

Rapport de stage



Master DataScale: Data Managment in a Digital World

Description et formalisation de règle



Tuteur :

* **Elie NAJM** (Professeur à Télécom ParisTech).
* **Bruno TRAVERSON** (ingénieur chercheur EDF R&D).

Tuteur enseignant **: Zoubida KEDAD**

**Réaliser par : OUHABI Adel**

SOMMAIRE

[**Table des Figures**](#_rbbofhi2hkt5) **3**

[**Remerciements**](#_od80bzga6umv) **4**

[**1 Introduction générale**](#_gjdgxs) **5**

[1.1 Entreprise d’accueil](#_30j0zll) 5

[1.1.1 EDF S.A](#_1fob9te) 5

[1.1.2 EDF R&D](#_tyjcwt) 7

[1.1.3 Le département ICAME](#_1t3h5sf) 8

[1.1.4 Laboratoire SEIDO](#_4d34og8) 9

[1.2 Contexte du stage](#_2s8eyo1) 10

[1.3 Sujet du stage](#_3rdcrjn) 11

[**2 Etat de l’art**](#_lnxbz9) **12**

[2.1 Le web sémantique](#_35nkun2) 12

[2.1.1 RDF, RDFS langages pour la description des ressources](#_44sinio) 12

[2.1.2 OWL, langage pour la description des ontologies](#_z337ya) 14

[2.1.3 SPARQL, langage pour l’interrogation des graphes RDF](#_1y810tw) 15

[2.1.4 SWRL, langage de règles](#_3whwml4) 17

[2.2 L’internet des objets](#_2bn6wsx) 18

[2.2.1 Les types d’objets connectés](#_qsh70q) 19

[2.2.1 Le Machine To Machine et l’IOT](#_3as4poj) 19

[3 Le contexte du smart Building](#_1pxezwc) 19

[**3 Solution**](#_49x2ik5) **21**

[3.1 Le web sémantique et l’internet des objets quel est le rapport ?](#_2p2csry) 21

[3.2 Description de l’ontologie](#_23ckvvd) 22

[3.3 Descriptions règles SWRL](#_vx1227) 27

[3.4 Orchestration de service et résolution des conflits](#_4f1mdlm) 29

[**4 Validation**](#_19c6y18) **30**

[4.2 Les outils utilisés](#_ma3a3tmykd2w) 30

[4.2.1 Outils pour les langages de règles](#_9eddhwrcj4rw) 30

[4.1.2 Outils pour les Ontologies](#_jq9cdhybezm2) 31

[4.3 Réalisation sur Freedomotic](#_yjihgvsfg09m) 33

[4.4 Scénarios de validation](#_3l18frh) 34

[**5 Conclusion générale**](#_p2vt00puq2t) **39**

[5.1 Bilan](#_9hz43828a4eu) 39

[5.2 Perspectives](#_z9fieaqep3c0) 39

[**Bibliographie**](#_ayuw2uq1p1l2) **40**

# Table des Figures

[Figure 1 Production d'électricité EDF 6](#_3znysh7)

[Figure 2 Organisation d'EDF 7](#_2et92p0)

[Figure 3 Chiffres clés d'EDF R&D 8](#_3dy6vkm)

[Figure 4 Les trois niveaux de modèles proposés dans l'architecture 10](#_17dp8vu)

[Figure 5 Couches du web sémantique 12](#_1ksv4uv)

[Figure 6 Représentation graphique d'un triplet RDF 13](#_2jxsxqh)

[Figure 7 Structure d'une requête SPARQL 15](#_4i7ojhp)

[Figure 8 Requête SPARQL simple 16](#_2xcytpi)

[Figure 9 Requête SPARQL avec PREFIX 16](#_1ci93xb)

[Figure 10-a Hétérogénéité entre les services 20](#_147n2zr)

Figure 10-b partage de capteur entre les services 20

[Figure 11 Apport du web sémantique pour l'IOT 21](#_3o7alnk)

[Figure 12 Concepts de base du smart building 22](#_ihv636)

[Figure 13 Hiérarchie du concept " ConnectedObject" 23](#_32hioqz)

[Figure 14 Hiérarchie du concept "Location" 24](#_1hmsyys)

[Figure 15 Hiérarchie du concept "Service" 25](#_41mghml)

[Figure 16 Hiérarchie du concept "Settings" 25](#_2grqrue)

[Figure 17 Hiérarchie d'orchestration 30](#_2u6wntf)

[Figure 18 Interface SWRLTAB 31](#_nmf14n)

[Figure 19 Outil Protégé 32](#_46r0co2)

Figure 20 Interface Freedomotic 32

[Figure 21 Déroulement d'un scénario d'un service qui a pour but la gestion de la lumière 3](#_206ipza)3

[Figure 22 Déroulement d'un scénario d'un service qui a pour but de baisser la température 3](#_4k668n3)4

Figure 23 Déroulement d'un scénario d'un service qui a pour but l'économie d'énergie avec une lampe 35

[Figure 24-a Exécution des services sans orchestration 36](#_2zbgiuw)

[Figure 24-b Exécution des services sans orchestration sur Freedomotic 36](#_2zbgiuw)

[Figure 25-a Exécution des services avec orchestration 3](#_1egqt2p)7

[Figure 25-b Exécution des services avec orchestration sur Freedomotic 3](#_1egqt2p)7

[Figure 26-a Exécution des services avec résolution des conflits 3](#_3ygebqi)8

Figure 26-b Exécution des services avec résolution des conflits sur Freedomotic 39

# 

# 

# 

# 

# 

# Remerciements

Il n'est jamais facile pour un étudiant de trouver un stage, c'est pourquoi je remercie EDF R&D et Telecom parisTech de m'avoir accueilli durant ces six mois.

Je tiens à remercier tout particulièrement Mr Bruno TRAVERSON et Mr Elie NAJM qui m'ont accordé leur confiance, leur soutien, leurs conseils et leur disponibilité. Merci également à toute l'équipe de l'entreprise, car chacun d'entre vous a su trouver un peu de temps pour m'aider dans mes missions.

Faire mon stage de dernière année dans votre entreprise a été un plaisir, j'ai pu apprendre beaucoup grâce à vous, et j'ai surtout été conforté dans mon projet professionnel, ce qui est un aboutissement de mon cursus universitaire.

# **1 Introduction générale**

## 1.1 Entreprise d’accueil

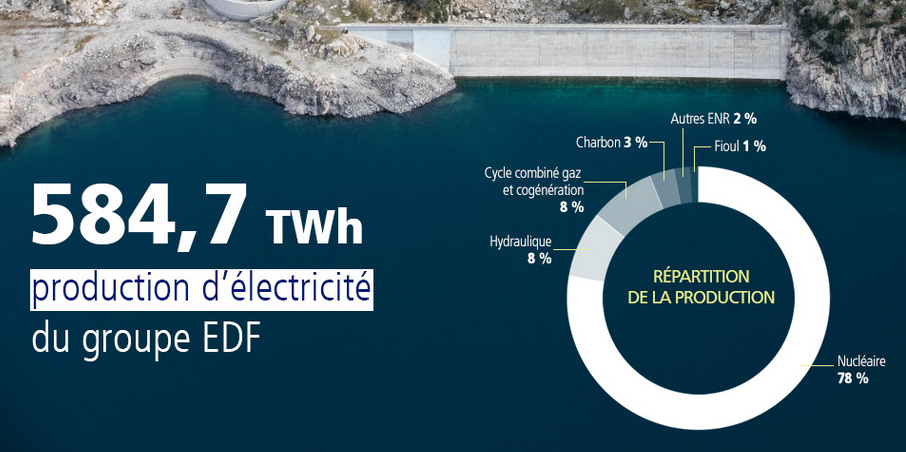
### 1.1.1 EDF S.A

Autrefois connu par sa nomination originale «Electricité de France », EDF alimente plus de 36 million de foyers en électricité. Pour cause, EDF est considéré comme l’un des leaders mondiaux de la production à la commercialisation d’électricité avec 78 réacteurs nucléaires.

EDF ce n’est pas seulement le nucléaire mais aussi le premier producteur d’énergies nouvelles et renouvelables en Europe avec 5000 éoliennes en France.

Pas moins de 154.845 collaborateurs d’EDF dans le monde sont mobilisés afin de répondre en permanence aux attentes et aux besoins en électricité des clients de l’entreprise, tant industriels que particuliers. EDF veille à assurer une sécurité énergétique

maximale ainsi qu’à préparer l’avenir en inventant l’électricité de demain afin de réduire les émissions de CO2 et contribuer à la bonne santé de la planète. Actuellement produit 584,7 TWh dont 88% sans CO2 (Figure1).



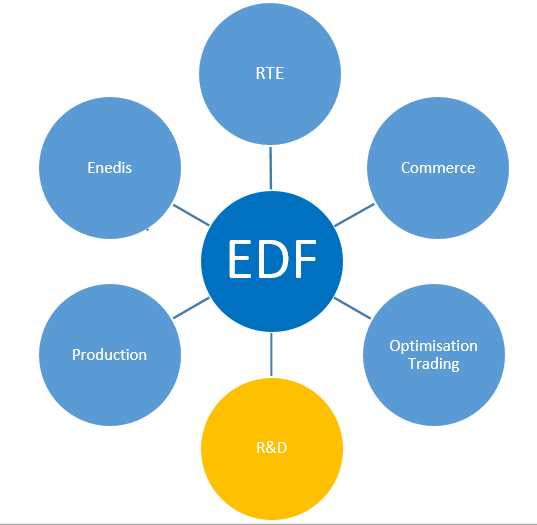
*Figure 1 Production d'électricité EDF*

La fonction principale d’EDF est la production d’électricité. Pour cela, le groupe exploite différentes énergies à savoirs le nucléaire, le thermique à flamme, l’hydraulique, l’éolienne, le solaire, l’énergie marine, l’énergie biomasse et géothermique.

Certaines méthodes sont régulièrement au cœur de l’actualité, telle que le nucléaire alors que d’autres se dessinent plutôt sur l’avenir à l’image de la biomasse. Du fait que l’électricité se stocke très mal pour de gros volumes, EDF doit être capable de fournir la juste quantité d’électricité en fonction de la demande, et ce au meilleur prix pour ses 27,2 millions de clients en France et environ 37 millions dans le monde.

Enfin, EDF, c’est aussi toute une gamme de services et d’offres pour particuliers et professionnels relatifs à l’électricité et à sa consommation.

