Rôle d'une base de connaissance dans SemIoTics, un système autonome contrôlant un appartement connecté

Nicolas Seydoux, Khalil Drira, Nathalie Hernandez, Thierry Monteil

Rôle d'une base de connaissance dans SemIoTics, un système autonome contrôlant un appartement connecté

Nicolas Seydoux[1,2,3], Khalil Drira[2,3], Nathalie Hernandez[1], Thierry Monteil[2,3]

[1] IRIT Maison de la Recherche, Univ. Toulouse Jean Jaurès,
5 allées Antonio Machado, F-31000 Toulouse
{nseydoux,hernande}@irit.fr

[2] CNRS, LAAS, 7 avenue du Colonel Roche,
F-31400 Toulouse, France
{nseydoux,khalil,monteil}@laas.fr

[3] Univ de Toulouse, INSA, LAAS, F-31400, Toulouse, France

Abstract : L'Internet des Objets représente une réalité de plus en plus concrète au fur et à mesure que se déploient de larges réseaux d'objets connectés. Ceux-ci ouvrent de larges perspectives d'applications, mais rencontrent des difficultés en terme d'interopérabilité, de configuration ou de passage à l'échelle. Ces problématiques peuvent être traitées par le recours aux principes du web de données liées, d'où l'émergence d'ontologies dédiées aux applications de l'IoT, comme IoT-O, une ontologie pour l'IoT.Par ailleurs, une description enrichie des systèmes permet d'envisager leur configuration autonome : on parle alors d'autonomic computing. Ce papier présente SemIoTics, un système autonome reposant sur des bases de connaissance pour la gestion d'un appartement connecté. Nous présentons tout d'abord une vision générique d'une architecture de réseaux d'objets connectés qui permet de guider une analyse des travaux à l'interface du web sémantique et de l'IoT. Nous décrivons ensuite les deux bases de connaissances spécialisant IoT-O sur lesquelles s'appuie SemIoTics, et leur relation avec le dispositif expérimental. Enfin, la structure de ce système autonome de domotique est présenté en détails, et mis en relation avec l'architecture identifiée dans l'état de l'art.
Mots-clés : Web sémantique, Internet des Objets, Autonomic computing, Domotique

1 Introduction

L'Internet des Objets, Internet of Things (IoT) en anglais, désigne des réseaux d'objets connectés communiquant les uns avec les autres pour étendre leurs fonctionnalités Gubbi *et al.* (2013). Dans un premier temps plus axés sur les capteurs connectés Barnaghi *et al.* (2012), les réseaux d'objets tendent à inclure des objets de plus en plus variés : objets collectant l'énergie, actionneurs (objets qui agissent sur le monde)... Le développement important de l'IoT permet d'envisager des applications dans des domaines nombreux : ville intelligente, agriculture, domotique, usine intelligente, télésanté, etc. L'hétérogénéité des domaines d'application tend à poser des problèmes d'interoperabilité entre les solutions, dites verticales, développées dans une approche orientée silos. Cette problématique d'interopérabilité se pose à deux niveaux : l'interoperabilité architecturale et l'interopérabilité sémantique, décrites dans Gyrard *et al.* (2015). L'interopérabilité architecturale est composée de l'interoperabilité technique et organisationnelle, et les efforts de standardisation actuels tentent de la résoudre. L'interopébilité sémantique se pose sur le sens associé aux interactions entre les objets (appels de services, découverte de fonctionnalités...) ainsi qu'aux données qu'ils échangent. C'est sur cette forme d'interopérabilité que se concentre la communauté du web sémantique, en appliquant des principes et des technologies rendant les données plus compréhensibles par les systèmes.

De plus, résoudre ce problème d'interopérabilité amène à une autre problématique majeure de l'IoT : la complexité des systèmes. En effet, comme le soulignent Zanella *et al.* (2014), Barnaghi *et al.* (2012) ou Foteinos *et al.* (2013), la grande hétérogénéité des composants d'un système d'objets connectés en fait des entités complexes à gérer, surtout à grande échelle. Plus les interactions se multiplient, plus les technologies mises en jeu sont variées, et plus la gestion du système par des opérateurs humains est difficile et coûteuse. À partir de ce constat, qui englobe mais ne se limite pas au domaine des objets connectés, Kephart & Chess (2003) propose un nouveau paradigme, l'autonomic computing, qui vise à rendre possible l'auto-configuration ou l'auto-réparation des systèmes. Les comportements autonomes sont guidés par des politiques de haut niveau définies par les opérateurs humains, et par la connaissance que le système possède sur ses composants. Nous proposons dans ce papier de représenter cette connaissance à partir des formalismes du web sémantique, ce qui permet de traiter à la fois la problématique de la complexité du système par l'autonomic computing et la problématique de l'interoperabilité.

