

	Pattern	**Exemple**
P19	**Définition Extensionnelle:** Sujet + verbe copulaire + Predicat 1 + "Or" + Predicat 2 …	La règle suivante '*Rental is advance rental or walk-in rental'* est une definition extensionnelle du concept '*Rental*'
P20	**Définition Intentionnelle :** Sujet + verbe copulaire + (NN [+clause adjectival])	La règle suivante '*Driver is a person [who is authorized for rental'* est une définition intentionnelle du concept '*Rental*'

– Définition adoptée

Contrairement à la définition propre (owned definition), la définition adoptée (*adopted definition*) est une définition issue des ressources extérieures (e.g. dictionnaires). Dans notre approche, nous avons utilisé les définitions proposées par le dictionnaire WordNet [53], tandis que d'autres dictionnaires peuvent être ajoutés, comme le dictionnaire Merriam-Webster [68], puis la communauté peut choisir la définition la plus pertinente. Pour donner un exemple simple de définition fournie par le dictionnaire WordNet : Le terme '*Renter*' est définit par '*Someone who pays for goods or services*'.

- *Concepts généraux*

À cette étape, nous essayons de détecter les relations de spécialisation / généralisation entre les concepts de noms identifiés. Pour ce faire, nous avons utilisé deux patterns, comme l'explique la Table 8.

Table 8 Nos patterns des relations de généralisation

	Pattern	Exemple
P21	Sujet + verbe copulaire + NN1 + "Or" + NN2 … **Le concept général identifié** : - "Sujet" est le concept générale de NN1 - "Sujet" est le concept général de NN2 …	'*Rental is open rental or closed rental*' > '*Rental*' est le concept général de '*open rental*' > '*Rental*' est le concept général de '*closed rental*'
P22	Sujet + verb copulaire + (NN [+clause Adjectival]) **Le concept général identifié** : - NN est le concept général de "Sujet"	'*Gold customer is a customer [that has 5 reservations]*' > '*Customer*' est le conept général de '*Gold customer*'

- *Synonymes et expressions préférées*

Dans les règles métier, un concept peut être exprimée en utilisant différentes expressions, en particulier lorsque ces règles sont rédigées par différents auteurs, ce qui peut générer une confusion conduisant à des incohérences. Ainsi, notre objectif est de détecter les synonymes entre les termes utilisés pour exprimer les concepts et de calculer la fréquence d'apparition de chacun d'eux dans toutes les règles, pour identifier l'expression préférée.

Dans la littérature, le concept de synonyme est lié au concept de similarité et de relation entre entités (mots, phrases, paragraphes ou même documents). Un grand nombre d'algorithmes ont été proposés pour mesurer la similarité entre les mots, classés générales en deux catégories : les similarités lexicales (*String-based measures*) et les similarités sémantiques (*Corpus-based* et *Knowledge-based measures*) [69]. La première catégorie (lexicale) se base sur la distance entre les chaînes de texte pour calculer le degré de similarité entre les séquences de caractères. Tandis que la seconde catégorie (sémantique) utilise des statistiques issues de collections de textes (basées sur des corpus), ou des informations linguistiques prédéfinies issues des réseaux sémantiques comme WordNet (basées sur la connaissance).

Dans notre approche, nous visons à extraire les synonymes en tant que termes ayant la même signification, au lieu de ceux ayant une similarité partielle. Pour atteindre cet objectif, nous ne pouvons pas nous fier à des mesures basées sur la similarité entre chaînes de caractères, sinon, les concepts comme '*car rental*' et '*car renter*' seront détectés comme des synonymes. De même, les mesures fondées sur les connaissances trouvées dans la littérature ont été testées indépendamment avec des résultats insatisfaisants, tels que '*responsibility*' et '*area*' qui ont été détectés comme très similaires, ce qui n'est pas correct. Finalement, la précision de la dernière solution (basé sur le corpus) est liée à la quantité d'informations utilisées, qui demande un grand nombre d'information concernant chaque concept.

Comme solution, nous avons essayé de combiner deux mesures : '*Knowledge based measure*' et '*Corpus-based measure*', la chose qui a généré de meilleurs résultats. Cette méthode de détection des synonymes, ainsi que la relation entre les abréviations et leur définition était le sujet de notre deuxième contribution détaillé dans le chapitre qui suit.

- *Les déclarations nécessaires ou possibles*

Selon le métamodèle du dictionnaire terminologique SBVR, les règles métier sont exprimées à l'aide de déclarations (*Statements*) et marquées comme nécessairement vraies ou fausses pour refléter sa modalité. Cette modalité est identifiée par notre algorithme en testant si le verbe principal a un auxiliaire modal (voir un exemple dans la Figure 14 Exemple de modalité : l'auxiliaire modal du verbe). Sinon, nous cherchons la modalité au début de la phrase comme dans '*It is obligatory that…*'.

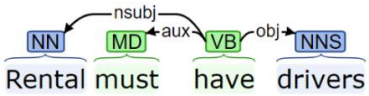


Figure 14 Exemple de modalité : l'auxiliaire modal du verbe

D'autre part, chaque règle est basée sur un concept de verbe, et chaque concept de verbe représente la signification d'un '*verb concept wordings*' qui est exprimé à l'aide de désignations représentant des concepts. Par conséquent, chaque concept est lié à ses règles connexes (*statements*), ce qui nous permettra d'identifier les règles nécessaires pour chaque concept extrait. Par exemple, '*rental must have exactly one grace period*' est une règle qui doit figurer comme légende de nécessité pour le concept '*rental*'.

4.1.5. Génération du fichier XMI

Après avoir identifié tous les concepts et leurs spécifications, notre Vocabulaire métier est enregistré dans un fichier XMI respectant le schéma XMI du standard SBVR. De plus, les patterns cités dans la documentation officielle de SBVR sont utilisés pour générer le dictionnaire terminologique SBVR final. La Figure 15 Extrait du fichier XMI final contenant est un exemple de la manière dont le concept général '*Renter*' a été enregistré avec ses spécifications dans le fichier XMI final.

```
<sbvr:generalConcept xmi:id="gc-120"/>
<sbvr:designation xmi:id="des-gc-120" signifier="txt-gc-120" meaning="gc-120"/>
<sbvr:text xmi:id="txt-gc-120" value="Renter"/>
<sbvr:thing1IsThing2 thing1="gc-120" thing2="gc-141"/>
<sbvr:preferredDesignation xmi:id="gc-120"/>
<sbvr:concept1SpecializesConcept2 concept1="gc-120" concept2="gc-115"/>
<sbvr:intentionalDefinition xmi:id="intDef-gc-120-1"
    expression="txt-intDef-gc-120-1" meaning="gc-120"/>
<sbvr:text xmi:id="txt_intDef-gc-120-1"
    value="A renter is the driver who is responsible for some recognized rental"/>
<sbvr:adoptedDefinition xmi:id="adptDef-gc-120-1"
    expression="txt-adptDef-gc-120-1" meaning="gc-120"/>
<sbvr:text xmi:id="txt_adptDef-gc-120-1"
    value="[WordNet] someone who pays for goods or services"/>
<sbvr:statement xmi:id="stmt-31" expression="txt-stmt-31" meaning="prp-31"/>
<sbvr:proposition xmi:id="prp-31" isNecessarilyTrue="true"/>
<sbvr:text xmi:id="txt-stmt-31"
    value="Each renter must provide contact details for each rental"/>
<sbvr:propositionIsBasedOnVerbConcept proposition="prp-31" verbConcept="vc-63"/>
```

Figure 15 Extrait du fichier XMI final contenant le vocabulaire métier généré

En conséquence, le fichier XMI généré est écrit dans un format standard facilement compréhensible par la machine, et peut être utilisé comme base pour les prochaines étapes du processus de développement ainsi que pour générer d'autres modèles tels que le diagramme de classes ou les ontologies. Cependant, dans notre cas, nous utilisons le fichier XMI pour présenter le dictionnaire terminologique SBVR sous forme de fichier MS-Word facilement compréhensible par les non-spécialistes IT, en utilisant la notation SBVR-SE comme exprimé dans la Table 9 Le mappage entre les éléments du Dictionnaire Terminologique SBVR et la notation SBVR-SE.

Table 9 Le mappage entre les éléments du Dictionnaire Terminologique SBVR et la notation SBVR-SE

SBVR métamodèle	La légende SBVR-SE
Owned definition: - Intentional definition - Extensional definition	Definition
Adopted definition	Dictionary Basis
Relation: Concept1 Specializes Concept 2	General Concept
Necessity statements	Necessity
Possibility statements	Possibility
Relation: Thing1 is Thing2 (for Noun Concept)	Synonym
Relation: Thing1 is Thing2 (for Verb Concept)	Synonymous Form
Preferred designation	See

4.2. Evaluation et discussion

Dans cette section, nous discutons comment on a choisi notre échantillon de test pour évaluer notre approche, ainsi que notre vérité terrain et nos résultats.

4.2.1. Echantillon

Pour évaluer notre approche, un échantillon a été choisis en prenant en considération trois éléments principaux :

- Pour enlever tous les facteurs qui peuvent affecter la réalité des résultats obtenus, nos règles métier sont censés être rédigées en respectant les bonnes pratiques pour qu'elles soient bien formées et ne contiennent aucune ambiguïté.
- Au lieu de tester notre approche sur une grand quantité de règles métier, nous préférons l'appliquée sur le maximum de structures de phrases possible, avec lesquelles les règles métier peuvent être formulées. En d'autres termes, nous avons essayé d'inclure toutes sortes de structures de phrases dans notre échantillon, contenant à la fois des clauses simples (une clause indépendante), des clauses composées (plus d'une clause indépendante) et des clauses complexes (clauses dépendantes ayant des clauses adverbiales)
- Les règles métier doivent être homogènes et complètes autant que possible (très important pour l'extraction des synonymes).

En conséquence, nous avons pensé aux règles métier « EU-RENT » utilisées dans la documentation officielle du standard SBVR comme les règles les plus adaptées à notre cas, car elles décrivent un exemple complet utilisant des déclarations bien formées, homogènes et sans ambiguïté.

Premièrement, nous avons sélectionné toutes les règles textuelles que nous avons trouvé dans la documentation SBVR. Puis, nous avons rejeté les règles exprimant des contraintes de calcul et de « date-heure », pour les raisons évoquées précédemment. En conséquence, 157 instructions ont été conservées représentant trois types de structure : simples, composées et complexes. Nous considérons que toute augmentation du nombre des règles métier n'affectera pas le résultat final, à moins que l'une des trois considérations précédentes ne soit changée.

4.2.2. La vérité terrain

Les règles métier ont été d'abord analysées manuellement pour extraire le vocabulaire métier avec leurs spécifications associées. Ensuite, les résultats ont été comparés avec d'autres générés automatiquement par notre approche. En conséquence, nous pourrions calculer le nombre de Vrai Positif (VP: correct), Faux Positif (FP: incorrect) et Faux Négatif (FN: correct mais non identifié). Ces valeurs nous ont permet de calculer la Précision (Eq 1), Rappel (Eq 2) et F1-Score (Eq 3).

$$\boldsymbol{Précision} = \frac{\mathrm{VP}}{\mathrm{VP} + \mathrm{FP}} \qquad (Eq\ 1)$$

$$\boldsymbol{Rappel} = \frac{\mathrm{VP}}{\mathrm{VP} + \mathrm{FN}} \qquad (Eq\ 2)$$

$$\boldsymbol{F1 - score} = \frac{2 \times \mathrm{Précision} \times \mathrm{Rappel}}{\mathrm{Précision} + \mathrm{Rappel}} \qquad (Eq\ 3)$$

4.2.3. Résultats de l'évaluation

Deux principaux types d'informations ont été automatiquement identifiés et comparés à ceux extraits manuellement :

- Concepts : Concepts verbaux (faits) et concepts nominaux (généraux et individuels).
- Spécifications des concepts : synonymes, définition, relations de généralisation et déclarations nécessaires ou possibles.

Les informations extraites à l'aide des algorithmes ont été présentées dans la Table 10 en utilisant les valeurs VP, FP, FN, précision, rappel et score F1. Cependant, deux éléments n'ont pas été inclus dans le tableau, à savoir les « définitions intentionnelles / extensionnelles » et les « déclarations nécessaire et possible », car nous n'avons utilisé que les patterns (sans aucun algorithme) pour les identifier. Néanmoins, nous nous limitons à montrer à quelle point ces deux éléments pourraient être utiles pour clarifier le sens des concepts extraits.

Table 10 Résultat de l'extraction automatique du vocabulaire métier par notre approche

		Extraction manuelle	Identification automatique	VP	FP	FN	Rappel	Précision	F1 Score
General Concept	**Un seul mot**	30	37	28	9	2	93%	76%	84%
	Deux mots	79	83	78	5	1	99%	94%	96%
	Plusieurs mots	25	27	24	3	1	96%	89%	92%
	Total	130	147	130	17	4	97%	88%	93%
Concept individuel		35	21	20	1	15	57%	95%	71%
Concept de Verbe	**Implicite**	151	153	146	7	5	97%	95%	96%
	Explicite	221	219	216	3	5	98%	99%	98%
	Total	372	372	362	10	10	97%	97%	97%
Relation de spécialisation		139	139	136	3	3	98%	98%	98%
Synonyme		19	14	8	6	11	42%	57%	48%

À partir de 157 déclarations, nous avons pu extraire 713 faits (372 sans duplication, 219 faits explicites avec 98% en F1-score et 153 faits implicites avec 96% de F1-score), contenant 172 concepts nominaux (168 sans duplication, 147 concepts avec 93% en F1-score et 21 concepts individuels avec 71% en F1-score). En revanche, 139 relations de spécialisation ont été identifiées avec 98% en F1-score, tandis que les synonymes extraits n'ont pas dépassé 48% en F1-score.

Enfin, en exploitant uniquement les déclarations des règles métier, nous avons pu affecter une « définition propre » à 30% des concepts de noms et une « définition adoptée » (du dictionnaire WordNet) à 81% des concepts de noms constitués d'un seul mot (les « définitions adoptées » pour les noms composés ne sont pas prises en charge par notre approche). De plus, nous avons pu affecter des déclarations de nécessité / possibilité à 45% des concepts extraits.

4.2.4. Discussion des résultats

Dans cette section, chaque étape du processus de notre méthode est discutée avec les problèmes potentiels et les actions pour les surmonter.

- *Étape 1 : Analyse linguistique*

Parmi 157 règles métier, 9 règles ont été incorrectement analysées par l'outil d'analyse linguistique Stanford CoreNLP. La source des erreurs provenait d'un mauvais 'POS tagging' causé par la confusion entre quelques termes et verbes, comme le terme '*references*' ou '*requests*' qui peuvent être utilisés en anglais à la fois comme nom et comme verbe. L'outil Stanford CoreNLP a été adopté dans notre approche vue sa popularité, mais d'autres outils devraient être testés pour sélectionner l'outil le plus précis qui peut surmonter ce type d'erreurs.

D'autre part, des ponctuations mal placées (ou absentes) nous ont empêché de générer l'arbre grammaticale correcte dans quelques règles complexes ayant des clauses dépendantes (if-then). La ponctuation est un autre problème qui peut affecter le résultat final, et doit être ajoutée avec soin, en particulier dans les phrases complexes qui incluent des conjonctions. Cependant, 94% de règles correctement analysés est considéré comme très satisfaisant.

- *Étape 2 : Extraction des clauses*

A partir des 157 règles, notre méthode a permis d'identifier automatiquement 713 clauses (48% en double). Ce nombre relativement important par rapport au nombre de règles métier choisie, est dû à la dissection des clauses composées ainsi que l'analyse des noms composés, permettant de révéler les clauses implicites.

Identifier à la fois des clauses implicites et explicites (concepts verbaux) avec 97% en F1-Score reflète l'exhaustivité de nos patterns choisis. Cela dit, certains patterns devraient être ajoutés, comme le pattern pour disséquer les clauses ayant plus d'un verbe en relation (e.g. '*EU-Rent site is owned or leased by EU-Rent*'). Cependant, pour une meilleure précision, les règles devraient être aussi simples que possible pour augmenter les chances de trouver des clauses qui correspondent à notre métamodèle générique. Le principal problème à ce niveau est que les règles ne sont pas toujours bien formées. D'autre part, l'utilisation des expressions régulières (Regex) pourrait être une bonne alternative pour couvrir toutes les formes de clauses et de patterns.

- *Etape 3 : Extraction des concepts*

Dans notre exemple, les concepts de nom et de verbe ont été identifiés avec un F1-score dépassant généralement 90%. Néanmoins, l'identification des termes constitués d'un seul mot n'a pas dépassé 84% en F1-score, contrairement aux termes constitués de plusieurs mots qui ont

dépassé 92%. En effet, les termes constitués d'un seul mot sont généralement sujette à l'ambiguïté plus que les termes constitués de plusieurs mots (e.g. '*requests*' peut être utilisées à la foi comme verbe et comme nom).