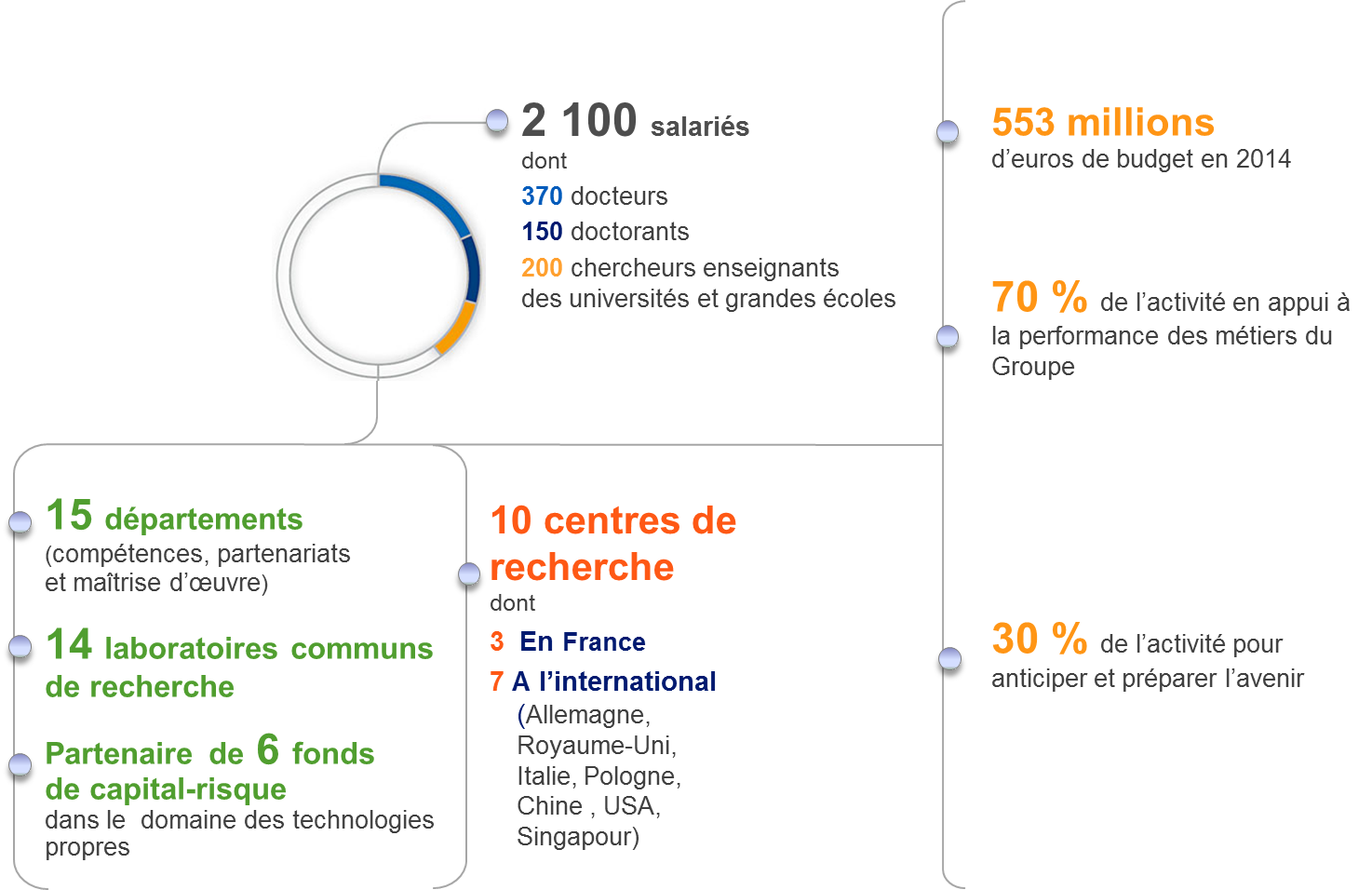
Le groupe est composé de plusieurs branches dont la R&D comme le montre la figure suivante.



*Figure 2 Organisation d'EDF*

### 1.1.2 EDF R&D

EDF R&D a pour missions principales de contribuer à l'amélioration de la performance opérationnelle du groupe, d'identifier et de préparer les relais de croissance à moyen et long terme. Elle est organisée autour de dix sites, dont trois situés en France, en région parisienne. Achevées sur le site de Chatou après la Seconde Guerre mondiale, les premières infrastructures d’EDF R&D accueillent le Laboratoire National d’Hydraulique (LNH) dont les travaux ont notamment permis le développement de l’hydroélectricité et plus particulièrement les grands barrages dans un contexte de forte pénurie d’énergie. Avec l’évolution des moyens de production et des besoins, les recherches sur le thermique, le nucléaire, les réseaux, l’environnement et la simulation numériquesont venues compléter les domaines d’expertise de la R&D avec l’ouverture de deux autres centres. Achevé en mars 2016, EDF Lab Paris-Saclay associe le plus grand centre de recherche et développement d’EDF (EDF R&D) au premier centre de formation professionnelle à l’échelle européenne (Campus formation). Il concentre en un même lieu des compétences d’excellence et des moyens d’essais sur des sujets stratégiques pour l’avenir du Groupe EDF : la production d’électricité décarbonée, les systèmes électriques du futur et les nouveaux services énergétiques (Figure 3).



*Figure 3 Chiffres clés d'EDF R&D*

### 1.1.3 Le département ICAME

Le département ICAME« Innovation Commerciale, Analyse desMarchés et de leur Environnement » est l’un des départements de la R&D d’EDF. Le budget du département est estimé à environ 25 millions d’euros. La direction Commerce, à qui ICAME dédie 80% de son activité, finance à hauteur de 14 millions d’euros. Le but est de proposer des services se différenciant de ceux de la concurrence, dans le cadre d'une relation commerciale en phase avec les évolutions de la société et les attentes de la clientèle. Dans ce cadre ICAME mène des activités de R&D principalement conduites pour EDF Commerce mais aussi pour l'ensemble des entités du Groupe. Le département est composé d’une centaine de chercheurs et d’ingénieurs, d’une dizaine d’agents de maîtrise ainsi que d’une dizaine de doctorants.

Le personnel est réparti en trois pôles principaux : le pôle Sciences Humaines et Sociales ; Statistiques, Fouille et Valorisation des Données ; et Méthodes et Solutions pour l’élaboration des nouvelles offres.

Trois missions principales sont confiées au département ICAME :

* Proposer des visions de la consommation des clients dans le système énergétique à des horizons et des temps différents,
* Co-construire de nouvelles offres de fournitures et de services, notamment avec la Direction Commerce et les filiales commerciales à l’étranger.
* Accompagner la Direction Commerce dans l’évolution des métiers de commercialisation en éclairant les sujets relatifs aux nouvelles attentes des clients, aux nouveaux canaux de commercialisation et aux performances de la relation client.

Le département participe à renforcer l’acceptabilité sociale des projets de développement et contribue à tirer profit des données massives et complexes pour l’ensemble des directions opérationnelles.

### 1.1.4 Laboratoire SEIDO

La R&D d’EDF et Télécom ParisTech collaborent depuis 2012 dans le cadre d’un laboratoire commun de recherche dont les travaux portent sur la thématique de l’Internet des Objets et la Cybersécurité pour les systèmes électriques. Ce laboratoire constitue une occasion de réunir les spécialistes de domaines disciplinaires hétérogènes (sécurité, réseaux, Internet, monde énergétique, réseaux électriques, etc.) autour de finalités industrielles qui les rassemblent. Il apporte des avancées en matière de solutions de sécurisation des systèmes d’information, de contrôle et de commande pour les objets connectés. Il contribue à mieux observer, contrôler et commander les objets du système électrique, plus vite, plus sûrement et en tenant compte de leur contexte d’usage, en respectant la confidentialité des échanges. Son enjeu est de préparer et faciliter le déploiement de services de gestion de la demande énergétique et d’efficacité énergétique s’appuyant sur l’interopérabilité d’objets énergétiques et ainsi contribuer à assurer la cohérence de l’ensemble du système.

Le laboratoire SEIDO a pour ambition de préparer le système électrique aux principaux enjeux qui accompagnent le développement des Smart Grids grâce à un programme de recherche couvrant les domaines de l’Internet des Objets et de la Cybersécurité.

Le système électrique repose aujourd’hui sur une architecture centralisée, reliant une multitude de consommateurs à quelques grands sites de production. Il évolue en accueillant de plus en plus de nouvelles sources d’énergie renouvelables et intermittentes ainsi qu’une micro-production décentralisée. Les consommateurs quant à eux, doivent devenir des acteurs à part entière de ce système produisant de l’énergie et contrôlant finement leur consommation.

Face à ces évolutions, le réseau de distribution, en particulier, doit s’adapter ; devenir plus flexible et plus réactif afin de maintenir des conditions de fonctionnement correctes et sûres, mais aussi afin de tirer parti au mieux du potentiel que représentent ces mutations pour résoudre plusieurs défis :

* assurer une gestion de la demande d’énergie capable de faire face aux impératifs environnementaux;
* assurer la stabilité du réseau malgré la part croissante de la production intermittente;
* proposer à l’usager de piloter ses équipements de manière à assurer son confort, au meilleur coût, dans le cadre des évolutions progressives des réglementations thermiques (RT2012, RT2020, etc.).

Pour répondre à ces questions, le laboratoire réunit des chercheurs du monde académique et du monde industriel disposant de compétences complémentaires. Télécom ParisTech apporte son expertise dans le domaine des réseaux et des télécommunications, des systèmes d’information, et de la sécurité informatique. La R&D d’EDF implique, en France et en Allemagne, des spécialistes du comptage d’énergie, des solutions électriques en aval du compteur, du véhicule électrique et de la relation clients. Ces collaborations s’illustrent au travers de plusieurs thèses co-encadrées entre les deux institutions et de manifestations.

Les principaux objectifs du laboratoire sont :

* de contribuer à la conception d’architectures informatiques distribuées, à large échelle, capables de surveiller et de gérer les systèmes de distribution de l’électricité jusqu’aux équipements terminaux en incluant la production locale et le stockage décentralisés;
* d’amener les résultats d’études sur des plateformes de simulation et de démonstration;
* de concourir à l’obtention de nouveaux concepts, de nouveaux standards de définition de nouveaux équipements, de nouveaux logiciels et développer ainsi un patrimoine intellectuel commun.

## 1.2 Contexte du stage

Le contexte de mon stage s’inscrit dans le cadre des travaux réalisés dans la thèse de Rayhana BAGHLI au sein du laboratoire SEIDO **[EDOC 2016]**. L’architecture proposée dans cette thèse permet de représenter les services mis en place dans un Smart home selon trois niveaux (Figure 4) : Sémantique (connaissance), Artefacts et Ressources. Le niveau sémantique spécifie le comportement du système par un ensemble de règles qui sont vérifiées à chaque évolution de son environnement. Le niveau d’artefacts représente les états des ressources sous forme de graphes à états/transitions. Le niveau des ressources assure la correspondance entre les ressources et leurs implémentations sous forme d’objets connectés.



*Figure 4 Les trois niveaux de modèles proposés dans l'architecture*

## 1.3 Sujet du stage

L’architecture à trois niveaux proposés dans la thèse peut s’adapter à une maison, un bâtiment, un quartier ou encore une ville. Dans le cadre de mon stage, il m’a été proposé de travailler sur le niveau sémantique de l’architecture dans le contexte du smart building.

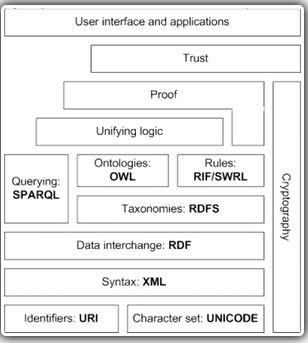
Le niveau sémantique tel qu’il est mentionné dans la thèse décrit les objets connectés et les services offerts par ces objets. Les services décrivent des règles déclaratives qui s’appuient sur des états du système.