Pour appuyer ce double apport du web sémantique à l'IoT, nous présentons SemIoTics, un système autonome de contrôle d'objets connectés guidé par une Base de connaissances (BC). Nous démontrons que SemIoTics, élaboré de façon générique décorrélée de tout domaine d'application particulier, est applicable au domaine spécifique de la domotique. Nous étudions pour cela un cas d'utilisation reposant sur le contrôle d'un appartement connecté. Le reste de ce papier est structuré comme suit. Tout d'abord, LMU-N (Lower, Middle and Upper Node), une structuration architecturale générique pour l'IoT, est proposée pour structurer la présentation de l'état de l'art sur l'intégration des principes et des technologies du web sémantique dans les réseaux d'objets connectés. Ensuite, après des rappels concernant l'ontologie IoT-O proposée dans Seydoux *et al.* (2015), nous décrivons la connaissance exploitée par le système qui reposent sur des modules spécialisant IoT-O. Enfin, l'architecture de SemIoTics est présentée, et mise en relation avec LMU-N.

2 LMU-N : une architecture pour classifier les contributions du web sémantique à l'IoT

2.1 Motivations et caractéristiques de la classification guidée par LMU-N

La notion d'IoT est fondée sur la notion de réseau d'objets connectés, un graphe où les sommets (que l'on appellera ici noeuds) sont les objets, et les arcs les interactions qui existent entre eux. Cette vision des objets connectés sous forme de noeuds amène une unification de deux composants fondamentaux de l'IoT De *et al.* (2011) : l'objet et le service. En effet, un noeud du réseau d'objets peut être vu comme un service : son interface est connue de ses noeuds voisins, et son implémentation sous-jacente peut être associée à un objet matériel comme elle peut être purement logicielle. Un arc est établi entre deux noeuds quand des données ou des services sont échangées entre les deux, ou plus simplement quand au moins un des deux noeuds a conscience de l'existence de l'autre et peut accéder à son interface. Les arcs sont donc orientés, depuis le noeud capable d'initier l'interaction vers l'autre.

Les noeuds d'un IoT sont très divers en terme de puissance de calculs, de proximité avec le monde physique ou de capacités de communication. La puissance de calcul représente la capacité de l'objet à appliquer des traitements complexes à des données en masse, ainsi que la complexité des types de données supportés. L'étendue des capacités de communications est mesurée à l'aulne du nombre de protocoles que l'objet utilise pour communiquer, de la bande

passante dont il dispose, et de sa disponibilité. La proximité avec le monde physique se caractérise par le nombre d'arcs à parcourir depuis le noeud considéré pour effectuer une action ayant directement prise avec l'environnement, que ce soit pour interagir avec un noeud effectuant une mesure (par exemple, accéder à une donnée issue d'un capteur de température) ou pour actionner un noeud permettant d'effectuer une modification sur le comportement d'un objet (par exemple lors de l'allumage ou le réglage d'un radiateur). À partir de ces trois paramètres (puissance, proximité avec le monde physique, capacités de communication), trois catégories homogènes de noeuds se dégagent, comme illustré dans la table 1 :

- Les **noeuds de haut niveau**, typiquement serveurs ou ordinateurs de bureau ou portables, représentent les noeuds faiblement contraints de l'architecture de Zanella *et al.* (2014), ou les noeuds de traitement de données dans Liu *et al.* (2015).

- Les **noeuds médians**, qui assurent la connexion entre les noeuds de haut niveau et les noeuds de bas niveau, sont souvent associés à la notion de gateway comme dans Ben-Alaya *et al.* (2015) et Desai *et al.* (2015). Ces noeuds n'ont pas pour rôle principal de traiter les données, mais de transformer les informations depuis les noeuds de haut niveau pour leur utilisation par les noeuds de bas niveau.

- Les **noeuds de bas niveau** représentent tous les objets ou programmes connectés qui sont le plus directement en prise avec le monde physique, mais très contraints en terme de puissance, de consommation et de communication. Ces objets sont par définition présents dans toutes les architectures d'IoT.

LMU-N (Lower, Middle and Upper Node) est le nom de cette architecture générique en trois niveaux de noeuds. LMU-N recouvre donc les architectures proposées dans Zanella *et al.* (2014), Liu *et al.* (2015), Desai *et al.* (2015), Ben-Alaya *et al.* (2015),Nikoli *et al.* (2011), Mrissa *et al.* (2015)Pour un noeud d'un niveau donné, on constate dans les architectures analysées qu'il aura des contacts privilégiés avec un petit nombre de noeuds de niveau supérieur, et avec de multiples noeuds du niveau inférieur. De plus, les connexions entre noeuds de même niveau vont croissant avec la "hauteur" de ce niveau. Un schéma représentant cette architecture générique est représenté sur la figure 2. En unifiant l'objet au service, et donc le réel au virtuel, la notion de noeud permet aussi l'abstraction des objets réels élémentaires en objets virtuels plus élaborés. Cette abstraction, proposée dans Foteinos *et al.* (2013), permet de composer des objets et services réels en des noeuds composites. À l'inverse, un noeud complexe peut être abstrait et décomposé en un ensemble de noeuds plus simples, chacun exposant une sous-partie cohérentes de ses fonctionnalités.

Toutefois, identifier ces trois niveaux d'architectures n'est pas suffisant pour situer précisément les façons dont le web sémantique permet de lever les verrous de l'IoT. Dans la suite de cette section, les flux opérationnels dans LMU-N ainsi que les connaissances qu'ils mettent en jeu sont caractérisés et mis en relation avec les travaux de l'état de l'art.

