الجمهورية الجزائرية الديمقراطية الشعبية

République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université de 8 Mai 1945 – Guelma -

Faculté des Mathématiques, d'Informatique et des Sciences de la matière

Département d'Informatique

Mémoire de Fin d'études Master

Filière : Informatique

Option : Systèmes Informatiques

Thème :

Utilisation des Ontologies pour l'Intégration d'Internet des Objets dans la gestion des processus métier

Encadré Par :

Mme Djakhdjakha Lynda

Présenté par :

Boukara Djehina

Juillet 2019

Remerciements

Je tiens à adresser mes plus vifs remerciements à mon encadreur Mme DJAKHDJAKHA Lynda, pour sa disponibilité sa patience et pertinence de ses remarques tout au long de la réalisation de ce modeste travail, je remercie aussi tous les membres de jury qui ont accepté de juger mon travail ainsi que tous les enseignants qui ont contribué à ma formation. D'autre part, j'adresse une chaleureuse pensée à toute l'équipe pédagogique du département informatique.

Et le plus grand merci je l'adresse à my soul mate ma Bushra qui m'a toujours soutenue, merci d'être dans ma vie.

Résumé

Au cœur des dernières années, la notion d'Internet des Objets (IdO) (ou Internet of Things (IoT) en anglais) a évolué à une vitesse très exceptionnelle. IdO désigne un réseau d'équipements mobiles et intelligents connectés entre eux et communiquant de manière automatique. Les données produites de ces équipements sont principalement exploitées pour construire des applications « Internet des Objets ».

Dans le présent travail, nous cherchons à combiner cette révolution technologique avec des technologies du web sémantique pour gérer les processus métier, qui occupe aujourd'hui une place primordiale dans le domaine des systèmes d'information.

L'objectif de ce travail consiste à intégrer les données issues de l'IdO dans les processus métier exécutables en utilisant des ontologies OWL (Ontology Web Language) et des règles SWRL (Semantic Web Rule Language).

Mots clés

IdO, ontologie, OWL, SWRL, processus métier, BPEL,

TABLE DES MATIERES

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE D'ABREVIATIONS :

IdO	Internet des Objets
OWL	Ontology Web Language
WS-BPEL	Web Service Business Process Execution Language
RDF	Ressource Description Framework
RDF-S	Ressource Description Framework-Schema
BPEL	Business Process Execution Language
SWRL	Semantic Web Rule Language
BPM	Business Process Management
XML	Extensible Markup Language
WSDL	Web Service Description Language
IBM Knowledge Center n.d.	IBM Knowledge Center, Présentation des processus BPEL. Available at: http://www.ibm.com/support/knowledgecenter/fr/SSNJFY_8.0.1/com.ibm.wbpm.bpc.doc/topics/cbpel.html.
XPATH	XML Path Language
Kevin ASHTON	British technology pioneer cofounded the Auto-ID Center
UIT	Union Internationale des Télécommunications
OC	Objet Connecté
RFID	Radio Frequency Identification
WSN	Wireless Sensor Network
M2M	Machine to Machine
RF	Radar Fixe
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
MEMS	Microelectromechanical systems
RF-MEMS	Radio Frequency Microelectromechanical systems
NEMS	Nano Electro-Mechanical Systems
NFC	Near Field Communication
Bluetooth LE	Bluetooth Low Energy
CoAP	Constrained Application Protocol
MQTT	Message Queuing Telemetry Transport
REST HTTP	Representational state Transfer HyperText Transfer Protocol
OWL API	Application Programming Interface
OWL-S	OWL for Web Service
LOV4IoT	Linked Open Vocabularies for Internet of Things
SSN	Semantic Sensor Network
W3C	World Wide Web Consortium
RF	Radio Frequency

INTRODUCTION GENERALE

Introduction

De nos jours, la croissance importante du nombre d'objets connectés sur internet, et en particulier son influence sur la gestion des entreprises a conduit à la proposition de nouveaux thèmes de recherches. L'un des plus importants dans ce domaine est d'intégrer les données issues d'Internet des Objets (IdO) dans les processus métier, où les entreprises cherchent à utiliser ces données détectées depuis l'IdO en temps réel pour améliorer la qualité de ses produits ainsi que la prise de décisions par ses managers.

L'IdO désigne un réseau d'équipements mobiles et intelligents connectés entre eux et communiquant de manière automatique. Les données produites par ces équipements sont principalement exploitées pour construire des applications « Internet des Objets ».

Dans un contexte dynamique, les processus métier peuvent acquérir un avantage concurrentiel en utilisant des données réelles fournies par des objets connectés au cours de leur exécution.

Dans ce mémoire, nous cherchons à utiliser les technologies standards du Web Sémantique, en particulier les ontologies OWL pour donner une représentation sémantique qui permet de décrire les concepts relatifs à l'IdO et les éléments d'un processus métier exécutable décrit en WS-BPEL.

Dans le contexte du Web Sémantique, les ontologies sont utilisées pour représenter les connaissances d'un domaine particulier de manière formelle et réutilisable. Une ontologie est définie comme une spécification explicite d'une conceptualisation, où une conceptualisation est une vue abstraite du monde réel que nous représentons pour un objectif spécifique [1].

Dans notre travail, nous basons sur le langage OWL, qui a été développé par W3C afin de permettre aux applications de réaliser des raisonnements plus complexes [2]. OWL est une extension de RDF et RDFS pour la description des ontologies complexes par des classes et de types de propriétés. Il a l'avantage de donner de sens aux classes et relations définies entre celles-ci, de manière plus expressive. Il apporte une meilleure intégration, une évolution, un partage et une inférence plus facile des ontologies. OWL est un langage qui offre un cadre unificateur et fournit des primitives pour améliorer la communication entre les personnes, et entre les processus [3].

Problématique

Cependant, dans la description d'un processus exécutable BPEL, où les activités sont considérées comme des parties les plus influencées par les données envoyées par les objets connectés et les changements d'environnement, OWL reste incapable de représenter ces activités. Afin de contribuer à ce problème, nous pensons de compléter l'ontologie OWL proposée par un ensemble de règles SWRL afin de rendre la représentation plus complète.

De ce fait nous essayons dans ce mémoire de répondre à un certain nombre de question :
Comment intégrer les données issues d'IdO dans ces processus ?
Comment peut-on représenter les différentes activités avant l'intégration de ces données ?
Comment peut-on représenter les mêmes activités après la réception des données mesurées par l'IdO ?

Objectifs et contribution

Notre but vise à améliorer la gestion des processus métier exécutable pour le succès et la pérennité des entreprises.
Donc, dans ce mémoire :
Nous présentons un design d'une ontologie OWL qui permet de décrire les éléments d'un processus métier ainsi que les éléments relatifs à l'IdO.
Nous proposons un ensemble de règles SWRL permettant de représenter les activités après la réception des données mesurées par l'IdO.
Nous intégrons des équipements matériels permettant de mesurer des valeurs réelles.

Plan du mémoire

Dans ce mémoire, nous essayons dans la première partie de dresser un état de l'art pour inventorier les concepts de bases manipulés. Nous terminons cet état par une synthèse sur les travaux existants dans la littérature. La deuxième partie sera consacrée à notre contribution et son implémentation.
La première partie est composée de 4 chapitres :
Chapitre 1 présente le processus métier, l'approche de gestion de processus d'entreprise et son cycle de vie.

Chapitre 2 présente le terme IdO, le concept d'objet connecté, le fonctionnement d'IdO et les domaines d'application.

Chapitre 3 présente la notion d'ontologie et ses constituants, et décrit en détail les langages OWL et SWRL.

Chapitre 4 L'objectif de ce chapitre est de présenter des solutions et des approches existantes dans la littérature proches de notre travail et de discuter leurs limites.

La deuxième partie est composée de deux chapitres :

Dans le chapitre 5, nous présentons le cadre conceptuel que nous avons proposé pour intégrer l'IdO dans les processus métier et nous présentons l'architecture générale et les différents modules de notre approche.

Le chapitre 6 présente les outils et langages utilisés pour déployer notre approche avec quelques captures d'écrans.

Nous clôturons ce mémoire par une conclusion et quelques perspectives.

PARTIE I : ETAT DE L'ART

Chapitre 1 : Processus Métier

Introduction

Ces dernières années, les entreprises ont largement adopté l'approche processus pour faciliter la compréhension, le partage et l'évolution de ses activités.
Dans ce chapitre, nous présentons la définition du processus métier retenue dans notre travail de recherche. Nous présentons aussi le cycle de vie BPM et le langage d'exécution BPEL.

1. Définition du Processus Métier

Dans [4], l'auteur a cité plus de 13 définitions. Toutes les définitions ont met l'accent sur un ensemble de caractéristiques d'un tel processus : les activités, la structure, l'objectif et les acteurs. Dans notre travail, nous retenons la définition proposée par [5], où un processus métier de l'entreprise est défini comme une collection d'activités ordonnées servant à invoquer des services et à produire des résultats métier pour la satisfaction d'un client.

2. Cycle de vie BPM

Le cycle de vie d'un processus métier signifie l'ensemble de phases depuis l'identification jusqu'à l'amélioration de ce processus, à l'aide de BPM [6]. Il peut être décomposé selon la figure suivante en :
Etude : cette étape consiste à analyser les objectifs et les organisations d'une entreprise
Modélisation des processus métiers : c'est la phase la plus importante, elle consiste à représenter l'ensemble des processus par des modèles.
Implémentation : consiste à implémenter la solution modélisée.
Exécution : c'est la phase de mise en œuvre de la solution.
Pilotage : cette phase permet d'analyser l'état des processus à travers des tableaux de bords pour montrer les performances des processus identifiés.
Optimisation : elle consiste à proposer des améliorations des performances des processus métiers.

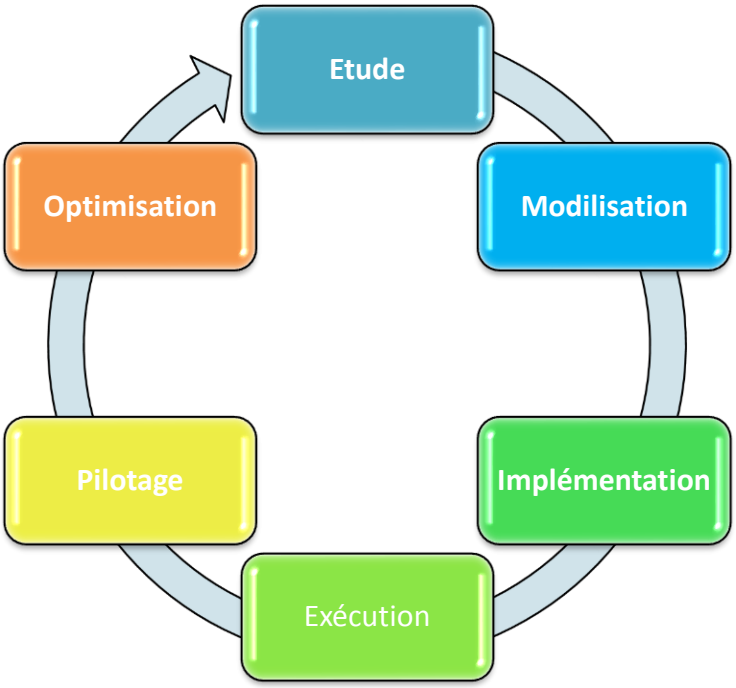


Figure 1.1 : *cycle de vie d'un BPM* [7]

3. Langage d'exécution du processus métier BPEL

BPEL (Business Process Execution Language) est un langage spécialisé et standardisé, qui est considéré comme le langage dominant dans sa catégorie, vu sa large adoption dans les entreprises. Il est un langage qui utilise la syntaxe XML pour la description de la logique interne des processus métier. Ce langage est basé sur le trio : - WSDL afin de pouvoir utiliser les services requis pour la composition, - XML Schema et - XPath permettent le traitement de données XML Schema via des expressions XPath [8].

Un processus BPEL est un processus défini en langage WS-BPEL (Web Services Business Process Execution Language). WS-BPEL comprend les éléments suivants (IBM Knowledge Center n.d.) :

- Les activités composant chaque étape du processus.
- Les liens partenaires, également appelés partenaires d'interface ou partenaires de référence, qui spécifient l'interaction avec les partenaires externes via les interfaces WSDL.
- Les variables qui enregistrent les données échangées entre les activités.

- Les ensembles de corrélations utilisés pour mettre plusieurs interactions de service en corrélation avec la même instance de processus BPEL.
- Des gestionnaires d'erreurs traitant les situations exceptionnelles qui peuvent se produire lors de l'exécution d'un processus.
- Des gestionnaires d'événements recevant et traitant les messages non sollicités parallèlement à l'exécution normale des processus.
- Des gestionnaires de compensations définissant la logique de compensation d'une activité, ou d'un groupe d'activités.

3.1. Les activités d'un processus BPEL

Les activités constituent la majorité des structures d'un processus BPEL. Chaque activité comporte cinq attributs [9]:

- **partnerLink :** Contient le nom du partnerLink à utiliser. Ce nom est utilisé par le moteur d'exécution BPEL pour identifier la destination actuelle.
- **portType :** La valeur donnée indique le type de port WSDL du service cible qui contient l'opération à invoquer.
- **opération :** cet attribut identifie l'opération WSDL à invoquer.
- **variable d'entrée :** cet attribut contient les données à envoyer.
- **variable de sortie :** cet attribut est initialisé avec la réponse reçue du partnerLink.

Les activités sont classées en deux catégories : les activités de base et les activités structurées.

- **Les activités de base :**

Parmi les activités de base les plus importantes, on trouve celles qui sont liées à la réception et à l'envoi des messages, respectivement, de la part et à destination des partenaires. Ces activités sont :

- **<receive>** : autorise de recevoir un message provenant d'un Web Service.

<receive **name**="Receive1" **createInstance**="yes" **partnerLink**="Achat_Client" **operation**="Achat_WSDLOperation" **portType**="tns:Achat_WSDLPortType" **variable**="Achat_WSDLOperationIn"/>

- **<reply>** : autorise de répondre au Web Service appelant.