L’objectif de mon stage est d’utiliser le web sémantique pour décrire et formaliser le format des règles utilisé dans le niveau sémantique de la thèse dans le contexte de l’internet des objets et du smart building, ainsi que d’apporter une solution pour orchestrer les différents services et éviter les conflits entre les services et enfin apporter des exemples de validations avec des scénarios réalisés sur le logiciel libre Freedomotic.

# **2 Etat de l’art**

## 2.1 Le web sémantique

Le web sémantique **[WebS]** est une extension du web où on peut donner et comprendre la sémantique de l’information sur le web. Avec le web sémantique, le web devient intelligent car, en donnant de la sémantique aux données, on pourra affiner les recherches et on pourra faire de l’apprentissage sur ces données pour pouvoir donner des résultats beaucoup plus précis et qui ont du sens pour l’utilisateur. La figure 5 nous montre les différentes couches qui composent le web sémantique où chaque couche utilise et exploite les capacités des couches qui se situent en-dessous d'elle.

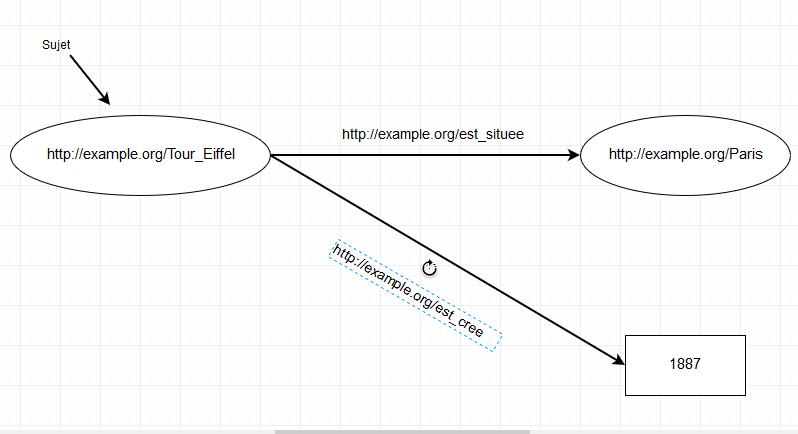


*Figure 5 Couches du web sémantique*

Différentes technologies sont à disposition du web sémantique entre autre le langage RDF ([Resource Description Framework](https://fr.wikipedia.org/wiki/Resource_Description_Framework) ) qui en est la base et le langage d’interrogation SPARQL qui permet de requêter les graphes RDF.

### 2.1.1 RDF, RDFS langages pour la description des ressources

Le langage RDF se base sur le langage XML et permet de décrire les ressources sur le web sous forme de graphe de triplets (sujet, prédicat, objet). Le sujet (ou ressource) est une ressource à décrire pouvant être n’importe quoi, le prédicat (ou propriété) est une propriété qu’on peut appliquer sur ce sujet pour décrire la ressource, l’objet représente la valeur de la propriété et peut être soit une donnée de type de base (int , string …) ou une autre ressource. La figure 6 nous montre une représentation graphique d’un triplet où les ellipses représentent des ressources, la forme rectangulaire représente une donnée (ou littéral) et les flèches orientés représentent des propriétés. Le sujet, et l'objet dans le cas où c'est une ressource, peuvent être identifiés par une URI (Uniform Resource Identifier) ou être des nœuds anonymes (une ressource qui n’a pas d’URI ou qui n’a pas d’identification). Le prédicat est nécessairement identifié par une URI.



*Figure 6 Représentation graphique d'un triplet RDF*

Certaines situations ne sont pas modélisées par RDF seul, il est parfois intéressant de représenter des relations plus complexes comme les sous-classes par exemple ou donner plus de sémantique à la description de la ressource. Le langage RDFS (RDF Schema)fournit des moyens pour représenter de tels cas, avec l’introduction de nouvelle primitives comme rdfs:subClassOf, rdfs: range ou rdfs: domain. Le langage RDFS permet de décrire des vocabulaires RDF. Il offre un modèle qui permet de donner plus de sémantique et du sens aux propriétés d’une ressource et permet de définir des contraintes sur les objets du triplet en associant un domaine (rdfs:domain) et co-domaine (rdfs:range) au sujet. Cependant, ce langage permet juste d’écrire des ontologies légère (c.-à-d. une ontologie ne comportant que des hiérarchies de types ou de propriétés) ne permet pas de raisonner et de faire des déductions sur la base de connaissances décrites grâce à RDF/RDFS. Pour pallier ce problème, le langage OWL (Web Ontology Language) est apparu.

### 2.1.2 OWL, langage pour la description des ontologies

Le schéma RDF comprend les primitives pour spécifier un vocabulaire qui comprend des classes, des propriétés, et les relations de sous-classe et de sous-propriété. Pour caractériser plus précisément le vocabulaire, on doit ajouter plus d'expressivité dans la spécification, par exemple les restrictions de cardinalité, les opérateurs booléens, notion de classe équivalente, de propriété équivalente, de différence de deux ressources, de contraire, de symétrie, de transitivité. Le langage OWL apporte toutes ces notions dans le vocabulaire RDF. Ce vocabulaire devient ce que nous pouvons appeler par une ontologie lourde.

Le langage OWL, qui est basé sur RDF permet de définir des ontologies. OWL est composé de trois sous-langages qui offrent une expressivité croissante **[OWL]**

**OWL Lite** est la version légère du langage OWL. Il utilise que certaines des fonctionnalités d’OWL. Par exemple, il permet d’exprimer les hiérarchies ou les restrictions de cardinalité (seulement 0 ou1) mais ne permet pas d’exprimer l’union ou l’intersection par exemple. Le langage *OWL Lite* est destiné aux utilisateurs ayant besoin principalement d'une hiérarchie de classifications et de contraintes simples

**OWL DL (OWL Description Logics)**  permet une expressivité plus importante qu’OWL Lite. Il se base sur la logique descriptive qui lui confère la capacité de raisonnement et d’inférence et garantit la complétude (toutes les inférences valides peuvent être déduites) et la décidabilité des inférences (le calcul des inférences prend un temps fini).

**OWL Full** est la version la plus complexe d’OWL et permet le plus haut niveau d’expressivité. En contrepartie, il ne garantit pas la décidabilité du calcul sur l’ontologie

Il existe un lien hiérarchique entre ces trois sous langage, une ontologie valide exprimée en OWL Lite est également valide en OWL DL et toute ontologie valide exprimée en OWL DL est aussi une ontologie OWL Full valide.

Toutefois OWL n’est pas capable de déduire toutes les relations comme, par exemple la relation *est oncle de.*  Pour pouvoir le faire, OWL peut être associé à des règles d’inférence.

Le langage OWL est composé de trois parties : les classes, les propriétés et les instances

**Les classes**

Une classe est comme un ensemble d’instances ou d’individus qui partage les mêmes descriptions et caractéristiques. Une classe peut être vue comme un concept ou comme une table dans le domaine des bases de données relationnelle. En OWL, toute les class que les utilisateurs vont définir seront des sous-classes de la super classe OWL:Thing.

**Les propriétés**

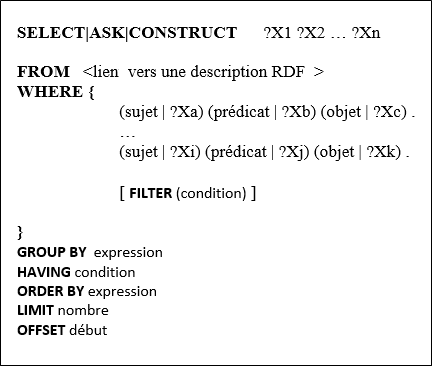
Les propriétés permettent de définir des faits et des relations entre ces classes. En langage OWL, on définit des propriétés en donnant leur domaine (domain) et co-domaine (range). Deux sortes de propriétés sont à définir. D’une part, les propriétés de type de données (owl:DataTypeProperty) lient un concept (domain) au domaine de valeur (range) que peut prendre cette propriété (ex : la propriété *hasName* a pour domaine *Personne*, et pour co-domaine *xsd :string)*. D’autre part, les propriétés d’objet (owl:ObjectProperty) lient des concepts à un autre concept (ex : la propriété *auteur* a pour domaine le concept *Livre*, et pour co-domaine le concept *Personne)*. Les propriétés peuvent aussi être organisées hiérarchiquement.

**Les instances**

Les instances sont des individus pour qui nous allons déclarer leur appartenance à une ou plusieurs classes. Ces individus respectent les caractéristiques définies par les propriétés et toutes les restrictions et la hiérarchie des classes auxquelles ces individus appartiennent.

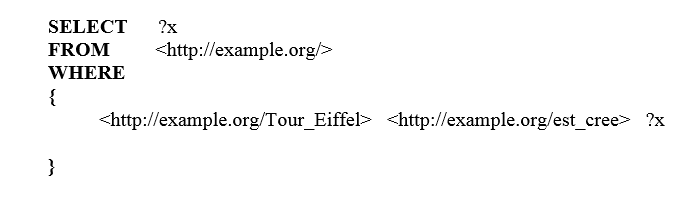
### 2.1.3 SPARQL, langage pour l’interrogation des graphes RDF

SPARQL **[SPARQL]** est un langage développé par le W3C pour interroger des graphes RDF.la figure 7 nous montre la structure d’une requête SPARQL qui est proche de la structure d’une requête SQL. Leur différence réside dans le fait que SQL interroge des données d’une base de données et SPARQL interroge des descriptions RDF



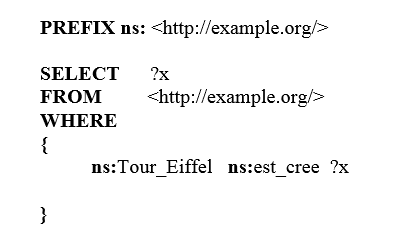
*Figure 7 Structure d'une requête SPARQL*

Les éléments recherchés dans une requête SPARQL ( ?X1 …) sont appelés des graphes patterns. Une requête n’a pas comme seul but seulement la recherche de résultat, mais une requête avec *ASK* permet de tester s’il existe au moins une solution qui correspond à un graphe pattern, comme il est possible de construire un nouveau graphe pattern ou triplet avec *CONSTRUCT*. La figure 8 nous montre une requête SPARQL simple qui retourne l’année de création de la tour Eiffel.



