Extraction de contenus sémantiques pour la vérification d'exigences systèmes

Aurélien Lamercerie, David Rouquet, Valérie Bellynck, Christian Boitet, Vincent Berment

Extraction de contenus sémantiques pour la vérification d'exigences systèmes

Aurélien Lamercerie*, David Rouquet**,
Valérie Bellynck*, Christian Boitet***, Vincent Berment****,***

* G-INP/LIG/GETALP, ** Tétras-Libre, *** UGA/LIG/GETALP, **** CS Group

Résumé. Cet article présente l'application d'une méthode d'extraction de contenu sémantique dans un contexte industriel, avec pour objectif la vérification automatique d'exigences systèmes rédigées non pas dans un langage contrôlé, mais en langue naturelle non contrainte. L'étape d'extraction utilise une analyse par transduction sémantique, implémentée en s'appuyant sur les standards du Web Sémantique du W3C. Elle part d'une représentation sémantique des textes, exprimée sous forme de graphes UNL (Universal Networking Language) à "sens garanti" (obtenus grâce à une étape de désambiguïsation interactive), qui produit d'abord une structure semi-formelle et indépendante de la langue source. Les outils développés construisent ensuite automatiquement une ontologie OWL à partir des spécifications du système, exprimées par des énoncés non contraints. Finalement, une vérification automatique des exigences est réalisée à l'aide de règles SPARQL génériques et de raisonneurs logiques. La fin de l'article décrit une mise en pratique sur des exigences extraites d'une spécification réelle.

1 Introduction

Le projet RAPID UNSEL [1], financé par la DGA [2] de 2019 à 2021, vise à fournir des outils pour accompagner la spécification de "systèmes de systèmes" (par exemple, un système de communications sol-air pour un aéroport ou un système de freinage d'urgence). Un tel système est construit à partir d'un cahier des charges, puis de documents de plus en plus détaillés : document de spécifications externes ou internes, document d'analyse générale ou détaillée, documents de programmation. Une spécification est un ensemble structuré d'exigences, qui sont majoritairement des énoncés descriptifs ou prescriptifs en langue naturelle (LN).

Les problèmes liés aux spécifications sont bien connus [3]. Ce sont *l'incohérence* [4], *l'incomplétude* et *l'inadéquation*. Ainsi, il arrive qu'il y ait des exigences contradictoires, que l'ensemble des exigences ne contienne pas tout ce qui est nécessaire ou que des exigences ne soient pas comprises par leurs lecteurs. Le coût d'une erreur s'avère parfois considérable. Ces problèmes, dont l'enjeu est important, sont pourtant peu ou mal traités au niveau opérationnel.

1. UNseL : Universal Networking system engineering Language
2. Direction Générale de l'Armement, `https://www.defense.gouv.fr/dga`
3. voir par exemple Dick et al. (2017).
4. ou plus précisément *l'inconsistance*, si on est dans un système logique comme une ontologie.

Par ailleurs, dans l'idée d'une application à des systèmes critiques, l'usage de représentations *à sens garanti* s'est ajouté aux contraintes du projet. La revue de l'état de l'art, par exemple Kamath et Das (2019), n'a pas permis de trouver des approches permettant de répondre à ce besoin. Nos travaux y apportent une réponse originale.

La vérification d'anomalies semble devoir s'appuyer sur la construction d'un système formel de type "ontologie métier" ou "ontologie de domaine". Pour cela, il faut représenter le sens de chaque exigence, ainsi que les liens structurels et séquentiels entre exigences, de façon aussi précise, exacte et complète que possible. Dans le projet UNSEL, nous implémentons les ontologies de domaine dans le langage logique OWL [5].

Il est difficile, voire impossible (Kay (2017) [6]), d'obtenir automatiquement une représentation linguistique non ambiguë, pour chaque exigence, telle que l'on puisse en dériver le sens dans une ontologie. Aucun analyseur disponible ne le fait, pour aucune langue. Une raison majeure de cette impossibilité est que les énoncés en LN sont presque toujours ambigus, et souvent imprécis, voire flous, même si on utilise un langage restreint. Pour garantir l'adéquation entre une exigence et sa représentation, le projet UNSEL implémente un système de désambiguïsation interactive intuitive permettant de voir qu'un passage est ambigu [7], et de choisir l'interprétation désirée.

Une fois que les exigences sont représentées au niveau linguistique de façon correctement désambiguïsée, dans notre cas grâce à des (hyper-)graphes UNL, il est nécessaire de les traduire dans l'ontologie considérée. C'est en effet seulement à ce niveau que les calculs de consistance et de complétude peuvent se faire. Pour répondre à cet enjeu, une technique d'extraction de sens, basée sur le concept de *transduction sémantique compositionnelle* (Lamercerie (2021)), a été adaptée aux besoins du projet.

La suite de cet article détaille la mise en œuvre de ce procédé. Partant de graphes UNL (section 2), le processus d'extraction de contenu (section 3) est appliqué à la vérification automatique d'un corpus d'exigences systèmes (section 4). Les outils présentés sont disponibles sous licence CeCILL-B dans un dépôt Gitlab dédié [8].

