Formalisation en OWL pour vérifier les spécifications d'un environnement intelligent

Driss Sadoun, Catherine Dubois, Yacine Ghamri-Doudane, Brigitte Grau

Formalisation en OWL pour vérifier les spécifications d'un environnement intelligent

Driss Sadoun[1,2] Catherine Dubois[3,4] Yacine Ghamri-Doudane[3,5] Brigitte Grau[1,3]

[1]LIMSI/CNRS, B.P. 133 91403 Orsay Cedex, France
[2]Université Paris-Sud, 91400 Orsay, France
[3]ENSIIE, 1 square de la résistance, 91000 Evry, France
[4]CNAM-CEDRIC, 292 Rue St Martin FR-75141 Paris Cedex 03, France
[5]LIGM, Université Paris-Est-Marne-la-Vallée, 75420 Champs sur Marne, France

Résumé

De nos jours capteurs et effecteurs peuvent être utilisés de multiples façons et dans différents espaces, créant ainsi des environnements intelligents. Cet article décrit une formalisation d'un environnement intelligent et de son fonctionnement, qui permet de vérifier la consistance logique et la conformité de l'environnement. Cetteformalisation est faite à travers une ontologie représentant le domaine de connaissance, ses éléments seront instanciés à partir de textes en langue naturelle. Ces textes décrivent la configuration physique de l'environnement et le fonctionnement désiré par l'utilisateur de l'environnement. Nous avons choisi OWL pour représenter formellement notre environnement, augmenté de règles SWRL pour représenter l'aspect dynamique du fonctionnement du système et SQWRL pour interroger notre modèle conceptuel. Nous montrons comment la consistance et la conformité sont vérifiées grâce à ce formalisme.

Mots Clef

Conception d'ontologie ; spécification ; vérification formelle ; environnement intelligent.

Abstract

Nowadays sensors and actuators are increasingly used in different spaces, creating intelligent environment. This article aims to describe a formalisation of an intelligent environment and its operation, in order to check its consistency and its conformity. This is done through an ontology representing the domain knowledge, whose elements will be instantiated from natural language texts describing the physical configuration of an intelligent environment and a scenario describing the operation desired by the user of the environment. We chose OWL to represent formally our environment augmented with SWRL rules to represent the dynamic aspect of the operation system and SQWRL to query our conceptual model. We show how consistency and conformity are checked thanks to this formalism.

Keywords

Ontology conception ; specifications ; formal verification ; intelligent environment ;

1 Introduction

La profusion et la diversité des capteurs laisse imaginer une multitude de configurations de manière à ce qu'un environnement intelligent s'adapte aux besoins d'une personne. Le projet ENVIE VERTE [1] dans lequel se place le travail décrit dans cet aricle, a pour but de permettre le pilotage d'un environnement intelligent à l'aide de textes en langue naturelle, décrivant l'aspect physique de l'environnement (le nombre de capteurs, leur type, leur localisation, leurs interactions, ...), ainsi que les besoins utilisateurs (ne pas éclairer les pièces vides, éclairer une pièce à l'arrivée d'une personne, ...). Avant de déployer un tel système, il est nécessaire de vérifier sa conformité, i.e. l'environnement est correctement configuré par rapport aux spécifications utilisateurs, et sa cohérence logique, i.e. aucune contradiction n'apparaît dans son fonctionnement. Ces vérifications exigent des spécifications précises et non ambiguës de l'environnement et des besoins utilisateurs. Comme les textes en langue naturelle ne satisfont pas ces exigences, nous proposons de construire une représentation conceptuelle intermédiaire, qui fera le lien entre langue naturelle et spécifications formelles ([4] [11] [17]).

Notre premier objectif est donc de construire une représentation conceptuelle d'un environnement intelligent qui sera instanciée à partir de descriptions d'environnements et des exigences d'utilisateurs fournies sous forme de scénarios. Cette représentation conceptuelle devra permettre de générer des spécifications formelles qui seront vérifiées à l'aide de méthodes formelles, et d'en dériver le déploiement de la configuration du réseau de capteurs. A cette fin, nous proposons la création d'une ontologie de haut niveau en OWL [21], qui modélise les types d'éléments présents dans un environnement intelligent et leurs

1. financé par DIGITEO, projet DIM LSC 2010.

interactions. L'originalité de notre approche repose sur l'utilisation du formalisme logique de OWL pour représenter l'environnement et raisonner dessus, ce qui permet de vérifier sa cohérence et sa conformité. Cette modélisation contient deux parties : une partie statique qui conduit à créer des individus pour représenter un environnement donné, et une partie dynamique qui conduit à créer des règles pour la représentation des scénarios utilisateurs. La cohérence et la conformité sont vérifiées si les instanciations décidées à partir des textes sont réalisables, donc consistantes avec l'ontologie, et par la définition de requêtes systématiques sur la base de connaissances créées.

Cet article suit l'organisation suivante. D'abord nous décrivons dans la section 2 l'environnement intelligent, ses composantes et les deux types de descriptions textuelles. Section 3, nous précisons le type de vérifications envisagées. Dans la section 4 nous présentons le modèle conceptuel élaboré pour représenter les descriptions. Dans la section 5, nous montrons l'utilisation de notre modèle pour vérifier la cohérence des descriptions en langue naturelle sur un cas d'application. Dans la section 7, nous présentons l'état de l'art, avant de conclure et présenter les travaux futurs.