2.2 Classification des apports du web sémantique à l'IoT sous forme de flux dans LMU-N

La notion de noeud de réseau s'articule avec la notion de flux opérationnel (workflow en anglais) partiellement définie dans Poslad *et al.* (2015). Ces flux sont de deux types : montants (up-

Type de noeud	Puissance de calcul	Capacités de communication	Proximité avec le monde physique	Exemples
Noeud de haut niveau	Importante	Étendues (bande passante importante, protocoles supportés variés...)	Très faible	Serveurs, smartphones, ordinateurs
Noeud médian	Moyenne à faible	Étendues (bande passante importante, protocoles supportés variés...)	Faible	Gateway, box domestique
Noeud de bas niveau	Faible à très faible	Limitées (peu de bande passante, support de protocoles contraints uniquement)	Très forte	Capteur, actionneur

Figure 1: Caractérisation des types de noeuds d'un réseau d'objets connectés

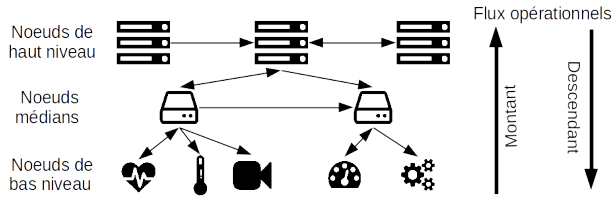

Figure 2: Représentation de LMU-N, de ses trois niveaux de noeuds et ses flux opérationnels

stream) si le noeud émetteur est de niveau inférieur au noeud récepteur, et descendant (downstream) dans le cas contraire. Cette distinction est rendue nécessaire par les contraintes liées aux différents niveau de noeuds. Ces flux opérationnels mettent en oeuvre des connaissances de nature variée, décrites par des ontologies. Celles-ci sont de deux types : les ontologies spécifiques à un domaine d'application, et les ontologies core-domaine dédiées à la description des objets et des services qu'ils offrent d'une manière décorrélée de l'application. Un grand nombre d'ontologies de ces deux types sont répertoriés dans le LOV4IoT[1], initiative visant à favoriser la réutilisation d'ontologies déjà existantes.

- **Enrichissement montant et traduction descendante** : Ce flux opérationnel porte sur les données manipulées par le système, dont le niveau d'enrichissement doit être modulé selon le niveau du noeud la manipulant. La remontée d'information permet d'enrichir une donnée pour la transformer en information, voire en connaissance, par l'annotation sémantique, comme dans Sheth *et al.* (2008), Le-phuoc & Hauswirth (2009) ou Poslad *et al.* (2015). À l'inverse, un noeud de haut niveau peut transformer une information pour l'exprimer sous une forme compréhensible par un noeud de niveau inférieur, tout en devant garantir son intégrité. Les ontologies mises en jeu sont principalement des ontologies de domaine en lien avec les domaines visés par les systèmes, ainsi que des ontologies dédiées à l'IoT permettant de décrire comment les données sont liées au système, comme dans Sheth *et al.* (2008).

- **Exposition montante et découverte descendante** : Pour que des noeuds distants puissent accéder à son interface, tout noeud doit procéder à une étape d'exposition de celle-ci. Dans le cas des noeuds de bas niveau, il est possible qu'un noeud intermédiaire assure

[1] `http://www.sensormeasurement.appspot.com/?p=ontologies#home`

leur exposition, en général un noeud médian. Le noeud intermédiaire est dans ce cas dépositaire de connaissance concernant le noeud de bas niveau, qu'il contextualise et rend accessible, agissant comme un proxy Nikoli *et al.* (2011). La découverte est l'opération complémentaire à l'exposition, puisqu'elle consiste à prendre conscience des interfaces exposées. La découverte dynamique de noeuds est conditionnée par leur description compréhensible, permise par un vocabulaire partagé, comme souligné dans Barnaghi *et al.* (2012). Ce flux opérationnel porte sur les connaissances représentant les objets constituants le système, et s'appuie donc principalement sur des ontologies de core-domaine de l'IoT. Les connaissances échangées décrivent les capacités des noeuds, les services qu'ils offrent, leur état de fonctionnement, les moyens par lesquels ils peuvent être contactés... L'ensemble de ces connaissances sur les noeuds permet, la découverte passée, une sélection guidée par un besoin applicatif, comme dans Perera *et al.* (2014) où des objets sont sélectionnés à partir d'une description sémantique.

- **Notification montante et contrôle descendant** : dans un réseau d'objets connectés, les objets les plus complexes commandent les objets les plus simples. Par exemple, les applications sur smartphones allument les lampes ou collectent les informations de capteur. Pour éviter la scrutation active, certaines architectures incluent un mécanisme de notification qui permet à un noeud de bas niveau de faire passer une information à un noeud de haut niveau, comme prévu dans le standard oneM2M[2] par exemple. Ce flux opérationnel peut être associé au premier flux pour enrichir les notifications et traduire les commandes.

Ces trois types de flux opérationnels sont génériques à l'IoT, et permettent de caractériser les apports des principes et des techniques du web sémantique à l'IoT. Ces flux peuvent être composés pour construire des applications complexes : dans la section suivante, nous établissons un parallèle entre ces flux et une boucle générique d'autonomic computing.