*Figure 8 Requête SPARQL simple*

Un élément important existe dans le langage SPARQL. Il permet de spécifier des préfixes (PREFIX) qui vont simplifier l’écriture des requêtes en préfixant les espaces de nommages. Ces derniers comprennent tous les éléments et les déclarations qui sont compris dans le vocabulaire d’un document RDF. Devant chaque propriété utilisée dans un triplet, on doit indiquer le préfixe suivi de la propriété comme nous le montre la figure 9 :  
« rdf:proprieteRDF » ou « ont:proprieteOntologie»



*Figure 9 Requête SPARQL avec PREFIX*

### 2.1.4 SWRL, langage de règles

**SWRL** est un langage de règles pour le web sémantique. Ce langage se base sur des sous-langages de OWL (OWL DL et LITE) et sur le langage de balisage RuleML. SWRL comprend une syntaxe abstraite de haut niveau pour les règles de type Horn. Ces règles sont des règles DATALOG avec des prédicats unaires pour décrire les classes et les types de données, et des prédicats binaires pour les propriétés **[SWRL].** Exemple de prédicat: **X** est de type Homme → en SWRL : Homme ( ?X)

**X1** est âgé de **X2** ans →en SWRL hasAge( ?X1, ?X2)

Note: Les variables en SWRL doivent être préfixées par « ? »

**Forme d’une règle en SWRL** : antécédent(Body) -> conséquence (head)

atome (^ atome)\* -> atome (^ atome)\*

Les règles en SWRL sont sous la forme d'une implication entre une conjonction antécédente (body) et une conjonction conséquente (head). Cette forme implique que la conséquence (head) doit être évaluée à vrai lorsque l’antécédent est évalué à vrai. L’antécédent et la conséquence sont constitués d’atomes sous la forme **C(x)**, **P(x,y)**, où **C** c’est une description OWL, **P** est une propriété OWL et x,y sont soit des variables, soit des individus OWL, soit des valeurs de données OWL. Tout comme le SWRL intègre diverses fonctions qui sont utilisables via le préfixe « swrlb:». Elles permettent entre autre de faire des comparaisons entre un data Type et une valeur par exemple (ex : swrlb:greaterThan(?y,18) ) ou des opérations sur des chaines de caractères (swrlb:contains, swrlb:matches …).

Exemple:

**Si** x est un Homme **alors** x est une Personne

* **En SWRL :** Homme (?x) -> Personne(?x)

**Si** x est une Femme **alors** x est une Personne

* **En SWRL :** Femme (?x) -> Personne(?x)

**Note**: Le langage SWRL ne permet pas de traiter une disjonction d’atome dans le body. Par exemple on peut pas écrire**:** Homme (?x) OR Femme (?x) -> Personne(?x).

**Si** x est une Personne et que l’âge y de x est supérieur à 18 ans **alors** x est Adulte

* En SWRL : Personne(?x) hasAge(?x,?y) swrlb:greaterThan(?y,18)->Adulte(?x)

## 2.2 L’internet des objets

Au cours des cinq dernières années, les technologies des objets connectés et de l’Internet des objets(ou IOT pour Internet Of Things) ont évolué à une vitesse significative, connectant un nombre conséquent d’objets connectés. Selon le cabinet de recherche américain Gartner, il y aura, fin 2017, 8,4 milliards d’objets connectés à Internet et il prévoit leur nombre à plus de 20 milliards en 2020. Ils en existent de plus en plus de toute sorte dans des domaines différent tel que la logistique (entrepôts entiers entièrement automatisés d’Amazon), la pharmaceutique, la santé, la domotique ou encore l’énergie. Du fait du nombre conséquent d’objets connectés, ces dernier participe grandement à la production de données dans le cadre du BigData. Le volume de données total pourrait atteindre 44.000 milliards de giga-octets en 2020, soit 35% du trafic Internet mondial (source IDC). Étant donné que l’IoT est un domaine récent et en plein essor, cela engendre quelques défis à relever :

* Traitement, analyse et interprétation des données : la génération de masses de données ad-hoc sont peu réutilisables du fait de leur hétérogénéités et cela nécessite un traitement particulier.
* L’Interopérabilité : pour que les objets connectés puissent communiquer, il faut une standardisation d’un langage de communication. Cela nécessite d’établir des normes de communication entre les technologies et les fabricants.
* Sécurité des données collectées : faire transiter et stocker les informations sensibles de façon sécurisée et l’un des principaux problèmes de l’IoT.

### 2.2.1 Les types d’objets connectés

Nous pouvons distinguer deux types d’objets: les capteurs, les actionneurs. Les capteurs sont destinés à collecter et transmettre les informations de notre environnement, et les actionneurs permettent de déclencher une action à distance.

### 2.2.1 Le Machine To Machine et l’IOT

L'intérêt des objets connectés est le fait qu’ils interagissent avec leur environnement et fournissent des informations. La communication entre ces objets ne nécessite pas forcément l’intervention de l’Homme mais peut se faire de machine à machine (M2M). Pour permettre cette communication entre les objets connectés, sept organismes de normalisation se sont mobilisé pour créer en 2012 le standard OneM2M [**oneM2M**]. Ce dernier propose une norme unique pour les communications M2M.

## 3 Le contexte du smart Building

Un smart building est un bâtiment équipé de divers objets connectés, qui anticipe et répond aux besoins de ses occupants en essayant de gérer de manière optimale leur confort et leur sécurité toute en visant à optimiser la consommation. Les bâtiments d’EDF Lab à Saclay offre un environnement idéal pour un smart building du fait du nombre d'objets connectés qui y sont déployés. Ces derniers sont gérés par le système de Gestion Technique du Bâtiment (GTB).

**Description des équipements**

Les installations qui sont placées sous contrôle de la GTB sont les suivantes :

* La production de chaud/froid, la ventilation et la climatisation,
* L'électricité courants forts et les groupes électrogènes,
* L’électricité courante faible (armoires divisionnaires, éclairage…),
* La gestion des espaces au niveau climatisation, éclairage et store,
* Les alarmes techniques : machinerie ascenseur, plomberie,...

Pour pouvoir réaliser ces tâches plusieurs capteurs et actionneurs sont mis en place dans le bâtiment. Le tableau suivant montre la liste des objets connectés déployés dans le bâtiment.

|  |  |  |
| --- | --- | --- |
| Object | Type | Fonctionnalité |
| Centrale Météo | capteur | Mesure les données météo à proximité du bâtiment  • Vitesse et direction du vent  • Humidité relative et la température  • Rayonnement solaire  • Pression atmosphérique |
| Détecteur de présence | capteur | Détecteur de présence |
| Capteur de luminosité | capteur | Calcule la luminosité de la pièce |
| Capteur de condensation | capteur | Détecte la condensation |
| Sonde d’ambiance | capteur | mesure la température d’une pièce |
| circuits d’éclairage | Actionneur | regroupe des lampes pour apporter la lumière |
| Ballast Dali | Actionneur | composant électronique utilisé pour réduire la chaleur dans un circuit électrique |
| Store | Actionneur | le store est utilisé pour apporter la lumière |
| ventilo-convecteur | Actionneur | utilisé pour la climatisation de la pièce |
| Ouvrants de Ventilation Motorisé (OVM) | Actionneur | gère l'ouverture des OVM en fonction de la température dans la pièce |
| Batterie froide | Actionneur | elle s’active en cas de demande de froid et se ferme en cas contraire |
| Dalle active | Actionneur | sert à gérer la température de la pièce |
| trame | Actionneur | gère 16 éléments de trames, chaque élément de trame possède 1 ou plusieurs ballasts qui seront gérés  comme un seul et unique groupe luminaire |
| Télécommande | Actionneur | gère les éléments d’une trame |

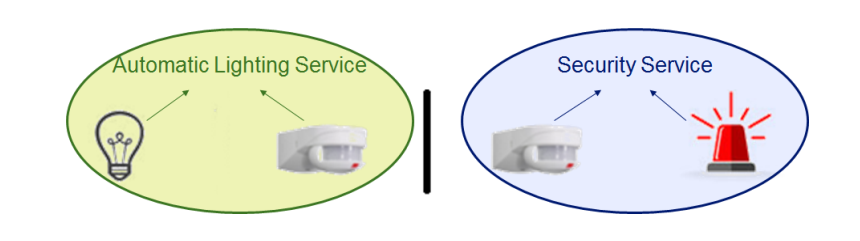
*Tableau 1 Type d'objets connectés du bâtiment Opale*

# 

# **3 Solution pour la formalisation des règles**

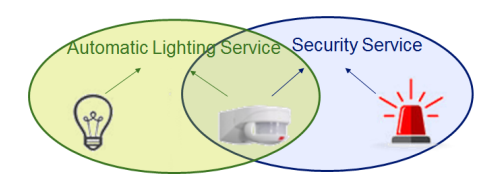
## 3.1 Le web sémantique et l’internet des objets quel est le rapport ?

De nous jour les constructeurs d’objets connectés offre des services qui ne sont compatible qu’avec leur propre objets et qu'il n’y a pas interopérabilité entre les objets d’autres constructeurs, comme le montre la figure 10-a.



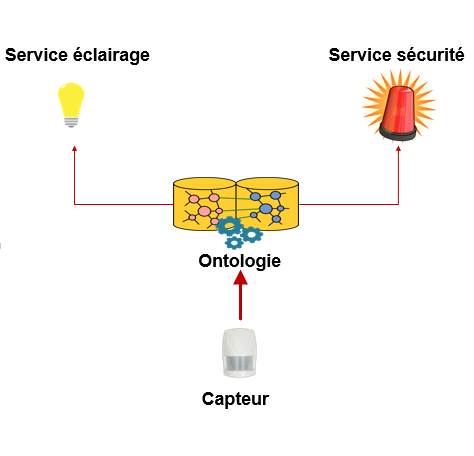
*Figure 10-a Hétérogénéité entre les services*

L’objectif est de fournir des services qui sont indépendants des objets connectés et de permettre le partage d’objets connectés entre les services comme nous le montre la figure suivante.



*Figure 10-b partage de capteur entre les services*

Dans cette partie, nous allons présenter l’approche apportée par le web sémantique pour résoudre le problème présenté précédemment (à savoir l’interopérabilité entre les différents objets) comme nous le montre la figure suivante

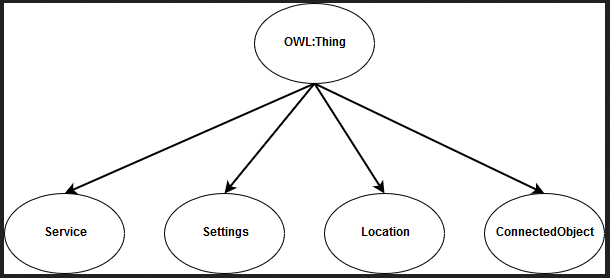


*Figure 11 Apport du web sémantique pour l'IOT*

### 3.2 Description de l’ontologie

Dans cette partie, nous allons vous présenter l’ontologie qui a servi de décrire notre environnement et les objets connectés qui s’y trouvent.

