Gestion intelligente de réseaux de capteurs, intégrés à des vêtements sportifs instrumentés

Samya Sagar

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

THESE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPERIEURE MINES-TELECOM ATLANTIQUE
BRETAGNE PAYS DE LA LOIRE - IMT ATLANTIQUE
COMUE UNIVERSITE BRETAGNE LOIRE

ECOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication

Spécialité : Informatique

Par
Samya SAGAR

Gestion intelligente de réseaux de capteurs, intégrés à des vêtements sportifs instrumentés

Thèse présentée et soutenue à IMT Atlantique – campus de Brest, le 28 janvier 2019
Unité de recherche : Lab-STICC
Thèse N° : 2019IMTA0129

Rapporteurs avant soutenance :

Sylvie DESPRES	Professeur des universités, Université Paris 13
Amel BOUZEGHOUB	Professeur, Télécom SudParis

Composition du Jury :

Président :	Gilles COPPIN	Professeur, IMT Atlantique
Rapporteurs :	Sylvie DESPRES	Professeur, Université Paris 13
	Amel BOUZEGHOUB	Professeur, Télécom SudParis
Examinateur :	Maxime LEFRANÇOIS	Maître de conférences, École des Mines de Saint-Étienne
Directeur de thèse :	Serge GARLATTI	Professeur, IMT Atlantique
Co-dir. de thèse :	Maha KHEMAJA	Maître-assistante, Université de Sousse, Tunisie
Co-encadrant :	Issam REBAI	Maître de conférences, IMT Atlantique

Remerciements

Je tiens tout d'abord à remercier les membres de mon jury d'avoir accepté d'être rapporteurs et examinateurs de ce travail : Je suis très honoré de compter Amel Bouzeghoub, Professeur à Télécom SudParis, Sylvie Desprès, Professeur à l'Université Paris 13, Maxime Lefrançois, Maître-assistant à l'École des Mines de Saint-Étienne et Gilles Coppin, Professeur à l'IMT Atlantique et Directeur du Lab-STICC.

J'adresse mes sincères remerciements à Serge Garlatti, mon directeur de thèse, qui a activement encadré ce travail et m'a fait bénéficier de sa grande expérience et de ses conseils. Merci d'avoir été si patient et disponible malgré tes nombreuses responsabilités.

Un seul merci ne suffirait pas pour exprimer toute ma gratitude à l'égard de Maha Khemaja, qui a fait bien plus qu'encadrer ma thèse. Tu m'as guidée et soutenue tout au long de mon parcours. J'espère avoir acquis un peu de ta rigueur scientifique. Un grand Merci à Issam Rebai, pour ton suivi et tes encouragements continus. Tu m'as donné l'occasion de faire mes preuves en informatique. Merci pour ta confiance durant toutes ces années.

Je tiens aussi à remercier tous les membres du département Informatique qui m'a accueilli. Je remercie les anciens et présents collègues avec qui j'ai partagé mon bureau pendant mes séjours au département. Mes remerciements distingués à Armelle Lannuzel, notre secrétaire, pour sa gentillesse et sa disponibilité. Son professionnalisme m'a beaucoup aidé dans toutes les formalités administratives.

Ce travail a été mené et accompagné avec le soutien en pensée, en prière et en amour de plusieurs personnes. Je pense à mon mari Foued, qui m'a encouragée chaque jour et avec qui nous avons dépassé tous les combats et difficultés pour que je puisse finir mes travaux, à mes fils Mohamed et Ahmed qui m'ont toujours apporté des motivations et du courage pendant ces travaux de thèse. Je pense aussi à ma mère, mes sœurs et mon frère qui m'ont soutenu depuis toujours. Je ne saurai oublier ma belle-famille.

GESTION INTELLIGENTE DE RÉSEAUX DE CAPTEURS, INTÉGRÉS À DES VÊTEMENTS SPORTIFS INSTRUMENTÉS

Résumé

L'Internet des Objets (IdO) intègre les réseaux de capteurs à Internet, et ouvre la voie pour des systèmes ou des écosystèmes ayant pour but d'aider les gens à vivre dans des mondes à la fois physiques et cybernétiques. L'IdO offre l'omniprésence d'objets capables d'interagir les uns avec les autres et de coopérer avec leurs voisins pour atteindre des objectifs communs. Ces objets dits "Intelligents" (OI), peuvent détecter l'environnement et communiquer avec d'autres objets. La création d'OI et de systèmes d'IdO fait intervenir des acteurs d'expertises très diverses. Ainsi, il devient indispensable d'avoir des descriptions standardisées et sémantiques pour résoudre les problèmes liés à l'interopérabilité et l'hétérogénéité sémantique entre les différentes ressources disponibles d'une part, et entre les différents intervenants à la conception/fabrication des OI, d'autre part. De ce fait, nous avons proposé le Framework sémantique et générique FSMS, structuré en un ensemble de modules ontologiques pour la conception/fabrication d'un OI. Une méthodologie de support à ce Framework a été proposée. Elle se fonde sur les mêmes modules ontologiques identifiés dans la composante sémantique du FSMS. Ces modules ontologiques forment l'ontologie SMS pierre angulaire de cette thèse. Un processus générique basé sur une description sémantique des composants structurels et comportementaux d'un OI a été également proposé en vue d'une gestion intelligente de la conception d'un OI. Ce processus a ensuite été mis en application pour des Vêtements Intelligents de sport. Un OI étant destiné à être réutilisé à différents contextes d'usage, une approche de reconfiguration/adaptation du fonctionnement de l'OI a été proposée. Celle-ci trouve à son tour son fondement dans l'ontologie modulaire SMS.

Mots clés : internet des objets, objet intelligents, web sémantique, ontologie modulaire, framework sémantique,

Abstract

The Internet of Things (IoT) integrates sensor networks with the Internet, and paves the way for systems or ecosystems to help people live in both physical and cyber worlds. IoT offers the ubiquity of objects that are able to interact with each other and cooperate with their neighbors to achieve common goals. These objects, called "Smart" (SO), can detect the environment and communicate with other objects. The creation of SO and IoT system involves actors of very diverse expertise. Hence, it becomes essential to have standardized and semantic descriptions to solve the problems related to the interoperability and the semantic heterogeneity between the different available resources on the one hand, and between the different stakeholders designing/manufacturing the SO, on the other hand. Therefore, we have proposed the FSMS semantic and generic framework, which is structured into a set of ontological modules to design/manufacture a given SO. A support methodology for this framework has been equally proposed. It is based on the same ontological modules identified in the semantic component of the FSMS. These ontological modules form the SMS ontology that is proposed and constitutes the cornerstone of this thesis. In order to intelligently manage an SO design, we proposed a generic process based on a semantic description of the structural and behavioral components of an SO. This process was thereafeter implemented for Smart Clothing of sports. This Sportswear is intended to be used in different contexts of use, an approach to reconfiguration/adaptation of the operation of the Smart Clothing has been proposed. This one is also based on the modular ontology SMS.

Keywords: internet of things, smart objects, semantic web, modular ontology, semantic framework.

Table des matières

Liste des tableaux

Table des figures

Chapitre 1

Introduction

Sommaire du présent chapitre

Ce chapitre introduit le contexte général des travaux de recherche décrits dans le présent manuscrit. Il commence par présenter la méthodologie de recherche utilisée et précise l'application de chacune de ses étapes dans le contexte de ce travail de thèse. Il présente alors le contexte de recherche, les motivations, la problématique ainsi que les verrous scientifiques qui ont guidé ce travail. Il définit et délimite également les axes de recherche et permet, de plus, de les positionner dans le contexte de recherche existant et de souligner nos principales contributions. Il dresse enfin l'organisation du manuscrit.

1.1 Méthodologie de recherche

Dans le contexte du présent travail de recherche nous adoptons la méthodologie de recherche en science du design (*Design Science Research Methodology* (DSRM) en Anglais). Cette méthodologie s'adapte très bien aux projets de recherche en systèmes d'informations et en génie logiciel [1]. Elle permet de conduire deux types d'activités qui sont la conception d'artefacts et leur investigation en contexte. Les artefacts peuvent être des méthodologies, des algorithmes, des outils, etc. qui permettent d'interagir avec un certain contexte pour générer les résultats escomptés. Dans un projet fondé sur la DSRM, deux types de contextes sont considérés, le premier est le contexte social qui décrit les désirs, les objectifs, les attentes des parties prenantes et leurs exigences par rapport au projet, et le contexte théorique ou de connaissances qui contient tous les fondements théoriques en relation avec le projet et dont l'évaluation (en quelques sortes négative) permet de motiver l'une des orientations du projet et donc les contributions. La démarche que nous avons adoptée dans le contexte spécifique de ce travail de thèse est illustrée par la figure 1.1. Celle-ci illustre les étapes du processus suivi en identifiant les entrées et les sorties de chaque étape ainsi que le numéro du chapitre où l'étape est décrite de manière détaillée.

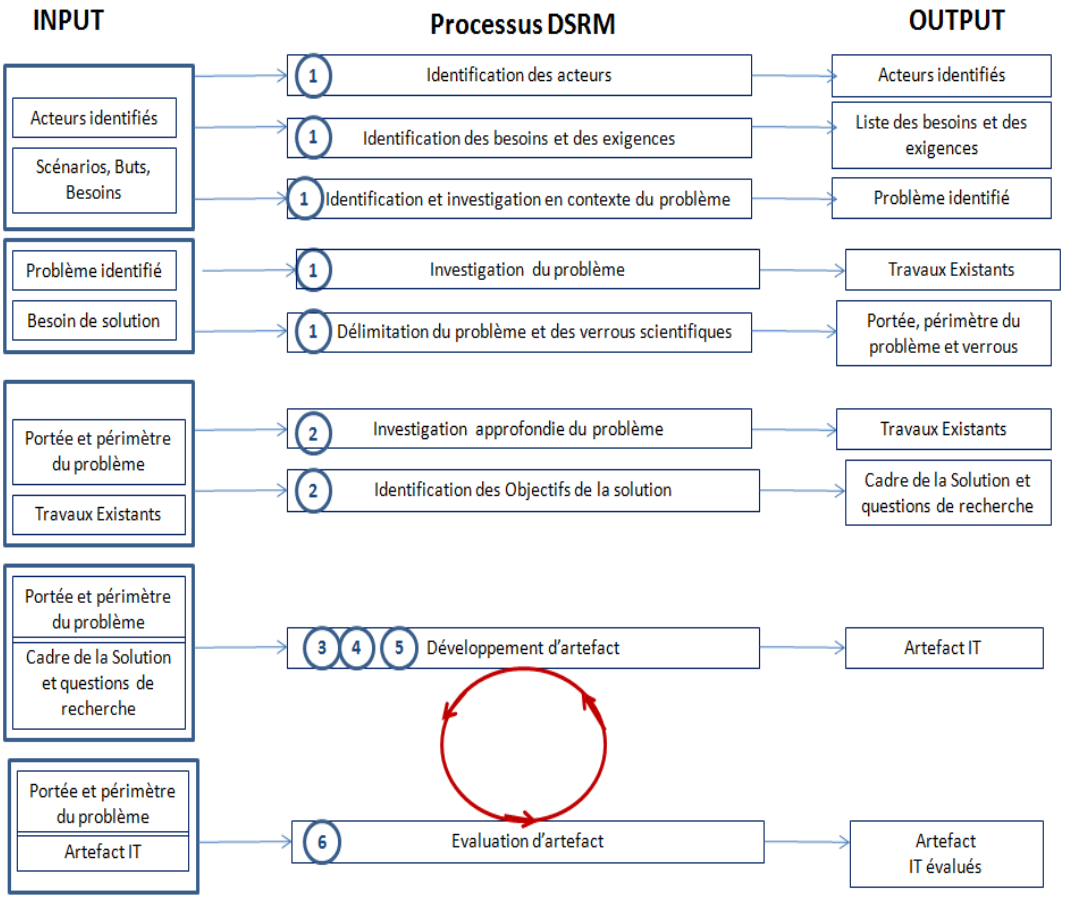

FIGURE 1.1 – Processus DSRM

Les toutes premières étapes sont traités au niveau de ce premier chapitre et permettent de :

— Identifier les acteurs et leurs besoins à travers des scénarios de motivation.

— Identifier le problème et d'en faire une investigation en contexte. Ceci est réalisé au travers d'entretiens avec des experts.

— Identifier une taxonomie des travaux connexes à étudier et synthétiser.

— Délimiter le problème, les axes de recherches ainsi que les verrous scientifiques.

Les étapes réalisées dans le second chapitre concernent :

— Une investigation plus poussée de la problématique à travers une revue de la littérature identifiée précédemment.

— L'identification des objectifs et du cadre conceptuel de la solution à développer.

— La formulation des questions de recherche.

Les étapes réalisées dans les chapitres 3,4 et 5 concernent l'activité de conception. Elles visent à :

— Identifier et préciser les portée des artefacts à développer. Une investigation du contexte théorique et pratique de chaque artefact peut s'avérer des fois nécessaire en vue de cerner les différents défis qu'il engendre.

— Concevoir et développer les artefacts identifiés tout en respectant les fondements méthodologiques spécifiques à la nature de chaque artefact.

Les dernières étapes sont développées au niveau du chapitre 6. Elles concernent le test et la validation de la solution. Celles-ci s'attachent tout d'abord à concevoir l'approche de test et de validation pour ensuite l'exécuter avec la solution. En particulier dans le cadre de cette thèse et à cause de l'arrêt du projet SmartSensing, une expérimentation avec les acteurs n'a pas pu être réalisée. Ces étapes ont donc permis de :

— Définir une approche de prototypage et de simulation en laboratoire.

— Evaluer l'intégration de l'ensemble des artefacts proposés.

— De réaliser plusieurs itérations en vue d'améliorer la solution.

— De rédiger un ensemble de publications ([2] et [3]) avec le présent rapport de thèse.

Dans la suite de ce chapitre nous allons donc dérouler les premières étapes de cette méthodologie.

1.2 Contexte de recherche et projet SmartSensing

1.2.1 Contexte de recherche

Depuis quelques années, le domaine de l'électronique, des réseaux et de l'informatique ont vu une évolution rapide et importante grâce entre autre à la miniaturisation des circuits programmables et au développement de nouveaux protocoles de communication sans fil, d'équipements mobiles, de la gestion massive de données (*BigData* en Anglais) et de l'informatique en nuage (*Cloud computing* en

Anglais). Ces avancées technologiques démocratisent aujourd'hui les systèmes embarqués ainsi que ceux utilisant des objets et des capteurs sans fil connectés à Internet (c'est-à-dire les systèmes basés sur l'Internet des Objets (IdO) (*Internet of Things* (IoT) en Anglais)) et ouvrent de nouvelles perspectives d'usage de l'informatique inimaginables il y a encore quelques années. Selon [4] et [5], les systèmes IdO ont été définis comme étant "des systèmes qui permettent, une omniprésence continue autour de nous, d'une variété de choses ou d'objets - tels que des étiquettes RFID (Identification par Radio Fréquences), des capteurs, des actionneurs, des téléphones portables,. . . - qui, à travers des systèmes d'adressage uniques sont capables d'interagir les uns avec les autres et de coopérer avec leurs voisins pour atteindre des objectifs communs". Ces objets omniprésents (encore dits pervasifs) peuvent ainsi détecter l'environnement (ou le contexte de l'utilisateur), communiquer avec d'autres objets, détecter, interpréter et comprendre des données portant même sur des situations complexes ou des scénarios difficiles et y apporter des réponses de manière autonome sans aucune intervention humaine. En particulier, une nouvelle catégorie d'usage se concrétise dans les systèmes dits d'assistance intelligente aux usagers. Ceux-ci tout en combinant les différentes technologies IdO permettent, de manière transparente, de collecter de façon continue des données capteurs et d'appliquer des mécanismes de raisonnement et de prise de décision afin d'apporter l'assistance adéquate aux usagers leur permettant de réaliser et pratiquer des activités de tous les jours (c-à-d des activités de travail, de shopping, des activités sportives. . .) tout en combinant des mondes à la fois physiques, virtuels et intelligents.

Afin d'atteindre ces objectifs, plusieurs travaux de recherches ont été réalisés touchant divers aspects et créant ainsi des visions différentes et conjointes dans le domaine de l'IdO [5] : La vision orientée objets (*Things-oriented*) qui prend en charge l'utilisation de l'*Electronic Product Code* (EPC) en conjonction avec la technologie RFID pour la collecte et le suivi des données de capteurs. La vision orientée Internet (*Internet-oriented*), qui correspond à la construction des protocoles de communication pour permettre aux objets intelligents de communiquer entre eux et/ou de se connecter à Internet. La vision orientée sémantique (*Semantic-oriented*) ; celle-ci aborde les problèmes de gestion de la sémantique des données qui surviennent dans le contexte de grandes quantités d'informations échangées par des objets intelligents et des ressources disponibles via le Web. L'idée est qu'il devienne indispensable d'avoir des descriptions standardisées, sémantiques et uniformes pour résoudre le problème de l'interopérabilité et l'hétérogénéité sémantique de toute ressource disponible dans le système ou qui interagit avec celui-ci quelque soit son type physique ou virtuel. Cette dernière vision soulève aussi le problème de l'intelligence de ces objets connectés. En d'autres termes comment représenter cette intelligence et où est ce qu'elle sera implémentée ? Une autre vision hybride qui se base sur les trois autres souligne les problèmes liés à la conception et au développement de tels systèmes et dégage ainsi les problématiques liées aux parties prenantes, leurs différentes compétences, préoccupations et points de vue. Dans nos travaux, nous nous intéressons à cette dernière vision hybride. Plus particulièrement nous nous focalisons, d'une part, sur le problème de conception et développement de tels systèmes et d'autres part, sur leur exploitation, adaptation et reconfiguration pour différents usages, tout en assurant la gestion, la représentation et l'intégration sémantiques des

données provenant des divers capteurs composants le système développé.

1.2.2 Le projet SmartSensing

Ce contexte de recherche a été initié dans le cadre du projet SmartSensing et a donc été fortement inspiré par lui.

L'objectif de ce projet est de produire des panoplies ou des vêtements dits "intelligents" permettant d'assurer le pilotage (*monitoring* en Anglais) de l'usager en connexion avec les outils de communication du quotidien. Un vêtement intelligent n'est autre qu'un objet intelligent complexe, comprenant à son tour des micro-capteurs miniaturisés intégrés dans le textile qui peuvent capturer des mesures et des indicateurs physiologiques, des passerelles qui permettent de transmettre ces données *via* une connexion sans fil et des applications pour collecter, analyser, restituer, afficher et prendre des décisions adéquates à partir de ces données. Une panoplie est formée d'un ensemble de vêtements intelligents ou de systèmes portés (comme par exemple un casque, une montre, ...) embarquant des capteurs intelligents disposés en réseaux et offrant à leur usager les services requis. Ces panoplies ou vêtements intelligents représentent une composante importante d'un système IdO destiné à des applications spécifiques centrées usager. Celles-ci sont destinées à offrir des services au porteur du vêtement intelligent ou à toute autre personne concernée par ce dernier. En particulier, dans le cadre du projet SmartSensing, ces vêtements intelligents sont destinés au monde sportif professionnel et grand public. Ils visent à offrir à la fois un moyen d'optimiser les performances des sportifs et de suivre leurs indicateurs de forme pour une meilleure approche de prévention de risques. Cependant, le champ d'application du vêtement intelligent ne se limite pas uniquement au secteur du sport ou du bien-être mais peut couvrir d'autres secteurs comme la santé, l'automobile ou la domotique.

De manière plus générale, différentes applications et différents écosystèmes intégrant cette notion d'objets intelligents (OI) sont actuellement en train de se développer en vue d'exploiter les potentialités offertes par ces OI et de servir différents objectifs.

1.2.3 Motivations

Nous illustrons les principales motivations de ce travail de recherche à travers un ensemble de scénarios. Pour l'ensemble de ces scénarios, nous focalisons notre analyse sur la notion de vêtement intelligent de sport que nous considérons comme un exemple représentatif d'un Objet Intelligent dont le rôle particulier est de collecter des données physiologiques de son porteur. Ces scénarios sont inspirés du projet SmartSensing et permettent, toutefois, de souligner les différents défis, exigences et préoccupations des différents acteurs.

Scénarios de Conception/fabrication de vêtements intelligents (VI)

Les premiers scénarios A, B et C concernent la phase de conception/fabrication de cet Objet Intelligent. A l'aide de ces scénarios, nous illustrons un exemple des activités, des connaissances et des

données délivrées par les experts du projet SmartSensing. Nous illustrons ainsi les différents aspects observés pour la conception/fabrication d'un Vêtement Intelligent (VI), panoplie intelligente (PI) ou encore chaussures intelligentes.

Scénario A. SmartTextile est une société nouvellement créée qui se spécialise dans le domaine de la fabrication de vêtements intelligents dédiés au sport. Elle reçoit une commande de création d'une tenue dédiée au football. Pour répondre à son client, elle fait appel à son équipe d'experts qu'elle décide de renforcer par d'autres experts externes à l'entreprise.
L'équipe résultante est formée par :

1. Des artistes, des experts en textile et des stylistes modélistes de panoplies vestimentaires qui vont concevoir l'objet vêtement ou panoplie ;
2. Des experts en sport qui vont définir l'ensemble des pratiques du sport en question et de ses indicateurs de performances ;
3. Des médecins de sport qui spécifient aussi bien des indicateurs de performances que d'autres indicateurs physiologiques associées à d'éventuels risques associés au sport considéré ;
4. Et pour chaque indicateur préalablement défini :
   — Des électroniciens se chargent de concevoir et créer des capteurs intelligents en assemblant des capteurs basiques (des accéléromètres par exemple), en les reliant en réseau avec un micro-contrôleur (ou n'importe quelle unité de traitement miniature) ;
   — Et des informaticiens (ou développeurs) se chargent de concevoir et développer les logiques de traitement à intégrer dans les capteurs pour les doter de l'automatisme (ou intelligence requise). Ceux-ci ont aussi la responsabilité de concevoir, développer, intégrer et déployer l'ensemble de la plateforme IdO ou de l'écosystème qui va intégrer le VI, le faire fonctionner, le piloter, analyser les données collectées, émettre des décisions,....

Le but étant de choisir essentiellement les composants adéquats d'une « PI ou VI » pour le football tout en tenant compte d'un ensemble de contraintes liées au matériel, au logiciel, au coût et aux différents usages souhaités. En particulier, cette équipe d'experts doit tenir compte des contraintes et aspects suivants :

1. Le contexte d'utilisation (y compris les pratiques du football et d'autres informations contextuelles) ;
2. La concordance entre les logiques de traitements embarquées et les résultats attendus ;
3. La cohérence entre les données en entrées des logiques de traitement embarquées et les propriétés observées par les capteurs basiques à intégrer dans les capteurs intelligents ;
4. Les capacités de calcul et de stockage du capteur intelligent à concevoir et à fabriquer doivent être suffisantes pour supporter les logiques de traitement embarquées ;
5. La position des capteurs intelligents dans la panoplie à créer doit être calculée de manière très précise.

Cette mission a commencé par l'organisation d'un certain nombre de réunions et de discussions entre les différents intervenants appartenant à une même discipline, et après des concessions et des compromis quant aux caractéristiques, fonctionnalités et exigences souhaitées, différents livrables décrivant le produit, son design et ses spécifications techniques ont été produits en guise de préalables à la phase de fabrication. Ensuite, un ensemble de sessions de discussion réunissant l'ensemble des intervenants a été planifié.

Malheureusement, ces sessions ont été vécues par tous les intervenants avec beaucoup de stress et de tension qui étaient dus essentiellement à des problèmes d'incohérences entre les différentes visions et modèles de conception (maquettes) fournis par les intervenants des différentes disciplines du VI conçu. En effet, les spécialistes de la mode et du design ont trouvé que la solution avec laquelle les électroniciens ont pensé intégré les composants électroniques souffrait de beaucoup de problèmes d'esthétique et de portabilité (wearability), alors que les électroniciens voyaient une inefficacité de la solution offerte par les experts de la mode et que l'aspect esthétique était très exagéré et qu'il sacrifiait beaucoup l'aspect fonctionnel et utile du VI.

Les médecins et les professionnels du sport quant à eux n'ont pas pu trouver de concordances entre ce qu'ils demandaient comme indicateurs et mesures et la solution offerte par les électroniciens et les informaticiens.

Scénario B. La société SmartShoes spécialisée dans la fabrication de chaussures intelligentes a vécu un départ important de son personnel compétent dans la conception de capteurs intelligents. N'ayant plus la possibilité de répondre rapidement au nombre croissant des commandes de ses clients, son Gérant Paul a décidé de s'approvisionner en capteurs intelligents chez la société SmartTextile. Les responsables de cette dernière, heureux de pouvoir réutiliser et rentabiliser le coût de conception et de fabrication de leurs capteurs intelligents on accepté l'offre. Le contrat entre les deux sociétés a concerné aussi bien les composants matériels que l'ensemble des programmes embarqués.

Scénario C. La société SmartTextile a reçu cette fois-ci une commande pour des panoplies intégrant des chaussures Intelligentes pour une équipe de football où les joueurs souffrent tous de différents types d'handicaps et de besoins spécifiques. Pour gagner du temps et rentabiliser les résultats des réalisations précédentes, SmartTextile s'est alliée avec Smartshoes. Elles ont engagé directement des adaptations aux modèles et maquettes conçues préalablement pour l'ensemble des panoplies et des chaussures, étape suivie ensuite par leur fabrication et tests sur terrain. Cependant, l'étape de test n'a pas été fructueuse, car un bon nombre de joueurs n'a pas trouvé les vêtements conçus adéquatement à leurs besoins.

Scénarios d'exploitation de VI

Dans cette section, nous décrivons également un ensemble de scénarios d'usage d'un VI ou PI préalablement conçus. Pour cela nous considérons Pierre et John deux footballeurs amateurs qui sont

impliqués dans différentes pratiques du football (comme par exemple l'entraînement, la compétition, l'entraînement spécifique comme Gardiens de buts). Nous considérons aussi Abdel un sportif pratiquant le cyclisme et la course à pied, Beth une athlète professionnelle qui s'entraîne pour le biathlon et Charles une personne âgée nécessitant une surveillance constante.

Scénario D. Pierre a donc acheté une panoplie formée d'un maillot et d'un short intelligents. Ceux-ci embarquent les capteurs intelligents initialement conçus et fabriqués et communiquent avec son smart phone pour lui envoyer les indicateurs collectés. Très satisfait de son achat Pierre, voudrait avoir la possibilité de réutiliser sa panoplie pour réaliser d'autres types d'activités ou d'autres sports tout en pilotant ses indicateurs de performance. Malheureusement, Pierre n'a pas trouvé de moyens pour adapter sa panoplie. Il a donc envoyé un mail pour exprimer sa suggestion d'adaptation de la panoplie à son vendeur. Deux jours plus tard, il reçoit un mail de réponse, lui expliquant l'impossibilité d'adapter la panoplie. Cette opération ne sera éventuellement possible qu'en usine.

Scénario E. John a eu plus de chance que Pierre, il a commandé sa panoplie directement à l'usine. Il a également ressenti le même besoin d'adaptation, il a donc ramené sa panoplie à l'usine, suggérant son adaptation au « footing ». Deux jours après il est retourné pour récupérer sa panoplie adaptée et a commencé à l'exploiter pour le footing. Mais il a constaté malheureusement qu'après cette adaptation, il ne peut plus l'utiliser pour le football. Il est revenu voir les concepteurs/fabricants. Ceux-ci embarrassés par le résultat de leur intervention sur la panoplie, ont essayé d'expliquer à John l'impossibilité de prévoir deux types de sports en même temps pour des problèmes techniques qu'il leur est difficile actuellement de résoudre.

Les trois scénarios suivants illustrent quant à eux des détails techniques relatifs à l'usage d'un exemple de capteur intelligent hébergeant des accéléromètres mesurant l'accélération à 3 axes de types LIS2DH. Ce capteur Intelligent est composé d'un réseau reliant des capteurs basiques à un micro-contrôleur doté de capacités de traitement plus élaborées que celles des capteurs basiques.

Scénario F. Abdel a acheté un short (un cuissard) de sport doté de ce capteur intelligent. Il souhaite surveiller son activité en utilisant une connexion Bluetooth Low Energy avec sa Smart Watch (montre connectée). Pendant une course, il souhaite piloter le nombre de pas qu'il réalise, une estimation de la distance qu'il parcourt, et la cadence de ses foulées. Quand il pratique du cyclisme, il souhaite surveiller la cadence instantanée de pédalage, le temps de pratique (ou durée) de la danseuse et le niveau de dénivèlement de la route. Ces six indicateurs sont calculables à partir des mêmes données brutes (ici, mesures d'accélérations) fournies par le capteur (l'accéléromètre LIS2DH) intégré au capteur intelligent. Ce qui fait la différence, ce sont les programmes de calcul embarqués et exécutés par le micro-contrôleur. Lorsque le sportif sélectionne l'activité qu'il veut pratiquer, le capteur intelligent modifie son fonctionnement en chargeant les algorithmes adéquats. Maintenant, Abdel veut commencer à faire du roller, il parcourt le Web à la recherche d'un petit programme qu'il peut charger sur son capteur intelligent pour obtenir une estimation de son taux de patinage. Ce programme ne peut être

installé sur le capteur intelligent que si : (1) les entrées du programme peuvent être fournies par le LIS2DH et (2) que les capacités du micro-contrôleur du capteur intelligent sont suffisantes pour l'exécuter.

Scénario G. Beth, une athlète professionnelle, veut s'entraîner au biathlon. Son staff technique Bob veut évaluer sa technique de ski et sa stabilité durant les séances de tir. Pour cela, la panoplie portée par Beth est équipée d'un LIS2DH dans chacun de ses membres (bras et jambes). Ceux-ci sont connectés à un capteur intelligent central qui diffuse les données en utilisant le protocole CoAP via une connexion WiFi. Pendant qu'elle skie, les données d'accélération sont analysées pour calculer l'amplitude du mouvement des bras et des jambes, la technique de ski utilisée et l'angle d'inclinaison et de rotation des différents membres. Durant la phase de tir, les données d'accélérations sont analysées pour mesurer le niveau de sa stabilité et de sa capacité de concentration. Le capteur intelligent aura donc des rôles différents selon l'étape du biathlon. Il doit détecter les changements d'étapes et adapter son fonctionnement en appliquant à chaque fois les programmes adéquats.