En ce qui concerne les concepts individuels, l'outil Stanford CoreNLP a détecté inexplicablement le terme '*web Reservations Center*' comme nom d'organisation, tandis que le même outil a ignoré le terme '*telephone reservation center*' même s'ils ont la même structure. En outre, les termes '*GPL*', '*EU-RENT*' et '*ISO*' n'ont pas été détectés comme des concepts individuels, ce qui explique les 57% en valeur de Rappel. Le principal problème à ce niveau est que notre approche dépend entièrement de l'outil NLP pour faire le 'POS-tagging' et générer l'arbre d'analyse grammatical. Dans notre cas, si nous considérons les noms écrits en majuscules, ou les noms qui n'existent pas dans le dictionnaire (WordNet) comme des concepts de noms individuels, la valeur de 'Rappel' va s'augmenter, mais une solution plus sérieuse devrait être appliquer à ce niveau. Ainsi, même si l'outil Stanford CoreNLP n'a généré que 6% de règles incorrectement analysées, cela a dû être corrigé manuellement ou automatiquement avant tout nouvel incrément.

Un autre élément clé à retenir est que les termes constitués de plusieurs mots représentaient 75% des concepts généraux extraits, ce qui reflète la quantité d'informations cachées (e.g. les faits implicites) qui ne seront pas pris en considération si nous les ignorons. Ceci est clairement montré dans le fait que 40% des concepts verbaux extraits (faits) étaient cachés dans les termes constitués de plusieurs mots ainsi que les clauses composées. Ceci est l'un des avantages de notre approche qui permet d'enrichir le vocabulaire métier généré pour le rendre plus complet.

- *Étape 4 : Extraction des spécifications des concepts*

À cette étape, nous avons essayé de collecter plus d'informations sur chaque concept identifié pour enrichir sa signification. Ainsi, certaines spécifications ont été automatiquement identifiées à partir des règles textuelles, puis affectées aux concepts identifiés, tels que les définitions, les synonymes et les déclarations nécessaires ou possible. En conséquence, 30% des concepts nominaux ont reçu une définition uniquement en recyclant les règles métier qui respectent les patterns qu'on a définis, et 45% des concepts ont reçu au moins une déclaration nécessaire ou possible. Concernant les synonymes, ils ont été identifiés avec un F1-score de 48%.

Certes, ces valeurs ne sont pas convaincantes, étant donné que notre approche ne couvre que les concepts fréquents qui apparaissent dans plusieurs règles. En d'autres termes, plus les concepts sont moins fréquents, plus ils ont moins de chance d'avoir des spécifications supplémentaires.

De la même manière, la diminution de l'efficacité de l'extraction des synonymes est attribuée au fait que notre algorithme d'extraction des synonymes dépend du contexte dans lequel les concepts ont été identifiés. Ce contexte est défini par d'autres concepts coexistant dans la même règle. Ainsi, plus le concept est moins fréquent, plus son contexte est faible, et moins on a de la chance à trouver son synonyme. Ceci est clairement montré dans le fait que tous les synonymes non identifiés (faux négatifs) n'apparaissaient pas plus de trois fois dans toutes les déclarations.

Compte tenu de ces points, le principal problème à ce niveau dépend totalement du '*speech community*' qui devrait définir plus de règles métier afin donner une idée plus complète à chaque

concept. En même temps, les résultats obtenus sont tout à fait normaux puisque les concepts les plus fréquents sont généralement les plus importants à définir.

- *Étape 5 : La génération du modèle SBVR*

Toutes les informations identifiées dans les étapes précédentes sont enregistrées dans un fichier XMI temporaire (intermédiaire) avant d'être transformer vers un autre fichier XMI qui respecte le standard SBVR. Ce dernier est sans doute facilement compréhensible par la machine et peut être facilement combiné ou intégré à d'autres approches, vue son format standard. Néanmoins, sa fiabilité dépend totalement de toutes les étapes qui le précèdent.

En dernier lieu, vue le nombre des règles métier utilisées pour valider notre approche, nous pouvons confirmer que les résultats obtenus sont très encourageants, et l'application de notre approche à des règles métier plus complètes produira sans doute des résultats plus précis.

Conclusion

Nous avons présenté une approche entièrement automatisée pour extraire le vocabulaire métier à partir de règles métier textuelles selon le métamodèle du dictionnaire terminologique SBVR. Nous avons utilisé le traitement du langage naturel et les deux méthodes de transformation Text2Model et Model2Model, en s'appuyant uniquement sur les règles métier sans aucune ressources externes ni intervention humaine. Le dictionnaire terminologique généré contient des connaissances explicites et implicites cachées dans des noms composés et des phrases complexes, renforcées par des informations supplémentaires telles que des définitions, des synonymes et des relations de généralisation.

Notre approche utilise les règles métier en entrée comme seule ressource, ce qui rend notre méthode entièrement automatique. Cependant, son efficacité est liée au nombre de règles utilisées, surtout pour l'extraction des définitions et des synonymes. Ainsi, pour avoir une signification plus complète pour chaque concept, nous devrions avoir d'autres déclarations pour chaque concept. En revanche, les règles ne sont pas toujours bien formées, ce qui nécessite fortement de commencer par une couche de filtrage supplémentaire. Heureusement, nous avons rencontré un seul problème causé par un mauvais placement de ponctuation, ce qui nous incite à prendre aussi ce problème en considération dans la phase de prétraitement. De plus, comme mentionné précédemment, nous avons ignoré certains types de règles métier qui doivent également être pris en charge en ajoutant leurs patterns, ou au moins, une couche de filtrage doit être ajoutée pour sélectionner les règles métier éligibles. Enfin, nos patterns d'extraction de clauses à partir des déclarations semblent être les patterns les plus utilisés étant donné les résultats obtenus. Cependant, cela n'empêche pas que d'autres patterns puissent être utilisés dans le monde réel. Par conséquent, l'utilisation des expressions régulières sera un bon choix pour extraire des patterns personnalisés.

Pour conclure, ce travail a révélé que nous pouvons largement nous fier qu'aux déclarations textuelles des règles métier pour générer automatiquement un dictionnaire plus complet pour le vocabulaire métier. Le dictionnaire généré est compréhensible par les êtres humains et les machines, ce qui rend possible l'automatisation du processus de développement dans les phases initiales, à condition que les règles définies soient plus complètes.

5. L'identification automatique des synonymes

Pour prendre des décisions cohérentes, les décideurs s'appuient sur des règles métier cohérentes et bien formées. Le fait que les règles métier sont généralement rédigées par différentes parties prenantes les rend vulnérables à contenir différentes expressions pour un même concept. De tels problèmes peuvent être à l'origine d'un comportement mal orchestré. Le problème c'est que les règles métier sont généralement plus nombreuses pour qu'elles soient traitées manuellement, et moins nombreuses pour qu'elles soient traitées à l'aide des méthodes d'apprentissage automatique. Dans ce chapitre, nous présentons une approche automatisée pour identifier la similarité sémantique entre les termes utilisés dans les règles métier textuelles en utilisant le traitement du langage naturel et un algorithme affiné par des heuristiques. Notre approche a été appliquée sur plus de 160 règles métier réparties sur trois cas avec une précision comprise entre 67% et 87%, ce qui le rend une extension indispensable pour d'autres méthodes traitant les règles métier textuelles.

Sommaire

Introduction

Les approches traitant les règles métier textuelles (ou toutes autres spécifications) reposent principalement sur le traitement du langage naturel (NLP), ou adoptent un langage naturel contrôlé (CNL) [63] facilement compréhensible par les machines. Les approches existantes ont tendance à extraire des connaissances (sémantiques) à partir du texte, mais ignorent d'autres problèmes conduisant à des incohérences, résultant notamment du fait que les règles métiers sont généralement rédigées par différentes parties prenantes. L'utilisation de plus d'une expression pour le même concept en est une.

Les règles métier ne sont généralement ni de petite taille pour qu’elles soient traitées manuellement, ni de grande taille pour qu’elles soient traitées en utilisant les méthodes d'apprentissage automatique. Cela justifie probablement le fait que les approches ont tendance à ignorer l'existence des synonymes [46], ou à fournir une interface utilisateur (UI) pour les ajouter manuellement [42], ou à supposer l'existence préalable d'une base de données contenant le vocabulaire métier [43]. Alors que d'autres approches tentent d'identifier les synonymes à des étapes avancées du processus de développement [52]. Cependant, l'identification de la similarité sémantique entre les termes au début du processus du développement des logiciel va sans doute améliorer la qualité des modèles des étapes suivantes, et par la suite diminuera le nombre d'itérations et le temps de chaque étape.

Dans ce chapitre, nous présentons une méthode [70] pour améliorer les approches existantes traitant les règles métier textuelles en identifiant automatiquement les synonymes à l'aide du traitement automatique du langage naturel, et des heuristiques. Notre approche a été testée sur plus de 160 déclarations de règles métier réparties sur 3 cas différents, avec une précision comprise entre 67% et 80%.

Ce chapitre devrait intéresser ceux qui font le traitement du langage naturel, en particulier ceux qui souhaitent transformer un texte informel en spécifications formelles. Ce chapitre apporte les contributions suivantes :

- Améliorer les méthodes de transformation de texte en modèle, en identifiant les termes ayant la même signification.
- Améliorer la formulation du vocabulaire métier.
- Préservez l'intégrité des règles métier.

L'extraction automatique des termes synonymes est largement liée à l'identification de similarité sémantique entre des entités, à savoir: des termes, des phrases, des paragraphes ou des documents [71] [72]. Pour identifier la similarité entre les termes, 3 méthodes principales sont généralement utilisées dans la littérature [69]: (i) les méthodes basées sur les caractères dans lesquelles la similarité entre deux chaînes de caractères est mesurée; (ii) Les méthodes basées sur un corpus dans lesquelles des statistiques faites sur une collection de textes sont utilisées; (iii) et des méthodes basées sur des informations linguistiques prédéfinies dérivées des réseaux sémantiques tels que WordNet [53]. Certes, ces méthodes ne sont pas toutes adaptées à notre

objectif. Par exemple, les méthodes basées sur des chaînes vont sûrement identifier les entités ayant des séquences de caractères proches, mais des résultats tels que '*Renter*' et '*Rental*' qui sont similaire vue leur racine commun, doivent être considérés comme Faux Positifs dans notre cas. De plus, les méthodes fondées sur les informations linguistiques ne sont pas suffisantes, et doivent être normalement supportées par leur contexte dans lequel le terme a été identifié, comme le montre clairement la définition du concept de «synonyme»: «des mots qui désignent le même concept et qui sont interchangeables dans de nombreux contextes »(WordNet [53]). En ce qui concerne les approches basées sur les corpus, le contexte joue un rôle important. Par exemple, des phrases peuvent être transformées en modèles d'espace vectoriel [73] pour en extraire leur signification. La méthode Word2Vec [74] est largement utilisée à ce niveau pour établir une similarité sémantique via des statistiques basés sur le contexte dans lequel les termes sont utilisés. La précision des méthodes reposant sur cette solution est liée à la taille de données disponibles.

Pour remédier à ce problème, nous avons essayé de nous mettre dans la peau des rédacteurs des règles métier, pour reconnaître les différentes sources de synonymes possibles avant de déterminer les hypothèses suivantes :

- **H1)** Les termes d'une même règle métier sont considérés comme voisins les uns des autres. Autrement dit, tous les termes d'une même règle participent d'une manière ou d'autre à la définition du contexte de chaque terme de cette déclaration.
- **H2)** Les termes n'ayant pas de voisin en commun signifient qu'ils ont une signification différente, même s'ils sont synonymes du point e vue des dictionnaires.
- **H3)** Les termes voisins ne sont pas considérés comme synonymes. Autrement dit, les termes ont deux significations différentes s'ils sont utilisés dans une même règle.

Les règles métier devraient être atomiques, bien formées et rédigées en utilisant le vocabulaire métier au lieu de paragraphes. Ainsi, limiter le contexte des termes aux autres termes qui l'entourent (comme dans les méthodes d'apprentissage automatique) n'est pas un bon choix dans l'environnement des règles métier (H1 et H2). En outre, une règle est souvent formulée par une seule personne ; ainsi, la probabilité de trouver des termes synonymes dans la même règle est très faible par rapport à d'autres règles rédigées par d'autres personnes (H3). Enfin, nous considérons que l'auteur d'une règle choisit intentionnellement des termes propres pour des expressions propres ; ainsi, les hyponymes ne sont pas considérés comme des synonymes.

A la fin, nous avons décidé d'identifier les termes ayant le même sens en se basant sur le dictionnaire avant d'affiner les résultats obtenus à l'aide des heuristiques. D'autre part, une autre solution permettant de relier les abréviations avec leurs extensions (terme non abrégé) est également proposée.

La section suivante détaille notre approche d'identification des synonymes à partir des règles métier textuel ainsi que les paires abréviation-extension, avant de présenter l'évaluation et la discussion des résultats obtenus. Ce chapitre présentant à la fin les limites de cette approche et la conclusion.

5.1. Présentation de l'approche

Notre approche commence par une analyse linguistique utilisant un pipeline NLP pour tokeniser, étiqueter et générer l'arbre d'analyse grammaticale de chaque règle. Après, les termes (simples ou composés) sont extraits de la même manière que notre contribution précédente dans laquelle on a généré le vocabulaire métier. Ensuite, les termes ayant le même sens sont identifiés, ainsi que les abréviations, en utilisant des algorithmes. Finalement, un fichier XMI respectant le standard SBVR est généré, contenant les concepts ainsi que leurs synonymes.

5.1.1. Identification de similarité sémantique

Dans cette étape, les termes identifiés sont comparés les uns aux autres pour identifier ceux qui ont la même signification. En même temps, les termes abrégés sont identifiés avant de rechercher leurs extensions s'elles existent. Enfin, la fréquence de chaque terme est calculée pour identifier l'expression préférée.

- *L'identification des synonymes*

Pour marquer les termes susceptibles d'être sémantiquement similaires, ces termes doivent avoir une distance supérieure à « 0,5 » dans le SynSet (arborescence des sens des termes dans le dictionnaire). De plus, selon les hypothèses précédemment citées (H1, H2, H3), des concepts similaires devraient avoir au moins un voisin en commun, mais n'ont pas été utilisé dans une même règle métier (voir l'algorithme de la Figure 16 Notre algorithme d'identification des synonymes). La distance «0,5 » a été adoptée après des expériences détaillées dans la section suivante.

```
   Data: t1, t2 ∈ Terms / t1 ≠ t2
   Result: boolean : t1 and t2 are synonym or not
1  begin
2      if t1 ∈ ProperNoun ∨ t2 ∈ ProperNoun then
3          return false
4      end
5      //∃t3 ∈Terms ∧ ∃s1,s2 ∈statements/
       //{t1,t3}⊂s1 ∧ {t2,t3}⊂s2 ∧ s1≠s2
       Let Nt1 be terms coexisting with t1 in at least 1 statement
       Let Nt2 be terms coexisting with t2 in at least 1 statement
       if Nt1 ∩ Nt2 = ∅ then
6          return false
7      end
8      // ¬∃s ∈ statements / {t1,t2} ⊂ s
       Let St1 be statements containing term t1
       Let St2 be statements containing term t2
       if St1 ∩ St2 ≠ ∅ then
9          return false
10     end
11     if similarity(t1,t2) < 0.5 then
12         return false
13     end
14     return true
15 end
```

Figure 16 Notre algorithme d'identification des synonymes

- *L'identification des abréviations-extensions*

Dans cette sous-section, nous présentons notre méthode pour identifier et relier les abréviations à leurs formats non abrégés (s'elle existe).

Dans la littérature, il existe trois approches principales pour identifier les paires d'abréviation-extension [75] : (i) les approches basées sur les heuristiques (patterns et NLP) dans lesquelles des règles sont utilisées [76] [77] [78] [79] [80] [81]; (ii) des approches basées sur l'apprentissage automatique [82] dans lesquelles des exemples étiquetés sont utilisés pour l'apprentissage automatique ; (iii) et des approches basées sur le Web, dans lesquelles les abréviations - extensions sont identifiées en se basant sur des ressources Web [83]. Il existe également des approches hybrides comme dans [84] qui ont utilisé des contraintes codées à la main avec un apprentissage supervisé.

Les approches basées sur le Web peuvent être un bon choix pour les abréviations ordinaires, mais les règles métier utilisent généralement une terminologie spécifique au domaine dans laquelle des abréviations peu courantes sont utilisées. D'autre part, les approches basées sur l'apprentissage automatique nécessitent un grand nombre de données pour un bon entraînement automatique, ce qui n'est pas possible avec un nombre limité de règles métier. Étant donné que les abréviations

suivent généralement le format standard déterminé dans leur définition dans le dictionnaire[1], nous avons décidé d'utiliser des heuristiques dont chacune peut rejeter des extensions d'abréviation candidates.

