Développer des applications sémantiques en expliquant les conséquences de conception de la base de connaissances

Gaëlle Lortal

Développer des applications sémantiques en expliquant les conséquences de conception de la base de connaissances

Lortal Gaëlle[1]

[1] Thales Research and Technology, LRASC

gaelle.lortal@thalesgroup.com

Résumé

Les applications à base de sémantique intéressent de plus en plus les industriels, principalement parce qu'elles permettent le partage d'information et l'explicabilité des résultats. Afin de développer l'utilisation des applications sémantiques, nous proposons un outil de création de bases de connaissances (ontologies) pour l'industrie, c'est-à-dire pour non experts.
L'outil GLUON prend en compte (1) l'interaction nécessaire avec l'utilisateur (interaction linguistique grâce à un verbaliseur), (2) différentes logiques nécessaires à l'utilisateur (dans la mesure du possible) ainsi que (3) l'existant (création de base de connaissances de zéro, à base de texte, à base de réutilisation ou d'alignement/fusion). (4) La vérification des bases de connaissances créées se fait avant tout par conception, au fil de l'eau.

Mots-clés

Ontologie, Reasoners, Verbalizers

Abstract

Semantic-based applications are of increasing interest to industrialists, mainly because they allow the sharing of information and the explainability of results. To develop the use of semantic applications, we propose a tool to create knowledge bases (ontologies) for industry i.e. non-experts.

The tool GLUON takes into account (1) the necessary interaction with the user (linguistic interaction thanks to a verbalizer), (2) different logics necessary for the user (as much as possible) as well as (3) the existing (knowledge base creation from scratch, text-based, reuse-based or alignment/merge). (4) The verification of the created knowledge bases is done above all by design, as it happens.

Keywords

Ontology, Reasoners, Verbalizer

1 Introduction

Aujourd'hui, les entreprises comprennent les enjeux et les gains commerciaux liés aux produits et services à base de sémantique. Cependant, pour mettre en place des telles applications et services, il est nécessaire de développer le cœur de ces applications, la base de connaissances. Or, les connaissances sont propriétés des experts du domaine de l'application. Une des problématiques à résoudre est donc la création de base de connaissances par un expert domaine non expert des bases de connaissances.

Les outils support permettant aux experts de domaines de créer les bases de connaissances nécessaires aux applications et services à mettre en place doivent répondre à un certain nombre d'exigences fonctionnelles. L'outil doit prendre en compte (1) l'interaction nécessaire avec l'utilisateur (interaction linguistique grâce à un verbaliseur), (2) différentes logiques nécessaires à l'utilisateur (dans la mesure du possible) ainsi que (3) l'existant (création de base de connaissances de zéro, à base de texte, à base de réutilisation ou d'alignement/fusion). (4) La vérification des bases de connaissances créées doit se faire avant tout par conception, au fil de l'eau.

La partie 2 présente nos besoins. La partie 3 présente des travaux existants sur la création d'ontologie. Des méthodologies, outillées ou non, répondent à un principe sous-jacent qui implique en général le suivi d'étapes de création. La troisième partie présente l'outil développé et les premiers tests avant de conclure et de présenter les perspectives prenant en compte différents besoins opérationnels.

2 Besoins utilisateur

Nos utilisateurs travaillent dans le domaine de l'avionique. Ils sont experts, sans connaissance particulière des ontologies mais avec le plus souvent une forte connaissance en conception de systèmes (design authorities) ou en logiciel.

Ils ont donc besoin que leurs possibilités de conception ne soient pas entravées et que les outils proposés soient non seulement très flexibles mais s'intègrent à leurs activités.

Il s'agit aussi de ne pas les écraser avec l'acquisition rapide de nouvelles connaissances tant sur les ontologies et les raisonnements que sur les applications sémantiques en général. Les interactions Homme-Machine sont donc pensées avec soin pour une utilisation naturelle.

La fonctionnalité principale doit rapprocher les utilisateurs finaux de la construction de leur ontologie en leur expliquant les éléments qu'ils sont en train de représenter en langue naturelle. Toutefois, pour ce faire, les experts ont malgré tout besoin de fonctionnalités de bas niveaux de construction d'ontologies.

Pour résoudre cette problématique, nous avons développé un module nommé GLUON qui est l'acronyme de « Génération de Liens Unifiant une Ontologie » dans le but de rendre les ontologies utilisables par des non experts.

Comme notre objectif est de simplifier l'approche, nous avons

proposé en premier lieu le format d'ontologie le plus utilisé, à savoir OWL 2 DL. Le pont que nous construisons se situe entre un format logique et une interaction simple et naturelle.