Scénario H. Ce dernier scénario illustre comment le même capteur intelligent peut être réutilisé dans un domaine différent. Supposons que Charles, une personne âgée, nécessite une surveillance constante de son état de santé. Il est équipé d'un bracelet intégrant le capteur intelligent. Pendant que Charles s'endort, le capteur intelligent analyse sa qualité de sommeil et envoie régulièrement des données à un courtier MQTT local. Il détecte ses mouvements corporels et les classe en catégories (brusques, micro-mouvements, etc.). Au milieu de la nuit, le capteur intelligent détecte des mouvements indiquant que Charles sort du lit (par exemple, pour boire ou aller aux toilettes), le capteur intelligent bascule son mode de fonctionnement pour détecter des cas de chute, mode qui est également toujours utilisé en journée. Il analyse les données de l'accéléromètre et alerte le courtier MQTT en cas de chute. Etant donné la situation de risque dans laquelle ce capteur intelligent est utilisé, son fonctionnement est augmenté d'une capacité de détection d'anomalies ou de dysfonctionnement. Le capteur intelligent envoie ainsi une valeur de code d'erreur, potentiellement avec la spécification de la cause de l'anomalie. Les soignants ou les services d'urgence peuvent alors être avertis et réagir en conséquence.

1.2.4 Analyse des scénarios

Notre analyse des scénarios décrits précédemment nous permet de dégager les problèmes suivants auxquels peuvent être confrontés les différents acteurs :

1. Tout d'abord, le scénario A met l'accent sur l'absence d'un guide méthodologique et organisationnel permettant aux différents acteurs de coordonner leurs activités en vue de concevoir et fabriquer l'OI désiré.
2. Même en la présence de ce guide méthodologique, les concepteurs/fabricants se fient beaucoup plus à leurs expériences respectives pour dégager les différents besoins et les différents usages de l'OI. Une conception faisant participer l'utilisateur final n'est pas toujours appliquée.

3. Chacun des acteurs, même s'il ne l'exprime pas de manière directe lors des réunions d'équipe, il essaie de favoriser son point de vue et d'imposer ainsi sa vision de futur objet conçu par rapport aux choix des capteurs, de leurs positions, de leurs types, etc. Cet aspect, pourrait engendrer plus tard des problèmes comme par exemple l'impossibilité d'adapter une panoplie à un autre sport particulier.
4. Nous remarquons également qu'en l'absence d'utilisation de standards du domaine, les différents acteurs se trouvent obligés de faire face à l'usage d'une terminologie multidisciplinaire avec des fois des termes disparates bien qu'ils puissent des fois porter un sens identique. Ceci ne fait qu'augmenter la complexité du travail, les chances d'incompréhension, d'ambiguïté et d'erreurs de conception. De plus, ceci rend difficile l'exploitation de livrables fournis par les membres d'une discipline donnée par ceux d'une autre discipline.
5. Le scénario B, bien qu'il souligne l'importance de la réutilisation des composants électroniques intelligents pour d'autres types d'objets, il met de plus l'accent sur le problème de départs des experts de l'entreprise et que l'absence d'un moyen de réutiliser et/ou de gérer leur expertise et leur savoir faire n'a fait qu'amplifier le problème.
6. Le scénario C, souligne l'importance d'une étude poussée et systématique des besoins des usagers finaux avant même d'engager les phases de conception et de fabrication. Il souligne aussi la possibilité de réutilisation des modèles de conception préalablement réalisés pour les adapter à de nouveaux besoins.
7. Concernant les scénarios D et E, nous pouvons déduire la nécessité de prévoir l'adaptation d'un objet intelligent à différents usages et à différents contextes voire à différents domaines. Nous constatons également la difficulté que pourrait engendrer ce type d'adaptation, si initialement les concepteurs/fabricants n'ont pas tenu compte de cet aspect au tout début du choix des composants matériels et logiciels embarqués.
8. La solution qui sera appréciée par les usagers et qui les incitera forcément à acheter ces types d'objets intelligents est qu'il leur soit possible d'acheter un seul objet et de le réutiliser plusieurs fois dans des usages différents. Aspect qui sera également apprécié par les concepteurs si la solution technique qu'ils adoptent leur permet de développer l'objet une fois et de le reconfigurer plusieurs fois.
9. À travers les scénarios F, G, et H nous pouvons déduire les détails de composition d'objets intelligents et plus particulièrement les capteurs basiques qu'ils hébergent ainsi que la terminologie censée être utilisée par différents experts.

   Nous pouvons également déduire différents éléments contextuels spécifiques qui guident l'adaptation ou la reconfiguration des objets intelligents ainsi que la méthode utilisée pour réaliser l'adaptation.

   Le scénario F illustre la possibilité de reconfiguration de l'objet intelligent à différents sports en téléchargeant des programmes existants sur le Web et d'usage grand public. Le scénario

G illustre l'adaptation de l'objet en cours d'exécution des étapes du même sport, alors que le scénario H traite de la reconfiguration en fonction des signaux physiologiques de l'utilisateur et l'heure de la journée. Ces deux derniers scénarios exigent une reconfiguration des objets intelligents à chaud, ce qui implique aussi la présence de l'ensemble des programmes nécessaires au sein du capteur intelligent ou alors la possibilité de les télécharger et les installer de manière instantanée sans arrêter le capteur ou l'objet Intelligent. Dans tous les scénarios, les composants matériels restent donc inchangés et l'adaptabilité est assurée par la capacité à reconfigurer le capteur intelligent, à changer le code exécuté par le micro-contrôleur et à modifier le paramétrage de communication du capteur intelligent avec le monde extérieur.

Cette analyse nous guide vers un ensemble de besoins importants :

1. Tout d'abord, en considérant le premier scénario ainsi que l'observation de certaines activités des équipes multidisciplinaires du projet SmartSensing, nous pouvons d'ores et déjà identifier un ensemble de modèles de tâches ou de processus métiers à part entière pour la conception de VI et du système dans lequel il sera intégré. Ces processus doivent être cependant clairement formalisés, développés et validés pour permettre de coordonner les activités des différents acteurs et pour permettre à terme d'outiller et d'automatiser leur travail.
2. Ensuite, ce domaine engage différents acteurs de différentes disciplines, voire même plusieurs acteurs de la même discipline. Ceux-ci peuvent avoir des avis non homogènes. Ils peuvent même avoir différentes alternatives. Des compromis ou des consensus avec éventuellement une validation expérience à l'appui permettront une meilleure capitalisation de la connaissance liée à la définition d'un objet Intelligent et surtout à sa modélisation, conception, fabrication et exploitation.
   Disposer d'un Framework qui regroupe cet ensemble de connaissances, de concepts, de processus et de modèles (ou méta-modèles) qui soient sémantiquement riches et validés en permettra une meilleure réutilisation. Il permettra de plus, de cerner et délimiter le contexte de travail des différents acteurs et de cerner également le vocabulaire qu'ils utilisent. Il leur permettra enfin, d'être plus focalisés sur les objectifs de la tâche ou de la mission qui leur est allouée.
3. Un troisième besoin est lié à l'absence d'une vraie méthodologie de modélisation de l'ensemble des concepts, des processus et des connaissances manipulés et de tous les aspects liés à la fabrication et à l'exploitation de l'Objet Intelligent en traduisant l'ensemble des perspectives des différents acteurs, en autorisant et en permettant une vraie collaboration pour des décisions consensuelles.
   Il est important que cette méthodologie couvre les différentes perspectives (structurelles, comportementales, . . .) pour définir et décrire un OI ainsi que le système qui l'intègre et qu'elle permette d'appliquer un ensemble de principes connus et éprouvés dans le domaine des méthodologies de conception et de développement logiciel tels que la généricité, le faible couplage entre la spécification des objets intelligents et leurs usages, la séparation des préoccupations pour toutes les logiques de traitement conçues, développées et embarquées dans les objets avec un bon

niveau de scalabilité, vu le nombre de concepts et composants structurels et comportementaux importants liés aux objets intelligents qui peuvent être disponibles et traités.

4. Cette méthodologie doit également permettre de concevoir l'ensemble du système ou de l'écosystème qui intégrera à terme le ou les OI conçus et fabriqués.
5. Enfin, elle doit permettre de définir une approche ou un mécanisme d'adaptation et/ou de reconfiguration de l'OI pour le réutiliser dans différents contextes et différents domaines.

1.2.5 Synthèse

Quels que soient, le domaine d'application, l'écosystème IdO et les Objets Intelligents à développer dans le cadre d'un projet donné (comme c'est le cas du projet SmartSensing), il est nécessaire de souligner que chaque composant de cet écosystème y compris l'écosystème lui-même doit passer par des phases de conception/fabrication, développement, intégration, déploiement et tests de validation ou d'évaluation sur terrain.

La phase de conception/fabrication nécessite l'intervention d'un ensemble d'acteurs de spécialités différentes et appartenant à des communautés hétéroclites tels que des usagers finaux, des spécialistes du métier, des agents commerciaux, des artistes (Aesthetic designers), des électroniciens, des informaticiens, des fabricants de matières premières ou de matériaux pour les Objet physiques intégrés, ... Ces acteurs sont appelés à collaborer pour concevoir et développer un Objet Intelligent simple, un Objet Intelligent composite ou encore un Ecosystème ou Plateforme IdO à part entière. Les utilisateurs ou experts métiers ou du domaine concerné interviennent pour exprimer leurs besoins quant à la nature de l'information, la manière et la fréquence de la capturer et de la traiter et ce en fonction de l'activité (physique ou même mentale) de l'usager et/ou du contexte de fonctionnement de l'écosystème.

En vue de répondre à ce besoin, les concepteurs et fabricants des matériaux, les concepteurs et fabricants des objets physiques (comme par exemple, un vêtement, une voiture, etc) ou plus généralement d'écosystèmes intelligents ainsi que ceux responsables de la conception de composants électroniques intelligents (tels que des capteurs ou des actionneurs) ont besoin tous de travailler de manière collaborative et consensuelle. Ils doivent tout d'abord choisir les types de composants électroniques basiques à utiliser (par exemple des capteurs). Ils doivent décider ensuite de leurs aspects physiques (taille, forme et poids par exemple), de leurs fonctionnalités, des protocoles de communications qui vont les doter, ... Enfin ils doivent décider de la manière d'agencer l'ensemble des composants électroniques au niveau des objets intelligents ou au niveau des éléments composant l'écosystème, en vu d'offrir les fonctionnalités requises soit pour l'usager final soit pour les utilisateurs métiers, soit encore pour d'autres intervenants comme des décideurs, des concepteurs d'activités (physiques ou autres), ...

Les informaticiens doivent développer les logiques de traitement adéquates pour, par exemple, transformer les données capteurs et en faire des informations traitables portant une signification à leurs utilisateurs ou pour faire fonctionner et piloter les composants électroniques, etc. Ces développeurs peuvent soit réutiliser des logiques de traitement existantes, soit en développer de nouvelles. Il est

cependant utile de leur permettre d'interroger des bibliothèques ou des répertoires de logiques de traitement (d'algorithmes ou de programmes) existants pour une réutilisation éventuelle. De plus ils doivent développer les logiques de traitement permettant le fonctionnement global de l'écosystème (comme par exemple la collecte et l'analyse des données, la prise de décision, etc).

Les informaticiens architectes doivent décider de l'architecture finale de l'écosystème, de la manière d'intégrer les objets intelligents, de la manière de déployer leur logiques de traitement et enfin de la manière d'installer l'ensemble des ces éléments sur les différents nœuds composant l'écosystème, y compris les nœuds externes (Web ou Cloud). Mise à part l'intégration des objets intelligents dans leurs écosystèmes, la phase d'exploitation de chaque objet intelligent conçu, fabriqué et développé, nécessite de l'adapter, de le configurer et/ou de le piloter pour différents usages et différents contextes. En effet, une fois ces systèmes mis en exploitation, ils permettront aux utilisateurs métiers, aux décideurs, et même aux utilisateurs finaux de mieux comprendre et connaître les difficultés et les problèmes du domaine (comme par exemple les problèmes de santé du porteur d'un vêtement intelligent). Ils auront un meilleur suivi en temps réel de toutes les activités impliquant l'usage de ces objets intelligents. Ils pourront d'autant plus utiliser ces données dans d'autres systèmes de suivi et d'analyse.

Les concepteurs d'activités (métiers, sportives, ...) seront quant à eux, plus avertis pour le choix de la nature de l'activité à proposer à l'usager, pour élaborer des plans d'activités (ou processus) plus adéquats, pour faire le suivi du progrès et des performances de l'usager et pour éventuellement proposer des plans optimisés d'activités permettant d'atteindre des objectifs identiques dans des meilleurs délais, etc.

Enfin, les utilisateurs finaux des objets intelligents et/ou de l'écosystème mis à part qu'ils soient mieux informés grâce aux informations qui leurs sont transmises en temps réel, ils pourront également réutiliser le même objet pour réaliser des activités différentes (comme pour le cas du vêtement intelligent de sport). L'écosystème, de manière intelligente et autonome, pourra pour cela bénéficier des logiques de traitement préalablement stockées dans des bibliothèques dédiées, pour réaliser les reconfigurations nécessaires sans exiger à l'utilisateur d'avoir des connaissances spécifiques sur les logiques de traitement à mettre en place, sur la manière de les installer ou de les configurer.

1.3 Problématique, verrous scientifiques et questions de recherche

L'analyse des buts, objectifs et exigences des différents acteurs de ces systèmes nous a permis de dégager la problématique générale décrite dans ce qui suit. Cependant, nous ne prétendons pas l'avoir traité entièrement dans le contexte de cette thèse. Une analyse plus approfondie de la littérature, nous a permis donc de la délimiter, d'en délimiter les verrous scientifiques ainsi que les questions de recherche.

1.3.1 Problématique

Tout d'abord, le processus de conception/fabrication des objets intelligents et connectés ainsi que des systèmes ou écosystèmes intégrant ces objets est un processus à la fois complexe, collaboratif et pluridisciplinaire. Il exige de plus l'usage d'une terminologie et d'une expertise spécifique à chaque discipline tout en considérant différents angles de perception d'un objet connecté. Toutes les activités collaboratives de ce processus exigent également la définition d'un cadre de travail ou Framework de modélisation d'un objet intelligent qui soit dirigé par l'expertise des différents acteurs, les caractéristiques de l'objet intelligent, le domaine dans lequel il sera exploité ainsi que son contexte d'usage. Ceci en garantissant un formalisme de représentation uniforme à l'ensemble de ses composantes, qui soit à la fois intelligible à l'Homme et à la machine et qui puisse permettre une capitalisation et réutilisation faciles de chacun de ses modèles.

De plus, le choix des composants électroniques d'un objet étant l'une des activités les plus importantes du processus de conception. Celle-ci dépend de plusieurs critères antagoniques (capacité, besoin, coût, taille, etc.).Trouver un juste compromis entre les acteurs sur le choix du bon composant n'est pas une tâche triviale. Elle exige non seulement l'accès à de grandes bases de connaissances décrivant les composants mais aussi un support adéquat pour la recherche et l'interrogation multi-critères (voire sémantique) de ces bases de connaissances qui peuvent être très hétérogènes. Ensuite, l'intégration des objets intelligents dans des environnements ou des écosystèmes embarqués permettant leur exploitation et adaptation à différents contextes d'usage, nécessite un fondement méthodologique adéquat, permettant de concevoir l'architecture logicielle et matérielle de ces écosystèmes ainsi que les mécanismes d'adaptation et de reconfiguration à chaud de leurs fonctionnalités embarquées. Ces mécanismes d'adaptation et de reconfiguration seront guidés par les caractéristiques de l'objet intelligent ainsi que par son contexte d'usage. Différents travaux de recherche ont porté sur un ou plusieurs aspects de cette problématique. Notre investigation de ces travaux nous a permis d'en identifier les limites et de préciser et délimiter davantage la problématique qui sera développée dans le reste de cette thèse.

En effet :

— Bien qu'il existe actuellement différentes propositions de modèles ontologiques où la sémantique est utilisée comme solution d'interopérabilité entre les différentes données transmises par des capteurs intégrés dans des systèmes IdO, ces solutions n'ont pas été dédiées à la conception et au développement de la notion de capteur ou d'objet intelligent.

— De plus, elles n'ont pas été initiées dans le but de servir à la fois pour l'interopérabilité sémantique des données capteurs et pour supporter et modéliser l'expertise des différents experts.

— Ces solutions n'ont pas non plus été fournies dans le cadre de Framework sémantiques dont le but serait de délimiter l'ensemble des préoccupations, de la terminologie et des domaines ou champs sémantiques concernés par la conception/fabrication et développement d'écosystèmes et/ou d'objets intelligents.

— Elles n'ont pas été intégrées, à notre connaissance, dans le cadre de méthodologies de conception/développement d'Objets Intelligents et d'écosystèmes.

— De plus, il est très important de signaler que les méthodologies de conception de solutions IdO ou d'écosystèmes actuelles, combinent des approches de conception logicielle (comme par exemple, les approches en cascade, les approches agiles ou encore celles qui adoptent la construction/déploiement continus de logiciels) à des approches d'intégration matériel/logiciel, sans pour autant se focaliser ou éventuellement aborder les aspects liés à la conception/fabrication d'objets intelligents à intégrer.

— Les détails relatifs à la description d'un objet intelligent d'un point de vue structurel et comportemental n'ont pas été l'objectif principal de ces méthodologies.

— Aucune prescription ou recommandation quant à la décomposition et la définition de la logique comportementale des objets intelligents (c-à-d. la logique de traitement embarquée) n'est fournie par ces méthodologies.

— Finalement, ces méthodes n'intègrent pas des approches ou des descriptions des mécanismes de reconfiguration ou d'adaptation du comportement de ces objets intelligents. Elles ne prescrivent pas les composants de l'écosystème qui devront avoir cette responsabilité et sous quelles conditions ?

Nous pouvons donc résumer la problématique de ce travail de recherche au niveau de la question suivante :

> Quelles approches, méthodologies et outils peut-on proposer pour permettre à des acteurs multidisciplinaires de concevoir, fabriquer, déployer et adapter des objets et/ou des systèmes intelligents et connectés tout en tenant compte de leurs caractéristiques structurelles et comportementales, de leurs différents cas d'usage ainsi que du domaine de leur utilisation ?

1.3.2 Verrous scientifiques

Le Framework de modélisation d'objets intelligents ainsi que l'écosystème qui est destiné à exploiter, adapter et/ou reconfigurer ces derniers constituent deux environnements que nous souhaitons être en perpétuelle synergie, où les limites et/ou l'évolution de l'un peuvent influencer celles de l'autre. Nous proposons donc que ces deux environnements sont fondés fortement sur des modèles ontologiques. Ceci, afin de réduire d'une part l'écart terminologique entre l'ensemble des acteurs. D'autres part, grâce aux mécanismes de raisonnement permis par ces modèles ontologiques, la prise de décision quant à la conception et aux choix des composants s'en sera facilitée.

Ces modèles ontologiques permettent de plus, de part leurs structures et l'expressivité de leur langages, de tenir compte plus facilement de l'évolution de la connaissance experte des différents acteurs et ainsi d'évoluer de manière simultanée à celle des usages et de la technologie sous-jacente

aux réseaux de capteurs et aux objets intelligents. L'ensemble de ces modèles ontologiques peuvent servir alors de méta modèles ou alors de guides pratiques aux concepteurs et développeurs.

De plus, étant donné que la structure et le comportement du VI ainsi que ceux de l'écosystème sont fondés sur ces modèles ontologiques il sera d'autant plus naturel de modéliser les mécanismes d'adaptation et de reconfiguration avec les mêmes modèles ontologiques. Ce dernier permettra de plus d'exprimer des stratégies et des règles de reconfiguration et d'adaptation qui pourront évoluer en même temps que le système. Nous définissons ainsi l'ensemble des verrous scientifiques suivants :

1. Le premier verrou concerne la modélisation ontologique de systèmes ou d'objets intelligents connectés et de l'expertise sous-jacente à leur conception/fabrication et exploitation.
2. Le deuxième verrou concerne la modélisation ontologique des mécanismes d'évolution, d'adaptation et/ou de reconfiguration de systèmes ou d'objets intelligents connectés.
3. Le troisième verrou concerne le fondement méthodologique liés à la conception de l'ensemble de ces modèles ontologiques ainsi que les environnements qui les utilisent ou les mettent en œuvre.

1.3.3 Questions de recherche

Les questions de recherche auxquelles nous essayerons d'offrir des réponses tout au long de ce manuscrit sont les suivantes :

1. Quels sont les composants d'un Framework sémantique générique de modélisation d'un objet intelligent connecté et quels sont les principes qui le régissent ?
2. Quels sont les modules ontologiques nécessaires pour former la composante sémantique de ce Framework ?
3. Quel fondement méthodologique pour la conception développement de chacun de ces modules ontologiques, et comment serait-il possible de les réutiliser dans différents contextes ?
4. Comment exploiter ces modules ontologiques pour enrichir et supporter méthodologiquement la conception/fabrication d'objets intelligents ainsi que la conception et le développement d'écosystèmes qui permettrait d'intégrer ces objets ?
5. Comment exploiter les modules ontologiques pour proposer des mécanismes d'adaptation/reconfiguration des objets intelligents et de leurs comportements ?

1.4 Axes de recherche et orientations

En vue de fournir des éléments de solution à la problématique et aux verrous scientifiques soulignés précédemment, les quatre axes de recherche suivants sont traités :

- Le premier axe porte sur l'identification et la spécification des principaux éléments conceptuels composant le Framework générique de modélisation sémantique des objets intelligents et connectés. Il s'intéresse également à préciser le fondement méthodologique de sa conception.

- Le deuxième axe concerne la définition et/ou la mise en œuvre de méthodologies pour la conceptualisation de la composante sémantique du Framework.
- Le troisième axe concerne l'identification et l'application de méthodologies qui soient fondées sur les modules ontologiques et qui vont guider la conception et développement de la plateforme ou de l'écosystème IdO. Ces méthodologies doivent également exploiter les modules ontologiques pour spécifier des mécanismes d'adaptation et de reconfiguration automatique d'écosystèmes.
- Le quatrième axe concerne les principes qui vont guider le développement de l'aspect comportemental des objets intelligents (c-à-d. les logiques de traitement embarquées). Il concerne également les approches permettant de capitaliser ce développement, en particulier, en construisant et gérant des répertoires de logiques de traitement (fragments de code) réutilisables et à granularité fine. Le but étant de permettre à différents développeurs de contribuer à l'enrichissement des fonctionnalités des objets intelligents en développant leurs propres logiques de traitement embarquées. La publication de ces fragments de code dans des répertoires sémantiques facilitera leur recherche et chargement. Ils permettront à terme aux objets intelligents de s'auto-configurer et de changer de comportement sans aucune intervention du développeur ou de l'utilisateur final.

1.5 Positionnement

Nous pouvons considérer que les travaux de cette thèse se positionnent dans le cadre général d'approches méthodologiques et de Framework sémantiques de conception, de développement et d'exploitation de produits tangibles intelligents (smart objects). Ces approches méthodologiques sont surtout centrées et dirigées par les caractéristiques de l'objet intelligent, de ses attributs, ses exigences vis à vis de l'ensemble des parties prenantes ainsi que ses différents usages et applications. Elles visent, en particulier, à offrir un ensemble de modules ontologiques permettant de décrire sémantiquement les aspects structurels et comportementaux de ces objets. Elles mixent, pour cela, un ensemble de fondements théoriques liés d'une part :

— aux approches et Framework de gestion de connaissances et plus spécifiquement de définition, extraction, modélisation et représentation sémantique de connaissances

— d'autres part aux architectures et infrastructures distribuées de systèmes (ou écosystèmes) permettant, l'intégration, la gestion, le pilotage d'objets intelligents ainsi que la (re) configuration et l'adaptation de l'écosystème.

— et finalement aux approches fondées sur les modèles ontologiques pour élaborer des stratégies de prise de décision guidant chacune des étapes du cycle de vie de l'objet intelligent depuis sa conception, développement et exploitation jusqu'à sa mise à jour, adaptation, reconfiguration et voire même son recyclage à terme.

1.6 Principales contributions

Les contributions de cette thèse, s'articulent tous autour de la proposition d'un Framework sémantique structuré en un ensemble de composants pour la conception/fabrication et l'exploitation d'un Objet Intelligent. Ce Framework nommé FSMS est concrétisé dans la contribution principale de ce travail et qui est le développement d'une ontologie modulaire que nous avons baptisé SMartSensing (SMS). L'ontologie modulaire SMS est considérée comme la pierre angulaire qui a guidé l'émergence de l'ensemble des autres contributions de cette thèse et qui sont les suivantes :

1. Une **méthodologie de modélisation d'un Objet Intelligent** a été proposée. Celle-ci suggère un ensemble d'activités incluant des itérations successives d'analyse et de raffinement des composantes structurelles et comportementales de l'objet considéré. Son application nous a permis d'identifier les composants fondamentaux du Framework sémantique et de l'ontologie SMS.
2. En vue de supporter cette méthodologie, **Un ensemble de processus permettant d'exploiter l'ontologie SMS en vue de répondre aux besoins de conception/fabrication ou de réutilisation et d'adaptation** d'un objet intelligent a été proposé.
3. Une **architecture** fondée sur l'ontologie SMS a été proposée et a été appliquée dans le contexte du projet SmartSensing pour supporter le déploiement de vêtements intelligents et reconfigurables.
4. Celle-ci trouve ses fondements dans une **méthodologie de conception et de développement de plateformes ou écosystèmes IdO** que nous avons enrichis sémantiquement à travers l'usage de l'ontologie SMS.
5. Finalement, une **approche de reconfiguration/adaptation du fonctionnement d'un objet intelligent** à différents contextes d'usage est proposée. Celle-ci est à son tour fondée sur les modules SMS et suppose la construction et l'évolution progressive de répertoires de logiques de traitement à embarquer dans les composants de l'écosystème IdO.

1.7 Organisation du document

A part cette introduction, le présent manuscrit est organisé en 6 chapitres. Le chapitre 2 traite de l'état de l'art relatif aux Frameworks sémantiques de modélisation des objets intelligents connectés, ou IdO ou encore objet pervasifs. Il traite également des fondements méthodologiques liés à la conception ou construction de modèles ontologiques. Le chapitre 3 constitue la première contribution de ce travail, il décrit donc le Framework sémantique ainsi que l'ensemble des principes et des fondements théoriques qui le dirigent. Le chapitre 4 décrit en détail l'ontologie modulaire SMS (SMartSensing), en particulier il décrit ses différents modules sémantiques et leurs règles d'interaction et/ou médiation. Le chapitre 5 s'intéresse à définir l'écosystème D-SHIRT ou la plateforme IdO d'exploitation du VI ou PI. Il décrit son architecture de base et dresse une analyse de choix et de compromis entre

différentes alternatives. Le chapitre 6 décrit quant à lui l'architecture expérimentale ou le prototype de test. Il décrit également l'approche de validation de l'ensemble du Framework méthodologique et des modules ontologiques qui le supportent. En fin au niveau du dernier chapitre nous dressons le bilan de l'ensemble des travaux de cette thèse et nous soulignons les perspectives possibles à ces travaux.

Première partie

État de l’art

Dans le cadre du présent travail, nous nous intéressons à la proposition d'un Framework sémantique pour la conception/fabrication et l'exploitation des objets et les systèmes intelligents connectés. En suivant notre démarche, définie dans le chapitre 1, nous réalisons dans cette partie une investigation plus poussée de la problématique à travers une revue de la littérature des travaux connexes.

Nous avons élaboré un état de l'art sur :

— la modélisation sémantique dans le contexte de l'IdO, nous nous sommes focalisés sur les ontologies de capteurs existantes, sur leur généricité, leur réutilisation et leur portée et capacité à représenter les capteurs intelligents et leur caractéristique d'adaptabilité ;

— les méthodologies et les Frameworks existants pour l'IdO, nous avons mis l'accent sur leurs aspects sémantique et générique, sur leur niveau d'intervention dans la conception d'un OI et leur capacité de garder trace de l'expertise des différents intervenants ;

— les plateformes de l'IdO où l'intérêt s'est porté sur la reconfiguration/adaptation sémantique des OI et des solutions IdO.

Une synthèse a été proposée à la fin de chaque étude.

Chapitre 2

Les Frameworks et plateformes IdO basés sur le Web sémantique

Sommaire du présent chapitre

2.1 Introduction

Cette partie, est consacrée à l'investigation des travaux de recherche ayant traité et apporté d'une certaine manière une réponse à la problématique de cette thèse. L'objectif étant d'approfondir davantage cette problématique, d'analyser et de mettre en avant les avantages et les limites des travaux existants et enfin à mieux positionner nos contributions. Étant donné que le projet SmartSensing représente un contexte applicatif à cette thèse, mis à part que nos contributions puissent avoir un caractère théorique, nous visons également apporter des contributions d'ordre pratique qui serviront dans le

cadre de projets similaires. Nous rappelons que notre problématique concerne essentiellement les approches, méthodologies et outils qui permettront à des acteurs multidisciplinaires de concevoir, fabriquer, déployer et adapter des objets et/ou des systèmes intelligents et connectés.

Nous commençons donc ce chapitre par dresser une carte conceptuelle relative aux travaux connexes que nous allons étudier, comparer et évaluer par rapport à des critères que nous allons définir pour chaque catégorie de travaux de notre carte conceptuelle. Cette dernière est illustrée dans la figure 2.1. Nous organisons ensuite le chapitre en fonction de cette carte conceptuelle. Une première section traite des travaux ayant défini les différents concepts et principes liés à l'Internet des Objets (IdO) en mettant l'accent sur les différentes visions et orientations de la recherche dans ce domaine. La deuxième section, se focalise sur les approches de modélisation des connaissances pour l'IdO et des réseaux de capteurs. Elle met l'accent sur les travaux ayant proposé des approches basées sur le Web sémantique comme par exemple les modèles sémantiques ou les ontologies. Dans la troisième section, nous présentons les différents Frameworks et méthodologies pour l'IdO. Nous essayons d'évaluer ces Frameworks quant à la méthodologie qui les supporte ainsi que leur capacité à garder trace de toute l'expertise de leurs utilisations ou alors leur capacité à permettre la participation collaborative d'acteurs multidisciplinaires et si éventuellement ils se fondent sur des représentations ou des technologies du Web sémantique. Finalement la quatrième section s'intéresse aux plateformes IdO. Elle se focalise essentiellement sur les approches d'adaptation et de reconfiguration embarquée dans les plateformes.