2 Environnement intelligent

L'environnement intelligent consiste en un ensemble d'objets communicants (capteurs, effecteurs et processus de contrôle) qui peut être vu comme un réseau de capteurs. Ces objets influencent le fonctionnement des équipements de l'environnement, sous des conditions bien définies. On peut distinguer une partie matérielle : les différents équipements, leur nombre, leur type, leur localisation etc. et une partie logicielle : la configuration du fonctionnement.

Ces deux aspects permettent de décrire le fonctionnement général d'un environnement intelligent :

- Un capteur détecte l'apparition d'un phénomène ou mesure un phénomène quantifiable dans un espace restreint.
- Un phénomène, pour être détecté ou mesuré par un capteur, doit être localisé dans la zone de capture du capteur et être du type perçu par le capteur (température, mouvement, ...)
- Un effecteur est fixé ou connecté à un appareil de l'environnement.
- Quand un phénomène (ou un ensemble de phénomènes) est mesuré ou détecté, un contrôle des informations collectées est effectué et peut conduire à l'activation d'un ou plusieurs effecteurs pour enclencher une ou plusieurs actions (allumer, éteindre, diminuer, augmenter) sur les appareils auxquels ils sont connectés.
- Un effecteur peut être activé par un capteur (ou un ensemble de capteurs) s'il est localisé dans la zone (leurs zones) de contrôle et est capable d'analyser les informations perçues et transmises par le(s) capteur(s).

Pour piloter son environnement, un utilisateur détermine, selon la configuration physique de l'environnement, les fonctions devant être installées pour satisfaire ses besoins. La figure 1 illustre comment un utilisateur pourra configurer son environnement à partir de textes. Dans le système complet, les descriptions seront analysées automatiquement pour instancier les concepts d'une ontologie en OWL représentant le réseau de capteurs et l'environnement dans lequel il est déployé. Cette représentation sera utilisée pour engendrer un modèle formel. La vérification du modèle résultant sera effectuée pour détecter les inconsistances et les informations manquantes et ainsi permettre de corriger et améliorer le modèle en interagissant avec l'utilisateur jusqu'à ce qu'il soit consistant et corresponde à ses besoins. Nous avons choisi de répartir les vérifications à mettre en place selon les formalismes choisis. Celles-ci seront donc effectuées en deux phases. Nous effectuerons d'abord les vérifications de cohérence et de conformité lors de la représentation en OWL des informations données en langage naturel. Ces vérifications sont bien adaptées à ce type de formalisme. Les vérifications de propriétés générales, plus lourdes et complexes, seront effectuées par application de méthodes formelles, dans un deuxième temps. Dans cet article nous nous concentrons sur la conception de l'ontologie et des vérifications qu'elle autorise.

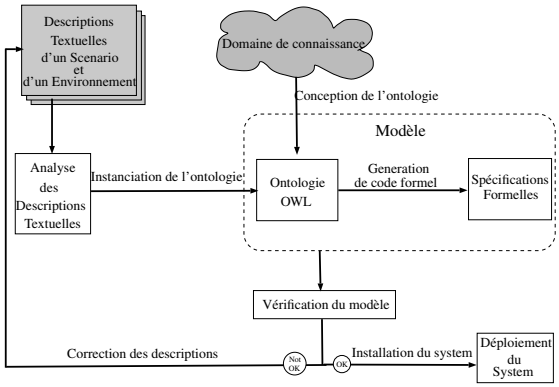

FIGURE 1 – Architecture du processus de configuration

Dans le cadre de notre projet les descriptions sont écrites en anglais. Ces descriptions peuvent être séparées en deux parties :

1. *Description de l'environnement intelligent* : décrit les composants de l'environnement (capteurs, effecteurs, processus physique, ...), leur nombre, leur type, leur localisation et leur manière d'interagir. Cette partie définit l'état statique de l'environnement et doit être analysée avant les besoins utilisateur.

   L'environnement intelligent décrit ci-dessous est représenté par la figure 2.

   Exemple : *L'appartement vert possède un couloir, deux chambres, une salle de bain, et un grand living qui contient une salle à manger et une cuisine. Chaque pièce est équipée de capteurs de mouvement. Chaque ampoule est équipée d'un effecteur.*

2. *Besoins utilisateur* : décrit comment et sous quelles conditions les objets de l'environnement doivent interagir. Cela permet de produire différentes instanciations de l'ontologie selon différents scénarios.
   Exemple : *Quand le mouvement d'une personne est détecté dans le salon, allumer le salon et la cuisine.*

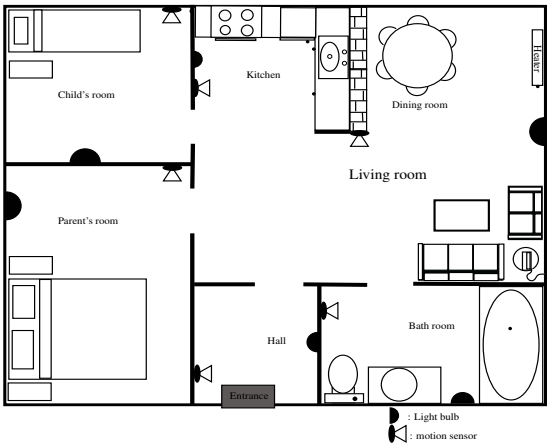

FIGURE 2 – L'appartement vert

3 Vérification de l'environnement intelligent

Avant le déploiement de l'environnement intelligent, il est important de vérifier que son fonctionnement est conforme aux spécifications. Ces vérifications seront réalisées à l'aide de méthodes formelles, qui requièrent des spécifications précises et non ambigües. Ainsi la représentation conceptuelle du fonctionnement de l'environnement ne doit contenir ni contradictions ni ambiguïtés. Elle doit de plus représenter toutes les propriétés de fonctionnement de l'environnement pour être admissible.