3 Travaux antérieurs sur la création d'ontologie

Depuis les années 1990 [6], l'ontologie et sa création est scrutée, déconstruite pour être mieux reconstruite ou réutilisée. De Noy et Hafner en 1997 [8] à aujourd'hui, un grand nombre d'état de l'art sur la construction d'ontologies a vu le jour.

Les sources utilisées pour la construction des ressources terminologiques peuvent être structurées (base de données), semi-structurées (dictionnaire, corpus annoté) ou brutes (texte). De même, la constitution peut être semi-automatique ou guidée par des principes (manuelle) selon le but dans lequel l'ontologie s'inscrit, respectivement, son intégration dans un système transparent, ou sa construction elle-même comme objectif. Les méthodologies, outillées ou non, utilisées dans ces constructions répondent à un principe sous-jacent qui implique le suivi d'étapes de création. Le cycle classique de développement d'une ontologie est constitué des étapes de spécification, planification, conceptualisation, formalisation, implémentation. Cependant, des travaux récents font un retour d'expérience sur le mise en place de ces méthodologies. Dans [9] les auteurs concluent que le développement de la structure et des instances ne se produit pas séparément mais conjointement et qu'au final un cycle de développement stéréotypé ne peut être clairement identifié.

Ainsi, les méthodologies existantes, même déclinées en modules prenant en compte un cycle de construction itératif voire des étapes facultatives montrent leurs limitations quand il s'agit de prendre en compte les activités industrielles. Il est indispensable dans notre cas que la méthodologie considère un cycle de vie de l'ontologie de domaine dans la vie du produit ou service, et non seulement un cycle de construction. Cela sera fait dans un deuxième temps avec la construction d'un système complet encapsulé dans les plateformes métier existantes.

Le fait qu'un cycle de développement stéréotypé ne peut être clairement identifié est problématique et nous indique que pour respecter la liberté de conception des experts, il s'agit d'avoir une grande flexibilité dont la seule limite sera la vérification de la base de connaissances créée, de préférence par conception et au fil de l'eau, pour donner une souplesse à l'activité de l'expert. L'outil se définit donc par des fonctionnalités basiques de création des éléments constitutifs d'une ontologie, alliées à une vérification au fil de l'eau de sa constitution permise par l'utilisation de raisonneurs (un programme qui effectue un raisonnement sur une ontologie ou une base de connaissances). Le raisonneur s'appuie sur les connaissances ainsi que sur l'ensemble des règles de l'ontologie pour inférer (déduire logiquement) de nouvelles connaissances ajoutées au contenu de la base de connaissances.

Pour assurer la qualité de l'ontologie, il est nécessaire de gérer les incohérences et les incertitudes dans les ontologies dans les applications du monde réel [1]. Une ontologie incohérente signifie qu'une erreur ou un conflit existe dans une ontologie, et que certains concepts ne peuvent être interprétés correctement. La cohérence assure des liens ou des relations entre des concepts qui ont un sens explicite [3]. À titre d'exemple, on considère que deux classes déclarées disjointes n'héritent pas l'une de l'autre : une incohérence sera donc clairement levée si une instance doit hériter des deux classes disjointes à la fois. Par exemple, en avionique, un pilote effectuant un vol effectue sa « Mission ». Si Mission et Insatisfiable Mission sont deux classes disjointes et qu'une instance m:Mission est une instance d'un m':UnsatisfiableMission) alors la représentation est inadmissible logiquement et lèvera une erreur.

Après une étude détaillée des raisonneurs pour les ontologies [7], et à partir de nos différents critères pour trouver un bon compromis entre rapidité, expressivité et précision, Pellet, basé sur l'algorithme Tableau et supportant OWL2 DL est notre meilleur candidat. Néanmoins, pour les tests unitaires, nous utiliserons également ELK basé sur l'algorithme consequence-based, supportant un profil moins expressif de OWL, OWL2 EL, mais effectuant des traitements en un temps record, pour des tests plus légers. Une fois défini un raisonneur pour "comprendre" et lever les incohérences, il s'agit de l'inclure d'une façon transparente à notre outil afin que les utilisateurs puissent comprendre aussi sans être dépassés.

Afin d'offrir une solution viable entre la cohérence axiomatique et l'interaction avec un utilisateur, nous proposons d'utiliser un verbaliseur [7]. En effet, pour faciliter la compréhension des ontologies, plusieurs verbaliseurs d'ontologies ont été développés [11]. Les verbaliseurs traduisent généralement les axiomes de l'ontologie un par un dans un langage contrôlé malheureusement sans prêter attention à la cohérence du résultat en tant que texte.