2.3 Décomposition d'une boucle autonomique basée sur une base de connaissance dans LMU-N

L'autonomic computing est un paradigme de programmation qui vise à limiter l'intervention humaine dans le fonctionnement des systèmes complexes Kephart & Chess (2003). Des propriétés de configuration autonome Chatzigiannakis *et al.* (2012) ou de gestion autonome Vlacheas *et al.* (2013) par exemple facilitent la mise en oeuvre à grande échelle de systèmes d'objets connectés hétérogènes. Le rôle de l'opérateur humain est transformé : il fixe des objectifs de haut niveau que le système va répercuter de manière cohérente sur ses différents composants. La gestion autonome d'un système s'appuie sur une boucle générique, dite boucle MAPE-K Kephart & Chess (2003) : Monitoring, Analysis, Planning, Execution, le tout lié par une BC (Knowledge). La boucle de gestion autonomique est une approche tout à fait générique, abstraite de tout domaine d'application. Elle peut donc être utilisée pour guider le comportement d'un réseau d'objets connectés. Dans ce cas, les différentes étapes d'une boucle MAPE-K peuvent être rapportées à des flux dans l'architecture LMU-N :

- L'**observation** (Monitoring) est un **flux montant de notification et d'enrichissement**

[2] `http://onem2m.org/`

d'information. Les données brutes produites par les capteurs sont enrichies et stockées dans la BC de l'agent, et entrent ainsi dans son processus de prise de décision.

- L'étape d'**analyse** (Analysis) s'appuie sur du filtrage de données, du raisonnement et sur les connaissances de l'agent pour donner un sens aux signaux qu'il a observés. Elle est effectuée dans un noeud médian ou de haut niveau.
- L'étape de **planification** peut avoir lieu dans des noeuds médians ou des noeuds de haut niveau. Elle consiste en une **découverte** de noeuds de bas niveau et en une prise de décision répondant à l'analyse de la situation, guidée par les **politiques de haut niveau** définies par l'administrateur. L'ensemble des connaissances nécessaires à cette étape est stocké dans la BC de l'agent, et celle-ci est enrichie par les déductions issues de l'analyse ainsi que par les décisions découlant de la planification.
- L'étape d'**exécution** correspond à un **flux descendant de traduction et de contrôle**. À partir de ses connaissances de haut niveau, l'agent va émettre des données qui instancient ces connaissances dans un format sémantiquement dégradé, mais cohérent dans le contexte de leur interprétation par les objets qui en sont la cible.

La notion de BC utilisée dans Kephart & Chess (2003) est à prendre au sens large : il s'agit d'un ensemble de règles et d'informations qui guident le comportement de l'agent. Nous proposons de représenter ces connaissances à l'aide des formalismes du web sémantique. C'est sous cette forme que les connaissances de l'agent sont exprimées dans SemIoTics : il dispose d'un ensemble de règle et d'individus qui instancient des ontologies. L'agent tire parti de l'ouverture des données, du partage de vocabulaires, ainsi que de la maturité des formalismes du W3C qui permettent de mettre en oeuvre du raisonnement.

3 Spécialisation de IoT-O pour représenter la connaissance nécessaire à la gestion intelligente d'un bâtiment

SemIoTics implémente une boucle MAPE-K qui s'appuie sur une BC exprimée dans les formalismes du W3C. Cette connaissance est décrite à l'aide de IoT-O et de modules qui l'étendent, et caractérise l'appartement connecté géré par SemIoTics.

3.1 IoT-O, une ontologie de core-domain pour l'IoT

IoT-O[3] est une ontologie décrivant des connaissances génériques à tous les domaines de l'IoT. Sa structuration initiale a été proposé dans Ben-Alaya *et al.* (2015), et une version enrichie, structurée par des patrons de conception a été présentée dans Seydoux *et al.* (2015). Sa structure modulaire régie par des patrons, ainsi que son import de nombreuses ressources existantes en font une ontologie de core-domain pour l'IoT-O plus facile à réutiliser et à étendre que d'autres ontologies de l'IoT. la figure 3 donne une vision de haut niveau de l'architecture de IoT-O.

Cette figure souligne la modularité de IoT-O, construite sur 5 axes : Observation, Action, Service, Énergie et Cycle de vie. Les noms des modules créés dans la construction de IoT-O sont en rouge surligné, et le nom de ceux qui ont demandé des modifications (alignement