<reply **name**="Reply1" **partnerLink**="Achat_Client" **operation**="Achat_WSDLOperation" **portType**="tns:Achat_WSDLPortType" **variable**="Achat_WSDLOperationOut"/>

- **<invoke>** : autorise d'invoquer un service web à travers un Partner Link.

<invoke **name**="UpStockProduction" **partnerLink**="Stock_Production" **operation**="update" **portType**="tns:jdbcPortType" **inputVariable**="UpdateInBlProduction"/>

- **<assign> :** autorise d'assigner la valeur d'une variable. Les deux règles copie (copy) et ajout (append) sont les principales opérations proposées par Assign.

<assign **name**="Assign4">

<copy> <from>… </from> <to>… </to> </copy>

</assign>

Comme il existe aussi les activités :<empty>, <throw>, <exit>, <wait> et <rethrow>.

- **Les activités structurées :**

Les activités structurées décrivent l'ordre d'exécution de leurs sous-activités, telles que les activités <sequence>, <if-else >, et l'activité <flow> permet une exécution parallèle de ses activités ainsi que leur synchronisation. Enfin, l'activité <pick> définit un choix contrôlé par l'arrivée d'un événement, par exemple elle met en attente une activité jusqu'à l'arrivé d'un message [10].

- **Activités répétitives**

BPEL offre des activités permettant une exécution répétitive en utilisant des boucles. Depuis la version 2.0 de WS-BPEL, il existe 3 types de boucles qui sont : while, forEach et repeatUntil [11].

- **While** : une boucle permet de faire répéter une activité tant que la condition reste satisfaite (vraie).
- **ForEach** : c'est une boucle aussi mais cette dernière permet de répéter une activité autant de fois que défini par une règle de comptage.
- **RepeatUntil** : elle permet de répéter une activité tant qu'une condition n'est pas devenue vraie.

4. WSDL :

WSDL (Web Services Description Langage) [12] intègre les informations sur les fonctions disponibles publiquement, les types de données des messages, le protocole de transport à utiliser, l'adresse pour localiser le service, et les définitions abstraites des données [13]. L'objectif du langage WSDL est de fournir la description des services sous une forme, que

des personnes ou des programmes peuvent interpréter [14]. La figure suivante montre la structure d'un fichier WSDL.

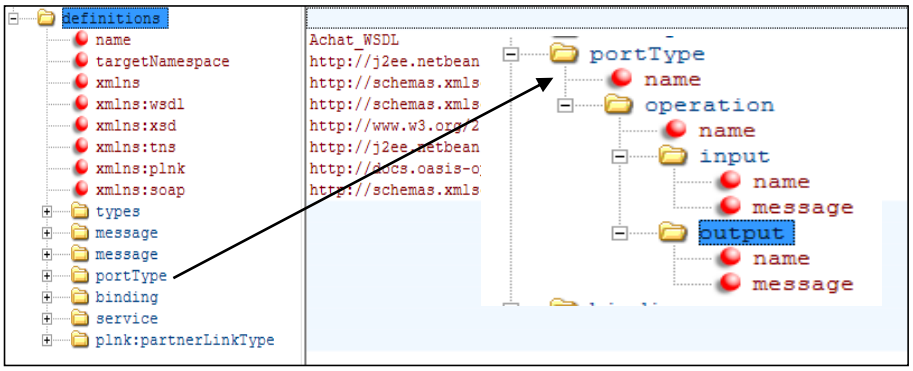


Figure 1.2 : *Structure d'un fichier WSDL*

Un fichier WSDL [11] contient donc sept éléments : - ‹*Type*›: fournit la définition de types de données utilisés pour décrire les messages échangés. - ‹*Message*› : représente une définition abstraire (noms et types) des données en cours de transmission. -‹*PortTypes*› : décrit un ensemble d'opérations. Chaque opération à zéro ou un message en entrée, zéro ou plusieurs messages de sortie ou d'erreurs. - ‹*Binding*› : spécifie une liaison entre un ‹portType› et un protocole concret (SOAP, HTTP...). - ‹*Service*› : indique les adresses de port de chaque liaison. ‹*Port* : représente un point d'accès de services défini par une adresse réseau et une liaison. - ‹*Opération*› : c'est la description d'une action exposée dans le port.

Conclusion

Dans ce chapitre nous avons présenté la définition de processus métier retenue dans notre travail, nous avons ainsi fait un survol sur les différents éléments de BPEL.

Dans le chapitre suivant, nous allons aborder la technologie d'IdO.

Chapitre 2 : Internet des Objets

Introduction :

Durant les dernières années, les chercheurs se sont dirigés vers l'IdO et son impact sur notre vie quotidienne, spécialement après l'augmentation du nombre d'objets connectés autour du monde qui aujourd'hui dépasse même le nombre de personnes [15].
Au début, l'IdO servait à désigner des systèmes capables de créer et transmettre des données pour créer de la valeur pour ses utilisateurs via une connexion entre des objets et d'Internet. Avec le temps, le terme IdO a évolué et il inclut maintenant tout l'écosystème des objets connectés [16].
Dans ce chapitre, nous présentons la définition de l'IdO et celle d'objet connecté. Ensuite nous décrivons le fonctionnement d'IdO. Enfin, nous citons un ensemble de domaines d'application de l'IdO.

1. Définitions d'internet des objets :

Définition 1/ Selon l'UIT, l'Internet des Objets est définie comme « une infrastructure mondiale pour la société de l'information, qui permet de disposer de services évolués en interconnectant des objets (physique ou virtuels) grâce aux technologies de l'information et de la communication interopérables existantes ou en évolution » [17].
Définition 2/ L'IdO définit différentes solutions techniques avec un ensemble de caractéristiques : identification des objets, capter, stocker, traiter, et transférer des données dans les environnements physiques [18].

2. Objet Connecté(OC)

L'OC est un dispositif qui peut interagir avec le monde physique indépendamment, sans intervention humaine.
Il a plusieurs caractéristiques comme : la mémoire, la bande passante, consommation d'énergie, … Ce concept doit être adopté à un usage, il a divers forme d'intelligence, une capacité de recevoir, de transmettre des données avec des logiciels à cause des capteurs embarqués.

L'intégration d'Internet avec un OC permet de l'enrichir fonctionnellement, et en terme d'interaction avec son environnement, il devient un OC Enrichi (OCE) [19].

Les trois éléments clés d'OC:

- Le stockage et la transmission des données produites ou reçues.
- Des algorithmes pour traiter ces données
- L'écosystème où il va réagir et s'intégrer.

2.1. Les propriétés d'usage d'un OC [19] :

- Ergonomie (utilisabilité, maniabilité, …).
- Esthétisme (formes/couleurs/sons/sensations, …).
- Usage (histoire culturelle, profil, matrice sociale, …).
- Méta Morphisme (adaptabilité, personnalisation, modulation, …)

3. Les technologies de l'IdO :

Afin d'assurer le fonctionnement de L'IdO, il existe plusieurs technologies utilisées, on va parler seulement sur quelques technologies tel que : RFID, WSN et M2M [20] :

- RFID : ce concept englobe toutes les technologies qui travaillent avec les ondes radio afin d'identifier automatiquement des objets ou des personnes. Ces caractéristiques sont: le stockage et la récupération des informations à distance [21].
- WSN : c'est un réseau coopératif, chaque nœud dans le réseau possède un ensemble de caractéristique telles que : capacité de traitement, différents types de mémoires, un émetteur-récepteur RF et une source d'alimentation, et divers capteurs et des actionneurs [22].
- M2M : c'est les technologies de l'information, combinées avec la communication des objets intelligents, afin de fournir à ces derniers la capacité d'interagir sans intervention humaine avec le système d'information d'une organisation ou d'une entreprise [23].

4. Domaine d'application de l'IdO :

Dans [24], les auteurs ont cités différent domaine d'application et d'autre auteurs ont les classées dans l'article [23] comme suit :

- **Domaine personnel :** Agriculture intelligente, domotique, ..
- **Domaine du transport :** transport intelligent, positionnement, circulation routière intelligente, gestion des flux et aussi systèmes intelligents coopératifs, qui permettent aux véhicules de communiquer entre eux et avec leur environnement.

- **L'environnement :** prédiction des séismes, détection d'incendies, qualité de l'air,…
- **L'infrastructure, les services publics et services de la ville intelligente**
- **L'industrie :** mesure, pronostic et prédiction des pannes, dépannage à distance, …
- **La santé :** Les opérations chirurgicales, le contrôle et la suivie des paramètres biologiques à distance,…
- **l'éducation et la recherche.**

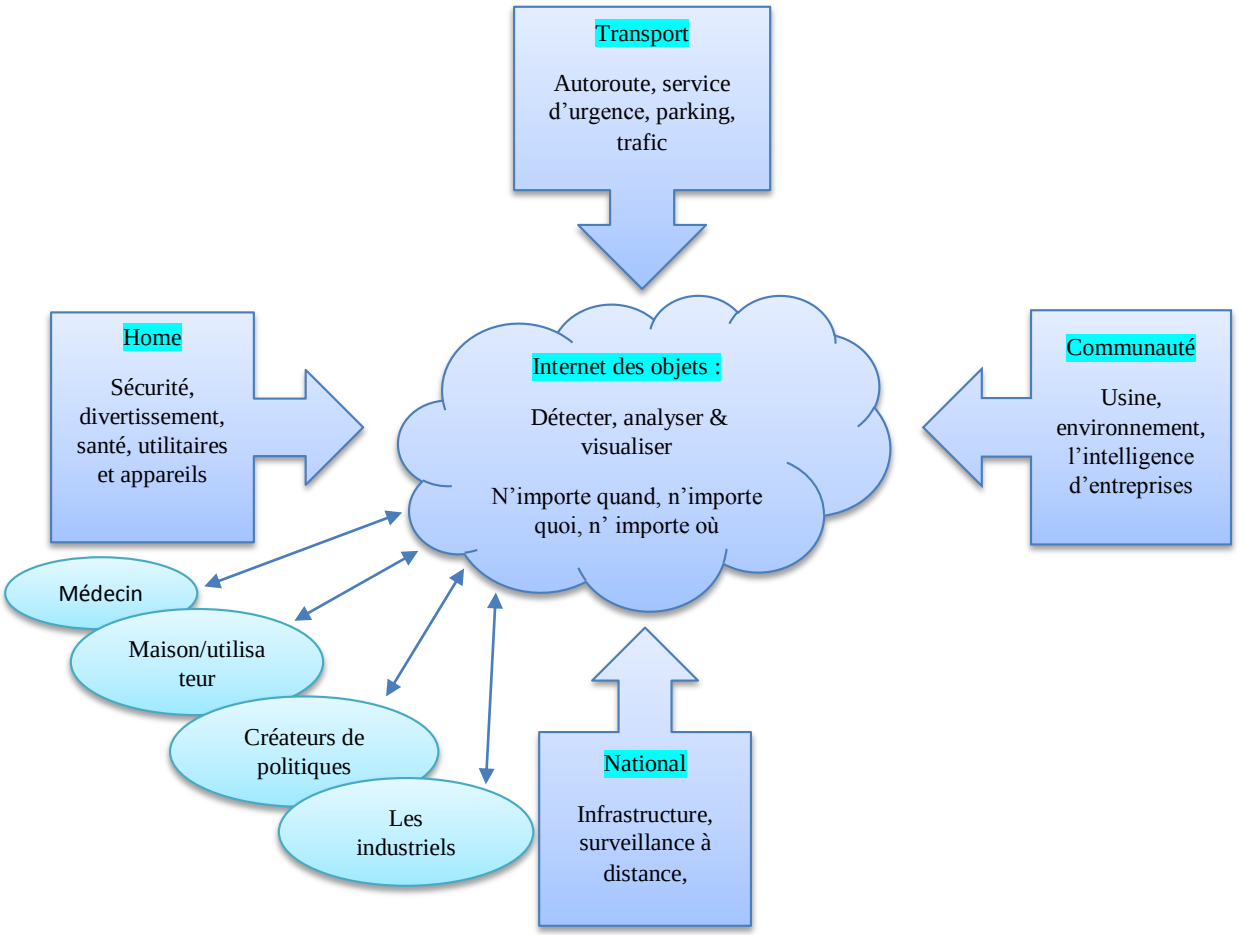


Figure 2.1 : *Les domaines d'Internet des Objet*. Adaptée de [24]

5. Les étapes et les technologies dans l'écosystème de L'IdO :

Les objets connectés sont au milieu d'IdO, et il est important de pouvoir connecter l'ensemble de ces objets, les faires échangé des informations et interagir, au sein d'un même environnement [19]. La mise en place de l'IdO passe par :

- **L'identification** : Rendre possible l'identification de chaque élément connecté, les technologies utilisé pour la mise en place de l'IdO sont : IPv4, IPv6, 6LoWPAN, …

- **L'installation des capteurs :** Mise en place de dispositifs nous rapprochant du monde réel, les technologies utilisées pour le mise en place de l'IdO sont : MEMS, RF-MEMS, NEMS,…
- **Les fonctions de base des objets** (capteur de température pour le thermomètre)
- **La connexion des objets entre eux :** faire une connexion entre les objets pour qu'ils puissent dialoguer et s'échanger des données : les technologies utilisé pour la mise en place de l'IdO sont : SigFox, LoRa, …
- **L'intégration :** Disposer d'un moyen de communication rattachant les objets au monde virtuel, les technologies utilisé pour la mise en place de l'IdO sont : RFID, NFC, Bluetooth LE, …
- **La connexion à un réseau :** Relier les objets et leurs données au monde informatique via un réseau par exemple l'internet, les technologies utilisé pour la mise en place de l'IdO sont : CoAP, MQTT, AllJoyn, REST http, ...

Conclusion

Durant les dernières années, la plupart des entreprises ont mené d'une stratégie IdO, afin de créer de la valeur soit pour développer et augmenter le chiffre d'affaires, ou bien rationnaliser ses processus métier afin d'améliorer la rentabilité.

Le problème est comment peut-on intégrer cette stratégie dans une telle entreprise ? Dans le présent travail, nous cherchons à utiliser les formalismes standards du web sémantique. De ce fait, les ontologies OWL et les règles SWRL seront l'objet de la section suivante.