ACE [4] est bien adapté à la verbalisation d'ontologies OWL, car les axiomes sont exprimés en phrases compactes n'explicitant que des phrases compatibles avec l'expressivité de OWL. OWL Verbalizer basé sur ACE permet une intégration dans notre plateforme. Une comparaison de systèmes de verbalisation OWL est disponible dans [7].

4 Spécifications de l'outil

Cette section traite de la solution proposée pour GLUON (Génération de Liens Unifiant une Ontologie) et la figure ci-après montre l'architecture de haut niveau de GLUON.

La nouveauté de la solution réside dans la l'intégration d'un verbaliseur OWL et dans le fait que des phrases explicatives sont générées pour décrire l'ontologie. GLUON est constitué d'un constructeur d'ontologie, d'un système de raisonnement et d'un générateur de langue naturelle (verbaliseur).

4.1 Constructeur d'ontologie

Ce composant permet de créer une ontologie à partir de rien (from scratch) ou de charger une ontologie déjà conçue via le gestionnaire d'ontologie, qui est un groupe d'instructions à partir d'un moteur d'ontologie. Ce bloc permet la création de concepts, de relations et d'individus à ajouter à l'ontologie en question, voire d'appliquer Opérations CRUD (Créer, Lire, Mettre à jour, Supprimer) sur le contenu de l'ontologie pour assurer la persistance des données ontologiques. L'utilisateur a également la possibilité d'ajouter des règles (ex. règles SWRL) à son ontologie en les adaptant au domaine et à la

logique métier qui organisent son domaine d'expertise. Concernant le moteur d'ontologie, nous avons opté pour l'API OWL comme bibliothèque de manipulation d'ontologies. Ce module est présent dans de nombreux outils et une grande communauté l'utilise, de sorte que la documentation et les références sont disponibles.