[3] `http://www.irit.fr/recherches/MELODI/ontologies/IoT-O`

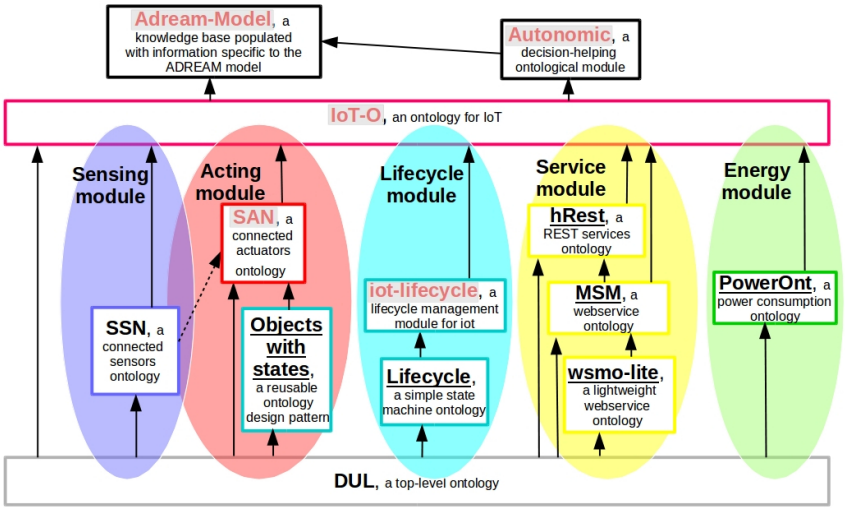

Figure 3: Architecture de haut niveau de IoT-O

avec DUL par exemple) est souligné. En tant qu'ontologie de core-domaine, IoT-O est conçue pour être spécialisée par des modules dédiés à une application particulière. Le reste de cette section est dédié à la description de Autonomic, module étendant IoT-O pour représenter la connaissance nécessaire à la mise en place d'un agent autonome, ainsi que de Adream-Model, module spécialisant lui aussi IoT-O pour représenter la connaissance propre à notre système.

3.2 Autonomic : un module spécialisant IoT-O pour mettre en place un agent autonome

Le module Autonomic[4] est une BC qui étend IoT-O pour permettre d'orienter les choix d'un agent autonome par rapport aux politiques de haut niveau fixées par l'utilisateur et aux données collectées par les objets. Ce module contient la connaissance nécessaire à la mise en place d'une boucle MAPE-K basée sur une BC, décrite dans la section 2.3.

Le module Autonomic définit la notion de *ConstrainedProperty*, une propriété de l'environnement (température, luminosité...) que l'utilisateur veut maintenir entre des bornes qu'il fixe, en spécialisant la propriété de IoT-O importée de SSN *ssn:Property*, représentant toute propriété dont le système a connaissance. Cette même classe *ssn:Property* est étendue par les classes *AboveMaxValueProperty* et *BelowMaxValueProperty*, qui servent à classifier les propriétés dont une observation permet de déterminer qu'elles ne respectent pas une contrainte.

Cette classification s'appuie sur une étape de raisonnement. Les observations sont liées aux objets qui les ont produites, ce qui permet à l'agent de prendre en compte leurs caractéristiques dans son traitement. Le module Autonomic spécialise la notion d'*iot-o:Impact*, pour indiquer si les actions qu'un objet peut effectuer sont de nature à faire augmenter (*PositiveImpact*) ou diminuer (*NegativeImpact*) la valeur de la propriété impactée. De plus, un *iot-o:Impact* peut être relatif (*RelativeImpact*) si le système peut en graduer l'effet (par exemple, fixer une valeur de consigne) ou absolu (*AbsoluteImpact*) dans le cas d'une interaction binaire de type on/off.

[4] `https://www.irit.fr/recherches/MELODI/ontologies/Autonomic`

3.3 Adream-Model : un module représentant la connaissance propre à ADREAM

3.3.1 Description de l'appartement connecté

Le système SemIoTics a été appliqué sur le bâtiment ADREAM[5], qui comporte un ensemble d'objets connectés, capteurs et actionneurs. Le cas d'utilisation présenté dans ce papier s'appuie sur un sous-ensemble de ceux-ci, organisés en deux systèmes séparés : la gestion de la température et la gestion de la luminosité. Pour gérer la température, l'appartement dispose d'un ventilateur, d'un chauffage électrique, ainsi que d'un capteur de température. La gestion de la luminosité repose sur un capteur de luminosité et une lampe.

Les objets sont connectés à une plateforme open-source, OM2M, implémentant le standard oneM2M, ce qui assure leur interopérabilité. La plateforme associe aux objets une interface REST qui permet d'y accéder de manière uniforme. Toutefois, dans sa version actuelle, oneM2M n'assure que l'interoperabilité architecturale. L'interopérabilité sémantique nécessite donc l'utilisation des technologies du web sémantique pour enrichir les données et les rendre interopérables et compréhensibles afin d'en garantir la consistance d'une application à l'autre, problème évoqué dans Corcho & García-Castro (2010).

La plateforme s'exécute sur une gateway supportant différents protocoles : Phidget, ethernet, sigfox, LoRa, enOcean. Pour établir un parallèle avec LMU-N, les objets connectés sont des noeuds de bas niveau : chacun ne maîtrise qu'un protocole de communication, et n'est dédié qu'à une tâche simple. La gateway sur laquelle s'exécute la plateforme est un noeud médian : elle permet la communication selon divers protocoles, et est suffisamment puissante pour exécuter le code de la plateforme, mais ne dispose que de capacités de traitement et de stockage limitées. La plateforme n'intègre pas de noeud de haut niveau, l'agent autonome est donc exécuté sur une machine tierce communiquant avec le noeud médian intégré à l'appartement.