Chapitre 3 : Ontologie et règles SWRL

Introduction

Depuis l'émergence du web sémantique avec ses différents formalismes, les chercheurs ont mis l'accent sur l'utilisation de ces derniers pour donner des solutions à plusieurs problèmes. Dans cette section nous présentons, le concept d'ontologie, et les formalismes standards OWL et SWRL.

1. Ontologie:

Définition 1/ Le terme ontologie est repris en informatique, particulièrement en Ingénierie des Connaissances (IC), pour modéliser des connaissances pour les Ssystèmes à Base de Connaissances (SBC).

En 1993, Gruber a définit l'ontologie comme étant une représentation explicite et formelle d'une modélisation abstraite d'un domaine [25].

1.1. Les composants d'une ontologie :

Une ontologie est composée de : concepts, propriétés, relations, axiomes et des instances [26].

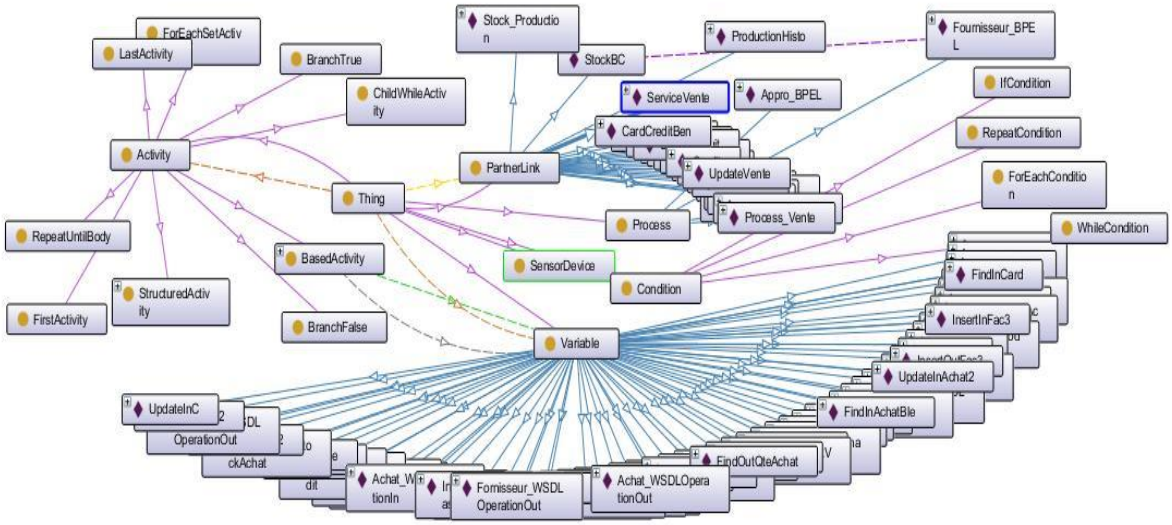


Figure 3.1 : *Une ontologie sous forme de graphe.*

- **Le concept (classe)** : Il peut représenter un ensemble d'objet, une idée, un principe ou une notion abstraite. Il est caractérisé par une extention et une intension.
- **Les propriétés :** sont des caractéristiques valuées 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 permetent de structurer hiérarchiquement les concepts d'une ontologie par exemple : Identité, synonymie(symétrique), sort de(unidirectionnelle), l'équivalence, homonymie, antonomie.
- **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.

1.2. Le rôle d'ontologies

Une ontologie permet de [27] :

- 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.

1.3. Langage d'ontologies :

Pour décrire une ontologie, plusieurs langages ont proposés dans la littérature tels que : XML, RDF, RDFS et OWL. XML fournit une syntaxe à des documents structurés, mais il n'impose pas les contraintes sémantiques. - RDF est un modèle de données qui peut être représenté dans une syntaxe XML.- RDF Schema (RDFS) est un langage de définition pour la description de propriétés et de classes représentées par des ressources RDF. - Il définit des graphes de triplets RDF, avec une sémantique de généralisation / hiérarchisation de ces propriétés et de ces classes. - OWL ajoute des vocabulaires pour la description des propriétés et des classes, des relations entre classes (exemple disjointness), des cardinalités, des caractéristiques de propriétés (symmetry), et des classes énumérées. OWL est développé comme une extension du vocabulaire de RDF[28].

1.1.1. Langage d'ontologie OWL :

OWL vise à rendre les ressources sur le Web facilement accessibles aux processus automatisés, en les structurant d'une façon compréhensif et standardisée, et en les ajoutant des méta-informations. Cependant, OWL a des moyens puissants pour exprimer la signification et la sémantique que XML, RDF, et RDFS [28]

OWL a trois sous langages expressifs : OWL Lite, OWL DL, et OWL Full.

- **OWL Lite :** il est conçu pour améliorer la hiérarchie de classification et les contraintes simples. Il est plus simple qu'OWL DL. OWL Lite supporte seulement un sous-ensemble de constructions du langage OWL [29].
- **OWL DL (Description language)** : utilise la logique de description DL. Il est conçu pour les utilisateurs pour garantit la complétude des raisonnements, toutes les inférences sont calculables, et leur décidabilité. Il inclut toutes les constructions du langage OWL, qui ne peuvent être utilisées que sous certaines restrictions [29].
- **OWL Full :** OWL Full a été défini pour mettre une expressivité maximale et une liberté syntaxique de RDF sans des garanties informatiques aux utilisateurs.

  Il permet : - à une ontologie de faire accroître la signification du vocabulaire prédéfini (RDF ou OWL). - le mélange libre d'OWL avec RDF Schéma et comme RDFS n'impose pas une séparation stricte des classes, des propriétés, des individus, et des valeurs de données.

2. Les règles SWRL :

SWRL est un langage qui enrichit la sémantique d'une ontologie définit en OWL. Il permet de décrire des règles sous la forme d'implications logiques entre des conditions et des conclusions [30]. Contrairement à OWL, il permet de manipuler des instances par des variables (?x, ?y, ?z).

SWRL ne permet pas de créer des concepts ni des relations. Il permet simplement d'ajouter des relations suivant les valeurs des variables et la satisfaction d'une règle. Il permet la définition des règles explicitant des comportements à suivre en fonction des informations, afin de créer une nouvelle information d'une information existante.

Dans le domaine des ontologies, une règle est une instruction conditionnelle simple du type :

« SI condition (corps) ALORS conséquence (tête) » qui signifie que :

Si on a pu démontrer la partie corps (antécédent), alors les conditions spécifiées dans la tête (conséquent) le sont aussi (antécédent ⇒conséquent).

Le corps et la tête d'une règle sont des conjonctions d'atomes. Les atomes peuvent avoir les formes [28] :

- C(x) où C est une description OWL, x est soit une variable, un individu OWL ou encore des valeurs de données (data values) de OWL.
- P(x,y) où P est une propriété OWL (Object_property ou data_type_property), x est soit une variable ou un individu OWL et y est soit une variable, un individu OWL ou encore des valeurs de données de OWL.

Le fonctionnement d'une règle est basé sur le principe de satisfiabilité de l'antécédent ou du conséquent.

Ce langage de règles puissant fait l'objet d'intégrations dans de nombreux outils actuels tels que Protégé ou OWL API [31].

Conclusion

Dans ce chapitre, nous avons présenté la notion d'ontologie, ses composants et le rôle joué par celle-ci dans le contexte du Web Sémantique. Nous avons présentés les langages OWL et SWRL.

Dans le chapitre suivant, nous examinons quelques travaux existant afin de bien positionner notre travail.

Chapitre 4 : Les travaux connexes

Introduction

Après une étude bibliographique sur les travaux en ligne, qui sont proches de notre travail, nous avons pu classer les travaux étudiés en quatre points :

1. Intégration des données transmises depuis l'IdO dans BPEL :

L'intégration des données transmises depuis l'IdO dans les processus BPEL a été traitée dans quelques travaux de recherche [32], [33]. Les auteurs de ces travaux ont étendu WS-BPEL avec des variables de ''contexte '' pour supporter des changements d'environnement. Ils ont ajouté de nouveaux constructeurs au langage WS-BPEL pour gérer les changements. Le problème avec ces solutions réside dans la limitation sémantique du langage BPEL.

2. Enrichissement sémantique des processus métier

Pour remédier aux limitations sémantiques des langages de processus métier BPEL, WSDL et BPMN, de nombreux travaux [34]… [35] et [36] ont focalisé leurs efforts sur l'introduction de la sémantique dans le domaine des processus métiers en utilisant des ontologie, ou en mappant les processus vers OWL ou OWL-S. Dont l'objectif est de proposer des solutions pour représenter l'aspect statique et dynamique de l'entreprise. L'ensemble de ces solutions basées sur des ontologies de domaine ont amélioré le niveau sémantique des aspects dynamiques du processus métier.

3. Développement des ontologies d'IdO.

Dans le domaine d'IdO, LOV4IoT a référencé plus de 330 ontologies [37]. Les ontologies existantes sont des ontologies de domaine [37]. Plusieurs papiers sont proposés pour étudier ces ontologies [37], [38] et [39]. L'ontologie (SSN) est considérée comme une ontologie standard proposée par le W3C [40] pour décrire les capteurs et les données fournies sous forme d'observations.

Dans [37], les auteurs ont présenté les concepts ontologiques fondamentaux nécessaires à une application basée sur l'IdO.

Dans notre travail, la liste des concepts proposée dans [37] pourrait être très utile pour l'intégration les valeurs transmises à travers les objets dans les processus métier.

4. Raisonnement OWL/SWRL

Dans cette catégorie, les auteurs ont présenté dans [41] l'application de SWRL / OWL pour la représentation des connaissances pertinentes pour un scénario de la chaîne logistique. Ils ont utilisé les règles SWRL pour démontrer leur application de connaissance de la situation SAWA. Dans [42], les auteurs ont proposé un raisonneur basé sur des règles pour déduire et fournir des services de requête basés sur OWL et SWRL.

Conclusion

Après cette étude, nous pensons de profiter des avantages des travaux cités auparavant, et nous proposons d'intégrer l'IdO dans le processus métier BPEL en utilisant la sémantique offerte par les ontologies OWL, les concepts ontologiques fondamentaux requis pour une application IdO et les règles SWRL.

PARTIE II : CONCEPTION ET IMPLEMENTATION

Chapitre 5 : Conception

Introduction

Dans ce chapitre, nous commençons par la conception de notre ontologie, nous décrirons l'architecture générale de notre système et nous détaillons ensuite chaque module composant du système.

1. Conception de l'application

1.1. Première étape : Conception de l'ontologie

Dans cette section, nous présentons notre ontologie dédiée à la représentation des éléments d'un processus métier exécutable BPEL ainsi que les concepts d'IdO. Notre conception a été faite 1) suivant un ensemble de règle de transformation des éléments BPEL sous forme d'éléments OWL, 2) d'intégrer ensuite les concepts relatifs à l'IdO, et 3) de proposer un ensemble passerelles entre les deux hiérarchies [43].

1.2. Construction de la partie terminologique TBox :

1) **Transformation des éléments BPEL en termes d'OWL :**
   a) **Eléments de base :** Notre design est une amélioration du travail de master de [36], il suit toujours la règle suivante : chaque élément BPEL sera transformée en une classe OWL pour construire la partie TBox.

Les éléments de base du processus BPEL sont représentés comme OWL classes comme suit :

Elément BPEL	Représentation **en OWL**	**Représentation de ses attributs**
Process	Declaration (Class(:Process))	Declaration (DataProperty (:hasName)) DataPropertyDomain(:hasName :Process) DataPropertyRange (:hasName xsd :string)
PartnerLink	Declaration (Class(:PartnerLink))	Declaration (Class (:PartnerLinkType)) Declaration (Class (:MyRole)) Declaration (ObjectProperty (:hasPartnerLinkType)) ObjectPropertyDomain (:hasPartnerLinkType :PartnerLink) ObjectPropertyRange (:hasPartnerLinkType :PartnerLinkType) Declaration (ObjectProperty (:hasMyRole)) ObjectPropertyDomain (:hasMyRole :PartnerLink) ObjectPropertyRange (:hasMyRole :MyRole)
Variable	Declaration (Class(:Variable))	Declaration (DataProperty (:hasName)) DataPropertyDomain (:hasName :Variable) DataPropertyRange (:hasName xsd :string) Declaration (DataProperty (:hasMessageType)) DataPropertyDomain (:hasMessageType :Variable) DataPropertyRange (:hasMessageType xsd :string) Declaration (DataProperty (:hasElement) DataPropertyDomain (:hasElement :Variable) DataPropertyRange (:hasElement xsd :string) Declaration (DataProperty (:hasType)) DataPropertyDomain (:hasType:Variable)

		DataPropertyRange (:hasType xsd :string)
Activity	declaration (Class :Activity)	
Basic Activity	SubClassOf(: BasicActivity : Activity)) SubClassOF(:InvokeA ctivity :BasicActivity) SubClassOf (:ReceiveActivity : BasicActivity) SubClassOf(:ReplyAct ivity :BasicActivity)	Declaration(ObjectProperty(:hasPartnerLink)) ObjectPropertyDomain (:hasPartnerLink :BasicActivity) ObjectPropertyRange (:hasPartnerLink :PartnerLink) Declaration(ObjectProperty(:hasOperation)) ObjectPropertyDomain (:hasOperation :BasicActivity), ObjectPropertyRange (:hasOperation :Operation)
Structured Activity	SubClassOf (: StructuredActivity : Activity)	

Tableau 4.1 : *Représentation des éléments BPEL en OWL*

b) Représentation des activités BPEL en OWL

i) Les activités structurées

(1) L'activité if-then-else

L'activité if-then-else est définie comme une sous classe de la classe StructuredActivity. Elle permet d'exécuter un seul chemin selon la condition à vérifier. Pour répondre à cette définition, l'activité if-then-else est représentée par l'ensemble de déclarations suivantes :

Declaration (Class(: IfThenElseActivity))

SubClassOf (:IfThenElseActivity :StructuredActivity)

Declaration (Class(: IfCondition))

SubClassOf(:Condition)

Declaration (DataProperty (:hasValue))

DataPropertyDomain (:hasValue :IfCondition)

DataPropertyRange (:hasValue xsd :boolean)

ObjectExactCardinality(1 :hasBranchTrue :BrachTrue)

ObjectExactCardinality(1 :hasBranchFalse :BrachFalse)

Declaration (Class(:BranchTrue))
SubClassOf(:BranchTrue :Activity)
Declaration (Class(:BranchFalse))
SubClassOf(:BranchFalse :Activity).