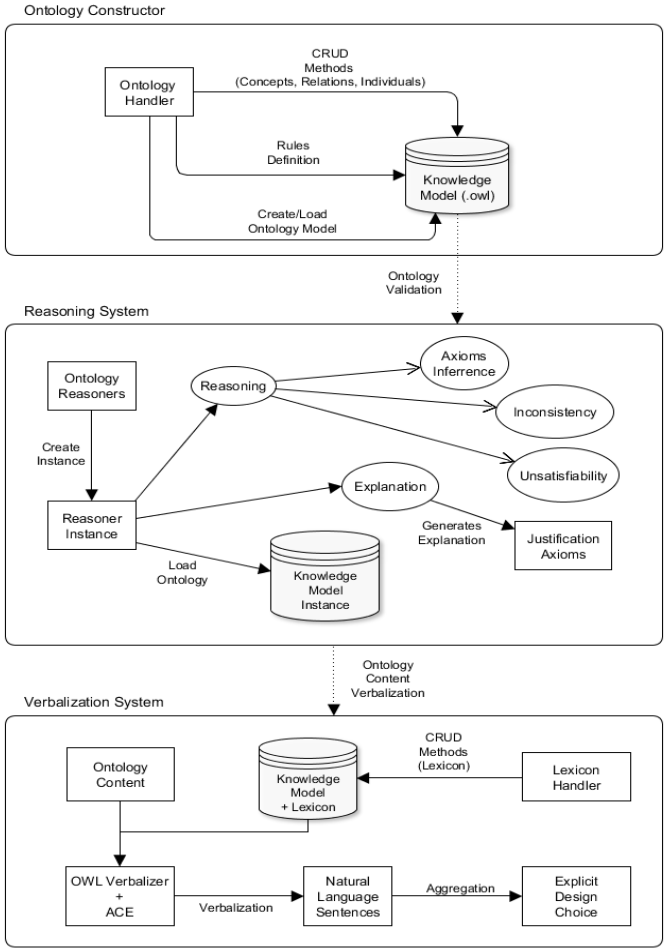

Figure 1 : Architecture de haut niveau de GLUON

4.2 Système de raisonnement

Ce composant est un ensemble de sous-systèmes permettant l'utilisation d'un raisonneur autonome ou de plusieurs raisonnements parallèles. En plus de raisonner sur une ontologie (inférences, détection des incohérences et insatisfiabilités), le raisonneur permet d'extraire les axiomes en OWL contenus dans l'ontologie. Ce module vérifie la cohérence de l'ontologie après chaque opération ou changement que l'utilisateur applique à l'ontologie en cours de conception. Le module génère des explications sous la forme d'axiome OWL ou d'une liste d'axiomes OWL. Cette explication est générée après chaque opération appliquée sur l'ontologie courante en utilisant le raisonneur définit par défaut dans un premier temps Openllet et à terme possibilité d'utiliser plusieurs raisonneurs, par exemple. Pellet, ELK, et bientôt DeLorean [2], DRAON [5] et STARE[1]...) pour la génération d'explications. Ces raisonneurs ont été choisis pour leur application à différents formats d'ontologie, leur utilisation de différents algorithmes de raisonnement ou encore leurs raisonnements sur des logiques différentes. Pour le moment, Pellet (Openllet) basé sur l'algorithme Tableau prend en charge OWL2 DL et ELK basé sur l'algorithme Consequence-based prend en charge OWL2 EL. Ce choix nous permet de couvrir dans un premier temps les expressivités DL et EL et permet également la justification, la prise en charge des règles SWRL, le raisonnement au niveau de la Abox et l'utilisation de l'API OWL dans le cadre de processus de raisonnement.

4.3 Système de verbalisation

Le système de verbalisation fait partie intégrante du système. Les phrases sont générées à partir de ce module avec le contenu de l'ontologie. Pour ce projet, nous avons utilisé OWL Verbalizer comme serveur HTTP et en même temps exploité les méthodes Java du module pour la verbalisation hybride. Les lexiques peuvent soit être générés à partir du noms associés aux concepts, relations et individus (par exemple #Aircraft, #flownWith, #MissionA, etc.) ou être définis par l'utilisateur via le module gestionnaire de lexique en l'adaptant aux différentes formes qu'il peut prendre (singulier, pluriel, passé forme participe pour les verbes, etc.). Noms, adjectifs et verbes peuvent être définis pour une meilleure génération ou importés en ressource prédéfinie grâce aux méthodes de gestion de lexique. Ainsi, OWL Verbalizer permet aux experts sans connaissance en logique de lire et comprendre les ontologies. Évidemment certains inconvénients subsistent, principalement la difficulté à agréger les phrases dans le cas de la verbalisation d'une liste d'axiomes.

5 Expérimentation de l'outil

5.1 Cas d'utilisation

Nous avons évalué l'utilisabilité et la performance des différents choix faits de GLUON sur la base de scénarios de validation proposés par des experts de l'Air Traffic Management. Ceux sont 3 cas d'utilisation (UC) d'une mission aérienne et modélisés sous forme d'ontologie :

- Défaillance du système
- Déviation en raison de la fermeture de la piste d'arrivée
- Événement météorologique

Chaque scénario reflète un incident, qui peut survenir pendant une mission aérienne. Voici la transcription ontologique du 1er scénario seulement pour exemple :

System failure (MissionSMAScenarioA)

Property assertions:

```
Individual: MissionSMAScenarioA
MissionSMAScenarioA Type Mission
MissionSMAScenarioA hasActiveAirportDeparture LFPB
MissionSMAScenarioA hasActiveAirportArrival KTEB
MissionSMAScenarioA hasActiveArrivalRunway KTEB06
MissionSMAScenarioA isFlownWith F-DEV1
F-DEV1 hasFuel FDEV1CurrentFuel
FDEV1CurrentFuel Type Fuel
FDEV1CurrentFuel hasFuelValue "0.0"
Mission DisjointWith UnsatisfiableMission
LowFuel SWRL Rule:
```

[1] https://gitlab.inria.fr/DLreasoners/stare

Mission(?m) ^ Fuel(?fcv) ^ IsFlownWith(?m, ?a) ^hasFuel(?a, ?fcv) ^ hasFuelValue(?fcv, 0.0) -> hasUrgency(?m, "3")

Urgency SWRL Rule:

Mission(?m) ^ hasUrgency(?m, ?urg) -> UnsatisfiableMission(?m)

Explanation: F-DEV1 has fuel with fuel value 0.0

Implies: MissionSMAScenarioA hasUrgency "3" -> MissionSMAScenarioA Type UnsatisfiableMission

5.2 Résultats

GLUON permet des sorties en Langage Contrôlé des axiomes OWL présentés ci-dessus. Notre plan d'évaluation est en deux volets. La première partie est le test unitaire sur chacun des scénarios ci-avant sur les besoins des experts en avionique. Nous présentons ici le scénario Alpha c'est-à-dire l'ontologie consistante puis le scénario révélant des échecs. Une évaluation sera à conduire dans une deuxième étape sur une nouvelle mission (Sick Passenger Onboard) par les experts en avionique sur une ontologie qu'ils auront créée avec GLUON.