3.3.2 Caractérisation des noeuds avec le module Adream-Model

Le module Adream-Model[6] spécialise IoT-O en proposant un ensemble de classes, de relations et d'individus permettant la description des objets connectés accessibles dans l'appartement, ainsi que des services qui en ouvrent l'accès. Un schéma illustrant l'instanciation du sous-système gérant la température est présenté dans la figure 4. L'élément au coeur du système est la propriété avec laquelle il interagit, ici *qudt:LuminousIntensity*. Cette propriété est mesurée par un capteur et impactée par deux actionneurs, l'ensemble de ces objets ayant des interfaces de *msm:Service* constituées d'*msm:Operations*. Celles-ci sont associées aux impacts que les objets peuvent avoir sur la propriété (augmentation ou diminution), et l'accès à ces opérations est déterminé par une machine à état modélisant le fonctionnement de l'objet. Adream-Model réutilise aussi une partie du vocabulaire du module Autonomic pour décrire les objets de Adream, pour que l'agent puisse prendre des décisions les concernant.

[5] `https://www.laas.fr/public/en/adream`

[6] `https://www.irit.fr/recherches/MELODI/ontologies/Adream-Model`

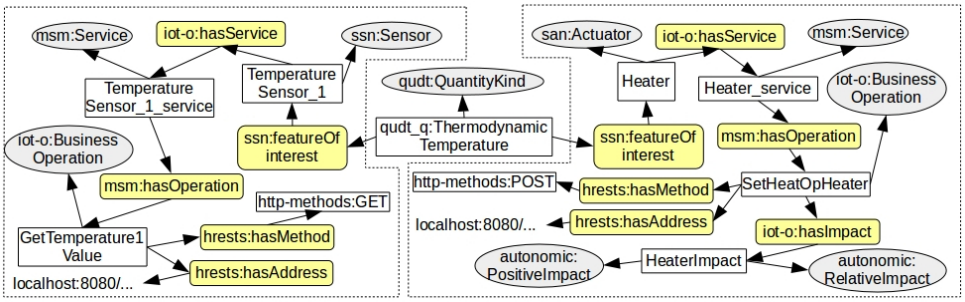

Figure 4: Extrait de la base de connaissance décrivant la gestion de température dans ADREAM

4 Mise en oeuvre : contrôle autonomique d'un appartement

4.1 SemIoTics, un système guidé par les décisions de l'utilisateur

Les décisions de l'agent autonome sont guidées par les contraintes imposées par l'utilisateur. Celles-ci sont vues comme des objectifs de haut niveau qu'il incombe au système d'atteindre. L'utilisateur exprime ses contraintes à travers une interface graphique, celles-ci sont traduites en connaissances et directement incorporées à la BC. Le coeur de SemIoTics reposant sur des traitements lourds (raisonnement, code Java) et demandant des capacités de stockage importantes pour la BC, le système s'exécute donc dans un noeud de haut niveau de LMU-N.

La BC de SemIoTics n'est pas exclusive à l'agent, elle expose une interface qui permet à un tiers de la consulter ou d'y ajouter des données. Cette configuration rend l'interaction avec les objets dynamiques : la gateway ou un opérateur humain peuvent déclarer des modifications dans les noeuds de bas niveau, rendant possible la découverte d'un nouveau noeud, la mise à jour de caractéristiques d'un noeud connu ou la suppression d'un noeud devenu inaccessible. Ce dynamisme permet la modification du comportement de l'agent en temps réel par des connaissances injectées par un opérateur. Cette BC est donc le point d'aboutissement des flux opérationnels d'exposition, et les flux de découverte par les noeuds de haut niveau s'appuient sur les informations qui y sont stockées. Pour les noeuds médians, la découverte de noeuds de bas niveau doit être assurée par un autre moyen, comme OM2M pour SemIoTics.

4.2 Monitoring : D'une donnée mesurée par un capteur à une connaissance

La collecte des données depuis les capteurs est assurée en premier lieu par la plateforme OM2M. Celle-ci offre un accès unifié aux données, et assure toujours un accès à la donnée la plus récente mesurée par le capteur. L'agent autonome de SemIoTics interroge la plateforme grâce aux descriptions des services associés aux capteurs contenues dans la BC (voir figure 4). À partir de cette description, la donnée brute retournée par la plateforme, contenant une date, une valeur et une unité est transformée en une *ssn:ObservationValue*. Cet enrichissement permet d'expliciter la connaissance implicite dont le système est en possession : le lien est établi avec le capteur à son origine, ainsi qu'avec la propriété sur laquelle porte l'observation. Ce processus est un flux opérationnel montant d'enrichissement décrit par LMU-N.

4.3 Analysis : Abstraction de l'observation en évènements porteurs de sens

L'étape d'analyse consiste à déterminer si les éléments observés à l'étape de monitoring demandent une réaction de la part de l'agent. Celle-ci peut donc entièrement se dérouler dans le noeud de haut niveau dans lequel SemIoTics s'exécute. Un premier filtrage est effectué sur les observations pour ne tenir compte que de la plus récentes portant sur chaque propriété contrainte par l'utilisateur. Ensuite, une étape de raisonnement permet de déterminer si une propriété est violée par l'observation en question. Ce raisonnement s'appuie sur une règle définie à l'aide du moteur de règle de Jena[7] embarqué dans SemIoTics. Ce raisonnement amène à la création de symptômes représentant une abstraction des observations. Si une action de l'agent est nécessaire, il entre en phase de planification.