Les contraintes fortes peuvent rejeter les vraies paires d'abréviation-extension, telles que: une abréviation doit être dans un mot entre parenthèses [85] [86], ou bien doit être en majuscules [81], ou adjacente à des parenthèses [77], ou bien les extensions entourent toujours leurs abréviations [87]. Pour cette raison, nous nous sommes basés sur les heuristiques suivantes qui ne vont pas au-delà de la définition d'abréviation dans le dictionnaire :

- Les lettres d'une abréviation doivent correspondre à la première lettre (ou aux lettres) de chaque mot dans sa forme non abrégée.
- L'abréviation potentielle sera comparée à des termes composés de plusieurs mots.
- Une abréviation n'est pas un mot connu (par le dictionnaire). Pour cette raison, elles sont généralement marquées comme des noms propres (NNP).

Quelques cas particuliers sont aussi pris en considération tels que :

- «X » peut être lié à un mot commençant par « EX » (e.g. '*Exchange*' de 'XML').
- Le « s » minuscule à la fin de l'abréviation peut être ignoré car il est généralement utilisé pour exprimer le pluriel.
- «2 » et «4 » peuvent signifier respectivement «to » et « for ».
- Une abréviation candidate est comparée en deux temps avec son extension potentielle : avec et sans prise en compte des prépositions.

Ces heuristiques sont implémentées dans l'algorithme de la Figure 17 Notre algorithme d'identification et liaison des abréviations à leur extentions.

[1] "An abbreviation consisting of the first letters of each word in the name of something, pronounced as a word" - Cambridge dictionary

```
   Data: abbrev = abbrevaition condidat (concept)
         exp = expansion condidat (concept)
   Result: boolean b = abrev and exp are synonym or not
 1 begin
 2   //∀ a ∈ abbreviations, |wordsOf(a)|=1
     //∀ e ∈ expansions, |wordsOf(e)|>1
     if ( ¬isSingleWord(abrev) ∨ ¬isMultiWord(exp) ) then
 3     return false
 4   end
 5   //∀ a ∈ abbreviations, a ∈ ProperNouns
     if abrev ≠ ProperNoun then
 6     return false
 7   end
 8   //∃t ∈Terms ∧ ∃s1,s2 ∈statements/
     //{t,abbrev}⊂s1 ∧ {t,exp}⊂s2 ∧ s1≠s2
     Let Na be terms coexisting with abbrev in at least 1 statement
     Let Ne be terms coexisting with exp in at least 1 statement
     if Na ∩ Ne = ∅ then
 9     return false
10   end
11   /*∀ word ∈ exp, firstCharacterOf(word) ∈ characterOf(abbrev)
     for each w ∈ wordsOf(exp) do
12     if firstCharacterOf(w) ∉ charactersOf(abbrev) then
13       return false
14     end
15   end
16   // ∀ c ∈ charactersOf(abbrev), c ∈ charactersOf(exp)
     if charactersOf(abbrev) ⊄ charactersOf(exp) then
17     return false
18   end
19   /* the sequencing of characters in abbreviation should
     respect the sequencing of those characters in its expansion*/
     Let Ca(i) be the character at position(i) in abbrev.characters
     Let ICa(i) be the list of indexes of Ca(i) in exp.characters
     with i > 0 and i<abbrev.lenght-1
     if min(ICa(i)) ≥ max(ICa(i+1) then
20     return false
21   end
22   return true
23 end
```

Figure 17 Notre algorithme d'identification et liaison des abréviations à leur extentions

5.1.2. Génération du fichier XMI

Après avoir extrait tous les termes et identifié les synonymes, les abréviations et les expressions préférées, notre résultat est enregistré dans un fichier XMI qui respecte le schéma SBVR XMI. Voire un extrait dans la Figure 18 Un extrait du fichier XMI dans lequel deux synonymes sont enregistrés selon le format SBVR.

```
<sbvr:generalConcept xmi:id="gc-13"/>
<sbvr:designation xmi:id="des-gc-13"
    signifier="txt-gc-13" meaning="gc-13"/>
<sbvr:text xmi:id="txt-gc-13"
    value "berred client"/>
<sbvr:generalConcept xmi:id="gc-21"/>
<sbvr:designation xmi:id="des-gc-21"
    signifier="txt-gc-21" meaning="gc-21"/>
<sbvr:text xmi:id="txt-gc-21"
    value "blocked customer"/>
<sbvr:thing1IsThing2
    thing1="gc-13" thing2="gc-21"/>
<sbvr:preferredDesignation xmi:id="gc-13" />
```

Figure 18 Un extrait du fichier XMI dans lequel deux synonymes sont enregistrés selon le format SBVR

5.2. Evaluation et discussion

Nos expériences ont été réalisées en deux étapes. Tout d'abord, nous avons testé certains algorithmes basés sur le SynSet du dictionnaire (arborescence des synonymes) pour sélectionner l'algorithme le plus précis afin de l'adopter dans notre approche. Ensuite, les résultats obtenus à partir de l'algorithme sélectionné ont été affinés en utilisant nos heuristiques.

5.2.1. Choix de l'algorithme adéquat

Pour choisir l'algorithme le plus précis pour la détection des synonymes, nous avons injecté 20 paires aléatoires des synonymes dans un groupe de 157 règles métier sélectionnées au hasard de EU-Rent utilisée dans la documentation SBVR. Ensuite, nous avons utilisé 8 algorithmes pour identifier les synonymes injectés, à savoir: WUP [88], RES [89], JCN [90], LIN [91], LCH [92], LESK [93], HSO [94] et PATH . Ces algorithmes ont été testés en utilisant différentes distances (à chaque fois) entre les termes. Les résultats obtenus sont présentés dans la Figure 19 Comparatif des résultats obtenus des différents algorithmes d'identification des synonymes.

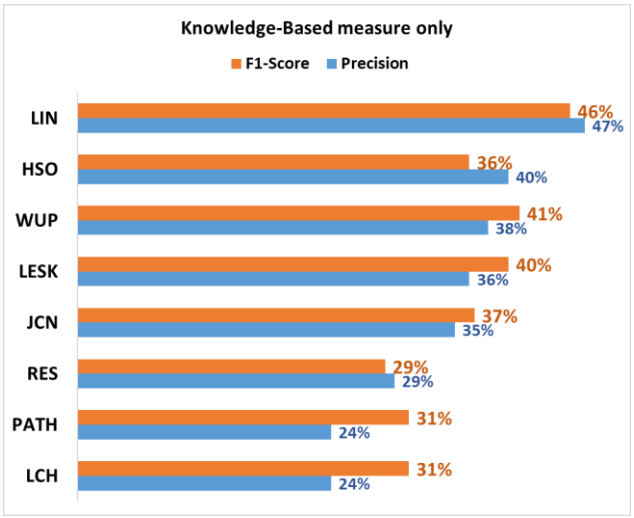


Figure 19 Comparatif des résultats obtenus des différents algorithmes d'identification des synonymes.

Cette figure montre clairement que l'algorithme LIN génère des résultats plus précis par rapport aux autres, et c'est la raison pour laquelle on l'a adopté dans notre approche. Cet algorithme a été testé en utilisant différentes distances entre les termes. A la fin, les résultats les plus précis ont été obtenus en sélectionnant les termes ayant une distance supérieure ou égale à 0.5 comme montré sur la Figure 20 Résultats de l'algorithme LIN pour la détection des synonymes.

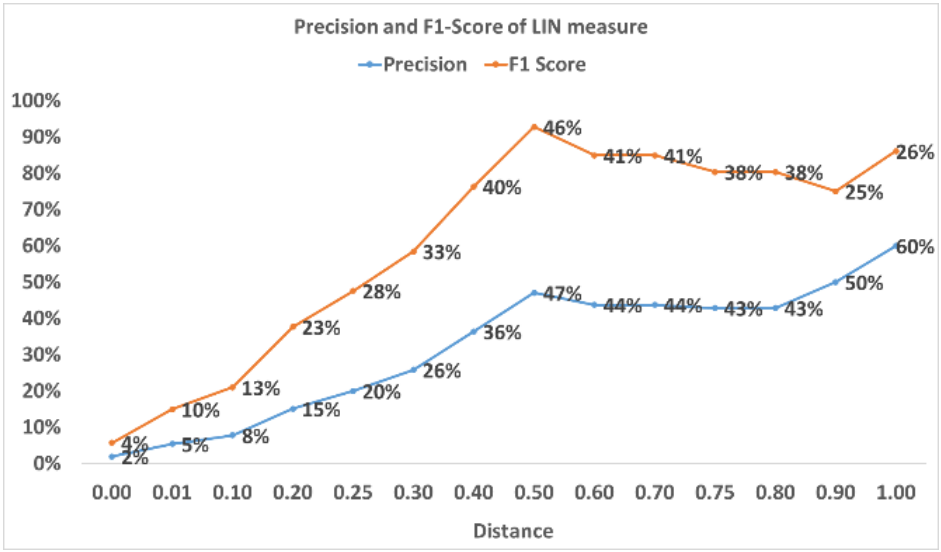


Figure 20 Résultats de l'algorithme LIN pour la détection des synonymes selon les distances entre termes

5.2.2. Evaluation

Notre approche a été testée sur plus de 160 règles métier textuelles, prises de 3 sources différentes: T.Morgan [66] (cas 1), G.Witt [4] (cas 2) et I.Graham [67] (cas 3).

Tout d'abord, pour supprimer toutes les perturbations qui peuvent être causées par l'ambiguïté du langage naturel, nous avons reformulé certaines règles métier pour qu'elles respectent les bonnes pratiques de rédaction cités dans [4]. Malgré cela, quelques règles métier (3%) ont été incorrectement étiquetées (POS tagging) par l'outil Stanford CoreNLP. La source d'erreur à ce niveau était une confusion entre quelques termes qui peuvent être traités à la fois comme des noms et des verbes (e.g. '*references*' et '*requestes*'). D'autres règles avaient un problème de ponctuation qui a affecté l'analyse grammaticale de certaines règles. Néanmoins, 97% des règles correctement analysées est considéré comme très satisfaisants. La Table 11 résume les résultats obtenus à cette première étape de notre démarche qui concerne l'analyse linguistique.

Table 11 Résultats de l'analyse linguistique des règles métier textuelles par notre approche

Source	Règles métier	Termes	Termes sans répétition	Termes incorrectement analysés
Case 1	48	141	83	2,9%
Case 2	56	216	91	5,5%
Case 3	58	310	66	0,6%

Pour simuler le contexte dans lequel les règles ont été rédigées, nous avons divisé chaque cas en 3 parties, avant d'envoyer une partie de chaque cas à deux ingénieurs différents en leur demandant d'en reformuler certaines, en introduisant des synonymes et des abréviations. Ensuite, on a tenté de les identifier automatiquement en utilisant notre approche.

- *L'identification des synonymes*

Pour évaluer l'identification des synonymes, nous avons calculé le nombre de Vrai Positif (VP), Faux Positif (FP) et Faux Négatif (FN) conduisant à la calcule de la Précision, Rappel et le F1-Score, en utilisant les mêmes équations du chapitre précédent (Eq 1, Eq 2 et Eq 3).

La Figure 21 La précision de l'algorithme LIN choisi pour identifier les synonymes (avec et sans heuristique). montre à quel point nos heuristiques ont pu améliorer la précision de l'identification des synonymes par l'algorithme choisi (LIN).

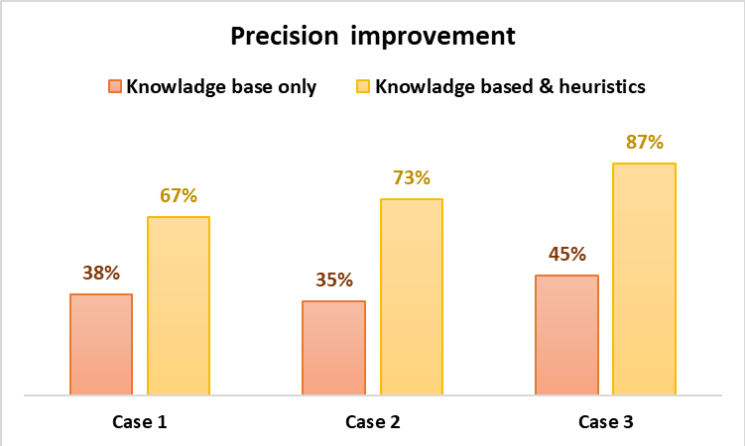


Figure 21 La précision de l'algorithme LIN choisi pour identifier les synonymes (avec et sans heuristique).

La figure montre clairement qu'il y avait une amélioration entre 29% et 42%, lorsque nous avons sélectionné des termes ayant au moins un terme voisin en commun.

La précision, le rappel et le F1-score pour chaque cas sont résumés dans la Table 12 et la Figure 22 La relation entre la précision de notre approche d'identification des synonymes et la fréquence des termes.

Table 12 Résulte de l'identification des synonymes par notre approche

Source	Précision	Rappel	F1-Score	Termes répétés
Case 1	67%	71%	69%	41%
Case 2	73%	79%	76%	58%
Case 3	87%	87%	87%	78%

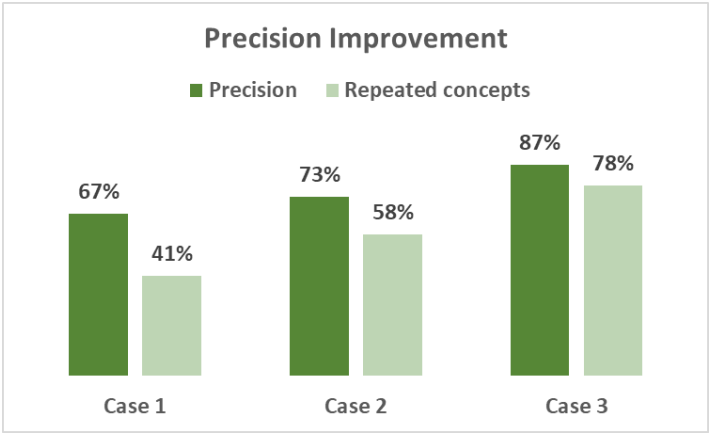


Figure 22 La relation entre la précision de notre approche d'identification des synonymes et la fréquence des termes utilisés

Le diagramme montre clairement que la précision est proportionnelle à la fréquence d'utilisation de chaque terme. Cela pourrait être relié au fait que plus les termes apparaissent

fréquemment dans des règles différentes, plus ils reflètent la bonne signification et plus leurs termes similaires sont susceptibles d'être identifiés. Cette conclusion est approuvée par le fait que la plupart des synonymes qui n'ont pas été identifiées apparaissent dans moins de 3 regelés différentes.

Finalement, ces résultats suggèrent que les termes les plus susceptibles d'être synonymes sont ceux ayant au moins un terme voisin en commun mais n'apparaissant pas dans une même règle, et ayant une distance supérieure à 0,5 selon l'algorithme LIN.

Néanmoins, peu de termes qui sont synonymes par rapport au contexte dans lequel ils ont été utilisés (ayant des termes voisins en communs) n'ont pas été identifiés en raison de leur distance inférieure à 0.5 (par exemple la distance entre '*confirmation*' et '*response*' est de 0.43 alors qu'ils avaient la même signification). D'autre part, malgré que certains termes aient une distance supérieure à 0.5 mais n'ont pas été identifiés également à cause du manque du voisin en commun.

- *L'identification des abréviations*

En ce qui concerne l'identification des extensions des abréviations, toutes les paires ayant respecté le format standard déterminé par la définition de l'abréviation, ont été correctement identifiés avec une grande précision. Les abréviations standard telles que "second=2nd" et "centimeter=cm" peuvent être identifiées facilement à l'aide des dictionnaires. C'est pourquoi elles ont été négligées pour ne pas affecter nos résultats.

La précision, le rappel et le F1-score pour chaque cas sont résumés dans la Table 13.

Table 13 Le résultat de l'identification des abréviations et leurs extensions obtenus par notre approche

Source	Precision	Recall	F1
Case 1	86%	86%	92%
Case 2	86%	86%	92%
Case 3	100%	100%	100%

D'autre part, quelques abréviations ont été reliées à plus d'une extension, plus précisément celles composées de deux lettres telles que '*BC*' qui a été liée à la fois au terme '*Business Class*' et à '*Booking Confirmation*'. De plus, certaines abréviations qui ne respectaient pas la définition n'étaient pas non plus identifiées comme '*Average Hours of work per Week = Ah/w*'. Cette dernière devrait être écrite comme '*Ahw/w*' pour qu'elle respecte la définition et par la suite être identifiée par notre approche.

Conclusion

Notre contribution concerne l’amélioration des approches existantes traitant des règles métier textuelles, en identifiant automatiquement les termes synonymes ainsi que les paires abréviation-extension. Notre objectif est d'extraire des termes ayant la même signification mais des expressions différentes. Pour cette raison, des heuristiques ont été utilisées avec le traitement du langage naturel sans intervention humaine. Les résultats obtenus sont stockés dans un fichier XMI respectant le standard SBVR pour faciliter son intégration dans les approches existantes. L'efficacité de notre approche repose principalement sur deux facteurs : (i) la fréquence d’utilisation des termes : car elle dépend des termes voisins pour refléter sa signification réelle. (ii) la mesure basée sur les informations du dictionnaire : qui donne le degré de similarité entre les termes en utilisant des informations dérivées des réseaux sémantiques.

