REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider – BISKRA
Faculté des Sciences Exactes, des Sciences de la Nature et de la Vie
Département d'informatique

N° d'ordre : GLSD06/M2/2021

Mémoire

Présenté pour obtenir le diplôme de master académique en

Informatique

Parcours : **Génie logiciel et Systèmes Distribués (GLSD)**

Une composition des services IoT basée ontologie

Par :

Abdelli Meriem

Soutenu le …/../…. devant le jury composé de :

Nom Prénom	Grade	Président
Bendahmene Asma	MMA	Rapporteur
Nom Prénom	Grade	Examinateur

Année universitaire 2020-2021

Remerciements

Je tiens à remercier vivement mon encadreur

Bendahmene Asmaa

pour avoir encadré et conseillé dans la réalisation de ce travail.

Je remercie ***Abdelli Belguecem*** *qui m'a aidé pendant la préparation*

de ce travail malgré ses occupations.

Je remercie aussi les membres du jury qui ont

accepté de juger ce modeste travail.

Mes remerciements vont également à ma famille, mes amis et mes proches qui ont beaucoup soutenu pour réaliser ce mémoire.

Dédicace

Je dédié ce travail

A mes très chers parents qui m'ont aidé durant toute ma vie que dieu les protègent,

A mes chers frères et ma sœur,

A mes proches, à tous mes amis

et à ma grande famille.

Résumé

L'Internet des objets (IoT) se compose d'appareils physiques et de composants logiciels interconnectés. Ces objets ou objets connectés échangent des informations afin de fournir un service à l'utilisateur final. Pour remplir cet objectif, de telles applications doivent être conçues en composant des objets existants. Cependant, il s'agit d'une tâche très difficile principalement en raison de l'hétérogénéité et de la diversité des objets disponibles. Nous proposons un framework basé sur des ontologies pour faciliter la composition automatique d'applications appropriées. Le système compose automatiquement les services appropriés en fonction des équipements disponibles. Pendant ce temps, il ajuste dynamiquement les paramètres de l'environnement pour répondre aux besoins du client et englober la ressource disponible.

Abstruct

Internet of Things (IoT) consists of interconnected physical devices and software components. These connected things or objects exchange information in order to provide an end-user service. To fulfil this objective, such applications have to be designed by composing existing objects. However, this is a very difficult task mostly due to the heterogeneity and diversity of available objects. We propose an ontology-based framework to facilitate the automatic composition of appropriate applications. The system composes appropriate services depending upon the available equipments automatically. Meanwhile, it dynamically adjusts the environment parameters to match the customer needs and to encompass the available.

ملخص

يتكون إنترنت الأشياء (**IoT**) من أجهزة مادية ومكونات برامج مترابطة. تتبادل هذه الأشياء أو الكائنات المتصلة المعلومات من أجل توفير خدمة للمستخدم النهائي. لتحقيق هذا الهدف، يجب تصميم هذه التطبيقات عن طريق تكوين كائنات موجودة. ومع ذلك، فهذه مهمة صعبة للغاية بسبب عدم تجانس وتنوع الكائنات المتاحة. نقترح إطار عمل قائم على الأنطولوجيا لتسهيل التكوين التلقائي للتطبيقات المناسبة. يؤلف النظام الخدمات المناسبة حسب المعدات المتوفرة بشكل آلي. وفي الوقت نفسه، يقوم بضبط معلمات البيئة ديناميكيًا لمطابقة احتياجات العملاء ولتضمين المورد المتاح.

Table des matières

Table des matières 7
Liste des tables 9
Liste des figures 10
INTRODUCTION GENERALE 11
Contexte 12
Problématique 12
Objectives et contributions 12
Organisation du mémoire 12
Chapitre 1: Etat de l'art 14
Introduction 15
1. Internet des Objets 15
1.1 Définition de l'internet des objets 15
1.2 Capteurs et Actionneurs 16
1.3 Fonctionnement d'internet des objets 16
1.4 Etapes et technologies dans l'écosystème de L'IdO : 17
1.5 Domaine d'application 18
2. Ontologie 19
2.1 Définition de l'ontologie 19
2.2 Composants d'une ontologie 20
2.3 Les types d'ontologies 21
2.4 Rôle d'une ontologie 21
2.5 Langages de description d'une ontologie 22
3. composition des services IoT 23
3.1 Définition de Service 23
3.2 Définition de la composition des services : 23
3.3 Approche existantes pour la composition des services dans l'IoT 24
Conclusion 28
Chapitre 2: Composition des services IoT basée ontologie 29
Introduction 30
1. Présentation de l'environnement domotique 30
2. Architecture du système 31
3. Scenario proposé 31
4. L'environnement domotique proposé 32
Conclusion 47

Chapitre 3: Implémentation.....48
Introduction.....49
1. langage et les outils de développements :.....49
2. L'implémentation.....50
2.1 La création les classe d'ontologie dans Protégé :.....50
2.2 Définition des Propriétés de l'ontologie développée :.....51
2.3 Ajout d'individus aux classes :.....56
3.4 L'ontologie sous forme d'un graph.....58
Conclusion.....58
Conclusion Général.....59
Biographie.....61

Liste des tables

Table 1 comparision entre les approches de composition de service IoT. 27
Table 2 : attributs de la classe PhysicalObject. 39
Table 3 : attributs de la classe Actor. 39
Table 4 : Associations de la classe Actor. 39
Table 5 : attributs de la classe Platform. 39
Table 6 : Associations de la classe Platform. 40
Table 7 : attribut de la classe Tv. 40
Table 8 : attribut de la classe MachCoffee. 40
Table 9 : attributs de la classe Lamp. 40
Table 10 : attribut de la classe SmartPhone. 40
Table 11 : attributs de la classe PC. 41
Table 12 :attribut de la classe Tablet. 41
Table 13 :attributs de la classe room. 41
Table 14 : attribut de la classe Sensor. 41
Table 15 : associations de la classe Sensor. 42
Table 16 : attribut de la classe HumiditySensor. 42
Table 17 : attributs de la classe temperatureSensor. 42
Table 18 : attributs de la classe LightSensor. 42
Table 19 : attribut de la classe MotionSensor. 42
Table 20 : attribut de la classe WaterSensor. 43
Table 21 : attributs de la classe IpCamera. 43
Table 22 : attribut de la classe Cammand. 43
Table 23 : associations de la classe Cammand. 43
Table 24 : attribut de la classe SystemCmd. 43
Table 25 : attributs de la classe PlatformCmd. 44
Table 26 : attributs de la classe System. 44
Table 27 : associations de la classe System. 44
Table 28 : attribut de la classe Location. 44
Table 29 : attributs de la classe Observation. 45
Table 30 : associations de la classe Observation. 45
Table 31 : attributs de la classe Unit. 45
Table 32 : attribut de la classe Property. 45
Table 33 : attribut de la classe Agenda. 46
Table 34 : attributs de la classe service. 46
Table 35 : attribut de la classe Activity. 46

Liste des figures

Figure 1 Les étapes et les technologies pour la mise en place de l'IdO **[5].** 18
Figure 2 Une ontologie sous forme de graphe **[8].** 20
Figure 4 : Architecture Système. 31
Figure 5 Figure 1 Diagramme de Classe de l'environnement domotique. 37
Figure 6 creation des classes dans protégé. 50
Figure 7 : déclaration de la subClass System. 51
Figure 8 : déclaration de la propiérité d'objet OnPlatform. 51
Figure 9 la creation de la propriété "onPlatform" 52
Figure 10 : création de la propriétés d'objet "hasActivity". 53
Figure 11 la création de la propriété d'objet "commandToSystem". 54
Figure 12: declaration de la propriété de type ObservatioValue. 54
Figure 13 : déclaration de la propriété de type ObservationValue. 55
Figure 14 : la déclaration de la propriété de type hasSymbol. 55
Figure 15 : la création de la propriété de type hasSymbol. 56
Figure 16 Ajout d'individus au classe "System". 57
Figure 17 Ajout d'individus au classe "Platform 57
Figure 18 l'ontologie de system sous forme d'un graphe. 58

INTRODUCTION GENERALE

Contexte

L'Internet des objets (IoT) est un réseau d'appareils physiques et de composants logiciels connectés entre eux afin d'échanger des informations et de fournir un service IoT. Les applications IoT sont censées renforcer les objets interconnectés afin de construire des services puissants et à valeur ajoutée.

Généralement, les données générées par les systèmes IoT sont massives et hétérogènes. Par conséquent, la composition de ces services et l'utilisation efficace de ces données devient de plus en plus complexe. Alors, il est nécessaire d'avoir un modèle commun pour spécifier tous les objets et la manière dont ils interagissent ensemble et les transmettre à des connaissances de haut niveau.

En outre, dans ce mémoire, les ontologies seront utilisées pour résoudre les problèmes liés à la composition des services IoT.

Problématique

Le nombre croissant d'objets connectés et la diversité leur utilisation ont créé un défi à savoir :

- Hétérogénéité des données : L'hétérogénéité des données (différents sources et formats) entraîne un manque d'interopérabilité entre les applications IoT.

Objectifs et contributions

L'objective essentielle de ce mémoire étant de proposer une approche d'annotation qui permet de composer des services IoT. Notre approche permet à :

- Supporter le traitement de données hétérogènes en utilisant les ontologies.
- Composer des services pour remplir les fonctions demandées en fonction des appareils disponibles.
- Supporter la réutilisation et le partage des connaissances entre les différentes applications IoT.

Organisation du mémoire

Après l'introduction générale, ce mémoire contient trois chapitres et une conclusion générale.

Chapitre 1 Etat de l'art :

Ce chapitre est consacré aux concepts de base : Internet des Objets, Ontologie, composition des services IoT, Nous terminons ce chapitre par des approches existantes dans la littérature.

Chapitre 2 Composition des services IoT basé ontologie :

Dans ce chapitre, on a présenté le cadre conceptuel, Nous s'intéresse à la description des étapes suivies dans le développement de notre ontologie et cela après une présentation du scenario, l'architecture générale, et les différents concepts et composants de l'environnement proposé, la conceptualisation avec le langage UML et la formalisation de l'ontologie avec le langage OWL.

Chapitre 3 Implementation :

Le dernier chapitre, concerne la mise en œuvre de notre approche, en commençant par la description des outils et l'environnement de développement. Ensuite, nous présentons quelques interfaces qui montrent les résultats obtenus d'après l'implémentation de notre modèle.

Nous clôturons ce mémoire par une conclusion et quelques perspectives.

Chapitre 1: Etat de l'art

1 ETAT DE L'ART

Introduction

L'Internet des objets est une nouvelle vague de l'Internet, est en réalité une partie naissante de l'Internet du futur, appelé IoT, qui vise à interconnecter les gens, les données et tous les objets, de telle sorte qu'il y ait une fusion entre le monde réel (physique) et le monde numérique (virtuel), les objets du monde physique vont être incorporés dans le monde virtuel de l'Internet.

L'IoT se compose d'appareils physiques interconnectés et de composants logiciels. Ces objets ou objets connectés échangent des informations afin de fournir un service à l'utilisateur final. Pour remplir cet objectif, de telles applications doivent être conçues en composant des objets existants. Cependant, il s'agit d'une tâche très difficile principalement en raison de l'hétérogénéité et de la diversité des objets disponibles. L'outil IoT Composer a été développé pour soutenir le développement d'applications IoT en fournissant d'abord un modèle comportemental pour les objets et leur composition.

L'ontologie a reçu une attention considérable ces dernières années avec l'émergence du web sémantique. C'est une spécification explicite de conceptualisations qui organise l'information sémantique comme la base de connaissances des domaines d'application spécifiques. De plus, son utilisation est étendue à l'informatique orientée service (SOC) pour faciliter une découverte et une composition de services plus intelligentes. Les langages d'ontologie actuels, tels que RDF et OWL, sont souvent représentés en XML et peuvent être traités par des machines. Ils prennent en charge la spécification de concepts, de relations et de mécanismes de classification et de raisonnement associés pour les données.

Dans ce chapitre est consacré à donner un survol des généralités sur l'Internet des objets, les ontologies et la composition des services IoT, on conclut le chapitre par des travaux connexes sur la composition des services IoT.

1. Internet des Objets

1.1 Définition de l'internet des objets

L'Internet des objets (IdO) est une « infrastructure mondiale pour la société de l'information, qui permet de disposer de services évolués en interconnectant des objets (physiques ou virtuels) grâce aux technologies de l'information et de la communication interopérables existantes ou en évolution ». En réalité, la définition de ce qu'est l'Internet des

objets n'est pas figée. Elle recoupe des dimensions d'ordres conceptuel et technique. D'un point de vue conceptuel, l'Internet des objets caractérise des objets physiques connectés ayant leur propre identité numérique et capables de communiquer les uns avec les autres. Ce réseau crée en quelque sorte une passerelle entre le monde physique et le monde virtuel. **[1]**

Les capteurs et actionneurs forment les éléments clés de l'internet des objets. Ils suivent l'état de leur environnement, obtiennent des informations sur la température, le mouvement, la position, etc. Ils constituent un réseau généralement composé d'un nombre potentiellement élevé de nœuds. Ces capteurs doivent faire face à de nombreux problèmes de communication comme la sécurité et confidentialité, leur mobilité, leur courte portée, leur fiabilité, leur robustesse, leur évolutivité et leurs ressources (énergétiques, capacité de stockage et de traitement limitées, bande passante, etc.). **[2]**

1.2 Capteurs et Actionneurs

Les capteurs sont les yeux et les oreilles de l'objet IoT pour voir son environnement, et les actionneurs sont les jambes et les mains qui remplissent ses fonctions **[3]** :

- **Capteur :** Un capteur est un appareil qui convertit un paramètre physique en une sortie électrique. Un capteur est un type de transducteur. Il disposant de capacités de mesures, voire d'actions, sur leur environnement La température, le taux d'humidité, la luminosité ambiante, la détection de présences ou de mouvements via un accéléromètre, présence de gaz, de polluants ou encore la géolocalisation font partie des informations les plus couramment collectées sur ce type de matériel.