3.1 Modèle choisi

Nous avons décidé d'utiliser OWL (Web Language Ontology) pour la conception de notre modèle. OWL grâce à son formalisme logique basé sur les logiques de descriptions [2] offre une grande expressivité de représentation des connaissances [14]. OWL permet de raisonner sur une ontologie afin de vérifier sa consistance logique. OWL étendu par un Langage de Règles pour le Web Sémantique SWRL [13], permet de représenter l'aspect dynamique du fonctionnement de l'environnement. Le langage de requêtes SQWRL [19] permet d'interroger l'ontologie de manière à y détecter anomalies et informations manquantes.

3.2 Vérifications nécessaires

Dans ce travail, les vérifications portent sur deux aspects :

1. Consistance logique : le fonctionnement de l'environnement ne contient aucune contradiction ni incohérence ;
2. Conformité : le fonctionnement de l'environnement est conforme aux attentes des utilisateurs.

Ces deux aspects sont vérifiés sous deux angles :
– Vérification de la configuration physique ;
– Vérification des scénarios utilisateur.

Nous projetons aussi de vérifier l'état de l'environnement pour différents scénarios et vérifier des propriétés supplémentaires comme la sécurité, l'économie d'énergie, ...
Une part importante de ces vérifications est faite à l'aide d'OWL. Néanmoins OWL ne permettant pas le raisonnement non monotone, cela limite la portée des vérifications. Par exemple, il n'est pas possible de représenter le changement dynamique de l'état d'un individu. De plus la vérification des propriétés supplémentaires est en dehors de la portée d'OWL. Nous envisageons donc pour vérifier ces propriétés supplémentaires d'utiliser Focalize (`http://focal.inria.fr`) [10], atelier qui permet d'écrire des spécifications et de faire des preuves. Dans l'article nous montrons comment nous utilisons OWL dans ce contexte de vérifications.

4 Conception de l'ontologie

Une ontologie représente un domaine de connaissance de façon compréhensible pour un humain et un ordinateur. Elle est formée par un ensemble de concepts organisés hiérarchiquement et définis par des propriétés. Plusieurs études portent sur le développement d'ontologies pour représenter des réseaux de capteurs [20], [1], des environnements intelligents [12], ou le fonctionnement de composants de réseaux de capteurs[21]. Notre problématique nous amène à modéliser le fonctionnement d'un environnement intelligent à partir du fonctionnement d'un réseau de capteurs et de ses interactions avec les différents objets de l'environnement. Le but est d'identifier les concepts, individus et propriétés du domaine, ainsi que la représentation la plus appropriée pour représenter formellement le fonctionnement de l'environnement intelligent.
La modélisation de l'environnement étant moins soumise au changement que les besoins utilisateurs, la structure de l'ontologie (concepts et propriétés) peut être conçue et figée à partir de l'étude du domaine des environnements intelligents et des réseaux de capteurs. Au contraire, l'instanciation de l'ontologie variera en fonction des descriptions de la configuration physique de l'environnement et des besoins utilisateur et sera réalisée automatiquement. Les sections suivantes définissent les concepts et propriétés que nous avons retenus.

4.1 Concepts

Il apparaît dans la section 2 que nous avons besoin de définir les concepts qui représentent les *composants d'un réseau de capteurs*, les *localisations*, les *phénomènes* et *processus physiques*. Dans notre modèle, cf. figure 3, nous considérons que les composants du réseau de capteurs sont les capteurs et les effecteurs. Le fonctionnement des con-

trôleurs, qui mettent en relation capteurs et effecteurs, peut être modélisé à l'aide du formalisme logique d'OWL (contraintes et règles d'inférence). Nous faisons aussi la distinction entre deux types de phénomènes, ceux qui apparaissent soudainement (mouvement, fuite de gaz, ...) qui feront partie de la classe *Event* et ceux qui peuvent être mesurés (température, humidité, ...) qui appartiendront à la classe *Measurable*. Il s'ensuit que nous distinguons deux types de capteurs, ceux qui détectent et appartiennent à la classe *Detecting_sensor* et ceux qui mesurent et appartiennent à la classe *Measuring_sensor*.
La figure 3 montre l'organisation hiérarchique de l'ontologie. Il est à noter que nous ne représentons pas en détail les différents types de capteurs, ces connaissances n'intervenant pas dans les vérifications que nous effectuons. Ils seront pris en compte lors du déploiement.