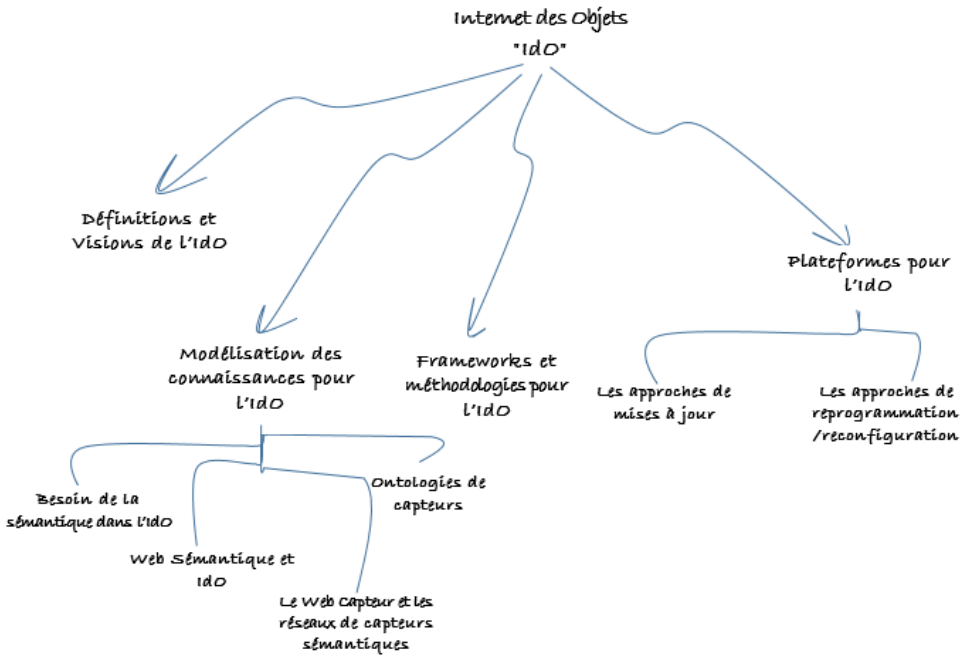

FIGURE 2.1 – Carte conceptuelle relative aux travaux connexes

2.2 L'Internet des Objets : Définitions et visions

L'Internet des Objets (IdO) est un concept créé et développé durant ces dernières années. Il a été proposé pour la première fois par les laboratoires MIT Auto-ID en 1999. Selon les auteurs dans [6], l'IdO fait référence à des objets adressables de manière unique et à leurs représentations virtuelles dans une infrastructure de type Internet. L'IdO, également appelé «Réseau de détection» (*Sensing Network*), fait référence à un système de gestion et de contrôle de réseau dans lequel des objets se connectent les uns aux autres en installant divers types d'équipements de détection d'information. L'identification par radiofréquence (*Radio-Frequency Identification Technology* ; RFID), les capteurs omniprésents, les codes à barres ou les codes 2D représentent des technologies adaptées pour l'IdO pour la détection d'informations. Les concepts d'informatique mobile et d'informatique omniprésente sont également liés à l'IdO. Ceux-ci se basent sur l'interconnexion de dispositifs de détection embarqués à grande échelle, qui sont reliés à Internet et qui forment un réseau intelligent géant, permettant d'offrir des mécanismes dits sensibles au contexte.

Le concept d'IdO varie toutefois d'un domaine de recherche à l'autre. Définir ce concept est une tâche difficile d'autant plus que ce concept évolue avec l'évolution du matériel et du logiciels (à l'instar de nombreux autres concepts liés à l'électronique et / ou au calcul) et des technologies ICT.

Le groupe IEEE Internet of Things a compilé les définitions de diverses associations d'Internet et groupes de recherche dans la publication "*Towards a definition of the Internet of Things*." [7]. Nous relevons ici les définitions qui nous semblent les plus pertinentes pour cette thèse :

(1) "*L'idée de base est que l'IdO connectera les objets autour de nous (électroniques, électriques, non électriques) pour fournir une communication transparente et des services contextuels. Le développement d'étiquettes RFID, de capteurs, d'actionneurs et de téléphones portables permet de matérialiser un IdO qui interagit et coopère pour rendre un service meilleur qui soit accessible à tout moment, de n'importe où.*" —Internet Engineering Task Force (IETF), 2010 ;

(2) "*réseau d'éléments—chacun intégré avec des capteurs—qui sont connectés à Internet.*" — Institute of Electrical and Electronics Engineers (IEEE), 2014 ;

(3) "*L'Internet des objets fait référence à l'identification unique et à l''Internetization' des objets du quotidien. Cela permet une interaction humaine et le contrôle de ces "objets" de partout dans le monde, ainsi qu'une interaction entre appareils sans intervention humaine.*" — HP, 2014 ;

(4) "*L'idée de base de ce concept est l'omniprésence (pervasive presence) autour de nous d'une variété d'objets - tels que l'identification par radiofréquence (RFID), des capteurs, des actionneurs, des téléphones portables, etc - qui, à travers des systèmes d'adressage uniques sont capables d'interagir les uns avec les autres et de coopérer avec leurs voisins pour atteindre des objectifs communs*" [4][5].

D'autres concepts en relation avec l'IdO sont également apparus, parmi lesquels nous citons les objets **pervasifs**, objets **connectés**, objets **communicants** et objets **intelligents**. Ces objets pervasifs de l'IdO sont des objets du quotidien (par exemple, électroménager, montre, voiture, vêtement,

etc.) intégrant des composants électroniques (capteurs) et peuvent ainsi détecter l'environnement et communiquer avec d'autres objets. Ils sont capables de comprendre des scénarios complexes, et souvent y apporter des réponses autonomes, sans l'intervention humaine. Les applications exploitant ces informations sont de plus en plus populaires et touchent plusieurs secteurs comme la santé ou le transport. Connectés à Internet, ces objets communiquent, *via* le réseau, les données détectées dans le Cloud et font ainsi partie de l'infrastructure de l'Internet des Objets. Ces objets sont alors qualifiés de connectés ou communicants, et par abus de langage ils sont également dits intelligents. Cisco [1] estime que plus de 50 milliards d'appareils seront connectés à Internet d'ici 2020 et les évolutions technologiques très rapides donneront naissance à d'autres types d'objets connectés aux technologies plus sophistiquées. Ce sont ces même objets qui sont utilisés en informatique pervasive ou omniprésente.

Nous notons toutefois l'existence de plusieurs visions de l'IdO. En effet l'augmentation du nombre d'objets intelligents, communicants, etc., nécessite des capacités centrées sur des volumes de données très importants. Ceux-ci sont le principal moyen de communication entre les différentes entités. Par conséquent, la capacité de recueillir, gérer, indexer, interroger et traiter de grandes quantités de données est essentielle [8]. Afin d'atteindre ces objectifs, plusieurs travaux de recherche ont été entrepris touchant divers aspects créant ainsi des visions différentes et conjointes dans le domaine de l'IdO. Dans [5], les auteurs ont classé l'IdO selon trois paradigmes, à savoir : i) l'orienté Internet (*Internet-oriented*), ii) l'orienté objet (*Things-oriented*) (capteurs) et iii) l'orienté sémantique (*Semantic-oriented*).

- La vision orientée objets prend en charge l'utilisation de l'"*Electronic Product Code* (EPC) en conjonction avec la technologie RFID pour la collecte et le suivi des données de capteurs ;
- La vision orientée Internet correspond à la construction des protocoles IP pour permettre aux objets intelligents de se connecter à Internet ;
- La vision orientée sémantique aborde les problèmes de gestion de données qui surviennent dans le contexte des grandes quantités d'informations échangées par des objets intelligents et des ressources disponibles via l'interface Web. L'idée est que des descriptions standardisées de ressources sont essentielles pour permettre l'interopérabilité des ressources hétérogènes disponibles.

Choix terminologique

Étant donné la diversité des terminologies dans le domaine de l'Internet des Objets et tout en considérant le domaine spécifique de cette thèse, nous adaptons le terme "Objet Intelligent" pour spécifier les objets, appareils ou tout périphérique intégrant des capteurs et offrant des services aux utilisateurs.

Tout au long de cette thèse nous adoptons la définition suivante :"L'Internet des Objets est l'infrastructure permettant, à ces objets intelligents d'interagir entre eux *via* Internet avec d'autres systèmes et en particulier avec l'Homme".

1. http ://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/iot-aag.pdf

La diversité et l'hétérogénéité des composants mis en jeu dans l'IdO nous mènent également au choix d'une vision sémantique. En effet, dans notre travail, nous nous intéressons plus particulièrement à la représentation et l'intégration sémantique de diverses données descriptives liées aux objets intelligents, et à la gestion et la représentation des données provenant des divers capteurs intégrés à ces objets, d'autre part.

2.3 La Modélisation des connaissances pour l'Internet des objets

Dans le but de modéliser les connaissances liées aux OI ou encore aux systèmes intelligents connectés, nous parcourons dans cette partie de l'état de l'art les principaux concepts définis dans le cadre de l'IdO. Les travaux qui ont porté sur l'intégration de la sémantique aussi bien pour les réseaux de capteurs que pour l'IdO. Cette investigation nous permettra, premièrement, d'identifier les concepts intrinsèques de l'IdO et les caractéristiques des capteurs comme composant essentiel, intégré dans les objets de l'IdO. Deuxièmement, de répondre à notre problématique concernant l'existence de solutions sémantiques prêtant attention à l'interopérabilité entre les différentes données transmises par des capteurs, à la conception et le développement de capteurs ou d'objets intelligents, ou encore à des solutions ontologiques permettant de décrire les capteurs intelligents et leur logique de traitement. Ainsi dans ce qui suit, nous traiterons des besoins inhérents de l'intégration et de la conjonction des techniques du Web sémantique avec ceux de l'IdO et des réseaux de capteurs. Nous nous intéresserons plus particulièrement aux ontologies.

2.3.1 Contexte et besoin de la sémantique dans l'Internet des Objets

Les capteurs déployés dans les infrastructures IdO sont polyvalents et hétérogènes, ils peuvent être exploités dans plusieurs domaines. Des exemples témoignant de leurs succès prometteurs, sont les applications dans le suivi environnemental, l'agriculture, la surveillance et la détection d'intrusions, la santé, la domotique, la sécurité publique et les trafics urbains. Ces infrastructures IdO deviennent ainsi une source importante d'énormes quantités de données, mesurant un large éventail de propriétés physiques. Dans des déploiements à grande échelle, il devient difficile d'intégrer et d'exploiter efficacement ces flux de données complexes et hétérogènes. Cette utilisation croissante des capteurs augmente considérablement la difficulté des applications à gérer et à interroger les données capteurs [9]. Cette difficulté est d'autant plus perceptible lorsque les applications ont besoin de rechercher un ensemble d'informations particulières sur des réseaux de capteurs fédérés et hétérogènes, offrant d'énormes volumes de données capteurs pour de grandes communautés d'utilisateurs [10]. Dans ces environnements, les capteurs de différents fournisseurs ayant des caractéristiques spécifiques sont installés et ajoutés à un système. Chacun d'eux produit des valeurs différentes, avec différents formats de données, différentes précisions ou exactitudes, et dans différentes unités de mesure. Cette hétérogénéité complique la tâche d'interrogation des données capteurs ainsi que les métadonnées correspondantes.

Par conséquent, il est important d'interpréter les données capteurs en fonction des informations pertinentes pour les applications déployées. Une description sémantique des nœuds de ces réseaux de capteurs ainsi que les flux de données produites, améliorera nettement la capacité de recherche, de chargement, de contrôle, de fusion et d'interprétation de ces données collectées. L'intégration intelligente de ces données capteurs facilitera alors la découverte de connaissances à valeur ajoutée. Dans [11], les auteurs affirment également, que pour rendre les flux de données capteurs utilisables en tant qu'une nouvelle source clé de connaissance, il est important de les intégrer dans l'espace d'informations existant, c'est-à-dire le Web. Cela permettra d'accroître l'interopérabilité entre les différents types de capteurs ainsi que de fournir des informations contextuelles indispensables à la connaissance de la situation dans le réseau de capteurs. En outre, des méthodes appropriées de traitement des données peuvent se révéler utiles, en particulier dans les cas où les données provenant de plusieurs sources hétérogènes doivent être comparées et/ou combinés et différents événements doivent être corrélés. Dans cette directive, l'Open Geospatial Consortium (OGC) a développé le *Sensor Web Enablement* (SWE) [2], ensemble de spécifications, de modèles de données et des services Web relatifs aux capteurs, qui permettra l'accessibilité et le contrôle de ces données et services via le Web. Le *Sensor Web* est un type particulier d'infrastructure d'informations centrée-Web, pour la collecte, la modélisation, le stockage, la récupération, le partage, la manipulation, l'analyse et la visualisation des informations sur les capteurs et sur les observations et les mesures des phénomènes détectées par ces capteurs.

Une technologie prometteuse qui est en mesure de relever les défis techniques d'extraction d'événements significatifs et permettant une interopérabilité entre les données provenant de différentes sources, est le Web sémantique. L'annotation sémantique sous la forme de métadonnées peut être ajoutée à n'importe quel Framework. En vue d'ajouter des sémantiques bien définies l'utilisation des ontologies pourrait améliorer l'usage des connaissances extraites à partir des informations disponibles ainsi que l'ajout de relations entre les données contextuelles et les règles d'application définis. Récemment, des efforts considérables ont été présentés visant à lever les flux de données ainsi que les caractéristiques de leur acquisition et de leur transmission à un niveau sémantique. Le *Semantic Sensor Web* (SSW) [12] est une extension du *Sensor Web*, où les nœuds de capteurs pourront découvrir leurs capacités respectives et échanger, automatiquement, des données et des processus, sans l'intervention humaine. Les données capteurs seront dotées de métadonnées sémantiques qui permettront d'augmenter l'interopérabilité et de fournir des informations contextuelles indispensables à la connaissance de la situation.

Dans ce cadre, beaucoup de travaux ont été développés, nous présentons dans ce qui suit un ensemble de concepts et de standards émergeant de la fusion des réseaux de capteurs et de l'IdO avec le domaine du Web sémantique.

2. http ://www.opengeospatial.org/ogc/markets-technologies/swe

Le web de capteurs et les réseaux de capteurs sémantiques (ou Sémantistion des réseaux de capteurs)

Le "Web de capteurs" (*Sensor Web*). Le terme *Sensor Web* a été utilisé à l'origine pour décrire une architecture de réseau de capteurs sans fil dans lequel les nœuds du réseau sont autonomes et capables de réagir au niveau de leurs données mesurées ainsi que celles des autres nœuds du réseau [13]. Depuis, le terme a été utilisé de façon plus générale, le décrivant comme une nouvelle tendance qui permet à des capteurs de différents types et des référentiels de données capteurs résidents sur le Web d'être intégrés dans une plate-forme uniforme pour être accessibles, contrôlées et identifiées dynamiquement via le *World Wide Web* [14]. Les auteurs dans [15], le décrivent comme une architecture de services web distribuée pour la publication, la découverte, et la combinaison de données provenant des réseaux de capteurs multiples et des sources de données liées.

De façon plus générale, les principales caractéristiques d'une architecture Web capteur sont sa capacité à [15] :

1. identifier les sources de données pertinentes ;
2. accéder aux données capteurs en temps *quasi* réel conjointement avec des données contextuelles ;
3. combiner et corréler des données provenant de sources disparates avec des modalités différentes (par exemple, un flux de données d'un capteur avec des informations contextuelles stockées dans une base de données) ;
4. permettre aux utilisateurs et aux fournisseurs de données de travailler avec leur propre conceptualisation de données, c'est-à-dire, ne pas forcer les utilisateurs et les fournisseurs de données à utiliser un modèle de données commun, d'autant plus que les sources de données ne seront pas sous le contrôle des utilisateurs, et pourront déjà leurs être publiées en fonction de leur propre conceptualisation.

Les réseaux de capteurs sémantiques (*Semantic Sensor Networks*). Le terme *Semantic Sensor Networks* a été introduit dans les travaux du *W3C Semantic Sensor Network Incubator Group*. Un réseau de capteurs sémantiques utilise des descriptions déclaratives du réseau, des capteurs, des services, ainsi que celle du domaine et sa relation avec les observations et les mesures détectées [16]. Ces réseaux favorisent la réutilisation et l'intégration afin d'aider à résoudre les difficultés d'installation, de recherche, d'interrogation, de gestion et de maintien des réseaux de capteurs complexes et hétérogènes. Ainsi, ils peuvent offrir l'analyse, l'interprétation et la prédiction de données en temps réel et historiques. Ils sont généralement, au niveau de la littérature, confondus au *Semantic Sensor Web*.

Les architectures des réseaux de capteurs sémantiques [17], [16] utilisent de multiples couches sémantiques et des technologies pour fournir une infrastructure de services. Ces couches permettent de représenter les capteurs, les données, les traitements et l'application.

[17], [12] démontrent les capacités potentielles des réseaux de capteurs sémantiques :

1. Classer les capteurs selon la fonctionnalité, la sortie, ou la méthode de mesure. Nécessite des spécifications interprétables des capteurs, de leurs types de sortie et les domaines dans lesquels

ils opèrent ;

2. Trouver les capteurs capables d'effectuer une mesure particulière, ou qui peuvent fournir une mesure particulière dans un format particulier. Nécessite les mêmes spécifications que 1. Cependant, un système pourrait faire plus que rechercher des capteurs existants, il pourrait composer des capteurs existants et des flux de données pour créer des capteurs virtuels. Les incompatibilités de format de données pourraient également être supprimées en composant des fonctions de transformation appropriées ;
3. Collecter des données spatiales, temporelles, ou encore de précision. Nécessite des spécifications de capteurs qui incluent les localisations, l'exactitude et la modélisation des données d'observation ;
4. Inférer les connaissances du domaine à partir des données de bas niveau. L'inférence nécessite un mécanisme de raisonnement, des spécifications du domaine et des capteurs et les données annotées ;
5. Produire un événement lorsqu'une condition particulière est atteinte dans un délai. Nécessite les spécifications dans les cas d'utilisation antérieures, ainsi que le traitement des requêtes, la gestion de l'énergie et de la configuration, et les spécifications du capteur qui comprennent l'énergie, les conditions de fonctionnement du capteur et les durées de vie. Les capacités connexes pourraient inclure les capteurs trouvés pour satisfaire des tâches particulières, et utilisant un raisonnement pour aider à planifier un déploiement.

Les Web de capteurs sémantiques (*Semantic Sensor Web*). Le terme *Semantic Sensor Web* a été inventé par les auteurs dans [12] combinant la technologie du Web capteurs avec celle du Web sémantique. Une première étape vers la réalisation du Web capteur sémantique a été présentée par [18] en introduisant un service d'observation capteur (*Sensor Observation Service*) sémantique, appelé SemSOS qui annote sémantiquement les réponses du service. Après cette étape, de nombreux autres travaux ont adopté l'idée d'appliquer les technologies sémantiques pour les différentes tâches liées à la découverte et l'intégration des données issues de sources de données hétérogènes autonomes. Le travail présenté dans [19] utilise des descriptions sémantiques des déploiements de capteurs pour accélérer la découverte et la conversion syntaxique des données capteurs pour une publication dans un Web capteur OGC SWE (*OGC-SWE sensor web*). Dans [9], cinq challenges sont identifiés pour ajouter de la sémantique au Web capteur.

1. Le niveau d'abstraction dans lequel les données des capteurs peuvent être obtenues, traitées et gérées ;
2. La nécessité de traiter de manière adéquate la qualité des données ;
3. L'intégration et la fusion de données provenant des réseaux de capteurs hétérogènes et déployés de façon autonome ;
4. L'identification et la localisation des sources de données pertinentes à base de capteurs ;
5. Le développement rapide d'applications qui sont basées sur ces types de sources de données.

Les standards SWE

L'objectif de l'initiative *Sensor Web Enablement* (SWE) de l'Open Geospatial Consortium (OGC) [20] est de définir des interfaces de services Web et les encodages de données pour rendre les capteurs accessibles via le Web. Les spécifications SWE offrent une communication et une interaction standardisées entre différents types de capteurs et de systèmes de capteurs.

Observations & Measurements Schema (O& M). Modèles standards et schéma XML pour les encodages des observations et des mesures d'un capteur, aussi bien historiques qu'en temps réel.

Sensor Model Language (SensorML). Modèles standards et schéma XML décrivant les processus de traitement et les systèmes de capteurs, fournissent les informations nécessaires pour la découverte de capteurs, leur localisation, le traitement des observations au niveau des capteurs et listent les propriétés de tâches.

PUCK. Définit un protocole pour récupérer une description SensorML, le code "driver" du capteur, et d'autres informations à partir de l'appareil (device) lui-même, ce qui permet l'installation, la configuration et l'exploitation automatiques du capteur.

Transducer Markup Language (TransducerML ou TML). Modèle conceptuel et schéma XML pour décrire en temps réel les transformations et le support des flux de données des systèmes de capteurs.

Sensor Observation Service (SOS). Interface ouverte pour un service Web permettant d'obtenir des observations et des descriptions des capteurs et de la plate-forme à partir d'un ou de plusieurs capteurs.

Sensor Planning Service (SPS). Une interface ouverte pour un service Web par lequel un client peut 1) déterminer la faisabilité de la collecte de données à partir d'un ou plusieurs capteurs ou de modèles et 2) soumettre des demandes de collection).

Sensor Alert Service (SAS). Interface standard de service Web pour la publication et l'acceptation des alertes provenant des capteurs.

Web Notification Services (WNS). Interface standard de service Web pour la distribution asynchrone des messages ou des alertes des services Web SAS et SPS.

Le Web sémantique et l'internet des objets (De l'IoT au SWoT)

Dans ce contexte, les principaux défis des travaux développés visent à relier des domaines hétérogènes entre eux et interpréter les données IdO. Les auteurs, dans [21], soulignent clairement le manque de normalisation des modèles et des formats des données IdO. Ils décrivent le besoin : (1) d'applications inter-domaines (cross-domain applications), (2) d'une gestion des données IdO et de l'interopérabilité sémantiques pour l'échange et l'analyse de ces données afin de déduire des informations utiles et d'assurer l'interopérabilité entre les applications IdO ainsi que le raisonnement, et (3) de la sécurité et la vie privée par la conception d'architecture IdO et plus précisément, la confidentialité des données. Dans [22], les auteurs introduisent la nécessité d'un traitement intelligent des données IdO et expliquent le problème lié aux applications spécifiques au domaine (*domain specific-applications* en Anglais) : les

applications ne peuvent pas combiner les données provenant de différents sources ("cannot correlate and integrate the data from different silos and getting the data from heterogeneous sources"). Selon les auteurs dans [23], l'un des défis majeurs actuels de l'IdO est de connecter les objets du monde réel au Web, afin de recueillir leurs données et de les traiter pour créer des applications intelligentes.

D'où l'émergence du concept le "Web des Objets" (*Web of Things* (WoT) en Anglais). Ce dernier assure l'interopérabilité entre les périphériques matériels et les protocoles de communication ou d'application. L'objectif est d'accéder à des périphériques et aux données du Web. Les plates-formes du WoT existantes traitent les protocoles hétérogènes et le partage de données capteurs sur le Web. Cependant, elles n'utilisent pas les technologies du Web sémantique. La notion du WoT était introduite par les auteurs dans [24], et dont l'idée était de relier les objets physiques sur le Web en utilisant le RESTful. L'étude faite dans [25] sur le WoT met en évidence les problématiques suivantes : (1) l'hétérogénéité des dispositifs et des protocoles de communication, (2) la sécurité et la vie privée, (3) la découverte de capteur, (4) la sensibilité au contexte pour adapter l'environnement en fonction du profil utilisateur, et (5) les services du Web sémantique adaptés au WoT. L'utilisation de la sémantique est également fortement recommandée par le W3CWeb of Things [3].

L'intégration des techniques du Web sémantique à l'IdO donne naissance au nouveau paradigme, le "Web Sémantique des Objets" (*Semantic Web of Things* (SWoT) en Anglais). Le SWoT est un domaine de recherche récent. Il s'agit d'une évolution du Web des objets en intégrant la sémantique à l'IdO. L'objectif du SWoT est de fournir de l'interopérabilité entre les ontologies et les données. Le SWoT a été introduit en 2011, dans le contexte du projet Spitfire [26]. Parmi les travaux au niveau de SWoT, on note [27] où les auteurs présentent un cadre pour le Web Sémantique des Objets (*Semantic Web of Things framework*) qui vise à enrichir les objets du monde réel avec des annotations sémantiques.

Dans cette directive de nombreuses applications utilisent les technologies du Web sémantique dans la recherche IdO, en particulier l'ontologie SSN pour l'annotation de capteurs et les réseaux de capteurs ; les données liées (*Linked Data* en Anglais) [28] pour la publication de données de capteurs (*sensor data publishing* en Anglais) [29] et de la découverte (*discovery* en Anglais) [30], et les services d'observation de capteurs sémantiques (*semantic sensor observation services* (SemSoS) en Anglais) [18].

Synthèse

L'intégration des techniques du Web sémantique aux réseaux de capteurs et à l'IdO a permis de résoudre un certains nombres de problèmes par les solutions précédemment présentées. Ces travaux ont surtout porté sur la gestion des ressources IdO. Ils se sont principalement focalisés sur l'interopérabilité des données transmises des capteurs, l'hétérogénéité des capteurs déployés, l'échange, le contrôle et l'accessibilité de ces données et services via le Web. Par contre, aucun de ces travaux ne s'est intéressé aux défis sémantiques que nous nous sommes posés, qui sont :

3. http ://www.w3.org/2014/02/wot/

— la modélisation conceptuelle de systèmes ou d'objets intelligents connectés et de l'expertise sous-jacente à leur conception/fabrication et exploitation ;

— la modélisation conceptuelle des mécanismes d'évolution, d'adaptation et/ou de reconfiguration de systèmes ou d'objets intelligents connectés.

2.3.2 Les ontologies de capteurs

Dans l'objectif de modéliser les OI et les systèmes intelligents connectés, nous nous focalisons principalement à décrire sémantiquement le composant clé de ces objets à savoir les capteurs classiques de façon générale et les capteurs intelligents adaptables/reconfigurables plus particulièrement. Nous faisons ainsi recours aux ontologies. Les ontologies fournissent un modèle formel et extensible qui convient à la représentation des informations, telles les données capteurs, à différents niveaux d'abstraction et avec de riches descriptions sémantiques qui peuvent être utilisés pour la recherche et le raisonnement [9].

Nous présenterons dans ce qui suit les concepts clés que devrait couvrir les ontologies de capteurs suivis de la description d'un certain nombre d'entre elles.

Concepts pour la modélisation des réseaux de capteurs

Généralement, on peut voir un capteur comme constitué de deux principales grandes parties : la première est celle qui spécifie le dispositif ou la plate-forme de détection, la deuxième décrit les mesures et les observations détectées par ce dispositif, toutes deux souvent connues comme des données capteurs. Par exemple, si nous prenons le cas de l'utilisation d'un capteur de température pour mesurer la température de l'eau : si quelqu'un est à la recherche d'un capteur qui peut être utilisé dans un environnement spécifique, dans ce cas, les informations telles que la température détectée, la profondeur exploitée ou les dimensions peuvent être importantes. Pour quelqu'un qui traite les données mesurées, il peut être plus utile d'utiliser des informations sur la résolution ou l'exactitude des données générées.

En général, l'emplacement du capteur est considéré comme une information pertinente pour l'utilisation future des données d'observation générées par le capteur. Cependant, il est possible que l'emplacement change au fil du temps, par exemple lorsque le capteur est attaché à une voiture. Alors, les positions peuvent être considérées comme une mesure supplémentaire et doivent être observées. Au-delà des instants de temps précisant le moment où l'observation a été faite, le temps est très important pour l'analyse des données, il est le plus souvent utilisé comme une dimension d'évaluation ou de visualisation des données.

En outre, les valeurs les plus mesurées, notamment physiques, sont inutiles sans les unités correspondantes.

Ces principaux attributs doivent être pris en compte lorsque l'on travaille avec des mesures et des observations capteurs. Dans certains cas, l'influence de la précision est aussi très importante, en

particulier quand il ya des évaluations scientifiques. Ici, la description d'une éventuelle modification de précision dans certaines circonstances, est importante.

Selon une description donnée par le W3C , les concepts clés peuvent être regroupés dans cinq classes décrivant : les capteurs et les systèmes, les capacités d'un capteur, les processus, les aspects physiques des capteurs, et les concepts pour les observations.

La figure 2.2, montre les cinq classes regroupant les différents concepts d'une ontologie de capteurs :

Capteur (Sensor). Ce sont les concepts pour décrire les capteurs, les systèmes, les composants et les dispositifs (device).

Observation. Ce concept peut être étendu en se basant sur le standard (O & M) pour les observations et les mesures.

Capacités (Capabilities). Ce sont les concepts pour décrire ce que le capteur mesure ; le mesure-t-il correctement ? Comment peut-il le mesurer ? les types de valeurs des mesures (field of view) et sa politique d'acquisition.

Processus (Process). Ce sont les concepts pour les entrées et sorties, la politique de transmission, les fonctions et procédures. Les concepts pour décrire comment une mesure est faite et pour décrire comment les capteurs et autres systèmes sont combinés ensemble.

Propriétés physique (Physical Properties). Ce sont les concepts pour décrire : où se trouve le capteur, où il mesure, son niveau d'énergie, à quoi il est attaché, ses dimensions physiques et son poids, ses conditions opérationnelles et son fabriquant.