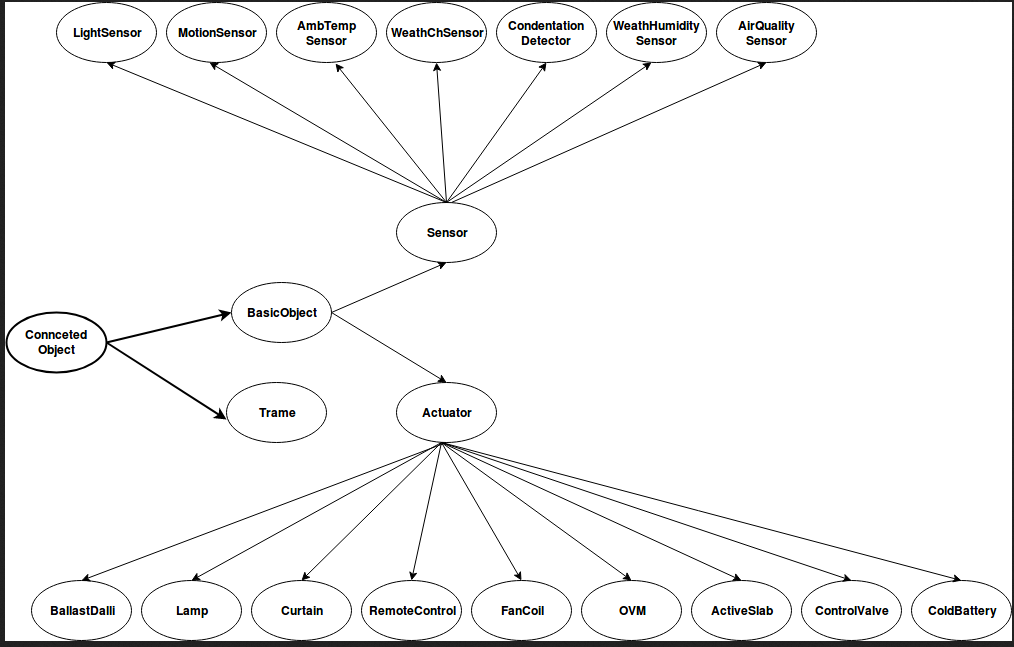
Notre ontologie peut se décomposer en quatre parties (Figure 12): la partie « Service », la partie « Settings » ou encore la partie « Location » et la partie « ConnectedObject »



*Figure 12 Concepts de base du smart building*

**La partie « ConnectedObject »**

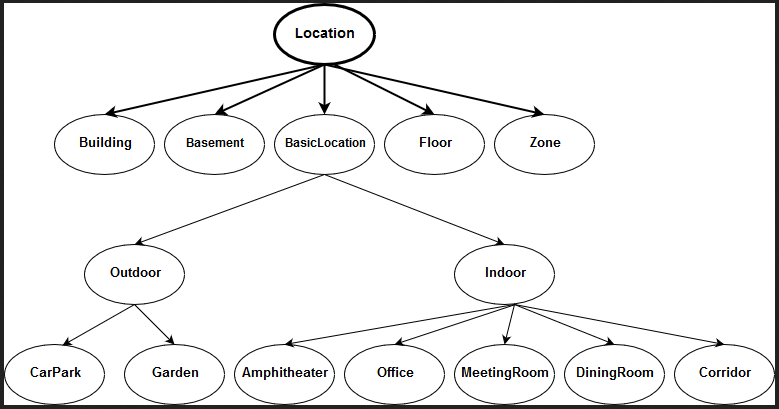
Le concept « ConnectedObject » regroupe tous les différents objets connectés qui se trouvent dans notre domaine. Le concept « ConnectedObject » possède deux sous-classes : la classe « BasicObject » et la classe «Trame ». Le concept « BasicObject » représente les objets connectés de base qui sont soit des actionneurs ou de capteurs. La classe « Trame » peut regrouper plusieurs objets de base (BasicObject) pour qu’ils puissent être gérés ensemble. Ce fait sera réalisé grâce à une propriété d’objet ( *hasElements* ) qui sera défini plus tard. La figure 13 représente la hiérarchie de classe de la super-classe « ConnectedObject ».



*Figure 13 Hiérarchie du concept " ConnectedObject"*

**La partie « Location »**

Le concept « Location » représente la structure ou les différentes localisations pertinentes d’un bâtiment. Les localisations de base (BasicLocation) représentent le plus bas niveau de granularité d’une localisation (bureau, Amphithéâtre...). Une  « Zone » regroupe des localisations de bases et chaque zone ne peut être affiliée que soit à un étage(Floor) ou à un sous-sol (Basement). Le concept « Building » représente le plus haut niveau de granularité d’une localisation et peut regrouper des étages et des sous-sols. Ces différents faits ou liens entre les différents granules de localisation seront exprimés à l’aide des propriétés d’objet. La figure 14 détaille les différentes classes qui représentent une localisation.



*Figure 14 Hiérarchie du concept "Location"*

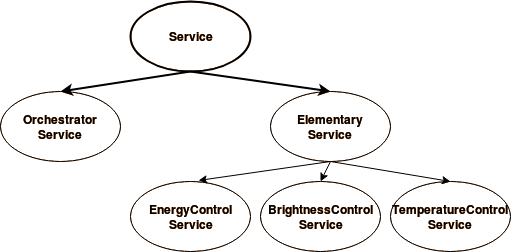
**La partie « Service »**

Le concept « Service » représente les différents services qui seront déployés sur notre environnement. Le concept «ElementaryService» regroupe les services de base qui ont le même objectif, tandis que le concept « OrchstratorService » regroupe plusieurs services de base pour pouvoir les orchestrer de façon intelligente*.*La Figure 15 nous montre les services déployé pour montrer le bon fonctionnement de notre ontologie.

Les services de type « EnergyControlService » s’occupent de gérer l’économie d’énergie dans le smart building

Les services de type « BrightnessControlService » s’occupent de contrôler le niveau de luminosité dans une localisation.

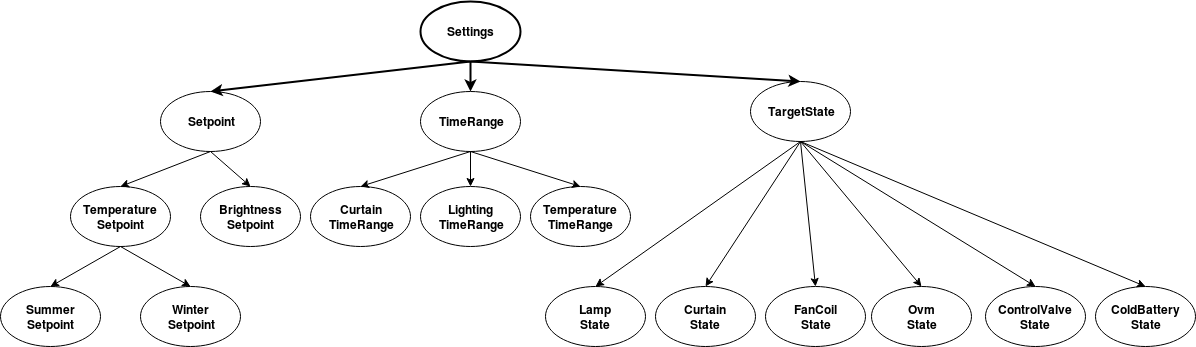
Les services de type « TemperatureControlService » s’occupent de contrôler le niveau de température dans une localisation.

**

*Figure 15 Hiérarchie du concept "Service"*

**La partie « Settings »**

La partie « Settings » regroupe les différents paramétrages de notre environnement tout comme il regroupe les états cibles des actuateurs. Le concept « Setpoint » représente les points de consigne qui seront donnés par l’utilisateur. Deux points de consigne sont disponibles, un point de consigne pour la température pour l’été et l’hiver et un point de consigne pour la luminosité. Le concept « TimeRange » représente des plages horaires, pour régler par exemple l’ouverture des stores ou l’extinction de l’éclairage. Le concept « TargetState » représente les états cible ou les états objectifs des objets connectés après l’exécution des règles d’inférences.



*Figure 16 Hiérarchie du concept "Settings"*

**Les propriétés de l’ontologie**

Le tableau suivant rassemble les différentes propriétés de notre ontologie. Les propriétés d’objets qui mettent en lien deux concepts, et les propriétés de type de données qui mettent en lien un concept et le domaine de valeurs que peut prendre ce concept.

|  |  |  |  |
| --- | --- | --- | --- |
| Property | Type | Domain | Range |
| hasBuilding | Object | Location | Building |
| hasControlOn | Object | ControlValve | ActiveSlab, Fancoil |
| hasElements | Object | Trame | ActiveSlab , BallastDalli , Lamp , LightSensor, MotionSensor |
| hasExecutionContext | Object | Service | Location |
| hasLayer | Object | Zone | Basement or Floor |
| hasServiceLink | Object | Service | ConnectedObject |
| hasLocation | Object | ConnectedObject | Location |
| hasBrightnessSetpoint | Object | LightingControl  Service | BrightnessSetpoint |
| hasTemperSetpoint | Object | TemperatureControl  Service | TemperatureSetpoint |
| hasCurtainTimeRange | Object | Zone | CurtainTimeRange |
| hasLightingTimeRange | Object | Zone | LightingTimeRange |
| hasTempTimeRange | Object | Zone | TemperatureTimeRange |
| hasTrame | Object | Corridor or Office | Trame |
| hasZone | Object | Location | Zone |
| Manage | Object | RemoteControl | Trame |
| Orchestrate | Object | OrchestratorService | Service |
| hasId | Data | « domain » | string |
| hasName | Data | « domain » | string |
| airQualitySensorValue | Data | AirQualitySensor | int |
| AmbientTempSensorValue | Data | AmbientTemperatureSensor | int |
| CondensationDetectorValue | Data | CondensationDetector | boolean |
| lightSensorValue | Data | lightSensor | int |
| temperatureSensorValue | Data | WeathChTempSensor | double |
| humiditySensorValue | Data | WeathChHumiditySensor | double |
| motionSensorValue | Data | MotionSensor | {"noPresence" , "presence"} |
| brightnessSetpointValue | Data | BrightnessSetpoint | double |
| coldSetpoint | Data | TemperatureSetpoint | double |
| hotSetpoint | Data | TemperatureSetpoint | double |
| endTimeValue | Data | TimeRange | dateTime |
| startTimeValue | Data | TimeRange | dateTime |
| hasTarget | Data | ElementaryService | « actuator » |
| IsEnable | Data | « actuator » | boolean |
| coldBatteryStateValue | Data | Coldbattery | {"close" , "open"} |
| curtainStateValue | Data | Curtain | {“up”,”down”} |
| lampStateValue | Data | Lamp | {“on”,”off”} |
| ovmstateValue | Data | OVM | {"close" , "open"} |

*Tableau 2 Les différents propriété de l'ontologie*

« domain » :Location, Floor,ConnectedObject, Building, Basement, Setpoint, Trame.

« actuator » :BallastDalli,ControlValve, ColdBattery, OVM, Curtain, Lamp

FanCoil,RemoteControl,ActiveSlab

## 3.3 Descriptions des règles

Les langages de règles sont apparus avec l’apparition des langages logiques et déclaratifs comme Prolog. Ces langages sont une pièce maîtresse dans la programmation logique et le domaine de l’intelligence artificielle. Leur utilisation s’est étendue avec l’avènement du web sémantique et dans la modélisation des entreprises afin de définir des objectifs à atteindre. Les règles permettent de modéliser et de décrire à très haut niveau le logique métier dynamique d’une application, où nous avons besoin de voir en temps réel les répercussions de ces changements dans le comportement de l’application.

En utilisant l’outil Protégé, nous allons créer une ontologie pour représenter notre environnement intelligent et le langage SWRL va nous permettre d’écrire des règles. Ces dernières seront utilisées par le raisonneur sémantique (Pellet) de Protégé pour pouvoir inférer et raisonner sur cette ontologie dans le but d’effectuer des *tâches* automatiquement en fonction des états des entités qui constituent le contexte de l’ontologie. Ce sont ces règles qui font qu’un contexte est dit intelligent.