- **Actionneur :** Un actionneur est un dispositif qui convertit un signal électrique en sortie physique, c'est-à-dire un mouvement. Un actionneur peut être contrôlé par la tension ou le courant électrique, la pression pneumatique ou hydraulique, ou même la puissance humaine. Dans les systèmes embarqués, les actionneurs sont principalement contrôlés par l'électricité. Lorsque le signal de commande est reçu, l'actionneur convertit l'énergie électrique en mouvement mécanique.

1.3 Fonctionnement d'internet des objets

L'Internet des objets (IdO) permet l'interconnexion des différents objets intelligents via l'Internet. Ainsi, pour son fonctionnement, plusieurs systèmes technologiques sont

nécessaires nous mettons l'accent seulement sur quelques-unes qui sont les technologies clés de l'IdO. Ces technologies sont les suivantes : RFID, WSN et M2M, et sont définies ci-dessous **[4] [19]** :

- **RFID** (Radio Frequency Identification) : est une technologie sans fil qui est utilisée pour l'identification des Objets, elle englobe toutes les technologies qui utilisent des ondes radio pour identifier automatiquement des Objets ou des personnes. C'est une technologie qui permet de mémoriser et de récupérer des informations à distance grâce à une étiquette qui émet des ondes radio. Il s'agit d'une méthode utilisée pour transférer les données des étiquettes à des Objets, ou pour identifier ces Objets à distance. L'étiquette contient des informations stockées électroniquement pouvant être lues à distance **[19].**
- **WSN** (Wireless Sensor Network) : c'est un ensemble de nœuds qui communiquent sans fil et qui sont organisés en un réseau coopératif. Chaque nœud possède une capacité de traitement et peut contenir différents types de mémoires, 18 un émetteur-récepteur RF et une source d'alimentation, comme il peut aussi tenir compte des divers capteurs et des actionneurs. Comme son nom l'indique, le WSN constitue alors un réseau de capteurs sans fil qui peut être une technologie nécessaire au fonctionnement de l'IdO **[4]**.

- **M2M** (Machine to Machine) : c'est « l'association des technologies de l'information et de la communication avec des objets intelligents dans le but de donner à ces derniers les moyens d'interagir sans intervention humaine avec le système d'information d'une organisation ou d'une entreprise » **[4]**.

1.4 Etapes et technologies dans l'écosystème de L'IdO :

Les objets connectés sont au cœur de l'IdO, mais il est important de pouvoir connecter l'ensemble de ces objets, les faire échanger des informations et interagir, au sein d'un même environnement. La mise en place de l'IdO passe par les étapes suivantes : l'*identification,* l'*installation de capteurs,* la *connexion des objets* entre eux, l'intégration et la *connexion à un réseau*. Le tableau suivant présente les étapes et les protocoles éventuels [5] :

Identifier	Capter	Connecter	Intégrer	Mettre en réseaux
Rendre possible l'identification de chaque élément connecté.	Mise en place de dispositifs nous rapprochant du monde réel. Les fonctions de base des objets (le capteur de température pour le thermomètre par exemple).	Établir une connexion entre tous les objets afin qu'ils puissent dialoguer et s'échanger des données.	Disposer d'un moyen de communication rattachant les objets au monde virtuel.	Relier les objets et leurs données au monde informatique via un réseau (Internet par exemple).
IPv4, IPv6, 6LoWPAN	MEMS, RF MEMS, NEMS	SigFox, LoRa	RFID, NFC, Bluetooth, Bluetooth LE, ZigBee, WiFi, réseaux cellulaires	CoAP, MQTT, AllJoyn, REST HTTP

Figure 1 *Les étapes et les technologies pour la mise en place de l'IdO [5].*

1.5 Domaine d'application

Plusieurs domaines d'application sont touchés par l'IdO, Parmi ces principaux domaines nous citons **[6] [19] [20]** :

- **Les villes intelligentes (Smart Cities)** : l'IoT permettra une meilleure gestion des réseaux divers qui alimentent nos villes (eaux, électricité, gaz, etc.) en permettant un contrôle continu en temps réel et précis. Des capteurs peuvent être utilisés pour l'économie de l'eau et pour améliorer la gestion des parkings et du trafic urbain et diminuer les embouteillages et les émissions en CO2 **[6].**
- **La santé (Smart Health)** : dans le domaine de la santé, l'"IoT permettra le déploiement de réseaux personnels pour le contrôle et le suivi des signes cliniques, notamment pour des personnes âgées, les objets connectés permettent de suivre la tension, le rythme cardiaque, la qualité de respiration ou encore la masse graisseuse. Ceci permettra ainsi de faciliter la télésurveillance des patients à domiciles, et apporter des solutions pour l'autonomie des personnes à mobilité réduite **[6].**
- **Le Transport :** Le transport intelligent est confronté à trois conceptions principales ils sont l'analyse des transports, le contrôle des véhicules connectées. L'analyse de transport représente l'analyse de la prédiction de la demande et de détection anomalie. Le routage des véhicules et le contrôle de la vitesse en plus de la gestion

du trafic sont tous connu comme le contrôle du transport qu'ils ont réellement étroitement lié au 19 véhicules connecté (par la communication V2X) , et globalement régie par la diffusion multi-technologie **[20].**

- **Le Smart Grid** : L'un des domaines d'application de l'IoT est le secteur de la distribution d'énergie intelligente, dit « Smart Grid ». En France, ERDF est très actif dans le développement de ce domaine, où un besoin clair en récupération d'information à différents points du réseau électrique est devenue nécessaire pour une meilleure intégration des différentes sources d'énergies et une meilleure gestion de la distribution jusqu'aux utilisateurs finaux **[20].**
- **L'industrie** : La technologie IoT permettra un suivi total des produits, de la chaîne de production, jusqu'à la chaîne logistique et de distribution en supervisant les conditions d'approvisionnement. Cette traçabilité de bout en bout permet aux usines d'améliorer l'efficacité de ses opérations, d'optimiser la production et d'améliorer la sécurité des employés **[6].**
- **Le bien-être et le confort** : La domotique ou la maison intelligente est un classique. Imaginez un instant que votre thermostat soit capable de se mettre en marche tout seul en fonction de l'emplacement de votre voiture vous permettant de vous réchauffer une fois rentré à la maison. Aussi, imaginez que votre réfrigérateur vous informe lorsque vous aurez besoin d'acheter du lait ou qu'il soit capable de créer une liste d'achats personnalisée en fonction de vos articles les plus achetés **[6].**
- **Sécurité et surveillance** : La surveillance de sécurité est devenue une nécessité pour les bâtiments d'entreprise, centres commerciaux, usine, parkings et de nombreux autres lieux publics. Tout en préservant la vie privée des utilisateurs, des capteurs ambiants peuvent être utilisés pour surveiller la présence de produits chimiques dangereux. Des capteurs de surveillance du comportement des personnes peuvent ˆêtre utilisés pour évaluer la présence de personnes qui agissent de manière suspecte **[19].**

2. Ontologie

2.1 Définition de l'ontologie

Une ontologie est une spécification explicite et formelle d'une conceptualisation faisant l'objet d'un consensus. La conceptualisation fait référence à un modèle (au sens ensemble

structuré) de concepts ce qui signifie que tous les types primitifs, les concepts et les contraintes qui sont utilisés dans les spécifications de l'ontologie sont explicitement définis.

Une extension de cette définition considère qu'une ontologie est un système formel dont l'objectif est de représenter les connaissances d'un domaine spécifique au moyen d'éléments de base, les concepts, définis et organisés les uns par rapport aux autres. Elle est aussi considérée comme une collection de descriptions explicites, complète et consensuelles de concepts, en mettant l'accent sur la précision de leur sens. **[7]**

2.2 Composants d'une ontologie

Une ontologie est composée de : concepts, propriétés, relations, axiomes et des instances **[8]**. La figure suivant représente une ontologie sous forme d'un graphe :

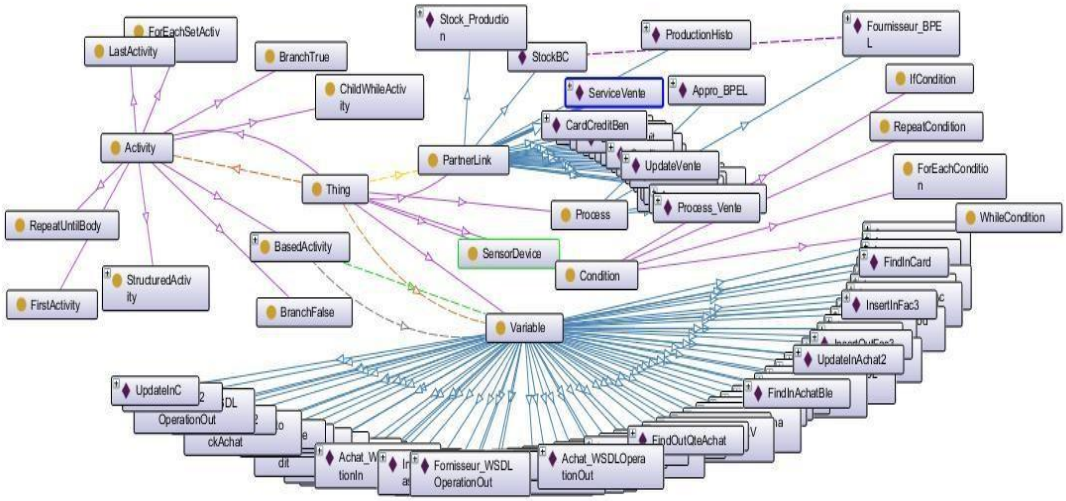

Figure 2** Une ontologie sous forme de graphe **[8].

- **Le concept (classe)** ● **:** Il peut représenter un ensemble d'objet, une idée, un principe ou une notion abstraite. Il est caractérisé par une extension et une intention.
- **Les propriétés :** sont des caractéristiques évaluées attachées aux concepts.
  - **Data Properties** ▭
  - **Object Properties** ▭
- **Les relations sémantiques :** sont des relations entre les concepts d'un domaine. Elles sont définies comme tout sous- ensemble d'un produit cartésien de n ensembles, ces dernières permettent de structurer hiérarchiquement les concepts d'une ontologie par

exemple : Identité, synonymie (symétrique), sort de (unidirectionnelle), l'équivalence, homonymie.

- **Les axiomes :** constituent des assertions, acceptées comme vraies, à propos des abstractions du domaine traduites par l'ontologie.
- **Les instances (Individuals) ♦ :** Elles constituent la définition extensionnelle d'une ontologie.

2.3 Les types d'ontologies

Les types d'ontologie sont **[16]** :

2.3.1 Les ontologies de haut niveau (top-level ontologies) : Elles décrivent des concepts très généraux comme l'espace, le temps, la matière, les objets, les événements, les actions, etc.

2.3.2 Les ontologies de domaine (domaine ontologies) et les ontologies de tâche (task ontologies) Elles décrivent le vocabulaire lié à un domaine générique (comme la médecine, ou les automobiles) ou une tâche ou une activité générique (comme le diagnostic ou la vente).

2.3.3 Les ontologies de tâches : Ce type d'ontologies décrit le vocabulaire concernant une tâche générique (ex. : enseigner, diagnostiquer…), notamment en spécialisant les concepts d'une ontologie de haut niveau. Certains auteurs emploient le nom « ontologie du domaine de la tâche » pour faire référence à ce type d'ontologie.

2.3.4 Les ontologies d'application (application ontologies) : Elles décrivent des concepts dépendant à la fois d'un domaine et d'une tâche particuliers dans ce domaine. Ces concepts correspondent souvent aux rôles joués par des entités.

2.3.5 Les ontologies de représentation des connaissances (méta ontologie) : Elles décrivent les concepts utilisés par les langages de représentation des ontologies.

2.4 Rôle d'une ontologie

Une ontologie permet de **[9]** :

- Acquérir et représenter les connaissances
- Rechercher et faire l'extraction des connaissances c'est-à-dire inférer la connaissance qui est pertinente face à la requête de l'utilisateur
- Partager et intégrer les connaissances
- Gérer des connaissances : simplification du dialogue homme-machine.

2.5 Langages de description d'une ontologie

Une ontologie peut être représentée par plusieurs langages, dans ce qui suit, nous citons les plus fréquents **[10]** :

- **RDF** : acronyme de Resource Description Framework, développé et présenté par le W3C au début de 1999 pour l'intégration des métadonnées et montre également comment décrire les ressources disponibles sur Internet telles que le contenu des sites web. Le RDF soutient directement l'intégration et l'analyse parmi les applications hétérogènes de l'IDO (Internet Des Objets) qui échangent des données lisibles par la machine sur le Web.
- **RDF schema** : c'est une extension sémantique du vocabulaire RDF de modélisation des données. Il illustre les descriptions connexes et établit les liens significatifs entre les ressources.
- **OWL** : "Web Ontology Language". Il est défini par le W3C, Le langage OWL est basé sur la recherche effectuée dans le domaine de la logique de description. OWL permet de décrire des ontologies, c'est-à-dire qu'il permet de définir des terminologies pour décrire des domaines concrets. Une terminologie se constitue de concepts et de propriétés (aussi appelés rôles en logiques de description). Un domaine se compose d'instance de concepts.

OWL est divisé en trois sous-langages, OWL Lite, OWL DL, et OWL Full. Ces langages visent différents groupes d'utilisateurs **[11]** :

- **OWL Lite** est le sous langage d'OWL le plus simple. Il est destiné aux utilisateurs qui ont besoin d'une hiérarchie de concepts simple.
- **OWL DL** est plus complexe qu'OWL Lite, permettant une expressivité bien plus importante.
- **OWL Full** est la version la plus complexe d'OWL, mais également celle qui permet le plus haut niveau d'expressivité. OWL Full est destiné aux situations où il est plus important d'avoir un haut niveau de capacité de description, quitte à ne pas pouvoir garantir la complétude et la décidabilité des calculs liés à l'ontologie.