La présente méthode ne traite que les paires abréviation-extension, ainsi que les termes synonymes composés du même nombre de mots ayant une relation basée sur un dictionnaire. Néanmoins, d'autres types de synonymes sortant du cadre de cette approche devraient également être identifiés, notamment ceux composés de termes ayant des sens différents. Les méthodes d'apprentissage automatique sont - sans aucun doute - la solution la plus adéquate pour identifier les termes complexes qui n'ont pas de relation basée sur un dictionnaire mais qui nécessitent davantage de données pour l’apprentissage, ce qui n'est généralement pas disponible dans le contexte des règles métier. Ainsi, notre approche pourrait être l'alternative le plus efficace pour le contexte des règles métier, vue leur nombre relativement petit, et pourrait être employée pour améliorer les approches existantes traitant les règles métier textuelles.

6. Les règles métier : de l'informel à SBVR

Dans ce chapitre, nous présentons une approche pour transformer automatiquement les règles métier textuelles en modèle SBVR pour faciliter son intégration dans les infrastructures informatiques. Notre approche se distingue des travaux existants en ce qu'elle utilise un traitement du langage naturel approfondi pour extraire un modèle SBVR plus complet. Pour trois ensembles de règles métier issus de domaines différents, nous avons pu générer la sémantique correcte des règles métier avec une moyenne qui dépasse généralement 87% en F1-score.

Sommaire

Introduction

Dans ce chapitre, nous proposons une méthode pour transformer automatiquement les règles métier textuelles (informelles) à un format plus formel en les présentant sous une forme respectant le standard SBVR. Ce format permettra de diminuer le fossé entre les experts métier et les experts IT, et par la suite de démarrer la transformation du processus de développement logiciel à partir des phases préliminaires.

Généralement, il existe trois grandes catégories des règles métier (BR) [23] : i) les règles exprimant des faits qui doivent ou devraient être vraies ou faux, ii) les règles vérifiant une condition avant d'activer d'autres actions, iii) et les règles générant de nouvelles informations (e.g. les calculs).

La différence entre ces types de règles métier nécessite une différence de modèles de phrases exprimant ces règles. Vue le grand nombre de modèles et de patterns suivant lesquelles les règles métier peuvent être formulées, nous nous sommes concentrés sur les règles de la première et de la deuxième catégorie ; à savoir, les règles métier exprimant des instanciations et des caractérisations de concepts, et les règles métier ayant une formulations de 'antécédents / conséquences', car ces deux types de règles métier sont généralement riche en matière de vocabulaire métier et par la suite ils vont jouer un rôle important dans l'enrichissement du dictionnaire terminologique que nous essayons de générer dans notre approche. D'autres types de règles peuvent être facilement intégrées en ajoutant leurs patterns associés (et lexique si nécessaire), mais cela se fera au détriment de la clarté de notre approche. Alors, ces règles ignorées seront prises en compte dans les travaux futurs.

Notre approche suit trois étapes avant de générer le résultat final :

1. L'analyse linguistique : l'étiquetage des parties de parole (POS tagging) et l'arbre de l'analyse grammaticale sont générés pour chaque règle, avant d'être disséqués pour identifier automatiquement les clauses.
2. La génération du vocabulaire métier : un dictionnaire terminologique basé sur SBVR est généré en utilisant les patterns. Ce dictionnaire contient des concepts nominaux, des concepts verbaux, des définitions, des concepts généraux, des déclarations nécessaires et des synonymes pour chaque concept identifié.
3. La génération des règles métier : la formulation sémantique utilisée pour exprimer le sens de chaque règle est identifiée ; à savoir : la formulation atomique, les formulations d'instanciation et les formulations logiques telles que les implications, la négation, les quantifications, la modalité et les conjonctions / disjonctions.

Pour évaluer notre méthode, des expériences ont été réalisées sur trois cas différents provenant de trois sources différentes [66] [4] [67]. Les valeurs obtenues à partir des traitements manuels et d'autres automatiques ont été comparées, et elles ont révélé que notre approche est suffisamment fiable pour transformer automatiquement les règles métier textuelles en un format formel.

Notre approche est une nette amélioration par rapport aux méthodes actuelles traitant les règles textuelles, car elle génère une signification plus complète du vocabulaire et ainsi que des règles métier.

Ce chapitre est structuré comme suit : premièrement nous détaillerons notre approche pour la génération automatique de la sémantique du vocabulaire et des règles métier à partir des règles textuelles. Puis, nous donnons un exemple illustratif de notre approche. Ensuite, nous présentons l'évaluation et les expériences faites ainsi qu'une discussion des résultats obtenus en termes d'identification du vocabulaire métier et de formulation sémantique des règles textuelles. Puis, nous présentons les menaces qui peuvent affecter la validité de l'approche avant de conclure.

6.1. Présentation de l'approche

Dans cette section, nous présentons étape par étape notre proposition pour extraire la sémantique des règles métier textuelles selon le standard SBVR. Un aperçu sur l'approche est présenté dans la Figure 23 Un aperçu de notre approche d'identification de la sémantique des règles métier.

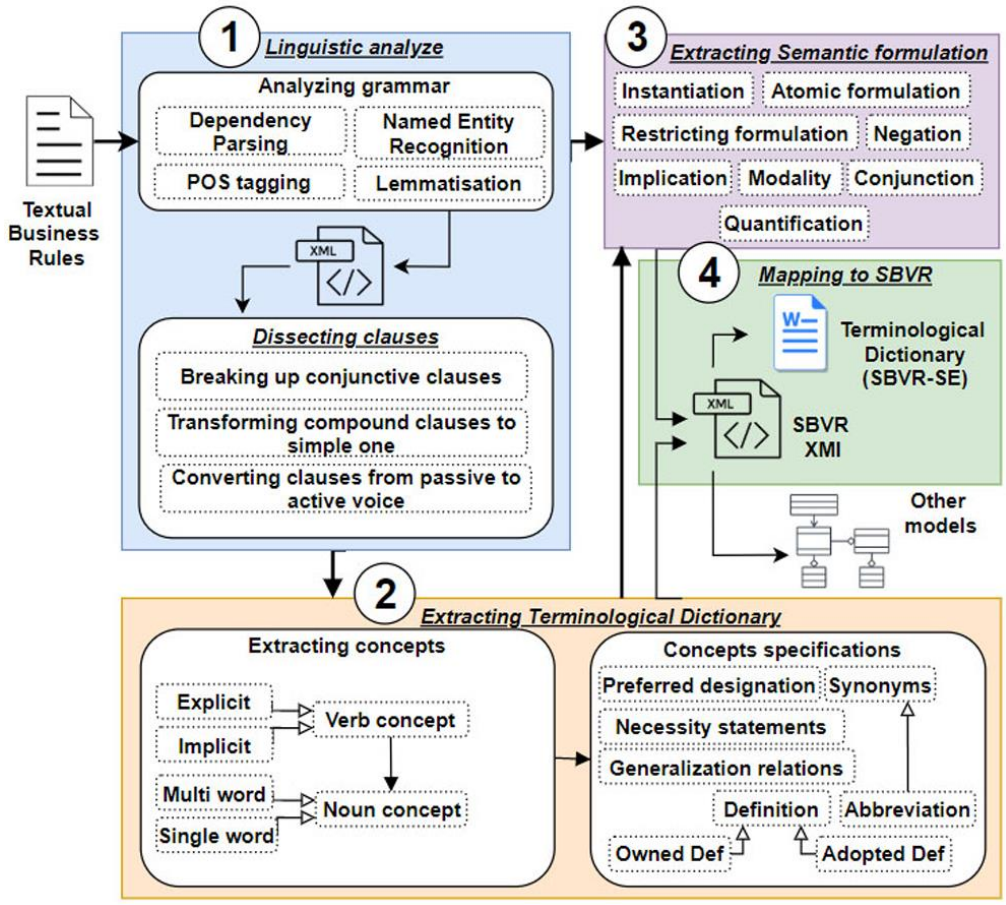


Figure 23 Un aperçu de notre approche d'identification de la sémantique des règles métier

En fait, le standard SBVR fournit un métamodèle pour décrire à la fois le vocabulaire métier en plus des règles métier. L'identification du vocabulaire métier était le sujet de nos contributions précédentes qu'on a décrit dans les deux derniers chapitres. Alors que ce chapitre représente une extension permettant de générer (en plus du vocabulaire) toute la sémantique des règles métier à travers l'identification des formulations sémantiques définis par le standard SBVR. La **Figure 24** présente notre métamodèle SBVR cible représentant à la fois le vocabulaire métier et la sémantique des règles métier à extraire depuis les règles métier textuelles.

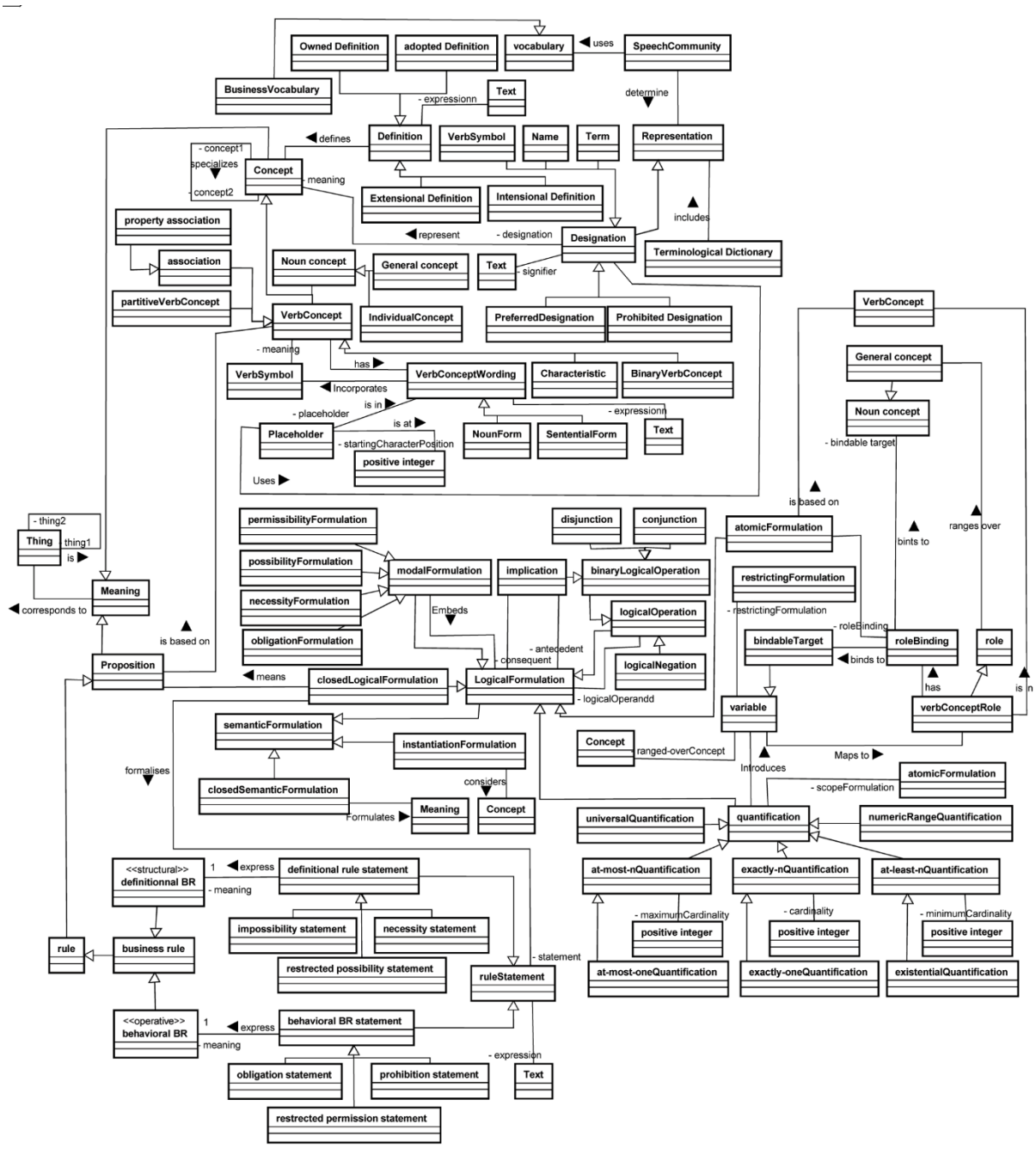


Figure 24: le métamodèle SBVR cible représentant le vocabulaire et la sémantique des règles métier

Tout d'abord, on commence par l'extraction du vocabulaire suivant la méthode décrite dans les deux chapitres précédents. Nous commençons premièrement par une analyse linguistique de chaque règle afin d'identifier les parties du discours (POS) de chaque mot de chaque règle. Puis, les clauses qui composent chaque règle sont extraites et simplifiées pour obtenir leur forme la plus atomique. Ensuite, les concepts verbaux explicites sont identifiés en se basant sur les clauses précédemment extraites, ainsi que les concepts de noms, avant d'ajouter d'autres spécifications,

telles que les définitions, les synonymes, les expressions préférées, les règles nécessaires et les relations de généralisation.

L'extraction du dictionnaire terminologique nous a permis d'identifier les formulations sémantiques exprimant la signification des règles métier, à savoir : les formulations atomiques, les formulations d'instanciation, les formulations de restriction, les modalités, les implications, les quantifications, les conjonctions et les négations. Enfin, Cette sémantique est enregistrée dans un fichier XMI respectant le standard SBVR pour des fins de portabilité et de réutilisation.

Dans notre contribution, nous nous sommes concentrés à nouveau sur les règles qui portent des informations ou des contraintes sur d'autres concepts, car ils jouent un rôle important dans l'enrichissement de notre dictionnaire terminologique généré, comme déjà cité dans les chapitres précédents. Les types de règles métier ignorées dans notre approche sont celles qui expriment un processus métier (e.g. '*A passenger may board a flight only after checking flight*') ou des calculs (e.g. '*income limit is the EITC income limit plus $1.000*').

Le reste de cette section détaille chacune des étapes de notre approche.

6.1.1. L'Analyse linguistique

Après une analyse des différentes structures des phrases exprimant des règles métier discutées dans différents livres spécialisés dans le domaine [66] [4] [67], nous avons créé un métamodèle générique (Figure 25) regroupant les éléments grammaticales nécessaires à identifier dans la structure des phrases exprimant les règles métier, que nous allons utiliser pour générer notre modèle SBVR.

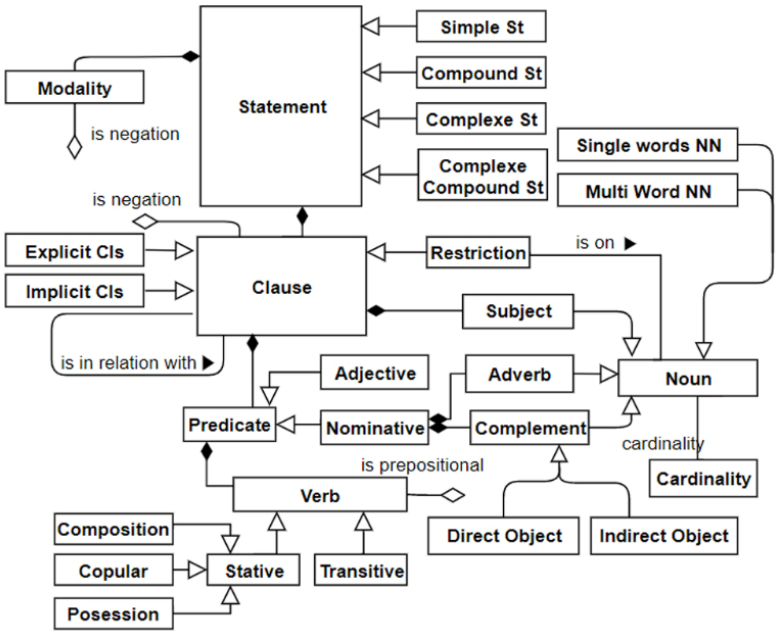


Figure 25 Notre métamodèle générique pour les règles métier textuelles pour l'extraction de leurs sémantiques

Selon ce métamodèle générique, les règles métier textuelles doivent inclure une modalité et être composées d'au moins une clause explicite (phrase simple avec clause indépendante) qui peut être en relation avec d'autres clauses en utilisant des prépositions telles que des conjonctions (phrase composée avec des clauses indépendantes) ou une implication (phrase complexe avec clauses dépendantes).

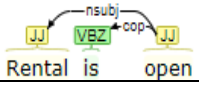
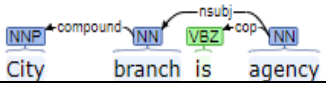
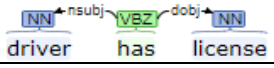
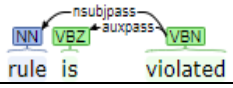
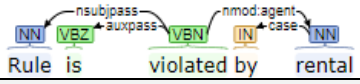
De plus, les clauses doivent être composées d'au moins un sujet avec son prédicat, qui est à son tour composé d'au plus un verbe et des compléments (directs et indirects). Les verbes peuvent exprimer l'état du sujet (verbe statif), ou bien ils dénotent une action effectuée par le sujet envers l'objet (verbe transitif). D'autre part, les noms peuvent être introduits par une cardinalité et peuvent être trouvés sous une forme simple (un mot), ou sous une forme composée (multi-mot). Les noms peuvent également avoir une restriction explicite comme dans '*the car that is red*', ou bien implicite comme dans '*red car*' qui signifie la même chose.