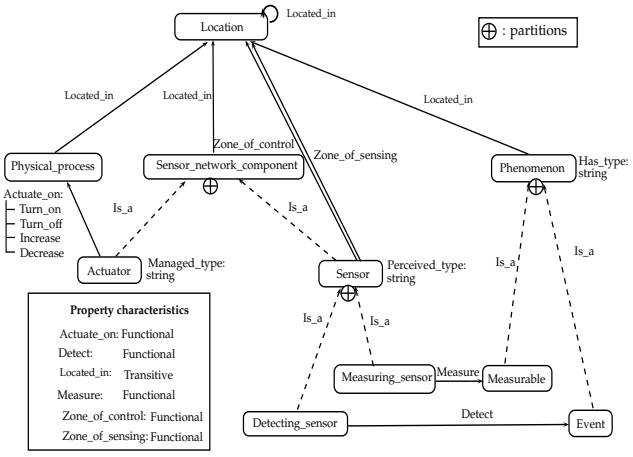

FIGURE 3 – L'ontologie

4.2 Propriétés

Les concepts sont définis par un ensemble de propriétés (cf. figure 3). Ci-dessous sont énumérées les propriétés qui modélisent le fonctionnement de l'environnement. (*entre parenthèses figure le nom de la propriété*) :
– Un phénomène a un type (*Has_type*).
– Un capteur a une localisation (*Located_in*), une zone de capture (*Zone_of_sensing*), une zone de contrôle (*Zone_of_control*) et un type de phénomène perçu (*Perceived_type*).
– Un élément de la classe *Measuring_sensor* mesure un mesurable (*Measure*).
– Un élément de la classe *Detecting_sensor* détecte un événement (*Detect*).
– Un effecteur actionne un processus physique (*Actuate_on*) et gère un ou plusieurs types de phénomènes (*Managed_type*)
La figure 3 montre comment les concepts sont reliés dans l'ontologie via les propriétés. Le lien *Is_a* décrit une taxonomie le long de laquelle les propriétés sont héritées.
La notion de type est utile pour associer un phénomène et les capteur et effecteur qui doivent participer au même processus. Ainsi nous pouvons garantir qu'un phénomène sera pris en charge par le capteur approprié qui activera l'actionneur approprié.
On distingue deux types de propriétés : i) les propriétés qui sont associées aux individus lors de l'instanciation de l'ontologie, *Located_in*, *Zone_of_sensing* et *Actuate_on*, qui doivent être définies dans les descriptions et seront utilisées dans le processus d'inférence ; ii) les propriétés qui sont générées par raisonnement, e.g. *Detect* et *Measure*, qui seront associées aux individus par les règles d'inférence.

4.3 Individus

La partie de l'ontologie soumise au changement est son instanciation selon un environnement spécifique et les besoins utilisateur. Les individus et leurs propriétés seront extraits automatiquement des descriptions d'environnement et de scénarios. Leurs propriétés seront utilisées pour classifier automatiquement chaque individu. Ainsi chaque pièce de l'environnement sera identifiée comme individu de la classe *Location*. Chaque phénomène détecté dans les textes sera classé comme individu de la classe*Event* ou de la classe *Measurable*. De cette façon chaque instance pertinente extraite des textes trouvera sa place en tant qu'individu de l'un des concepts de l'ontologie représentée dans la figure 3.

4.4 Raisonnement sur l'ontologie

Durant le raisonnement, des inférences sont faites, classifiant les individus de l'ontologie et créant de nouvelles assertions tout en maintenant la consistance logique.

Classification et Assertion. Les axiomes d'OWL sont utilisés pour organiser hiérarchiquement les concepts et pour classifier les individus. Soient C_1, C_2 deux concepts et P_1, P_2 deux propriétés. OWL définit deux types d'axiomes :
i) *Axiome d'inclusion* $C_1 \sqsubseteq C_2$ ($P_1 \sqsubseteq P_2$), e.g. $Detecting\_sensor \sqsubseteq Sensor$
Chaque élément de C_1 est un élément de C_2, i.e. C_1 est une sous-classe de C_2
ii) *Axiome d'égalité* $C_1 \equiv C_2$ ($P_1 \equiv P_2$), e.g. $Detecting\_sensor \equiv Sensor \sqcup \exists Detect.Event$
Chaque élément de C_1 est un élément de C_2 et vice versa.
Pour écrire des règles plus expressives, nous utilisons *SWRL*. Nous distinguons deux types de règles SWRL.
Règles générales : elles sont indépendantes des descriptions textuelles et représentent le comportement général d'un environnement intelligent.
Pour éviter d'écrire une règle pour chaque sorte de phénomène, nous modélisons deux règles (Règle 1, Règle 2) permettant de déduire qu'un phénomène, un capteur et un effecteur partagent le même type.

Règle 1 : Si le type d'un phénomène est le même que celui perçu par un capteur, alors ils partagent le même type.
$Has\_type(?p, ?t), Perceived\_type(?s, ?t)$
$\rightarrow Shared\_type(?p, ?s)$

Règle 2 : Si le type géré par un effecteur et le même que celui perçu par un capteur, alors ils partagent le même type.

$Managed\_type(?a,?t), Perceived\_type(?s,?t),$
$\rightarrow Shared\_type(?a,?s)$

Les règles 3 et 4 indiquent comment les capteurs détectent ou mesurent un phénomène.
Règle 3 : Un capteur détecte un événement s'il se produit dans sa zone de capture.
$Event(?e), Detecting\_sensor(?s), Shared\_type(?s,?e),$
$Located\_in(?e,?l), Zone\_of\_sensing(?s,?l)$
$\rightarrow Detect(?s,?e)$

Règle 4 : Un capteur mesure un phénomène mesurable si le phénomène est présent dans sa zone de capture.
$Measuring\_sensor(?s), Measurable(?m), Located\_in(?m,?l),$
$Shared\_type(?s,?m), Zone\_of\_sensing(?s,?l)$
$\rightarrow Measure(?s,?m)$

Règles générées : L'analyse textuelle permettra de générer automatiquement les règles représentant les besoins des utilisateurs. Nous avons modélisé ces règles et nous distinguons deux parties dans celles-ci, une partie fixe indépendante du texte et une partie générée à partir des descriptions (*cette seconde partie est soulignée dans les règles*). Elle correspond au choix de l'action appropriée à effectuer (allumer, éteindre, augmenter, diminuer).