Une règle est une séquence logique utilisée par le raisonneur sémantique. Grâce à l’ensemble des règles un environnement devient intelligent : elle adapte l’éclairage, le chauffage et la climatisation en fonction des paramètres du contexte.

Dans la suite de cette partie, nous expliciterons une partie des cas d’utilisations de notre solution sous forme de règles d’inférences conforme à l’ontologie décrite plus haut. Le but des règles d’inférences serait de définir un contexte actuel de manière précise afin de déduire la meilleure action à exécuter au sein de l’environnement intelligent.

Nous disposons d’un ensemble de règles qui forment des sous-ensembles qui regroupent les services qui partagent le même objectif:

* **Environnement de contrôle de luminosité :** Concerne l’environnement alimenté par les capteurs de luminosité et les actionneurs qui peuvent apporter de la lumière et se compose en deux sous service :
* **Service de lumière avec le store** : ce service s’intéresse aux actions visant à lever le store en fonction de la luminosité de la pièce et la présence d’une personne.
* **Rule 1 : IF(** MotionSensorValue= « presence » **&&** (Indoor)lightSensorValue < « 350 lumens »  **&&** curtainStateValue= « down » ) **THEN TargetState= «** CurtainToUp »
* **Service gestion de la lumière avec la lampe :** ce service s’intéresse aux actions visant à allumer la lampe (Rule2) ou à l'éteindre(Rule3) en fonction de la luminosité de la pièce.
* **Rule 2 : IF(** MotionSensorValue= « presence » **&&** (Indoor)lightSensorValue < «  350 lumens »  **&&** LampState= « off » ) **THEN TargetState= «** LampToOn »

* **Rule 3 :IF(**MotionSensorValue= « presence » **&&** (Indoor)lightSensorValue> «  500 lumens »  **&&** LampState= « on » **THEN TargetState** **= «** LampToOff »
* **Environnement de contrôle de température :** Ces règles définissent les actions à réaliser pour la régulation de la température, en s’appuyant sur la température d’une localisation et la température extérieure pour pouvoir agir éventuellement sur le chauffage et la climatisation ou le store :
* **Service de température avec le store :** ce service s’intéresse aux actions visant à descendre le store en fonction de la température et la luminosité extérieur.
* **Rule 4 : IF(** temperatureSensorValue >«  25 °c »  **&&** (Outdoor)lightSensorValue>«  500 lumens »)

**THEN TargetState = «** CurtainToDown »

* **Environnement de contrôle d'énergie :** Ces règles définissent les actions à réaliser qui vont nous permettre de faire de l'économie d'énergie, tel que d’éteindre la lampe en cas de non présence d’une personne dans la pièce :
* **Service d’économie d'énergie avec la lampe** : ce service vise à éteindre la lampe dans cas où il n'y a personne dans une pièce.

* + **Rule 5 : IF(**MotionSensorValue= « NoPresence » **&&**

LampSate= « on» **) THEN** **TargetState** **=  «** LampToOff »

## 3.4 Orchestration de service et résolution des conflits

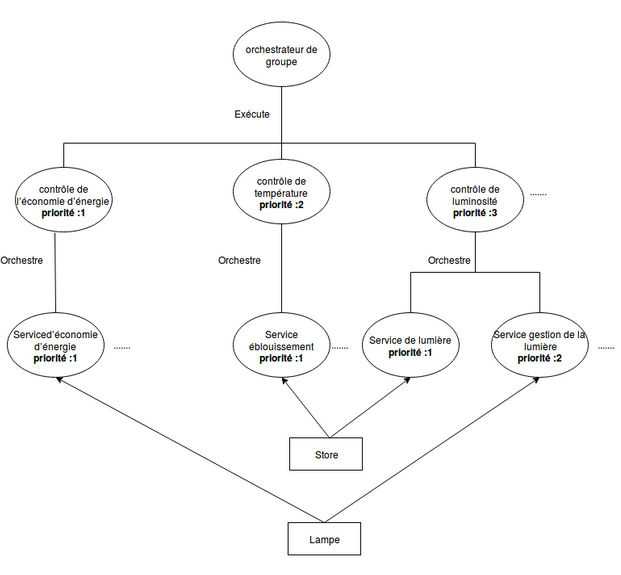
Les services sont classifiés selon deux catégories : les services élémentaires (service de base) et les services orchestrateurs.

Un service élémentaire cible un et un seul actuateur. Pour agir sur cet actuateur, nous associons à un service une règle d'inférence ou plusieurs règles qui sont exclusives et divers capteurs qui peuvent éventuellement participer à l'accomplissement de la tâche pour laquelle le service a été créé. Par exemple, pour un service qui s’inscrit dans le cadre de l'économie de l'énergie, on pourra créer un service qui cible une “ lampe” et on lui associe le capteur “MotionSensor”. Ce service éteindra une lampe s’il n’y a personne dans une pièce (Rule5).

Les services orchestrateur quant eux sont divisés en deux services les services qui orchestrent les services élémentaires (groupe objectif) et les services qui orchestrent les groupes objectifs.

Les orchestrateurs des services élémentaires sont des services qui regroupent plusieurs services élémentaires qui peuvent partager le même objectif. Ces services doivent être regroupés pour être orchestrés ensemble. Nous appelons ces groupes « groupe objectif ». Par exemple, dans la section précédente, nous avons défini deux services qui ont le même objectif c’est à dire : la gestion de la luminosité. “Le service lumière avec le store” avec la règle 1 et “le service gestion de la lumière avec la lampe” avec la règle 2 et 3, ces deux services partagent le même objectif qui est d’amener de la lumière, par conséquent ils seront regroupés dans un même service orchestrateur que nous appellerons “service lumière”. Ce dernier va orchestrer les deux services pour des raisons d’optimisation, par exemple privilégier la levée du store plutôt que d’allumer la lampe si le store apporte assez de lumière dans la pièce.

Les services qui orchestrent les orchestrateurs sont considérés comme le plus haut niveau d’orchestration. Ils prennent en charge l'exécution des services selon un ordre de priorité pour éventuellement éviter les conflits comme nous allons le voir dans le scénario “résolution de conflits”. Pour ce faire un service qui a une plus grand priorité s'exécutera et posera un verrou sur l’actuateur ciblé de telles sortes que les autres services de basse priorité ne pourront pas agir sur cet actuateur. La figure suivante nous montre la hiérarchie d’orchestration et des priorités.



*Figure 17 Hiérarchie d'orchestration*

# **4 Validation**

Dans cette partie, nous allons passer à la mise en pratique et la réalisation de notre application. Nous allons présenter, les outils utilisés et les différents scénarios réalisés pour le cadre de ce projet.

## 4.2 Les outils utilisés

### 4.2.1 Outils pour les langages de règles

Plusieurs outils sont disponibles pour éditer des règles en SWRL. Il existe des solutions payantes comme TopBraid Composer qui est un IDE pour les standards du web sémantique y compris SWRL, tout comme il existe des solutions libres, comme Protégé **[protege]** qui est un éditeur d’ontologie modulable. Il intègre le plug-in SWRLTAB qui fournit une interface pour pouvoir éditer des règles SWRL.

**SWRLTab**

SWRLTab est un plug-in pour le logiciel Protégé. Il offre un environnement de développement pour travailler avec les règles SWRL. Il permet de lire, écrire des règles SWRL et modifier les règles existantes.

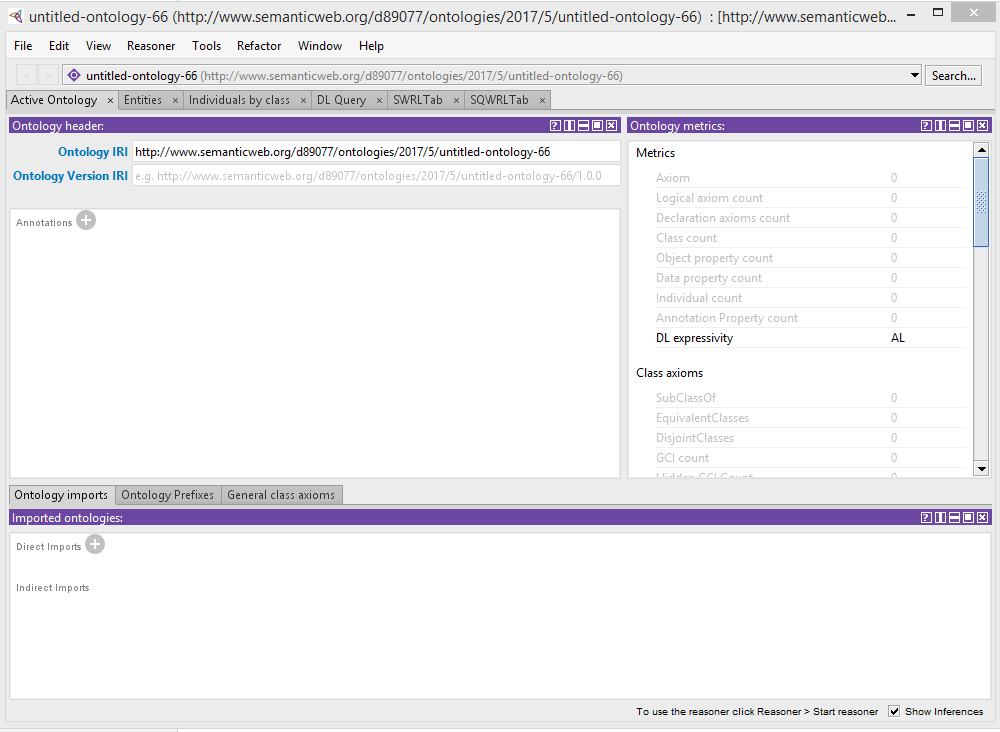
**PELLET**

Pour pouvoir utiliser les règles SWRL, nous avons besoin d’un raisonneur pour faire des inférences sur les données d’une ontologie. Protège peut intégrer diffèrent raisonneurs sous forme de plug-in. Dans le cadre de notre projet nous avons opté pour le raisonneur PELLET qui est un raisonneur open source écrit en Java

### 4.1.2 Outils pour les Ontologies

**Protégé**

Pour la création de notre ontologie, nous avons utilisé l’outil Protégé (figure 19) qui est un logiciel open source développé en Java pour la création et l’édition des ontologies. Il a été créé à l’université de Stanford. Il peut lire et sauvegarder des ontologies dans la plupart des formats (RDFS, OWL..). Protégé va nous permettre de modéliser les différentes classes, leurs propriétés et les liens entre celles-ci. Il permet aussi d’instancier l’otologie pour concrétiser les concepts décrits par l’ontologie.



*Figure 19 Outil Protégé*

**Apache Jena**

Jena **[jena]** est un ensemble d'outils dédiés à la construction d'applications orientées Web sémantique. Parmi ces outils, on trouve notamment une API Java open-source permettant de manipuler de nombreux langages tels que OWL, RDF/RDFS, SPARQL et de raisonner sur des modèles ontologiques à l'aide de moteurs d'inférences inclus dans Jena ou externes. Le raisonnement sémantique importe d'abord les règles et l'instance de l’ontologie puis il infère un ensemble d’actions à exécuter.