(2) L'activité Séquence

L'activité « sequence » permet de définir un ensemble d'activités qui peuvent être exécutées séquentiellement. Il peut exister que la même activité peut être exécutée dans la séquence plusieurs fois. De ce fait, nous ajoutons une classe SequenceActivity comme une sous classe de la classe Activité ainsi qu'une propriété de donnée pour indiquer son ordre dans la séquence.

Declaration (Class (:Sequence))
SubClassOf (:Sequence :StructuredActivity).
Declaration (Class(:SequenceActivity))
SubClassOf(:SequenceActivity :Activity)
Declaration (DataProperty (:hasOrder))
FunctionalDataProperty(:hasOrder)
DataTypeRange (:hasOrder xsd :integer)

(3) Las activités répétitives

Dans BPEL, nous pouvons trouver trois types d'activités répétitives : while, repeatUntil et forEach. Nous avons représenté ces activités comme des sous classes de la classe StructuredActivity.

La condition de chacune de ces activité est représentée comme une sous classe de la classe Condition.

Declaration(Class(: Condition))
Declaration(Class(: WhileCondition))
SubClass (:WhileCondition :Condition)
Declaration(Class(: RepeatCondition))
SubClass (:RepeatCondition :Condition)
Declaration(Class(:ForECondition))
SubClass (:ForEachCondition :Condition))

(4) L'activité While

L'activité While permet d'exécuter une activité définie comme tant que la condition WhileCondition est évaluée à vrai :

Declaration(Class (:ChildWhileActivity)

SubClassOf(:ChildWhileActivity : Activity)

(5) L'activité for-each

L'activité forEach, peut être répétée séquentiellement N fois. Donc, nous ajoutons les déclarations suivantes pour spécifier que le nombre d'itérations est entre 1 et N fois :

Declaration (DataProperty (:hasValue))

DataPropertyDomain (:hasValue :ForEachCondition)

DataPropertyRange (:hasValue xsd :integer).

datatype restriction (:hasValue DataTypeRestriction(xsd :integer xsd :minExclusive "1"^^xsd :integer maxExclusive "N"^^xsd :integer)).

ObjectMinCardinality(1 :hasForEachSetActivity :ForEachSetActivity),

ObjectMaxCardinality(N : hasForEachSetActivity :ForEachSetActivity)

Declaration (Class (:ForEachSetActivity))

SubClassOf(:ForEachSetActivity :Activity).

(6) L'activité repeat-until

L'activité repeat-until est évaluée à la fin de chaque itération. Nous proposons les déclarations suivantes pour assurer que son corps est peut être exécuté au moins une fois :

Declaration (Class(: RepeatCondition))

SubClassOf(:Condition)

Declaration (DataProperty (:hasValue))

DataPropertyDomain (:hasValue :RepeatCondition)

DataPropertyRange (:hasValue xsd :boolean)

ObjectMinCardinality(1 :hasRepeatUntilBody :RepeatUntilBody)

Declaration Class(:RepeatUntilBody)

SubClassOf(:RepeatUntilBody :Activity).

2) Représentation des concepts d'IdO

Dans la deuxième étape, nous ajoutons les éléments relatifs à l'IdO :

Le tableau ci-dessous présente les trois concepts d'IdO requis pour notre application retenus à partie des concepts fondamentaux cités dans [41].

Concept IdO	Description du concept	Représentation en OWL
SensorDevice	représente un objet physique ou virtuel	Declaration (Class (:SensorDevice))
Location	représente la localisation d'un objet	Declaration (Class (:Location))
Observation	représente l'observation donnée par l'objet	Declaration (Class (:Observation))

Tableau 4.2 *: Représentation des concepts d'IdO et ses attributs*

Le tableau suivant illustre la liste des relations dans la hiérarchie des concepts d'IdO :

ObjectProperty	Description	Domain	Range
hasObservation	indique q'un objet peut avoir une observation	SensorDevice	Observation
hasLocation	indique qu'un objet a une localisation	SensorDevice	Location

Tableau 4.3 : *Liste des relations dans la hiérarchie des concepts d'IdO*

Nous présentons dans le tableau ci-dessous l'ensemble de propriétés de données des concepts d'IdO :

Classes	Dataproperty	Représentation en OWL	Description
SensorDevice	HasId	Declaration (DataProperty (:hasId)) FunctionalDataProperty(:hasID) DataTypeRange (:hasId xsd :integer)	Identifiant d'un objet
	hasType	Declaration (DataProperty (:hasType)) DataOneOf ("dynamic"^^xsd:string "static"^^xsd:string)	le type d'un objet peut être : dynamic ou bien static
	hasState	Declaration (DataProperty (:hasState)) DataOneOf ("internal"^^xsd:string	état d'un objet peut être interne

		"external"^^xsd:string)	ou externe
	hasUnit	Declaration (DataProperty (:hasUnit)) DataTypeRange (:hasUnit xsd :string)	l'unité de mesure de valeurs transmises par l'objet
Location	hasIdLocation	Declaration (DataProperty (:hasIdLocation)) DataTypeRange (:hasId xsd :integer)	identifiant de la location d'un objet
	hasName	Declaration (DataProperty (:hasName)) DataTypeRange (:hasName xsd :string)	nom de la location
Observation	hasValue	Declaration (DataProperty (:hasValue)) DataTypeRange (:hasValues xsd :float)	la valeur observée mesurée par l'objet
	hasTime	Declaration (DataProperty (:hasTime)) DataTypeRange (:hasTime xsd :date time)	le temps de prise d'une valeur

Tableau 4.4 : *Représentation des propriétés de données des concepts d'IdO*

3) Nous définissons dans le tableau ci-dessous l'ensemble de passerelles entre les deux hiérarchies précédentes :

Object Property	Description	Domain	Range
hasImpactOnProcess	indique qu'une observation a une influence sur un processus	Observation	Process

hasImpactOnVariable	indique qu'une observation a une influence sur une variable	Observation	Variable
hasImpactOnCondition	indique qu'une observation a une influence sur une condition	Observation	Condition
hasImpactOnActivity	indique qu'une observation a une influence sur une activité	Observation	Activity

Tableau 4.5 : *Description des liens entre l'hiérarchie des concepts de BPEL et d'IdO*

a) Extension de TBox avec les règles SWRL :

Dans notre travail, OWL ne permet pas de représenter les blocks d'activités BPEL. Comme nous avons déjà mentionné dans la problématique, notre objectif est de compléter la représentation des activités d'un processus exécutable BPEL avec un ensemble de règles SWRL :

Règle 1 : cette règle permet de compléter la représentation de l'activité if-then-else, elle permet de spécifier que si la valeur de condition d'une activité if est évaluée à vrai, dans ce cas l'activité associée à BranchTrue sera exécuté :

(R1) IfThenElseActivity(?x) ^ IfCondition(?y) ^ hasCondition(?x, ?y) ^ hasValue(?y, true) ^ BranchTrue(?z) ^ hasBranchTrue (?x, ?z)->BranchTrue(?z)

Règle 2 : Toujours dont le but de compléter l'activité if-then-else, la règle 2 permet de spécifier que si la valeur de condition d'une activité if est évaluée à faux, dans ce cas l'activité associée à BranchFalse sera exécuté :

(R2) IfThenElseActivity (?x)^IfCondition (?y)^hasCondition (?x, ?y)^hasValue (?y,"false")^BranchTrue(?z)^hasBranchTrue(?x, ?z) ->BrachFalse Variable (?z)

Règle 3 : Pour compléter la représentation de l'activité While, nous utilisons la règle SWRL suivante :

(R3) WhileActivity (?x) ^ WhileCondition (?y) ^ hasWhileCondition (?x, ?y) ^ hasValue (?y, true) ^ ChildWhileActivity (?z) ^ hasChildWhileActivity(?x, ?z) ->ChildWhileActivity(?z)

Règle 4 : Pour compléter la représentation de l'activité Repeat-until :

(R4) RepeatUntilActivity (?x) ^ RepeatCondition (?y) ^ hasCondition (?x, ?y) ^ hasValue (?y, false)

RepeatUntilBody (?z) ^ hasRepeatUntilBody (?x, ?z) ->RepeatUntilBody(?x)

Règle 5 : Pour compléter la représentation de l'activité For-Each, nous proposons la règle 5 :

(R5) ForEachActivity(?x) ^ ForEachCondition(?y) ^ hasForEachCondition(?x , ?y) ^ ForEachSetActivity(?z) ^ hasForEachSetActivity(?x , ?z) ^ hasValue(?y , value)^ integer [>=1,<=N] (value) ->ForEachSetActivity(?z)

Règle 6 : Pour déduire la liste des processus influencés par une observation

(R6) Process(?x) ^ Observation(?y) ^ Variable(?z) ^ hasVariable(?x, ?z) ^ hasImpactOnVariable(?y, ?z) ->hasImpactOnProcess(?y, ?x))

Règle 7 : Pour déduire la liste des activités influencées par une observation :

(R7) Variable(?x) ^ Observation(?y) ^ Activity(?z) ^hasInPutVariable(z,x)^hasImpactOnVariable (?y, ?x) -> hasImpactOnActivity (?y, ?z)

Règle 8 : Pour représenter l'effet d'une observation sur la valeur d'une condition, nous :

(R8) WhileActivity(?x) ^ Variable(?y) ^ haImactOnActivity (?y, ?x) ^ WhileCondition(?z) ^ hasValue (?z,true) ->hasImpactOnCondition(?y, ?z) ^ hasValue(?z,false)

1.3. Deuxième étape : architecture du système

Dans cette partie, nous proposons l'architecture suivante :

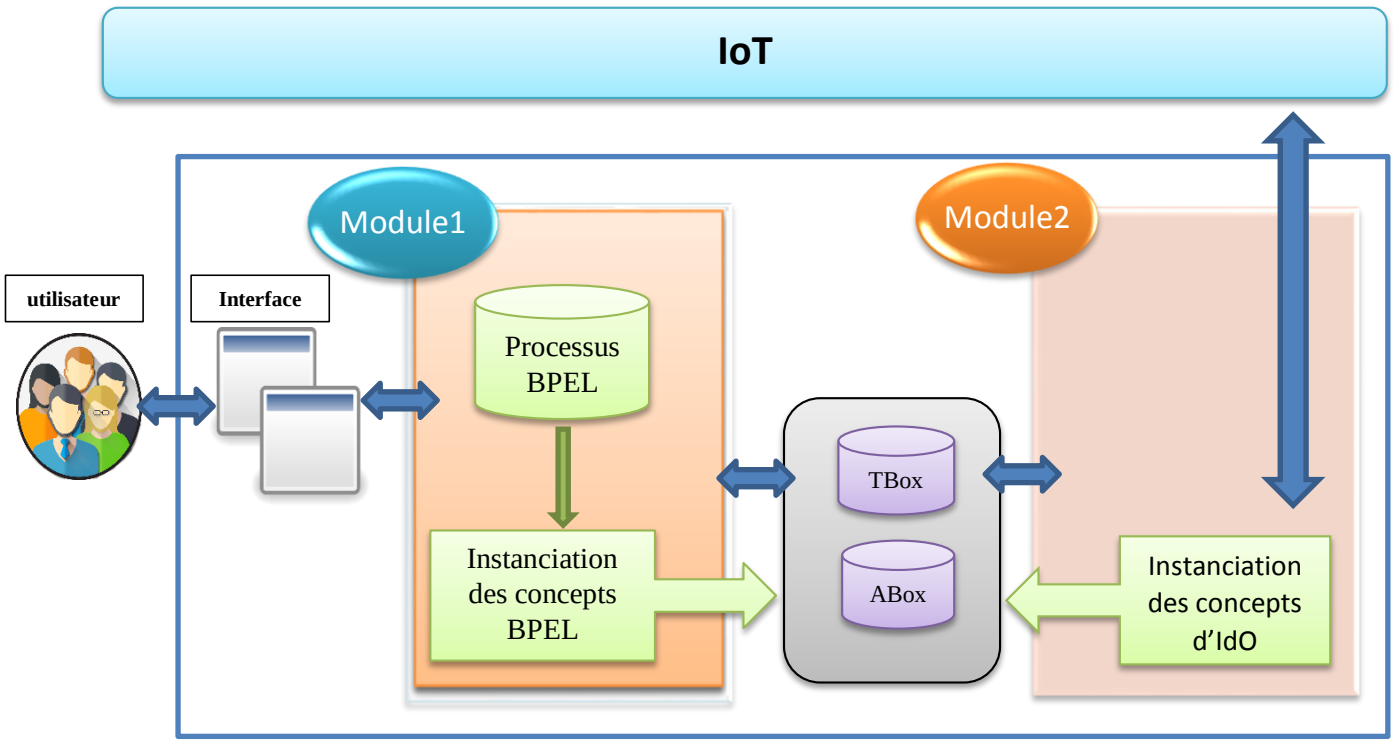


Figure 4.1 : *Architecture du système*

Cette architecture est composée de deux modules principaux :

Module 1 : Après la construction de l'ontologie, ce module permet de créer les instances des concepts BPEL. De ce fait, nous proposons une extension de l'algorithme proposé dans [36], et nous nous intéressons à l'instanciation des activités à partir d'un fichier BPEL (voir algorithme1).