2 Représentation du sens des textes avec UNL

La mise en œuvre d'une approche d'extraction par transduction sémantique requiert une représentation sémantique des énoncés linguistiques, sous forme de graphes (ou mieux de réseaux) sémantiques. Dans le cadre du projet UNSEL, la représentation d'un énoncé est un (hyper-)graphe du langage UNL [9], qui est un langage formel permettant d'exprimer des structures abstraites de l'anglais, et dont le vocabulaire est formé de "lexèmes interlingues" (les UW, ou *Universal Words*).

5. W3C Working Group (2012)
6. *You can't get blood out of a stone*, Martin Kay, 2009, 40-ième anniversaire de l'ATALA.
7. projet LIDIA, Blanchon (1994), projet Eureka Eurolang, Boitet et al. (1995)
8. `https://gitlab.tetras-libre.fr/unl/tenet`
9. UNL Specification 3.3 (2004)

2.1 Le langage UNL

Le sigle UNL désigne un projet, un langage de graphes sémantiques d'énoncés en langue naturelle et un format de documents multilingues (intégré à Html via des balises de forme [xyz] et {attr=val}). Le projet international UNL a été lancé en décembre 1996 [10], avec initialement les 12 langues les plus parlées au monde, à l'initiative de l'Institute of Advanced Studies (IAS) de l'*Université des Nations Unies* [11] (UNU). Dans le *langage UNL*, le sens d'une phrase est représenté par un (hyper)-graphe, comme illustré dans la figure 1. L'idée de base est de rendre les graphes sémantiques compréhensibles et constructibles par les chercheurs et développeurs du monde entier. C'est pourquoi tous les symboles utilisés dans un graphe UNL proviennent de l'anglais. On peut néanmoins, par le jeu des restrictions sémantiques, dénoter des acceptions n'existant pas en anglais (e.g. 'tatami', 'alunir', certains niveaux de politesse, etc.).

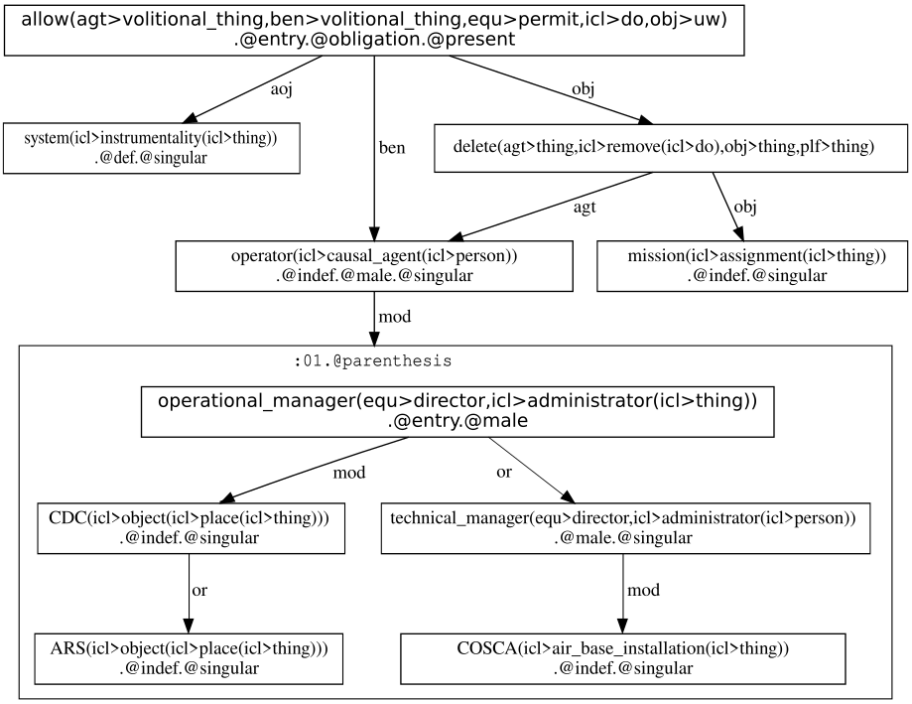

FIG. 1 – *Un graphe UNL possible pour l'exigence "Le système doit permettre à un opérateur (gestionnaire opérationnel d'un CDC ou d'un ARS ou gestionnaire technique d'un COSCA) de supprimer une mission".*

On convient qu'un graphe UNL représentant un énoncé E dans une langue quelconque L doit être la structure abstraite d'un énoncé anglais équivalent à l'énoncé d'origine E.

10. Uchida et al. (1996)
11. `https://unu.edu/`, Tokyo

C'est important parce que les relations sémantiques portées par les arguments d'un lexème de L (verbe ou déverbal comme 'permettre', 'permission', 'dépendre', ou adjectif comme 'utile', voire nom non déverbal comme 'méthode') ne sont pas toujours les mêmes que celles portées par les arguments correspondants d'un lexème anglais équivalent, bien que le répertoire des relations sémantiques soit universel. C'est pourquoi on peut dire que le langage UNL est un *langage pivot anglo-sémantique*.

Un (hyper-)graphe UNL est formé de nœuds, d'arcs orientés et d'attributs booléens, ces différents éléments portant des informations de natures différentes.