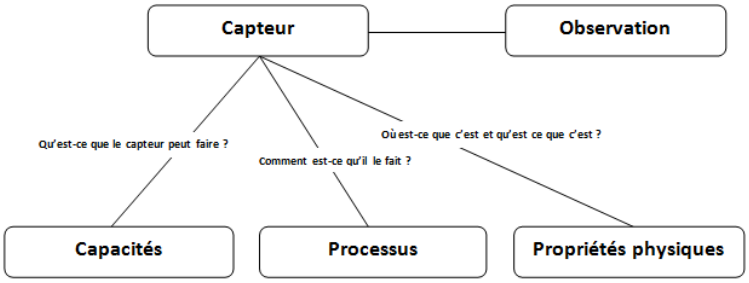

FIGURE 2.2 – Concepts de base des capteurs

Travaux sur les ontologies de capteurs

Depuis environ l'année 2004, un certain nombre d'ontologies pour la spécification des capteurs ont été élaborées et publiées, focalisées sur les descriptions des capteurs (Ontologie centrée-capteur) et/ou les observations (Ontologie centrée-observation), nous présentons dans cette section, quelques-unes de ces ontologies.

L'ontologie d'Avancha et al. [31] est une ontologie destinée aux réseaux de capteurs adaptatifs, où les nœuds réagissent en fonction de l'énergie disponible et des facteurs environnementaux pour pouvoir déterminer les états de fonctionnement adaptés. L'ontologie décrit les capteurs et les clusters de capteurs, l'organisation et la communication entre les différents types de nœuds capteurs d'un

WSN. Elle est écrite en OWL-Lite et composée principalement de quatre modules, qui sont : le module processeur (*Processor*) ; le module alimentation (*Power Supply*) ; le module radio (*Radio*) ; le module capteur (*Sensor*).

L'ontologie OntoSensor. [32], [33] est une ontologie OWL qui a été conçue pour la conception d'une base de connaissance globales de capteurs. Elle est utilisée pour l'interrogation, l'intégration et l'inférence. L'ontologie OntoSensor est fondée sur le standard SensorML, étend l'ontologie de haut niveau SUMO (*Suggested Upper Merged Ontology*) de la norme IEEE, et comprend les concepts de la norme ISO 19115. OntoSensor permet de décrire un large éventail de concepts, elle inclut principalement : les classes auxquelles appartiennent les objets (par exemple, les types de capteurs) ; l'hiérarchie de classes ou la structure taxonomique (par exemple, un ensemble de capteurs radiants est un sous-ensemble de tous les capteurs) ; la base relationnelle établie entre les classes (par exemple, un élément de détection fait partie d'un capteur) ; la base fonctionnelle des objets (par exemple, la largeur la bande passante) ; et la capacité pour exécuter des programmes ou des procédures spéciaux pour examiner la véracité des littéraux ou des valeurs d'attributs (par exemple, l'attachement procédural).

L'ontologie SDO. [34], [35] Dans leurs travaux les auteurs proposent un cadre à deux niveaux pour une ontologie de capteurs et de données capteurs (mesures retournées par les capteurs suite à des observations). Dans leur hiérarchie, on trouve au niveau supérieur l'ontologie SUMO, qui est utilisée pour la définition des concepts et des relations généraux, au deuxième niveau on trouve deux sous-ontologies étendant l'ontologie SUMO : la sous-ontologie des données capteurs et la sous-ontologie d'hiérarchie de capteurs. L'ontologie globale est ainsi composée de quatre sous-ontologies : l'ontologie SUMO ; l'ontologie d'hiérarchique de capteurs (*Sensor Hierarchy Ontology* — SHO) ; l'ontologie de données capteurs (*Sensor Data Ontology* — SDO) ; et l'ontologie de Plug-in d'extention (*Extension Plug-in Ontologies* — EPO). Développée en OWL, elle est destinée à être utilisée dans la recherche de données distribuées et hétérogènes au niveau des réseaux de capteurs. L'ontologie proposée a été évaluée en utilisant la langue RDQL (RDF Data Query Language).

L'ontologie OOSTethys. [36] La communauté OOSTethys développent des ressources open-source destinées à être utilisées dans l'installation, l'intégration et la mise à jour des services Web conformes aux standards pour l'observation océanographique, avec un accent particulier sur les standards OGC. L'ontologie de capteurs se concentre sur la structure et le fonctionnement du système et le résultat des observations. Elle offre ainsi un modèle conceptuel pour les systèmes d'observation. La description est écrite en OWL, basée sur les deux ensembles de standards, OGC SWE et IEEE 1451. Les principaux concepts décrits au niveau de l'ontologie sont : Une observation (*Observation*) telle que définie dans l'O & M ; Une caractéristique (*feature*) ; et Un capteur (*Sensor*).

L'ontologie CSIRO. [37], [38] est une ontologie générique pour décrire les capteurs et leurs déploiements. Elle est destinée à être utilisé pour faciliter l'intégration, la recherche et la classification des capteurs et des données capteurs. C'est une ontologie OWL pour décrire des raisonnements sur les capteurs et les observations. Les capteurs sont vus comme des unités de traitement qui peuvent être intégrées pour former des capteurs virtuels. Deux exemples de définitions de capteurs sont

disponibles pour l'ontologie CSIRO montrant une interrogation (SPARQL-DL) utilisée pour rechercher et construire des instruments virtuels. L'ontologie est organisée autour de quatre clusters principaux de concepts décrivant : le domaine de détection (Feature) ; le capteur (Sensor) ; les composantes physiques et l'emplacement du capteur (SensorGrounding) ; les fonctions et les traitements (OperationModel and Process).

L'ontologie CESN. [39] Dans le cadre du projet Coastal Environmental Sensing Networks (CESN) et dans le but de développement d'un réseau de capteurs dédiés aux observations côtières, les auteurs proposent une ontologie de types de capteurs et des règles logiques (DL) de raisonnent spécifiques au domaine pour créer des inférences au niveau des données et des anomalies mesurées. L'ontologie est destinée pour la validation de données sémantique capables d'observer en temps réel les données issues des capteurs et la détection d'événement. Elle est écrite en OWL et permet de décrire les capteurs, leurs déploiements, les événements qui peuvent se produire ainsi que les relations entre les capteurs et leurs mesures. Les principaux concepts de cette ontologie sont : les dispositifs physiques (Sensor) ; les PhysicalProperty, ce que le capteur permet de mesurer ; les PhysicalPropertyMeasurement, ce que le capteur mesure ; la classe Instrument, décrit les objets qui peuvent contenir des capteurs ; la classe Deployment représente le déploiement d'un instrument à un moment donné et en un lieu particulier.

L'ontologie SOSA/SSN. Toutes les ontologies précédemment mentionnées ont été examinées par le groupe W3C *Semantic Sensor Network Incubator Group*, qui à son tour a développé l'ontologie SSNX publiée en 2011 [40]. Cette ontologie a été largement réutilisée dans d'autres ontologies et implémentée dans plusieurs projets. Raison pour laquelle l'OGC et le W3C ont revisité l'ontologie SSNX afin de prendre en compte les lacunes du travail initial, de considérer la représentation des actionneurs et des échantillonnages en plus des observations. La dernière version [41] étant publiée comme standard depuis octobre 2017. Le résultat de ce travail a été publié sous la forme d'une ontologie modulaire nommée SOSA / SSN, spécifiant la sémantique des capteurs, des observations, des actionneurs et de l'échantillonnage. La figure 2.3 présente les principaux composants et termes de l'ontologie SOSA / SSN. Les principaux modules de l'ontologie SSN pertinents pour notre travail sont les suivants :

— SOSA (capteur, observation, échantillonneur et actionneur) est un module de base léger avec peu de termes et peu d'axiomatisation ;
— SSN (réseau de capteurs sémantiques) importe SOSA, introduit d'autres termes et ajoute davantage d'axiomatisation aux termes SOSA et SSN ;
— SSN-System (module SSN System Capability) importe le SSN (et indirectement SOSA) et définit des classes et propriétés supplémentaires utilisées pour modéliser les capacités du système, la plage de fonctionnement et la plage de survie.

Synthèse

Une description d'un certain nombre d'ontologies de capteurs a été présentée. La plupart des ontologies décrites ci-dessus sont souvent spécifiques à un projet, ou abandonnées, et ne couvrent

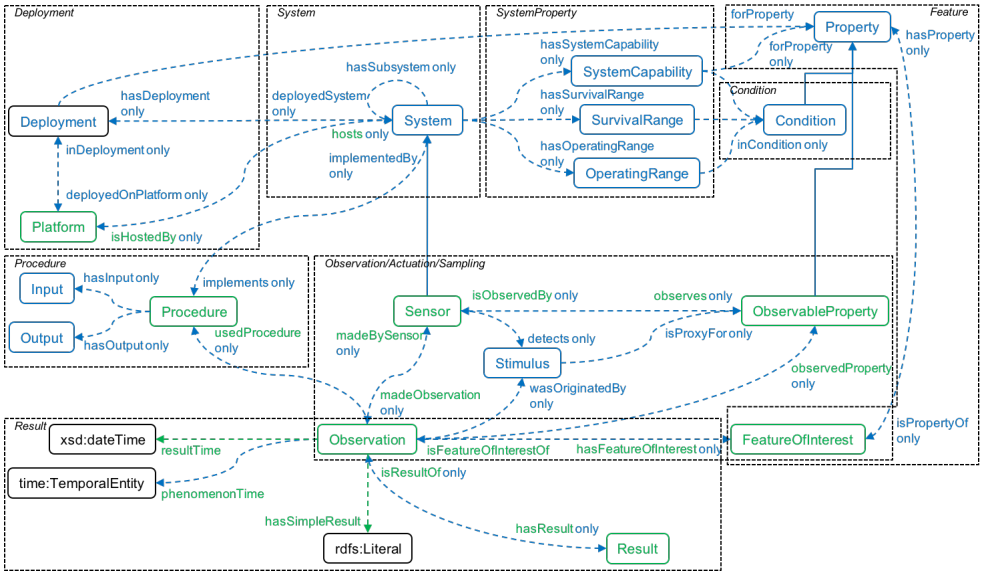

FIGURE 2.3 – Vue d'ensemble des classes et des propriétés de SOSA / SSN (perspective "Observation")

qu'un nombre limité de concepts dans le domaine des capteurs et des observations. En outre, plusieurs de ces ontologies n'ont pas suivi un processus de modélisation solide ou n'ont pas réutilisé les normes existantes. De plus, il est important de noter que toutes ces ontologies ont été développées pour décrire des capteurs classiques et non pas pour décrire des capteurs intelligents (voir la différence entre un capteur classique et un capteur intelligent au niveau de la section 4.3.1, tableau 4.3). Ces derniers offrent aux OI la possibilité d'être adaptables au contexte d'utilisation.

Néanmoins, l'ontologie SOSA/SSN peut être consultée et utilisée pour capturer diverses propriétés d'entités du monde réel. Elle peut être centrée sur les données observées et les capteurs ainsi que sur les métadonnées associées. De plus, cette ontologie est à présent un standard fortement recommandé par le W3C. Cette ontologie sera réutilisée pour la création de l'ontologie SMS. Le tableau 2.1 présente un résumé sur l'état de l'art des ontologies de capteurs en fonction de leur utilisation et sur leurs capacités de décrire les concepts indiqués dans la figure 2.2 pour la modélisation des réseaux de capteurs. Les trois signes (+++) signifie que la totalité de ces concepts sont pris en compte dans l'ontologie, Les deux signes (++) montre qu'un certain nombre de ces concepts sont représentés, la présence d'un seul signe (+) signifie que très peu de ces concepts sont représentés par l'ontologie.

Ontologies	**Utilisation de l'ontologie**	**Périmètre**	**Concepts décrits pour les capteurs et les propriétés physiques**	**Concepts décrits pour les observations, les capacités et les processus**	**Concepts décrits pour les capteurs intelligents**
CSIRO	Description, classification et intégration des capteurs et des données ; Composition des capteurs	Générique, centrée-capteur et centrée-observation	+++	++	Non
OntoSensor	Interrogation, intégration et inférence	centrée-capteur	++	++	Non
OOSTethys	Description des systèmes d'observation et des résultats ; Intégration des services Web (OGC SWE)	centrée-observation	+	+	Non
SDO	Extension du SUMO pour la représentation des données distribuées et hétérogènes	centrée-observation et données	+	++	Non
SOSA/SSN	Description des capteurs et des observations (actionneur et échantillon) ; raisonnements sur les capteurs et les observations	Générique, standard, centrée-capteur et centrée-observation	+++	++	Non
CESN	Description des données du domaine et des événements	centrée-observation	+	+	Non
D'Avancha et al.	Description des réseaux de capteurs adaptatifs	centrée-capteur	++	++	Non

TABLEAU 2.1 – Revue de la littérature sur les caractéristiques des ontologies de capteurs

2.4 Frameworks et méthodologies pour les applications IdO

En poursuivant notre investigation théorique, nous attaquons dans cette section une revue de la littérature concernant les Frameworks et les méthodologies pour l'IdO. Cette investigation nous permettra de cadrer nos questions de recherches relatives ; (1) au Framework sémantique générique pour la conception/fabrication d'objets intelligents ainsi que la conception et le développement d'écosystèmes intégrant ces objets, et (2) à un support méthodologique pour ces fins.

2.4.1 Les Frameworks pour l'IdO

Avec l'émergence de nouvelles applications de l'IdO, plusieurs Frameworks ont pu être développés pour aider au développement de ces applications. Ces Frameworks sont destinés à traiter ou à mettre en application un éventail de dispositifs allant des plus petits micro-contrôleurs, à des appareils plus complexe dotés de processeurs multi-cœurs.

Nous décrivons ici un ensemble de Frameworks pour les applications IdO. Nous mettons principalement l'accent sur leurs fonctionnalités principales pour les systèmes et solutions IdO (conception, développement, intégration ou déploiement), sur leur capacité à garder la trace de l'expertise de conception/fabrication d'un OI et sur les deux aspects clés que nous visons pour un Framework IdO qui sont la sémantique et la généricité.

Le Framework Eclipse Smarthome

Le framework Eclipse Smart Home (ESH) [42] est conçu pour faciliter la résolution des problèmes liés au développement de systèmes IdO rencontrés par les développeurs bénéficiant de ses interfaces, de ses règles d'automatisation, de son mécanisme de persistance et de son implémentation SOA. L'ESH est un framework de connexion et d'intégration pour le domaine "*IoT Smart Home*". Il est indépendant des fonctionnalités de connectivité du matériel, mais met plutôt l'accent sur l'implémentation d'un connecteur dans la structure. L'ESH est devenu largement adopté car il est open source et donc largement implémenté comme solution des maisons intelligentes par un grand marché. Cela a contribué à son grand nombre d'API partagées disponibles pour une gamme de produits commerciaux [42].

L'ESH est explicitement fixé sur la domotique et repose sur les cinq couches suivantes [43] :

- Un système d'exploitation. L'ESH fonctionne de manière équivalente sous Linux, macOS et Windows ;
- Un conteneur d'applications. Ce dernier est entièrement écrit en Java et utilise les Runtimes d'OSGi (Eclipse Equinox) avec Apache Karaf et l'intègre au serveur Jetty HTTP ;
- Communication et connectivité. La vaste acceptation d'ESH a permis de voir sa vaste implémentation fournir la connectivité et la communication entre de nombreux produits domotiques prêts à l'emploi. Belkin WeMo, Philips Hue, Sonos, etc. en sont des exemples ;

- Gestion des données et messagerie. L'approche SOA du Framework ESH propose une implémentation d'un "bus événementiel" interne. Ce bus est accédé et exposé en externe par une liaison de protocole implémenté, par exemple SSE ou MQTT ;
- Gestion à distance. le Framework ESH est conçu pour permettre la surveillance à distance, la mise à jour de micro-programmes et de la configuration des périphériques connectés.

L'ESH fonctionne principalement comme un logiciel incorporé dans le matériel afin de fournir un point de coordination collectif pour la connectivité des objets entre eux et avec un réseau externe. Le Framework ESH a été incorporé et implémenté par openHAB pour fournir des logiciels open source facilitant le développement d'applications IdO.

Le Framework Calvin

Calvin [44] est un Framework hybride des modèles de programmation IdO et Cloud pour la prise en compte de la complexité de l'informatique distribuée, des divers langages de programmation et des protocoles de communication. Ce Framework est développé en consolidant les théories du "*Actor model*" et de "*Flow Based Computing*". Pour assurer une certaine flexibilité, Calvin divise le développement d'applications IdO en quatre activités distinctes et exécutées séquentiellement, à savoir :

- *Describe*. Décrit les parties fonctionnelles des applications, transformées en composants réutilisables. Calvin caractérise un acteur (*Actor*) comme étant constitué d'actions, de ports de communication et de conditions susceptibles de déclencher une action. Ces acteurs communiquent entre eux via les ports de communication par échange de jetons ;
- *Connect*. Sert à connecter les ports des acteurs déjà décrits en utilisant un langage intuitif et déclaratif léger appelé CalvinScript. Les paramètres nécessaires aux acteurs sont inclus et leur modèle de connexion est initialisé dans cet aspect. Les interactions entre les composants sont décrites sous forme de graphiques ;
- *Deploy*. Cette activité se concentre sur le déploiement du composant décrit et connecté dans un environnement d'exécution distribué. Les acteurs décrits et connectés sont légers, ce qui permet une migration entre un réseau maillé d'exécutions ;
- *Manage*. Gére le mappage autonome et dynamique des composants sur le matériel au cours de la durée de vie de l'application. Cette activité implique la gestion des acteurs de migration entre les exécutions, la récupération des erreurs, la mise à l'échelle, l'utilisation des ressources et les mises à jour par l'environnement d'exécution distribué.

Le Framework SOCRADES

SOCRADES [45] est une architecture d'intégration basée sur les services. Elle fournit des composants génériques pour faciliter la modélisation de processus. Elle cible les objets intelligents dans le domaine de la fabrication en représentant leur comportement en tant que services Web. SOCRADES

intègre le modèle, le concept et le code de SIRENA [46] (projet financé par l'Union européenne) pour proposer et concevoir une infrastructure d'intégration pour les services Web et un Framework pour la supervision des appareils et le cycle de vie de SIRENA.

Le Framework FRASAD

FRASAD [47] est un Framework de développement destiné à permettre aux développeurs de concevoir leurs applications IdO en utilisant des concepts de domaine de nœuds de capteurs. Ce concept est piloté par un modèle et le code d'application est généré à partir du modèle conçu via un processus de transformation. Le Framework FRASAD (*FRAmework for Sensor Application Development*) est une extension de [48] par l'ajout et l'intégration de deux couches à l'architecture existante du nœud de capteur. Ces deux couches supplémentaires sont la couche d'application (APL) et la couche d'abstraction du système d'exploitation (OAL). L'essence de ces deux couches est d'amplifier le niveau d'abstraction et de dissimuler ainsi les niveaux inférieurs. Pour ce faire, la structure utilise un langage DSL (*Domain Specific Language*) de conception pour modéliser les nœuds de capteurs et séparer le système d'exploitation de l'application. L'OAL est ensuite engagé pour interpreter l'application modélisée en fonction du système d'exploitation spécifique à mettre en œuvre. La couche OAL peut être considérée comme un générateur d'applications : sa fonction principale est de générer le code d'application à déployer sur la plate-forme ciblée. Le Framework FRASAD suit de manière inhérente l'approche MDA (*Model Driven Architecture*) en adoptant les trois modèles au niveaux d'abstraction du MDA à savoir :

- Le *Computation Independent Model* (CIM) ;
- Le *Platform Independent Model* (PIM) ;
- Le *Platform Specific Model* (PSM).

Ainsi, FRASAD est un Framework qui utilise une architecture MDA multicouches avec interaction entre les couches via une interface prédéfinie. Il utilise le modèle centré sur les nœuds pour faciliter la programmation de nœuds de capteurs individuels à l'aide de son modèle de programmation basé sur des règles.

Le Framework AVIoT

AVIoT [49] est un Framework IdO permettant de visualiser et de gérer les objets IdO présents dans un environnement spécifique (par exemple une maison intelligente). Ce Framework vise à permettre aux utilisateurs d'appliquer leurs outils de création visuels basés sur le Web pour abstraire et programmer le comportement des objets IdO. Ainsi, les utilisateurs finaux peuvent facilement surveiller et définir le comportement des objets IdO sans connaître préalablement l'architecture ou le système de connexion des capteurs. Le Framework AVIoT est proposé pour permettre la configuration visuelle et la gestion des objets dans un environnement IdO. Ce Framework suit un processus fondé sur les principes suivants :

- *Abstraction of Sensors and Actuators*. Dans ce processus, les objets physiques présents dans un environnement IdO sont virtualisés pour interagir avec d'autres objets virtuels en les définissant comme des nœuds. Chaque nœud sera défini en fonction des caractéristiques génériques telles que le nom, le type de l'objet, le type de visualisation, la position, les nœuds enfants (le cas échéant) et le comportement des chaînes fonctionnelles (contient un code de script pouvant être évalué à l'exécution). Les objets sont virtualisés pour contenir des capteurs physiques, des capteurs virtuels, des actionneurs virtuels et des actionneurs physiques. Alors que les capteurs et les actionneurs physiques sont les objets du monde réel, la fonction centrale des capteurs virtuels consiste à détecter les événements importants pour lesquels l'utilisateur final a exprimé son intérêt. La fonction principale des actionneurs virtuels est d'adapter le comportement prédéfini aux actionneurs physiques. Les actionneurs virtuels ne contiennent pas de valeurs comme les capteurs virtuels, ils déclenchent plutôt les actions des actionneurs physiques. L'interaction entre les composants abstraits suit l'ordre "*physical sensor ->virtual sensor ->virtual actuator ->physical actuator*" ;
- *Interactive IoT Visualisation*. C'est la visualisation d'objets virtuels dans l'environnement IdO pour promouvoir la gestion des objets physiques. Cette phase utilise un navigateur Web ou une application Web client et un serveur pour faciliter la visualisation et la gestion des objets virtuels. L'hypothèse est que le serveur identifie les périphériques et détient des informations sur leurs différents modules de connectivité. Le serveur est également utilisé comme espace de stockage pour les informations relatives à la visualisation et à la création d'objets. Cette phase exploite REST comme interface de communication entre le client et le serveur. Pour la visualisation 3D, le serveur conserve un plan virtuel d'un environnement intérieur dans son état de modèle 3D. Cela peut être demandé par un client en cours d'exécution pour fournir une visualisation 3D des objets à l'exécution. Les images 3D sont rendues comme des images de fil et des icônes visuelles sont affichées en fonction de l'emplacement des objets physiques pour identifier les capteurs et les actionneurs ;
- *Interactive IoT Authoring*. Dans cette phase, les objets virtuels peuvent être édités, ajoutés ou supprimés de l'ensemble d'éléments physiques. Les objets peuvent également être déplacés par l'utilisateur final dans une position différente et redécouvertes dans le nouvel emplacement par le serveur. Le capteur physique interagit avec le monde réel et produit des données numériques observées qui sont reçues par le capteur virtuel. Les capteurs virtuels se concentrent sur les événements (préconfigurés) qui concernent l'utilisateur final. Cette information est traduite et transmise à l'actionneur virtuel qui agit alors en déclenchant l'action appropriée des actionneurs physiques.

L'AVIoT est un Framework IdO pour la visualisation et la création d'objets dans un environnement intelligent par les utilisateurs finaux. Il est orienté Web et entièrement dédié aux environnements IdO intérieurs. Sa structure permet aux utilisateurs finaux d'abstraire et de définir les objets par ordre hiérarchique en utilisant ses outils d'application Web pour la visualisation et la création. Ce Framework

représente des aspects très intéressants relativement à notre problématique sauf que l'expertise et les connaissances de ses utilisateurs restent tacites.

Synthèse

Chacun des Frameworks IdO présentés a été développé pour différents objectifs tels que la conception, le développement, le déploiement, l'intégration, la visualisation ou encore la création d'applications IdO. Ils ont utilisé des approches de modélisation ou de méta-modélisation comme MDA. Les Frameworks examinés présentent des inconvénients et des avantages qui sont attribués à l'architecture sélectionnée pour fournir une solution IdO ou au modèle d'implémentation. Par exemple, l'ESH a ses points forts en fournissant un support pour des protocoles de communication hétérogènes tout en fournissant des moyens de créer des "*bindings*" pour de nouveaux protocoles. Il prend en charge un large éventail de produits hors-ligne, il est facile à développer et à déployer. Cependant, l'ESH se limite aux maisons intelligentes. Le Framework Calvin présente certaines limites car les développeurs doivent apprendre un nouveau langage peu précis. De plus, il ne permet pas la combinaison de services ni la prise en charge de la gestion des services. Le framework SOCRADES prend en charge les systèmes hétérogènes, son composant "*Enterprise Application*" offre aux utilisateurs une interface graphique facile à utiliser, réduisant ainsi le temps de déploiement et facilitant la gestion du système. Néanmoins, une défaillance ponctuelle est présente dans son utilisation d'un "*Enterprise service bus*", il n'existe aucune connaissance intrinsèque des services.

Cependant, malgré la flexibilité offerte par ces systèmes, les problèmes liés à la création collaborative multi-domaine d'objets IdO persistent. L'une des principales causes est l'absence d'un Framework intelligent pour prendre en charge l'interaction entre les différents acteurs dans les systèmes IdO. Ces différents frameworks sont utilisés principalement pour créer des applications où la définition d'une application IdO est identifiée par l'ajout de devices et de leurs protocoles de communication. Ces Frameworks n'offrent pas un environnement pour la conception d'un objet IdO faisant appel à différents domaines. De plus, les processus de création des applications IdO sont des processus *ad hoc*, aucun consensus ou méthodologie n'est proposée pour ces Frameworks. L'expertise reste dans la tête des développeurs. Ces Frameworks se concentrent sur la facilité de développement et de déploiement d'applications, avec des interactions riches avec Internet et avec des systèmes distants chargés d'élaborer les données collectées. Ils omettent toute la phase de conceptualisation consensuelle sur les aspects les plus importants de l'application et de ses différents usages.

L'objectif de cette étude était principalement la recherche de Frameworks pour aider les différents experts du projet SmartSensing à concevoir le vêtement intelligent (VI) et plus particulièrement les capteurs à utiliser dans le cadre du projet à savoir les capteurs intelligents qui sont généralement fabriqués-maison. Il s'agit plus particulièrement de les aider à trouver, se baser et se mettre d'accord sur les concepts et leur signification. Les Frameworks étudiés ne se base pas sur des représentations sémantiques et ne permettent pas de garder trace de toute l'expertise et des connaissances relatives à la conception/fabrication et exploitation d'objets intelligents. De plus les artefactes issus de ces

Frameworks ne sont pas adaptables. Le tableau 2.2 synthétise l'étude de ces Frameworks selon les critères visés au debut de cette investigation.

Framework	**Objectifs et fonctionnalités**	**Généricité**	**Utilisation de la sémantique**	**Expertise sauvegardée et partagée**
ESH	Développement + Intégration	Non	Non	Non
Calvin	Développement + Déploiement	Oui	Non	Non
SOCRADES	Intégration	Oui	Non	Non
FRASAD	Conception	Oui	Non	Non
AVIoT	Création + visualisation	Oui	Non	Non

TABLEAU 2.2 – Frameworks pour l'IdO

2.4.2 Les méthodologies pour l'IdO

Le processus de conception/fabrication d'un OI ou encore d'un système IdO est un processus complexe, ceci est dû principalement à la multidisciplinarité des acteurs intervenant tout au long du cycle de vie de l'objet intelligent ou de la solution IdO. Notons en plus la prise en compte des caractéristiques structurelles et comportementales des objets et/ou des systèmes intelligents et connectés et de leurs différents cas d'usage. Au fur et à mesure que la recherche dans le domaine de l'Internet des objets progresse cette complexité augmente et le besoin d'une méthodologie pour le développement d'OI, de systèmes et d'applications IdO se fait ressentir. Dans le cadre de cette thèse, nous visons une méthodologie générique qui ne soit pas spécifique à un domaine applicatif particulier, offrant un ensemble d'étapes (ou de directives) claires, couvrant les différents éléments d'abstractions constituant un OI ou un système intelligent connecté. De plus, vu l'intérêt et les avantages que peuvent apporter les technologies du Web sémantique, et particulièrement les ontologies, dans l'IdO une méthodologie fondée sur la sémantique et sur des standards et composants réutilisables et indépendants du domaine offrira une plus grande flexibilité.

Malgré l'importance d'une telle méthodologie peu de travaux se sont intéressés à ce type de problématique. Néanmoins, durant ces dernières années quelques travaux de recherche ont explicitement abordé le problème de la définition de nouvelles approches de génie logiciel spécifiquement conçues pour l'IdO. Certaines propositions de Frameworks de développement pour l'IdO ou pour le WoT (qu'il s'agisse d'une architecture middleware [50] ou de modèles de programmation [51], [52]) sont également accompagnées de lignes directrices pour le développement d'applications.

Dans [53], les auteurs proposent une approche "*design-thinking*" basée sur un processus pour la découverte de cas d'utilisation pour des scénarios IdO. Un *Privacy-by-Design* Framework d'évaluation des applications et des plates-formes de l'Internet des objets a été proposé dans [54]. Les auteurs se

concentrent sur l'adoption des meilleures pratiques en matière de conception de la confidentialité dans le domaine IdO. Le Framework a été proposé pour la protection de la vie privée par la conception pour évaluer les applications et les plates-formes de l'Internet des objets.

Des considérations similaires s'appliquent au domaine des villes intelligentes et de l'informatique urbaine [55], où des intergiciels et des approches de programmation sont proposés - la plupart du temps à des fins spécifiques et axés sur des scénarios d'application spécifiques tels que la détection participative [56], [57] ou gestion de la mobilité [58]. Cependant, ces travaux se concentrent sur des architectures concrètes. Les directives proposées ne sont pas fondées sur des abstractions génériques et ne peuvent pas être appliquées de manière générale en dehors du Framework spécifique dans lequel elles sont conçues. Toutefois, nous retenons deux travaux assez proches de notre vision de la méthodologie pour l'IdO :

La première est une méthodologie de génie logiciel générale pour l'IdO décrite dans [59]. C'est une méthodologie d'ingénierie logicielle centrée sur un ensemble de concepts clés et d'abstractions intervenant dans la conception et dans le développement de systèmes et d'applications IdO, qui sont ; les objets, les parties prenantes et les utilisateurs, les infrastructures logicielles, les avatars et les coalitions. La méthodologie est décrite par un ensemble de directives facilitant l'ingénierie de développement. Ces directives de bonnes pratiques sont regroupées en 3 principales phases pour le développement de systèmes ou d'applications IdO. Ces phases sont décrites par un ensemble d'activités :

— La phase d'analyse : Cette phase permet d'analyser et d'identifier les utilisateurs du système, d'analyser des caractéristiques de l'infrastructure matérielle IdO et l'identification d'objectifs, de stratégies, de fonctions et la définition d'objectifs et de stratégies locales ou globales. Cette phase comprend l'activité d'élicitation d'exigences qui peut impliquer principalement des gestionnaires globaux qui ont le droit de décider quels services fournir aux utilisateurs finaux et quels objectifs et politiques locaux peuvent être définis par les gestionnaires locaux.