3. composition des services IoT

3.1 Définition de Service

Un service est un mécanisme capable de fournir une ou plusieurs fonctionnalités. Un service peut être utilisé conformément aux restrictions et règles définies par le fournisseur et via une interface. Les services pour environnements intelligents sont généralement considérés comme des extensions de services Web existants (par exemple, messagerie, services de localisation, gestion d'appareils, applications multimédias...) tout en reprenant les principes de base. On distingue deux catégories de prestations : Les services utilisateurs et les services logiciels **[12].**

Les services contextuels connaissent aujourd'hui un grand développement, par exemple dans la sécurité, le contrôle de la vie privée et l'adaptation des contenus. Il existe quatre grandes familles de services : les services de contenu, les services de surveillance, les services de communication et les services de commerce électronique **[13].**

3.2 Définition de la composition des services :

La composition des services consiste à combiner la fonctionnalité de plusieurs services au sein d'un même processus afin de répondre à des demandes complexes qu'un seul service ne pourrait satisfaire. La composition de service offre la possibilité d'effectuer des activités complexes à partir de services existants, et les services composés forment un nouveau service, qui peut être réutilisé dans une autre composition **[14].**

Autre définition :

La composition de services peut être vue comme un mécanisme qui permet l'intégration des services pour réaliser une application. Le résultat d'une composition est un nouveau service, appelé service composite. Ce type de composition est dite récursif ou hiérarchique. D'autre sort la composition consiste à combiner les fonctionnalités de plusieurs services au sein d'un même processus métier dans le but de répondre à des demandes complexes qu'un seul service ne pourrait pas satisfaire **[15].**

3.3 Approche existantes pour la composition des services dans l'IoT

Nous allons présenter quelques approches déjà proposées pour la composition de services dans l'IoT **[12]** :

1. **Une méthode basé QoS** pour la composition des services a été proposée, les auteurs ont utilisé une prise de décision multi-attributs pour calculer les performances QoS et évaluer chaque service individuel. Selon les performances de QoS et la valeur fonctionnelle du service, plusieurs services sont choisis pour être dans la composition. Les auteurs divisent la composition des services en analyse de la demande des utilisateurs, recherche et correspondance de services, sélection de services et enfin composition de services, et un algorithme génétique amélioré est utilisé pour trouver les solutions optimales de composition de services.

2. **Un nouvel algorithme de composition basé sur le réseau de Petri (FindTOptimal)** a été proposé pour trouver le chemin le plus optimal, qui utilise une fonction de performance complète nommée (rtc) pour évaluer la rentabilité. Pour gérer le changement dynamique des environnements, les auteurs ont proposé un algorithme de surveillance (FbasedMonitor) pour surveiller le système IoT de manière la moins chère.
3. **Une méthode contextuelle** pour la composition de services a été proposée, où les auteurs ont utilisé OWL pour construire l'ontologie de contexte dans l'environnement de l'IoT. Afin de partager des concepts et des attributs dans différents sous-domaines, les auteurs divisent le champ de découverte de services en différents sous-domaines. Dans la composition des services, le processus de sélection des services a été divisé en deux sous-processus. Premièrement, les auteurs utilisent le contexte de calcul pour sélectionner les services appropriés. Ensuite, en fonction de la qualité de service des attentes des utilisateurs, ils choisissent les meilleurs services pour la composition des services afin de répondre aux besoins de l'utilisateur. Ainsi, les auteurs ont utilisé les informations contextuelles pour optimiser la composition du service dans un contexte particulier, pour trouver le plus compatible avec les besoins des utilisateurs du service tout en satisfaisant les contraintes contextuelles.

4. **Une approche middleware** pour la composition des services logistiques dans l'IoT a été proposée. Un middleware qui se compose d'agents de ressources et d'agents de tâches est utilisé pour désigner au moment de l'exécution des services composants candidats pour participer à la composition d'une transaction logistique. Chaque service composant est représenté par un agent de ressource, qui est responsable du maintien des informations de QoS du service correspondant. Les agents de tâche représentent les instances de service composite et sont chargés de trouver des services composants qui satisfont aux exigences de QoS de bout en bout de l'utilisateur. Les auteurs n'utilisent que quelques moyens de surveillance pour suivre et assurer les opérations et garantir la robustesse du système logistique.

5. **Une approche de composition dynamique** de services a été proposée, basée sur la génération en ligne et la reprogrammation de plans optimaux pour l'orchestration de services en utilisant une technique de planification heuristique. Le mécanisme de sélection proposé est basé sur une technique d'optimisation qui détecte les situations où certaines exigences de composition ne sont pas satisfaites ou certains services deviennent indisponibles ou lorsque des défaillances/exceptions se produisent. Il a la capacité de fonctionner malgré les changements qui se produisent dans le comportement des services sélectionnés et réduit par conséquent le besoin d'intervention humaine dans la reconfiguration de la composition.

6. **Une approche de composition de services Web RESTful** asynchrone légère pour la composition de services dans l'IoT, qui est basée sur l'extension BPEL [25]. Les auteurs ont divisé l'architecture en six couches et ajouté une interaction asynchrone pour gagner du temps et améliorer les performances.

7. **Une cadre pour la composition des services IoT dans l'IoT** a été proposée. Les auteurs prennent en compte la réputation du prestataire de services et la fiabilité du prestataire de réputation afin d'identifier les meilleurs candidats à la création d'un service composé, pour faire face à une tâche donnée. La réputation d'un fournisseur est calculée localement, de sorte que chaque fournisseur peut calculer une réputation différente en fonction de ses expériences personnelles antérieures avec

différents fournisseurs (asymétrie). Cette approche est coûteuse en calcul et ne gère pas les échecs de connexion, ce qui n'est pas très adapté aux grands environnements

	Adaptabilité	Autonome et indépendance	Décentrali-sation et répartition,	L'optimi-sation de la composition	Approche utilisée

comme un scénario de composition de service IoT.

❖ **Etude comparative entre approches :**

Notre comparaison est basée sur les critères suivants :

- ✓ Adaptabilité,
- ✓ Autonomie et indépendance,
- ✓ Décentralisation et répartition,
- ✓ L'optimisation de la composition.

Le tableau (table 1) dans la page suivante représente la comparaison :

1	-	+	-	+	Algorithme génétique
2	+	+	+	+	Approche du réseau de Petri basée sur la fonction RTC
3	+	+	+	+	Méthode basée sur le contexte
4	+	+	+	+	Une approche middleware
5	+	+	-	-	Technique d'optimisation
6	-	+	+	-	Approche BPEL étendue et interaction asynchrone
7	-	+	-	-	Une approche basée sur la réputation du prestataire et la fiabilité du prestataire de reputation

Table 1 *comparision entre les approches de composition de service IoT.*

Après cette étude, nous pensons de profiter des avantages des travaux cités auparavant, et nous proposons d'utilisé la méthode basé sur le contexte offerte par les ontologies OWL, les concepts ontologiques fondamentaux requis pour les services IoT.

Conclusion

Dans ce chapitre nous avons défini les concepts de base nécessaire et en conclut par une comparaison pour choisir une approche de composition pour résolu notre problème de ce travail.

Le chapitre suivant, on passe à la conception et le développent de notre approche.

Chapitre 2: Composition des services IoT basée ontologie

2 COMPOSITION DES SERVICES IOT BASEE ONTOLOGIE

Introduction

Comme déjà vu dans le chapitre précédent, l'utilisation d'ontologies est l'un des approches de composition des services IoT grâce à ses avantages multiples tel que : le partage des connaissances, l'expressivité, l'inférence logique, la réutilisation des connaissances et l'extensibilité. De plus, faciliter une découverte et une composition de services plus intelligentes.

L'objectif principal de notre ontologie proposée est de :

- Supporter le traitement de données hétérogènes en utilisant les ontologies.
- Composer des services pour remplir les fonctions demandées en fonction des appareils disponibles.
- Supporter la réutilisation et le partage des connaissances entre les différentes applications IoT.

Les détails sur notre ontologie (classes avec leurs individus et propriétés) sont décrits dans cette section.

1. Présentation de l'environnement domotique

Nous proposons ici une étude spécifique qui soutient les utilisateurs dans un environnement intelligent (smart home). Cet environnement intelligent peut réagir aux événements de la vie quotidienne de l'utilisateur. L'objectif de notre projet est de gérer efficacement toutes les données contextuelles du domaine de la vie quotidiennes en améliorant la surveillance et le signalement des conditions critiques.

Il existe plusieurs objets intelligents. Chaque objet intelligent sert à surveiller des informations telles que capteur de mouvement pour surveiller le déplacement, une caméra intelligente pour surveiller l'activité (par exemple, regarder la télévision, boire un café ou une boisson, dormir, douche, etc.) et d'autres capteurs pour détecter d'autres informations contextuelles (température et taux d'humidité, état des portes et des lumières et des mouvements de l'utilisateur).

2. Architecture du système

L'architecture générale d notre système comporte les quatre (4) couches suivantes :

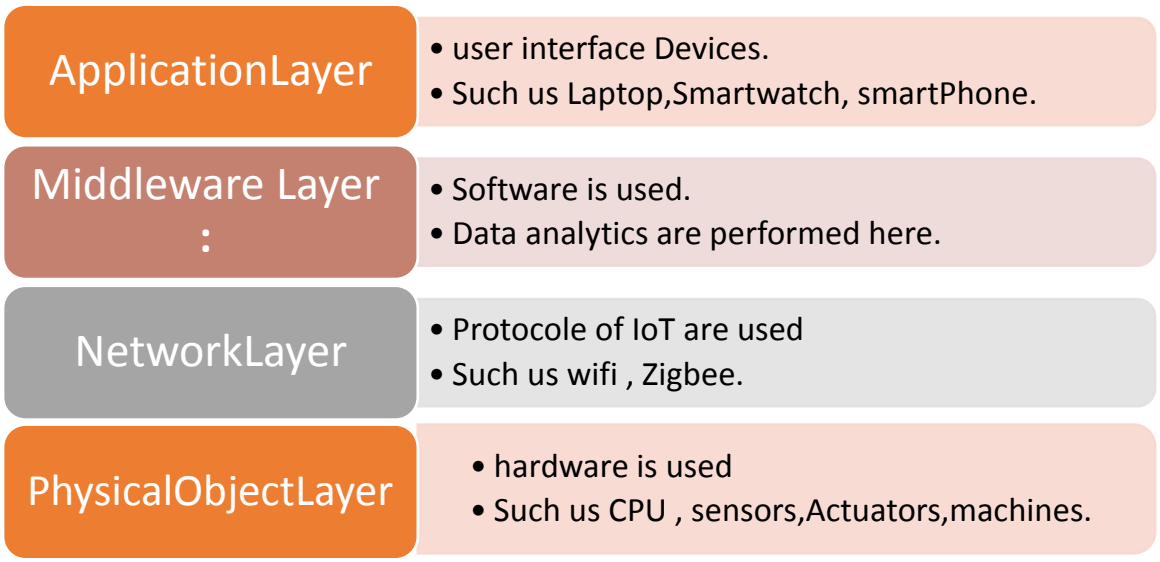

***Figure 3** : Architecture Système.*

3. Scenario proposé

Nous nous intéressons à la vie quotidienne d'un utilisateur (Mehdi). Il vit dans le smart home décrite ci-dessus. Il travaille régulièrement dans un bureau de banque. Il a des besoins spécifiques basés sur ses besoins, préférences et situations actuels.

Il dispose un agenda qui décrit précisément les activités qu'il a prévues. Le système vérifie son agenda personnel sur Google pour trouver une façon appropriée de gérer ses journées de travail.

Les fonctionnalités qui doivent être assurées par le système sont les suivantes :

- Lorsque le moment est venu, le système augmente l'éclairage de la chambre à coucher de façon régulière et automatique. La caméra de la chambre peut reconnaître automatiquement s'il se réveille ou non.
- Quand il se réveille, le système vérifie la prochaine tâche programmée de Mehdi, qui est : (prendre une douche). Le système adapte la température de l'eau lorsqu'il quitte la salle de bain.

- Lorsqu'il se rend à la cuisine, le dispositif envoie une commande à la cafetière intelligente pour préparer le café. Ensuite il se rend dans le bureau où il boit tranquillement son café.
- Le framework déploie un service de messagerie sur son ordinateur personnel avec tous les outils qu'il trouve utiles dans son domaine de travail et déploie les nouvelles du matin sur son Smartphone.

4. L'environnement domotique proposé

Pour le développement de notre ontologie nous avons suivi les quatre phases du cycle de développement suivant :

Phase 1 : évaluation des besoins

Cette étape doit préciser :

- **L'objectif opérationnel de l'ontologie :**
  - C'est l'utilisation de cette ontologie pour la composition des services IoT et la représentation de contexte hétérogène dans un environnement domotique (smart home).
- **Le domaine de connaissances à modéliser :** C'est la modélisation des connaissances de contexte de l'utilisateur dans un environnement domotique.
- **Les utilisateurs** : personne qui habite dans cette maison.

Dans notre étude de cas, smart home agit comme un environnement composé de plusieurs pièces : cuisine, salon, salle de bain, bureau, garage et une chambre à coucher. Chaque chambre est équipée d'objets intelligents, Le tableau suivant présente une liste des objets et dispositifs intelligents disponibles dans la maison :

Chambres intelligentes	Capteurs	Actionneurs
Chambre à coucher	Capteur de température, capteur de lumière, capteur d'état de porte, caméra IP,	Actionneur d'interrupteur d'éclairage, verrouillage de la porte.

Salle de séjour	Capteur de température, capteur de lumière, capteur de mouvement, caméra IP.	Actionneur d'interrupteur d'éclairage. Ordinateur portable, tablette, Télévision intelligente.
Cuisine	Capteur de lumière, caméra IP, capteur cafetière.	Actionneur d'interrupteur d'éclairage, Télévision intelligente. cafetière intelligente.
Salle de bain	Capteur de température, capteur d'humidité, capteur d'eau, capteur de mouvement, Capteur de lumière.	Actionneur d'interrupteur d'éclairage, Interrupteur d'eau
Bureau	Capteur de température, capteur de lumière, caméra IP	Actionneur d'interrupteur d'éclairage, Ordinateur portable.

Tableau 1 *Différents objets intelligents dans le smart home.*

Remarque : Évidemment un humain peut également intervenir manuellement sur chacun des éléments indépendamment du système.

Dans notre solution de smart home, une ontologie est introduite pour traiter les problèmes énoncés ci-dessus sur la base des informations sémantiques. Notez qu'il est possible qu'une fonction demandée ne puisse pas être remplie par un seul appareil mais par un ensemble d'appareils fonctionnant ensemble. Par conséquent, en plus de la spécification des relations statiques entre les concepts dans les systèmes d'ontologie, nous introduisons le modèle d'application pour la spécification de la composition de service.