Un *nœud* peut être élémentaire ou composé.

— Un *nœud élémentaire* contient un UW (*Universal Word*, ou *lexème interlingue*) et des attributs booléens sémantiques ou pragmatiques (comme `.@past`). Un UW dénote une acception. Il est formé d'un mot-vedette (*headword*) [12], et d'une liste de restrictions. L'UW suivant peut dénoter 'amerrir' : `land(icl>do, aoj>flying thing, gol>water)`.

— Un *nœud composé* est un sous-graphe, appelé *scope*, représentant une partie de la phrase. Le scope i est constitué de tous les arcs de numéro de scope i et des nœuds qu'ils relient. Un arc ne peut appartenir qu'à un scope, dont il porte le numéro ($:00, :01, :02\ldots$ [13]), mais un nœud peut être commun à plusieurs scopes. Tout scope doit contenir un et un seul nœud portant l'attribut `.@entry`. Enfin, un scope doit être connexe par arcs, si on néglige l'orientation des arcs.

Sur l'exemple de la figure 1, l'UW `operator(icl>causal_agent(icl>person))` dénote un *opérateur*, tandis que le verbe *supprimer* est représenté par l'UW `delete(agt>thing, icl>remove(icl>do), obj>thing, plf>thing)`.

Les *arcs* définissent des relations entre les nœuds. Ils sont orientés et étiquetés par des relations sémantiques binaires prises dans un répertoire fixé (une quarantaine).

— Les relations sont notées par des abréviations (ex : agent (`agt`), objet (`obj`), bénéficiaire (`ben`), localisation (`plc`), durée (`dur`), destination (`plt`), possesseur (`pos`), etc.).

— Une extension due à UNL-ru et UNL-fr, très utile mais pas encore introduite dans le standard UNL, consiste à ajouter un attribut de *position argumentaire* (`.@A`, `.@B`, `.@C`, `.@D`) à certains arcs, ce qui permet de distinguer deux prédicats d'arités différentes [14].

Les *attributs* (booléens) sont préfixés par `.@`, et attachés aux nœuds (élémentaires ou composés). Ils apportent des précisions sur les nœuds du graphe, telles que :

— le rôle dans le graphe (ex : `.@entry` indique le nœud principal d'un scope, d'où on doit partir pour interpréter le graphe) ;

— des informations sémantiques et pragmatiques, par exemple le temps abstrait (*time* par opposition à *tense*), l'aspect, la modalité, la négation ;

— d'autres éléments de communication comme les actes de parole, le niveau de politesse, le nombre (singulier, pluriel, collectif), la détermination (défini, indéfini), etc.

Par exemple, l'utilisation de l'imparfait en français, ou du progressif passé en anglais, peut être représentée par la combinaison d'attributs `.@past.@repetition`.

12. un lemme anglais ou un lemme emprunté par l'anglais à une autre langue, comme *tatami*.
13. le numéro du scope principal, : 00, est en général omis.
14. Par exemple, `delete(agt.@A>thing, icl>remove(icl>do), obj.@B>thing, plf>thing)` pour 'A supprimer/tuer B', et `delete(agt.@A>thing, icl>remove(icl>do), obj.@B>thing, plf.@C>thing)` pour 'A supprimer/ôter B de C'.

2.2 Enconversion et sérialisation des graphes UNL

La transformation d'un texte en graphes UNL (un par phrase) est appelée *enconversion*, et non analyse, pour souligner qu'on passe d'un espace lexical (français, anglais...) à un autre (celui des UW UNL). L'opération inverse est appelée *déconversion*.

La première étape de l'enconversion appelle un analyseur structural (syntaxique et sémantique) développé avec l'environnement Ariane-H [15] et permettant de garantir le sens analysé. En présence d'ambiguïtés, il produit toutes les analyses possibles et pose des questions à l'utilisateur pour obtenir le sens correct. La représentation obtenue est, par construction, une image complète et fidèle du texte analysé. Cette représentation est ensuite convertie en graphes UNL.

Un des problèmes des approches dites "à l'état de l'art" est qu'elles utilisent un parseur vers une représentation sémantique interlingue, comme UNL ou AMR, en supposant faussement qu'il fournit automatiquement un résultat correspondant à ce qu'a voulu dire le rédacteur. Or il y a beaucoup d'ambiguïtés lexicales, d'ambiguïtés d'attachement et d'ambiguïtés de dépendance sémantiques : même avec une phase de désambiguïsation automatique, les résultats sont en pratique très souvent incorrects, confirmant l'affirmation de Martin Kay (2017). L'étape de désambiguïsation interactive apporte une réponse à ces problèmes, réponse renforcée par le choix d'UNL. Ce formalisme est moins connu qu'AMR, mais nettement plus expressif et précis, notamment pour la partie lexico-sémantique.

Afin d'obtenir une chaîne d'extraction entièrement fondée sur les standards du Web sémantique du W3C, nous utilisons une sérialisation RDF des graphes UNL (Rouquet et al. (2020)). Un convertisseur du format standard UNL vers la sérialisation dite *UNL-RDF* a été développé dans le cadre du projet. Il est disponible sous forme d'exécutable Java [16] et de service Web [17].