— La phase de conception : Les activités de cette phase comprennent la conception d'avatars, de groupes et de coalitions. Ceci doit être défini dans le but de produire une conception globale capable de réaliser les objectifs, les politiques et les fonctions nécessaires, en accord avec les exigences. La conception des coalitions n'implique pas la conception de nouveaux avatars, mais principalement la conception du système de coordination qui définira et gouvernera les activités de la coalition. La phase de conception permet d'identifier des nouveaux besoins en infrastructures et de comprendre quels nouveaux périphériques ou services middleware doivent être intégrés dans l'infrastructure afin de réaliser les fonctionnalités requises en fonction des besoins.

— La phase d'implémentation : Cette phase envisage l'implémentation d'avatars et de coordinateurs. Cela dépend clairement de l'infrastructure du middleware et du modèle de programmation adoptés. Cette phase envisage également l'activité de déploiement de nouveaux objets et de nouvelles caractéristiques du middleware. Cela se fera en fonction des nouveaux besoins en infrastructure identifiés lors de la phase de conception.

Cette méthodologie est générale et indépendante de toute solution IdO. Bien qu'elle propose un ensemble de directives nécessaires pour la conception et le développement d'un système IdO, elle se concentre principalement sur la définition des utilisateurs et de leurs rôles. La méthodologie n'utilise pas de normes ou de modèles pour la conceptualisation des outputs de chaque phase. En d'autres termes elle n'offre pas la possibilité de la définition de modèles et d'outils permettant de représenter les différents artefacts conceptuels et logiciels produits au cours des différentes étapes (par exemple, comment modéliser une exigence IdO ? Le langage formel peut-il représenter les caractéristiques d'un avatar ou d'un groupe ? Comment transformer une telle représentation abstraite en une implémentation ?).

La deuxième méthodologie que nous retenons est l'"*IoT Solution Methodology*" [4]. Cette méthodologie est une continuation de la méthodologie Ignite qui est étroitement liée aux travaux proposés dans [60], par les contributeurs du projet "Eclipse Ignite|IoT" [5]. La méthodologie Ignite trouve ses fondements dans les méthodologies du génie logiciel. Elle est destinée aux projets multidisciplinaires tels que les projets IdO. Selon [60] les projets IdO prolongent les projets traditionnels, car ils associent quatre domaines différents : un projet de conception et de fabrication de produits pour la partie matérielle, un projet informatique embarqué pour le micro-logiciel, un projet de télécommunications et un projet informatique d'entreprise traditionnel pour le back-end et la partie d'intégration. Pour relier ces domaines et dégager une compréhension commune de toutes les parties prenantes, les auteurs proposent une méthodologie d'analyse de solutions IdO basée sur les processus métiers en tant qu'artefacts clés. L'objectif étant de fournir une méthodologie de haut niveau reliant les différentes disciplines IdO requises, puis de collaborer avec des experts de différentes disciplines pour saisir leurs expériences et leurs meilleures pratiques et les intégrer à la méthodologie.

Dans l'"*IoT Solution Methodology*", un projet IdO est structuré selon un modèle de base, nommé "*IoT Solution Draft*" qui est utilisé dans les ateliers initiaux du projet IdO et est créé avec toutes les parties prenantes requises. Ce modèle fournit une vue intégrée de tous les artefacts qui peuvent être décomposés par la suite en différents sous-projets. Le projet porte sur les principales parties prenantes, le modèle de domaine de haut niveau, les processus et règles métier, les interfaces utilisateurs et les modèles, ainsi que les objets et la connectivité.

Par ailleurs, les éléments du "*IoT Solution Draft*" sont décrits dans un méta-modèle de classes UML où la classe de haut niveau "*Solution Draft*", regroupe les différents éléments du modèle, qui sont les règles, les interfaces utilisateurs, les rôles, les processus, le modèle de domaine, le modèle d'analyse, les objets ainsi que le concept de connectivité correspondant. En outre, les associations les plus importantes définissent les liens entre les différents éléments.

Contrairement à la méthodologie Ignite qui se concentre sur l'organisation du projet, les directives de la méthodologie "*IoT Solution Methodology*" ne sont pas explicitées, les composants du modèle "*IoT Solution*" sont juste expliqués par un exemple. Cette méthodologie est pratique pour analyser les exigences initiales de toutes les parties prenantes impliquées. Elle suppose que le cas d'utilisation ou

4. http ://frapu.de/iot/
5. https ://projects.eclipse.org/proposals/igniteiot

le domaine d'intervention a déjà été identifié. Elle s'arrête avant une décision d'architecture concrète. La méthodologie elle-même repose sur des concepts classiques, tels que l'analyse de cas d'utilisation et les diagrammes de classes UML, qui mettent fortement l'accent sur les processus métier.

A notre connaissance, et dans le cadre de l'investigation faite dans le contexte des Frameworks et des méthodologies pour la fabrication/conception et l'exploitation d'OI et de systèmes intelligent connectés, il n'existe pas de Frameworks sémantiques générique pour l'IdO. Un Framework fondé sur le Web sémantique (particulièrement les ontologies) et offrant une méthodologie générique collaborative à composants génériques réutilisables.

2.5 Les plateformes IdO pour la re-programmation des capteurs

Les capteurs intégrés dans les objets de l'IdO forment un réseau de capteurs généralement sans-fil et distribué (*Wireless Sensor Networks* : WSN). Un réseau de capteurs est composé d'un grand nombre de nœuds capteurs [61]. Les capteurs sont ainsi, déployés de manière dense à l'intérieur d'un environnement (pour mesurer un phénomène) ou tout près de celui-ci. Ces capteurs sont, géographiquement, distribués et leurs positions ne sont pas préalablement conçues ou prédéterminées. Cela permet leur déploiement aléatoire sur des terrains inaccessibles ou pour des opérations de secours. D'autre part, les protocoles des réseaux de capteurs doivent également, posséder des capacités d'auto-organisation. Au lieu d'envoyer des données captées brutes vers des nœuds intermédiaires dédiés aux opérations de fusion et de traitement, les nœuds capteurs sont équipés de processeurs embarqués. Ceux-ci dotent les nœuds de capacités de traitements locales permettant d'effectuer des opérations de calcul simples et de ne transmettre que les données requises et partiellement traitées.

Le développement de logiciels pour le nœud capteur est une tâche fastidieuse en raison de la modification de la topologie du réseau. L'état actuel de la technique a mis aux points différentes solutions pour surmonter les problèmes de performances tels que la surcharge de mémoire, l'hétérogénéité, la portabilité, l'évolutivité, le coût et la qualité de services. Une solution de plateforme adaptative devrait minimiser la consommation de ressources et fournir une solution optimale pour la re-programmation à l'exécution.

Dans cette section, nous discutons des paradigmes de re-programmation importants pour les capteurs. Une analyse de la littérature sur la classification des modèles de re-programmation des WSNs est réalisée [62] et [63]. L'accent est mis sur les modèles de re-programmation à composants qui sont en relation étroite avec les principales préoccupations de cette thèse.

2.5.1 Intérêt de la re-programmation

Les premières applications utilisant les WSN étaient limitées à une seule fonction appelée "**détecter et envoyer**" ("*sense and send*") avec des tâches de traitement de données locales triviales. Les nouvelles applications émergentes pour les WSN se dirigent progressivement vers des environnements

informatiques omniprésents, où les nœuds capteurs interagissent étroitement avec les actionneurs et se comportent en fonction des informations qui les entourent [64], [65].

D'un autre côté, les nœuds capteurs sont dotés d'un certain degré d'intelligence pour traiter les données, et ce *via* des composants logiciels déployés dans ces capteurs. Les composants logiciels peuvent être installés et réinstallés selon les exigences de l'application IdO (ou de l'utilisateur d'un objet IdO) et de façon dynamique. La re-programmation fournit une solution efficace aux besoins changeants des diverses applications du WSN. Il est nécessaire de reprogrammer les nœuds capteurs chaque fois qu'une nouvelle fonctionnalité doit être envoyée *via* des liaisons sans fil. Le mécanisme de diffusion du code propagerait le code au capteur souhaité. Les logiciels déployés sur les WSN doivent souvent être mis à jour après le déploiement pour diverses raisons, telles que la mise à niveau du logiciel du nœud, la correction de bogues logiciels, la modification de la fonctionnalité du réseau, le réglage des paramètres du module et la correction des failles de sécurité. Selon [63] :

— Un WSN déployé peut être confronté à des erreurs sporadiques qui n'étaient pas observables avant le déploiement [66] ;

— Les logiciels déployés sur les nœuds de capteurs deviennent de plus en plus volumineux à différents niveaux, des pilotes matériels de bas niveau aux systèmes d'exploitation, des services middleware et des modules d'application. À mesure que la taille et le nombre de modules logiciels augmentent, la tâche de maintenance devient également plus importante ;

— Les exigences des configurations et des protocoles réseau peuvent changer tout au long de la vie des applications en raison de l'hétérogénéité et de la nature distribuée de la plupart des applications WSN [67]. Par conséquent, en raison de contraintes de ressources, il est impossible de charger de manière proactive tous les services prenant en charge l'hétérogénéité dans des nœuds. Par conséquent, les variations des exigences sont essentiellement satisfaites par la mise à jour du logiciel du capteur ;

— Enfin, le nombre croissant de déploiements WSN dans des environnements omniprésents fait de la re-programmation et de l'auto-adaptation deux fonctions vitales : une application de détection détecte les modifications internes et externes du système, les analyse et s'adapte de manière transparente aux nouvelles conditions en mettant à jour les fonctionnalités du logiciel. Des techniques de re-programmation fiables et flexibles deviennent la partie centrale de tout framework d'auto-adaptation lorsque le composant de re-programmation subit le remplacement d'un module logiciel existant par un module mis à jour.

Dans [68] et [69], trois exigences fonctionnelles importantes ont été identifiées : le besoin de modifications dynamiques, le besoin de prise en charge de nœuds hétérogènes et la nécessité d'intégrer de nouvelles versions logicielles dans un système en cours d'exécution. La mise à jour logicielle à distance est un composant essentiel permettant d'améliorer la facilité d'utilisation en offrant une flexibilité dans un réseau déployé et en prenant directement en charge la fonction de maintenance.

2.5.2 Les approches de re-programmation de WSN

Les nœuds sans fil peuvent fonctionner dans un environnement hétérogène où les conditions et les protocoles environnementaux varient avec le lieu et le temps. Cela suggère la nécessité pour les nœuds de détecter l'environnement et de déployer de nouveau codes à la plate-forme si nécessaire. Dans ce contexte, nous distinguons deux types de déploiements ; i) la mise à jour qui consiste principalement à modifier le même code existant par une nouvelle version de celui-ci (maintenance logiciel des nœuds capteurs) ; ii) la re-configuration (ou encore l'adaptation) impliquant le chargement et le déchargement dynamique ou statique de codes logiciels appropriés ou le réglage des paramètres pour obtenir les performances souhaitées. La re-configuration est généralement liée à l'adaptation du nœud capteur par l'ajout de nouvelles fonctionnalités.

La mise à jour et la re-configuration/adaptation sont des tâches représentant un défi majeur aussi bien pour les programmeurs que pour les concepteurs des réseaux de capteurs sans-fils. Ces derniers sont des infrastructures à contraintes avec des ressources limitées pour les capteurs et la bande passante du réseau, rendent la mise à jour ainsi que la re-configuration de codes logiciels sur les nœuds capteurs difficiles à résoudre. Il sera ainsi important de savoir comment activer efficacement la mise à jour à distance et la re-configuration/adaptation du logiciel du capteur en ce qui concerne les restrictions du WSN et les exigences de diverses applications. Nous présentons dans cette section une revue de la littérature sur les approches de mise à jour et de re-configuration/adaptation des capteurs.

Les approches de mises à jour de codes

Selon des classification faite dans plusieurs travaux tels que [62] et [70] on distingue trois approches pour la mise à jour des nœuds capteurs : i) Remplacement complet de l'image ("*Full Image Replacement*") ; ii) Remplacement différentiel d'image ("*Difference Image Replacement*") et iii) Mises à jour incrémentielles du code ("*Incremental Code Updates*"). Nous décrivons ci-dessous brièvement les systèmes et les mécanismes les plus rencontrés dans la littérature.

Mise à jour par remplacement complet de l'image. Selon [71], la mise à jour complète d'images logicielles est le mécanisme le plus courant pour la mise à jour de composants logiciels dans les systèmes intégrés et les réseaux de capteurs. Ce mécanisme consiste à compiler une nouvelle image binaire complète du logiciel de l'application avec le système d'exploitation et à remplacer l'ancienne image système du nœud du capteur par la nouvelle. Le remplacement complet de l'image ne nécessite aucun traitement supplémentaire de l'image système chargée. Comme l'image est compilée et reliée à chaque itération, ces solutions offrent un contrôle très précis sur les reconfigurations possibles et une flexibilité maximale. Cependant, ces approches entraînent une surcharge de bande passante en transmettant et en téléchargeant des images binaires volumineuses. De plus, les parties inchangées d'une application doivent être redistribuées dans le réseau [72]. Dans ce cadre, le système d'exploitation le plus populaire des nœuds capteurs est TinyOS qui génère une image binaire monolithique de l'application entière. La mise à jour complète des images logicielles des applications TinyOS peut être réalisée via le chargeur et le protocole de diffusion Deluge [73].

Mise à jour incrémentielles. Ce sont des mécanismes qui prennent en charge les mises à jour modulaires. Dynamic TinyOS [74] est introduit pour permettre l'échange dynamique de composants logiciels et pour mettre à jour de manière incrémentielle le système d'exploitation et ses applications. Il permet d'activer la composition d'exécution des composants TinyOS sur le nœud capteur. La solution proposée s'intègre parfaitement à l'architecture système de TinyOS. Il ne nécessite aucune modification du modèle de programmation de TinyOS et les composants existants peuvent être réutilisés de manière transparente. Dynamic TinyOS subit une surcharge de performances faibles tout en conservant une empreinte mémoire inférieure à un tiers par rapport aux autres solutions comparables. D'autres systèmes d'exploitation comme SOS [75] et Contiki [71] permettent des mises à jour binaires modulaires au moment de l'exécution. Ils comprennent un chargeur et un éditeur de liens au moment de l'exécution. Le chargeur est chargé de suivre le stockage des modules binaires dans la mémoire de code et d'allouer les ressources appropriées à leur exécution. L'éditeur de liens est chargé de résoudre toute référence faite par les modules au noyau, aux bibliothèques communes ou à d'autres modules du système.

Approches différentielles. Souvent, une petite mise à jour du code du système, telle qu'un bug_x, entraînera uniquement des différences mineures entre la nouvelle et l'ancienne image système. Au lieu de distribuer une nouvelle image système complète, les différences binaires, entre le binaire modifié et le binaire original peuvent être distribuées. Cela réduit la quantité de données à transférer [71].

Synthèse

Un inconvénient majeur des approches de mise à jour complètes est qu'elles sont monolithiques et lourdes à gérer, la grande taille de l'image binaire monolithique entraine une surcharge d'énergie. Cet inconvénient bien que partiellement couvert par les mécanismes incrémentiels et modulaires, reste le problème de la réinstallation de l'application complète qui perturbe l'application en cours. Le mécanisme différentiel comme proposé dans [76] et [77], réduit la quantité des données à transférer. Ce dernier mécanisme présente quand même un inconvénient au niveau de l'exécution des algorithmes complexes pour patcher les images, sur les nœuds capteurs.

Les approches de re-configuration/adaptation

Selon [78], la re-configuration des réseaux de capteurs comporte deux dimensions importantes : locale et distribuée. La re-configuration locale surveille la qualité de service au niveau de la station de base et reprogramme les nœuds capteurs si nécessaire. Elle effectue une adaptation locale d'éléments logiciels sur un nœud individuel. Alors que la re-configuration distribuée surveille localement le contexte et effectue une re-configuration sur chaque nœud. L'adaptation est ainsi distribuée et consiste à une adaptation du comportement à travers le réseau de capteurs. Par conséquent, une adaptation distribuée consiste en une série d'adaptations locales.

Ces re-configurations sont susceptibles de se produire lors de l'exécution de l'application pour : i) le remplacement d'un composant par un nouveau, ii) l'ajout d'un nouveau composant, iii) la suppression

du composant, et iv) la modification des valeurs des variables du composant [63]. Il existe différents mécanismes parmi lesquels nous citons :

FiGaRo. [79] offre une solution de re-configuration dynamique pour les composants logiciels distribués dans un WSN. L'approche de FiGaRo permet de répondre à deux questions à savoir ; déterminer i) pour quoi ? et ii) où doit être reconfiguré ? La première question permet de déterminer ce qui devrait être reconfiguré en termes de composants logiciels à remplacer. La deuxième permet de déterminer les nœuds capteurs concernés par cette re-configuration et pour recevoir le code de mise à jour. Dans FiGaRo, un composant représente une seule unité de fonctionnalité et de déploiement. Les services fournis par un composant sont décrits par son interface. FiGaRo s'appuie sur un modèle de programmation a deux composants principaux : i) Le "*component model*", il définit des constructions pour structurer le code sur les nœuds simples. Il est conçu avec la re-configuration en tête, fournissant ainsi des constructions dédiées pour gérer les dépendances et les versions des composants, et pour simplifier le processus de re-configuration ; ii) le "*distribution model*" , il définit des constructions pour limiter la diffusion des composants uniquement à un sous-ensemble de nœuds donné - la cible de re-configuration - en fonction des caractéristiques spécifiées par les programmeurs des nœuds ou de leur configuration logicielle actuelle. Plus précisément, les dépendances sont explicitement déclarées par le programmeur à l'aide de la macro (généralement écrits en C). Cependant, l'aspect dynamique de FiGaRo - sa principale caractéristique - n'est exploitable que sur le système d'exploitation Contiki.

LooCI. [80] est un modèle de composant spécifique au WSN dont les nœuds de capteurs sont basés sur SunSPOT. L'approche fournit une infrastructure de composants faiblement couplée, axée sur un modèle de liaison événementiel pour les WSN, inspiré des modèles de programmation pilotés par les événements, des architectures orientées service (SOA), des modèles d'interaction publication/abonnement (*publish/subscribe*) et une prise en charge connectable. LooCI prend en charge deux types de composants, à savoir ; i) les macrocomposants qui font référence à un grain grossier et à un service, s'appuyant sur la notion d'isolats. Les isolats sont des unités d'encapsulation de type processus et offrent différents niveaux de contrôle lors de leur exécution. Chaque macrocomposant s'exécute dans un isolat séparé et communique avec le middleware d'exécution ; et ii) les micro-composants sont plutôt des unités de fonctionnalité à grain fin et autonomes. Le modèle de composant LooCI offre également une re-configuration à l'exécution, des définitions d'interface, une introspection et un support pour le re-câblage des liaisons.

FlexCup. [81] ("*Flexible Code UPdates*") est un mécanisme de mise à jour du code qui permet la réinstallation à la volée des composants logiciels dans les nœuds de capteurs, basé sur TinyOS. La re-configuration est effectuée en deux phases à savoir ; i) La phase de génération de code, pendant laquelle les composants logiciels sont compilés et installés sur les capteurs, FlexCup génère les informations pertinentes sous forme de métadonnées sur les composants compilés. ii) La Phase de liaison, durant cette phase FlexCup utilise les métadonnées lors de la mise à jour du code pour combiner les composants modifiés avec d'autres composants au moment de l'exécution et réalise la liaison d'adresse des objets de données. FlexCup est capable de reconfigurer, d'échanger ou de réinstaller des

parties d'une application s'exécutant sur des nœuds de capteur.

OpenCom. [82] Offre une plateforme de re-configuration de composants logiciels au moment de l'exécution, basé sur un modèle de composants générique pour la création d'applications système sans dépendre d'aucun environnement de plate-forme. OpenCOM offre un niveau d'abstraction élevé, connu sous le nom de "*Component Frameworks*" (CFs), utilisé pour modéliser les interactions entre les composants coopérants. Les liaisons de composants dans les CFs peuvent être locales ou distribuées. Les composants internes du système sont représentés par le graphe de composants, qui contient toutes les informations sur les composants déployés et leur mode de connexion. Toute modification de graphe entraine la modification de l'application décrite par ce graphe. En tant qu'application de OpenCom, Gridkit [78] se présente comme un middleware pour les réseaux de capteurs permettant la reconfiguration distribuée sur la base de règles et d'informations de contexte fournies par un moteur de contexte.

Think. [83] est une implémentation du modèle de composant Fractal [84] visant à prendre en compte les contraintes spécifiques des systèmes embarqués, y compris les WSN et fournit une reconfiguration fine au niveau des composants logiciels. Think permet de contrôler avec précision le temps système induit par la génération de métadonnées, requis uniquement par les artefacts reconfigurables [85]. Les spécifications Fractal définissent un modèle de composant hiérarchique, réflectif et à usage général. Une définition de composant exporte des interfaces fonctionnelles (fournies ou requises), des attributs de configuration, et peut également fournir des interfaces non fonctionnelles implémentant des services d'introspection et de re-configuration d'architecture au moment de l'exécution. Une architecture système est décrite à l'aide d'un langage de description d'architecture (ADL), les interfaces étant définies à l'aide d'un langage de description d'interface (IDL).

Maté. [86] est une architecture de machine virtuelle proposée pour les nœuds de capteurs à ressources limitées, basée sur TinyOS. Il offre une mise à jour distribuée, définie par l'ajout et la suppression d'applications. Cette re-configuration dépend de la spécification de l'application. Mais son modèle de programmation n'est pas suffisamment flexible pour prendre en charge un large éventail d'applications.

RUNES. [67] est une approche basée sur les composants de systèmes embarqués en réseau. Il prend en charge des services de middleware personnalisables. RUNES a été proposé dans le contexte d'incendies. Le middleware doit permettre la communication entre les différents périphériques et doit permettre une adaptation du comportement de chaque capteur en fonction du contexte. RUNES se concentre principalement sur les plateformes Unix et Java, et sa mise en œuvre sur les réseaux de capteurs sans-fil repose essentiellement sur les installations dynamiques du système d'exploitation, par exemple, dans [67] le modèle de mise à jour du module de Contiki est exploité.

WiSeKit. [87], [88] est une approche de middleware distribué à base de composants. Le middleware fournit une couche abstraite pour le développement d'applications WSN. Le code est développé selon les exigences d'adaptabilité au niveau de l'application. Les services sous-jacents middleware exposent les API prêtes à formaliser le développement adaptatif des applications WSN et masquent la complexité

des aspects techniques de l'adaptation. Les caractéristiques de la conception de base de WISEKIT sont : le temps d'adaptation, la portée de l'adaptation, la politique d'adaptation, la re-configuration fine et l'adaptation hiérarchique. Dans [87] seules quelques applications exécutées sur un nœud de capteur sont prises en compte.

REMOWARE. [89] est un middleware de re-configuration basé composant, proposé pour les application "*Home monitoring*". Remoware accorde une attention particulière aux coûts généraux liés aux ressources et à l'énergie induits par le processus de re-configuration sur les nœuds capteurs. Le middleware inclut un ensemble de services implémentés indépendamment du logiciel système sous-jacent. Ces services mettent systématiquement à jour les éléments de code requis. Ces services comprennent la préparation des mises à jour binaires, la distribution du code, la liaison à l'exécution, l'allocation et le chargement dynamiques de la mémoire et la préservation de l'état du système.

Synthèse

Le souci majeur des solutions proposées ci-dessus est la gestion des ressources limitées des WSN et particulièrement de ceux des capteurs. Selon l'étude réalisée les solutions à base de composants logiciels semblent les plus efficaces en terme de consommation de ressources. Cette approche de re-configuration modulaire est de loin la plus efficace et suffisamment souple pour prendre en compte les nouvelles exigences, tout en respectant les contraintes de ressources sévères imposées lors de la re-configuration.

Il faut aussi souligner que les mécanismes de la prise en compte de la re-configuration/adaptation sont codés au niveau du middleware et que toute adaptation du mécanisme peut s'avérer lourde et difficile. Le mécanisme de re-configuration s'attache aussi à une re-configuration liée aux caractéristiques physiques des composants, ne tiennent pas compte de leurs traitements ou de traitements équivalents et négligent la sémantique de leur logique de traitement. Bien que ces mécanismes soient inspirés par les IdO de la littérature, nous adoptons une prise de décision de re-configuration de haut niveau. Elle est aussi réalisée de manière déclarative chose qui rend sa modification assez facile (adaptation du mécanisme de re-configuration lui-même). Nous résumons dans le tableau 2.3 les principaux critères de ces solutions.

2.6 Conclusion

Ce chapitre nous a permis de réaliser une étape importante de notre méthodologie de recherche DSRM et qui consistait à une investigation du contexte théorique. Cette investigation a été élaborée selon trois principaux axes de recherches :

— Le premier concerne la représentation sémantique des objets intelligents et l'intégration des techniques du Web sémantique dans le cadre de l'IdO. Cette première investigation nous a permis de déduire que les solutions IdO ne traitent pas de l'adaptabilité des capteurs, et que

Solutions	**Paradigme de reprogrammation**	**Plateforme d'exécution**	**Niveau de la reconfiguration**	**Reconfiguration Distribuée /locale**	**Approche utilisées pour la reconfig.**
FiGaRo	Par Composants	Contik	Couche Applicative	Distribuée	Basée modèles de programmation
LooCI	Par Composants	SunSPOT	Services	Locale	Basée modèles d'interaction
FlexCup	Par Composants	TinyOS	Couche Applicative	Locale	Basée métadonnées
OpenCom	Par Composants	Indep.	Middleware	Distribuée	Basée graphe de composants
Think	Par Composants	Indep.	Toutes les couches	Distribuée	Basée métamodèles + modèle de composants
Maté	Machine virtuelle	TinyOS	Couche Applicative	Distribuée	Non spécifié
RUNES	Par Composants	Contiki	Middleware	Locale	Non spécifié
WiSeKit	Par Composant	Indep.	Middleware	Distribuée	Basée API des services
REMOWARE	Par Composant	Indep.	Middleware	Locale	Basée API des services

TABLEAU 2.3 – Évaluation des mécanismes de re-configuration/adaptation

les ontologies de capteurs existantes ne permettent pas de décrire les capteurs intelligents et les logiques de traitement associées ;

— Le deuxième axe porte sur l'étude des Frameworks pour l'IdO. Notre investigation a montré que les Frameworks existants ne permettent pas de garder trace de l'expertise relative à la conception d'un OI. Ceci est dû principalement à l'absence d'une représentation sémantique des connaissances relatives à la conception/fabrication d'un OI. Nous avons également mis l'accent sur la nécessité d'une méthodologie pour la conception/fabrication des OI et des systèmes intelligents et connectés. Nous avons constaté qu'il n'existe pas de méthodologie sémantique permettant d'accompagner les parties prenantes d'un projet IdO jusqu'à la conception de leur objet intelligent ;

— Le troisième axe nous a permis de réaliser une investigation des plateformes IdO pour la re-configuration/adaptation d'objets intelligents.

Cette revue de la littérature nous a permis de raffiner nos choix conceptuels aussi bien pour le développement du Framework sémantique pour la modélisation d'objets intelligents (objet des chapitres 3 et 4), que pour la spécification et le développement de l'écosystème d'adaptation et de reconfiguration d'objets intelligents (objet du chapitre 5).

Deuxième partie

Contributions théoriques : Approche ontologique pour la conception/fabrication et l'exploitation d'un Objet Intelligent

Dans le chapitre précédent, nous avons présenté une investigation approfondie des travaux connexes. L'analyse de ces travaux nous a permis d'abord de constater :

1. L'absence de Frameworks sémantiques qui permettent de garder trace de toutes l'expertise et des connaissances relatives à la conception/fabrication et exploitation d'objets intelligents.
2. Que les solutions ou modèles ontologiques existant n'ont pas considéré le concept de capteur intelligent, concept indispensable pour définir des écosystèmes IdO intégrant des objets intelligents.
3. Que les méthodologies existantes ne tiennent pas compte du détail de conception des objets intelligents
4. Que les solutions ou plateformes IdO existantes, même s'ils sont fondés sur le Web sémantique, ne proposent pas de mécanisme de reconfiguration des objets ou capteurs intelligents qui soit déclaratif et utilisant des inférences sur les modèles ontologiques.

Cette analyse nous a ensuite permis de définir le cadre global de la solution proposée. Celui-ci concerne la proposition de deux catégories de solutions :

1. Un cadre de travail (Framework) qui exploite les potentialités des technologies du Web sémantique et des ontologies pour supporter les phases de conception/fabrication et d'exploitation d'Objets Intelligents.
2. Une architecture de plateformes ou d'écosystèmes IdO fondée sur le Web sémantique intégrant un mécanisme de reconfiguration/adaptation de plateformes IdO. Il est à noter que notre contribution ne se situe pas au niveau de l'architecture de plateformes IdO fondées sur le Web sémantique. Elle se situe plutôt au niveau du mécanisme de reconfiguration/adaptation. Celui exploite encore une fois les potentialités de raisonnement et d'inférences, qu'il est possible de réaliser sur les modules ontologiques pour effectuer ces reconfigurations de l'écosystème. Ce mécanisme se base d'autant plus sur la granularité fine offerte au niveau des logiques de traitement embarquées et sur les services de reconfiguration fournies par le middleware choisi.

L'ensemble des éléments composant cette solution, regroupe alors les artefacts IT à concevoir et à développer tout au long de la thèse comme l'illustre la figure 1.1.

Cette deuxième partie de la thèse concerne toutefois, l'application du deuxième type d'activité de la méthodologie de recherche adoptée pour concevoir ou améliorer des artefacts IT nécessaires à la première catégorie de solution.