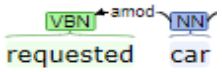
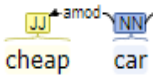
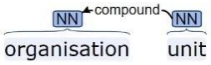
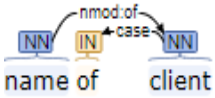
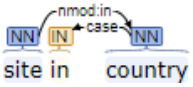
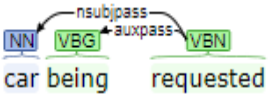
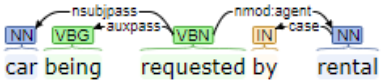
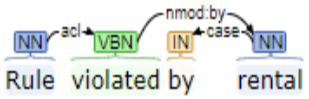
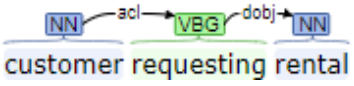
En générale, notre métamodèle générique comprend les éléments les plus importants qui devraient exister dans une déclaration de règles métier textuelle bien formée. Ainsi, en identifiant les composants de ce métamodèle générique, nous aurons tous les éléments nécessaires pour le mappage vers notre modèle SBVR cible.

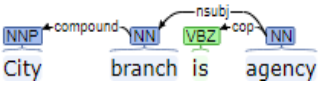
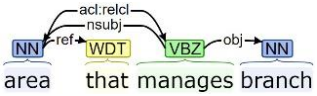
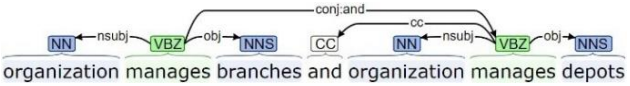
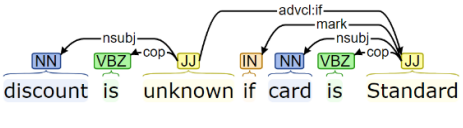
Pour cette raison, nous avons utilisé Stanford CoreNLP à nouveau [64] pour générer les étiquettes Part-Of-Speech (POS), et l'arbre d'analyse grammaticale de chaque règle, avant d'exploiter le résultat pour générer un fichier XML intermédiaire qui respecte notre métamodèle générique.

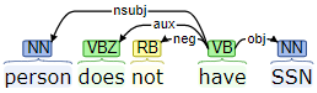
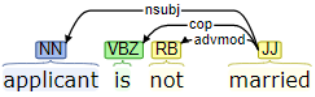
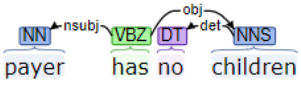
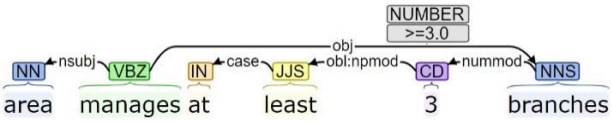
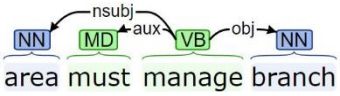
Les éléments composant ce métamodèle seront identifiés à l'aide des paternes de la Table 14 pour générer à la fois la sémantique du vocabulaire et celle des règles métier.

Table 14 Nos patterns des éléments principales constituants les règles métiers textuelles

	Patterns	**Exemple**
Patterns exprimant les clauses explicites		
P1	Suject + 'to be' + Adjective	
P2	Subject + 'to be' + Noun	
P3	Subject + Verb + Direct Object	
P4	Passive Nominal Subject +'to be'+ Verb in Past Participle + no agent	
P5	Passive Nominal Subject + 'to be' + Verb in Past Participle + agent	
Pattern exprimant les clauses implicites		

P6	Noun (NN) + Verb in past participle (VBN) as a modifier *Implicit clauses*: VBN + NN + "is" + NN & NN + "is" + VBN	
P7	Noun (NN) + Adjective (JJ) as a modifier *Implicit clauses*: JJ + NN + 'is' + NN & NN + 'is' + JJ	
P8	Noun (NN1) & Noun (NN2) in compound relation *Implicit clauses*: NN1 NN2 + 'is' + NN2	
P9	Noun (NN1) + 'of' +noun modifier (NN2) *Implicit clause*: NN1+'is for'+NN2 & NN2+'has'+NN1	
P10	Noun (NN1) + Preposition (≠'of') + noun modifier (NN2) *Implicit clause*: NN1 + 'is' + preposition + NN2	
P11	Passive Nominal Subject + "to be" in present participle + Verb in past participle + no Agent *Implicit clause*: nsubjpass + "is" + VBN	
P12	Passive Nominal Subject + "to be" in present participle + Verb in past participle + Agent *Implicit clause*: Agent + VBN + nsubjpass	
P13	Noun + adjectival clause (verb in past participial + "by" + noun modifier) *Implicit clause*: NN2 + VBN + NN1	
P14	Noun + adjectival clause (verb in present participle + Direct Object) *Implicit clause*: NN1 + VBG + Dobj	

Pattern exprimant des définitions pour les concepts		
P15	*Extensional definition*: Subject + 'to be' + Predicate1 + "Or" + Predicate2 …	'*Rental is advance rental or walk-in rental*' may be used as an extensional definition for the concept '*Rental*'
P16	*Intentional definition*: Subject + 'to be' + (NN [+ Adjectival clause])	'*Driver is a person [who is authorized for rental]*' may be used as an intentional definition for the concept '*Rental*'.
Pattern reflétant des concepts généraux		
P17	Subject + "to be" + NN1 + "Or" + NN2 … *Extracted general concept*: – 'Subject' is a General Concept of NN1 – 'Subject' is a General Concept of NN2	'*Rental is open rental or closed rental*' '*Rental*' is General concept of '*open rental*' '*Rental*' is General concept of '*closed rental*'
P18	Subject + "to be" + (NN [+ Adjectival clause]) *Extracted General Concept:* NN is a General Concept of 'Subject'	'*Gold customer is a customer [that has 5 reservations]*'. '*Customer*' is General concept of '*Gold customer*'
Pattern exprimant les formulations logiques de SBVR		
P19	*Atomic formulation:* NN1 (subject) + transitive verb+ NN2 (direct object)	
P20	*Instantiation formulation:* Noun Concept NN1 + 'to be' + Noun Concept NN2.	
P21	*Restricting Formulation:* NN + Relative clause modifier (acl:relcl).	
P22	*Conjunction/Disjunction:* Verb Concept 'p' + 'and / or' + Verb Concept 'q'	
P23	*Implication:* Verb Concept 'p' + Verb Concept 'q' as adverbial clause	

P24	*Negation:* Verb having the adverb 'not' as a negation.	
P25	*Negation:* Adjective having the adverb 'not' as adverbial modifier	
P26	*Negation:* a subject or object having the determinant "no".	
P27	*Quantification:* NN + cardinal number as a numerical modifier	
P28	*Modal Formulation:* Verb having modal (MD) as an auxiliary	

6.1.2. Extraction des clauses

Les règles textuelles sont composées de clauses contenant à la fois un sujet (au moins un) et un prédicat. Pour extraire la sémantique la plus complète de chaque règle, on a suivi la même méthode expliquée dans notre contribution de l'extraction du vocabulaire métier, en identifiant tout d'abord à la fois les clauses explicites et implicites.

- *Clauses explicites :*

Les POS-tags et l'arbre d'analyse grammaticale générés à partir de chaque règle sont analysés pour chercher les paternes P1, P2, P3, P4 et P5 exprimant les clauses explicites. Le pattern P5 exprime la voix passive, et doit être convertie à la voix active avant de le sauvegarder.

- *Clauses implicites :*

Pour effectuer une analyse linguistique approfondie de chaque règle, nous cherchons les modèles P6, P7, P8, P9, P10, P11, P12, P13 et P14 qui expriment des clauses implicites cachées dans les termes composés de plusieurs mots.

Les clauses ayant plus d'un sujet ou objet sont disséquées en sous clauses plus simple, pour pourvoir extraire les informations les plus atomiques.

Dans les étapes suivantes, nous détaillons comment nous pourrions utiliser des modèles pour extraire la sémantique et générer à la fois le vocabulaire métier et les règles métier qui respectent notre métamodèle SBVR cible.

6.1.3. La génération du vocabulaire métier

Dans cette étape, nous générant le dictionnaire terminologique en suivant la même méthode discutée dans les deux chapitres qui précèdent. Vous pouvez vous y référer pour plus de détails.

6.1.4. La génération des règles métier

Pour extraire la signification des règles métiers textuelles, le standard SBVR utilise des faits structurés sur les « Formulations sémantiques » au lieu d'utiliser un langage formel. Le SBVR définit deux types de formulation sémantique : les « formulations logiques » qui structurent les propositions (les règles métier), et les « projections » qui formulent des définitions, des agrégations et des questions. Ce dernier est hors du cadre de cet thèse. Alors, dans cette sous-section, nous détaillons comment les formulations logiques seront identifiées, à savoir : la formulation atomique, la formulation d'instanciation, la formulation modale, la formulation restrictive, les conjonctions, les quantifications et la négation.

- *Formulation atomique :*

Tous les concepts verbaux qui sont écrits selon le pattern P19 seront mappés en formulation atomique, avec NN1 comme leur premier rôle, et NN2 comme leur deuxième rôle.

- *Formulation d'instanciation :*

Tous les concepts verbaux qui sont écrits selon le pattern P20 sont mappés en formulation d'instanciation, avec NN1 comme « concept considéré » et NN2 comme cible.

- *Formulation de restriction :*

Les formulations logiques sont récursives. C'est-à-dire que plusieurs types de formulations logiques incorporent d'autres formulations logiques. Par exemple, une formulation logique pourrait être utilisée comme « formulation restrictive » qui définit les concepts d'une autre formulation atomique comme dans l'exemple suivant '*site that is used for rental business*' qui contient une restriction du concept de nom « site ». Pour cette raison, nous essayons d'extraire les « clauses relatives » écrite selon le pattern P21 et de les mapper en « formulation restrictive » des noms.

- *Conjonction / Disjonction*

Chaque deux concepts verbaux 'p' et 'q' qui sont liés avec une relation de conjonction / disjonction (pattern P22) seront mappés en formulation logique binaire ayant 'p' comme 'opérande logique 1' et 'q' comme 'opérande logique 2'.

- *Implication :*

Chaque clause 'p' ayant une « clause adverbiale » (advcl) 'q' comme dans le pattern P23 sera mappée en 'implication', avec 'p' comme son 'antécédent' et 'q' comme son 'conséquent'.

- *Négation :*

Les clauses extraites peuvent contenir une négation. Pour cela, nous avons défini trois patterns P24, P25 et P26 pour les identifier.

- *Quantification :*

Les quantifications sont des formulations logiques introduisant exactement une variable incluse dans une formulation logique. La Table 15 contient des exemples de quantifications prises en compte par notre approche.

Table 15 Les expressions de quantifications prises en charge par notre approche et leurs équivalences en SBVR

Expression	**Formulation Logique**
- A, each, all	Universal quantification
- at least n - no less than	at-least-n quantification
- at most n - no more than	at-most-n quantification
- more than n	more-than-n quantification
- less than n	less-than-n quantification
- exactly n	Exactly-n quantification

Par exemple, la déclaration '*each bad experience has at least one notification date-time*' est formulée par une formulation atomique ayant deux "role binding". Le premier est lié au concept '*bad experience*' qui est introduit par la quantification universelle 'each', tandis que le second est lié à la variable 'notification date-time' qui est introduite par la quantification existentielle 'at least one'.

Dans notre approche, nous nous sommes basés sur la reconnaissance d'entité nommée (NER ou Named Entity Recognition) affectée aux modificateurs numériques (nummod) des concepts de noms (s'il en existe), qui est générée par l'outil d'analyse. (Voir pattern P27)

- *Formulation modale :*

Les auxiliaires modaux attachés aux verbes sont mappés à des formulations modales comme dans le pattern P28. Sinon, la modalité pourrait également être trouvée sous forme de clause au début de la déclaration tel que «It is obligatory that…». La Table 16 comprend quelques expressions et leur équivalence en SBVR.

Table 16 List des expressions de modalité prises en charge par notre approche et leurs équivalences en SBVR

Expression	Logical Formulation
- …must… - It is obligatory that…	Obligation formulation
- …must not… - It is prohibited that…	Obligation formulation embedding logical negation
- …always… - It is necessary that…	Necessity formulation
- …never… - It is impossible that…	Necessity formulation embedding logical negation
- …may… - It is permitted that…	Permissibility formulation

6.1.5. La génération du fichier XMI

Finalement, la formulation sémantique de chaque règles métier ainsi que le dictionnaire terminologique contenant les concepts identifiés et leurs spécifications sont enregistrés dans un fichier XMI qui respect le schéma proposé dans la documentation officielle de SBVR.

Notre fichier généré pourrait être utilisé pour générer automatiquement une description textuelle structurée du vocabulaire commercial et des règles compréhensibles par tous les utilisateurs ainsi que par les machines. De plus, le fichier XMI rendra notre approche facilement intégrable dans d'autres approches basées sur les règles métiers textuelles.

Pour afficher le dictionnaire terminologique, la Table 17 présente la correspondance entre les éléments SBVR et l'annotation SBVR-SE proposées dans la documentation officielle.

Table 17 Les élément du dictionnaire terminologique de SBVR et leurs équivalences en notation SBVR-SE

Le métamodèle SBVR	**Légende SBVR-SE**
Owned definition (Intentional & Extensional)	Definition
Adopted definition	Dictionary Basis
Relation: Concept1 Specializes Concept 2	General Concept
Necessity statements (about our concept)	Necessity
Possibility statements (about our concept)	Possibility
Relation: Thing1 is Thing2 (for Noun Concept)	Synonym
Relation: Thing1 is Thing2 (for Verb Concept)	Synonymous Form
Preferred designation	See

Un exemple sur la façon dont les éléments identifiés sont enregistrés dans le fichier XMI peut être trouvé dans l'exemple illustratif de la section suivante.

6.2. Exemple illustratif

Différents éléments SBVR sont identifiés par notre approche en utilisant différents patterns à chaque étape. Ainsi, pour donner un exemple complet, nous avons besoin d'un grand nombre de règles métier pour couvrir toutes les étapes de notre approche. Pour cette raison, nous prendrons juste la règle métier complexe suivante et nous montrerons comment sa sémantique pourrait être identifiée en utilisant nos patterns :

Exemple 1 : '*The local area is the organization unit that manages exactly three branches and some service depots if the local area is an active area*'.

L'arbre d'analyse grammaticale et le POS-tagging effectués par l'outil d'analyse à la première étape sont illustrés dans la Figure 26.

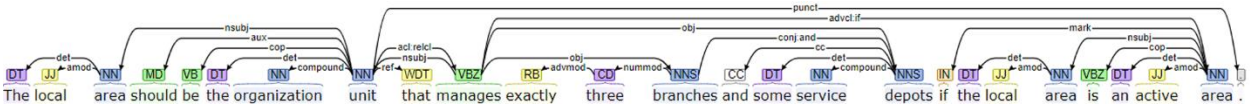


Figure 26 Exemple de résultat généré lors de l'analyse linguistique

En appliquant notre approche sur cet exemple, nous avons pu identifier 8 concepts de nom, 10 concepts de verbe, 3 concepts généraux, 2 formulations atomiques, 6 formulations d'instanciation, 1 formulation de restriction, 1 conjonction, 1 formulation d'implication et 1 quantification. L'exemple ne contient ni modalité ni formulation de négation. De plus, la déclaration contient une clause dépendante qui ne peut être utilisée comme définition d'aucun concept. Enfin, pour montrer comment les termes synonymes sont identifiés, nous avons besoin d'au moins six règles métier (trois règles pour chaque terme), car notre algorithme d'extraction de synonymes est sensible à la fréquence des termes. Ainsi, les synonymes ne sont pas non plus identifiés dans l'exemple. Les détails sur les éléments SBVR identifiés ainsi que les patterns utilisés à cet effet sont présentés dans la Table 18.