3 Extraction de contenu par transduction sémantique

La contribution principale de notre démarche est l'adaptation d'une technique d'analyse par transduction sémantique à des graphes UNL, transformés pour aboutir à une nouvelle structure formelle. Cette étape permet la construction automatique d'une ontologie OWL à partir d'une ontologie cadre passée en paramètre. L'implémentation est fondée sur les standards du Web sémantique (RDF, OWL, SPARQL, SHACL) [18].

3.1 Analyse par transduction sémantique

Le principe d'analyse par transduction sémantique (Lamercerie (2021)) a été validé par une première expérimentation sur des graphes AMR (Abstract Meaning Representation, Banarescu et al. (2013)). Nous l'avons adapté aux hypergraphes UNL [19], en reprenant les notions de filet sémantique et de schéma de transduction compositionnel.

15. `https://linguarium.org`
16. `https://gitlab.tetras-libre.fr/unl/unlTools`
17. `https://unl.demo.tetras-libre.fr/`
18. W3C Standards (2021)
19. Ces graphes ne sont pas des graphes classiques en théorie des graphes, où on ne peut avoir plus d'un arc allant d'un sommet s_i à un sommet s_j (dans un graphe orienté), ou plus d'un arc reliant un sommet s_i à un sommet s_j (dans un graphe non orienté). Il s'agit plutôt de *réseaux*, et d'un abus de langage. On parle aussi de *graphes de transitions* au lieu de réseaux de transitions d'automates.

Filets sémantiques. Intuitivement, un *filet sémantique* est un objet construit sur un graphe sémantique G de façon inductive, à partir d'une base formée de filets "atomiques" (de type 'atome') correspondant aux nœuds du graphe, en utilisant des règles de construction. Nous notons F_G (ou F) l'ensemble des filets ainsi définis. Un filet est un triplet *f = (ensemble de nœuds* ou *"ancrage", type, ensemble de valeurs)*. Dans un filet atomique, les valeurs possibles sont (1) celles des parties de l'information portée par le nœud (UW complet, mot vedette, valeurs des restrictions, traits sémantiques, identificateurs des nœuds et des arcs), et (2) les résultats de l'application de fonctions (ex : score calculé par une distance Lesk liée à un thésaurus).

Une règle de "transduction sémantique" d'arité k prend en argument k filets $f_1 \dots f_k$ et produit un filet f dont l'ensemble de nœuds est l'union des nœuds des f_i, dont le type est déterminé par la règle (par exemple, *list<atom>*), et dont les valeurs sont calculées à partir des valeurs des filets arguments (par exemple, le maximum ou la moyenne des scores, l'union des traits sémantiques de certains nœuds ou filets). La définition 1 suivante précise ce concept.

Définition 1 *Soit $\mathcal{G}$ un graphe étiqueté et S l'ensemble des sommets de G. Soit $\mathcal{T}$ un ensemble de types, et $\mathcal{V}$ un ensemble de valeurs. Un filet sémantique sur $\mathcal{G}$ est une structure $f = (x, \tau, v)$ telle que $x \subseteq S$ est une partie de S, $\tau \in \mathcal{T}$ est un type et $v \subseteq \mathcal{V}$ est un ensemble de valeurs.*

Le type d'un filet définit la nature des valeurs qui lui sont associées, et les opérations qui pourront lui être appliquées. L'ensemble des types $\mathcal{T}$ est supposé muni d'une relation d'ordre, permettant d'établir plusieurs niveaux d'analyse. Notre implémentation inclut le type *atome*, caractérisant des filets *atomiques* ne couvrant qu'un seul nœud, le type *composite*, définissant des filets *composite* associant un concept à plusieurs nœuds, ou encore le type *list*<τ> qui caractérise une liste d'éléments de type τ.

La notion de filet sémantique est illustrée par la figure 2. Le filet F_a, de type *atome*, s'ancre au nœud *1* et porte plusieurs valeurs, dont la classe de l'atome (ici *operator*) et sa classe parente (*agent*). Le filet F_b, de type *list<atom>*, s'ancre aux nœuds *2* et *4* et porte plusieurs valeurs, dont les éléments de la liste et la relation associée (*mod*). Le filet F_{ab}, de type *list<composite>*, s'ancre aux nœuds *1*, *2* et *4*, et porte plusieurs valeurs, dont les éléments composites de la liste (classes *operational manager* et *technical manager*) et la classe mère (*operator*). Nous verrons plus loin que le filet F_{ab} est obtenu par composition des filets F_a et F_b.

Schémas de transduction compositionnels (STC). La définition 2 spécifie une structure associant une formule logique et un opérateur de transduction. Elle s'applique aux filets d'un graphe sémantique, et permet l'obtention de nouveaux filets par composition. Dans cette optique, nous considérons les propriétés des filets, dérivées des types et des valeurs. L'ensemble des propriétés considérées est noté $\mathcal{P}$.