Les principales contributions de cette partie de la thèse sont donc :

1. Un Framework sémantique générique de modélisation d'objets intelligents dont la spécialisation permet d'offrir un ensemble de modèles sémantiques (ou d'ontologies) décrivant les différentes interactions et articulations à la fois structurelles et comportementales d'objets spécifiques particuliers comme par exemple un VI ou une PI.

   Ce Framework que nous avons baptisé FSMS trouve ses fondements dans les approches méthodologiques de conception logicielle ainsi que celles liées au développement de modèles ontologiques. Les détails de ce Framework sont présentés au niveau du chapitre 3

2. Sur la base du Framework FSMS, de ses modèles génériques et tout en tenant compte des différents besoins dégagés, les modules ontologiques sont développés, alignés et fusionnés dans une ontologie globale que nous avons baptisé SMS.

   Celle-ci représente le noyau générique de notre Framework de modélisation sémantique d'objets intelligents. Elle joue ainsi un double rôle ; celui de formalisme adéquat pour la structuration et la capitalisation des connaissances liées aux modèles structurels et comportementaux d'Objets Intelligents et celui d'un mécanisme d'interopérabilité sémantique du vocabulaire et des données manipulées pour fabriquer et exploiter un Objet Intelligent. De plus, parmi l'ensemble des modules identifiés, elle représente ceux qui sont les plus génériques et les plus réutilisables du Framework FSMS. Sa description détaillée est fournie dans le chapitre 4.

Chapitre 3

Framework sémantique générique de modélisation d'Objets Intelligents

Sommaire du présent chapitre

3.1 Introduction

Dans ce chapitre, nous présentons la première contribution de la thèse. Il s'agit d'un Framework conceptuel générique de modélisation sémantique d'objets intelligents connectés (OI). Afin de répondre aux besoins des différents acteurs concernés par les OI, ce Framework nommé FSMS offre les modèles conceptuels requis pour concevoir, fabriquer et développer ces objets connectés y compris leurs plateformes d'exploitation. Contrairement aux modèles conceptuels proposés dans les différents Frameworks et méthodologies revues dans la littérature, ceux proposés dans le cadre du Framework FSMS s'attachent à la représentation des connaissances expertes des concepteurs d'OI. Ils sont d'autant plus porteurs de sémantiques et donc exprimés sous forme d'ontologies ou de modules ontologiques réutilisables.

L'intérêt de ce Framework générique est qu'il est possible de le spécialiser dans le contexte de projets spécifiques comme par exemple le projet SmartSensing où le modèle d'un VI, d'une PI ou même celui de l'écosystème serait un cas spécifique du modèle de l'Objet Intelligent connecté.

Le Framework FSMS est illustré par la figure 3.1 où l'on identifie l'ensemble des acteurs qui vont interagir ensemble dans le cadre de ce Framework dans le but ultime de développer un écosystème. Ce dernier a pour rôle d'interagir avec l'utilisateur final pour lui offrir les services requis. Le Framework

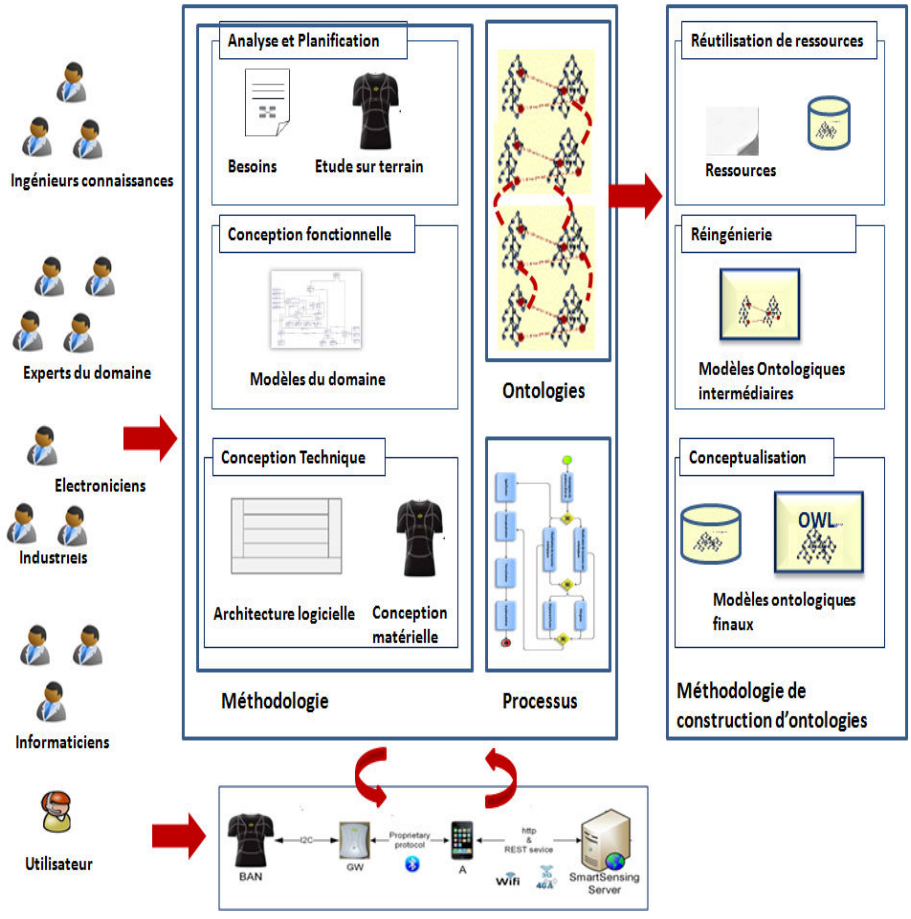

FIGURE 3.1 – Composantes du Framework sémantique

FSMS est formé de trois composantes :

— Une composante sémantique. Il s'agit tout d'abord de la contribution principale de ce travail réalisé dans cette thèse. Elle est composée à son tour d'un ensemble de modules sémantiques (c.-à-d. Ontologies) que nous développons et décrivons en détail dans le chapitre 4.

— Une composante méthodologique à deux niveaux. Celle-ci offre une approche systématique pour :

1. Prescrire et guider, dans un premier niveau, l'exécution d'un ensemble d'activités qu'il est possible de réaliser dans le cadre du Framework FSMS pour modéliser un OI.
2. Définir pour chaque activité l'ensemble des inputs (documents ou données en entrée) et des outputs (livrables).
3. Identifier, en particulier, les modèles du domaine ainsi que les composants ou modules nécessaires à modéliser sémantiquement un OI spécifique donné ; ceci en définissant les activités et les tâches à réaliser par chaque rôle ou acteur intervenant dans l'ensemble des phases du cycle de vie de l'OI.
4. Appliquer à chaque module sémantique requis, une méthode de construction/développement des modèles ontologiques ainsi que leurs logiques d'exploitation dans un contexte applicatif donné. Cette méthode constitue le deuxième niveau méthodologique préconisé par le Framework. Différentes méthodes de modélisation sémantique existent dans la littérature. Dans le cadre de cette thèse, nous favorisons celles qui fournissent des modèles sémantiques modulaires et réutilisables. Les détails quant aux étapes de cette méthode et notre manière de l'avoir utilisé seront décrits au fur et à mesure dans le chapitre 4.

— Et une composante processus. Il s'agit d'un ensemble de processus qui vont guider l'usage des activités de la méthodologie pour répondre à un besoin donné. Ces processus illustrent l'enchaînement (orchestration) des activités de la méthodologie supportée par le Framework FSMS. Ils jouent le rôle d'un support méthodologique et de bonnes pratiques pour l'ensemble de ces acteurs surtout quand il s'agit de problèmes de conception récurrents. Ils sont structurés en termes de rôles, de tâches, d'activités ou de sous-processus et ils peuvent à terme être automatisés pour former un outil de modélisation à intégrer au Framework FSMS.

Le reste du chapitre est structuré comme suit :
La section 2 se focalise sur la définition de la notion d'objet intelligent. La section 3 présente les différentes exigences et principes qui vont guider la spécification du Framework FSMS et en particulier son architecture. La section 4 montre un exemple de spécialisation du Framework FSMS pour la modélisation d'une catégorie spécifique d'objets intelligents (par exemple un VI ou une PI). L'objectif étant d'identifier les composantes génériques et réutilisables de la composante sémantique du Framework FSMS et la manière de la spécialiser et de la raffiner. La section 5 décrit la méthodologie et ses activités et la section 6 décrit l'ensemble des processus où l'on va montrer la manière d'exécuter les activités précédemment définies ainsi que leur interaction avec les modules ontologiques du Framework FSMS

soit en tant qu'input soit en tant qu'output.
Nous terminons le chapitre par nos conclusions quant aux apports de la proposition et son utilisation dans les chapitres qui suivent.

3.2 Définition de la notion d'Objet Intelligent

Quel que soit le domaine d'application considéré, l'objectif principal de la modélisation sémantique d'objets intelligents est de fournir une représentation complète des connaissances et des données relatives aux deux phases de conception/fabrication et d'exploitation d'un OI. Dans le cadre de projets de recherche dont l'objet d'étude principal est l'Objet Intelligent en question, ces deux phases, seront enrichies par une troisième phase de validation en laboratoire et/ou d'implémentation-évaluation sur terrain de l'ensemble des artefacts créés y compris l'OI lui-même. Étant donné que ces trois phases de fabrication, d'exploitation et de validation constituent des niveaux d'abstraction différents relatifs au même concept d'OI, nous mettons l'accent sur une approche qui favorise la séparation et le découplage entre ces niveaux d'abstraction.

Lorsque l'on effectue une recherche dans la littérature sur les objets intelligents, les objets connectés ou encore l'Internet des Objets, nous remarquons l'utilisation du terme « Intelligent » de manière légèrement abusive. Cela nous amène alors à nous poser les questions suivantes : Quelle est la signification du terme intelligent ? Qu'est-ce qui rend un Objet Intelligent ? Comment concevoir un Objet Intelligent et quels sont ses composants ? Comment un grille-pain ou un stylo par exemple, peuvent-il devenir « Intelligents » ?

Pour trouver un sens au terme «Objet Intelligent» nous commençons tout d'abord par définir les deux termes «Objet» et «Intelligent».

Il existe plusieurs définitions du terme «Objet», nous retenons celle qui nous semble la plus proche de notre perception d'un objet dans le cadre des présents travaux de thèse. Selon le dictionnaire Larousse un objet est défini comme une chose considérée comme un tout, fabriquée par l'Homme et destinée à un certain usage. Une lampe, un livre, un grille-pain ou un stylo sont des objets. De même, à partir des différentes définitions du terme « Intelligent » nous retenons celle du Larousse où le terme se dit d'un bien dont la maintenance ou le fonctionnement sont assurés par un dispositif automatisé capable de se substituer, pour certaines opérations, à l'intelligence humaine : Voiture intelligente ou immeuble intelligent. D'un autre côté, l'«Intelligence Humaine» se définit comme étant l'aptitude d'un être humain à s'adapter à une situation, à choisir des moyens d'action en fonction de circonstances données.

En nous appuyant sur ces définitions et en substituant ainsi le terme à «être humain» nous déduisons alors qu'un Objet Intelligent est un objet qui présente des caractéristiques matérielles et/ou immatérielles. Il est conçu et créé intentionnellement pour un usage précis et il a la capacité de s'adapter à des circonstances et d'agir en conséquence.

Dans la littérature et à notre connaissance il n'existe pas de définition claire ou standard d'un

Objet Intelligent. Un Objet Intelligent est généralement défini comme étant un objet physique du monde réel équipé de capteurs et/ou d'actionneurs qui lui permettent de transcender son usage initial pour proposer de nouveaux services et de nouvelles fonctionnalités. Doté d'un matériel électronique, un Objet Intelligent est capable de communiquer et d'échanger des données avec d'autres entités physiques ou numériques via un réseau qui le relie à Internet ou à un réseau local. Un OI est donc identifié par ses composants (ou sa structure) et son comportement (les services qu'il offre ou alors ses réactions à des stimuli externes).

Avant donc de concevoir ou encore d'exploiter un Objet Intelligent, il est important de spécifier à la fois sa structure et son comportement. Comme le montre la figure 3.2, un OI peut être représenté ou peut avoir une structure conceptuelle composée de trois éléments : le composant d'objet primaire, le composant du domaine d'application et le composant représenté par l'ensemble des composants électroniques.

— Le composant d'objet primaire contient des concepts génériques pour la description de la composante principale d'un objet en tant qu'objet classique, à savoir l'objet lui-même (par exemple un stylo ou un grille-pain). Il comprend des concepts utilisés pour spécifier les caractéristiques de composants-primaires (encre, métal), de forme, de couleur, ...qui sont nécessaires à une description complète à sa fabrication et à son usage. Un livre, par exemple, est représenté respectivement sous la forme d'un ensemble de pages ou d'un ensemble de caractères codifiés dans un certain format. Il peut être imprimé, lu ou encore transmis sur un réseau.

— Le composant du domaine d'application regroupe tous les concepts liés au domaine d'application et au contexte d'usage auquel peuvent être intégrés les concepts liés au contexte social du domaine d'application (c.-à-d. les usagers, leurs profils, leurs besoins, leurs préférences, ...).

— Le composant formé par l'ensemble des composants électroniques contient les concepts qui participent à la description des composants matériels et logiciels intégrés aux objets primaires ou qui viennent les augmenter. Ces concepts incluent ceux décrivant les capteurs, les actionneurs, les réseaux de capteurs, les systèmes et les protocoles de communication. Ils incluent également les concepts liés aux différents traitements possibles qui qualifient alors le comportement de ces composants électroniques. Il est toutefois possible de séparer les composants matériels et les composants logiciels en deux grandes catégories différentes.

Étant donné que dans le cadre de cette thèse, les OIs concernés sont uniquement formés de capteurs ou de réseaux de capteurs, nous leur fournissons également dans ce qui suit une ou plusieurs définitions.

Capteur ou réseau de capteurs intelligents Dans le but de pouvoir représenter sémantiquement les réseaux de capteurs, plusieurs définitions du concept capteur ont été proposées. Elles décrivent un capteur, le plus souvent, comme le composant électronique qui permet l'observation d'«une quantité physique (e.g. température, profondeur, etc) d'une caractéristique (e.g., un lac) et les observations signalées » [16] ou encore comme «une source de données qui produit une séquence d'éléments de données au fil du temps» [90]. Dans une vision un peu plus large on pourrait également se référer à

[37], où les auteurs décrivent un capteur comme «une source produisant une valeur parmi un intervalle de valeurs qui représente un phénomène dans un domaine d'intérêt donné».

Cependant, ces définitions ne se concentrent pas sur la vue "classique" d'un capteur comme un dispositif physique sensible à un phénomène déterminé qu'il capte/détecte ou mesure et transforme cette grandeur physique observée en signal (en général électrique) ou en grandeur utilisable. En se référant à la norme IEEE 1451, l'auteur dans [91] définit un capteur intelligent comme un capteur enrichi, par rapport à un capteur classique, d'un micro-contrôleur qui conditionne les signaux avant transmission au réseau de contrôle. Il filtre les bruits indésirables et compense les erreurs avant l'envoi des données. Certains capteurs peuvent être configurés sur mesure pour produire des alertes sur leur propre état lorsque les seuils critiques sont atteints.

Nous constatons donc qu'un capteur intelligent ou un réseau de capteurs intelligents est un composant d'un OI qui est à son tour composé d'autres éléments. Notre approche doit donc, de manière itérative, tenir compte des exigences de modélisation de chaque élément composite et de chacun des éléments le composant jusqu'à aboutir à des composants atomiques.

Logique de traitement embarquée Les traitements embarqués au niveau du micro-contrôleur permettent de décrire les nouvelles fonctionnalités ajoutées aux fonctions de base d'un objet. Ces fonctionnalités sont généralement déterminées par l'usage que l'on souhaite faire de l'objet et dépendent donc du domaine ou du sous domaine applicatif. Elles peuvent également avoir une description thématique fournissant davantage d'informations sur l'application au-delà des facettes électroniques.

Des relations d'interfaçage ou de médiation (ou de communication) possibles entre les descriptions des composants et de leurs comportements avec celles du domaine d'application peuvent être également définies, car chaque composant électronique intégré à un objet doit répondre aux nouvelles fonctionnalités définies par le domaine applicatif.

Il est toutefois important de souligner qu'un même objet peut avoir des usages très différents et des descriptions structurelles aussi différentes, qui varient d'un domaine d'application à l'autre ou d'un domaine à un autre domaine qui lui est subordonné. Par exemple, le domaine du vêtement de sport est considéré comme subordonné au domaine du vêtement en général.

Par analogie à la théorie sous-jacente aux approches et à la conception orientée objets, différents « niveaux d'abstraction » peuvent être identifiés. Ces niveaux peuvent guider la conception d'un Objet Intelligent depuis le niveau le plus abstrait, jusqu'à des niveaux beaucoup plus spécifiques voire concrets où une instance d'Objet Intelligent est entièrement conçue et définie.

Contrairement à ce qui est illustré dans la figure 3.2, un Objet Intelligent peut ne pas être toujours basique ou atomique. Il peut même représenter un système complexe tel une voiture ou un bâtiment. Un Objet Intelligent composite est donc formé d'autres objets composites et/ou atomiques dont la description fait appel de manière itérative à la description de l'ensemble de ses composants et des relations qui les relient ensemble.

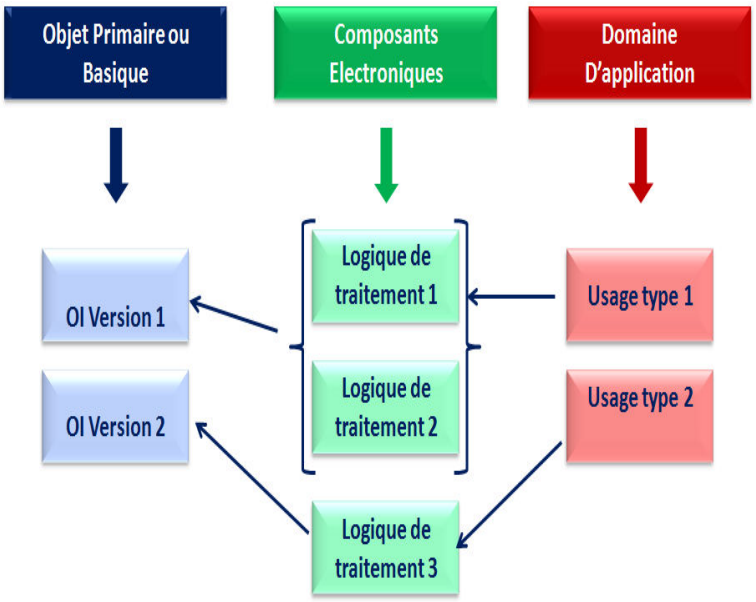

FIGURE 3.2 – Composants d'un Objet Intelligent

3.3 Principes et exigences du Framework sémantique de modélisation d'objets intelligents connectés

Le Framework FSMS est dirigé d'abord par :

1. les exigences des différents acteurs intervenants chacun pour sa part selon son domaine de compétences ;
2. les composants d'un Objet Intelligent connecté ;
3. ses caractéristiques et son comportement ;
4. ensuite, par les scénarios de ses différents usages possibles ;
5. les spécificités de son domaine d'application ;
6. ainsi que les différentes phases de son cycle de vie.

Ce principe que nous venons d'énoncer constitue un principe important de conceptualisation qu'il est nécessaire de considérer dans la méthodologie préconisée par le Framework FSMS (en particulier dans les activités de conceptualisation et de raffinement). Le schéma de la figure 3.3 illustre ce principe et le décline en un ensemble d'étapes et de bonnes pratiques.

De plus, le Framework FSMS est fondé sur des modèles sémantiques génériques représentant un cadre conceptuel à la fois précis (.c-à-d. délimitant les frontières d'usage du Framework) et extensible du corps de connaissances décrivant tous les aspects et perspectives liés au concept d'Objet Intelligent. La spécialisation ou l'instanciation de ces modèles génériques permettrait alors de générer soit des modèles plus spécifiques soit des descriptions concrètes d'objets intelligents spécifiques.

Notre philosophie s'attache, de plus, à une description de ces OIs qui soit fortement découplée de leurs scénarios d'usage, et qui respecte le principe de séparation des préoccupations ; ceci en vue de

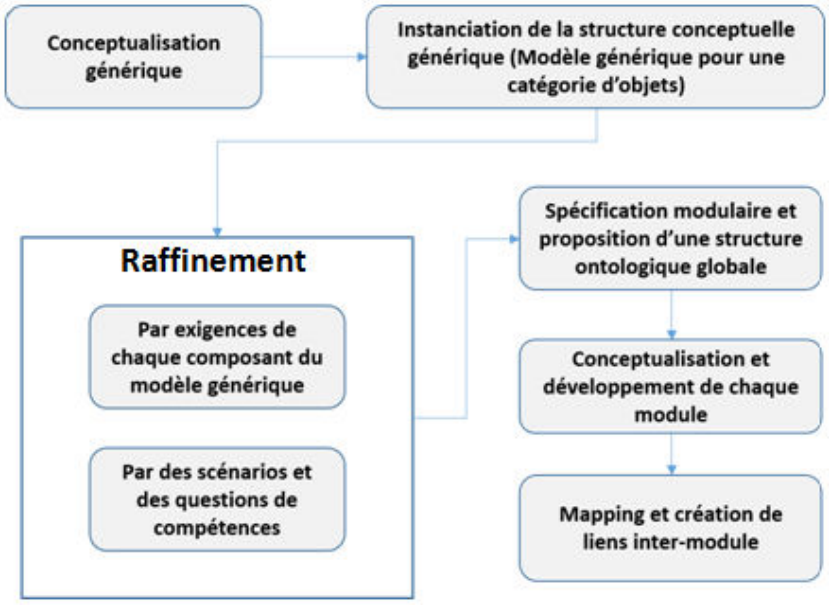

FIGURE 3.3 – Principe de conceptualisation sémantique d'OI

garantir une description qui soit fidèle à la réalité et une adéquation des modèles à différents types d'usages et/ou à des usages qui changent avec le temps et la progression des activités de l'utilisateur.

Il est toutefois, possible de réaliser différentes médiations, compositions et/ou inférences, en vue de faire correspondre ces descriptions sémantiques des objets intelligents à la sémantique de leurs usages.

Le Framework FSMS doit être collaboratif et centré sur la réutilisation, l'adaptabilité et l'enrichissement continu de l'ensemble de ses composantes tout en étant fidèle aux domaines respectifs de chacun de ses intervenants.

En guise d'un vrai système de gestion des connaissances des différents acteurs et domaines, il doit offrir des possibilités de récupérer des connaissances représentant différents points de vue et différents niveaux d'abstraction. Il doit être évolutif, en vue de tenir compte d'une part de l'évolution technologique qui pourrait engendrer davantage d'hétérogénéité et d'autres part de l'évolution des besoins des différents acteurs. Même dans le cas où l'ensemble des concepts et des connaissances sont formalisées sémantiquement de manière très précise, une ré-ingénierie des descriptions s'avère de temps à autre nécessaire pour tenir compte d'éventuelles évolutions.

Comme nous l'avons illustré et décrit précédemment dans la figure 3.1, le Framework FSMS est formé d'une composante sémantique, une composante méthodologique et une composante processus. Celles-ci seront décrites dans le reste de ce chapitre.

3.4 La Composante sémantique générique du Framework

D'après l'analyse faite précédemment, la composante sémantique du Framework est fortement liée aux primitives de modélisation d'un ou plusieurs objets intelligents connectés. Ces primitives véhiculent le vocabulaire utilisé par les différentes parties prenantes, y compris les utilisateurs finaux,

pour décrire la nature de l'objet, le domaine dans lequel il sera utilisé, son comportement, son mode de communication avec les autres objets ou systèmes.

En somme, ces primitives représentent les modèles ontologiques nécessaires à décrire les connaissances liées à ces objets intelligents et à leurs usages dans différents contextes. Nous les énumérons dans le présent paragraphe pour y faire référence dans l'architecture de la composante sémantique du Framework FSMS illustrée dans la figure 3.4. Il s'agit des modules ontologies de :

1. Domaine
2. Composants physiques de l'objet ou objet primaire
3. Composants électroniques
4. Logiques de traitement

Ils seront toutefois, spécialisés, décomposés en modules beaucoup plus spécifiques et décrits de manière plus détaillée dans le chapitre 4 pour tenir compte des spécificités de chaque projet comme par exemple le projet SmartSensing et son application au domaine du sport.

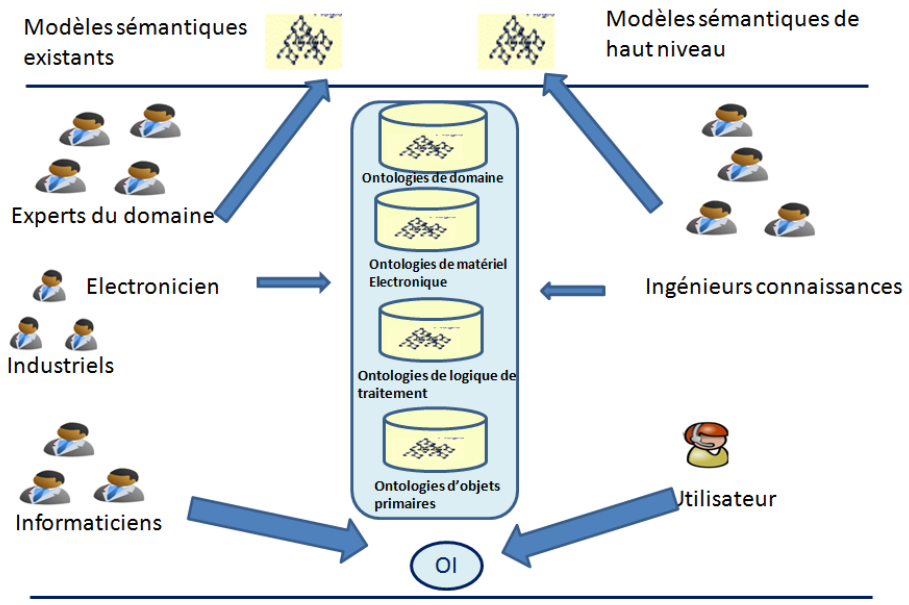

FIGURE 3.4 – Architecture de la composante sémantique du Framework FSMS

La conception de chaque module ontologique trouve ses fondements dans les méthodologies de conception liées aux ontologies où un intérêt spécifique est alloué à la construction collaborative d'ontologies faisant intervenir différents acteurs, différentes exigences et différents objectifs. Elle trouve également son fondement au niveau de la méthodologie de recherche en science du design pour les activités de support à la construction des modèles, en particulier pour les phases d'investigation, d'évaluation et de validation de chacun de ces modèles (considérés comme les artefacts IT à concevoir, implémenter et valider).

Cet ensemble d'ontologies constitue la pierre angulaire du présent travail. Tout au long des chapitres suivants, nous montrerons leurs utilités, dans toutes les phases de conception/fabrication et exploitation

des objets intelligents. Différentes méthodes de conception ou de modélisation sémantique existent. Dans le cadre de cette thèse nous utilisons une adaptation de la méthode Néon [92]. Cette méthode est illustrée à l'aide d'un ensemble de processus. Elle est utilisée pour modéliser / construire les ontologies. Nous détaillons l'utilisation que nous en faisons dans le chapitre 4.

3.5 Spécialisation de la composante sémantique du Framework FSMS : Application à la modélisation sémantique de Vêtements ou de panoplies intelligentes

Cette section présente la manière de spécialiser la composante sémantique générique de FSMS en vue de modéliser une catégorie plus spécifique d'OI. Un VI ou une PI sont considérés comme étant des objets intelligents ayant un niveau d'abstraction inférieur à celui de l'OI mais qui peuvent être aussi génériques relativement à leur domaine d'usage. Ils seront à leur tour spécialisés pour affiner les descriptions de leurs déclinaisons spécifiques en vêtement d'hiver ou d'été, en vêtement de sport ou de travail, etc.

La spécialisation de la composante sémantique générique est alors nécessaire tout d'abord pour définir le concept de VI. Cette spécialisation permet donc d'identifier les modèles génériques à spécialiser et ceux à développer pour tenir compte de la spécificité de la catégorie de l'objet en question, de son domaine et de ses exigences. Ces étapes de raffinement sont présentées dans ce qui suit et appliquées dans le cadre du VI.

3.5.1 Raffinement du Modèle générique d'un OI pour modéliser un vêtement intelligent (VI)

Cette étape de raffinement nécessite la définition des exigences de modélisation du VI considéré comme étant un OI spécifique. Pour cela, nous nous basons tout d'abord sur sa définition. La notion de vêtement intelligent (smart garnment également appelé smart clothing) a été introduite dans plusieurs disciplines et plus particulièrement celle du textile et celle de l'électronique. La définition d'un VI par rapport à une discipline donnée permet d'exprimer le point de vue des experts de cette discipline. Sa définition s'en trouve alors différente d'une discipline à l'autre et peut aussi engendrer l'usage d'une terminologie spécifique à chaque discipline, ce qui conforte une fois de plus notre choix pour les modèles sémantiques et plus spécifiquement ontologiques. Nous retenons les définitions suivantes identifiées suite à une revue de la littérature :

— Le "vêtement intelligent" est l'une des applications du domaine de textile intelligent. Ce terme désigne tous les vêtements confectionnés avec des textiles intelligents ou dans lesquels ils sont appliqués [93] ;

— Le "vêtement intelligent" est un produit matériel combinant une technologie électronique à un vêtement générant automatiquement certaines fonctions [94].

En se fondant sur ces différentes définitions trouvées dans la littérature, nous considérons un vêtement intelligent comme un vêtement ordinaire, augmenté de composants électroniques capables de détecter des stimuli de l'environnement et de les adapter en intégrant des fonctionnalités supplémentaires (intelligentes) dans la structure du vêtement. Sur la base de ces derniers, nous obtenons un vêtement intelligent, qui peut mieux remplir sa fonction principale en tant que vêtement et fournir une valeur ajoutée à l'utilisateur. En outre, les fonctionnalités intelligentes de ces vêtements sont traduites par exemple pour suivre les postures, les gestes, les signes vitaux d'un utilisateur, etc.