Table 18 Exemple des éléments SBVR identifiés par notre approche

SBVR	Id	Output
General concept		- En utilisant le pattern P7: (NN + modifier)
	gc-1	"Local area"
	gc-2	"Active area"
		- En utilisant le pattern P8 : (Compound NN)
	gc-3	"Organization unit"
	gc-4	"Service depot"

		- Les noms simples : NN
	gc-5	'branch'
	gc-6	'area'
	gc-7	'unit'
	gc-8	'depot'
Necessity statement	nec-x	Tous les concepts identifiés dans cette règle vont recevoir cette même règle comme une déclaration nécessaire pour eux.
Atomic Formulation	af-1	- En se basant sur le concept de verbe 'vc-9' : 'af-1' is an atomic formulation that is based on the verb concept 'vc-9'. 'af-1' has a role which binds to a first variable ranging over the concept 'gc-3. 'af-1' has a second role which binds to a second variable ranging over concept 'gc-5'
	af-2	- En se basant sur le concept de verbe 'vc-10': 'af-2' is an atomic formulation that is based on the verb concept 'vc-10'. 'af-2' has a role which binds to a first variable ranging over the concept 'gc-3. 'af-2' has a second role which binds to a second variable ranging over concept 'gc-4'
Restricting Formulation	rf-1	- En utilisant le pattern P21 : The variable ranging over the concept 'gc-3' (*organization unit*) is restricted by the logical formulation '*conj-1*' (af-1 & af-2).
Conjunction	conj-1	- En utilisant le pattern 21 : 'p' and / or 'q' The binary logical operation 'conj-1' has two logical operands: operand 1: 'af-1' operand 2: 'af-2'

Instantiation Formulation	if-1	- En se basant sur le concept de verbe : 'vc-1' 'if-1' is an instantiation formulation which considers the concept 'gc-1'. 'if-1' binds to the noun concept 'gc-3'
	if-2	- En se basant sur le concept de verbe : 'vc-2' 'if-2' is an instantiation formulation which considers the concept 'gc-1'. 'if-1' binds to the noun concept 'gc-2'
	if-3	- En se basant sur le concept de verbe : 'vc-3' 'if-3' is an instantiation formulation which considers the concept 'gc-1'. 'if-3' binds to the noun concept 'gc-6'
	if-4	- En se basant sur le concept de verbe : 'vc-5' 'if-4' is an instantiation formulation which considers the concept 'gc-2'. 'if-4' binds to the noun concept 'gc-6'
	if-5	- En se basant sur le concept de verbe : 'vc-7' 'if-5' is an instantiation formulation which considers the concept 'gc-3'. 'if-5' binds to the noun concept 'gc-7'
	if-6	- En se basant sur le concept de verbe : 'vc-8' 'if-6' is an instantiation formulation which considers the concept 'gc-4'. 'if-6' binds to the noun concept 'gc-8'
Implication	imp-1	- En utilisant le pattern P23 : The implication 'imp-1' has: *antecedent*: 'if-1' *consequence*: 'if-2'
Quantification	qtf-1	- En utilisant le pattern P27 : The quantification "*exactly three*" introduces a variable ranging over the concept 'gc-5' The quantification scopes over an "exactly-n" quantification. exactly-n quantification has cardinality '3'

Definition		No
Modality		No
Synonym		No
Negation		No

La sémantique de la règle textuelle est ensuite sauvegardée dans un fichier XMI respectant le standard SBVR en utilisant les patterns cités dans la documentation officielle (la Figure 27 est un extrait). Sur la base de ce fichier XMI générés, nous pouvons facilement générer un fichier MS-Word pour le dictionnaire terminologique.

```
<!--General Concept-->
<sbvr:text  xmi:id="txt-gc-3"  value="Organization unit"/>
<sbvr:generalConcept  xmi:id="gc-3"/>
<sbvr:designation  xmi:id="des-gc-3"  signifier="txt-gc-3"  meaning="gc-3"/>
<!--General Concept-->
<sbvr:text  xmi:id="txt-gc-4"  value="Service depot"/>
<sbvr:generalConcept  xmi:id="gc-4"/>
<sbvr:designation  xmi:id="des-gc-3"  signifier="txt-gc-4"  meaning="gc-4"/>
<!--Statement-->
<sbvr:statement xmi:id="stmt-1" expression="txt-stmt-1" meaning="prp-1"/>
<sbvr:text xmi:id="txt-stmt-1" value="The local area is the
organization unit that manages exactly three branches and some
service depots if the local area is an active area"/>
<sbvr:proposition xmi:id="prp-1" isNecessarilyTrue="true"/>
<sbvr:propositionIsBasedOnVerbConcept proposition="prp-1" verbConcept="vc-9"/>
<!--LogicalFormulation-->
<sbvr:closedLogicalFormulation xmi:id="clf-1"/>
<sbvr:closedLogicalFormulationFormalizesStatement closedLogicalFormulation="clf-1" statement="stmt-1"/>
<sbvr:closedLogicalFormulationMeansProposition closedLogicalFormulation="clf-1" proposition="prp-1"/>
<sbvr:obligationFormulation xmi:id="obf-1"/>
<sbvr:modalFormulationEmbedsLogicalFormulation modalFormulation="obf-1" logicalFormulation="imp-1"/>
<sbvr:thing1IsThing2 thing1="obf-1" thing2="clf-1"/>
<!--Implication-->
<sbvr:implication xmi:id="imp-1" />
<sbvr:implicationHasAntecedent implication="imp-1" antecedent="if-1"/>
<sbvr:implicationHasConsequent implication="imp-1" consequent="if-2"/>
<!--Atomic formulation-->
<sbvr:atomicFormulation xmi:id="af-1" roleBinding="vcr-9-1 vcr-9-2"/>
<sbvr:atomicFormulationIsBasedOnverbConcept atomicFormulation="af-1" verbConcept="vc-9"/>
<!--Verb Concept-->
<sbvr:verbConcept  xmi:id="vc-9" />
<sbvr:sententialForm xmi:id="sf-9" expression="txt-sf-9"
    placeholder="ph-sf-9-p1 ph-sf-9-p2" meaning="vc-9"/>
<sbvr:text xmi:id="txt-sf-9" value="organization unit manage branch"/>
<!-- Verb Concept Wording-->
<sbvr:text  xmi:id="txt-vs-9"  value="...manage..."/>
<sbvr:verbSymbol  xmi:id="vs-9"  signifier="txt-vs-9"  meaning="vc-9"/>
<sbvr:verbConceptWordingIncorporatesVerbSymbol verbConceptWording="sf-9" verbSymbol="vs-9"/>
<!-- PlaceHolders -->
<sbvr:placeholder xmi:id="ph-sf-9-p1" meaning="vcr-9-1"/>
<sbvr:placeholder xmi:id="ph-sf-9-p2" meaning="vcr-9-2"/>
<sbvr:placeholderUsesDesignation placeholder="ph-sf-9-p1" designation="des-gc-3"/>
<sbvr:placeholderUsesDesignation placeholder="ph-sf-9-p2" designation="des-gc-4"/>
<!-- VerbConceptRole -->
<sbvr:verbConceptRole xmi:id="vcr-9-1"/>
<sbvr:roleRangesOverGeneralConcept role="vcr-9-1" generalConcept="gc-3"/>
<sbvr:verbConceptRole xmi:id="vcr-9-2"/>
<sbvr:roleBinding xmi:id="rb-gc-4"/>
<sbvr:variable xmi:id="v-gc-4" ranged-overConcept="gc-4" restrictingFormulation=""/>
<sbvr:verbConceptRoleHasRoleBinding verbConceptRole="vcr-9-2" roleBinding="rb-gc-4"/>
<sbvr:roleBindingBindsToBindableTarget roleBinding="rb-gc-4" bindableTarget="v-gc-4"/>
<!--Quantification-->
<sbvr:exactly-nQuantification xmi:id="qtf-1" scopeFormulation="af-1" minimumCardinality="i3"/>
<sbvr:quantificationIntroducesVariable quantification="qtf-1" variable="v-gc-4"/>
<sbvr:positiveInteger xmi:id="i3" value="3"/>
```

Figure 27 Extrait du fichier SBVR-XMI généré contenant la sémantique de la règles métier

6.3. Evaluation de l'approche

Pour évaluer notre approche, une expérience a été menée sur trois ensembles différents de règles métier textuelles sélectionnées aléatoirement depuis différentes sources comme suit :

- Cas 1 [67] : 53 règles métier concernant les préférences du client.
- Cas 2 [4] : 53 règles métier concernant la réservation de vol.
- Cas 3 [66]: 46 règles métier concernant les prêts.

Pour éviter les problèmes qui peuvent résulter de l'ambiguïté du langage naturel qui peuvent à leur tour affecter les résultats, nos règles métier devrait être bien formées et sans ambiguïté. Ainsi, nous avons reformulé certaines règles qui ne correspondent pas aux recommandations proposées par [4], comme les règles qui utilisent des pronoms ou qui incluent à la fois les conjonctions et les disjonctions.

Pour obtenir notre table de vérité, nous avons tout d'abord extrait le vocabulaire métier manuellement, avec les spécifications ainsi que la sémantique des règles métier incluses dans chaque cas selon le standard SBVR. Ensuite, nous avons comparé les résultats avec d'autres générés automatiquement en utilisant notre méthode. Enfin, nous avons calculé le nombre de Vrai Positif (VP), Faux Positif (FP), Faux Négatif (FN), Précision (Eq 1), Rappel (Eq 2) et le F1-Score (Eq 3) en utilisant les équations suivantes :

$$\boldsymbol{Précision} = \frac{\boldsymbol{VP}}{\boldsymbol{VP + FP}} \qquad (\boldsymbol{Eq\ 1})$$

$$\boldsymbol{Reppell} = \frac{\boldsymbol{VP}}{\boldsymbol{VP + FN}} \qquad (\boldsymbol{Eq\ 2})$$

$$\boldsymbol{F1 - score} = \frac{\boldsymbol{2 \times Précision \times Rappal}}{\boldsymbol{Précision + Rapell}} \qquad (\boldsymbol{Eq\ 3})$$

Dans le reste de cette section, nous montrerons - étape par étape - à quelle point notre approche pourrait générer à la fois du vocabulaire des règles métier selon le standard SBVR.

6.3.1. L'analyse linguistique

Chaque ensemble de règles est enregistré dans un fichier texte distinct, dans lequel chaque règle est écrite sur une ligne distincte se terminant par un point. Ensuite, chaque fichier a été analysé linguistiquement avant de générer le fichier XML contenant le POS-tagging ainsi que l'arbre d'analyse des dépendances de chaque règle. En conséquence, 96% des règles (tous cas inclus) étaient correctement analysés linguistiquement par l'outil Stanford CoreNLP, répartis comme suit : 96% dans le cas-1, 93% dans le cas-2 et 98% dans le cas-3. Cependant, la génération de 4% des règles erronées est considéré comme très intéressantes. La source des erreurs à cette première étape était soit une ponctuation mal placée, soit certains mots qui sont généralement source d'ambiguïté comme «requests» et «references» qui pourraient être utilisées à la fois comme

verbe et comme nom, ou certaines règles formulées à l'aide d'une formulation sémantique qui n'est pas pris en considération par notre approche (e.g. la formulation 'whether-or-not').

La deuxième partie de cette étape d'analyse linguistique concerne la dissection de clauses qui composent les phrases formulant les règles pour en identifier ceux qui sont implicites. Comme mentionné précédemment, nous essayons de disséquer les clauses ayant plus d'un sujet ou d'objet liés par des conjonctions. Bien que tous les patterns définis étaient correctement identifiés, certaines clauses contenant plus d'un seul verbe, liés au même sujet et objet n'ont pas été identifiées. Par exemple, la déclaration suivante "*A passenger may check-in and board a flight*" comporte deux verbes en relation de conjonction qui ont le même sujet et objet. Lors de l'analyse linguistique, nous identifions deux clauses : la première n'a que le sujet et le verbe « *passenger check-in*» et une seconde clause qui n'a que le verbe et l'objet «*board flight*». La source d'erreur à ce niveau est que la règle n'est pas aussi simple que notre approche exige. Ce type de problèmes a été identifié trois fois, c'est pour cela, un pattern prenant en considération cette forme de clauses composées doit être ajouté, sinon les clauses doivent être plus simples en ajoutant un sujet et un objet explicites pour chaque verbe sans se soucier de la redondance des termes.

Le résultat généré à cette étape est enregistré dans un fichier XML qui sera utilisé à l'étape suivante pour générer à la fois le vocabulaire métier et les règles métier.

6.3.2. Extraction du vocabulaire et règles métier

En se basant sur l'analyse linguistique de la première étape, nous allons identifier tous les éléments nécessaires pour générer la sémantique du vocabulaire et des règles métier.

La Table 19 résume les résultats obtenus liés à l'identification automatique des éléments SBVR à partir des règles métier textuelles, à savoir : les concepts de nom, les concepts de verbe, les relations de généralisation, les implications, les quantifications et les synonymes. D'autres éléments ne sont pas cités dans la table car ils sont obtenus à travers un mappage direct à partir des éléments identifiés, à savoir des formulations atomiques et des formulations d'instanciation qui sont mappées à partir des concepts verbaux, et des formulations logiques binaires qui à leur tour sont obtenues à partir des clauses disséquées.

Table 19 Résultat obtenu par notre approche lié à la génération automatique de la sémantique des règles métier

	Cases	Identification manuelle	Identification automatique	VP	FP	FN	Précision	Rappel	F1-Score
General Concept	case1	273	271	261	10	12	96%	96%	96%
	case2	212	209	201	8	11	96%	95%	95%
	case3	138	135	121	14	17	90%	88%	89%
Verb Concept	case1	282	276	274	2	8	99%	97%	98%
	case2	181	178	175	3	6	98%	97%	97%
	case3	96	91	87	4	9	96%	91%	93%
Generalization	case1	19	17	15	2	4	88%	79%	83%
	case2	33	30	24	6	9	80%	73%	76%
	case3	27	27	22	5	5	81%	81%	81%
Implication	case1	53	49	49	0	4	100%	92%	96%
	case2	7	6	6	0	1	100%	86%	92%
	case3	20	18	18	0	2	100%	90%	95%
Quantification	case1	12	11	9	2	3	82%	75%	78%
	case2	13	11	8	3	5	73%	62%	67%
	case3	14	13	12	1	2	92%	86%	89%
Synonym	case1	22	21	16	5	6	76%	73%	74%
	case2	20	20	17	3	3	85%	85%	85%
	case3	15	16	14	2	1	88%	93%	90%

- *Extraction du vocabulaire métier SBVR*

Dans cette sous-section, les éléments nécessaires pour le dictionnaire terminologique basé sur SBVR sont extraits, contenant le vocabulaire métier ainsi que d'autres spécifications. Pour cela, nous avons utilisé nos deux méthodes expliquée dans les deux derniers chapitres.

Ce qui suit sont les résultats obtenus pour chaque élément identifié.

– Concepts de noms

Les concepts généraux ont été identifiés avec un F1-score compris entre 89% et 96%. Les concepts généraux composés d'un seul terme ont été moins efficacement identifiés que les concepts composés de plusieurs termes, car ils sont plus exposés à l'ambiguïté comme le terme «Gold» qui a été étiqueté comme un nom dans certaines phrases et comme un adjectif dans d'autres.

Les concepts individuels ne sont pas discutés car nous n'avons défini aucune stratégie pour les identifier, sauf ce que l'outil de Stanford a détecté.

– Concepts verbaux

Les concepts verbaux ont été identifiés avec un F1-score compris entre 93% et 98%. Le problème principal à ce niveau est que certains modes verbaux ne sont pas pris en compte par notre approche comme dans "*An applicant must be having adequate funding*" qui devrait être "*An applicant must have adequate funding*" pour qu'elle puisse être identifiée par notre approche. De plus, certaines règles ont utilisé des verbes de liaison qui ne sont pas inclus dans notre métamodèle générique comme le verbe «*become*» dans la règle "*A shipment must be assigned to a packer, when it becomes fillable*".

Un autre élément clé à mentionner est que 52% de tous les concepts verbaux identifiés ont été extraits depuis des clauses composées et des concepts de noms composés de plusieurs termes (54% dans le premier cas, 65%, dans le deuxième cas et 24% dans le troisième cas). Cela reflète la quantité d'informations manquées dans d'autres approches qui négligent les clauses implicites.

– Définitions

Dans le premier cas, toutes les règles métier sont exprimées en utilisant une formulation d'implication. Ainsi, aucun d'entre eux ne respecte nos patterns pouvant être utilisés comme définition de concept. Pour cette raison, nous n'avons pu affecter aucune définition à aucun concept de nom dans ce cas. Alors que 11 concepts de noms ont reçu une définition dans le deuxième cas, contre 8 dans le troisième cas. En fait, la recherche de définitions pour les concepts de noms est basée sur la quantité des règles définitionnels utilisées.

– Relations de généralisation

Les relations de généralisation / spécialisation ont été identifiées avec un F1-score compris entre 81% et 83%. Certaines relations identifiées étaient erronées, car certaines règles contiennent plus d'une expression pour le même concept, comme dans "*client preference*" et "*preference*", ce qui a généré le fait que le deuxième est le concept général du premier selon notre pattern. D'autre part, certains noms composés erronés ont généré une relation de généralisation erronée entre les concepts de noms, tels que "*same flight*", qui a été détecté comme une spécialisation du concept "*flight*".

– Synonymes

Les règles sont généralement rédigées par différents intervenants. Ainsi, pour imiter ce contexte, nous avons pris environ 30% des règles métier de chaque cas et nous les avons envoyées à deux ingénieurs différents pour qu'ils reformulent certaines règles en y injectant des synonymes et des abréviations.