Scénario Alpha : (Consistent ontology verbalization)

F-DEV1 is an Aircraft.
F-DEV1 has fuel FDEV1 current fuel.
FDEV1 current fuel is a fuel.
FDEV1 current fuel have fuel value 0.0.
KTEB is an airport.
KTEB06 is an operational runway.
LFPB is an airport.
Mission SMA A is a mission.
Mission SMA A has active airport arrival KTEB.
Mission SMA A has active airport departure LFPB.
Mission SMA A has active arrival runway KTEB06.
Mission SMA A is flown with F-DEV1.

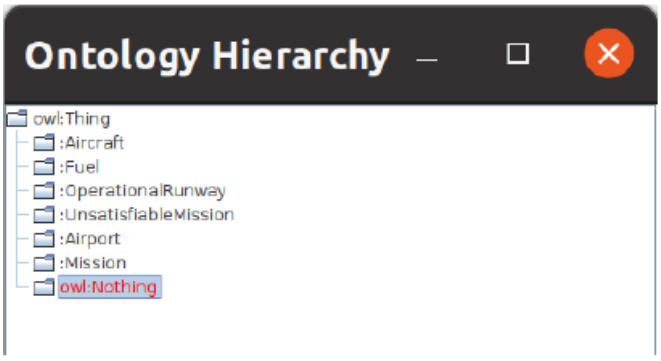

Figure 2: Hiérarchie de l'ontologie utilisée

Explication générée par GLUON pour le scenario « System failure inconsistency »:

You know that Mission SMA A is flown with F-DEV1 and FDEV1 current fuel has fuel value 0.0 and Mission SMA A is a mission and No mission is an unsatisfiable mission and F-DEV1 has fuel FDEV1 current fuel and FDEV1 current fuel is a fuel.

Les textes générés correspondent bien aux incohérences que les experts souhaitaient lever dans leurs 3 scénarios de missions aériennes. Malgré la simplicité de la plupart des axiomes dans notre ontologie, leur verbalisation est complexe. Certains axiomes complexes ne pourront être verbalisés par OWL Verbalizer.

Malgré tout, la présentation reste plus lisible que la syntaxe standard des logiques DLs.

6 Conclusion et perspectives

Permettant le partage d'information et l'explicabilité des résultats, les applications sémantiques sont un facteur clés de l'argumentation commerciale aujourd'hui. Afin de développer leur utilisation, GLUON permet aux experts industriels de concevoir et modéliser l'application proposée.

Pour cela, GLUON facilite l'interaction utilisateur/formalisme logique via un verbaliseur et à venir il proposera des modules de construction d'ontologies à base de texte et de données structurés (alignement d'ontologie, extension d'ontologies de haut-niveau). La problématique de vérification des ontologies est prise en compte en natif par la proposition d'explications des choix de conception de l'utilisateur mais des travaux méthodologiques sont en cours sur la validation et la vérification des bases de connaissances et des raisonnements.

Une première évaluation positive nous pousse à mettre en place un approfondissement des perspectives sur les axes raisonnement et logique et sur l'ajout de module d'alignement.

7 Références

[1] Abburu, S. (2012). A survey on ontology reasoners and comparison. International Journal of Computer Applications, 57:33–39

[2] Bobillo, F., Delgado, M., & Gómez-Romero, J. (2008). DeLorean: A Reasoner for Fuzzy OWL 1.1. In URSW.

[3] Euzenat, J., & Shvaiko, P. (2007). Ontology matching (Vol. 18). Heidelberg: Springer

[4] Kaljurand, K. (2007). Attempto controlled english as a semantic web language. University of Tartu

[5] Le Duc, C., Lamolle, M., Zimmermann, A., & Curé, O. (2013). DRAOn: A Distributed Reasoner for Aligned Ontologies. In ORE (pp. 81-86)

[6] Mars, NJI(Ed.), 1994. Workshop comparison of implemented ontologies, In ECAI, 8–12 08, Amsterdam, NL

[7] Mejdoul, Z. and Lortal, G. (2022). GLUON: A Reasoning-based and Natural Language Generation-based System to Explicit Ontology Design Choices. In IJC3K ISBN 978-989-758-614-9; SciTePress, pages 228-236.

[8] Noy, N. F., & Hafner, C. D. (1997). The State of the Art in Ontology Design: A Survey and Comparative Review. AI Magazine, 18(3), 53. https://doi.org/10.1609/aimag.v18i3.1306

[9] Reiz, A. and Sandkuhl, K. (2022). Debunking the Stereotypical Ontology Development Process. In IJC3K ISBN 978-989-758-614-9; SciTePress, pages 82-91

[10] Schwitter, R. (2010). Controlled natural languages for knowledge representation. In Coling 2010: Posters (pp. 1113-1121).