Module 2 : Le 2éme module permet de capter les valeurs mesurées par les objets connectés et les insérer dans le fichier (voir Algorithme 2)

```
Algorithme Algo1;
Debut
…
1  Pour chaque Activity
2  Pour chaque BasicActivity
3  Si ReceiveActivity alors
4       Créer ClassAssertion(:ReceiveActivity :ReceiveActivityName);
5       Créer ObjectPropertyAssertion (:hasPartnerLink :ReceiveActivityName :PartnerLinkName);
6       Créer ObjectPropertyAssertion (:hasPortType :ReceiveActivityName :PortTypeName);
7       Créer ObjectPropertyAssertion (:hasInputVariable :ReceiveActivityName :VariableName);
8       Créer ObjectPropertyAssertion (:hasOutPutVariable :ReceiveActivityName :VariableName)
9  Sinon Si InvokeActivity thenalors
10      Créer ClassAssertion(:InvokeActivity :InvokeActivityName);
11      Créer ObjectPropertyAssertion (:hasPartnerLink :InvokeActivityName :PartnerLinkName);
12      Créer ObjectPropertyAssertion (:hasPortType :InvokeActivityName :PortTypeName);
13      Créer ObjectPropertyAssertion (:hasVariable :InvokeActivityName : VariableName)
14 Sinon Si ReplyActivity alors
15      Créer ClassAssertion(:ReplayActivity :ReplayActivityName);
16      Créer ObjectPropertyAssertion (:hasPartnerLink :ReplyActivityName :PartnerLinkName);
17      Créer ObjectPropertyAssertion (:hasOperation :ReplyActivityName :OperationName);
18      Créer ObjectPropertyAssertion (:hasPortType :ReplyActivityName :PortTypeName);
19      Créer ObjectPropertyAssertion (:hasVariable :ReplyActivityName : VariableName);
20 Fin SI;
21 Fin Pour,
22 Pour chaque StructuredActivity
23Si sequenceActivity alors
24      Créer ClassAssertion(:SequenceActivity :SequenceActivityName);
25 Pour chaque Activity de sequence
26      Créer ObjectPropertyAssertion (:hasActivity : SequenceActivityName :ActivityName);
27      Créer DataPropertyAssertion (:hasValue :ActivityName "order""^^xsd:integer);
28 Fin pour;
29 Sinon Si  IfThenElseActivity alors
30      Créer ClassAssertion(:IfThenElseActivity :IfThenElseActivityName);
31      Créer ObjectPropertyAssertion(:hasIfCondition :IfThenElseActivityName :IfConditionName);
32      Créer DataPropertyAssertion (:hasValue :IfConditionName "value"^^xsd:boolean);
33      Créer ObjectPropertyAssertion(:hasBranchTrue :IfThenElseActivityName :BranchTrueName);
34      Créer ObjectPropertyAssertion(:hasBranchFalse :IfThenElseActivityName :BranchFalseName)
35 Sinon SI WhileActivity alors
36      Créer ClassAssertion(:WhileActivity :WhileActivityName);
37      Créer ObjectPropertyAssertion(:hasWhileCondition :WhileActivityName :WhileConditionName);
38      Créer DataPropertyAssertion (:hasValue :WhileConditionName "value" "^^xsd:boolean);
39      Créer ObjectPropertyAssertion(:hasChildWhileActivity :WhileActivityName :ChildWhileActivityName)
40 Sinon SI RepeatUntilActivity alors
41      Créer ClassAssertion(:RepeatUntilActivity : RepeatUntilActivityName);
42      Créer ObjectPropertyAssertion(:hasRepeatCondition : RepeatUntilActivityName : RepeatlConditionName);
43      Créer DataPropertyAssertion (:hasValue :RepeatConditionName "value""^^xsd:boolean)
44      Créer ObjectPropertyAssertion(:hasRepeatUntilBody : RepeatUntilActivityName : RepeatUntilBodyName);
45 Sinon SI ForEachActivity alors
46      Créer ClassAssertion(:ForEachActivity : ForEachActivityName);
47      Créer ObjectPropertyAssertion(:hasForEachCondition : ForEachActivityName : ForEachConditionName);
48      Créer DataPropertyAssertion (:hasValue :ForEachConditionName "N""^^xsd:integer);
49      Créer ObjectPropertyAssertion(:hasForEachSetActivity :WhileActivityName : ForEachSetActivityName);
50 Fin SI
51 Fin pour
```

Algorithme 4.1 : *Algorithme d'instanciation pour les concepts BPEL.*

```
Algorithme Algo2;
Debut
…
1   Pour chaque notification (Id, type, state, idlocalisation, name, unit, value, time) détectée
2       Créer DataPropertyAssertion (:hasId :SensorDevice "Id""^^xsd:integer);
3       Créer DataPropertyAssertion (:hasType :SensorDevice "type""^^xsd:string);
4       Créer DataPropertyAssertion (:hasState :SensorDevice "type""^^xsd:string);
5       Créer DataPropertyAssertion (:hasIdLocalisation :Localisation "idlocalisation""^^xsd:integer);
6       Créer DataPropertyAssertion (:hasName :Localisation "name""^^xsd:string);
7       Créer DataPropertyAssertion (:hasUnit :Observation "unit""^^xsd:string);
7       Créer DataPropertyAssertion (:hasValue :Observation "value""^^xsd:integer);
8       Créer DataPropertyAssertion (:hasTime :Observation "time""^^xsd:date);
9      Créer ObjectPropertyAssertion (:hasLocation :SensorDevice :Location);
10      Créer ObjectPropertyAssertion (:hasObservation :SensorDevice :Observation);
Fin pour;
Fin.
```

Algorithme 4.2 : *Algorithme d'instanciation des concepts d'IdO*

Conclusion

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

Chapitre 6 : Implémentation

Introduction

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

1. Le langage et les outils de développements :

1.1.JAVA

Java est un langage de programmation et une plate-forme informatique qui ont été créés par Sun Microsystems [44].

Java a les caractéristiques suivantes : rapidité, sécurité et fiabilité

1.2. L'outil de développement Netbeans

Cet IDE a été créé à l'initiative de Sun Microsystems. Il présente toutes les caractéristiques indispensables à un environnement de qualité.

NetBeans est Open Source, il permet de développer et déployer rapidement et gratuitement des applications graphiques Swing, des Applets, des JSP Servlets, des architectures J2EE, dans un environnement fortement personnalisable. L'IDE NetBeans repose sur un noyau robuste, la plateforme NetBeans, que vous pouvez également utiliser pour développer vos propres applications Java, et un système de plugins performant, qui permet d'avoir un IDE modulable.

Enfin cet IDE possède un débogueur de grande qualité ainsi qu'une interface graphique améliorée [45].

1.3. Protégé

ROTEGE-OWL 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 [46].

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.

1.4. 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 ou encore N3 et de raisonner sur des modèles ontologiques à l'aide de moteurs d'inférences inclus dans Jena ou externes.

Ainsi, Jena propose des systèmes permettant d'assurer la persistance des modèles. Voilà deux systèmes :- Jena SDB, un magasin de triplets utilisant une base de données relationnelle pour fonctionner. -Jena TDB, un système de stockage natif (c'est-à-dire qu'il utilise son propre système de stockage), reconnu comme étant plus performant que son homologue [47].

1.5. Arduino

Arduino a été fondé par un groupe d'étudiants Italiens en 2005.

Le système Arduino est conçu d'une plateforme Open Source installée sur une carte programmée à un microcontrôleur permettant l'écriture, la compilation et le test d'un programme. Les cartes et modules Arduino sont conçus avec des entrées/sorties qui reçoivent des signaux de capteurs.

Le logiciel de développement d'Arduino qui est Open-Source et Open-Hardware, est téléchargeable gratuitement. La programmation Arduino contient un langage d'implémentation Wiring [48].

Une carte Arduino est une interface programmable capable de piloter des capteurs et des actionneurs afin de créer des systèmes automatisés. Elle peut stocker un programme et le faire fonctionner [49].

1.6. Le capteur DHT22 : c'est un élément électronique qui capte la température et l'humidité.

2. L'implémentation

2.1. La création d'ontologie dans Protégé

Nous avons utilisé Protégé 5.5.0 pour éditer notre ontologie. La figure suivante représente l'édition du concept Activity : (voir figure 5.1)

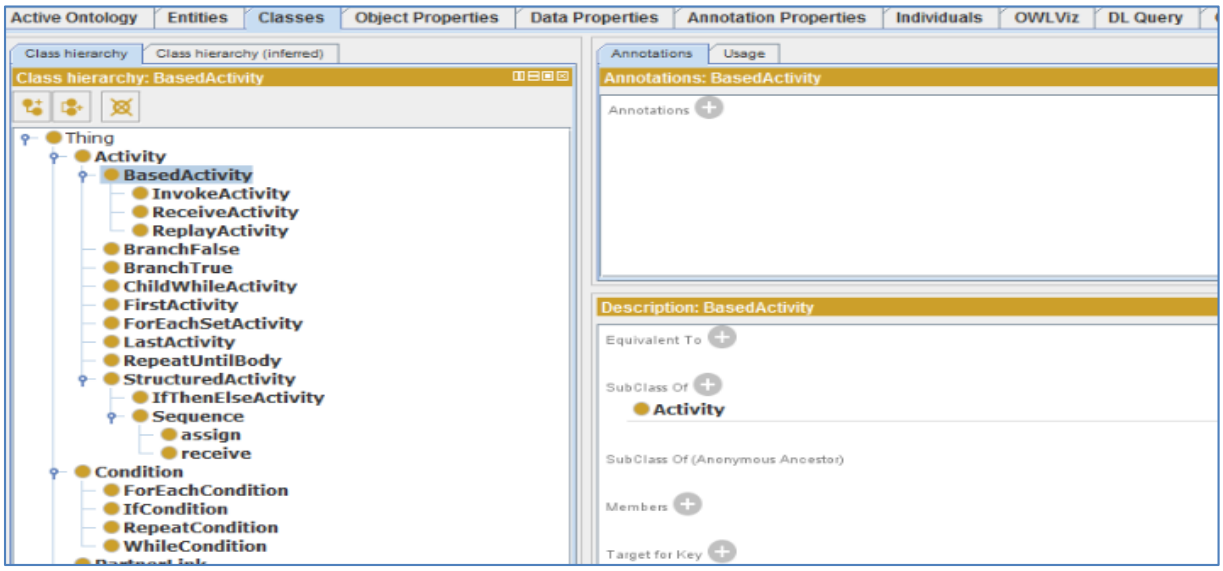


Figure 5.1 : *Edition du concept Activity dans Protégé.*

La figure suivante représente la création des concepts proposés pour l'IdO. (Voir figure 5.2)

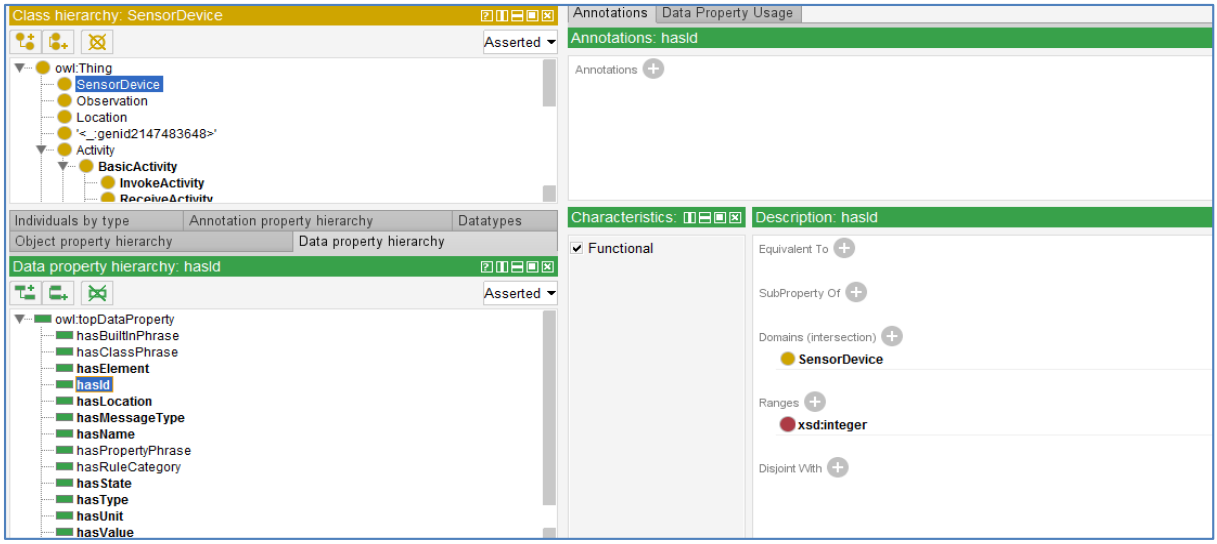


Figure 5.2: *Edition des concepts d'IdO.*

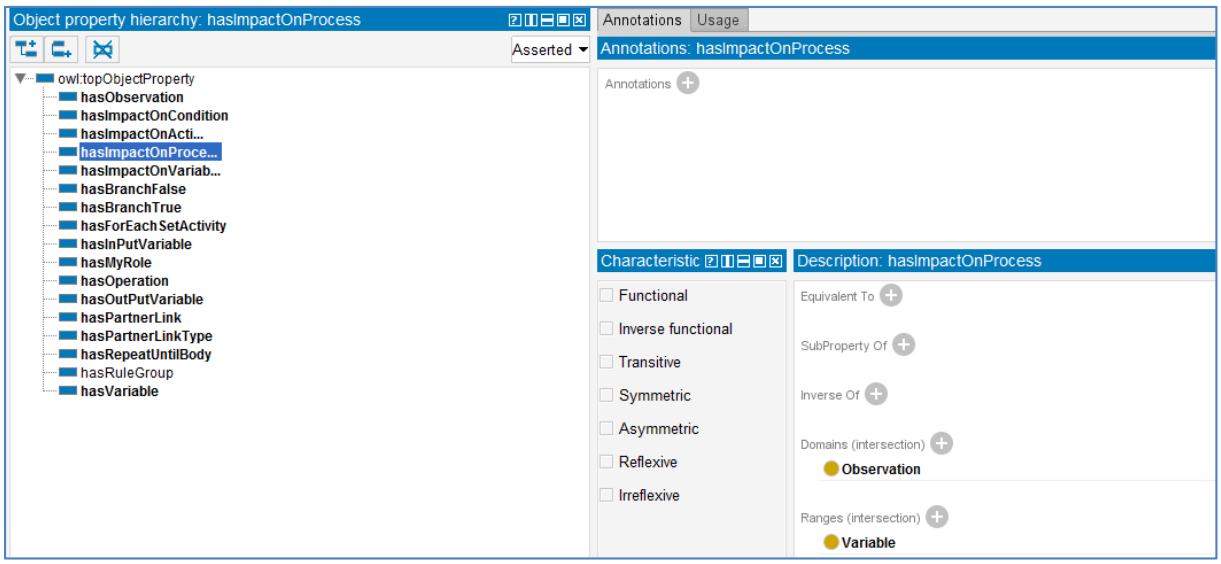


Figure 5.3 : *l'ensemble de passerelle entre les deux hiérarchies des concepts PBEL et IdO.*

Cette figure représente l'ensemble de passerelles entre les hiérarchies des concepts PBEL et hiérarchies IdO.