Lorsqu'un capteur détecte un événement, l'effecteur du même type allume le ou les appareils qu'il contrôle.

$Actuator(?a), Physical\_process(?d), Actuate\_on(?a,?d),$
$Detect(?s,?e), Shared\_type(?s,?a), Located\_in(?a,?l),$
$Zone\_of\_control(?s,?l) \rightarrow \underline{Turn\_on(?a,?d)}$

Lorsqu'un capteur mesure une valeur supérieure à 30 par exemple, l'effecteur du même type diminue l'appareil qu'il contrôle.
$Actuator(?a), Physical\_process(?d), Actuate\_on(?a,?d),$
$Measure(?s,?m), Shared\_type(?s,?a), Located\_in(?a,?l),$
$Zone\_of\_control(?s,?l), Has\_value(?m,?v),$
$\underline{greaterThan(?v,30)} \rightarrow \underline{Reduce(?a,?d)}$

Vérification de la consistance. Classifications et assertions peuvent s'effectuer seulement si elles sont consistantes avec l'ontologie.
Soient C et C' deux classes, P une propriété, x et y deux individus.
- Si C' est donné ou déduit comme étant une sous-classe de C, il est nécessaire de vérifier que chaque individu de la classe C' peut appartenir à la classe C. Donc si un individu associé à C' ne peut appartenir à C alors l'ontologie est inconsistante.
- Si $C(x)$ est donné ou déduit, il est nécessaire de vérifier qu'il est possible pour un individu x d'appartenir à la classe C. Donc si x appartient à C' qui est disjoint avec C alors l'ontologie est inconsistante.
- Si $P(x,y)$ est donné ou déduit alors il est nécessaire de vérifier que x puisse appartenir au domaine de la propriété et y à son image.
Avec OWL, il est également possible d'empêcher qu'un effecteur effectue deux actions opposées simultanément (ex : turn_on(A, light_kitchen) et turn_off(A, light_kitchen)). En revanche, deux actions opposées, effectuées simultanément par deux effecteurs différents (ex : turn_on(A_1, light_kitchen) et turn_off(A_2, light_kitchen)) ne sont pas incohérentes au sens d'OWL.
Aussi, pour empêcher ce type d'apparition, nous utilisons SQWRL [19], un langage basé sur SWRL pour l'interrogation d'ontologies, fournissant des opérateurs pour la recherche de connaissances sous OWL. Cela nous permet d'effectuer des requêtes sur les connaissances et donc de pouvoir vérifier la présence d'actions opposées pouvant engendrer des inconsistances.
La détection d'inconsistances permettra de reconsidérer les analyses de textes réalisées ou de faire modifier ces textes par l'utilisateur.

Vérification de la conformité. Pour vérifier la conformité de l'environnement intelligent, nous avons défini des requêtes systématiques sur l'ontologie en SQWRL, à la recherche de spécifications incorrectes ou manquantes. Ces requêtes permettent par exemple de rechercher :
- les capteurs qui n'ont pas de zone de capture définie,
- les capteurs qui n'ont pas de zone de contrôle définie,
- les capteurs qui ne peuvent activer aucun effecteur,
- les effecteurs qui ne sont reliés à aucun appareil,
- les effecteurs qui ne reçoivent d'information d'aucun capteur.

Le résultat de ces requêtes permettra de corriger et d'améliorer les descriptions textuelles et le modèle.

5 Application

Nous détaillons un cas sur lequel appliquer notre modèle, afin de montrer comment l'instanciation et les vérifications sont effectuées. La création de l'ontologie a été réalisée sous *Protégé*, la consistance et la conformité sont implémentées sous *Jena*, une api Java permettant de créer et manipuler des ontologies en OWL.

5.1 Description du cas

Physical configuration description : *the green apartment includes a hall, two bedrooms, one bathroom and a living room which includes a kitchen and a dining room. The sensor S_{temp} measures the temperature of the living room. The sensor S_{move} detects movements in the living room. The living is equiped with a light bulb L_{living} on which an actuator AL_{living} is fixed, and the dining room is equipped with a heater H_{dining} on which an actuator AH_{dining} is fixed.*
Living room lighting scenario : *When a person movement is detected in the living room, turn on the lights of the living room.*

Living room heating scenario : *When a person movement is detected in the living room, and the temperature is below 20 degrees turn on the heater of the dining room.*

5.2 Vérification de la consistance

Les individus tirés de la description de l'environnement :
- Location : $dining\_room, living\_room, hall, ...$
- Sensor : S_{move}, S_{temp}
- Phenomenon : $person\_movement, temp_{living}$

– Physical_process : L_{living}, H_{dining}
– Actuator : AH_{dining}

Ci-dessous figurent des assertions devant être extraites des textes, suivies des déductions qu'elles permettent.

Partie statique

1) - « living room which includes a kitchen. »
représenté par $Located\_in(dining\_room, living\_room)$
- « dining_room is equipped with a light bulb. »
représenté par $Located\_in(L_{living}, dining\_room)$

Comme *Located_in* est transitif,
de $Located\_in(dining\_room, livingroom)$
et $Located\_in(L_{living}, living\_room)$
est déduit $Located\_in(L_{living}, livingroom)$

2) « S_{temp} measures the temperature of the living room. »
représenté par $Measure(S_{temp}, temp_{living})$

Comme le domaine de la propriété *Measure* est *Measuring_sensor*, de $Measure(S_{temp}, temp_{living})$ est déduit $Measuring\_sensor(S_{temp})$

Toutes les assertions ne sont pas détaillées car elles sont analogues à celles-ci.