Définition 2 *Soit P un ensemble de prédicats. Soit $\mathcal{R}_\mathcal{P}$ un ensemble de conjonctions d'expressions $p(f_1, \dots, f_n)$, avec $p \in P$. Un schéma de transduction compositionnel (STC) σ est une paire $\sigma = (\varphi, tr)$ telle que :*

- *φ est une formule logique sur $\mathcal{R}_\mathcal{P}$;*
- *tr (d'arité k) est une fonction de F^k dans F définie par deux fonctions ψ_τ et ψ_v et telle que, si $f_i = (x_i, \tau_i, v_i)$ avec $i \in [1..k]$, et $f = (x, \tau, v) = \sigma(f_1, \dots, f_k)$, alors $x = \bigcup x_i$, $\tau = \psi_\tau(\tau_1, .., \tau_k)$, et $v = \psi_v(v_1, .., v_k)$, où ψ_τ et ψ_v sont deux fonctions retournant respectivement un type et un ensemble de valeurs.*

La figure 2 donne un exemple d'application d'un schéma $\sigma = (\varphi, tr)$, composant un filet *atomique* et un filet de type *liste*. La partie requête du schéma est définie par $\varphi = atom(f_1) \wedge$ *list<atom>*$(f_2) \wedge mod(f_1, f_2)$. Elle permet de sélectionner les filets à composer (le filet F_a de type *atom* et le filet F_b de type *list<atom>*, ces deux filets étant liés par la relation `mod`.

Le filet résultant F_{ab} est obtenu par application de la fonction tr sur les filets sélectionnés : $F_{ab} = (F_a.x \cup F_b.x, \psi_\tau(F_a.\tau, F_b.\tau), \psi_v(F_a.v, F_b.v))$. Les fonctions ψ_τ et ψ_v permettent de définir le type (ici, *list<atom>*) et l'ensemble de valeurs du nouveau filet (objets en relation définissant une hiérarchie de concepts).

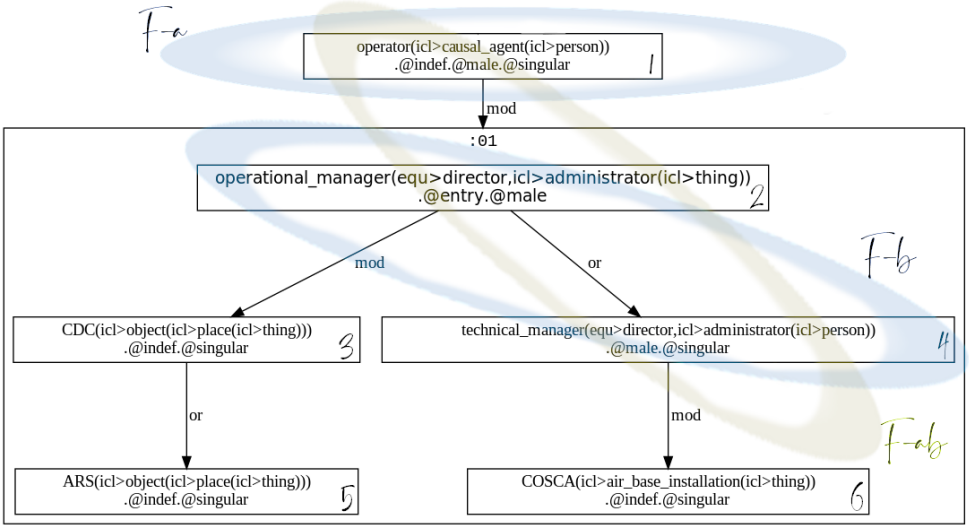

FIG. 2 – *Graphe UNL-RDF portant sur la désignation d'un opérateur, avec quelques filets.*

Application des STC. L'objectif du processus d'analyse est d'enrichir le graphe UNL-RDF en prenant en compte son contenu et sa structure. La mise en œuvre suit un mode d'exécution à quatre niveaux : (1) **extraction des éléments atomiques**, (2) **formation d'éléments composites** par un procédé récursif, (3) **extraction des propriétés et relations** pour les éléments atomiques et composites, (4) **construction de l'ontologie cible**. Le typage des filets permet d'ajuster le traitement pour en assurer l'efficacité et la terminaison. Un exemple d'implémentation est donné dans la section 3.2, avec quelques précisions complémentaires.

Les STC de niveau 1 s'appliquent sur les sommets du graphe UNL-RDF. Ils initialisent le traitement en générant des filets *atomiques*. L'extraction des éléments atomiques est paramétrée dans une *ontologie cadre* précisant le résultat attendu (voir section 3.2 ci-dessous).

Les STC de niveau 2 suivent un procédé récursif, contrôlé en s'appuyant sur le typage des filets. Cette étape permet de construire des filets *composites*, et d'obtenir ainsi une hiérarchie de concepts dans l'ontologie cible. La figure 2 est une illustration de ce procédé : le filet obtenu désigne une liste d'éléments composites (liste de classes) avec une hiérarchie entre la classe parente *operator* et les classes filles *operational manager*, *technical manager*.