La conception/fabrication et l'exploitation d'un vêtement intelligent sont alors dirigées par l'utilisation que l'on fait du vêtement, des besoins et des exigences des utilisateurs finaux (les clients) qui sont reliés au domaine pour lequel le vêtement a été conçu (comme par exemple, militaire, pompier, sport, santé, etc).

Ainsi, nous déterminons quatre entités qui guident la modélisation d'un vêtement intelligent. Par analogie aux composants de la composante sémantique générique FSMS, ces entités sont : le Vêtement, les Composants électroniques, les logiques de traitement et le Domaine d'application tels qu'ils sont illustrés dans la figure 3.5. Ce dernier pourra à son tour se décliner en plusieurs sous-domaines qui guideront la conception de plusieurs niveaux d'abstraction. Les cercles montrent les différentes entités qui ont un impact sur la description du VI. Au centre du modèle, et à l'intersection des cercles, se situe alors le Vêtement Intelligent. Ce dernier sera le résultat de l'ensemble des choix conceptuels des entités qui le composent.

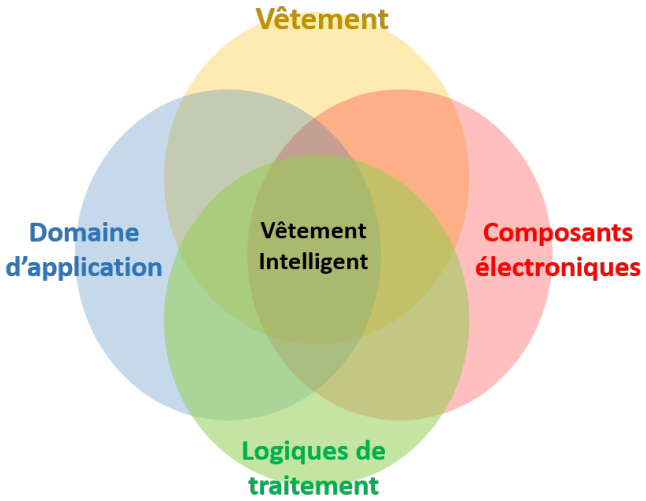

FIGURE 3.5 – Représentation de Vêtement Intelligent

— Le cercle du Vêtement se compose des caractéristiques présentes dans les vêtements ordinaires (taille, femme/homme, etc), nature du tissu, saison (printemps, hiver, etc) et définit une classification des différentes catégories de vêtements (veste, cuissard, etc.).

— Le cercle des Composants électroniques. Ceux-ci se distinguent par la nature des matériaux et des fonctionnalités qu'ils offrent et qui sont inclus dans le vêtement. Un composant électronique est lui-même constitué d'autres composants électroniques ayant différents niveaux de granularités (capteurs, micro-contrôleur, etc.).

— Le cercle des logiques de traitement à embarequer dans le micro-contrôleur. Ceux-ci permettent de modéliser et d'implémenter les fonctionnalités identifiées en réponse à des exigences fonctionnelles non basiques.

— Le cercle du Domaine d'application définit les exigences du domaine pour lequel le vêtement sera fabriqué et exploité. Il spécifie les besoins répondant aux questions ; Quel vêtement utiliser ? Quels sont les types de connaissances et d'informations à extraire ou transmettre dans le cadre du domaine ? Quels sont les composants électroniques nécessaires à intégrer dans le vêtement pour qu'il puisse répondre aux exigences fonctionnelles du domaine ?

3.5.2 Raffinements du modèle générique du VI pour sa modélisation structurelle et comportementale

Dans cette section, nous tentons d'affiner le modèle générique du VI en analysant ses exigences structurelles et comportementales aussi bien au niveau de sa fabrication qu'au niveau de son utilisation comme l'illustre bien la figure 3.6. Chacune des entités formant un VI sera à son tour spécialisée et affinée pour tenir compte de la spécialisation du VI à un domaine spécifique donné.

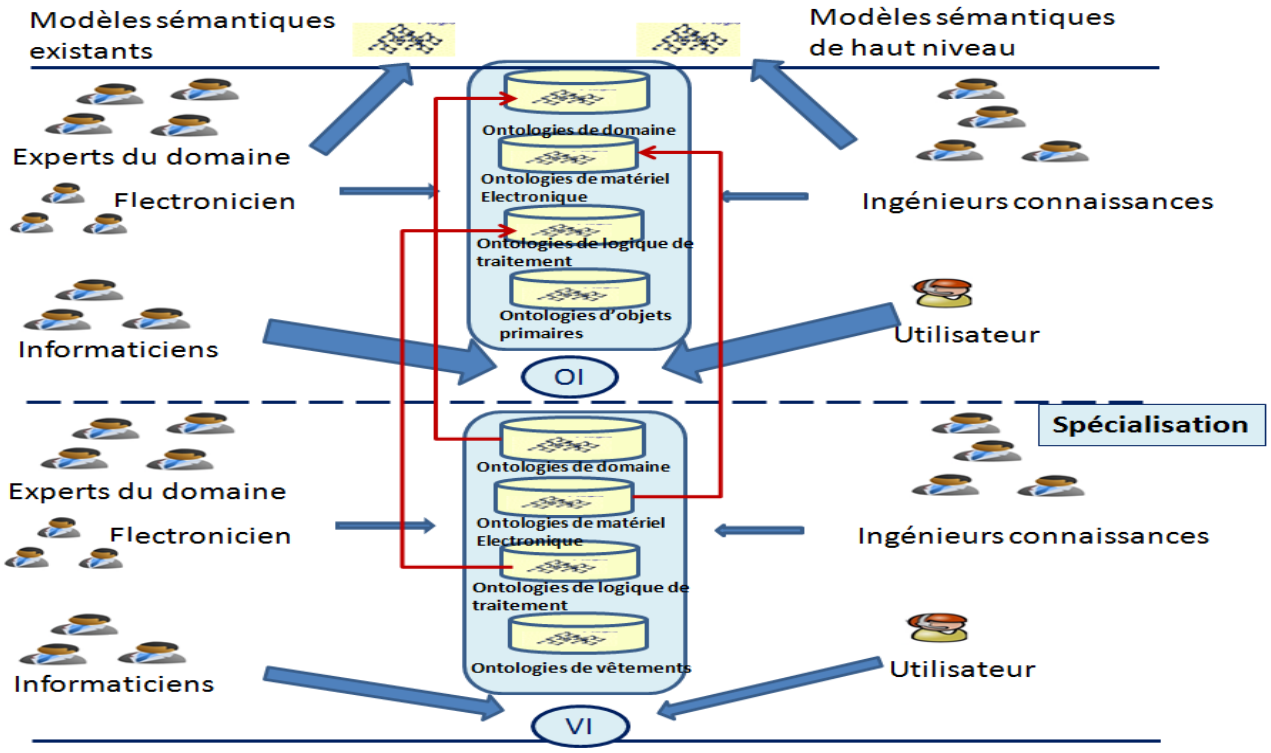

FIGURE 3.6 – Modélisation sémantique de VI

Étant donné qu'un VI est un OI composite, nous baserons sa modélisation structurelle (et même comportementale) sur les exigences de modélisation de chacun de ses composants. Nous dégageons ainsi les connaissances sémantiques liées aux panoplies de vêtements, celles liées aux composants électroniques (capteurs intelligents, capteurs basiques, micro-contrôleurs, . . .), ainsi que celles liées au domaine d'application et d'usage.

Exigences selon les composants du modèle d'un VI

Exigences liées à l'objet primaire : Vêtement ou panoplie. Un VI peut tout d'abord être constitué d'un seul élément ou de plusieurs éléments donnant ainsi lieu à une panoplie vestimentaire. En effet, le choix de la tenue vestimentaire dépend fortement du domaine d'utilisation, et de l'activité faite par son porteur. Par exemple, pour un plombier une tenue peut être une combinaison, ou un ensemble formé d'une veste et d'un pantalon. Il peut également ajouter à sa tenue un accessoire tel qu'un masque ou un casque. Actuellement, plusieurs domaines sont en train de définir des normes et des standards pour ces tenues ou panoplies vestimentaires dont l'élaboration tient compte de différents facteurs telles que la sécurité, l'esthétique et la flexibilité pour réaliser les activités du domaine en question.
Nous définissons alors une "Panoplie vestimentaire" comme une tenue composée d'un ensemble de vêtements et éventuellement d'un ensemble d'accessoires. Des capteurs peuvent être intégrés aussi bien dans les vêtements que dans les accessoires d'une panoplie, on parle alors de "panoplie intelligente".

Exigences de modélisation liées au domaine d'usage. Différentes exigences peuvent à la fois guider et contraindre un domaine d'application ou d'usage d'un OI ou d'un objet plus spécifique comme un VI ou une PI. Un premier niveau d'exigence vient du niveau de complexité du domaine d'usage et son degré de déclinaison en sous-domaines plus spécifiques et moins complexes. De plus, différents points de vue peuvent guider la définition d'un domaine d'usage. Celui-ci peut être considéré selon la perspective des activités à réaliser ou alors selon la perspective des phénomènes ou observations à mesurer. Une activité réalisée au moyen d'un VI peut être interprétée de manière différente selon le cas où celui qui l'observe est un médecin ou un coach, par exemple. Il en est de même pour les mesures et leurs indicateurs.
Il en est de même par rapport à l'identification des données brutes issues des capteurs ou alors de données calculées et/ou traitées grâce à des composants électroniques plus intelligents comme par exemple les micro-contrôleurs. Cela influence de la même manière les logiques de traitements spécifiées pour permettre et guider le comportement des capteurs intelligents. En effet, pour chaque domaine, sous-domaine, activité ou indicateur des logiques de traitements (algorithmes) spécifiques doivent être décrits, spécifiés et implémentés dans le langage informatique du capteur.

Exigences liées aux composants électroniques. Les capteurs inclus dans une panoplie ou un vêtement forment à eux seul un vrai réseau embarqué de capteurs et qui pourrait offrir différents services à son porteur ou à d'autres acteurs.
Il est à noter qu'il existe un nombre important de composants électroniques basiques qui peuvent être utilisés pour former des capteurs ou des réseaux de capteurs intelligents. A cause de ce nombre

important de capteurs, les concepteurs ainsi que l'ensemble des acteurs du projet SmartSensing font face de manière continue à une hétérogénéité accrue, quant aux protocoles, aux unités de mesures, aux fonctionnalités, . . . , malgré la présence sur le Web de différentes ressources de description des capteurs comme les catalogues, les bases de données ou encore les "data sheets".
Mise à part la dépendance du choix des composants électroniques (surtout ceux qui sont basiques) par rapport aux données spécifiques au domaine, il est important de considérer le choix de ces composants en fonction d'autres types d'exigences qui peuvent être aussi bien fonctionnelles que non fonctionnelles.

3.5.3 Synthèse de la phase de raffinement : Identification des modules ontologiques

L'exécution de cette étape d'analyse et de raffinement liée aux exigences de modélisation de l'ensemble des entités formant un OI et par conséquent un VI ou tout autre objet spécifique, permet de dériver un ensemble de modèles sémantiques qu'il est important de spécifier, de définir et d'implémenter. Ceux-ci concernent une décomposition des modèles initiaux en des modules plus spécifiques. En particulier, la composante sémantique générique de modélisation d'un VI, que nous proposons, est fondée d'après l'analyse précédente, sur principalement les modules ontologiques suivants où l'on identifie ceux qui sont génériques et réutilisables de ceux qui sont spécifiques :

— Le Module "S3N-Core" : Cette ontologie décrit les différents capteurs intelligents déployés dans un vêtement de sport. Elle réutilise par adaptation l'ontologie SOSA/SSN. Un capteur intelligent a la possibilité d'être adapté en fonction du sport sélectionné. Cette adaptation est définie par un ensemble d'algorithmes dédiés au calcul des indicateurs de performance du sport en question ;

— Le Module "S3N-Algorithm" : Cette ontologie est dédiée à décrire les algorithmes qui seront codés et déployés sur les capteurs intelligents et ce, en fonction du sport choisi. Cette ontologie décrit principalement les dépendances fonctionnelles et temporelles des données (dataflow) de ces algorithmes. Ces dépendances permettent de déterminer pour chaque indicateur la suite des entrées nécessaires pour le calculer ;

— Le Module "Sport" : cette ontologie de domaine définit les activités sportives. Elle permet de relier pour chaque sport l'ensemble de ses indicateurs de performance. Elle permet également d'associer pour chaque type de sport le vêtement adéquat.

Ces modules et d'autres sont décrits au niveau du chapitre 4. L'interaction entre l'ensemble de ces modules ontologiques, ainsi que les ontologies de références utilisées (SOSA/SSN et DOLCE UL), est décrite par la structure globale, illustrée dans la figure 3.7. Cette structure est créée à partir de liens et des relations sémantiques que nous avons identifiés entre les différents modules. Ces relations sont définies dans un premier temps, par la modélisation globale d'un vêtement intelligent et dans un second lieu, selon les scénarios de fabrication et d'exploitation de VI. Nous notons toutefois que dans le cadre du projet SmartSensing, le sous-domaine d'usage étant le sport, des modules sémantiques de

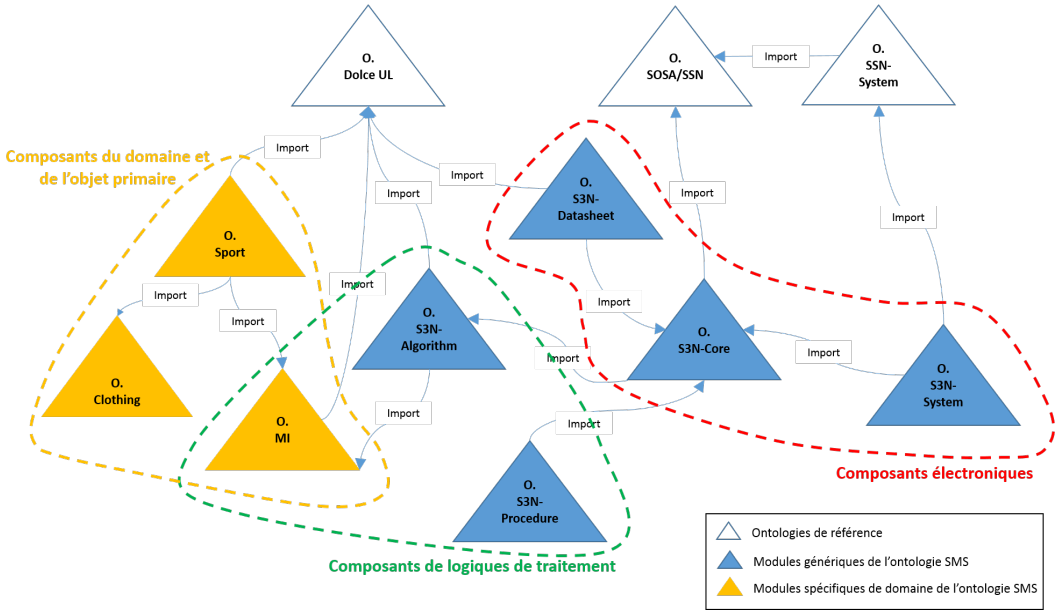

FIGURE 3.7 – Graphe de dépendance entre les modules sémantiques du Framework

domaines liés au sport ont été alors conceptualisés et délivrés comme résultat de l'étape de raffinement des modèles du Framework (c-à-d de la composante sémantique du Framework FSMS).

3.6 La composante Méthodologique

Dans ce chapitre et dans le cadre spécifique du projet SmartSensing, nous avons vu émerger une méthodologie de support au Framework de modélisation sémantique d'objets intelligents. Hormis les activités décrites dans la figure 3.3 et dont le résultat est d'itérer des raffinements successifs à appliquer sur la description structurelle et comportementale de chaque composant d'un Objet Intelligent, la méthodologie suivie préconise l'ensemble des activités décrites dans ce qui suit.

Cette méthodologie est d'autant plus généralisable à la conception/fabrication de n'importe quel OI. Inspirée d'un ensemble de méthodologies en conception logicielle et en conception- développement de plateformes ou d'écosystèmes IdO, celle-ci réutilise leurs même étapes. Nous soulignons cependant, que contrairement aux autres méthodes, celle que nous proposons est fondée sur des modèles ontologiques et en particulier ceux que nous avons identifiés dans la composante sémantique du Framework FSMS. Ceux-ci sont considérés soit comme des inputs aux activités de la méthodologie soit comme des outputs ou des livrables.

Comme nous l'avons précisé précédemment, cette composante méthodologique du Framework FSMS opère à deux niveaux méthodologiques. Le premier est décliné en l'ensemble des étapes suivantes :

— Analyse et planification

— Conception fonctionnelle

— Conception technique

Et le second s'attache davantage à développer les détails des modèles identifiés au niveau un en utilisant une méthodologie plus spécifique dépendant de la nature même des livrables qui ne sont autre que des modèles ontologiques, aspect qui sera détaillé dans le chapitre 4. Dans le reste de la présente section, nous développons le premier niveau méthodologique. Nous soulignons, toutefois, que grâce au Framework FSMS qui encourage la réutilisation et la spécialisation des modèles, il nous est possible d'adopter une approche de type bottom-up.

3.6.1 Analyse et planification

Cette étape concerne tout d'abord l'analyse des besoins des utilisateurs finaux d'un objet intelligent donné. Elle concerne aussi la planification de l'ensemble du projet en termes de temps alloué à chaque étape, de ressources humaines et financières. Elle est composée essentiellement de 5 activités. Nous n'en détaillerons que les deux première en raison de leur forte liaison avec les modèles ontologiques du Framework FSMS comme l'illustre la table 3.1 :

Input	**Activité**	**Output**	**Modèle ontologique**
Acteurs, Désirs, Objectifs	Etude des besoins	Scénarios et Liste des exigences	Modèle d'exigences, Modèle de contexte
Liste des exigences, Questionnaires	Etude et Analyse sur Terrain	Raffinement des exigences, spécification des scénarios	Modèle des exigences raffiné

TABLEAU 3.1 – Etape d'Analyse

3.6.2 Conception fonctionnelle

Cette étape concerne l'élaboration des modèles liés à un OI donné en termes de modules structurels et comportementaux. Elle consiste en particulier à identifier les modèles réutilisables du Framework FSMS ainsi que ceux à développer. Elle comporte les activités illustrées dans la table 3.2 :

3.6.3 Conception technique

Cette étape concerne l'élaboration des modèles liés à la conception et l'agencement des composants électroniques physiques d'un OI donné. Elle s'attache à définir également l'ensemble des Firmwares et des drivers, à embarquer dans les devices et les objets intelligents. . . .
Nous nous focalisons, de plus, au niveau de cette étape sur la décomposition de la logique de traitement à embarquer, en des fragments de code réutilisables, qui une fois installés sur les composants électroniques, comme par exemple au niveau des micro-contrôleurs, puissent collaborer et fournir les fonctionnalités requises.

Input	Activité	Output	Modèle ontologique
Modèle des exigences, Modèle du contexte raffiné	Identification des entités d'un OI	Objet Primaire, Composants électroniques, Logiques de traitement, Domaine	Modèle d'objet, Modèle de capteurs intelligents, modèle de domaine, modèle d'algorithmes
Modèle d'objet, Modèle de capteurs intelligents, Modèle de domaine, Modèle d'algorithmes	Raffinement par adaptation, par réutilisation, par spécialisation		Modèle objet, Modèle d'algorithmes, Modèle de capteurs intelligents, Modèle de domaine raffinés

TABLEAU 3.2 – Etape de conception fonctionnelle

L'ensemble des fragments installés forme la configuration logicielle embarquée qui devrait être gérée pour permettre de futures adaptations et reconfigurations des capteurs intelligents, des objets intelligents qui les intègrent et par conséquent de l'écosystème en général. Cette étape de conception technique comporte les activités illustrées dans la table 3.2 :

Input	Activité	Output	Modèle ontologique
Modèle objet, Modèle de capteurs intelligents, Modèle de domaine raffinés	Agencement des composants électroniques	Objet physique augmenté des composants électroniques	Modèle d'objet composite
Objet physique augmenté des composants électroniques	Développement des firmware et des drivers	Objet physique installable et déployable	Modèle d'OI.
Modèle de domaine raffinés, Modèle d'algorithmes	Décomposition et implémentation de Fragments de code	Fragments de code implémentés	Modèle d'algorithmes implémentés
Objet physique augmenté des composants électroniques, Fragments de code	Déploiement de la logique de traitement	Objet physique intelligent	Modèle de configuration de l'objet intelligent.

TABLEAU 3.3 – Etape de conception technique

3.7 La composante Processus

En vue de développer l'ensemble des processus supportés par le Framework FSMS, nous reconsidérons les scénarios que nous avons décrits dans le chapitre introductif. Nous classifions donc les processus en les différentes catégories suivantes :

— Conception/fabrication

— Réutilisation

— Adaptation

— Reconfiguration

En considérant par exemple le scénario A et en le comparant aux réelles pratiques de l'équipe de conception/fabrication, ce scénario ne représente qu'une possibilité parmi une multitude de scénarios alternatifs qui pourrait être engagés par les personnes intervenantes. Le mode d'organisation du travail et de collaboration entre membres de l'équipe ainsi que l'ordre de priorité des activités ne sont pas du tout formalisés ou écrits dans des documents de travail de l'équipe. Il s'agit d'un mode organisationnel tacite structuré dans les têtes des experts et non décrit explicitement. De plus, chacune des activités des différents processus possibles fait appel au savoir et au savoir faire spécifique au domaine de compétences de chaque intervenant. Les documents et les données échangés ainsi que les résultats obtenus varient également d'un domaine à un autre et sont exprimés dans des formats et des terminologies spécifiques à chaque domaine.

A partir du scénario A, nous pouvons donc identifier un modèle de tâches ou de processus métier à part entière que nous formalisons dans le cadre de la composante Processus du Framework FMSM à l'aide du langage de modélisation de processus BPMN.

De plus, sa description et celle de l'ensemble des processus requis doit tenir compte aussi bien de la composante sémantique que méthodologique du Framework FSMS. Ainsi les tâches ou les activités des processus représenteront les activités de la méthodologie, les entrées et les sorties représenteront les modèles ontologiques.

Processus Conception/fabrication. Ce processus déjà identifié dans le scénario A est illustré par la figure 3.8.

Processus de réutilisation. Ce processus identifié dans le scénario B est illustré par la figure 3.9. Il consiste à rechercher, en fonction d'usages et de profils similaires, des modèles de capteurs intelligents ayant servis précédemment.

Processus d'adaptation/re-configuration. Ce processus identifié dans le scénario C est illustré par la figure 3.10. Il consiste en fonction de profils et d'usage similaires à trouver des logiques de traitement préalablement définies et qui soient adéquates. Le cas échéant, ces logiques sont développées et installés conformément au mécanisme d'adaptation/re-configuration établi.

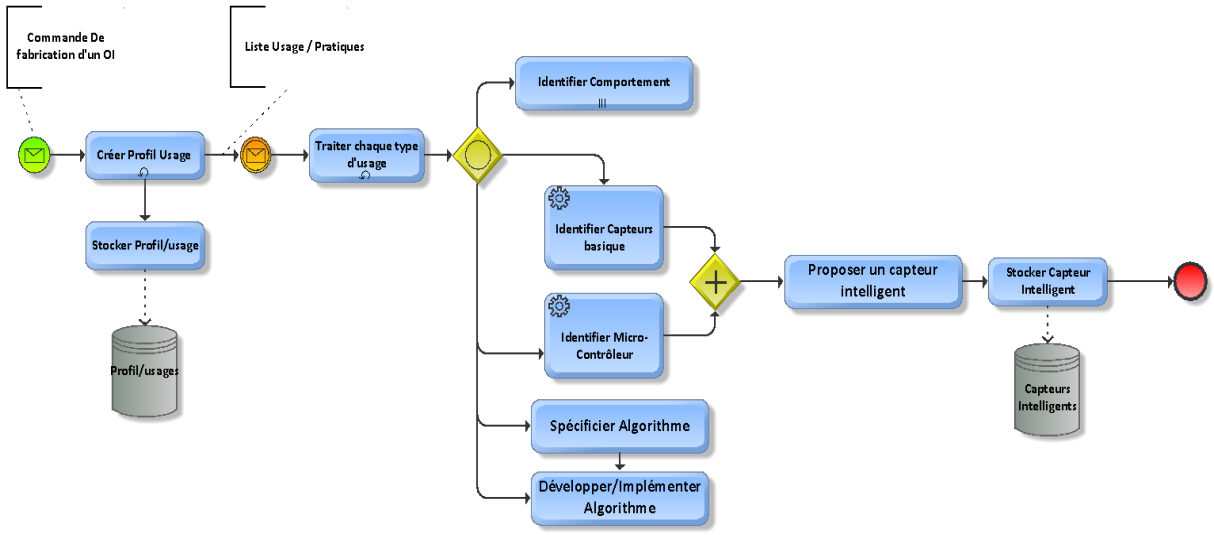

FIGURE 3.8 – Processus de conception/fabrication d'un Objet Intelligent

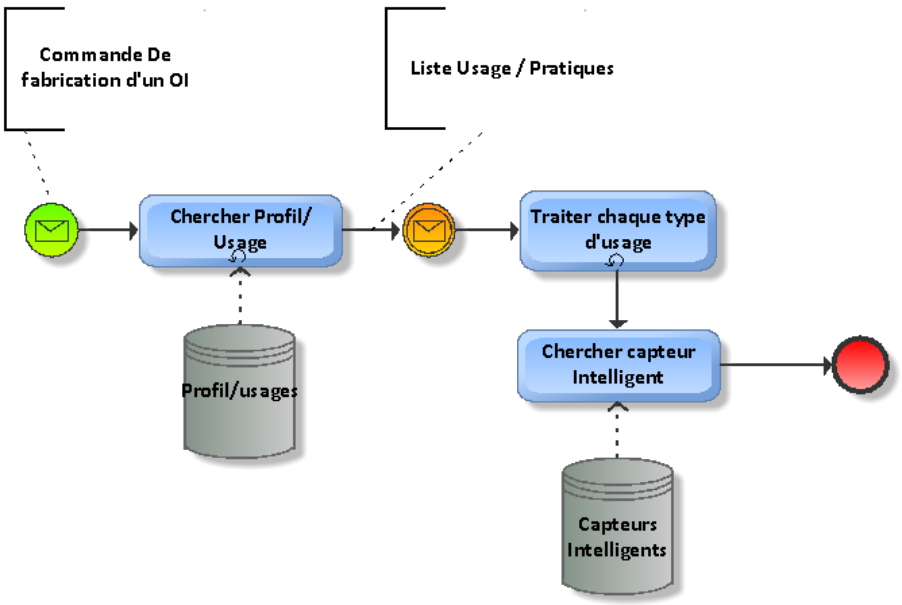

FIGURE 3.9 – Processus de réutilisation de modèles d'un Objet Intelligent

3.8 Conclusion

Dans ce chapitre, une investigation du problème de l'identification et de la représentation sémantique des connaissances multidisciplinaire relatives à la conception/fabrication de VI a été réalisée. Celle-ci a guidé la proposition d'un Framework générique de modélisation sémantique d'objets intelligents (OI) ainsi que l'identification de ses principales composantes. Une méthodologie et un processus générique de conception/fabrication d'OI ont également été proposés. Ceux-ci ont été guidés par l'analyse que nous avons faite de l'ensemble des exigences structurelles et comportementales d'un Objet Intelligent donné. Ces propositions présentent l'avantage d'offrir un cadre méthodologique sémantique et générique qui guide l'ensemble des acteurs concernés par la conception/fabrication d'objets intelligents spécifiques tels que les VI ou les PI. Ce cadre méthodologique sémantique n'a

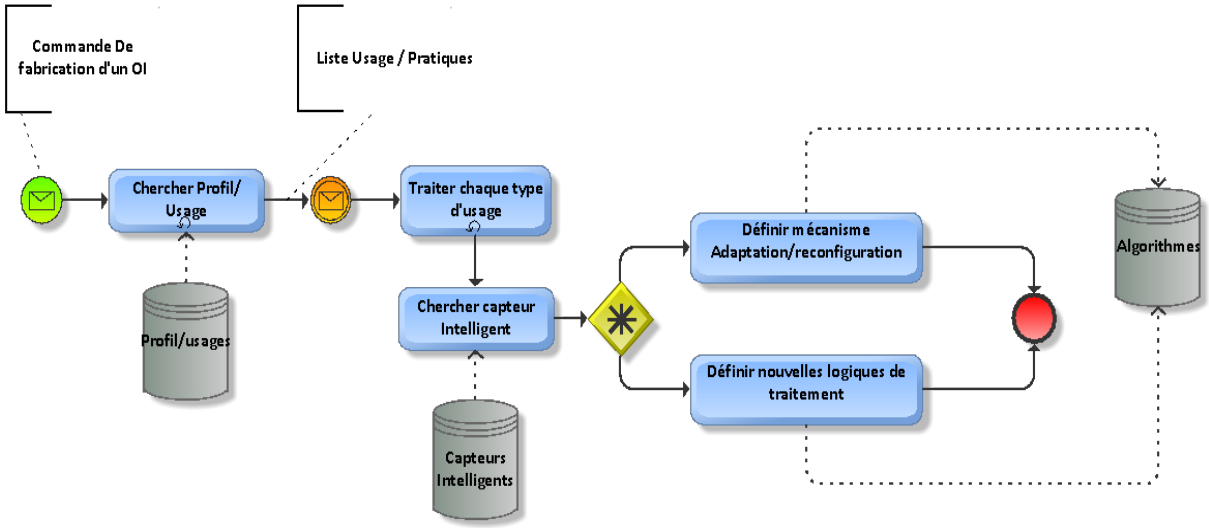

FIGURE 3.10 – Processus d'adaptation/re-configuration d'un Objet Intelligent

cependant pas été entièrement outillé dans le cadre de cette thèse. Néanmoins, son noyau sémantique représenté par l'ontologie SMS a fait l'objet d'un développement complet ; aspect que nous allons détailler dans le chapitre suivant.