Partie dynamique

1) « When a person movement is detected in the living room, »

$Event(person\_movement), Sensor(S_{move}),$
$Shared\_type(S_{move}, person\_movement),$
$Located\_in(person\_movement, living\_room),$
$Zone\_of\_sensing(S_{move}, living\_room)$
$\rightarrow Detect(S_{move}, person\_movement)$

Supposons que cette règle soit générée pour $Sensor(S_{temp})$. Puisque le domaine de *Detect* est *Detecting_sensor*, il sera inféré que S_{temp} est un *Detecting_sensor* et ainsi une inconsistance apparaîtra, car dans la formalisation *Detecting_sensor* et *Measuring_sensor* sont disjoints. Donc l'individu S_{temp} ne peut appartenir aux deux.

2) « When a person movement is detected in the living room, and the temperature is below 20 degree turn on the heater of the dining room. »

$Actuator(AH_{dining}), Located\_in(AH_{dining}, living\_room),$
$Actuate\_on(AH_{dining}, H_{dining}), Physical\_process(H_{dining}),$
$Detect(S_{move}, person\_movement),$
$Shared\_type(S_{move}, AH_{dining}),$
$Zone\_of\_control(S_{move}, living\_room),$
$Measure(S_{temp}, temperature),$
$Shared\_type(S_{temp}, AH_{dining}),$
$Zone\_of\_control(S_{temp}, living\_room),$
$Has\_value(temperature, ?v), lessThanOrEqual(?v, 19)$
$\rightarrow Detect(Increase(AH_{dining}, H_{dining})$

La propriété *Shared_type* est déduite par la règle 1 décrite dans 4.4. Les effecteurs peuvent réagir à différents types d'information. Les types de ces informations doivent être spécifiés dans la description de l'environnement.

Scénario incohérent. L'incohérence peut provenir du résultat du déclenchement de deux actions opposées.
Supposons qu'une deuxième règle déduise $Reduce(AH_{dining}, H_{dining})$. Puisque dans notre modèle, les propriétés *Increase* et *Reduce* sont disjointes, une inconsistance sera générée ; ceci garantit qu'un effecteur ne peut pour une même instance, effectuer des actions opposées.
Le cas où un autre effecteur réduit le radiateur au même moment $Reduce(another\_actuator, H_{dining})$ ne peut être empêché lors de déductions sous OWL. On résout le problème grâce à l'interrogation de l'ontologie.
Propriétés opposées : La requête suivante renvoie l'intersection des ensembles de propriétés *reduced* et *increased* sur les processus physiques. Si cette intersection n'est pas vide alors certaines propriétés sont opposées et donc le scénario doit être corrigé.
?s1 est l'ensemble des processus physiques qu'on réduit, *?s2* l'ensemble de ceux que l'on augmente et *?s3* leur intersection.

$Reduce(?a1, ?d1)sqwrl : makeSet(?s1, ?d1)$
$Increase(?a2, ?d2)sqwrl : makeSet(?s2, ?d2)$
$sqwrl : intersection(?s3, ?s1, ?s2)sqwrl : size(?n, ?s3)$
$\rightarrow sqwrl : select(?n)$

5.3 Vérification de la conformité

En interrogeant l'ontologie, on peut vérifier si l'environnement est correctement configuré ainsi qu'identifier les spécifications manquantes.

Spécifications manquantes. Les informations manquantes peuvent engendrer la non utilisation de certains appareils et donc amoindrir le potentiel de l'environnement.
Capteurs sans zone de capture : *?s1* est l'ensemble des capteurs dans l'ontologie, *?s2* est l'ensemble des capteurs qui possèdent une zone de capture et *?s3* l'ensemble de ceux qui n'en possèdent pas.

$Sensor(?s)sqwrl : makeSet(?s1, ?s)Sensor(?s\_zos)$
$Zone\_of\_sensing(?s\_zos, ?l)sqwrl : makeSet(?s2, ?s\_zos)$
$sqwrl : difference(?s3, ?s1, ?s2)sqwrl : size(?n, ?s3)$
$\rightarrow sqwrl : select(?n)$

Effecteurs non-connectés à un processus physique : ?*s*1 est l'ensemble des effecteurs de l'ontologie, ?*s*2 est l'ensemble des effecteurs qui peuvent actionner un processus physique et ?*s*3 est l'ensemble des effecteurs qui ne peuvent actionner aucun processus physique.