Les STC de niveau 3 opèrent de manière similaire pour détecter des propriétés sur les éléments, atomiques ou composites, et des relations entre ces éléments. Tandis que les STC des niveaux précédents produisent des filets rattachés à des classes d'éléments (par exemple, des agents, des messages, des actions), les STC de niveau 3 traduisent les relations entre ces filets en propriétés sur les classes. Les filets produits à ce stade sont de type *property*. Ils sont centrés sur les verbes, traduisant l'attribution d'une propriété ou la qualification d'une action, et rattachés à plusieurs classes d'éléments. Sur l'exemple présenté, la classe *operator* est reliée à l'action *delete*, avec pour objet la classe *message*. Les filets produits par les STC de niveau 3 semblent bien correspondre aux *factoïdes* [20] utilisés dans le système Watson.

Finalement, les STC de niveau 4 génèrent l'ontologie attendue à partir des filets générés. Le typage des filets oriente directement la construction de nouvelles classes, instances et propriétés pour mettre à jour la structure cible.

3.2 Implémentation

Cette section présente une implémentation des STC, entièrement basée sur les standards du Web sémantique. Elle est disponible en Open Source sous l'acronyme TENET (*Tool for Extraction using Net Extension by (semantic) Transduction*) [21].

Entrées et paramètres. Le processus d'extraction prend en entrée un ensemble de graphes UNL, dans leur sérialisation RDF. Il prend par ailleurs en paramètre une ontologie cadre, qui définit les objets visés selon le contexte métier, ainsi que des *graines d'extraction* permettant d'initialiser les filets de niveau 1. L'ontologie cadre dépend du point de vue attendu : elle contient les grandes classes d'éléments à extraire (une dizaine de classes dans notre cas). Pour chacune de ces classes, il est nécessaire de déclarer au moins une propriété de type *has-seed* (graines d'extraction).

Processus d'extraction à base de STC. Les STC sont implémentés sous la forme de requêtes SPARQL-construct qui s'appliquent aux graphes UNL-RDF. Les requêtes sont intégrées dans un graphe de contraintes (*shapes*) respectant la spécification SHACL-SPARQL [22]. Elles sont ordonnées en niveaux d'application (voir section 3.1), grâce au mécanisme *sh :order* [23].

Les règles SPARQL dépendent fortement de la structure des graphes sémantiques en entrée, elle-même dépendant principalement du formalisme UNL et de la phraséologie du corpus. Par contre, les règles sont génériques du point de vue du contenu métier des phrases. En effet, ce dernier est paramétré par l'ontologie cadre. Ainsi, des règles développées sur la base de notre corpus, décrivant un système de communication sol-air, restent *a priori* valables pour des spécifications métier très différentes, comme celles d'un système de freinage d'urgence.

Sortie. La sortie est un ensemble de triplets RDF-OWL enrichissant et instanciant une ontologie cadre passée en paramètre. Il est tout à fait possible que le processus d'extraction vise plusieurs ontologies décrivant des facettes différentes du système.

20. Hovy et al. (2002)
21. `https://gitlab.tetras-libre.fr/unl/tenet`
22. `https://www.w3.org/TR/shacl/#dfn-shacl-sparql`
23. `https://www.w3.org/TR/shacl/#order`

4 Application à la vérification d'un corpus d'exigences

Nous avons appliqué nos méthodes sur un corpus réel d'exigences système fourni par la DGA (*SRSA-IP*). Ce corpus est composé de 367 exigences décrivant un système de communication sol-air. Il n'est pour l'instant pas accessible publiquement. Les premières applications ont été réalisées sur un corpus pilote de 40 exigences, sélectionnées dans SRSA-IP pour leur complexité linguistique et leur représentativité du corpus. Nous présentons d'abord une extraction et des vérifications réalisées sur l'exigence de la figure 1, puis les résultats obtenus sur le corpus pilote.

4.1 Ontologie cadre et règles d'extraction

La figure 3 présente un aperçu de l'ontologie cadre utilisée dans les premières expérimentations. Elle contient essentiellement une dizaine de classes accompagnées de graines d'extraction. Les requêtes SPARQL qui composent les STC de niveau 1 utilisent les graines de l'ontologie cadre, pour identifier les éléments atomiques dans les exigences. Actuellement, les graines sont basées sur les restrictions des UW. Par exemple, dans la figure 3, on voit que la classe *Agent* sera initialisée avec toutes les UW portant les restrictions `icl>administrator`, `icl>person` ou `icl>human`. Nous envisageons d'autres méthodes, permettant de définir ces graines à partir d'exemples ou de les généraliser à des sous-graphes.

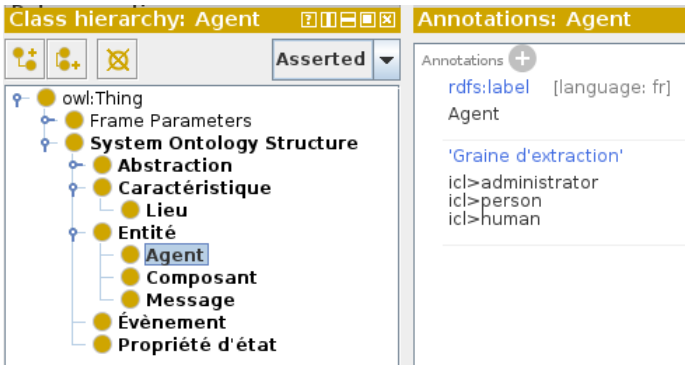

FIG. 3 – *Exemple d'ontologie cadre*

Nous avons défini 25 règles SPARQL pour l'extraction. Partant des éléments atomiques identifiés par les graines, les règles parcourent les (listes de) modificateurs (`mod` en UNL), précisant les éléments atomiques, pour aboutir à une hiérarchie d'éléments composites dans l'ontologie. D'autres règles extraient la liste ouverte des relations que les éléments composites peuvent entretenir entre eux, par exemple les actions des *Agents* sur les *Composants*. Enfin, les classes et relations de l'ontologie sont instanciées. Ce mécanisme permet en particulier d'assurer la traçabilité de ce qui est extrait, par exemple que l'exigence *STB_PHON_300* mentionne qu'un certain opérateur peut supprimer une mission particulière.