2.2. L'édition des règles SWRL dans protegé

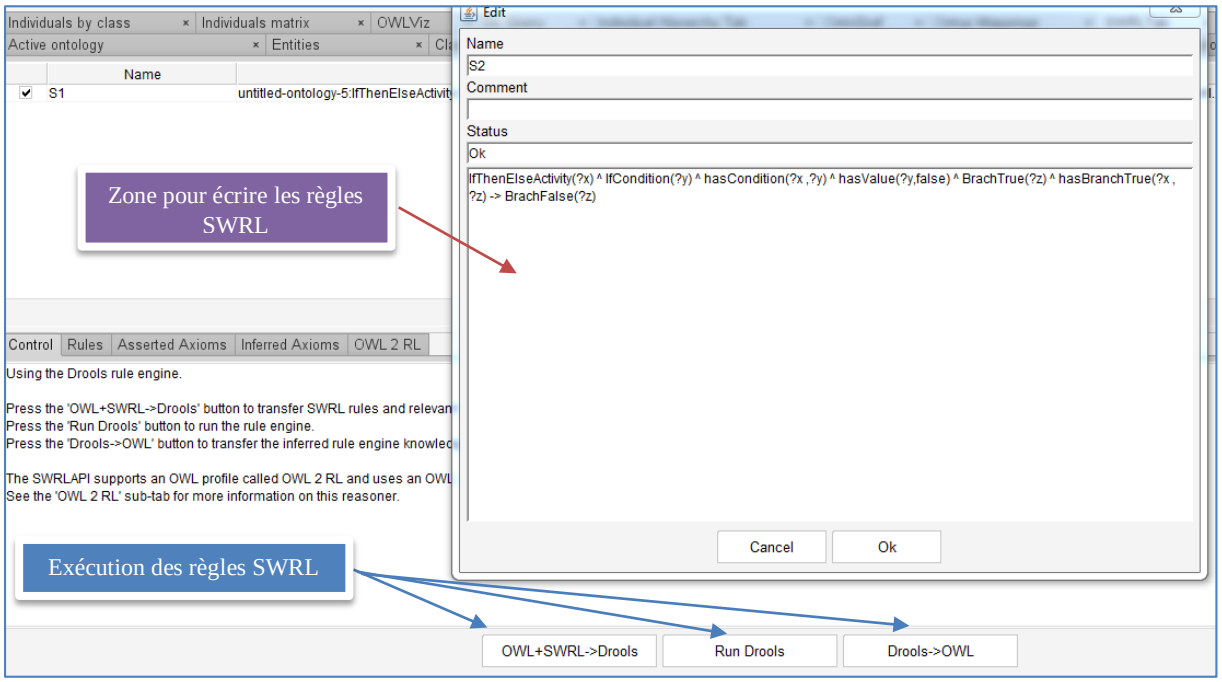


Figure 5.4 : *La création des règles SWRL*

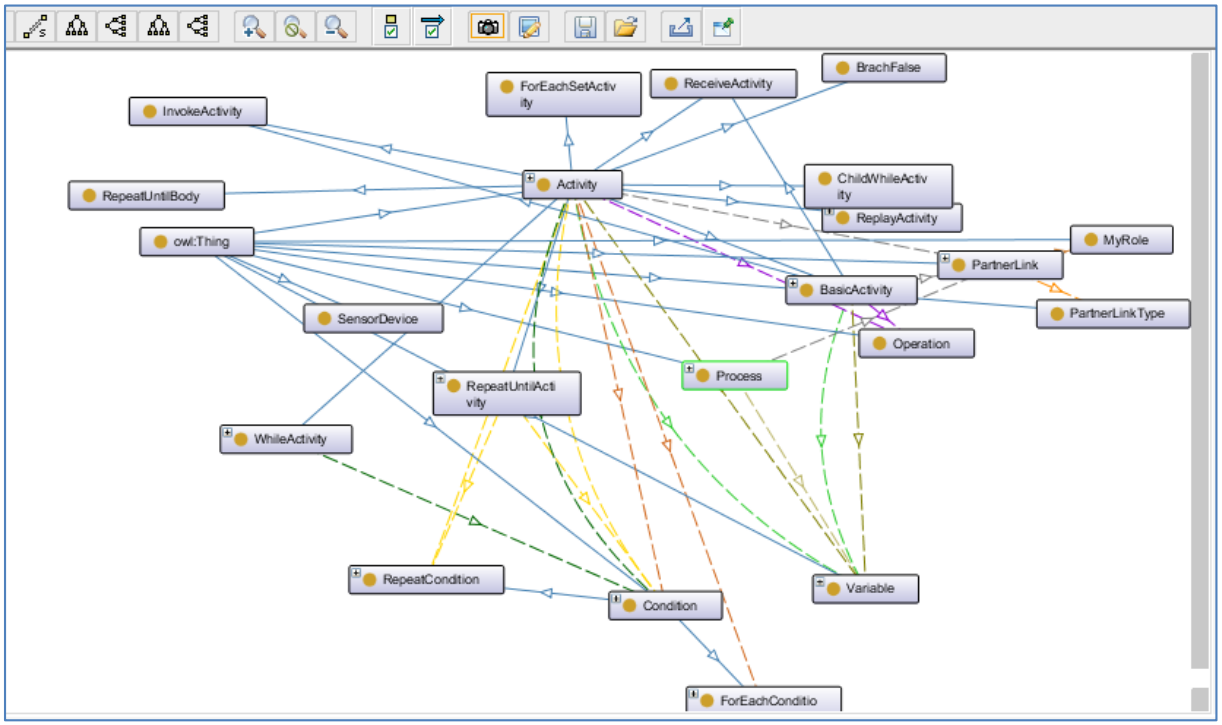


Figure 5.5 : *Les concepts sous forme d'un graph*

2.3. Programme Arduino pour capter les données de DHT22

Pour extraire les données transmises depuis le capteur DHT22 vers la carte Arduino on a utilisé ce code Arduino en langage C : (voir figure 5.6)

```
#define DHT22_PIN 2

void setup () {
  Serial.begin (115200);
  Serial.println ("DHT TEST PROGRAM ");
  Serial.print ("LIBRARY VERSION: ");
  Serial.println (DHT_LIB_VERSION);
  Serial.println ();
  Serial.println ("Type,\tstatus,\tHumidity (%),\tTemperature (C)");
}

void loop() {
  // READ DATA
  Serial.print ("DHT22, \t");
  int chk = DHT.read22 (DHT22_PIN);
  switch (chk) {
    case 0:  Serial.print ("OK,\t"); break;
    case -1: Serial.print ("Checksum error,\t");
    case -2: Serial.print ("Time out error,\t");
    default: Serial.print ("Unknown error,\t");
  }

  // DISPLAT DATA
  Serial.print (DHT.humidity,1);
  Serial.print (",\t");
  Serial.println (DHT.temperature,1);

  delay (1000);
}
```

```
Humidity: 45.10 %, Temp: 29.60 Celsius
Humidity: 45.10 %, Temp: 29.60 Celsius
Humidity: 45.10 %, Temp: 29.60 Celsius
Humidi                              sius
Humidi                              sius
Humidity: 45.20 %, Temp: 29.50 Celsius
Humidity: 45.40 %, Temp: 29.60 Celsius
Humidity: 45.50 %, Temp: 29.50 Celsius
Humidity: 45.50 %, Temp: 29.50 Celsius
Humidity: 45.50 %, Temp: 29.50 Celsius
Humidity: 45.50 %, Temp: 29.50 Celsius
```

Résultat du programme

Figure 5.6 : *Programme pour détecter les données depuis DHT22*

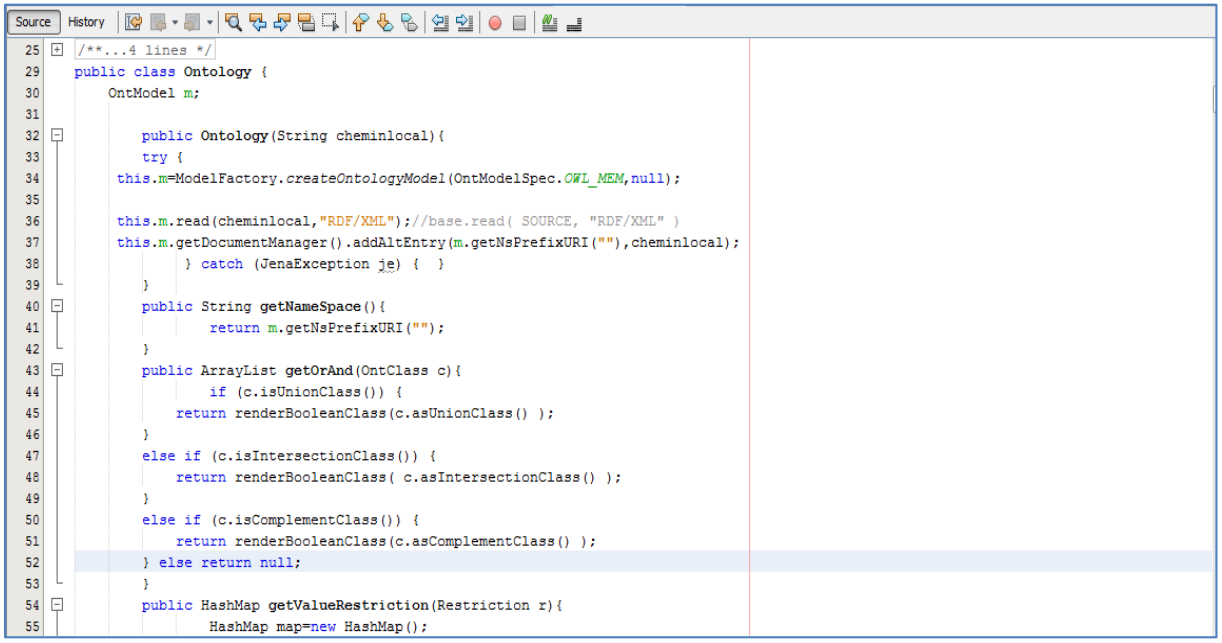
2.4. La connexion d'Arduino avec java :

```
////////////////connexion entre arduinoPort et java Netbeans///////////////
    public class PortReader implements SerialPortEventListener {
    @Override
    public void serialEvent(SerialPortEvent event) {
        if(event.isRXCHAR() && event.getEventValue() > 0) {
            try {
                do{
                String receivedData = ArduinoPort.readString(event.getEventValue());
                String[] splited = receivedData.split(" ");
                String text = arearx.getText();
                arearx.setText(text + receivedData);
                arearx.setCaretPosition(arearx.getDocument().getLength());
                  }
                while(arearx.getText().equals(""));
               }
             catch (SerialPortException ex) {
                System.out.println("Error in receiving string from COM-port: " + ex);
             }
        }
     }
}
```

Figure 5.7 : *Code java pour faire la connexion avec Arduino*

2.5. L'intégration d'ontologie avec notre application :

La figure suivante représente un code java pour intégrer notre ontologie dans notre application à l'aide de la bibliothèque Jena :



```
Source  History
25  /**...4 lines */
29  public class Ontology {
30      OntModel m;
31
32          public Ontology(String cheminlocal){
33          try {
34       this.m=ModelFactory.createOntologyModel(OntModelSpec.OWL_MEM,null);
35
36       this.m.read(cheminlocal,"RDF/XML");//base.read( SOURCE, "RDF/XML" )
37       this.m.getDocumentManager().addAltEntry(m.getNsPrefixURI(""),cheminlocal);
38              } catch (JenaException je) {  }
39          }
40          public String getNameSpace(){
41                  return m.getNsPrefixURI("");
42          }
43          public ArrayList getOrAnd(OntClass c){
44                  if (c.isUnionClass()) {
45              return renderBooleanClass(c.asUnionClass() );
46          }
47          else if (c.isIntersectionClass()) {
48              return renderBooleanClass( c.asIntersectionClass() );
49          }
50          else if (c.isComplementClass()) {
51              return renderBooleanClass(c.asComplementClass() );
52          } else return null;
53          }
54          public HashMap getValueRestriction(Restriction r){
55                  HashMap map=new HashMap();
```

Figure 5.8 : *Code d'intégration java avec ontologie.*

2.6. L'interface de notre application :

Les figures suivantes représentent les différentes interfaces de notre application :