Nous constatons que les entités Emplacement (Location), Observation (Observation), Capteur (Sensor) représentent les entités contextuelles sont les plus fondamentales pour capturer l'information du contexte dans un environnement intelligent.

Normalement, les modifications apportées à la spécification de l'ontologie n'affecteront pas le plan de fonction en cours d'exécution. Cependant, la maintenance de l'ontologie peut devoir être effectuée pour une utilisation future. Il y a trois changements fondamentaux sur système d'ontologie : ajout d'un nouvel élément, suppression d'un élément existant et modification d'un élément existant.

L'ajout de toute nouvelle information nécessite une mise à jour et une vérification importantes. Un élément ajouté doit satisfaire aux contraintes du système d'ontologie. Si une nouvelle relation est ajoutée à des classes existantes, les relations d'origine et dérivées doivent être examinées pour s'assurer qu'il n'y a pas de violation entre elles. Si un nouvel élément est ajouté sous un concept donné, toutes les relations associées au concept doivent être vérifiées avec le nouvel élément. De plus, la vérification de cohérence doit être effectuée sur des relations mises à jour.

Phase 2 : Conceptualisation par la technique UML

Cette étape concerne la conceptualisation des concepts du domaine dans un modèle conceptuel. Les connaissances du domaine domotique sont décrites par des concepts et des relations.

Dans notre système, on a utilisé le diagramme de classe UML pour représenter les différents concepts et relations entre concepts.

Après analyse de notre environnement proposé, les concepts que nous avons identifiés pour le développement de notre ontologie sont les suivants :

- Command
  - systemCmd
  - platformCmd
  - modeCmd
- Service
- Observation
- Property
- Unit
- Location
- PhysicalObject
  - Platform

- lamp
- pc
- Tablet
- coffee machine
- smartPhone
- tv
- room

➢ System

▪ Sensor

- Humidity sensor
- Motion sensor
- Coffee machine sensor
- Temperature sensor
- Water sensor
- Ip camera

• Actor

- Agenda
- Activity

La représentation UML avec un diagramme de classe, de ces concepts sont modélisés par des classes ou l'identité de chaque classe c'est le nom du concept qu'elle représente et ses attributs c'est les informations de même ce concept.

La représentation UML (dans le diagramme de classe) de ces concepts est décrite par les classes suivantes :

Activity : cette classe décrit l'état actuel de l'utilisateur dans l'environnement qui comprend l'activité décrivant la tâche en cour.

Actor : cette classe a pour objectif de décrire toute personne impliquée dans l'environnement domotique.

Platform : Une entité à laquelle d'autres entités peuvent être attachées - principalement des capteurs : cuisine, tablette, TV, cafetière, chambre, salle de bain, bureau, électricité, gaz et eau...

System : une entité d'abstraction pour des éléments d'infrastructure de détection. Un système peut avoir des sous-systèmes, qui sont également des systèmes. Il y a quatre personnes dans

notre système : le système d'appareils électroménagers, le système de chauffage, le système d'éclairage et le système de sécurité.

Sensor : Représente les capteurs, une sous-classe de System. Cette classe vise à décrire tous les capteurs utilisés dans l'environnement et cela pour capturer les différentes informations de contexte y compris : LightSensor, TemperatureSensor, waterSensor…etc.

PhysicalObject : Une classe générique pour tout élément physique. Il a deux sous-classes dans notre ontologie : Platform et System.

Cammand : cette classe une décrit des commandes pour fonctionner le système ou un device.

System : cette classe décrit les types des systèmes comme : système sécurité, système temperature…

Observation : Chaque sortie d'un capteur est un membre de la classe Observation dans notre ontologie.

Service : cette classe décrit une entité de service associe personne ou un platform.

Agenda : cette classe décrit précisément les activités de l'utilisateur.

Property : cette classe décrit les caractéristique, Il y a cinq individus dans cette classe : éclairement, mode, puissance, état et température.

Unit : Classe pour les types d'unité de mesure. Nous avons trois types d'unités : Celsius pour la température, Lux pour l'éclairement et Watt pour la puissance.

C'est différentes classes sont représentées par le diagramme de classe UML suivant :

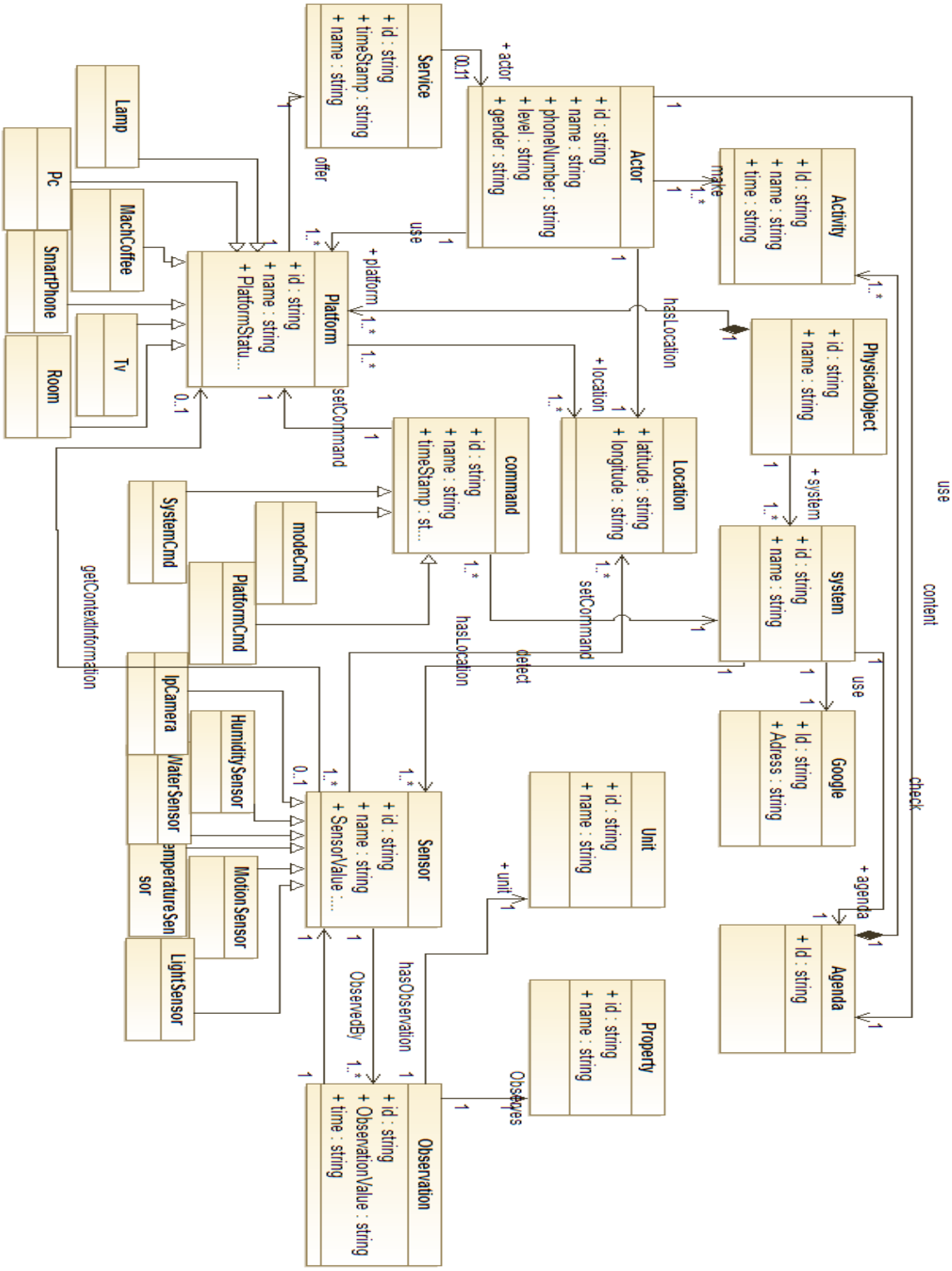

Figure 4 Figure 1 Diagramme de Classe de l'environnement domotique.

2 COMPOSITON DES SERVICES IOT BASEE ONTOLOGIE

Les propriétés des différentes classes de notre ontologie sont les suivantes :

- **hasLocation** : cette relation affecte une entité physique ou une personne à une localisation dans l'environnement.
- **hasActivity** : relation entre la classe Actor et la classe Activity. Cette relation définit l'activité courante des acteurs.
- **Observed property** : relation entre la classe Activity et la classe Devise. Elle définit les dispositifs utilisés dans une activité.
- **attachedSystem** : Relation entre une plate-forme et tout système.
- **hasSubSystem :** A une relation partielle entre un système et ses parties**.**
- **isPropertyOf :** Relation entre une caractéristique d'intérêt et une propriété.
- **observedBy :** relation entre sensor et observation**.**
- **Observes :** Relation liant une Observation à la Propriété observée.
- **commandToSystem :** exprime la commande au System.
- **featureOfInterest :** relation entre une observation et l'entité dont la qualité a été observée.
- **onPlatform :** Relation entre un système (par exemple, un capteur) et une plate-forme.
- **commandToPlatform** : exprime la commande au platform
- **start system** : exprime la commande start à un System.
- **stop system** : exprime la commande stop à un System.
- **start platform** : exprime la commande start à un Platform.
- **stop platform** : exprime la commande stop à un Platform
- **observationValue :** Décrit la valeur d'observation.
- **Time :** Affiche l'heure d'une commande ou d'une observation**.**
- **safeMax :** Décrit la valeur maximale de la plage de sécurité**.**
- **safeMin :** Décrit la valeur minimale de la plage de sécurité**.**

La description des attribues des classes et des cardinalités des relations :

- **classe "PhysicalObject"**

Nom	description
id : [1...1] string	L'Identité de la classe
name : [1...1] string	Name de chambre

***Table 2** : attributs de la classe PhysicalObject.*

❖ **classe "Actor"**

Nom	Description
id : [1...1] string	L'ID de la classe
name : [1...1] string	Son prénom
level : [1...1] string	Son nuveau
PhoneNumber : [1...1] string	Son Numéro de telephone
Gender : [1...1] string	Son sexe

***Table 3** : attributs de la classe Actor.*

- Relations de la classe "**Actor**"

Nom	Description
Use : [0...*] Device	Un acteur peut utiliser un appareil
Use : [1...1] Agenda	Un acteur peut utiliser l'agenda
LocatedIn : [0...1] Location	La localisation de l'acteur dans la chamber
hasActivity : [1...*] Activity	L'activité de l'acteur

***Table 4** : Associations de la classe Actor.*

❖ **Classe "Platform"**

Nom	Description
id : [1..1] string	L'ID du Dispositif
name : [1..1] string	Son nom
deviceStatus : [1..1] string	Son statut ('On' ,'Off')

***Table 5** : attributs de la classe Platform.*

- Reletion de la classe **"Platform"**

Nom	description
use : [0..*] Actor	L'utilisation du dispositif par un acteur
locatedIn : [0..1] Location	La localisation du dispositif

getContexteInformation : [1...*] sensor	La récupération des informations de contexte à partir d'un capteur
hasService : [0..*] Service	Offrir un service

***Table 6** : Associations de la classe Platform.*

- **Classe "TV"**

Hérite de : "Platform"

Nom	Description
Les mêmes attributs de la classe Platform	

***Table 7** : attribut de la classe Tv.*

- **Classe "MachCoffee"**

Hérite de : "Platform"

Nom	Description
Les mêmes attributs de la classe Platform	

***Table 8** : attribut de la classe MachCoffee.*

- **Classe "Lamp"**

Hérite de : "Platform"

Nom	Description
Les mêmes attributs de la classe Platform	

***Table 9** : attributs de la classe Lamp.*

- **Classe "SmartPhone"**

Hérite de : "Platform"

Nom	Description
Les mêmes attributs de la classe Platform	

***Table 10** : attribut de la classe SmartPhone.*

❖ **Classe "Pc"**

Hérite de : "Platform"

Nom	Description
Les mêmes attributs de la classe Platform	

***Table 11** : attributs de la classe PC.*

❖ **Classe "Tablet"**

Hérite de : "Platform"

Nom	Description
Les mêmes attributs de la classe Platform	

***Table 12** :attribut de la classe Tablet.*

❖ **Classe "Room"**

Hérite de : "Platform"

Nom	Description
Les mêmes attributs de la classe Platform	

***Table 13** :attributs de la classe room.*

❖ **Classe "Sensor"**

Nom	description
id : [1..1] string	L'ID du capteur
name : [1..1] string	Son nom
sensorValue : [1..1] string	La mesure de l'information

***Table 14** : attribut de la classe Sensor.*

- Relation de la classe **"Sensor"**

Nom	Description
LocatedIn : [0...1] Location	La localisation du capteur
getContexteInformation : [1...*] Device	pour capturer les formations de contexte à partir des dispositifs

***Table 15** : associations de la classe Sensor.*

❖ **Classe "HumiditySensor"**

Hérite de : "Sensor"

Nom	Description
Les mêmes attributs de la classe Sensor	

***Table 16** : attribut de la classe HumiditySensor.*

❖ **Classe "temperatureSensor"**

Hérite de : "Sensor"

Nom	Description
Les mêmes attributs de la classe Sensor	

***Table 17** : attributs de la classe temperatureSensor.*

❖ **Classe "LightSensor"**

Hérite de : "Sensor"

Nom	Description
Les mêmes attributs de la classe Sensor	

***Table 18** : attributs de la classe LightSensor.*

❖ **Classe "MotionSensor"**

Hérite de : "Sensor"

Nom	Description
Les mêmes attributs de la classe Sensor	

***Table 19** : attribut de la classe MotionSensor.*

❖ **Classe "WaterSensor"**
Hérite de : "Sensor"

Nom	Description
Les mêmes attributs de la classe Sensor	

***Table 20** : attribut de la classe WaterSensor.*

❖ **Classe "IpCamera"**
Hérite de : "Sensor"

Nom	Description
Les mêmes attributs de la classe Sensor	

***Table 21** : attributs de la classe IpCamera.*

❖ **Classe "Command"**

Nom	Description
id : [1..1] string	L'id de la commande
name : [1..1] string	Nom de commande
timeStamp : [1..1] string	Le temps associe à la commande