4.2 Exemple d'extraction et de vérification sur une exigence du corpus

L'application des règles d'extraction SPARQL sur le graphe UNL-RDF de la figure 1 produit 289 triplets RDF intermédiaires dans le processus de transduction, et 119 triplets ajoutés à l'ontologie cadre du système. Les informations extraites sont illustrées par la figure 4 avec :

— une hiérarchie des agents mentionnés, en haut à gauche (en jaune),

— une classe définie comme union de deux autres, en haut à droite (en jaune),

— une *propriété d'événement* dont le domaine et le but sont précisés, à droite (en bleu),

— l'assertion précisant qu'une instance d'opérateur "*delete*" une instance de mission, en bas (en violet).

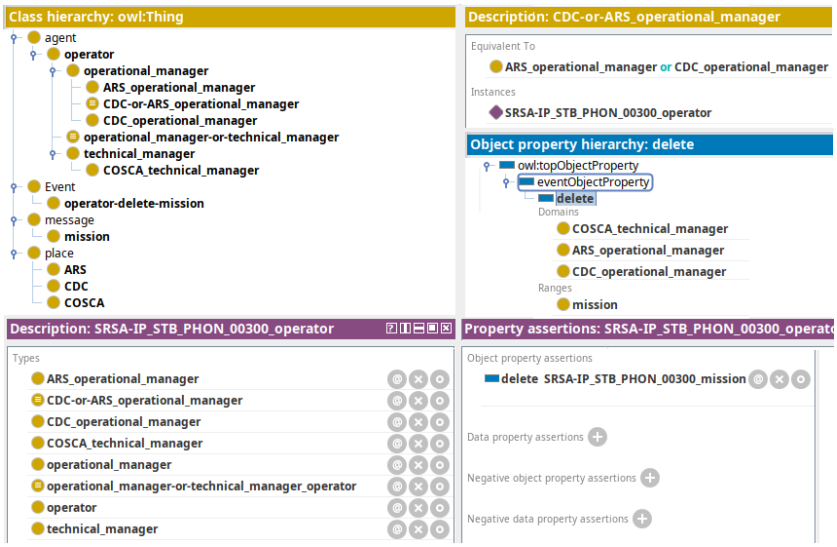

FIG. 4 – *Aperçu du résultat d'extraction sur l'exigence de la figure 1*

Le contenu RDF-OWL extrait permet de réaliser des vérifications, également implémentées sous la forme de règles SPARQL. Des messages d'alerte et des suggestions sont retournées à l'utilisateur après le contrôle des points suivants :

— **Sous-spécifications** dans les exigences, par exemple pour la mention *gestionnaire opérationnel* dans une exigence, parle-t-on bien de toutes les sous-classes comme *gestionnaires opérationnel* d'un *CDC*, d'un *ARS*, etc. ?

— **Défaut terminologique**, par exemple si la seule sous-classe de *mission*, extraite dans toutes les exigences, est *mission COSCA*, c'est soit que l'on a deux termes pour identifier le même concept, soit que l'on a sous-qualifié *mission* dans certaines exigences.

— **Qualification homogène des éléments en relation**, par exemple si une exigence mentionne qu'un *gestionnaire COSCA peut supprimer une mission COSCA* et si une autre indique qu'un *opérateur d'un CDC peut créer une mission*, on pourra proposer de compléter *mission* par *CDC* dans la seconde exigence.

D'autres vérifications de consistance entre les exigences peuvent être réalisées à l'aide de raisonneurs logiques génériques. Le rapport Rouquet et al. (2020) détaille un exemple de ce type, dont voici une version simplifiée : (1) "*Une voie radio peut prendre deux états : écoute et veille.*" et (2) "*L'opérateur place la voie radio dans l'état trafic.*". À partir de ces deux exigences, une ontologie incohérente est produite, car *trafic* ne fait pas partie de l'ensemble {*écoute*, *veille*} des états possibles pour une *voie radio*.

4.3 Résultats obtenus sur le corpus pilote

Partant des 40 exigences du corpus pilote, notre processus produit 4990 triplets RDF intermédiaires dans le processus de transduction, et 1930 triplets ajoutés à l'ontologie cadre du système. Les informations suivantes sont notamment ajoutées à l'ontologie :

— 33 classes organisées en hiérarchie et reliées par des relations ensemblistes dont 12 classes d'agents et 21 composants physiques ou "abstraits",

— 14 propriétés, liant les classes précédentes, dont 5 correspondent à des actions, des événements ou des états du système (*create*, *delete*, *release*, *set_up*, *include*).