4.4 Planification : Détermination des actions à effectuer par l'agent

Pendant l'étape de planification, l'agent autonome va déterminer sa réaction face au contexte analysé précédemment. Cette étape, comme l'étape d'analyse, est guidée par la connaissance stockée dans la BC de SemIoTics, et s'exécute donc dans un noeud de haut niveau. La propriété à corriger, ainsi que le type de correction à apporter (augmentation, diminution) sont établies dans l'étape d'analyse. L'agent autonome liste tous les objets agissant sur cette propriété, et sélectionne dans cette liste les objets pouvant avoir l'impact demandé.

À cette étape, si aucun objet n'est approprié pour répondre au besoin du système, la contrainte est déclarée insatisfiable (*autonomic:UnsatisfiableProperty*). Ce marquage est réévalué dès que l'ensemble d'objets connu par l'agent est modifié. Tant qu'une contrainte sur une propriété est estimée insatisfiable par le système, celui-ci ne tient pas compte des observations portant sur cette propriété dans la phase de monitoring. Cette mesure permet d'éviter de relancer des raisonnements inutiles tant qu'une action significative n'a pas été entreprise.

Pour chaque objet, l'agent va lister les services qu'il offre et effectuer un nouveau tri par rapport à leur description dans IoT-O. Par exemple, deux types de services sont identifiés : les services liés à la fonctionnalité spécifique de l'objet, *iot-o:BusinessService*, définis par opposition au service de management liés à l'entretien de l'objet, comme la consultation du niveau de batterie, la mise à jour du firmware, etc. De même, pour les services sélectionnés, l'agent découvre les opérations qu'ils exposent, leurs effets, et leur utilité potentielle concernant l'état de l'objet. C'est là que la modélisation des actionneurs sous forme de machine à état à l'aide de IoT-Lifecycle[8] intervient : les actions sont associés à des états, et déclenchent des transitions. Ainsi, pour augmenter la luminosité, l'agent ne considérera pas qu'allumer une lampe déjà en marche est une action valide. Après ces différentes étapes de filtrage, l'agent sélectionne une opération parmi celles qui ont été retenues, et ajoute dans la base de données une *san:Actuation* représentant l'appel à celle-ci. L'étape d'exécution s'appuie ensuite sur cette *san:Actuation*.

Dans notre cas, l'ensemble des filtrages effectués par l'agent est guidé par la nécessité. Dans un cas plus général où plusieurs objets pourraient convenir, cette sélection pourrait être enrichie par des expressions de politiques guidées par la qualité de service Chaocan Xiang *et al.* (2015), la consommation énergétique, ou toute autre politique de haut niveau exprimée comme dans Perera *et al.* (2014).

---

[7] `https://jena.apache.org/documentation/inference/index.html`

[8] `https://www.irit.fr/recherches/MELODI/ontologies/IoT-Lifecycle`

4.5 Execution : De l'abstraction à la commande

L'étape d'exécution met en place des décisions actées par l'agent pendant la planification, c'est donc un flux descendant de contrôle. Ce flux de contrôle est doublé d'un flux de traduction : le modèle de données utilisé dans le noeud de haut niveau prenant la décision peut ne pas être compris par le noeud ciblé. La traduction peut être vue comme une "dégradation sémantique" de la connaissance vers une donnée qui assure la consistance du sens de la donné dans le contexte où elle est utilisée par rapport au sens de la connaissance dont elle dérive. La traduction a lieu en deux étapes : le noeud de haut niveau transmet d'abord au noeud médian les connaissances nécessaire au contrôle. Le noeud médian construit ensuite une requête adaptée au noeud de bas niveau de destination, en tenant compte du format de données ou du canal de communication qu'il supporte. La commande est alors envoyée au noeud de bas niveau, qui met à jour son état en conséquence et permet au système d'agir sur le monde physique. La modification de la propriété impactée est mesurée par des capteurs, qui injecteront ces mesures dans le système à la prochaine étape de monitoring, amenant à un contrôle de l'environnement en boucle fermée.

5 Conclusion et travaux futurs

La contribution principale de ce papier est la proposition de SemIoTics, un système autonome de gestion d'un ensemble d'objets connectés s'appuyant sur une BC. Cette proposition est accompagnée d'une structuration architecturale générique, LMU-N, qui permet de caractériser les contributions du web sémantique à l'IoT. SemIoTics vise à surmonter certains verrous de l'IoT : les contraintes matérielles des objets, le manque d'interopérabilité, et la complexité du système dûe à l'hétérogénéité et au passage à l'échelle. La notion d'objet contraint est prise en compte dans LMU-N, ce qui permet sa prise en compte dans des flux opérationnels adaptés. SemIoTics s'appuie sur une plateforme standard, OM2M, pour apporter l'interopérabilité architecturale, et sur les principes du web sémantique pour construire l'interopérabilité sémantique. Enfin, la complexité du système est considérée par l'utilisation de la boucle de gestion autonomique, combinée à une BC : les données y sont enrichies à la fois par des connaissances spécifiques au domaine d'application, et par des connaissances sur le système lui-même. Ces dernières sont décrites par des modules spécialisant IoT-O, une ontologie de core-domaine pour l'IoT.