***Table 22** : attribut de la classe Cammand.*

Relations de la classe **"Command"**

Nom	Description
setCommand : [1..1] Platform	Envoyer une commande au Platform
setCommand : [1..1] System	Envoyer une commande système

***Table 23** : associations de la classe Cammand.*

❖ **Classe "SystemCmd"**

Hérite de : "Command"

Nom	Description
Les mêmes attributs de la classe Command	

***Table 24** : attribut de la classe SystemCmd.*

❖ **Classe "PlatformCmd"**

Hérite de : Command

Nom	Description
Les mêmes attributs de la classe Command	

***Table 25** : attributs de la classe PlatformCmd.*

❖ **Classe "System"**

Nom	Description
id : [1..1] string	L'Id de système
messageContent : [1..1] string	Le message envoyé par le systeme
name : [1..1] string	Nom de système

***Table 26** : attributs de la classe System.*

Relations de la classe **"System"**

Nom	description
Check [1..1] Agenda	Vérifier les activités de l'acteur dans l'agenda
Use [1..1] Google	Utiliser Google pour vérifier l'agenda
détecter [1..*] Sensor	Détecter les capteurs nécessaires

***Table 27** : associations de la classe System.*

❖ **Classe "Location"**

Nom	description
Latitude [1..1] string	La latitude de la localisation
Longitude [1..1] string	La longitude de la localisation

***Table 28** : attribut de la classe Location.*

❖ **Classe "Observation"**

Nom	description
id : [1..1] string	L'id de l'observation
observationValue : [1..1] string	La valeur observée par le capteur
time : [1..1] string	L'instant de l'observation

***Table 29** : attributs de la classe Observation.*

Relations de la classe **"Observation"**

Nom	description
Observes [1..1] Property	La propriété que le capteur observe.
hasUnit [1..1] Unit	l'unité de mesure de valeurs transmises par le capteur

***Table 30** : associations de la classe Observation.*

❖ **Classe "Unit"**

Nom	description
id : [1..1] string	L'id de l'unité
name : [1..1] string	Le nom de l'unité

***Table 31** : attributs de la classe Unit.*

❖ **Classe "Property"**

Nom	description
id : [1..1] string	L'id de la propriété
name : [1..1] string	Le nom de la propriété

***Table 32** : attribut de la classe Property.*

❖ **Classe "Agenda"**

Nom	description
id : [1..1] string	L'id de l'agenda

Table 33 : *attribut de la classe Agenda.*

❖ **Classe "Service"**

Nom	description
id : [1..1] string	L'ID de service
name : [1..1] string	Son Nom
time : [1..1] string	Le temps de déroulement de service

Table 34 *: attributs de la classe service.*

❖ **Classe "Activity"**

Nom	description
id : [1..1] string	L'ID de l'activité
name : [1..1] string	Son Nom
time : [1..1] string	Le temps de déroulement de l'activité

Table 35 *: attribut de la classe Activity.*

Phase 3 : formalisation de l'ontologie développée par le langage OWL

On parle ici de la troisième étape dans le cycle de vie de développement de l'ontologie. Il s'agit de l'expression explicite et formelle de la conceptualisation obtenue dans l'étape précédente par un langage formel qui offre :

✓ un ensemble des composants sémantiques,

- ✓ des règles structurelles et d'une notation formelle destinée à organiser les relations entre les éléments constituant l'ontologie,
- ✓ permettre de réduire les ambiguïtés du langage naturel en offrant une plus grande expressivité et rendre l'ontologie compréhensible par les machines.

Dans notre étude on a utilisé le langage OWL qui représente le standard proposé par le W3C pour la formalisation de notre ontologie. Le langage OWL utilise les primitives de bases modélisées par les schémas RDF (les classes et les propriétés). Le langage OWL permet de décrire les concepts d'une façon plus expressive que le diagramme de classe UML il permet aussi d'ajouter des restrictions formelles et des caractéristiques aux classes et aux propriétés qui modélise le contexte.

Pour la formalisation de notre ontologie, on a utilisé l'éditeur "Protege-2000». Ce dernier permet de construire une ontologie pour un domaine donné, de définir les formulaires de saisie de données ainsi que les données à atteindre à l'aide de ces formes sous la forme d'instances de cette ontologie.

La formalisation de notre ontologie suive ces étapes :

- ➢ **Définition des classes.**
- ➢ **Définition des propriétés (Object Properties et Data Properties).**
- ➢ **Ajoute d'individus.**

Conclusion

Dans ce chapitre nous avons présenté le processus de développement de notre ontologie spécifique pour la composition des services IoT dans un environnement domotique.

Le processus de développement adopté se résume en quatre étapes qui sont : l'évaluation des besoins, la conceptualisation de l'ontologie par le diagramme de classe UML, la formalisation de cette conceptualisation par le langage OWL et enfin l'implémentation qui a été faite par l'outil Protégé-5.5.0.

Dans le chapitre suivant, nous montrons les différents outils et les différentes étapes de la mise en œuvre de notre travail.

Chapitre 3: Implémentation

3 IMPLEMENTATION

Introduction

Dans ce chapitre, nous monterons les étapes d'implémentation et les différentes captures d'écrans de notre projet.

1. langage et les outils de développements :

1.1 Protégé

PROTEGE-OWL [17] est une interface modulaire, développée au (Stanford Medical Informatics) de l'Université de Stanford, permettant l'édition, la visualisation, le contrôle textuel, et la fusion

Semi-automatique d'ontologies. Le modèle de connaissances de (propriétés) et des facettes (valeurs des propriétés et contraintes), ainsi que des instances des classes et des propriétés. PROTEGE-OWL autorise la définition de méta-classes, dont les instances sont des classes, ce qui permet de créer son propre modèle de connaissances avant de bâtir une ontologie. De nombreux plug-ins sont disponibles ou peuvent être ajoutés par l'utilisateur. Les points forts de Protégé :

- Construire des ontologies.
- Personnaliser des formulaires d'acquisition des connaissances.
- Transférer la connaissance de domaine.
- Faire des contrôles de cohérence de l'ontologie.

1.2 Modelio

Modelio [18] est un outil de modélisation UML2 complet. Modelio fournit un référentiel central pour les modèles, permettant la combinaison de différents langages (profils UML) dans un même modèle. Modelio supporte la modélisation BPMN, la gestion des exigences, du dictionnaire, et fournit des générateurs vers les cibles techniques essentielles (Java, C#, C++, SQL, XML, ...). Il peut être paramétré pour adapter la modélisation, ou les générations.

Les fonctionnalités de modelio sont :

- **Modélisation UML et BPMN**

  – Support complet des derniers standards.

  – Intégration pour une continuité et une cohérence renforcées.

  – Format d'échange XMI.

- **Un référentiel unique et centralisé**

  –Un seul référentiel pour les modèles UML, BPMN, les exigences, les objectifs, et toutes extensions.

  – Contrôle de cohérence global sur l'ensemble des modèles.

  – Gestion de la traçabilité intégrée : liens de traçabilité, analyse d'impact, éditeur de liens**.**

2. L'implémentation

2.1 La création les classe d'ontologie dans Protégé :

Nous avons utilisé Protégé 5.5.0 pour éditer notre ontologie. La figure suivante représente les classes de notre ontologie :

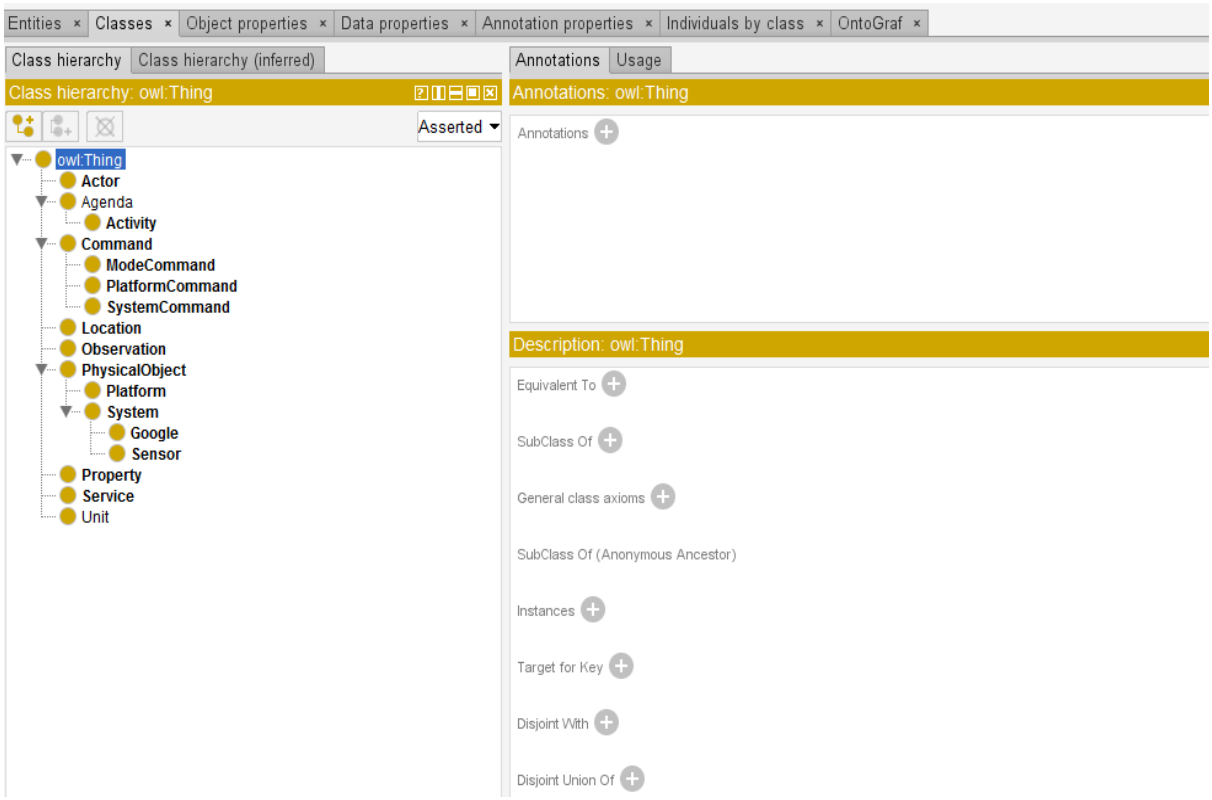

Figure 5 *creation des classes dans protégé.*

Toutes ces classes sont héritées de la superclasse Thing (donc toutes les autres classes sont des sous-classes).

Pour définir la relation d'héritage on utilise la propriété subClassOf. Par exemple la relation d'héritage entre la classe System et la classe PhysicalObject est définie comme suit :

```
<owl:Class rdf:about="&ssn;System">
<rdfs:subClassOf rdf:resource="&dul;PhysicalObject"/>
</owl:Class>
```

Figure 6 : *déclaration de la subClass System.*

2.2 Définition des Propriétés de l'ontologie développée :

L'objectif des propriétés dans une ontologie est d'exprimer les faits au sujet de ces classes ainsi que de leurs instances.

2.2.1 Les propriétés d'objet :

Une propriété d'objet est une instance de la classe owl:ObjectProperty permet de relier des instances à d'autres instances

Les propriétés d'objet de notre ontologie sont dérivées directement à partir des relations de diagramme de classe UML, et enrichis par les caractéristiques nécessaires qui permettent de raisonner sur l'ontologie.

Les figures suivantes représentent la création des propriétés d'objet de notre ontologie :

```
    <owl:ObjectProperty rdf:about="&ssn;onPlatform">
    <rdfs:label>on platform</rdfs:label>
    <rdfs:range rdf:resource="&ssn;Platform"/>
    <rdfs:domain rdf:resource="&ssn;Sensor"/>
    <rdfs:domain rdf:resource="&ssn;System"/>
    <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
</owl:ObjectProperty>
```