Comme indiqué précédemment, la plupart des approches traitant des spécifications textuelles ne faisaient confiance qu'aux dictionnaires pour identifier les synonymes. Alors que dans notre approche, nous utilisons l'algorithme LIN pour interroger la base de données WordNet et identifier les termes synonymes potentiels, avant d'affiner le résultat obtenu en utilisant des heuristiques. La Figure 28 montre les impacts positifs de nos heuristiques sur chaque cas.

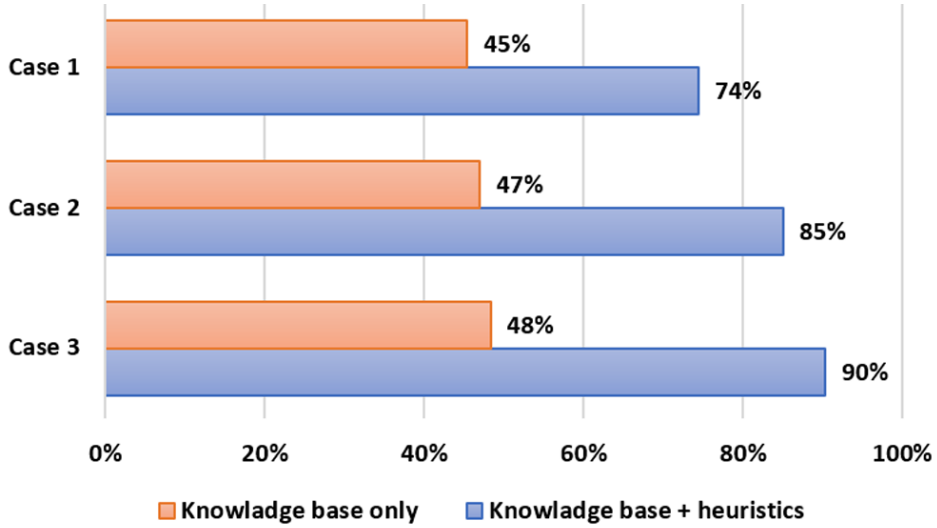


Figure 28 L'impact de nos heuristiques sur l'amélioration du F1-score pour l'identification des synonymes

Notre approche a pu extraire des synonymes avec un F1-score compris entre 74% et 90%. Le principal problème à ce niveau est que notre approche vise à détecter uniquement les synonymes composés de même nombre de mots. En revanche, les noms ayant des sens différents par rapport au dictionnaire sont hors du champ d'application de notre approche, tels que «*account balance*» et «*current balance*».

Un autre élément clé qui a affecté notre résultat est que nos heuristiques sont sensibles à la fréquence des termes. Autrement dit, plus les termes sont fréquents, plus ils ont une grande chance de détecter leurs synonymes, car ils reflètent leur contexte réel. La Figure 29 montre que le troisième cas utilise des termes plus fréquents par rapport aux autres cas. Pour cette raison, nous avons pu détecter 90% des synonymes injectés, car les termes sont utilisés plusieurs fois dans différentes règles, contrairement aux premiers et deuxièmes cas.

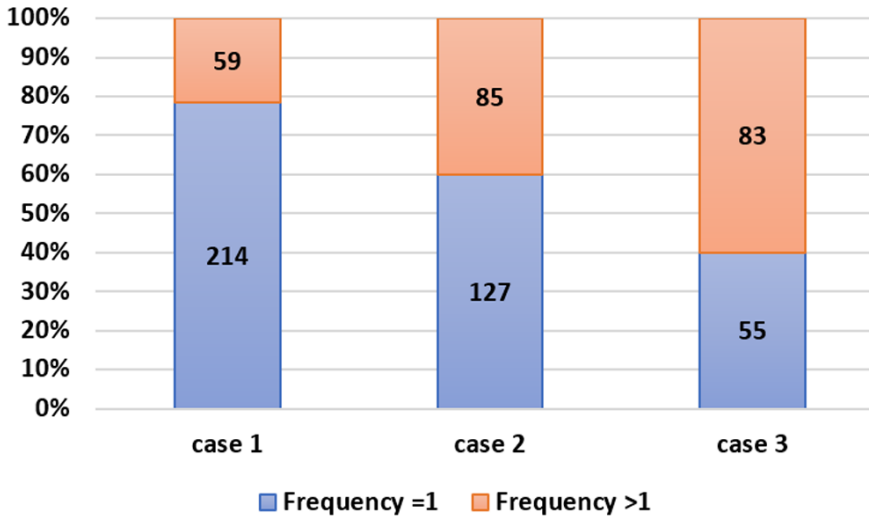


Figure 29 Le nombre des concepts de noms identifiés par notre approche comparé avec leurs fréquences

- *Extraction des formulations sémantiques SBVR*

– Modalité

Toutes les formulations de modalités ont été parfaitement extraites, à l'exception de celle dans la règle suivante "*if the flight booking confirmation is international, then it is obligatory that the flight booking confirmation have exactly one set of passport details for each passenger of the flight booking confirmation*.". Cette règle comprend une modalité sous forme de clause "*it is obligatory that*" au milieu de la phrase, contrairement à ce que notre approche exige. Ainsi, cette forme de formulation des modalités devrait également être prise en considération dans les travaux futurs. Enfin, 40 instructions contenant une formulation de modalité ont été identifiées dans le troisième cas, 39 formulations dans le deuxième cas, tandis qu'aucune expression de modalité n'a été incluse dans le premier cas.

Les déclarations extraites à cette étape ont été attachée en tant que règles nécessaire ou possible à 18 concepts de nom dans le deuxième cas, et à 26 concepts de nom dans le troisième cas.

– Implication

Notre méthode a pu identifier des formulations d'implication avec 100% de précision et 95% en F1-score. Le principal problème à ce niveau était une mauvaise analyse grammaticale de quelque règles complexes ayant de nombreuses clauses en relation de conjonction. Certaines autres règles ont utilisé des expressions spéciales qui n'ont pas été détectées comme des clauses adverbiales telles que «*p à condition que q*».

– Quantification

Toutes les formulations de quantification définies dans notre méthode ont été correctement identifiées. Cependant, certaines expressions devraient également être prises en considération, telles que «*exceeding*» et «*x to y*». En conséquence, les quantifications ont été identifiées avec un F1-score compris entre 67% et 89%.

– Négation

Nos modèles définis pour les formulations de négation étaient suffisants pour identifier parfaitement toutes les 57 situations de négation trouvées dans toutes les règles avec un F1-score de 100% (24 dans le cas-1, 13 dans le cas-2 et 16 dans le cas-3). Les patterns utilisés dans les formulations de négation sont pratiquement exhaustifs car ils sont souvent basés uniquement sur le terme «*not*» lié au verbe principal. Compte tenu de ce qui précède, la formulation de la négation n'a pas été incluse dans le tableau des résultats.

En guise de dernière remarque, même s'il existait des patterns et des expressions qui ne sont pas couverts par notre approche, nous avons pu obtenir des résultats encourageants. Quelques améliorations à ce niveau produiront sans aucun doute des résultats plus précis.

6.4. Menaces de validité

Malgré nos résultats optimistes, les menaces suivantes de la validité de notre approche doivent être prises en compte :

- *Menaces internes* : examinent l'impact des facteurs internes sur les résultats.
- *Menaces externes* : limitent la généralisation des résultats au-delà du cadre de l'expérience.
- *Menaces de conclusion* : examinent les facteurs menant à une conclusion incorrecte.

6.4.1. Menaces internes

Notre approche est conçue pour analyser les règles textuelles bien formées et sans ambiguïté. Pour cette raison, les règles textuelles qui ne sont pas rédigées en respectant les bonnes pratiques ont moins de chance d'être correctement traitées. Cette menace était minime dans nos expériences, car certaines règles ont été reformulées par les auteurs de ce manuscrite pour enlever l'ambiguïté avant d'être injectées dans le pipeline de notre approche.

D'autre côté, notre approche est destinée à traiter des règles écrites en langage naturel (informels), ce qui rend difficile de deviner leur structure grammaticale. Cette menace était également minime dans nos expériences, car les structures des déclarations textuelles les plus utilisées dans différents livres spécialisés dans le domaine [66] [4] [67] sont injectées dans nos échantillons, à l'exception de celles précédemment citées comme non supportées par l'approche.

6.4.2. Menaces externes

Dans notre approche, chaque étape est basée sur le résultat généré par son étape précédente, à l'exception de la première étape qui est totalement basée sur le résultat généré par l'outil d'analyse. Heureusement, nous avons adopté l'outil Stanford CoreNLP comme l'un des outils modernes, régulièrement mis à jour avec des analyses grammaticales de haute qualité. Cela se voie clairement dans le fait que seuls 4% des règles métier ont été mal analysées.

WordNet est un autre outil utilisé dans notre approche pour identifier les synonymes. Il est considéré comme la base de données la plus populaire dans ce domaine. Cependant, les résultats générés ont été affinés à l'aide des heuristiques.

6.4.3. Menaces de conclusion

La plus grande préoccupation de nos expériences est la taille de nos échantillons qui ne semble pas suffisante pour tirer des conclusions précises. Nos échantillons contiennent trois ensembles de règles métier provenant de trois sources différentes et contenant environ 50 règles métier dans chacun d'eux. Nous avons essayé de sélectionner au hasard des règles qui représentent toutes sortes de structures de phrases (simples, composées et complexes) ; ainsi, nous considérons que toute augmentation du nombre de règles métier n'affectera pas trop le résultat obtenu.

Conclusion

Nous avons proposé une méthode pour transformer automatiquement les règles métier textuelles en format respectant le standard SBVR compréhensible par la machine ainsi que les différentes parties prenantes, y compris les non spécialistes IT. Dans cette méthode, un traitement approfondi du langage naturel a été utilisé pour identifier les informations explicites et implicites des règles métier. Cette méthode est plus complète par rapport aux autres approches qui essayent de faire des transformations Text-to-SBVR, car nous avons extrait non seulement les concepts de nom et les concepts de verbe avec certaines formulations logiques, mais également un dictionnaire terminologique plus complet qui contient des spécifications supplémentaires sur les concepts extraits tels que les définitions et les synonymes. De plus, notre approche n'utilise aucun assistant ou information préexistante, ce qui la rend entièrement automatique. Nous nous sommes davantage concentrés sur les règles fournissant des informations pertinentes sur chaque concept extrait ainsi que sur les relations entre eux, pour encourager la génération du dictionnaire terminologique le plus complet. Par conséquent, pour prendre en charge d'autres types de règles métier, d'autres extensions devraient être ajoutées à notre métamodèle générique du langage naturel, ainsi qu'à notre métamodèle SBVR cible, en ajoutant des formulations logiques spéciales telles que les formulations «nand», «nor» et «whether-or-not» . Un autre problème qui doit être régler concerne nos patterns statiques qui devraient être modifiés afin que les utilisateurs puissent facilement les paramétrer. Cependant, les résultats obtenus avec un F1-score moyen de 87% nous ont incités à étendre cette approche afin qu'elle couvre d'autres éléments SBVR tels que les « projections ». Finalement, nous considérons que notre méthode génère le modèle SBVR le plus complet à partir

des règles métier textuelles et pourrait être utilement employée pour faciliter l'intégration de ces règles dans les infrastructures informatiques actuelles.

Conclusion et perspectives

La contribution de cette thèse était de générer le dictionnaire terminologique ainsi que la sémantique des règles métier écrite en langage nature, et les représenter selon le standard SBVR. Ce dernier a été choisi comme modèle pivot pour combler le fossé entre les experts IT et les experts métier afin de rendre possible l'intégration de ces derniers dans la validation des spécifications textuelles. Le standard SBVR est connu par son format compréhensible par toutes les parties prenantes y compris la machine, ce qui va nous permettre d'exploiter les règles métier ainsi que toutes les spécifications textuelles dans la phase d'Inception et les intégrer directement dans l'architecture MDA. D'autre part, cela va nous permettre aussi de filtrer les règles métier avant de les exploiter dans la création des modèles de la phase CIM.

En conclusion, les résultats de nos travaux soutiennent les idées suivantes :

- Fournir une signification plus complète des règles métier textuelles peut supprimer le fossé entre les experts métier et les experts IT, et accélérer le processus de développement logiciel, en particulier dans ses premières phases.
- Se contenter uniquement des règles métier textuelles pour générer automatiquement leur sémantique sans aucune intervention humaine est possible.
- Le respect des meilleures pratiques en matière de rédaction des règles métier textuelles est la clé d'une transformation Texte-to-modèle réussie.

La génération du modèle SBVR depuis des spécifications textuelles peut ne pas être limitée à seulement l'identification des concepts et les entités-relations, mais peut également être étendue pour couvrir d'autres spécifications incluses dans le métamodèle SBVR telles que les définitions de concepts, les synonymes et les formulations sémantiques.

Malgré qu'on ne puisse pas les quantifier, les gains potentiels de nos contributions comprend :

- Moins de défauts dans les spécifications textuelle.
- Moins de problèmes de communication.
- Développement et livraison plus rapides.
- Réduction des coûts d'amélioration.
- Réduction des retouches de développement.

En conclusion, d'autres extensions devraient être ajoutées à notre métamodèle source ainsi qu'à notre métamodèle SBVR cible, en ajoutant plus de formulations logiques pour pouvoir couvrir plus de patterns. De même, l'utilisation des expressions régulières pour la spécification des patterns va rendre notre méthode plus flexible. D'autre part, la création d'une application englobante toutes ces contributions va permettre aux autres chercheurs de comparer la performance de leurs méthodes avec la nôtre.

Bibliographie

[1] « The Standish Group ». https://www.standishgroup.com/ (consulté le mars 24, 2021).

[2] M. Luisa, F. Mariangela, et N. I. Pierluigi, « Market research for requirements analysis using linguistic tools », *Requirements Eng*, vol. 9, nº 1, p. 40-56, févr. 2004, doi: 10.1007/s00766-003-0179-8.

[3] « Model Driven Architecture (MDA) | Object Management Group ». https://www.omg.org/mda/ (consulté le mai 02, 2020).

[4] G. C. Witt, *Writing effective business rules a practical method*. 2012.

[5] B. Bousetta, O. El Beggar, et T. Gadi, « A methodology for CIM modelling and its transformation to PIM », *Journal of Information Engineering and Applications*, vol. 3, nº 2, p. 1-21, 2013.

[6] B. Bousetta, O. E. Beggar, et T. Gadi, « A model transformation approach for code generation from State Machine Diagram », *IADIS International Journal on Computer Science and Information Systems*, vol. 9, p. 1-15, 2014.

[7] E. B. Omar, B. Brahim, et G. Taoufiq, « Automatic code generation by model transformation from sequence diagram of system's internal behavior », *detail*, vol. 1, nº 02, 2012.

[8] B. Bousetta, O. El Beggar, et T. Gadi, « Automating Software Development Process: Analysis-PIMs to Design-PIM Model Transformation », *IJSEIA*, vol. 7, nº 5, p. 167-196, sept. 2013, doi: 10.14257/ijseia.2013.7.5.17.

[9] A. Kriouile, T. Gadi, et Y. Balouki, « CIM to PIM transformation: a criteria based evaluation », *International Journal of Computer Technology and Applications*, vol. 4, nº 4, p. 616, 2013.

[10] B. Bousetta, O. El Beggar, et T. Gadi, « Generating operations specification from domain class diagram using transition state diagram », *international journal of computer and information technology*, vol. 2, p. 29-36, 2013.

[11] N. Addamssiri, A. Kriouile, Y. Balouki, et G. Taoufiq, « Generating the PIM Behavioral Model from the CIM using QVT », *JCSIT*, vol. 2, nº 3 & 4, 2014, doi: 10.15640/jcsit.v2n3-4a4.

[12] A. Kriouile, N. Addamssiri, T. Gadi, et Y. Balouki, « Getting the static model of PIM from the CIM », in *2014 Third IEEE International Colloquium in Information Science and Technology (CIST)*, Tetouan, Morocco, oct. 2014, p. 168-173, doi: 10.1109/CIST.2014.7016613.

[13] A. Kriouile, T. Gadi, N. Addamssiri, et A. El Khadimi, « Obtaining behavioral model of PIM from the CIM », in *2014 International Conference on Multimedia Computing and Systems (ICMCS)*, Marrakech, Morocco, avr. 2014, p. 949-954, doi: 10.1109/ICMCS.2014.6911383.

[14] A. M. Davis, *Just Enough Requirements Management: Where Software Development Meets Marketing by Alan M. Davis*. Dorset House Publishing Co Inc.,U.S., 2005.

[15] « Semantics Of Business Vocabulary And Rules ». https://www.omg.org/spec/SBVR/About-SBVR/ (consulté le août 03, 2020).

[16] « REQUIREMENT | signification, définition dans le dictionnaire Anglais de Cambridge ». https://dictionary.cambridge.org/fr/dictionnaire/anglais/requirement (consulté le déc. 22, 2020).

[17] B. Lawrence, « Designers must do the modeling », *IEEE Software*, vol. 15, nº 2, p. 31-, mars 1998, doi: 10.1109/52.663782.

[18] I. Sommerville et P. Sawyer, *Requirements engineering: a good practice guide*. Chichester, Eng. ; New York: Wiley, 1999.