Les mesures classiques de précision et de rappel restent à produire sur l'ensemble du corpus, mais les résultats sur le corpus pilote sont plus qu'encourageants. Les informations extraites ont permis de détecter 100% des anomalies pointées manuellement sur le corpus. On note toutefois des aspects pas ou mal pris en compte dans l'extraction et qui laissent une bonne marge de progression, par exemple les modalités déontiques et temporelles.

Malgré ces limites, l'examen du graphe sémantique OWL produit est directement instructif pour un humain et fournit des informations denses et pertinentes pour la compréhension du système spécifié. Par exemple, la notion de *mission* est correctement explicitée comme une entité contenant des *voies radio*. Les différents types de *mission* sont extraits, ainsi que les types d'*agents* qui peuvent les créer, les supprimer, ou modifier leurs *voies radio*.

5 Conclusion

Nous avons présenté dans cet article l'application concrète d'une chaîne de traitement globale, partant d'exigences système exprimées en langue naturelle (LN) pour aboutir à une ontologie OWL du système décrit par ces exigences. Cette chaîne a plusieurs étapes : (1) l'enconversion des énoncés en LN dans le format standard UNL, avec une étape de désambiguïsation interactive intuitive en langue source (de rédaction), (2) la sérialisation RDF des graphes UNL, (3) l'extraction du contenu sémantique pour construire une ontologie OWL du système et (4) la vérification automatique de l'ontologie produite.

L'expérimentation réalisée a permis de montrer qu'il est possible de construire des axiomes OWL qui supportent un raisonnement non trivial. Il devient ainsi envisageable de gérer la proximité sémantique des termes, de vérifier la cohérence structurelle des entités décrites dans les exigences, et de mettre en évidence des incohérences sur les propriétés définies. L'usage des graphes UNL permet de réduire la dépendance monolingue des logiciels envisagés, tandis que le processus de désambiguïsation interactive apporte une garantie de sens sur les représentations pivot (UNL) exploitées lors de l'extraction. L'analyse par transduction sémantique est un procédé simple, traçable et adaptable pour l'interprétation des énoncés et la construction des ontologies visées.

Références

Banarescu, L., C. Bonial, S. Cai, M. Georgescu, K. Griffitt, U. Hermjakob, K. Knight, P. Koehn, M. Palmer, et N. Schneider (2013). Abstract Meaning Representation for Sembanking. In Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse, pp. 178–186. Sofia, Bulgaria : Association for Computational Linguistics.

Blanchon, H. (1994). LIDIA-1 : une première maquette vers la TA Interactive "pour tous". Thèse, Université Joseph-Fourier - Grenoble I.

Boitet, C., E. Planas, H. Blanchon, É. Blanc, J.-P. Guilbaud, P. Guillaume, M. Lafourcade, et G. Sérasset (1995). LIDIA-1.2, une maquette de TAO personnelle multicible, utilisant la messagerie usuelle, la désambiguïsation interactive et la rétrotraduction.

Dick, J., E. Hull, et K. Jackson (2017). Requirements Engineering. Springer International Publishing.

Hovy, E., U. Hermjakob, C.-Y. Lin, et D. Ravichandran (2002). Using Knowledge to Facilitate Factoid Answer Pinpointing. In Proceedings of COLING 2002.

Kamath, A. et R. Das (2019). A Survey on Semantic Parsing. In Automated Knowledge Base Construction (AKBC).

Kay, M. (2017). Translation : Linguistic and Philosophical Perspectives, Volume 221 of CSLI Lecture Notes. CSLI Publications.

Lamercerie, A. (2021). Principe de transduction sémantique pour l'application de théories d'interfaces sur des documents de spécification. Thèse, Université de Rennes 1.

Rouquet, D., V. Bellynck, V. Berment, et C. Boitet (2020). Natural Language Representation and Content Extraction using RDF, SHACL and UNL.

Uchida, H., M. Zhu, et T. Della Senta (1996). UNL : An Electronic Language for Communication, Understanding and Collaboration. UNU/IAS/UNL Center.

UNL Specification 3.3 (2004). (http ://www.unlweb.net/wiki/images/a/ab/Spec33.pdf).

W3C Standards (2021). Semantic web (https ://www.w3.org/standards/semanticweb/).

W3C Working Group (2012). OWL-2 Overview (http ://www.w3.org/TR/owl2-overview/).

Summary

This paper presents the application of a semantic content extraction method in an industrial context, with the objective of automatic verification of system requirements written not in a controlled language, but in an unconstrained natural language. The extraction step uses a semantic transduction analysis, implemented using the W3C Semantic Web standards. It starts from a linguistic representation of the texts, in the form of UNL (Universal Networking Language) graphs with "guaranteed meaning" (obtained thanks to an intermediate step of interactive disambiguation), which first produces a semi-formal structure independent of the source language. The system then builds an OWL ontology from the system specifications, expressed by unconstrained NL statements. Finally, an automatic verification of the requirements is performed using generic SPARQL rules and logical reasoners. The end of the article describes a practical implementation on requirements extracted from a real specification.