$Actuator(?a)sqwrl : makeSet(?s1, ?a)Actuate\_on(?ad, ?d)$
$sqwrl : makeSet(?s2, ?ad)sqwrl : difference(?s3, ?s1, ?s2)$
$sqwrl : size(?n, ?s3) \rightarrow sqwrl : select(?n)$

6 Discussion

L'exemple d'application ci-dessus montre l'intérêt d'exploiter ontologies et règles pour la représentation et la vérification des spécification données en langue naturelle. La définition manuelle de règles générales issues du domaine de connaissance qui décrivent de manière générique le

comportement des différents composants permet de fournir un cadre qui sera enrichi par des règles générées à partir de l'analyse des spécifications en langue naturelle. Ainsi la représentation conceptuelle ne repose pas seulement sur les connaissances issues des descriptions en langue et permet de lever les ambigüités, notamment lorsque la formulation du texte ne permet pas d'identifier un seul concept ou individu. Le cadre ainsi définit permet par exemple d'expliciter le fait qu'un composant qui mesure un phénomène est un *Measuring_sensor* même si il est désigné par un terme général, qu'une instance de composant ne peut à la fois être un *Measuring_sensor* et un *Detecting_sensor*, et qu'un effecteur ne peut effectuer en même temps deux actions opposées. Les règles produites sont formées d'une partie générique enrichie par des éléments reconnus à partir de l'analyse des spécifications. L'intérêt est d'allier flexibilité et bonne qualité de représentation.
Notre approche bien qu'appliquée au domaine des environnements intelligents est généralisable à tout domaine dans lequel on a à modéliser des interactions possibles.

7 État de l'art

Les modèles formels sont généralement issus de documents en langue naturelle. En effet la plupart des spécifications sont écrites en langue naturelle, ce qui leur permet d'être compréhensible par des non-experts du domaine. Donc la question du passage de l'informel au formel se pose naturellement. Plusieurs travaux portent sur ce point, on peut citer par exemple [17], [9], [11], [4]). Ces approches révèlent le besoin d'une représentation intermédiaire pour passer de spécifications informelles à des spécifications formelles. Dans [17] et [9], le projet EDEMOI (`http://vasco.imag.fr/EDEMOI`) a pour but d'appliquer différentes méthodes formelles pour la certification de documents de règlementation de sécurité d'un aéroport. Au préalable un modèle UML a été construit manuellement en analysant le document de réglementation. Dans [11] les auteurs utilisent des graphes conceptuels comme représentation intermédiaire pour formaliser les interactions de services de télécommunication et engendrer des spécifications formelles en notation Z. [4] présente l'utilisation des Two-Level Grammar (TLG), un langage de spécification des exigences [5], comme représentation intermédiaire pour passer de la langue naturelle à des spécifications formelles en VDM++.
Notre choix d'utiliser une ontologie OWL comme représentation conceptuelle intermédiaire est motivé par sa sémantique plus expressive que celles des approches citées et son formalisme logique permettant au système de raisonner sur les connaissances représentées. Cette représentation intermédiaire nous permet de faire un grand nombre de vérifications, et d'avoir recours aux méthodes formelles seulement pour prouver d'autres propriétés telles que la sécurité ou l'économie d'énergie.
La construction automatique d'une ontologie à partir des textes n'est pas une tâche facile. Une approche complètement automatique n'est pas réaliste. Identifier les concepts pertinents d'un domaine est trop difficile pour être fait sans l'aide d'un expert du domaine. C'est pourquoi la plupart des méthodes et outils de construction d'ontologie sont semi-automatiques ([3], [8], [7], [6]). Notre modèle ne reposant pas sur une organisation complexe de nombreux concepts et relations, nous n'avons pas eu à recourir à ces outils et méthodes.
Les ontologies en [20] et [1]sont analogues à des bases de données pour l'interrogation et la recherche de capteurs. Dans [20], tous les genres de capteurs sont modélisés selon leur type de phénomène perçu et [1] décrit les fonctionnalités des capteurs et leurs états courants.
[16] utilise une ontologie pour exprimer l'état d'un environnement et celui d'une personne assistée sans s'intéresser au processus de raisonnement.
L'approche présentée dans [21] est plus proche de nos travaux dans le sens où les auteurs proposent une ontologie pour la découverte d'effecteurs et définissent des comportements de haut niveau. En revanche leur modélisation nécessite la description de sous-classes d'appareils, ce qui implique de définir chacun d'eux, et peut être incomplète. De la même façon, elle nécessite la définition des différents types d'opérations, qui sont des propriétés génériques dans notre modèle qui sont inférées selon le type de phénomène perçu.
Plusieurs travaux ont déjà fait le lien entre ontologies et ingénierie des exigences comme par exemple [15] [18]. [15] représente des connaissances d'ingénierie à l'aide d'une ontologie, afin de vérifier à l'aide d'axiomes différentes contraintes. La représentation qu'ils proposent n'a pas vocation au changement ou à l'interaction avec un utilisateur. Notre ambition est de permettre à un utilisateur non-expert de piloter son propre environnement, de reconnaitre les spécifications erronées et incomplètes de sorte à l'aider à les améliorer. Ainsi, notre approche trouve son originalité dans l'instanciation automatique de l'ontologie et la production dynamique de règles à partir de l'analyse automatique des descriptions en langue naturelle, qui permettent de faire des vérifications. Dans notre approche, ce que nous avons besoin de vérifier n'est pas le modèle des connaissances (l'ontologie) mais son instanciation, c'est-à-dire les descriptions de la configuration de l'environnement et des besoins utilisateurs.