[19] D. D. Walden, G. J. Roedler, K. Forsberg, R. D. Hamelin, T. M. Shortell, et International Council on Systems Engineering, Éd., *Systems engineering handbook: a guide for system life cycle processes and activities*, 4th edition. Hoboken, New Jersey: Wiley, 2015.

[20] K. E. Wiegers et J. Beatty, *Software requirements*, Third edition. Redmond, Washington: Microsoft Press, s division of Microsoft Corporation, 2013.

[21] « Amazon.fr - Business Rules and Information Systems: Aligning IT with Business Goals - Tony Morgan - Livres ». https://www.amazon.fr/Business-Rules-Information-Systems-Aligning/dp/0201743914 (consulté le déc. 20, 2019).

[22] « Business Rules Group ». http://www.businessrulesgroup.org/ (consulté le août 02, 2020).

[23] B. Von Halle, *Business rules applied: business better systems using the business rules approach*. New York: Wiley Computer Publishing, 2002.

[24] « OMG | Object Management Group ». https://www.omg.org/ (consulté le déc. 29, 2019).

[25] « About the Semantics Of Business Vocabulary And Rules Specification Version 1.5 ». https://www.omg.org/spec/SBVR/ (consulté le mai 02, 2020).

[26] « RuleSpeak® || Let the Business People Speak Rules! » http://www.rulespeak.com/en/ (consulté le déc. 02, 2020).

[27] T. Yue, L. C. Briand, et Y. Labiche, « A systematic review of transformation approaches between user requirements and analysis models », *Requirements Engineering*, vol. 16, nº 2, p. 75-99, juin 2011, doi: 10.1007/s00766-010-0111-y.

[28] M. G. Ilieva et O. Ormandjieva, « Automatic Transition of Natural Language Software Requirements Specification into Formal Presentation », in *Natural Language Processing and Information Systems*, vol. 3513, A. Montoyo, R. Muńoz, et E. Métais, Éd. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, p. 392-397.

[29] C. Arora, M. Sabetzadeh, L. Briand, et F. Zimmer, « Extracting domain models from natural-language requirements: approach and industrial evaluation », in *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems - MODELS '16*, Saint-malo, France, 2016, p. 250-260, doi: 10.1145/2976767.2976769.

[30] H. M. Harmain et R. Gaizauskas, « CM-Builder: A Natural Language-Based CASE Tool for Object-Oriented Analysis », *Automated Software Engineering*, vol. 10, nº 2, p. 157-181, avr. 2003, doi: 10.1023/A:1022916028950.

[31] V. B. R. V. Sagar et S. Abirami, « Conceptual modeling of natural language functional requirements », *Journal of Systems and Software*, vol. 88, p. 25-41, 2014.

[32] R. Sharma, P. K. Srivastava, et K. K. Biswas, « From natural language requirements to UML class diagrams », in *2015 IEEE Second International Workshop on Artificial Intelligence for Requirements Engineering (AIRE)*, Ottawa, ON, août 2015, p. 1-8, doi: 10.1109/AIRE.2015.7337625.

[33] W. Ben Abdessalem Karaa, Z. Ben Azzouz, A. Singh, N. Dey, A. S. Ashour, et H. Ben Ghazala, « Automatic builder of class diagram (ABCD): an application of UML generation from functional requirements: Automatic Builder of Class Diagram (ABCD) », *Software: Practice and Experience*, vol. 46, nº 11, p. 1443-1458, nov. 2016, doi: 10.1002/spe.2384.

[34] « About the Unified Modeling Language Specification Version 2.5.1 ». https://www.omg.org/spec/UML/ (consulté le mars 20, 2021).

[35] A. Al-Hroob, A. T. Imam, et R. Al-Heisa, « The use of artificial neural networks for extracting actions and actors from requirements document », *Information and Software Technology*, vol. 101, p. 1-15, sept. 2018, doi: 10.1016/j.infsof.2018.04.010.

[36] S. Tiwari, D. Ameta, et A. Banerjee, « An Approach to Identify Use Case Scenarios from Textual Requirements Specification », in *Proceedings of the 12th Innovations on Software Engineering Conference (Formerly Known As India Software Engineering Conference)*, New York, NY, USA, 2019, p. 5:1-5:11, doi: 10.1145/3299771.3299774.

[37] F. Friedrich, J. Mendling, et F. Puhlmann, « Process Model Generation from Natural Language Text », in *Advanced Information Systems Engineering*, juin 2011, p. 482-496, doi: 10.1007/978-3-642-21640-4_36.

[38] D. Sadoun, C. Dubois, Y. Ghamri-Doudane, et B. Grau, « From Natural Language Requirements to Formal Specification Using an Ontology », in *2013 IEEE 25th International Conference on Tools with Artificial Intelligence*, nov. 2013, p. 755-760, doi: 10.1109/ICTAI.2013.116.

[39] R. J. Kate, Y. W. Wong, et R. J. Mooney, « Learning to transform natural to formal languages », in *Proceedings of the 20th national conference on Artificial intelligence - Volume 3*, Pittsburgh, Pennsylvania, juill. 2005, p. 1062-1068, Consulté le: avr. 20, 2020. [En ligne].

[40] J. L. Martínez-Fernández, J. C. González, J. Villena, et P. Martínez, « A Preliminary Approach to the Automatic Extraction of Business Rules from Unrestricted Text in the Banking Industry », in *Natural Language and Information Systems*, Berlin, Heidelberg, 2008, p. 299-310, doi: 10.1007/978-3-540-69858-6_29.

[41] M. Selway, G. Grossmann, W. Mayer, et M. Stumptner, « Formalising natural language specifications using a cognitive linguistic/configuration based approach », *Information Systems*, vol. 54, p. 191-208, 2015.

[42] S. Roychoudhury, S. Sunkle, D. Kholkar, et V. Kulkarni, « From natural language to SBVR model authoring using structured English for compliance checking », in *2017 IEEE 21st International Enterprise Distributed Object Computing Conference (EDOC)*, 2017, p. 73-78.

[43] I. S. Bajwa et M. A. Shahzada, « Automated Generation of OCL Constraints: NL based Approach vs Pattern Based Approach », *Mehran University Research Journal of Engineering and Technology*, vol. 36, nº 2, Art. nº 2, avr. 2017, doi: 10.22581/muet1982.1702.04.

[44] T. Skersys, P. Danenas, et R. Butleris, « Extracting SBVR business vocabularies and business rules from UML use case diagrams », *Journal of Systems and Software*, vol. 141, p. 111-130, juill. 2018, doi: 10.1016/j.jss.2018.03.061.

[45] P. Danenas, T. Skersys, et R. Butleris, « Natural language processing-enhanced extraction of SBVR business vocabularies and business rules from UML use case diagrams », *Data & Knowledge Engineering*, p. 101822, mai 2020, doi: 10.1016/j.datak.2020.101822.

[46] P. K. Chittimalli, C. Prakash, R. Naik, et A. Bhattacharyya, « An Approach to Mine SBVR Vocabularies and Rules from Business Documents », in *Proceedings of the 13th Innovations in Software Engineering Conference on Formerly known as India Software Engineering Conference*, Jabalpur, India, févr. 2020, p. 1-11, doi: 10.1145/3385032.3385046.

[47] V. Gruhn et R. Laue, « Detecting Common Errors in Event-Driven Process Chains by Label Analysis », *Enterprise Modelling and Information Systems Architectures (EMISAJ)*, vol. 6, nº 1, Art. nº 1, 2011, doi: 10.18417/emisa.6.1.1.

[48] A. Awad, A. Polyvyanyy, et M. Weske, « Semantic Querying of Business Process Models », in *2008 12th International IEEE Enterprise Distributed Object Computing Conference*, sept. 2008, p. 85-94, doi: 10.1109/EDOC.2008.11.

[49] C. Fellbaum, Éd., *WordNet: an electronic lexical database*. Cambridge, Mass: MIT Press, 1998.

[50] Y. Zhang, C. Wan, et B. Jin, « An empirical study on recovering requirement-to-code links », in *2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, mai 2016, p. 121-126, doi: 10.1109/SNPD.2016.7515889.

[51] M. Ehrig, A. Koschmider, et A. Oberweis, « Measuring similarity between semantic business process models », in *Proceedings of the fourth Asia-Pacific conference on*

Comceptual modelling - Volume 67, Ballarat, Australia, janv. 2007, p. 71-80, Consulté le: août 03, 2020. [En ligne].

[52] F. Pittke, H. Leopold, et J. Mendling, « Automatic Detection and Resolution of Lexical Ambiguity in Process Models », *IEEE Transactions on Software Engineering*, vol. 41, n° 6, p. 526-544, juin 2015, doi: 10.1109/TSE.2015.2396895.

[53] R. Navigli et S. P. Ponzetto, « BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network », *Artificial Intelligence*, vol. 193, p. 217-250, déc. 2012, doi: 10.1016/j.artint.2012.07.001.

[54] T. Ishioka, « Extended K-means with an Efficient Estimation of the Number of Clusters », in *Intelligent Data Engineering and Automated Learning — IDEAL 2000. Data Mining, Financial Engineering, and Intelligent Agents*, vol. 1983, K. S. Leung, L.-W. Chan, et H. Meng, Éd. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, p. 17-22.

[55] F. Dalpiaz, I. van der Schalk, S. Brinkkemper, F. B. Aydemir, et G. Lucassen, « Detecting terminological ambiguity in user stories: Tool and experimentation », *Information and Software Technology*, vol. 110, p. 3-16, juin 2019, doi: 10.1016/j.infsof.2018.12.007.

[56] « Cortical.io - Next generation of AI business applications ». https://www.cortical.io/ (consulté le août 03, 2020).

[57] B. Hnatkowska et A. Walkowiak-Gall, « Towards Definition of a Unified Domain Meta-model », in *KKIO Software Engineering Conference*, 2018, p. 86-100.

[58] « [PDF] SBVR Business Rules Generation from Natural Language Specification - Semantic Scholar ». https://www.semanticscholar.org/paper/SBVR-Business-Rules-Generation-from-Natural-Bajwa-Lee/34b376490098b61479ee3c6b3477047ed1f8829a (consulté le juin 19, 2019).

[59] F. Lévy et A. Nazarenko, « Formalization of Natural Language Regulations through SBVR Structured English », in *Theory, Practice, and Applications of Rules on the Web*, Berlin, Heidelberg, 2013, p. 19-33, doi: 10.1007/978-3-642-39617-5_5.

[60] J. Arlow et I. Neustadt, *UML and the unified process: practical object-oriented analysis and design*. Boston [u.a]: Addison-Wesley, 2002.

[61] A. Haj, Y. Balouki, et T. Gadi, « Automatic Extraction of SBVR Based Business Vocabulary from Natural Language Business Rules », in *Xenopus*, vol. 1865, K. Vleminckx, Éd. New York, NY: Springer New York, 2019, p. 170-182.

[62] A. Haj, Y. Balouki, et T. Gadi, « Automated generation of terminological dictionary from textual business rules », *J Softw Evol Proc*, mars 2021, doi: 10.1002/smr.2339.

[63] T. Kuhn, « A Survey and Classification of Controlled Natural Languages », *Computational Linguistics*, vol. 40, n° 1, p. 121-170, mars 2014, doi: 10.1162/COLI_a_00168.

[64] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, et D. McClosky, « The Stanford CoreNLP natural language processing toolkit », in *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, 2014, p. 55-60.

[65] H. Diessel, *The acquisition of complex sentences*. Cambridge, U.K. ; New York: Cambridge University Press, 2004.

[66] A. J. Morgan, *Business rules and information systems: aligning IT with business goals*. Boston: Addison-Wesley, 2002.

[67] I. Graham, *Business rules management and service oriented architecture: a pattern language*. Chichester, England ; Hoboken, NJ: John Wiley, 2006.

[68] « Dictionary by Merriam-Webster: America's online dictionary ». https://www.merriam-webster.com/ (consulté le mai 02, 2020).

[69] W. H. Gomaa et A. A. Fahmy, « A Survey of Text Similarity Approaches », *International Journal of Computer Applications*, vol. 68, n° 13, p. 13-18, avr. 2013.

[70] A. Haj, Y. Balouki, et T. Gadi, « Automated Identification of Semantic Similarity between Concepts of Textual Business Rules », *IJIES*, vol. 14, n° 1, p. 147-156, févr. 2021, doi: 10.22266/ijies2021.0228.15.

[71] D. R. Kubal et A. V. Nimkar, « A survey on word embedding techniques and semantic similarity for paraphrase identification », *IJCSYSE*, vol. 5, n° 1, p. 36, 2019, doi: 10.1504/IJCSYSE.2019.098417.

[72] M. Farouk et Department of Thermotechnik Computer Science, Assiut University, Markaz El-Fath, Assiut Governorate 71515,Egypt;, « Measuring Sentences Similarity: A Survey », *Indian Journal of Science and Technology*, vol. 12, n° 25, p. 1-11, juill. 2019, doi: 10.17485/ijst/2019/v12i25/143977.

[73] P. D. Turney et P. Pantel, « From Frequency to Meaning: Vector Space Models of Semantics », *Journal of Artificial Intelligence Research*, vol. 37, p. 141-188, févr. 2010, doi: 10.1613/jair.2934.

[74] T. Mikolov, K. Chen, G. Corrado, et J. Dean, « Efficient estimation of word representations in vector space », *arXiv preprint arXiv:1301.3781*, 2013.

[75] R. Menaha et VE. Jayanthi, « A Survey on Acronym–Expansion Mining Approaches from Text and Web », in *Smart Intelligent Computing and Applications*, Singapore, 2019, p. 121-133, doi: 10.1007/978-981-13-1921-1_12.

[76] J. Pustejovsky, J. Castano, B. Cochran, M. Kotecki, M. Morrell, et A. Rumshisky, « Extraction and disambiguation of acronym-meaning pairs in medline », *Medinfo*, vol. 10, n° 2001, p. 371-375, 2001.

[77] A. S. Schwartz et M. A. Hearst, « A simple algorithm for identifying abbreviation definitions in biomedical text », in *Biocomputing 2003*, World Scientific, 2002, p. 451-462.

[78] K. Taghva et J. Gilbreth, « Recognizing acronyms and their definitions », *International Journal on Document Analysis and Recognition*, vol. 1, n° 4, p. 191-198, 1999.

[79] S. Yeates, « Automatic Extraction of Acronyms from Text. », in *New Zealand Computer Science Research Students' Conference*, 1999, p. 117-124.

[80] L. S. Larkey, P. Ogilvie, M. A. Price, et B. Tamilio, « Acrophile: an automated acronym extractor and server », in *Proceedings of the fifth ACM conference on Digital libraries*, 2000, p. 205-214.

[81] Y. Park et R. J. Byrd, « Hybrid text mining for finding abbreviations and their definitions », 2001.

[82] J. Liu, C. Liu, et Y. Huang, « Multi-granularity sequence labeling model for acronym expansion identification », *Information Sciences*, vol. 378, p. 462-474, févr. 2017, doi: 10.1016/j.ins.2016.06.045.

[83] D. Sánchez et D. Isern, « Automatic extraction of acronym definitions from the Web », *Appl Intell*, vol. 34, n° 2, p. 311-327, avr. 2011, doi: 10.1007/s10489-009-0197-4.

[84] D. Nadeau et P. D. Turney, « A Supervised Learning Approach to Acronym Identification », in *Advances in Artificial Intelligence*, Berlin, Heidelberg, 2005, p. 319-329, doi: 10.1007/11424918_34.

[85] E. Adar, « SaRAD: A simple and robust abbreviation dictionary », *Bioinformatics*, vol. 20, n° 4, p. 527-533, 2004.

[86] J. T. Chang, H. Schütze, et R. B. Altman, « Creating an online dictionary of abbreviations from MEDLINE », *Journal of the American Medical Informatics Association*, vol. 9, n° 6, p. 612-620, 2002.

[87] J. Xu et Y. Huang, « Using SVM to extract acronyms from text », *Soft Computing*, vol. 11, n° 4, p. 369-373, 2007.

[88] Z. Wu et M. Palmer, « Verbs semantics and lexical selection », in *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, 1994, p. 133-138.

[89] P. Resnik, « Using information content to evaluate semantic similarity in a taxonomy », *arXiv preprint cmp-lg/9511007*, 1995.

[90] J. J. Jiang et D. W. Conrath, « Semantic similarity based on corpus statistics and lexical taxonomy », *arXiv preprint cmp-lg/9709008*, 1997.

[91] D. Lin, « Extracting collocations from text corpora », in *First workshop on computational terminology*, 1998, p. 57-63.

[92] C. Leacock et M. Chodorow, « Combining local context and WordNet sense similarity for word sense identification. WordNet, An Electronic Lexical Database », *The MIT Press*, 1998.

[93] S. Banerjee et T. Pedersen, « An adapted Lesk algorithm for word sense disambiguation using WordNet », in *International conference on intelligent text processing and computational linguistics*, 2002, p. 136-145.

[94] G. Hirst et D. St-Onge, « Lexical chains as representations of context for the detection and correction of malapropisms », *WordNet: An electronic lexical database*, vol. 305, p. 305-332, 1998.