Figure 7 *: déclaration de la propiérité d'objet OnPlatform.*

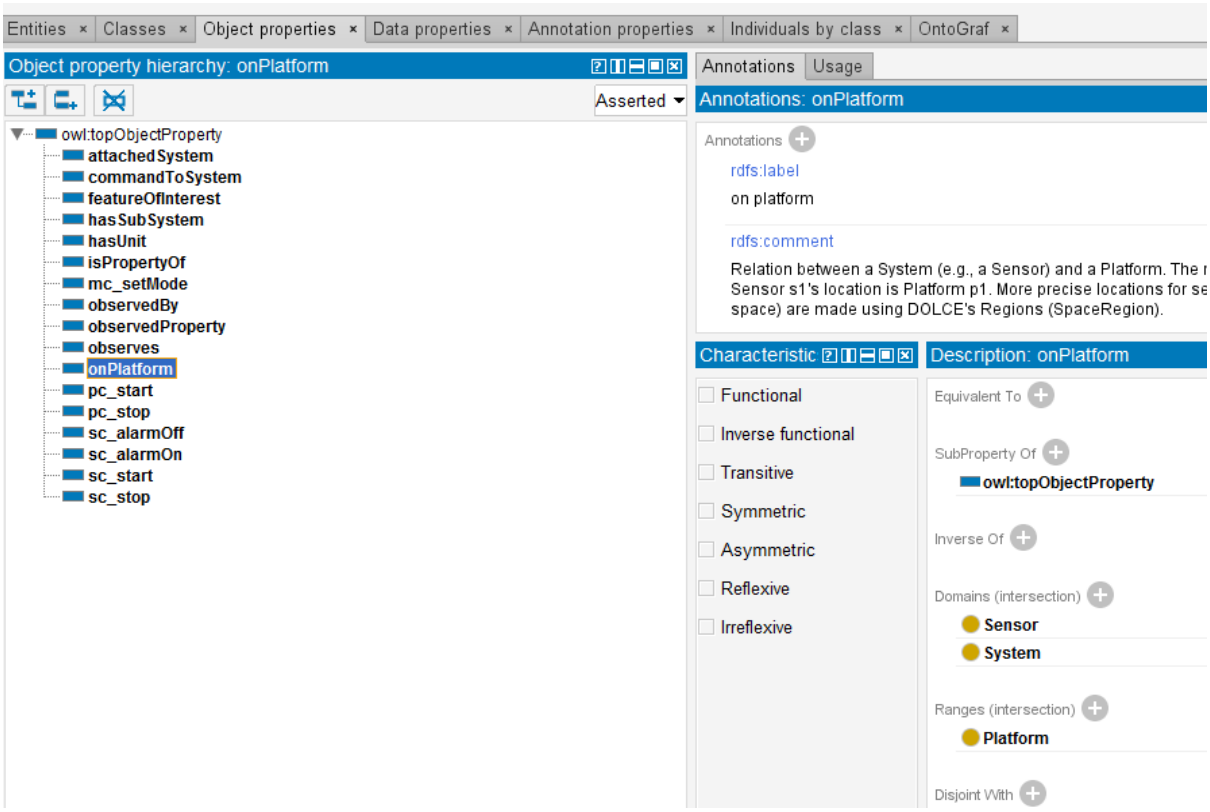

Figure 8 *la creation de la propriété "onPlatform"*

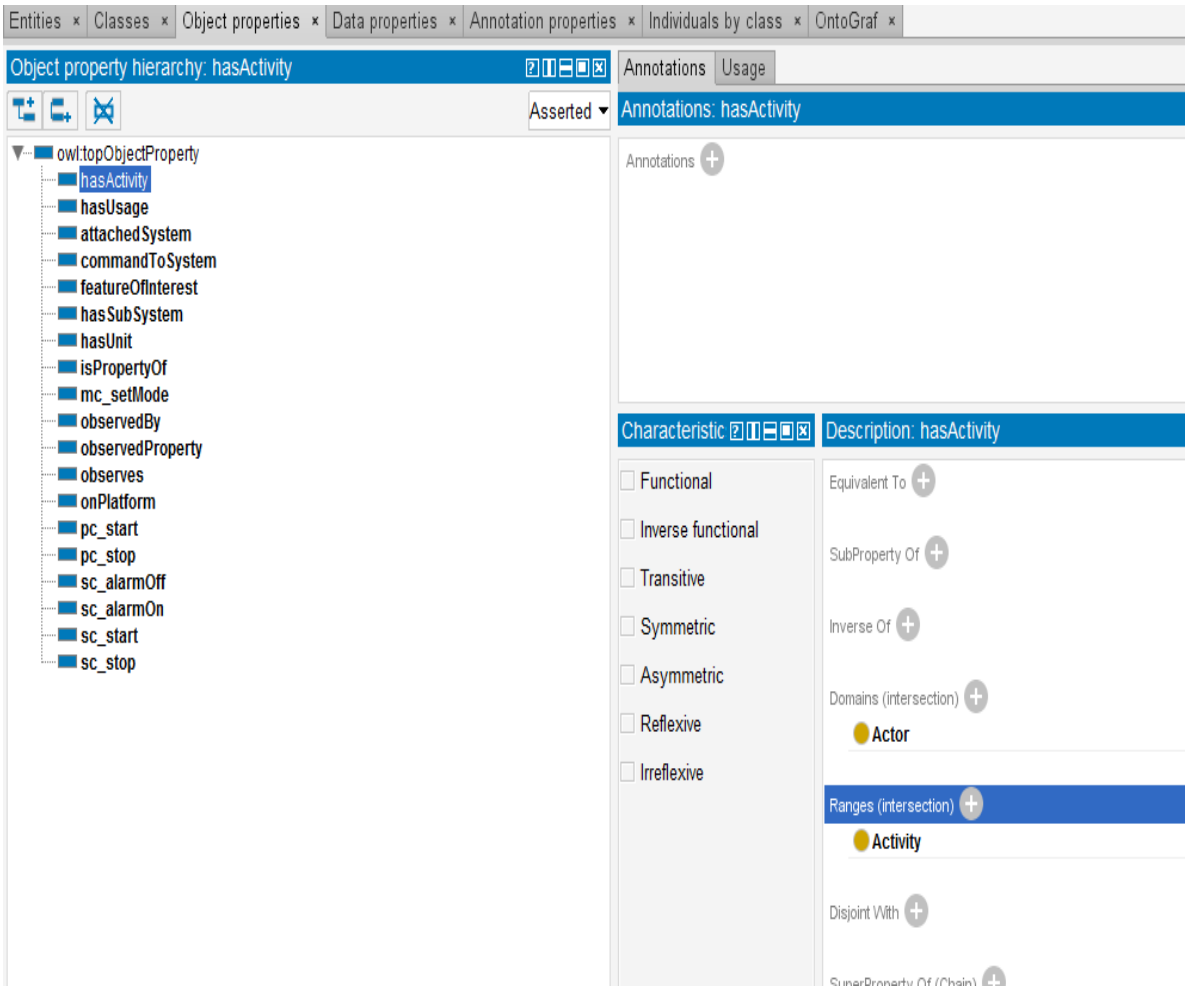

Figure 9 *: création de la propriétés d'objet "hasActivity".*

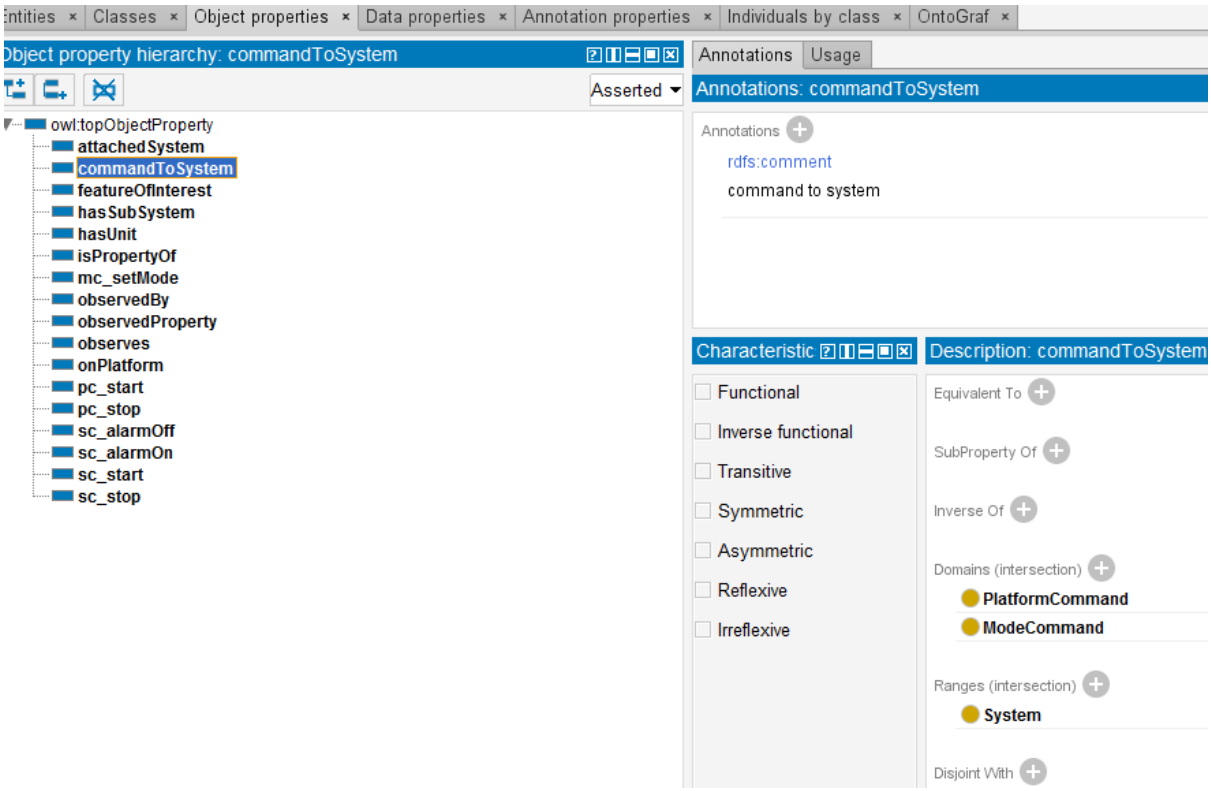

Figure 10 *la création de la propriété d'objet "commandToSystem".*

2.2.2 Les propriétés de type

Une propriété de type de donnée étant une instance de la classe owl:DatatypeProperty, permet de relier des individus à des valeurs de données.

Les figures suivantes représentent la création des propriétés de types de notre ontologie :

```
'

<owl:DatatypeProperty rdf:about="http://www.smarthome.com/ontology#observationValue">
    <rdfs:label>observation value</rdfs:label>
    <rdfs:domain rdf:resource="&ssn;Observation"/>
</owl:DatatypeProperty>
```

Figure 11*: declaration de la propriété de type ObservatioValue.*

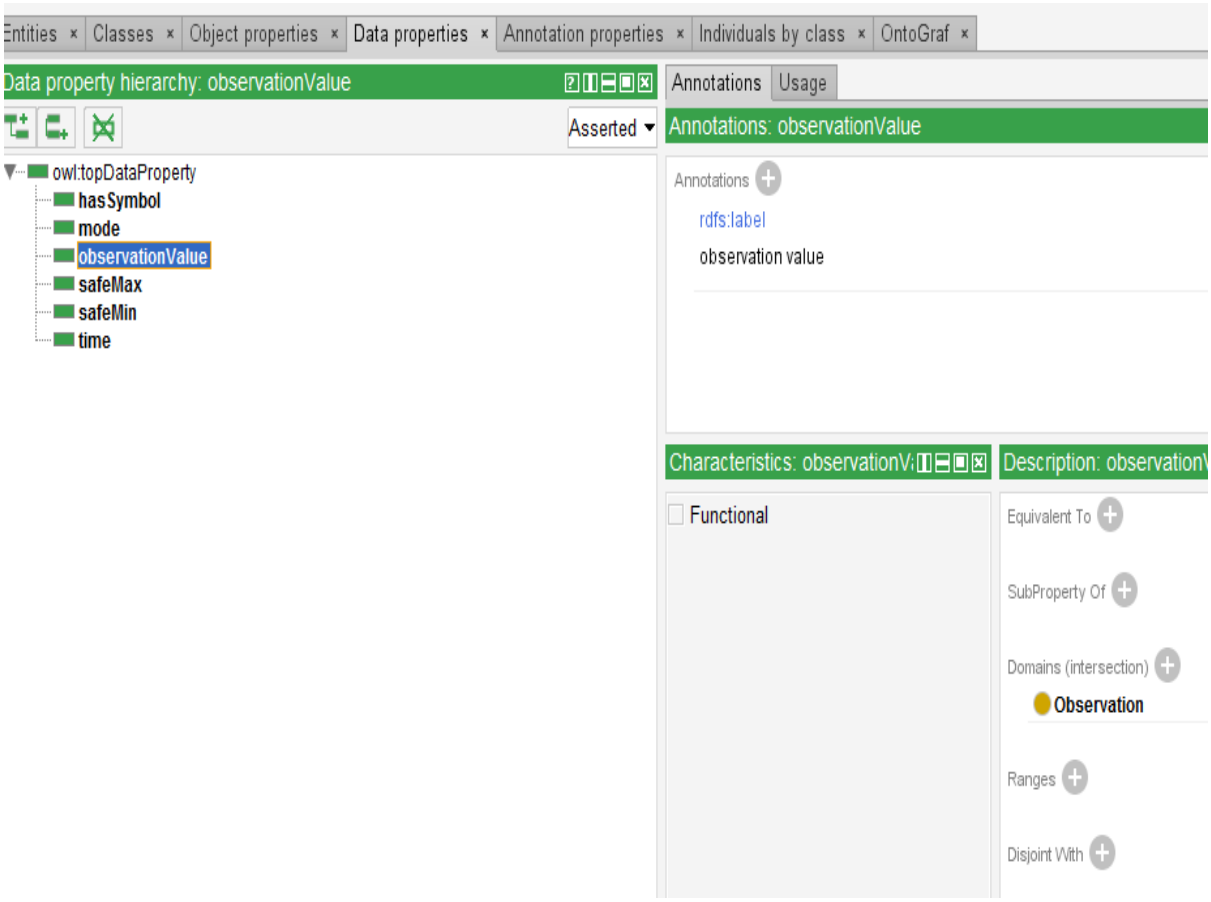

Figure 12 *: déclaration de la propriété de type ObservationValue.*

```
    <owl:DatatypeProperty rdf:about="http://www.smarthome.com/ontology#hasSymbol">
    <rdfs:label>has symbol</rdfs:label>
    <rdfs:domain rdf:resource="http://www.smarthome.com/ontology#Unit"/>
    <rdfs:range rdf:resource="&xsd;string"/>
    <rdfs:subPropertyOf rdf:resource="&owl;topDataProperty"/>
</owl:DatatypeProperty>
```