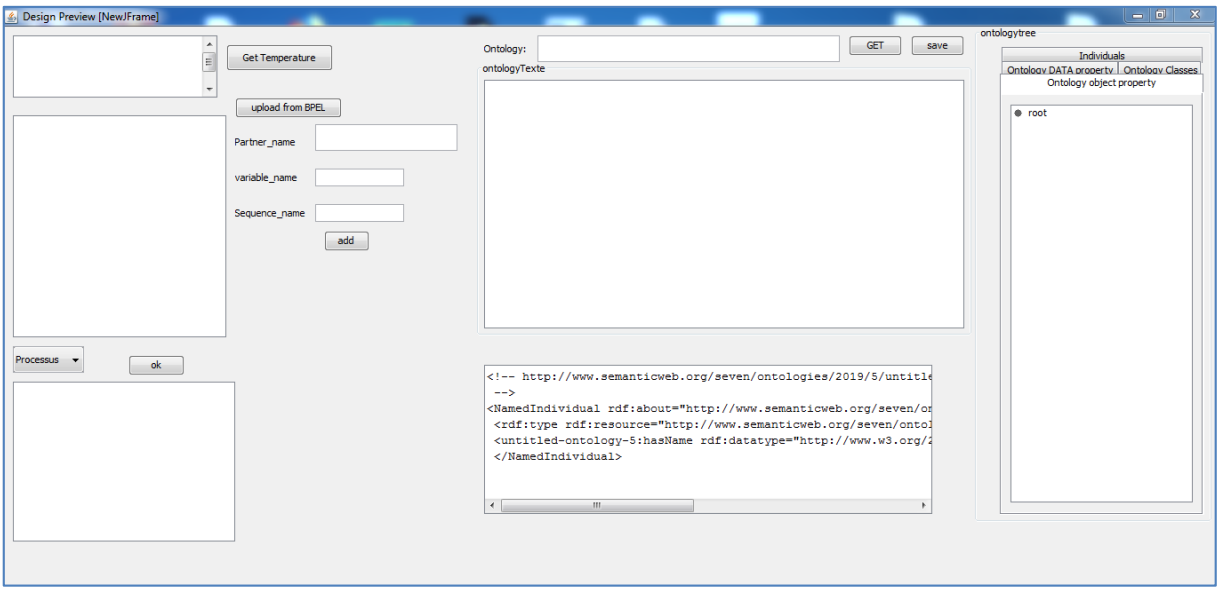


Figure 5.9 : *Capture sur l'interface*

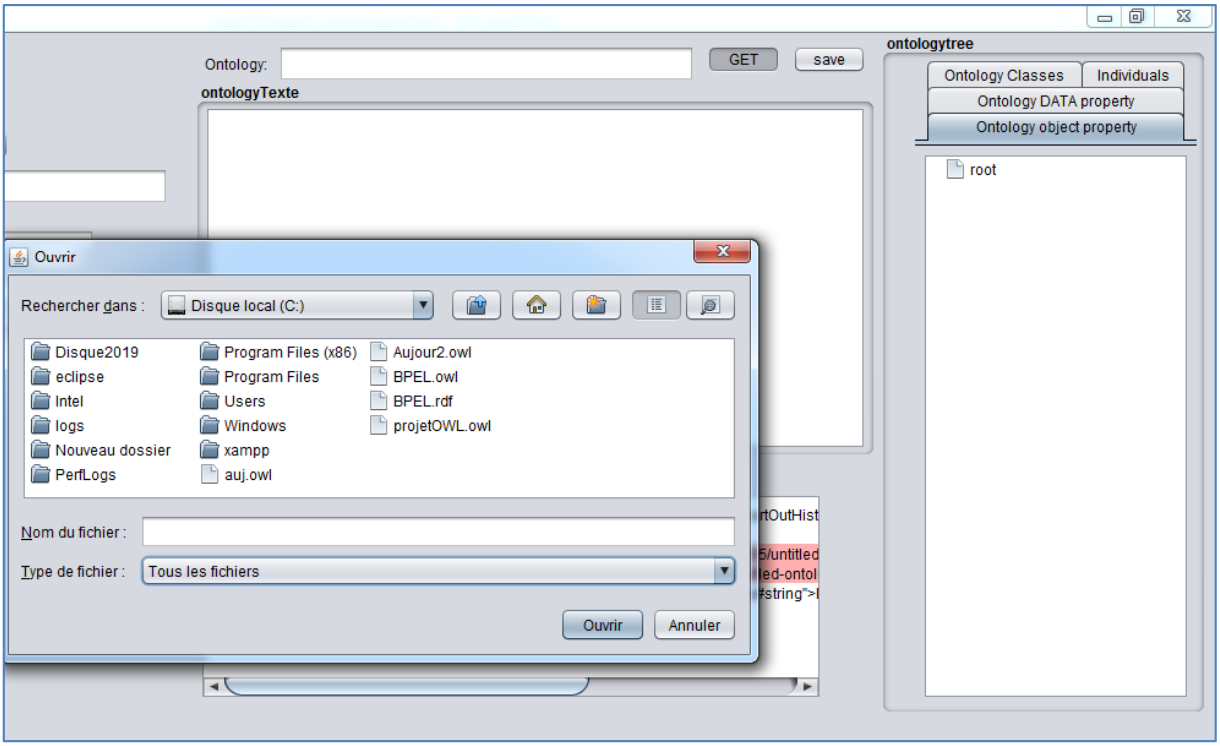


Figure 5.10*: Parcourir des fichiers OWL*

```
////////////////////////////// lecture de l'ontology texte

final NewJFrame owner=this;
JFileChooser jfc=new JFileChooser(".");
                                jfc.setFileFilter(new FileNameExtensionFilter(".owl",".rdf"));

                                if ( JFileChooser.APPROVE_OPTION == jfc.showOpenDialog(owner) ) {
                                        String filePath=jfc.getSelectedFile().getAbsolutePath();
                                        System.out.println("Load ontology "+filePath);


                                        try{
                                            FileReader reader=new FileReader(filePath);
                                            BufferedReader br=new BufferedReader(reader);

                                            queryListe1.read(br,null);

                                            //ueryListe1.getText()+"  "+i);

                                            br.close();
```

Figure 5.11 : *Code utilisé pour parcourir les fichiers OWL*

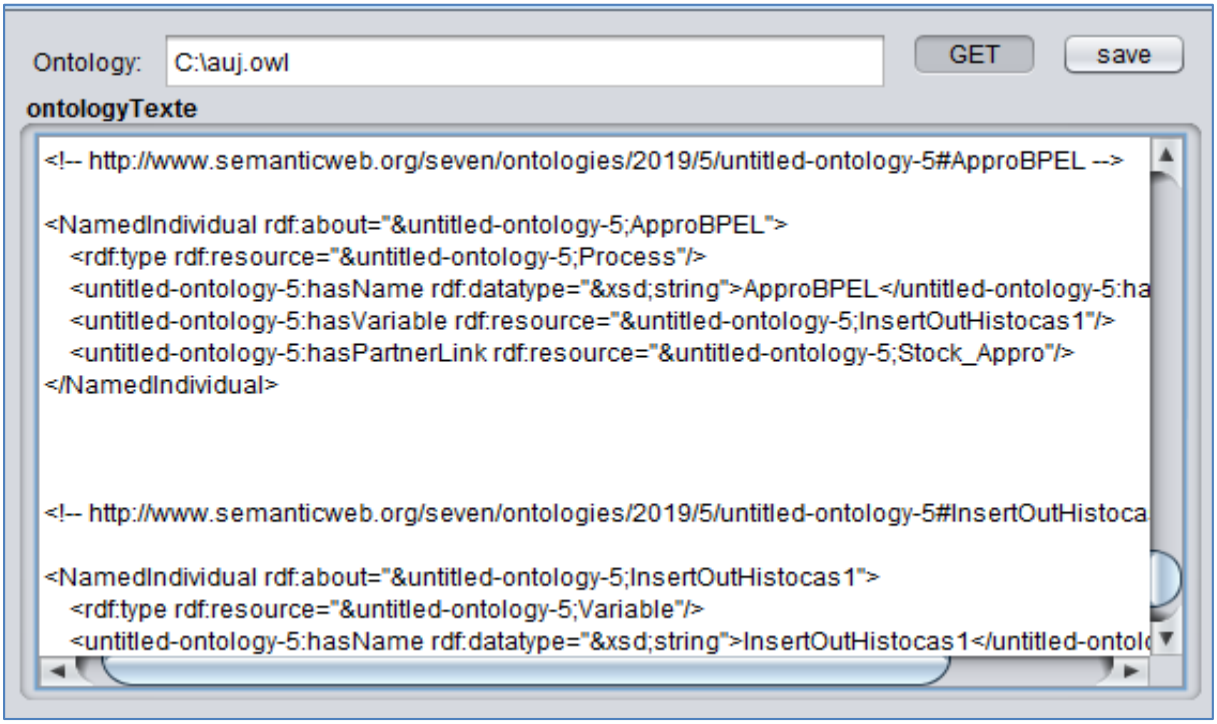


Figure 5.12 : *L'affichage du fichier OWL*

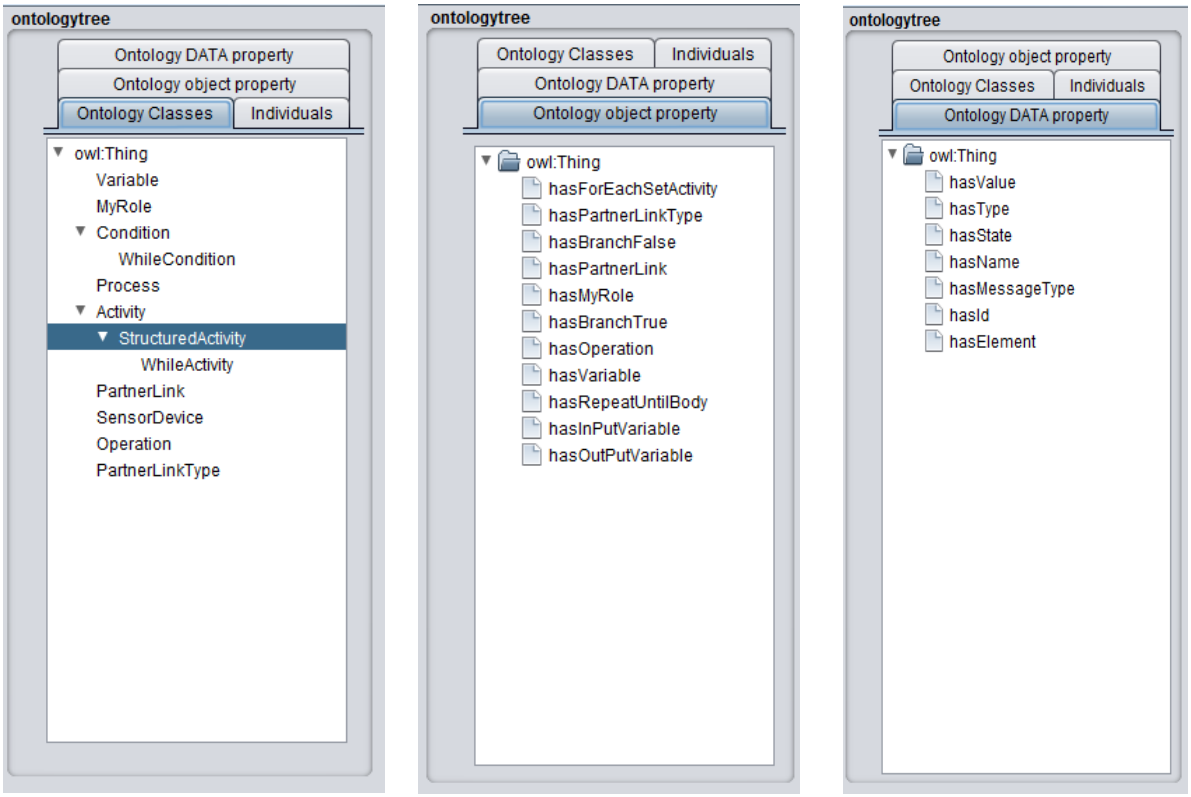


Figure 5.13 : *L'affichage sous forme d'un arbre*

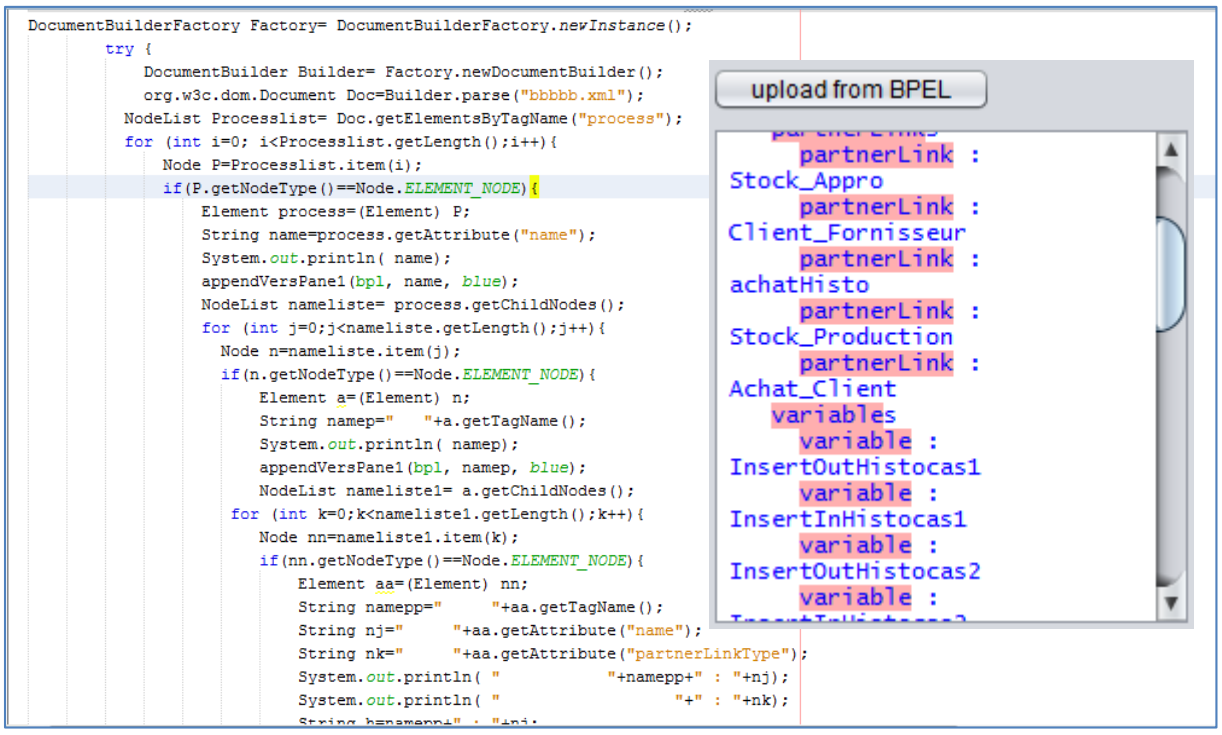


Figure 5.14 : *Code java pour afficher et extraire les données depuis le fichier BPEL*

```
private void addActionPerformed(java.awt.event.ActionEvent evt) {
    OntModel m = ModelFactory.createOntologyModel( OntModelSpec.OWL_MEM, null );
        FileManager.get().readModel( m, "C:/07.owl" );
        String xmlbase = "http://www.semanticweb.org/seven/ontologies/2019/5/untitled-ontology-5";

        OntClass al= m.getOntClass("partnerLink");
                Individual Client_Fornisseur=m.createIndividual(Partner_name.getText(), al);
        OntClass al1= m.getOntClass("Variable");
       Individual var=m.createIndividual(variable_name.getText(), al1);

//
```