Dans le futur, un open data permettra de consulter les données issues du bâtiment instrumenté du LAAS, ADREAM. Cet open data comprendra une interface permettant d'accéder à certaines fonctionnalités de SemIoTics, et d'un endpoint permettant une exploration des bases de connaissances de l'agent autonome. Les sources de données de SemIoTics s'étendront des objets de l'appartement (quelques dizaines d'objets) aux capteurs de ADREAM tout entier (plus de 2400 capteurs). Cette évolution amènera des problématiques de passage à l'échelle demandant une gestion des données en flux, mais aussi de qualité des données (capteurs défectueux, incohérences de la base de données) ou d'abstraction qui reposent sur de l'aggrégation de données issues de capteurs redondants ou complémentaires.

References

BARNAGHI P., WANG W., HENSON C. & TAYLOR K. (2012). Semantics for the Internet of Thingss: early progress and back to the future. In *International Journal on Semantic Web and Information Systems*, volume 8, p. 1–21.

BEN-ALAYA M., MEDJIAH S., MONTEIL T. & DRIRA K. (2015). Toward semantic interoperability in oneM2M architecture. *IEEE Communications Magazine*, **53**(12), 35–41.

CHAOCAN XIANG, PANLONG YANG, XUANGOU WU, HONG HE & SHUCHENG XIAO (2015). QoS-based service selection with lightweight description for large-scale service-oriented internet of things. In *Tsinghua Science and Technology*, volume 20, p. 336–347: Tsinghua University Press (TUP).

CHATZIGIANNAKIS I., HASEMANN H., KARNSTEDT M., KLEINE O., KRÖLLER A., LEGGIERI M., PFISTERER D., RÖMER K. & TRUONG C. (2012). True Self-Configuration for the IoT. In *3rd International Conference on the Internet of Things (IOT)*.

CORCHO O. & GARCÍA-CASTRO R. (2010). Five challenges for the Semantic Sensor Web. *Semantic Web*, **1**(1), 121–125.

DE S., BARNAGHI P., BAUER M. & MEISSNER S. (2011). Service modelling for the Internet of Things. In *2011 Federated Conference on Computer Science and Information Systems (FedCSIS)*, p. 949–955.

DESAI P., SHETH A. & ANANTHARAM P. (2015). Semantic Gateway as a Service architecture for IoT Interoperability. In *Kno.e.sis Publications*.

FOTEINOS V., KELAIDONIS D., POULIOS G., VLACHEAS P., STAVROULAKI V. & DEMESTICHAS P. (2013). Cognitive management for the internet of things: A framework for enabling autonomous applications. *IEEE Vehicular Technology Magazine*, **8**(4), 90–99.

GUBBI J., BUYYA R., MARUSIC S. & PALANISWAMI M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, **29**(7), 1645.

GYRARD A., SERRANO M. & ATEMEZING G. A. (2015). Semantic web methodologies, best practices and ontology engineering applied to Internet of Things. In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, p. 412–417: IEEE.

KEPHART J. & CHESS D. (2003). The vision of autonomic computing. *Computer*, **36**(1), 41–50.

LE-PHUOC D. & HAUSWIRTH M. (2009). Linked open data in sensor data mashups. In *Proceedings of the 2nd International Workshop on Semantic Sensor Networks (SSN09)*, volume 522, p. 1–16.

LIU J., LI Y., CHEN M., DONG W. & JIN D. (2015). Software-defined internet of things for smart urban sensing. *IEEE Communications Magazine*, **53**(9), 55–63.

MRISSA M., MEDINI L., JAMONT J.-P., LE SOMMER N. & LAPLACE J. (2015). An Avatar Architecture for the Web of Things. *Internet Computing, IEEE*, **19**(2), 30–38.

NIKOLI S., PENCA V. & KONJOVI Z. (2011). Semantic Web Based Architecture for Managing Hardware Heterogeneity in Wireless Sensor Network. In *International Journal of Computer Science and Applications*, volume 8, p. 38–58.

PERERA C., ZASLAVSKY A., LIU C. H., COMPTON M., CHRISTEN P. & GEORGAKOPOULOS D. (2014). Sensor search techniques for sensing as a service architecture for the internet of things. *IEEE Sensors Journal*, **14**(2), 406–420.

POSLAD S., MIDDLETON S. E., CHAVES F., TAO R., NECMIOGLU O. & BUGEL U. (2015). A Semantic IoT Early Warning System for Natural Environment Crisis Management. *IEEE Transactions on Emerging Topics in Computing*, **3**(2), 246–257.

SEYDOUX N., ALAYA M. B., HERNANDEZ N., MONTEIL T. & HAEMMERLÉ O. (2015). Sémantique et Internet des objets : d'un état de l'art à une ontologie modulaire. In *26es Journées francophones d'Ingénierie des Connaissances*.

SHETH A., HENSON C. & SAHOO S. S. (2008). Semantic Sensor Web. In *IEEE Internet Computing*, volume 12, p. 78–83.

VLACHEAS P., GIAFFREDA R., STAVROULAKI V., KELAIDONIS D., FOTEINOS V., POULIOS G., DEMESTICHAS P., SOMOV A., BISWAS A. & MOESSNER K. (2013). Enabling smart cities through a cognitive management framework for the internet of things. *IEEE Communications Magazine*, **51**(6).

ZANELLA A., BUI N., CASTELLANI A., VANGELISTA L. & ZORZI M. (2014). Internet of Things for Smart Cities. *IEEE Internet of Things Journal*, **1**(1), 22–32.