Jena va nous permettre de charger notre ontologie, l’instancier avec les données récupérées de notre environnement et enfin pouvoir faire de l’inférence sur ces données.

## 4.3 Réalisation sur Freedomotic

Pour pouvoir simuler un environnement intelligent, nous avons opté pour la solution logicielle Freedomotic **[freedo]** qui est un framework domotique open source développé en Java et distribué sous licence GNU GPL2. Modulaire et extensible, [Freedomotic](http://freedomotic.com) peut s’enrichir de nombreuses fonctionnalités en utilisant différents plugins ou encore en utilisant son API qui nous permettra de connecter d’autres systèmes communicants.

Le but de notre application est d’arriver à intégrer notre ontologie ainsi que nos règles à un environnement Freedomotic.

La première étape consiste à réaliser l’ontologie grâce à l’outil Protégé pour modéliser notre environnement, et tester les règles écrites en langage SWRL sur notre ontologie au sein même du logiciel Protégé.

Une fois que l’ontologie et les règles sont testées et validées, nous intégrons ces deux éléments par le biais de l’Api Jena sur notre module « MyOntology.java ». Cela permettra de charger l’ontologie et les règles. Ce module importe les différents outils de l’api Jena pour assurer ce traitement et nous permettre d’instancier des individus et notre ontologie, et en fin utiliser les règles pour pouvoir faire de l’inférence sur les données et ainsi récupérer interroger ces derniers dans l’ontologie via des requête SparQL.

En second lieu, nous incorporons le module précédent à un sous-projet du code de  
Freedomotic en tant que plugin où nous aurons à :

* Récupérer les valeurs des différents environnements via les capteurs pour capturer l’état actuel du système.
* Instancier toutes les localisations, objets présents sur l’environnement actuel de Freedomotic (le palier visible) dans le modèle de l’ontologie.
* Mettre au point les relations entre les concepts et leurs propriétés
* Faire de l’inférence sur ces données récupérées grâce aux règles pour avoir en sortie un nouveau modèle avec notre ontologie enrichi par les nouveaux fait inférer.
* Puis enfin interroger le nouveau modèle pour récupérer les actions qui doivent être exécuté dans l’environnement Freedomotic, pour avoir l’état du système tel qu’il devrait être.
* Exécuter les services concernés pour mettre à jour l’état du système d’une façon dynamique.



Figure 20 Interface Freedomotic

## 4.4 Scénarios de validation

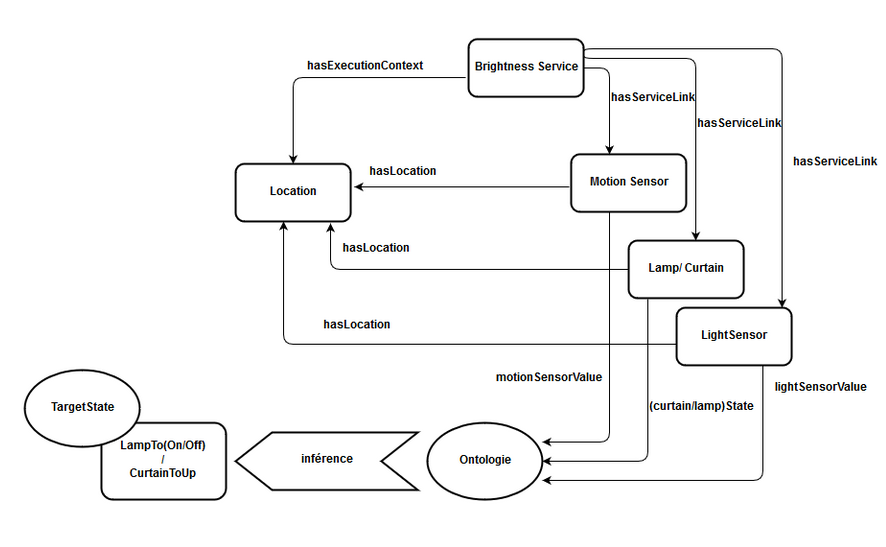
Dans cette partie, nous allons présenter quelques scénarios pour illustrer notre solution et le déroulement de l’exécution d’un service.

Le déroulement d’un service se fait de la manière suivante : en premier lieu, nous instancions dans l’ontologie les différents capteurs et actionneurs avec leurs données qui sont récupérées à partir des données du contexte. Par la suite, nous instancions le service tout en lui associant les différents objets qui seront utiles pour son bon fonctionnement avec la propriété « hasServiceLink ». Ensuite, l’ontologie s’occupera de balayer toutes les règles pour inférer celles qui sont exécutables. Enfin, nous interrogeons l’ontologie pour récupérer les données qui sont inférées. Ces données représentent l’état objectif du système.

Scenario 1 :

La figure suivante représente le déroulement des événements pour l'exécution d’un des services qui ont comme objectif la gestion de la luminosité.

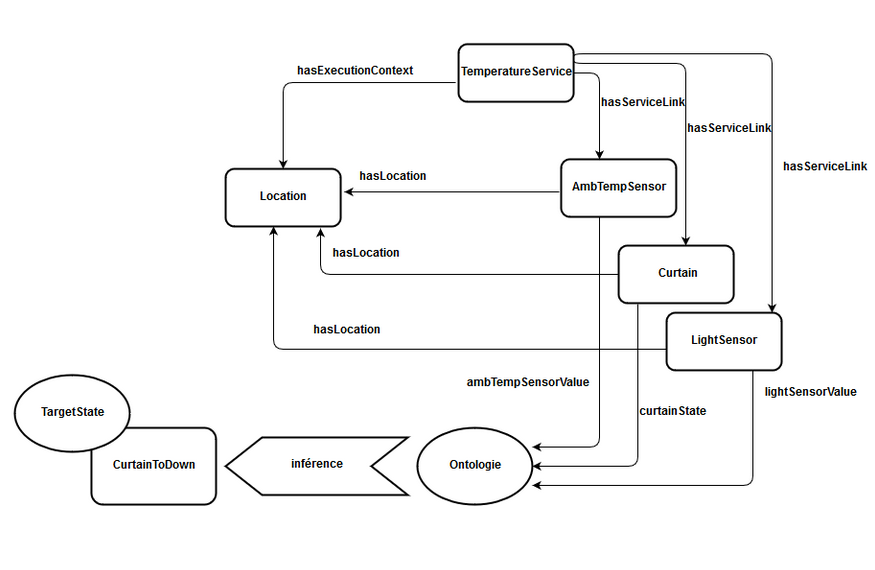
L'inférence conduit a trois états objectifs: lever le store (CurtainToUp) avec la règle 1, allumer la lampe (LampToOn) avec la règle 2 et éteindre la lampe (LampeToOff) avec la règle 3.L’apport de lumière est immédiat dans le cas de la lampe alors que, la levée du store pourrait prendre du temps. Quand le store sera completement levé, il peut arriver que la luminosité intérieurs ne soit pas suffisante. Dans ce cas, la lampe sera allumée à son tour. Dans le cas ou il y’a assez de lumière dans la pièce le service se chargera d'éteindre la lampe (règle 3).



*Figure 21 Déroulement d'un scénario d'un service qui a pour but la gestion de la lumière*

Scenario 2 :

La figure suivante nous montre le mode de fonctionnement du service de température. L'inférence conduits a l'état objectif “CurtainToDown” (règle 4) dans le cas où la luminosité et la température externe soit importante



*Figure 22 Déroulement d'un scénario d'un service qui a pour but de baisser la température*

Scenario 3 :

La figure suivante nous montre le mode de fonctionnement du service d’économie d’énergie avec la lampe. L'inférence conduits a l'état objectif “LampToOff” (règle 5) dans le cas où aucune personne ne se trouve dans la pièce.

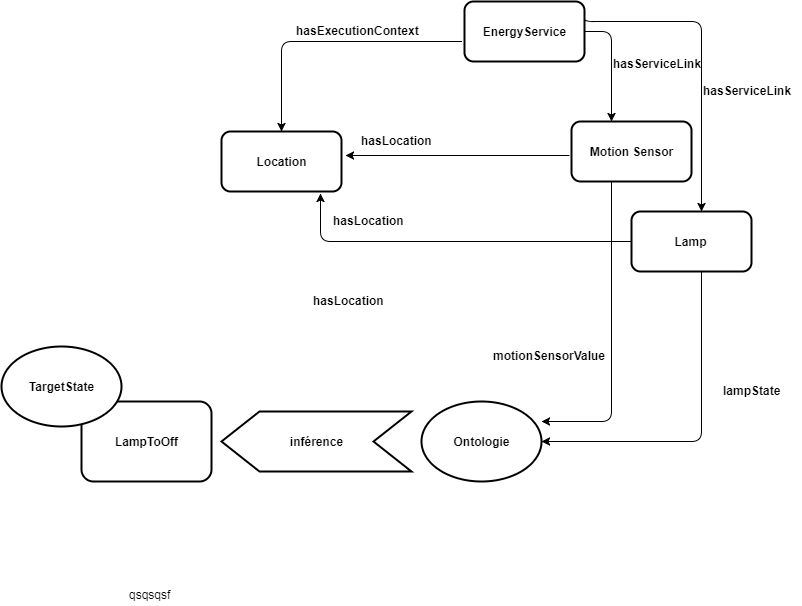
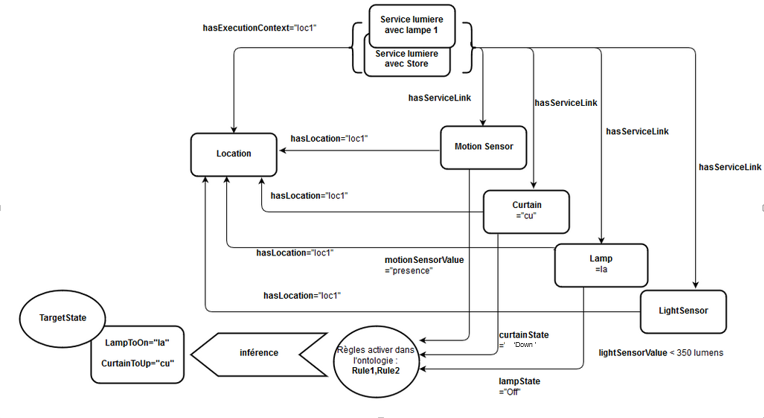


Figure 23 Déroulement d'un scénario d'un service qui a pour but l’économie de l’énergie avec la lampe

**Orchestration des services**

Les scénarios précédents nous montrent le fonctionnement des différents services. Si nous regardons le scénario 1, il représente le service orchestrateur “service Lumière” qui trois service de base (un service qui allume la lampe (rule2), un service qui éteint la lampe (rule3) et un service qui lève le store (rule1)). La figure suivante nous montre le résultat de l’exécution des services sans orchestration