Figure 5.15 : *Instanciation des concepts*

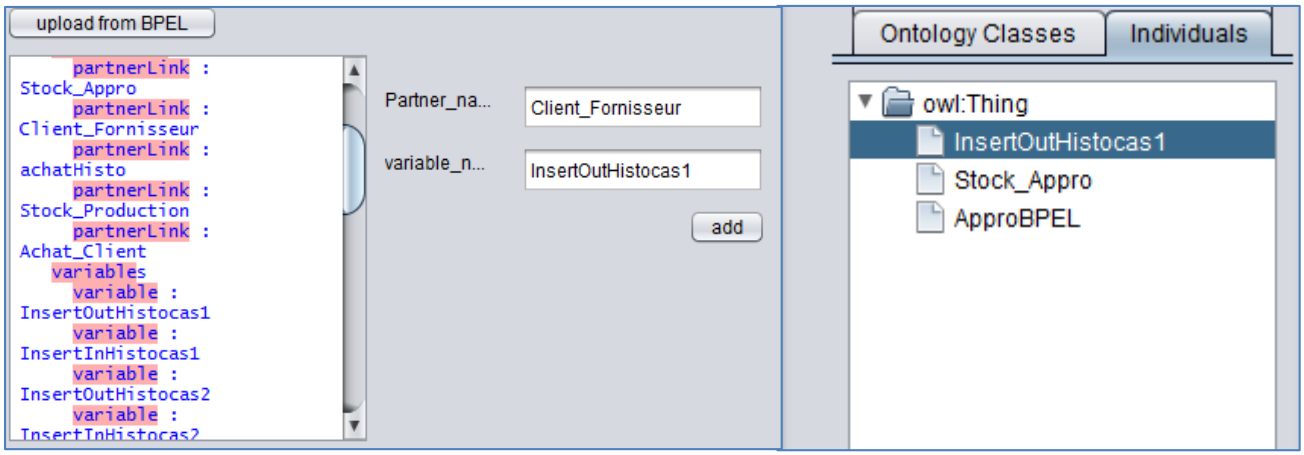


Figure 5.16 : *résultat affiché sur l'interface*

Et pour récupérer les données depuis la carte Arduino et les afficher sur l'interface on a besoin de code java suivant :

```
        // TODO add your handling code here:
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        if(jButton2.getText().equals("Get Temperature")){
        ArduinoPort = new SerialPort("COM3");
        try {
                ArduinoPort.openPort();
                ArduinoPort.setParams(SerialPort.BAUDRATE_9600
                                      SerialPort.DATABITS_8,
                                      SerialPort.STOPBITS_1,
                                      SerialPort.PARITY_NONE,
                                      false,
                                      true);
                ArduinoPort.setFlowControlMode(SerialPort.FLOWCONTROL_NONE);

                ArduinoPort.addEventListener(new PortReader(), SerialPort.MASK_RXCHAR);
               jButton2  .setText("Deconnecter");
                }
            catch (SerialPortException ex) {
                System.out.println("There are an error on writing string to port т: " + ex);
            }
        }  // TODO add your handling code here:
    }
```

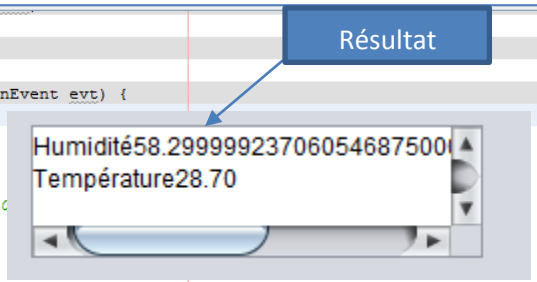


Figure 5.17 : *Le code et l'affichage des données du sensor data sur l'interface.*

Conclusion

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

CONCLUSION GENERALE

Dans ce mémoire, nous avons présenté notre proposition d'intégrer l'IdO dans un processus métier exécutable décrit en BPEL en utilisant les ontologies OWL et les règles SWRL.

Dans la partie conception, nous avons proposé une ontologie OWL qui permet de représenter les différents éléments d'un processus exécutable BPEL ainsi que les concepts relatifs à l'IdO. Les deux hiérarchies ont une racine commune « Thing ».

Elles sont reliées entre eux par des relations ''Object properties''. La représentation a été complétée par un ensemble de règles SWRL pour représenter les différentes activités et les différents effets des données issues d'IdO sur les concepts BPEL.

Nous avons concentré sur la représentation des activités, la partie la plus intéressante dans un processus métier et la plus influencée par les données transmises par l'IdO.

L'utilisation des règles SWRL impose l'instanciation d'une ontologie, de ce fait, nous avons aussi proposé deux simples algorithmes pour instancier notre ontologie afin d'utiliser ces règles.

Dans la phase d'implémentation, nous avons présenté les détails de réalisation de notre proposition, où nous avons édité la partie Tbox, ajouté les règles SWRL, connecté un ensemble d'objets pour mesurer la température et l'humidité, et implémenter la partie qui permet d'instancier l'ontologie et de créer l'Abox.

Comme perspectives, nous envisageons d'améliorer le travail engagé et d'ajouter un autre module « raisonnement sur Abox » qui permet de déduire des nouvelles connaissances, à partir des données d'IdO mesurées pour aider les utilisateurs métier.

REFERENCES

[1] F.Baader, D.Calvanese, D.L.McGuinness, D.Nardi, P.F.Patel-Schneider, Eds, «The Description Logic Handbook: Theory, Implementation, and Applications,» Description Logic Handbook, 2003.

[2] A.-S. Feyaerts, «Raisonnement sur les ontologies spatiales,» université libre de bruxelle, 2010.

[3] A. Yang LI, «OWL : Web Ontology Language,» 2006.

[4] W. Triaa, «Gestion agile de processus métier : proposition d'une approche tirée par les compétences,» Automatique / Robotique, 2018.

[5] B. M. P. S. Matjaz Juric, *Business Process Execution Language for Web Services,* 2nd Edition, 2006.

[6] J.-F. Pillou, «High-Tech,» 19 Avril 2011. [En ligne]. Available: https://www.commentcamarche.net/contents/306-bpm-business-process-management#introduction-au-bpm.

[7] «Haroun Ben Hmida's Blog,» [En ligne]. Available: https://harounbenhmida.wordpress.com/2013/04/16/quest-ce-que-le-bpm/.

[8] M. A. SELSOULI, « TEST UNITAIRE DE PROCESSUS BPEL : GÉNÉRATION ORIENTÉE CHEMINS DE CAS DE TEST MÉMOIRE PRÉSENTÉ,» MONTRÉAL, 2010.

[9] P. Mayer, «Design and Implementation of a Framework for Testing BPEL Compositions,» 2006.

[10] Mickaël Baron, «slideshare,» 28 mars 2011. [En ligne]. Available: https://fr.slideshare.net/baronm/bpel-7410586.

[11] F. ABOUZAID, «Analyse Formelle D'orchestration De Services Web,» 2010.

[12] E. e. a. Christensen, «Web Services Description Language (WSDL),» 2001.

[13] A. CHAMI, «VÉRIFICATION DE PROCESSUS BPEL À L'AIDE DE PROMELA-SPIN,» MONTRÈAL, 2008.

[14] M.-C. M. Dumas, « Chapitre 4 Les Services Web. In Intergiciel et Construction d'Applications R´eparties,» 2008.

[15] D.Evans, «The internet of things: How the next evolution of the internet is changing everything,» chez *C*, 2011.

[16] D. Lavoine, «DIGORA,» [En ligne]. Available: https://www.digora.com/fr/blog/definition-iot-et-strategie-iot.

[17] «OPENUMERIC,» [En ligne]. Available: https://www.openumeric.com/services/iot/.

[18] P.-J. Benghozi, S. Bureau, F. Massit-Folléa, C. Waroquiers et S. Davidson, «L'internet des objets: quels enjeux pour l'Europe,» n° %166, 2009.

[19] I. Saleh, «information et communication,» ISTE Ltd, 26 février 2018. [En ligne]. Available: https://www.openscience.fr/Internet-des-Objets-IdO-Concepts-Enjeux-Defis-et-Perspectives.

[20] H. Ali, «implémentation d'un protocole d'élection d'un serveur d'authentification dans l'internet des objets,» Bejaïa, 2016/2017.

[21] Ooreka, «Technique et solution,» [En ligne]. Available: https://rfid.ooreka.fr/comprendre/systeme-rfid.

[22] J. A. Stankovic, «Wireless sensor networks,» vol. 41, n° %110, pp. 92-95, 2008.

[23] S. Rabeb, «Modèle collaboratif pour l'Internet of Things (IoT),» Mai 2016.

[24] Gubbi, a. J. Gubbi, R. Buyya, S. Marusic et M. Palaniswami, «Internet of Things (IoT): A vision, architectural elements, and future directions,» vol. 29, n° %17, pp. 1645-1660, 2013.

[25] Gruber T, «A translation approach to portable ontologies,» *KnowledgeAcquisition, p. 199–220. 15*.

[26] L. Amir, «La Génération Automatique des Ontologies à partir des Diagrammes de classes UML,» 2017.

[27] N. Zouggar, B. Vallespir et D. Chen, «Enrichissement de la modélisation d'entreprise par les ontologies,» 2006.

[28] S. BOUARROUDJ, «Raisonnement sur une ontologie enrichie par des regles SWRL pour la recherche sémantique d'images a notées,» 2009-2010.

[29] S. A. Ghafour, «Méthodes et outils Pour l'intégration des ontologies,» 2003-2004.

[30] A. Galopin., «Modélisation ontologique des recommandations de pratique clinique pour une aide à la décision à niveaux d'abstraction variables.,» *Bio-informatique [q-bio.QM],* 2015.

[31] «Jonathan Vigneron,» *Contribution des ontologies à la création de bases de connaissances pour la maitrise des conformités réglementaires en santé, sécurité au travail et environnement,* 2013.

[32] Mendling, J. C. Recker and J., «On the Translation between BPMN and BPEL: Conceptual Mismatch between Process Modeling Languages,» chez *18th International Conference on Advanced Information Systems Engineering, Proceedings of Workshops and Doctoral Consortiums*, 2006.

[33] D. Domingos, F. Martins, C. Cândido, and R. Martinho, *Internet of Things Aware {WS-*

BPEL} Business Processes Context Variables and Expected Exceptions, vol. 20, 2014.

[34] A. Filipowska, M. Kaczmarek, M. Kowalkiewicz, X. Zhou, M. Born, *Procedure and guidelines for evaluation of {BPM} methodologies,* vol. 15, Bus. Proc. Manag. J., 2009.

[35] M. A. Aslam, «Towards integration of business processes and semantic web services,» 2008.

[36] Harath Ali, «BPEL Ontology : Une ontologie pour le processus BPEL,» guelma, 2017.

[37] G. Bajaj, R. Agarwal, P. Singh, N. Georgantas, V. Issarny, «A study of existing Ontologies in the IoT-domain,» 2017.

[38] a. R. Agarwal, «Unified IoT ontology to enable interoperability and federation of testbeds,» 2016.

[39] P. N. J. Ye, L. Coyle, S. Dobson,, «Ontology-based Models in Pervasive Computing Systems,» 2007.

[40] O. C. M. Compton, P. Barnaghi, L. Bermudez, R. Garcia-Castro and et al. S. Cox, J. Graybeal, M. Hauswirth, C. Henson, A. Herzog, «The SSN Ontology of the W3C Semantic Sensor Network Incubator Group, Web Semantics: Science, Services and Agents on the World Wide Web».

[41] L. J. J. Matheus C.J., Baclawski K., Kokar M.M, «Using SWRL and OWL to capture domain knowledge for a situation awareness application applied to a supply logistics scenario,» chez *the 1st International Conference on Rules and Rule Markup Languages for the Semantic Web*, Galway, Ireland..

[42] P. C. Zhaoyu Zhai, José-Fernán Martínez Ortega, Néstor Lucas Martínez,, *A Rule-Based Reasoner for Underwater Robots Using OWL and SWRL,* vol. 18(10), 2018.

[43] L.Djakhdjakha, D.Boukara, M.Hemam, Z.Boufaida, «An Extended Business Process Representation for Integrating IoT Based on SWRL/OWL,» chez *International Conference on Artificial Intelligence and Applied Mathematics in Engineering (ICAIAME 2019)*, Antalya, Manavgat, 2019.

[44] [En ligne]. Available: https://www.java.com/fr/download/faq/whatis_java.xml.

[45] [En ligne]. Available: https://www.oracle.com/tools/technologies/netbeans-ide.html.

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

[47] [En ligne]. Available: https://web-semantique.developpez.com/faq/?page=jena.

[48] «arduino-france,» [En ligne]. Available: https://www.arduino-france.com/tutoriels/quest-ce-que-arduino/.

[49] Dihia., MEGTIT Tedjini. DAHMANE, «Réalisation d'une serre agricole intélligente et contrôlable à distance par Internet,» 2017-2018.

[50] Z. SELLAMI, «Gestion dynamique d'ontologies à partir de textes par systèmes multi-agents adaptatifs,» 2012.

[51] R. Djedidi, «Approche d'évolution d'ontologie guidée par des patrons de gestion de changement.,» 2009.