8 Conclusion et travaux futurs

Nous avons décrit une méthode de représentation conceptuelle permettant de faciliter le passage d'une description en langue naturelle à une spécification formelle et de vérifier tout au long du processus la consistance et la conformité du modèle conceptuel à l'aide d'une ontologie en OWL. Cette formalisation autorise la navigation entre textes et spécifications formelles pour corriger et améliorer les descriptions textuelles. Nous avons appliqué ce formalisme au domaine de la domotique en décrivant une représentation conceptuelle d'un environnement intelligent

et de son fonctionnement. Le modèle proposé représente d'une part la partie statique de l'environnement et permet de vérifier la conformité et la consistance du système. La partie dynamique permet de créer des règles représentant des scénarios utilisateur pour vérifier leur consistance.
Cette représentation conceptuelle qui fait le lien entre langue naturelle et spécification formelle, est une première étape dans le processus de développement d'un outil simple et sûr pour la configuration d'un environnement intelligent. Nos travaux futurs vont se concentrer sur l'analyse automatique de descriptions et le développement du processus interactif pour traduire les besoins d'un utilisateur.

Références

[1] Avancha, S., Patel, C., Joshi, A. : Ontology-driven adaptive sensor networks. In : Proceedings of the First Annual International Conference on Mobile and Ubiquitous Systems : Networking and Services (MobiQuitous'04. pp. 194–202 (2004)

[2] Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.) : The Description Logic Handbook : Theory, Implementation, and Applications. Cambridge University Press (2003)

[3] Biebow, B., Szulman, S. : Terminae : a method and a tool to build a domain ontology. In : Proc. of International Workshop on Ontological Engineering on the Global Information Infrastructure, TIA 1999

[4] Bryant, B., Lee, B.S. : Two-level grammar as an object-oriented requirements specification language. In : Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02)-Volume 9 - Volume 9. pp. 280–. HICSS '02, IEEE Computer Society, Washington, DC, USA (2002)

[5] Bryant, B.R., Johnson, D., Edupuganty, B. : Formal specification of natural language syntax using two-level grammar. In : Proceedings of the 11th coference on Computational linguistics. pp. 527–532. COLING '86, Association for Computational Linguistics, Stroudsburg, PA, USA (1986)

[6] Buitelaar, P., Olejnik, D., Sintek, M. : OntoLT : A Protege Plug-In for Ontology Extraction from Text. In : Proceedings of the International Semantic Web Conference (ISWC). Florida, USA (2003)

[7] Buitelaar, P., Olejnik, D., Sintek, M. : A protege plug-in for ontology extraction from text based on linguistic analysis. In : In Proceedings of the 1st European Semantic Web Symposium (ESWS (2004)

[8] Cimiano, P., Völker, J. : Text2onto - a framework for ontology learning and data-driven change discovery. Lecture Notes in Computer Science, vol. 3513, pp. 227–238. Springer, Alicante, Spain (2005)

[9] Delahaye, D., Étienne, J.F., Viguié Donzeau-Gouge, V. : Certifying airport security regulations using the Focal environment. In : Misra, J., Nipkow, T., Sekerinski, E. (eds.) Formal Methods (FM). Lecture Notes in Computer Science (LNCS), vol. 4085, pp. 48–63. Springer, Hamilton (Ontario, Canada) (2006)

[10] Dubois, C., Hardin, T., Donzeau-Gouge, V. : Building certified components within focal. In : Trends in Functional Programming. pp. 33–48 (2004)

[11] Fougères, A.J., Trigano, P. : Rédaction de spécifications formelles : élaboration à partir des spécifications écrites en langage naturel. In Cognito - Cahiers Romans de Sciences Cognitives 1(8), 29–36 (1997)

[12] Gu, T., Wang, X.H., Pung, H.K., Zhang, D.Q. : An ontology-based context model in intelligent environments. In : In proceedings of communication networkd and distributed systems modeling and simulation conference. pp. 270–275 (2004)

[13] Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M. : Swrl : A semantic web rule language combining owl and ruleml. W3c member submission, World Wide Web Consortium (2004)

[14] Horrocks, I., Patel-schneider, P.F., Mcguinness, D.L., Welty, C.A. : Owl : a description logic based ontology language for the semantic web (2007)

[15] Jureta, I.J., Mylopoulos, J., Faulkner, S. : A core ontology for requirements. Appl. Ontol. 4, 169–244 (August 2009)

[16] Klein, M., Schmidt, A., Lauer, R. : Ontology-centred design of an ambient middleware for assisted living : The case of soprano. In : Towards Ambient Intelligence : Methods for Cooperating Ensembles in Ubiquitous Environments (AIM-CU), 30th Annual German Conference on Artificial Intelligence (KI 2007

[17] Laleau, R., Vignes, S., Ledru, Y., Lemoine, M., Bert, D., ccjuvetu, V., Dubois, C., Peureux, F. : Application of requirements engineering techniques to the analysis of civil aviation security standards. In : SREP'05 International Workshop, In conjunction with 13th IEEE International Requirements Engineering (2005)

[18] Lin, J., Fox, M.S., Bilgic, T. : A requirement ontology for engineering design. Concurrent Engineering : Research and Applications 4, 279–291 (1996)

[19] O'Connor, M.J., Das, A.K. : Sqwrl : A query language for owl. In : OWL : Experiences and Directions (OWLED), Fifth International Workshop, Chantilly, VA. (2009)

[20] Russomanno, D., Kothari, C., Thomas, O. : Sensor ontologies : from shallow to deep models. In : System Theory, SSST '05. Proceedings of the Thirty-Seventh Southeastern Symposium (2005)

[21] Wang, F., Turner, K.J. : An ontology-based actuator discovery and invocation framework in home care systems. In : Proceedings of the 7th International Conference on Smart Homes and Health Telematics : Ambient Assistive Health and Wellness Management in the Heart of the City. pp. 66–73. ICOST '09, Springer-Verlag, Berlin, Heidelberg (2009)