Figure 13 *: la déclaration de la propriété de type hasSymbol.*

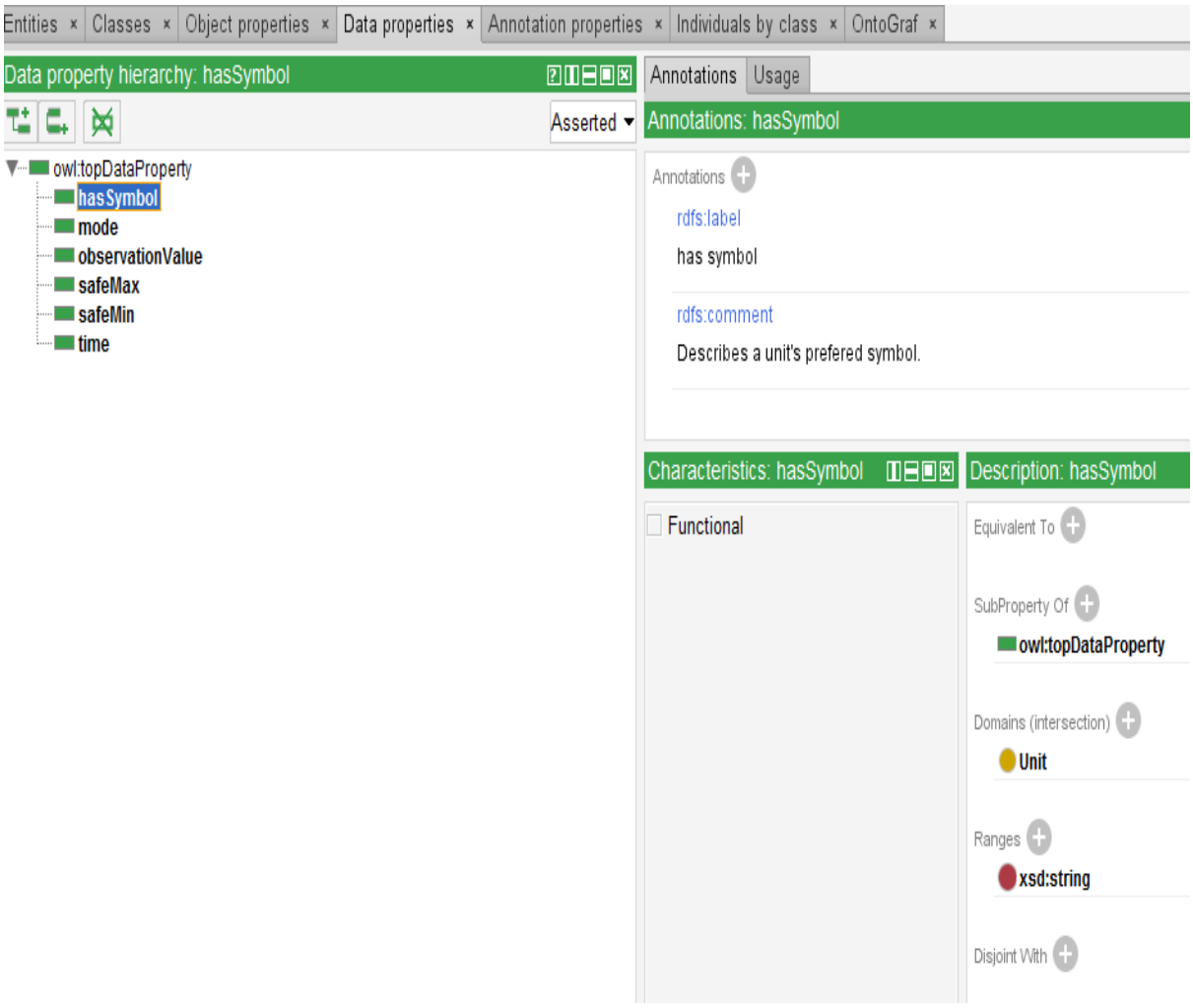

***Figure 14** : la création de la propriété de type hasSymbol.*

2.3 Ajout d'individus aux classes :

Ajouter des instances des classes dans l'onglet **Individuals by class** Les figures suivante représentent les créations des individuels :

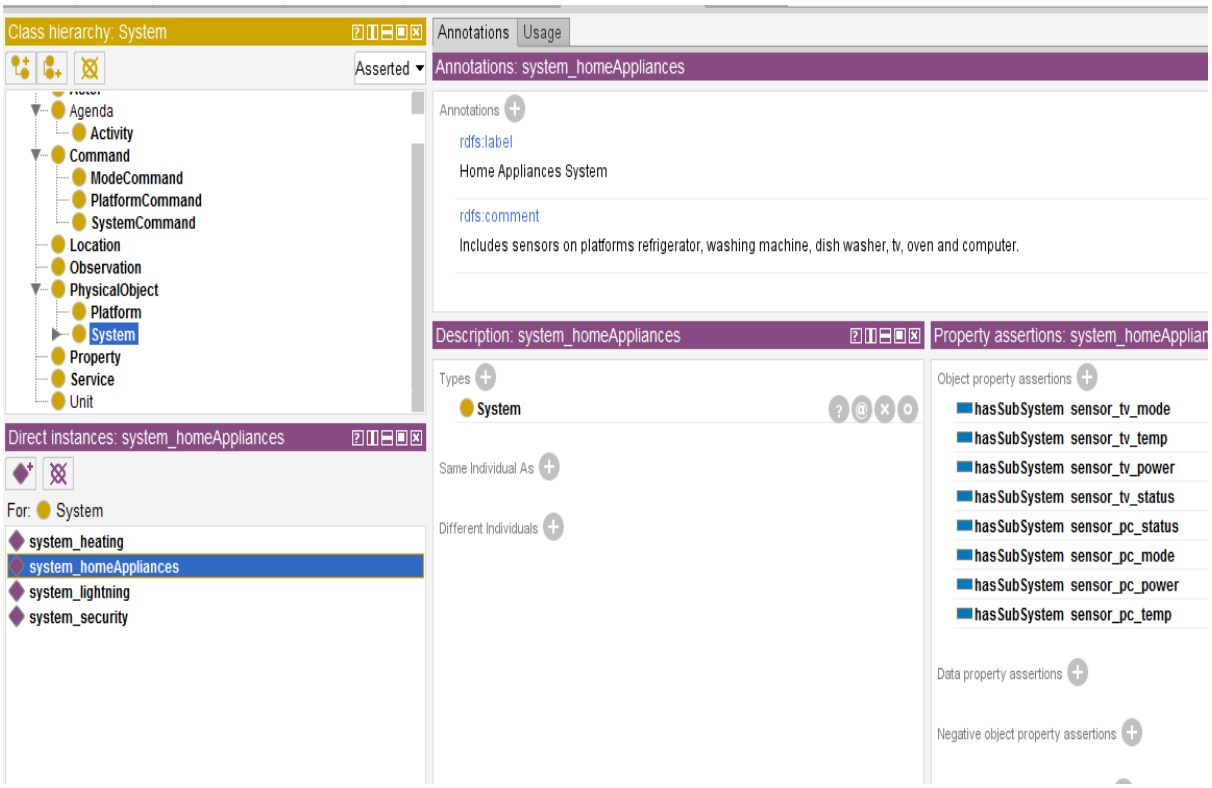

Figure 15 *Ajout d'individus au classe "System".*

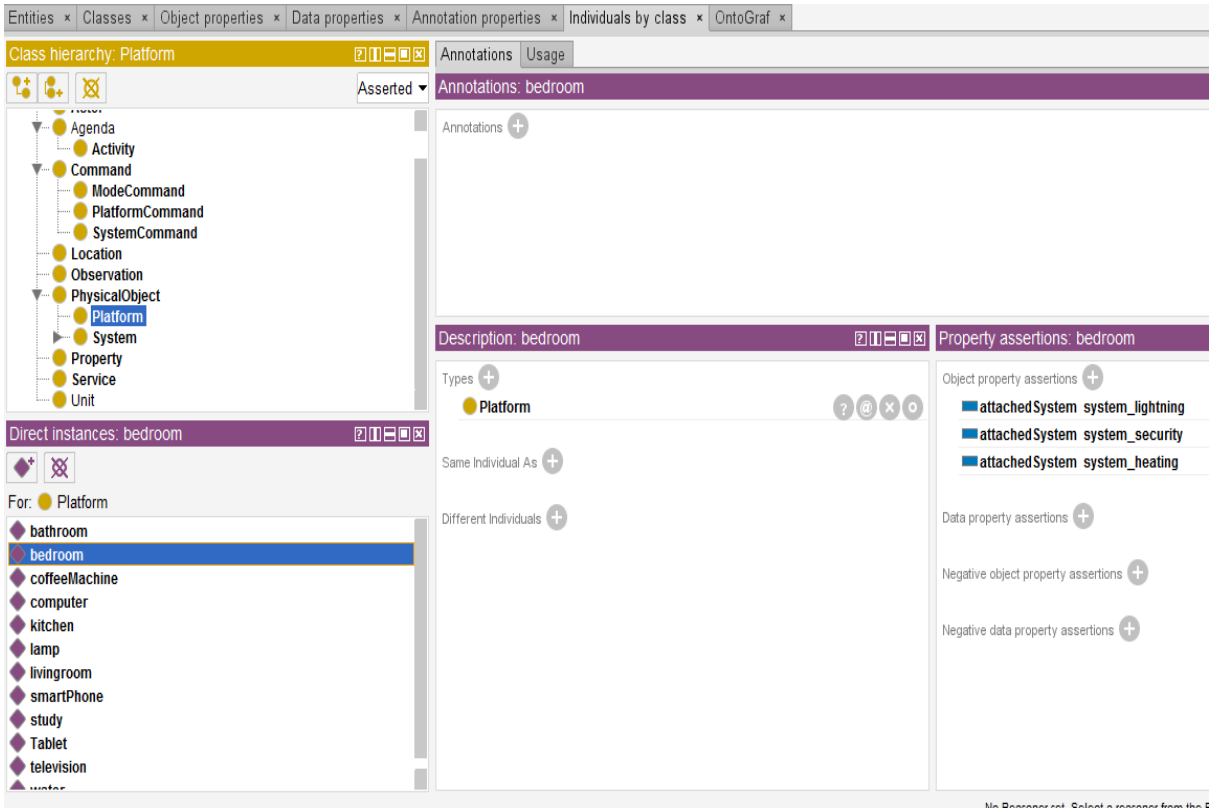

Figure 16 *Ajout d'individus au classe "Platform*

3.4 L'ontologie sous forme d'un graph

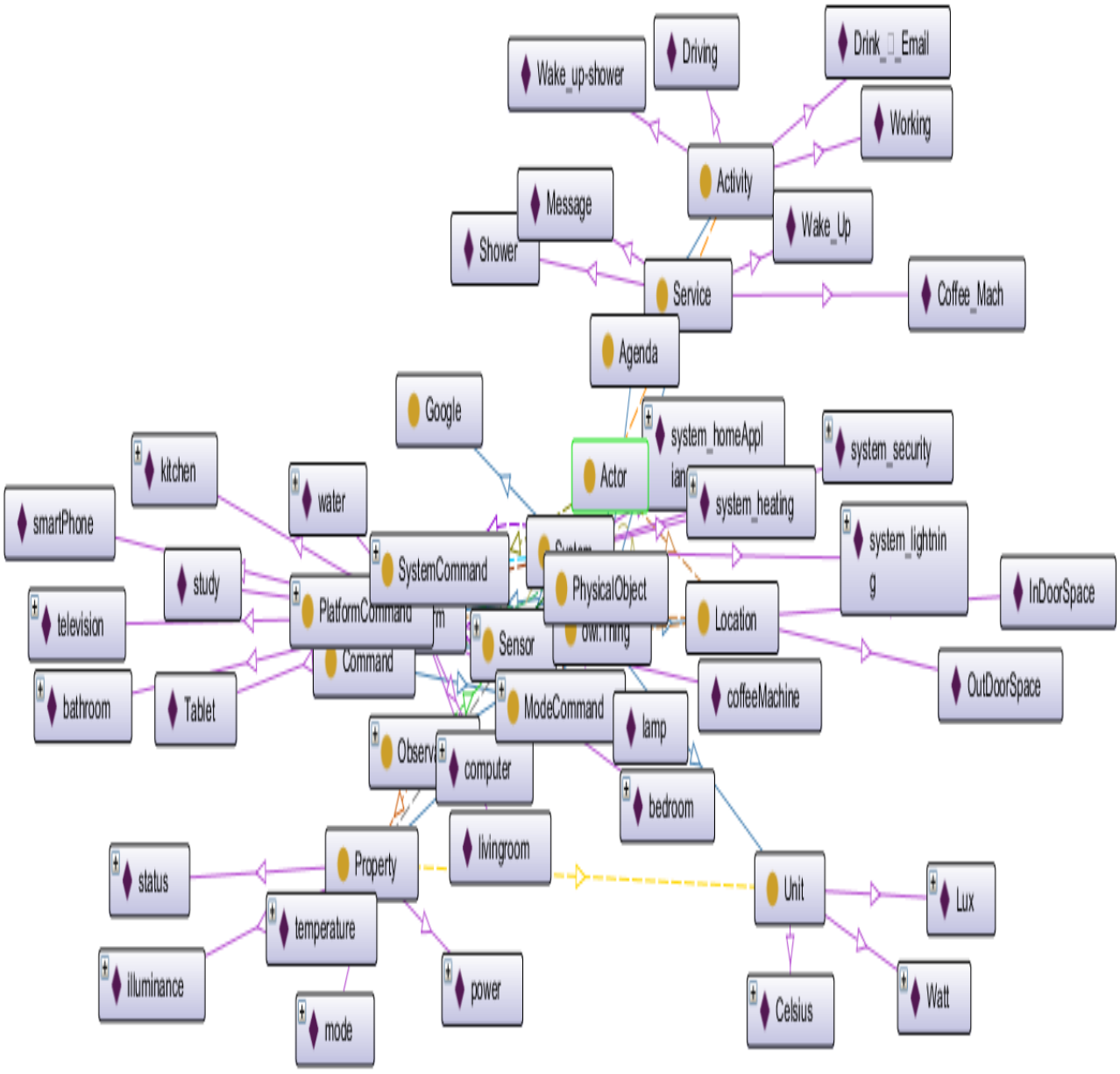

Figure 17 *l'ontologie de system sous forme d'un graphe.*

Conclusion

Dans ce chapitre, nous avons montré les différents outils et les différentes étapes de réalisation de notre travail.

Conclusion Général

Le nombre croissant d'appareils intelligents qui communiquent aujourd'hui sur Internet a conduit à de multiples appareils hétérogènes connectés dans un réseau. Par conséquent, il existe un besoin de mécanismes spécifiques pour parvenir à la composition de tous les services offerts par ces dispositifs, et ainsi exploiter le potentiel de l'IoT. Dans ce travail, nous proposons la conception d'une ontologie pour la composition de services IoT. Nous présentons une étude de cas pour l'orchestration des services dans un smart home.

Au cours de ce mémoire, nous avons présenté au début un état de l'art sur les concepts relatifs au notre sujet : l'internet des objets, ontologie et composition des services IoT. Afin de modéliser notre proposition, nous avons commencé par la conception de notre système en utilisant le formalisme UML, et pour la réalisation on a utilisé la plateforme Protégé-5.5.0. Cette dernière est présentée dans le dernier chapitre.

En effet, ce travail étant une œuvre humaine, n'est pas un modèle unique et parfait, c'est pour raisons que nous restons ouverts à toutes les critiques et nous sommes prêts à recevoir toutes les suggestions et les remarques tendant à améliorer d'avantage cette étude

Biographie

[1] walid djaafri. Composition des services dans l'internet des objets, Université Mohamed Khider – BISKRA, 2020.

[2] Frédéric LEMOINE Internet des Objets centré service autocontrôlé, École doctorale Informatique, Télécommunications et Électronique (Paris) 2019.