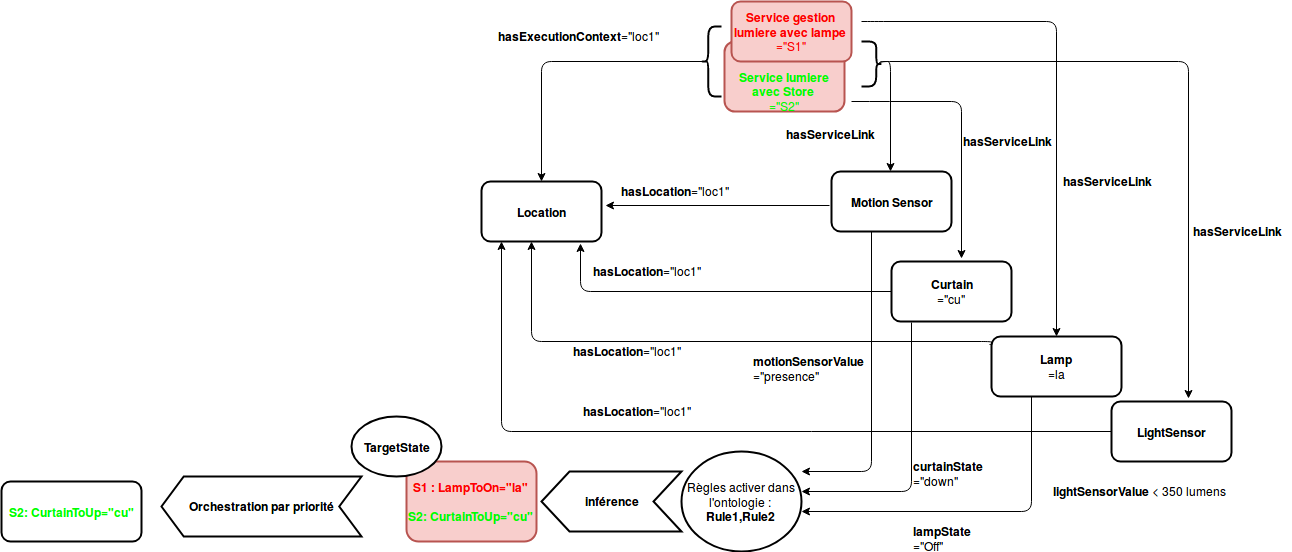


*Figure 24-a Exécution des services sans orchestration*



Figure 24-b Exécution des services sans orchestration sur Freedomotic

L'exécution nous montre qu’avec les données recueillies sur notre environnement dans ce cas de figure, notre ontologie va inférer les 2 actions suivantes : allumer la lampe “la” et lever le store “cu”. Le fait de lever le store pourrait apporter assez de lumière et nous n’aurons pas besoin d’allumer la lampe=”la”. D'où, la nécessité d'orchestrer ces deux services. La figure suivante nous montre le résultat après orchestration où nous avons donné la priorité au service qui lève le store pour apporter la lumière. Si la lumière n’est toujours pas suffisante, on allumera la lampe par la suite.



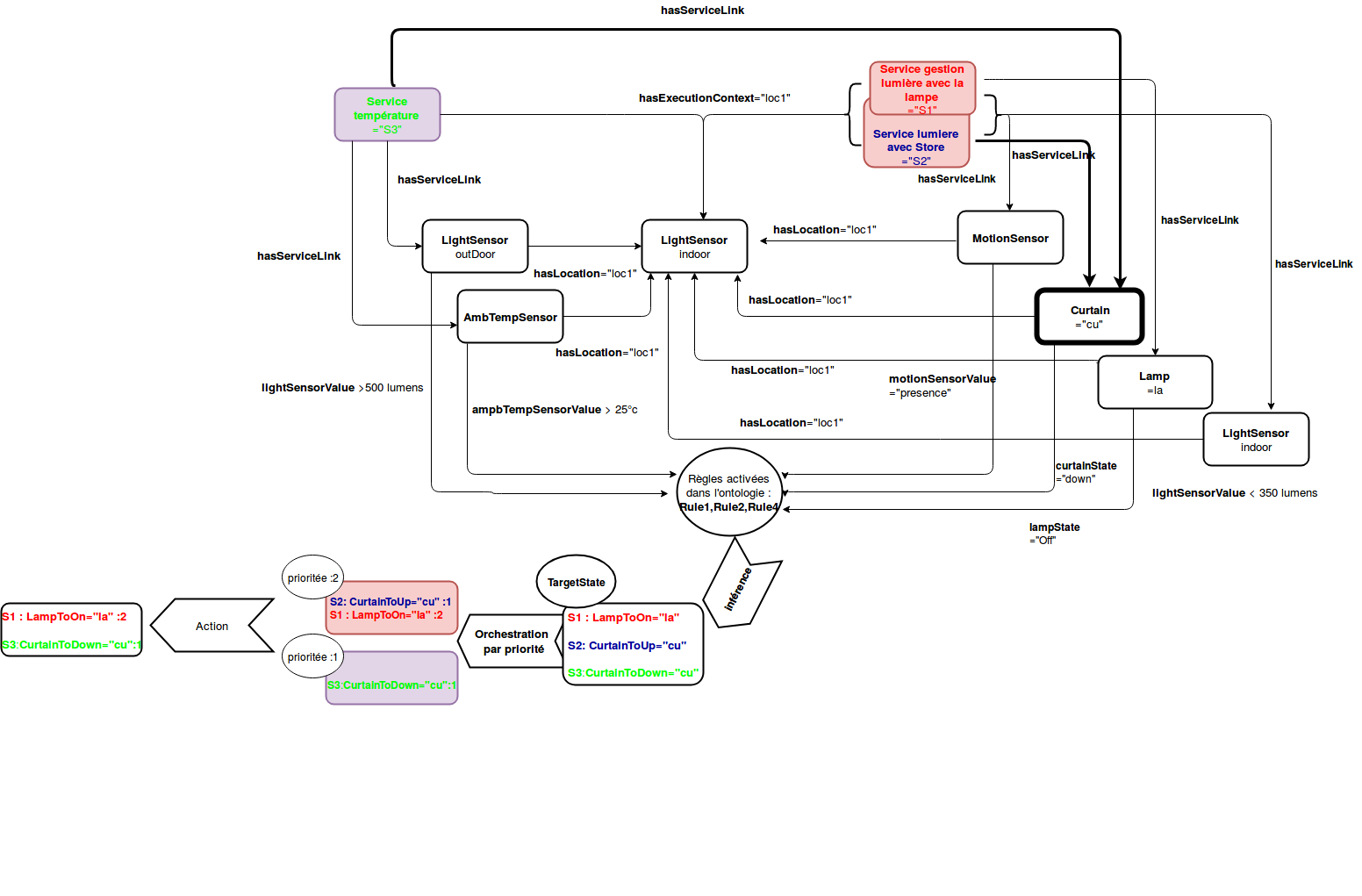
*Figure 25-a Exécution des services avec orchestration*



Figure 25-b Exécution des services avec orchestration sur Freedomotic

**Résolution des conflits**

Si nous prenons en considération les scénarios 1 et scénario 2 nous pouvons remarquer qu’ils peuvent éventuellement entrer dans une situation de conflit comme nous le montre la figure suivante où le service orchestrateur “Service lumière” demande la levée du store et le service de température demande la fermeture du store. La figure suivante nous montre la résolution de conflit entre ces 2 services.



*Figure 26-a Exécution des services avec résolution des conflits*



Figure 26-b Exécution des services avec résolution des conflits sur Freedomotic

La résolution de conflit se règle par priorité, où le service le plus prioritaire s'exécutera, et une fois exécuté il posera un verrou sur l’actuateur concerné par le conflit. Cette exécution nous montre que nous avons donné la plus haute priorité au service de température, de ce fait puisque les données recueillies remplissent les conditions du scénario 1 et 2 un conflit a été détecté. En exécutant le service du scénario 1, celui-ci exige que le store reste ou soit fermé. De ce fait, le scénario 2 est dans l'impossibilité d'exécuter le service le plus prioritaire de sa liste à savoir lever le store donc il se rabattra sur le service suivant qui est le service responsable d’allumer la lampe. Au final nous aurons à exécuter les services suivants : service température avec le store et le service lumière avec la lampe.

# 

# **5 Conclusion générale**

L’avènement de l’internet des objets et des nouvelles technologies de l’information a suscité de nombreux challenges comme la nécessité d’interconnecter ces objets communicants et l’automatisation de leurs actions. Tout comme leur dénouement a engendré l’apparition de nouveaux domaines d'application telle que les smart homes et smart building.

## 5.1 Bilan

Ce stage de fin d'étude m'a permis de me familiariser avec le domaine de la domotique et mettre en pratique les connaissances acquises durant ma formation. Durant les deux premier mois de ce stage, j’ai consacré mon temps de travail à l'étude de la thèse sur lequel va se baser mon travail, et faire des recherche sur les outils à utiliser pour réaliser mon travail. Il m’a été confié de travailler sur la couche sémantique de l’architecture proposée par la thèse. Après l'étude de cette partie, c’est en toute logique que je me suis dirigé vers le web sémantique, avec les outils qu’il offre tel que les ontologies et le langage SWRL. Ces outils répondent parfaitement à la problématique de la couche sémantique qui consiste à décrire les objets connecté et les règles d’automatisation d’un environnement intelligent. Une fois les outils choisis je me suis lancé dans la réalisation de notre ontologie qui consistait à représenter le bâtiment d’EDF, et la réalisation d’un ensemble de scénarios qui sont régit par un ensemble de règles, Au final, j’ai pu tester l’ensemble de la proposition sur un simulateur d’environnement intelligent.

Ce stage représente pour moi la première vraie expérience professionnelle. Il m’a permis de découvrir le milieu d’entreprise. J’ai particulièrement apprécié le fait que je n’ai pas été qu’un simple stagiaire, mais considéré comme un membre à part entière du groupe, pour cause le fait que j’ai pu choisir les technologies à utiliser et la prise en compte des idées que j’ai proposées.

## 5.2 Perspectives

Cependant, mon travail n’est pas entièrement fini, bien qu’il a permis de laisser prendre forme la couche sémantique de l’architecture, Il reste le fait de tester cette solution sur de vrais objets connectés et pouvoir enfin passer aux autres couches de l’architecture, les couches d’artefact et ressources.

# 

# Bibliographie

* **[EDOC 2016] :**  Rayhana Bouali Baghli, Elie Najm, Bruno Traverson. *Towards a Multi-Leveled Architecture for the Internet of Things*. EDOC'2016, Sep 2016, Wien, Austria
* **[WebS]:** Introduction au web sémantique

URL:<http://jplu.developpez.com/tutoriels/web-semantique/introduction/>

* **[OWL]:** Introduction à OWL, un langage XML d'ontologies Web,Xavier Lacot

URL: <http://lacot.org/public/introduction_a_owl.pdf>

* **[SPARQL]:** SPARQL Query Language for RDF

URL: https://www.w3.org/TR/rdf-sparql-query/

* **[SWRL] :**SWRL: A Semantic Web Rule Language Combining OWL and RuleML

URL: <http://www.w3.org/Submission/SWRL/>

* [**oneM2M**] :Towards Semantic Data Interoperability in oneM2M Standard

-Mahdi Ben Alaya, Samir Medjiah, Thierry Monteil, Khalil Drira

* **[protege] :** Protege outils pour les ontologies

URL: https://protege.stanford.edu/

* **[jena]:** Apache Jena

URL: <https://www.w3.org/2001/sw/wiki/Apache_Jena>

* **[freedo] :** outil Freedomotic

URL**:** <http://www.freedomotic.com/>