[3] Meftah ZOUAI. Une approche cloud computing basée IoT pour le smart House, Université Mohamed Khider – BISKRA 2020.

[4] Rabeb Saad. Modèle collaboratif pour l'Internet of Things (IoT) Université du Québec à Chicoutimi, 2016.

[5] Imad Saleh Internet. Des Objets (IdO) : Concepts, Enjeux, Défis et Perspectives 1 Laboratoire Paragraphe, Université Paris 8 2019.

[6] Hidouci Farid. Réalisation et implémentation d'une application à base de protocole MQTT dans IoT, 2019.

[7] Hamdi Ahmed. Ontologie de domaine pour un web sémantique destiné au e-Learning, 2011.

[8] Amir. La Génération Automatique des Ontologies à partir des Diagrammes de classes UML, 2017.

[9] Boukara Djehina, Utilisation des Ontologies pour l'Intégration d'Internet des Objets dans la gestion des processus métier, Université de 8 Mai 1945 – Guelma – 2019.

[10] R.BALI-H.HANI. Une approche d'annotation sémantique et légère pour minimiser la taille de données dans un environnement IoT, Université Echahid Hamma Lakhdar EL OUED 2018.

[11] S.Bechoua-S. Zertal. La proposition d'une approche pour la découverte et la sélection des services Cloud à base d'ontologie, Université Oum El Bouaghi, 2019.

[12] I.AOUDIA, S.BENHARZALLAH, L.KAHLOUL O.KAZAR LINFI Laboratory. A comparative analysis of IoT service composition approches, ACIT 2017.

[13] Y. BENAZZOUZ. Context discovery for the automatic adaptation of services in ambient intelligence, 2011.

[14] Service composition approaches for internet of things : à review Article in International Journal of Communication Networks and Distributed Systems · January 2019

[15] TOUMI SARRA Approche automatique de composition des services dans le cloud 2019

[16] Oucief Affef Construction d'une Ontologie pour Représenter La Sémantique des Langues Naturelles 2018

[17] [En ligne]. Available : https://protege.stanford.edu/about.php.

[18] [Enligne] Available : https://www.modelio.org/

[19] ACHOUR Raouf MAKHLOUFI Naima Authentificaiton dans l'IoT B´ejaia, Juillet 2017.

[20] Hadjadj Walid L'utilisation de N-Version de programmation pour la prise en charge des fautes dans un environnement IoT 2018.

Annexe

Extrait de l'ontologie de domaine de composition des services IoT d'un smart home en OWL

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
   <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
   <!ENTITY ssn "http://purl.oclc.org/NET/ssnx/ssn#" >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY dul "http://www.loa-cnr.it/ontologies/DUL.owl#" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>
<rdf:RDF xmlns="http://www.smarthome.com/ontology#"
     xml:base="http://www.smarthome.com/ontology"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:dul="http://www.loa-cnr.it/ontologies/DUL.owl#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:ssn="http://purl.oclc.org/NET/ssnx/ssn#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <owl:Ontology rdf:about="http://www.smarthome.com/ontology"/>
    <owl:ObjectProperty rdf:about="&ssn;attachedSystem">
        <rdfs:label>attached system</rdfs:label>
        <rdfs:isDefinedBy>http://purl.oclc.org/NET/ssnx/ssn</rdfs:isDefinedBy>
        <rdfs:domain rdf:resource="&ssn;Platform"/>
        <rdfs:range rdf:resource="&ssn;System"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:about="&ssn;hasSubSystem">
```

```
    <rdfs:label>has subsystem</rdfs:label>
    <rdfs:range rdf:resource="&ssn;Sensor"/>
    <rdfs:range rdf:resource="&ssn;System"/>
    <rdfs:domain rdf:resource="&ssn;System"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="&ssn;isPropertyOf">
    <rdfs:label>is property of</rdfs:label>
    <rdfs:domain rdf:resource="&ssn;Property"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="&ssn;observedBy">
    <rdfs:label>observed by</rdfs:label>
    <rdfs:domain rdf:resource="&ssn;Observation"/>
    <rdfs:range rdf:resource="&ssn;Sensor"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="&ssn;observedProperty">
    <rdfs:label>observerd porperty</rdfs:label>
    <rdfs:domain rdf:resource="&ssn;Observation"/>
    <rdfs:range rdf:resource="&ssn;Property"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="&ssn;observes">
    <rdfs:label>observes</rdfs:label>
    <rdfs:range rdf:resource="&ssn;Property"/>
    <rdfs:domain rdf:resource="&ssn;Sensor"/>
    <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="&ssn;onPlatform">
    <rdfs:label>on platform</rdfs:label>
    <rdfs:range rdf:resource="&ssn;Platform"/>
    <rdfs:domain rdf:resource="&ssn;Sensor"/>
```

<rdfs:domain rdf:resource="&ssn;System"/>

<rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>

</owl:ObjectProperty>

<owl:ObjectProperty
rdf:about="http://www.smarthome.com/ontology#commandToSystem">

<rdfs:range rdf:resource="&ssn;System"/>

<rdfs:domain rdf:resource="http://www.smarthome.com/ontology#ModeCommand"/>

<rdfs:domain
rdf:resource="http://www.smarthome.com/ontology#PlatformCommand"/>

</owl:ObjectProperty>

http://www.smarthome.com/ontology#hasActivity

<owl:ObjectProperty rdf:about=:hasActivity rdf:type owl:ObjectProperty ;

< rdfs:range rdf:resource="&ssn;Activity "/>;

<rdfs: domain rdf:resource="http://www.smarthome.com/ontology#Actor"/>

<owl:ObjectProperty rdf:about="http://www.smarthome.com/ontology#hasUnit">

<rdfs:label>has unit</rdfs:label>

<rdfs:domain rdf:resource="&ssn;Property"/>

<rdfs:range rdf:resource="http://www.smarthome.com/ontology#Unit"/>

</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="http://www.smarthome.com/ontology#mc_setMode">

<rdfs:label>mode platform</rdfs:label>

</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="http://www.smarthome.com/ontology#pc_start">

<rdfs:label>start platform</rdfs:label>

<rdfs:range rdf:resource="&ssn;Platform"/>

<rdfs:domain
rdf:resource="http://www.smarthome.com/ontology#PlatformCommand"/>

</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="http://www.smarthome.com/ontology#pc_stop">

<rdfs:label>stop platform</rdfs:label>

<rdfs:range rdf:resource="&ssn;Platform"/>

<rdfs:domain rdf:resource="http://www.smarthome.com/ontology#PlatformCommand"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.smarthome.com/ontology#sc_alarmOff">
<rdfs:label>alarm off</rdfs:label>
<rdfs:range rdf:resource="&ssn;Platform"/>
<rdfs:domain rdf:resource="http://www.smarthome.com/ontology#SystemCommand"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.smarthome.com/ontology#sc_alarmOn">
<rdfs:label>alarm on</rdfs:label>
<rdfs:range rdf:resource="&ssn;Platform"/>
<rdfs:domain rdf:resource="http://www.smarthome.com/ontology#SystemCommand"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.smarthome.com/ontology#sc_start">
<rdfs:label>start system</rdfs:label>
<rdfs:range rdf:resource="&ssn;System"/>
<rdfs:domain rdf:resource="http://www.smarthome.com/ontology#SystemCommand"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="http://www.smarthome.com/ontology#sc_stop">
<rdfs:label>stop system</rdfs:label>
<rdfs:range rdf:resource="&ssn;System"/>
<rdfs:domain rdf:resource="http://www.smarthome.com/ontology#SystemCommand"/>
</owl:ObjectProperty>

<owl:DatatypeProperty rdf:about="http://www.smarthome.com/ontology#hasSymbol">
<rdfs:label>has symbol</rdfs:label>
<rdfs:domain rdf:resource="http://www.smarthome.com/ontology#Unit"/>
<rdfs:range rdf:resource="&xsd;string"/>
<rdfs:subPropertyOf rdf:resource="&owl;topDataProperty"/>
</owl:DatatypeProperty>

```
    <owl:DatatypeProperty rdf:about="http://www.smarthome.com/ontology#mode">
        <rdfs:domain rdf:resource="http://www.smarthome.com/ontology#ModeCommand"/>
    </owl:DatatypeProperty>
    <owl:DatatypeProperty
rdf:about="http://www.smarthome.com/ontology#observationValue">
        <rdfs:label>observation value</rdfs:label>
        <rdfs:domain rdf:resource="&ssn;Observation"/>
    </owl:DatatypeProperty>
    <owl:DatatypeProperty rdf:about="http://www.smarthome.com/ontology#safeMax">
         <rdfs:domain rdf:resource="&ssn;Sensor"/>
        <rdfs:range rdf:resource="&xsd;int"/>
    </owl:DatatypeProperty>
    <owl:DatatypeProperty rdf:about="http://www.smarthome.com/ontology#safeMin">
        <rdfs:domain rdf:resource="&ssn;Sensor"/>
        <rdfs:range rdf:resource="&xsd;int"/>
    </owl:DatatypeProperty>
    <owl:DatatypeProperty rdf:about="http://www.smarthome.com/ontology#time">
        <rdfs:domain rdf:resource="&ssn;Observation"/>
        <rdfs:domain rdf:resource="http://www.smarthome.com/ontology#Command"/>
        <rdfs:range rdf:resource="&xsd;dateTime"/>
    </owl:DatatypeProperty>
    <owl:Class rdf:about="&ssn;Observation">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>
    <owl:Class rdf:about="&ssn;Platform">
        <rdfs:subClassOf rdf:resource="&dul;PhysicalObject"/>
        <rdfs:isDefinedBy>http://purl.oclc.org/NET/ssnx/ssn</rdfs:isDefinedBy>
    </owl:Class>
    <owl:Class rdf:about="&ssn;Property">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
```

```
</owl:Class>
<owl:Class rdf:about="&ssn;Sensor">
   <rdfs:subClassOf rdf:resource="&ssn;System"/>
</owl:Class>
<owl:Class rdf:about="&ssn;System">
   <rdfs:subClassOf rdf:resource="&dul;PhysicalObject"/>
</owl:Class>
<owl:Class rdf:about="&dul;PhysicalObject"/>
 <owl:Class rdf:about="http://www.smarthome.com/ontology#Command"/>
<owl:Class rdf:about="http://www.smarthome.com/ontology#ModeCommand">
   <rdfs:label>Mode Command</rdfs:label>
   <rdfs:subClassOf rdf:resource="http://www.smarthome.com/ontology#Command"/>
</owl:Class>
<owl:Class rdf:about="http://www.smarthome.com/ontology#PlatformCommand">
   <rdfs:label>Platform Command</rdfs:label>
   <rdfs:subClassOf rdf:resource="http://www.smarthome.com/ontology#Command"/>
</owl:Class>

<owl:Class rdf:about="http://www.smarthome.com/ontology#SystemCommand">
   <rdfs:label>System Command</rdfs:label>
   <rdfs:subClassOf rdf:resource="http://www.smarthome.com/ontology#Command"/>
</owl:Class>
<owl:Class rdf:about="http://www.smarthome.com/ontology#Unit"/>
<owl:NamedIndividual rdf:about="http://www.smarthome.com/ontology#Celsius">
   <rdf:type rdf:resource="http://www.smarthome.com/ontology#Unit"/>
   <hasSymbol rdf:datatype="&xsd;string">°C</hasSymbol>
</owl:NamedIndividual>
<owl:NamedIndividual rdf:about="http://www.smarthome.com/ontology#Lux">
   <rdf:type rdf:resource="http://www.smarthome.com/ontology#Unit"/>
```

<hasSymbol rdf:datatype="&xsd;string">lx</hasSymbol>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.smarthome.com/ontology#Watt">

<rdf:type rdf:resource="http://www.smarthome.com/ontology#Unit"/>

<hasSymbol rdf:datatype="&xsd;string">W</hasSymbol>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.smarthome.com/ontology#bathroom">

<rdf:type rdf:resource="&ssn;Platform"/>

<rdfs:label>Bathroom</rdfs:label>

<ssn:attachedSystem
rdf:resource="http://www.smarthome.com/ontology#system_heating"/>

<ssn:attachedSystem
rdf:resource="http://www.smarthome.com/ontology#system_lightning"/>

<ssn:attachedSystem
rdf:resource="http://www.smarthome.com/ontology#system_security"/>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.smarthome.com/ontology#computer">

<rdf:type rdf:resource="&ssn;Platform"/>

<rdfs:label>Computer</rdfs:label>

<ssn:attachedSystem
rdf:resource="http://www.smarthome.com/ontology#system_homeAppliances"/>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.smarthome.com/ontology#dishWasher">

<rdf:type rdf:resource="&ssn;Platform"/>

<rdfs:label>Dish Washer</rdfs:label>

<ssn:attachedSystem
rdf:resource="http://www.smarthome.com/ontology#system_homeAppliances"/>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.smarthome.com/ontology#electricity">

<rdf:type rdf:resource="&ssn;Platform"/>

<rdfs:label>Electricity</rdfs:label>

<ssn:attachedSystem
rdf:resource="http://www.smarthome.com/ontology#system_security"/>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.smarthome.com/ontology#water">

<rdf:type rdf:resource="&ssn;Platform"/>

<rdfs:label>Gas</rdfs:label>

<ssn:attachedSystem
rdf:resource="http://www.smarthome.com/ontology#system_security"/>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.smarthome.com/ontology#bedroom">

<rdf:type rdf:resource="&ssn;Platform"/>

<rdfs:label>Bedroom</rdfs:label>

<ssn:attachedSystem
rdf:resource="http://www.smarthome.com/ontology#system_heating"/>

<ssn:attachedSystem
rdf:resource="http://www.smarthome.com/ontology#system_lightning"/>

<ssn:attachedSystem
rdf:resource="http://www.smarthome.com/ontology#system_security"/>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="http://www.smarthome.com/ontology#livingroom">

<rdf:type rdf:resource="&ssn;Platform"/>

<rdfs:label>Livingroom</rdfs:label>

<ssn:attachedSystem
rdf:resource="http://www.smarthome.com/ontology#system_heating"/>

<ssn:attachedSystem
rdf:resource="http://www.smarthome.com/ontology#system_lightning"/>

<ssn:attachedSystem
rdf:resource="http://www.smarthome.com/ontology#system_security"/>

</owl:NamedIndividual>