Chapitre 4

L'ontologie SMS

Sommaire du présent chapitre

4.1 Introduction

Afin de permettre la mise en place du Framework pour la conception/fabrication et l'exploitation d'un objet intelligent, il est important de spécifier et de conceptualiser les connaissances relatives à ces objets. Ces connaissances appartenant à des domaines variés, il est alors difficile de les cerner et de les organiser. Nous présentons ainsi dans ce chapitre la méthodologie de développement des modèles sémantiques pour le Framework FSMS. Cette méthodologie est une combinaison entre différents scénarios de la méthodologie Neon et des principes de bonnes pratiques pour la modélisation

d'ontologies telle que la modularité, la réutilisation et la séparation des préoccupations.

Suite à la description de notre méthodologie pour le développement d'ontologie multi-domaine, nous l'adoptons et l'appliquons pour développer l'ontologie SMS du Framework FSMS, à savoir l'ontologie SMS. L'ontologie modulaire SMS représente la principale contribution de cette thèse. Elle représente le noyau du Framework FSMS car à la fois générique, modulaire et réutilisable dans différents cas de modélisation sémantique d'OI. Comme le montre la figure 4.1, l'ontologie est logiquement décomposable en trois principaux modules : (1) le module ontologique d'objet primaire ; (2) le module ontologique du domaine d'application ; et (3) le module ontologique pour les composants électroniques. Ce dernier module à un niveau d'abstraction plus élevé que les autres modules. Il est donc indépendant du domaine d'application et de l'objet considéré. Il est réutilisé pour tous les OI, il décrit les concepts liés aux capteurs intelligents et aux traitements logiques embarqués. Ce module, est nommé S3N, il est à son tour formé par un ensemble de module ontologiques décrivant chacun une spécificité particulière d'un composant matériel et/ou logiciel d'un composant électronique. Le module d'objet primaire et celui du domaine d'application sont soit importés ou développés dans le cadre d'une spécialisation du Framework FSMS.

Ce chapitre traite l'ensemble des étapes de construction de l'ontologie SMS. Dans la deuxième section, il décrit tout d'abord la méthodologie suivie à cet effet. Il décrit ensuite, dans la section trois, la spécification et la conceptualisation de chacun des modules formant l'ontologie SMS où l'on considère particulièrement la description du module S3N. La section quatre détaille quant à elle l'implémentation de l'ensemble de ces modules en mettant l'accent sur les décisions prises en vue de couvrir les objectifs visées par l'ontologie SMS. La section cinq permet de conclure le chapitre.

4.2 Principe et méthodologie de conception de l'ontologie SMS

L'analyse des besoins du projet SmartSensing a mis en exergue la diversité des connaissances associées aux différents domaines intervenant dans le projet. Cette diversité entraîne une complexité au niveau conceptuel et organisationnel de ces connaissances. Adopter la "modularisation" comme principe de construction de l'ontologie SMS, consiste à découper les connaissances du domaine en modules, contenant différents types de connaissances [95]. Selon [96] la "modularisation" est un moyen de structurer et d'organiser des ontologies et permet de construire de larges ontologies, fondées sur la combinaison de composantes de connaissances autonomes, indépendantes et réutilisables. Dans le cadre de cette thèse, nous considérons que ces composantes sont elles-mêmes des ontologies, dites "modules", et que l'ontologie résultante de cette composition est une "ontologie modulaire".

Pour la construction de l'ontologie SMS, nous avons donc adopté ce principe de modularisation d'ontologies. Nous avons choisi d'organiser les connaissances d'un OI en différents modules, relatifs aux différentes catégories de connaissances. Chaque module couvre un domaine d'expertise particulier pour la fabrication et l'exploitation d'un OI en général et pour un VI de sport en particulier (capteurs, vêtements, sport, etc.). L'avantage d'une telle structure modulaire est qu'elle offre une certaine

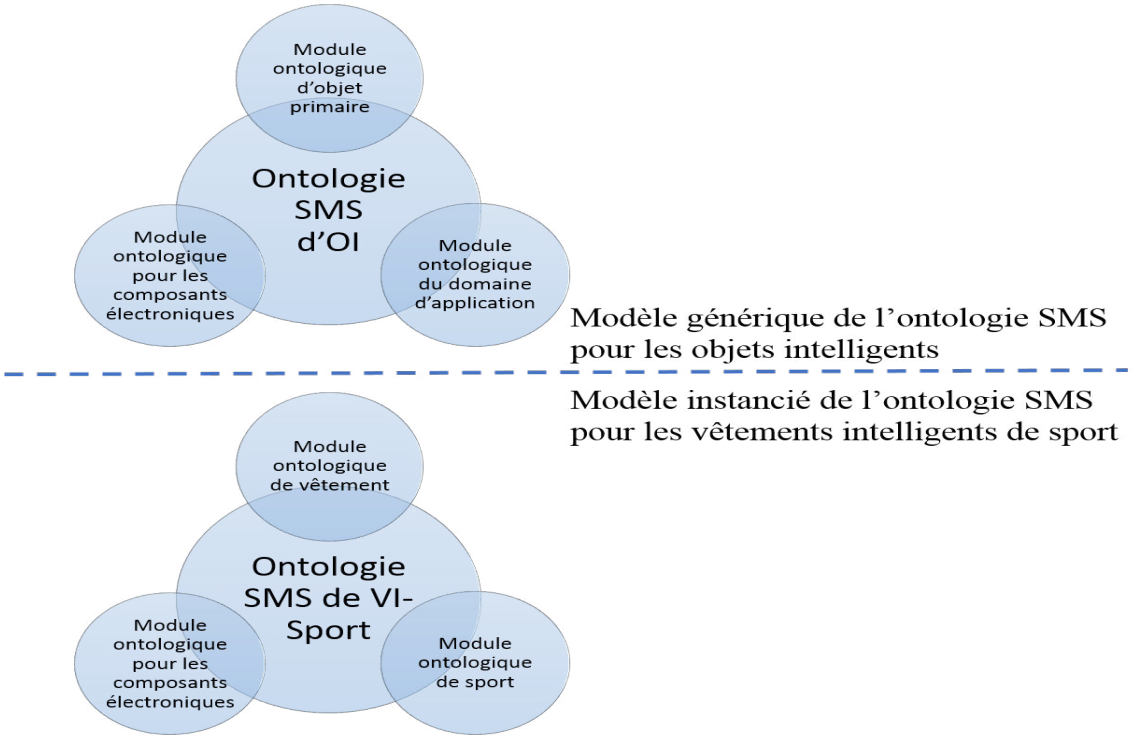

FIGURE 4.1 – Modèle de l'ontologie SMS

indépendance entre les différents modules et laisse la possibilité de les étendre sans remettre en cause la structure globale de l'ontologie modulaire. Ainsi, il est possible de réutiliser les différents modules indépendamment les uns des autres. En changeant le domaine de sport, par exemple, nous réutilisons le VI pour d'autres domaines d'application, comme le militaire, la santé, etc.

La conception modulaire poursuit deux autres objectifs : (1) favoriser leur réutilisation par d'autres ; (2) réutiliser des ontologies de référence (SOSA/SSN, TD, DOLCE Ultra Lite, etc.) pour assurer l'interopérabilité en utilisant l'ontologie SMS notamment dans le cadre de l'IdO.

La réutilisation d'ontologies est une pratique fortement recommandée dans la communauté du Web sémantique [1] [28]. Elle offre la possibilité de : (1) réduire le coût de création d'ontologies, (2) améliorer la qualité des ontologies résultantes et (3) faciliter l'interaction ultérieure entre les systèmes [97].

La réutilisation ontologique peut avoir plusieurs formes ; les ontologies peuvent être référencées, importées, prises comme point de départ pour des extensions, des révisions, etc. Dans [97], les auteurs distinguent trois types de réutilisation d'ontologies : (1) Réutilisation conservatrice où l'ontologie réutilisée reste inchangée. Les concepts, les propriétés ou les individus sont utilisés dans la manière dont ils sont définis dans l'ontologie réutilisée (par exemple, pour définir de nouvelles sous-classes). (2) Réutilisation adaptative où l'ontologie réutilisée fournit un point de départ pour les définitions locales, ce qui peut changer la façon dont les concepts et les propriétés sont définis en fonction des nouveaux objectifs. (3) Réutilisation de la bonne pratique, ceci consiste à réutiliser le savoir-faire, les

1. https ://www.w3.org/TR/ld-bp/

meilleures pratiques et les expériences de construction d'une ontologie.

Le type de réutilisation dépend de plusieurs facteurs tels que le type d'ontologie à construire et de l'ontologie à réutiliser (une ontologie de haut niveau par rapport à une ontologie d'application), de la disponibilité d'ontologies de référence et des besoins de l'ontologie à construire. Dans notre travail, nous avons adopté les trois types de réutilisations. (1) Conservatrice, où nous avons réutilisé l'ontologie de haut niveau DOLCE Ultra Lite par spécialisation ; (2) Adaptative, où nous avons adapté l'ontologie SOSA/SSN, ainsi que l'ontologie TD [98], selon les exigences de l'exploitation et la fabrication d'un OI ; (3) de bonnes pratiques, par la réutilisation du schéma de publication innovant de SEAS [99] afin que l'ontologie soit publié sous forme de différents modules définissant chacun des termes dans le même espace de nom.

De plus, pour créer des ontologies modulaires basées sur la réutilisation, le projet Neon [2] propose un ensemble de scénarios [92]. Dans notre travail, nous avons appliqué les 5 scénarios suivants :

Le scénario 1 (De la spécification à l'implémentation), présentant le processus classique de construction d'ontologies, que nous avons respecté pour la construction de chaque module.

Le scénario 2 (Réutilisation et réingénierie des ressources non-ontologiques), où l'analyse des ressources disponibles, fournies par les différents experts du projet ainsi qu'une investigation au niveau de la littérature, nous ont permis de modéliser les connaissances relatives aux différents modules de l'ontologie SMS.

Le scénario 3 (Réutilisation des ressources ontologiques), a été appliqué pour une réutilisation conservatrice d'ontologies de référence. Au niveau d'abstraction le plus élevé de l'ontologie SMS, nous avons réutilisé l'ontologie DOLCE Ultra Lite. La conception des modules a été fondée sur la spécialisation de cette ontologie.

Le scénario 4 (Réutilisation et réingénierie des ressources ontologiques), a été appliqué pour une réutilisation est une extension de l'ontologie de capteurs SOSA/SSN. L'ontologie TD a également été réutilisée.

Le scénario 8 (Restructuration des ressources ontologiques), a été appliqué pour l'organisation des modules. L'articulation entre ces modules dépend des scénarios d'exploitation et de fabrication d'un OI (ou un VI).

4.3 Spécification et conceptualisation de l'ontologie SMS

L'ontologie SMS est une ontologie modulaire. Elle délimite les différents modules qui sont nécessaires pour la conception/fabrication et l'exploitation d'un OI. Comme défini dans la figure 4.1, l'ontologie SMS est constituée de trois principaux composants structurels décrivant un OI, à savoir l'objet primaire, les composants électroniques (matériel/logique de traitement) et le domaine d'application. Le composant structurel objet primaire et le composant structurel de domaine sont très spécifique et ne seront réutilisables que dans des cas similaires. Par contre le composant structurel

2. http ://www.neon-project.org/

de composants électroniques étant générique du fait, que tout OI intègre ces composants pour avoir des comportements intelligents. Ce module peut être ainsi réutilisé pour la description sémantique de tout objet intelligent. Le module de composants électroniques, nommé S3N (*Semantic Smart Sensor Network*) [3], défini le module générique et principal de l'ontologie SMS. S3N est lui-même une ontologie formée d'un ensemble de modules.

Plus particulièrement, l'ontologie SMS a été développée dans le cadre du projet SmartSensing pour un VI dédié au sport [2]. Dans ce cadre, les Vêtements Intelligents (VI) sont des vêtements ordinaires augmentés par des capteurs mesurant des grandeurs physiologiques ou actimétriques, Ils sont considérés comme des instances d'OI. L'objectif du projet SmartSensing est de concevoir des vêtements intelligents et communicants pour le monitoring et le coaching sportifs. Plus spécifiquement, notre travail s'intéresse à la modélisation sémantique de la fabrication et de l'exploitation des VI. L'intérêt de cette modélisation est de (i) monitorer les sportifs ; (ii) apporter de l'aide à la conception d'un VI ; (iii) réutiliser un VI dans plusieurs activités sportives.

L'objectif de cette section est de présenter l'ontologie modulaire SMS (SMartSensing) créée dans le cadre de ce projet. SMS répond aux différents besoins des phases de fabrication et d'exploitation d'un VI de sport. Elle est constituée de différents domaines de connaissances décrivant un tel vêtement (capteur, vêtement, sport, etc.). Chaque domaine est décrit par un module de l'ontologie. SMS réutilise l'ontologie de haut niveau DOLCE Ultra Lite (UL) ainsi que l'ontologie de capteurs SOSA/SSN et l'ontologie *Thing Description* (TD) [98] recommandées par le W3C pour les projets IdO.

La première partie de cette section sera dédiée à la modélisation de l'ontologie générique S3N comme module de base de l'ontologie SMS, la deuxième partie est consacrée à la modélisation du domaine applicatif à savoir, le sport. Le module de sport a été développé pour valider l'utilisation de l'ontologie SMS au niveau du chapitre 6.

4.3.1 Spécification et conceptualisation du module générique de l'ontologie SMS

S3N est une ontologie modulaire, elle décrit les composants matériels\logiciels d'un capteur intelligent. Cette ontologie est formée de six modules :

— S3N-Algorithm ;

— S3N-Core ;

— S3N-Datasheet ;

— S3N-System ;

— S3N-Procedure ;

— S3N-Thing.

En respectant la méthodologie Neon, nous spécifions pour chaque module les éléments suivants :

— son objectif ;

— ses acteurs cibles ;

— son vocabulaire ;

— et les différentes questions de compétences pour lesquelles le module devrait répondre.

Le module "S3N-Algorithm"

Les logiques de traitement à intégrer dans les objets pour les rendre intelligent et adaptable aux exigences des utilisateurs sont spécifiées via un ensemble d'algorithmes. Un algorithme est le résultat d'une démarche logique (suite d'actions) de résolution d'un problème afin d'obtenir le résultat désiré. Une spécification d'un algorithme permettra ainsi de définir l'ensemble de ses opérations ainsi que les prédicats décrivant l'état initial et final (pré-condition et post-condition) de son exécution. Ainsi, le module S3N-Algorithm a été développé pour définir les logiques de traitement à déployer au niveau des composants électroniques et plus précisément au niveau des capteurs intelligents. Ce module est décrit comme suit :

Objectif. Ce module spécifie, de façon générique, la logique d'un algorithme comme une succession d'opérations à coder. Il décrit principalement les dépendances fonctionnelles et temporelles des données (*dataflow*) de ces algorithmes. Ces dépendances déterminent pour chaque post-condition (le résultat désiré après un traitement), la précondition déterminant l'état des variables avant le traitement (les entrées nécessaires à son calcul).

Acteurs. Les acteurs cibles à utiliser ce module sont : les concepteurs et fabricants d'un OI, principalement les informaticiens et les électroniciens.

Glossaire. Le tableau 4.1 regroupe les différents termes utilisés dans la conception du module S3N-Algorithm.

Termes	**Définitions**
Algorithm	Un algorithme décrit une succession d'opérations qui, si elles sont fidèlement exécutées, produiront le résultat désiré.
Operation	Instruction, étape d'un traitement logique (algorithme).
Snippet	Fragment de code ou de programme informatique, ayant une finalité celle de l'opération qu'il implémente.
SequenceNumber	Numéro de séquence utilisé pour indiquer une place de quelque chose dans une série.
PreCondition	Pré-condition : Un prédicat décrivant l'état initial, des entrées (avant exécution).
PostCondition	Post-condition : Un prédicat décrivant l'état final du résultat désiré (après exécution).
Parameter	Un paramètre peut être n'importe qu'elle caractéristique servant de pré ou post condition d'un algorithme ou de ses opérations.

TABLEAU 4.1 – Glossaire pour le module S3N-Algorithm

Les questions de compétences. Pour ce module nous avons dégagé 5 questions de compétences que

nous résumons dans le tableau 4.2.

N°	Questions de Compétences
QCA1	Pour chaque résultat désiré quel est l'algorithme adéquat ?
QCA2	Pour chaque algorithme, quelle est la succession d'opérations qui, si elles sont fidèlement exécutées, produiront le résultat désiré ?
QCA3	Pour chaque opération, Quels sont les fragments de code exécutables associés ?
QCA4	Pour un algorithme donné, quels est son code exécutable ?
QCA5	Pour chaque algorithme de calcule d'un résultat désiré, quelles sont les entrées nécessaires de la précondition pour le calculer ?

TABLEAU 4.2 – Questions de compétences pour le module S3N-Algorithm

Modèle ontologique de S3N-Algorithm. Pour la conception de ce module nous avons suivi le scénario 2 de la méthodologie Neon, où nous nous sommes appuyés sur les documents (Ressources non ontologiques) fournis par les experts dans le cadre du projet SmartSensing. Nous avons également suivi le scénario 3 où nous avons réutilisé l'ontologie de haut niveau DOLCE UL. L'alignement des concepts du module à ceux de l'ontologie DOLCE UL est montrée au niveau de la figure 4.2.

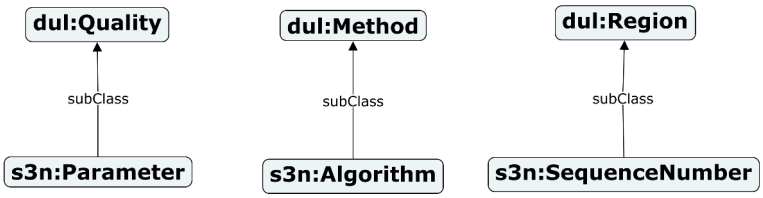

FIGURE 4.2 – Alignement des concepts du module S3N-Algorithm avec Dolce UL

La figure 4.3 illustre le module S3N-Algorithm et les liens sémantiques entre ses différents concepts.

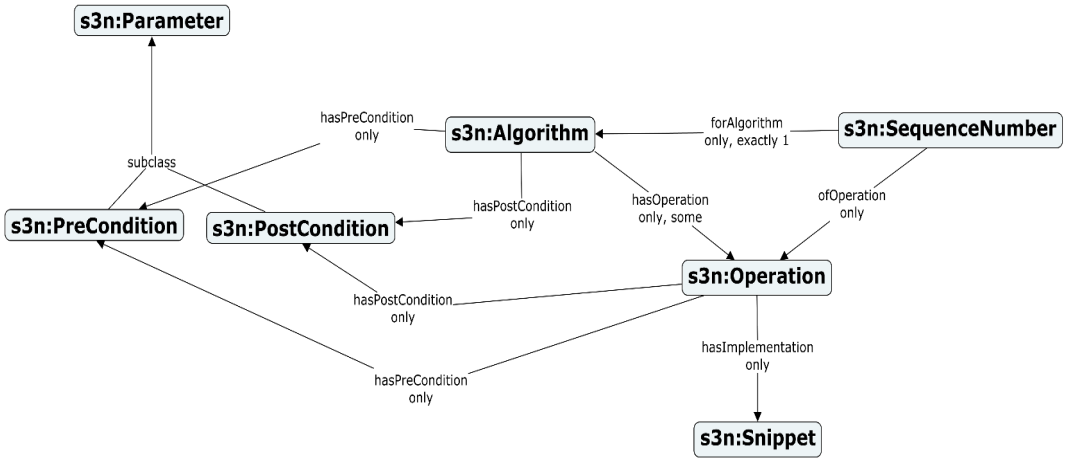

FIGURE 4.3 – Le module S3N-Algorithm

Le module "S3N-Core"

Avant de commencer la conceptualisation du module S3N-Core relatif au concept de Smart-Sensor, il est important de distinguer les capteurs intelligents des capteurs basiques ou classiques. Car certains capteurs basiques disposent déjà de certaines fonctionnalités, comme la détection ou le filtrage, des capacités de paramétrage (configuration de fréquence de rafraîchissement), d'un protocole de communication (I2C) sans pour autant être intelligents. En effet, un capteur basique est un capteur disposant d'une interface (membrane) permettant de détecter un stimulus mais aussi de certaines fonctionnalités basiques indépendantes de l'usage que l'on fait du capteur. Par contre, un capteur intelligent sera doté de fonctionnalités additionnelles paramétrées suivant les besoins du domaine applicatif en question ou encore adaptable suivant le contexte applicatif.

Prenons l'exemple d'un accéléromètre 3 axes (e.g., .un capteur de référentiel LIS2DH). Il permet de donner l'accélération en m/s^2 suivant les axes X, Y et Z ainsi que l'inertie. Ce capteur implémente des algorithmes pour transformer la grandeur physique captée en information numérique exploitable par d'autres algorithmes. En utilisant ce capteur basique tel qu'il est commercialisé, il est impossible de lui ajouter des traitements adaptés au contexte de son exploitation. C'est une sorte de boîte noire qui se limite à fournir des sorties avec des possibilités de paramétrage (comme la sensibilité).

Par contre, dans le projet SmartSensing, le capteur intelligent que nous voulons décrire s'adapte au contexte d'usage en offrant la possibilité d'être programmé et reflashé à chaud. Par exemple, nous considérons deux scénarios d'usage pour deux activités sportives différentes, à savoir le cyclisme et le running. Le SmartSensor utilisé nommé "SMSACTI" peut être programmé selon l'activité sportive exercée. Il peut être ainsi utilisé pour détecter la vitesse de pédalage du cycliste ou s'il est en mode danseuse lorsqu'il s'agit de la pratique du vélo, ou calculer le nombre de pas du sportif et le nombre de calories consommées dans le cas du running. Ces valeurs sont calculées à partir des mesures d'accélération en entrée des algorithmes fournissant ces données. Pour ce faire, notre SmartSensor intègrera un accéléromètre de référence LIS2DH et sera doté d'un micro-contrôleur qui peut être reprogrammé.

Suite à notre investigation théorique faite au niveau du chapitre 3 et celle que nous avons élaborée en collaboration avec les experts du domaine en électronique. Nous avons pu dégager les différences entre un capteur intelligent et un capteur basique, celles-ci sont résumées dans le tableau 4.3. Dans notre travail, nous adoptons la définition suivante pour spécifier les capteurs intelligents :

Le concept de capteur intelligent repose sur sa capacité à (1) acquérir des données grâce à ses capteurs embarqués, (2) traiter ces données grâce à un ou plusieurs algorithmes implémentés par son micro-contrôleur, (3) afficher les valeurs des indicateurs et les communiquer valeurs. Il doit également être reprogrammable et reconfigurable.

Sur la base de cette définition nous conceptualisons le module S3N-Core. Ce dernier est spécifié comme suit :

	héberge	implémente	délivre
capteur basique	-	Procédure unique	Observations
capteur intelligent	un ou plusieurs capteurs de base, un micro-contrôleur, un composant de communication	Plusieurs procédures utilisables dans différents contextes	Exécutions d'algorithmes

TABLEAU 4.3 – Principales différences entre les capteurs de base et les capteurs intelligents

Objectif. Ce module décrit les capteurs intelligents aussi bien au niveau de leur structure physique qu'au niveau de leur adaptabilité fonctionnelle.

Acteurs. Les acteurs cibles à utiliser ce module sont : Les électroniciens ainsi que les utilisateurs traitant des données collectées par les capteurs.

Glossaire. Le tableau 4.4 regroupe les différents termes utilisés dans la conception du module S3N-Core.

Termes	**Définitions**
SmartSensor	Un capteur Intelligent dispose au moins des capacités de traitement, une mémoire et un protocole de communication.
MicroController	Un micro-contrôleur est une unité de calcul.
CommunicatingSystem	Un système de communication est un système permettant la transmission d'information.
AlgorithmExecution	Une exécution d'une succession logique d'opérations (algorithme).
ComputableProperty	Propriété calculable
Error	Valeur erroné d'un résultat.

TABLEAU 4.4 – Glossaire pour le module S3N-Core

Les questions de compétences. Le tableau 4.5 présente quelles que questions de compétence que nous avons dégagée pour le module S3N-Core.

Modèle ontologique de S3N-Core. Pour la conception de ce module nous avons suivi le scénario 2 de la méthodologie Neon, où nous avons utilisé les documents fournis par les experts dans le cadre du projet SmartSensing. Nous avons également réutilisé des ressources ontologiques à savoir l'ontologie SOSA/SSN. Pour ce faire nous avons adopté la démarche du scénario 4 de la méthodologie Neon. Cette réutilisation, illustrée dans la figure 4.4, consiste à une extension de l'ontologie SOSA/SSN pour une prise en compte des caractéristiques des capteurs intelligent et décrire leur comportement adaptable.

Ainsi, le module permet de représenter les connaissances liées aux capteurs intelligents. Un capteur intelligent est un capteur qui doit avoir les mêmes caractéristiques et fonctionnalités de base qu'un capteur basique, plus un ensemble d'instructions ou d'algorithmes capables de faire certains traitements

N°	Questions de Compétences
QCC1	Quels sont les composants du capteur intelligent : ses capteurs basiques, son micro-contrôleur et son composant de communication ?
QCC2	Quel est l'ensemble des algorithmes qu'un micro-contrôleur implémente ?
QCC3	Pour quelle propriété calculable l'algorithme a-t-il généré une valeur ?
QCC4	Pour un résultat donné généré par le micro-contrôleur, quel algorithme a été utilisé ?
QCC5	Si une erreur survient lors de l'exécution d'un algorithme, quelle est la raison et l'origine de cette erreur ?
QCC6	Comment peut-on accéder à la sortie (résultats) du micro-contrôleur ?
QCC7	Quelles sont les résultats retournés par un capteur intelligent donné ?
QCC8	Pour un résultat retourné, quel est sa source (capteur intelligent) ?

TABLEAU 4.5 – Questions de compétences pour le module S3N-Core

en fonction du mode d'utilisation (réutilisation du module S3N-Algorithm). Ainsi, un capteur intelligent dispose d'au moins d'une unité de détection, de capacités de traitement (un micro-contrôleur) avec une mémoire et un protocole de communication. Un micro-contrôleur est défini comme un circuit intégré compact contenant un processeur, de la mémoire et des périphériques d'entrée / sortie (E/S) sur un seul puce, et est conçu pour gérer une opération spécifique dans un système embarqué. Un système de communication est utilisé pour échanger des informations avec un autre système de communication sur un réseau.

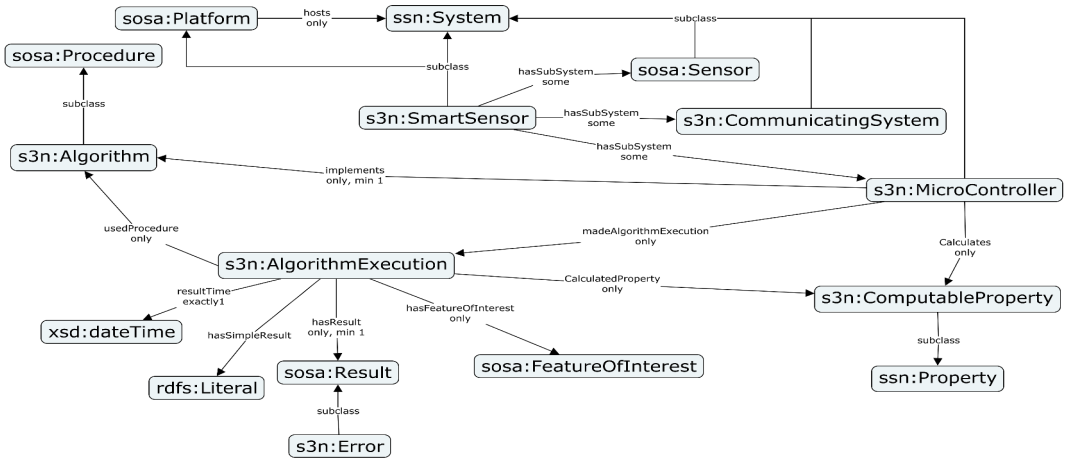

FIGURE 4.4 – Le module S3N-Core

Le module "S3N-Datasheet"

Lorsque les électroniciens veulent utiliser un composant électronique, en particulier un capteur basique à intégrer à un capteur intelligent, le premier réflexe est de se procurer sa documentation et la consulter. La datasheet (ou fiche technique) décrit les propriétés distinctives et essentielles du

composant. Dans le cas de la conception d'un capteur intelligent, les experts du domaine électronique utilisent les datasheets pour sélectionner les capteurs basiques à utiliser et à déployer. Ainsi, nous avons conceptualisé le module S3N-Datasheet dont :

Objectif. Ce module est une spécification des fiches techniques des capteurs et offre une classification de ces derniers. Lors de la conception d'un nouveau capteur intelligent, ce module sert à guider l'utilisateur à sélectionner les capteurs les plus adéquats.

Acteurs. Ce module est destiné à être utilisés par les concepteurs de capteurs intelligents, particulièrement les électroniciens.

Glossaire. Le tableau 4.6 regroupe les différents termes utilisés dans la conception du module S3N-Datasheet.

Termes	**Définitions**
SensorReference	Référence technique d'une datasheet.
ParameterCapability	Indicateur technique dans une datasheet.
Measurand	Grandeur physique à mesurée et qui peut être détectée par le type de capteurs référencés dans la Datasheet.
ParameterCondition	Condition environnemental dans laquelle sont considérées les différentes valeurs d'un paramètre de capacité.
MaxValue	Valeur maximale
MinValue	Valeur minimale
Symbol	Symbole graphique utilisé pour représenter un paramètre de capacité.

TABLEAU 4.6 – Glossaire pour le module S3N-Datasheet

Les questions de compétences. Pour ce module nous avons dégagé 6 questions de compétences que nous résumons dans le tableau 4.7.

N°	**Questions de Compétences**
QCD1	Quelle est le nom de la référence ?
QCD2	Quelles sont les paramètres de capacité de mesure de la référence ?
QCD3	Pour chaque paramètres, qu'elles sont ses valeurs min et max ?
QCD4	Quelle sont la condition de mesure pour chaque plage de valeurs ?
QCD5	Quelle est l'URL du dataseet de la référence en question ?
QCD6	La référence est destinée pour quel type de grandeur physique à mesurer ?

TABLEAU 4.7 – Questions de compétences pour le module S3N-Datasheet

Modèle ontologique de S3N-Datasheet. Pour la conception de ce module nous avons également suivi la même démarche méthodologique que le module S3N-Algorithm. S3N-Datasheet est aligné à

l'ontologie haut niveau DOLCE UL, figure 4.5.

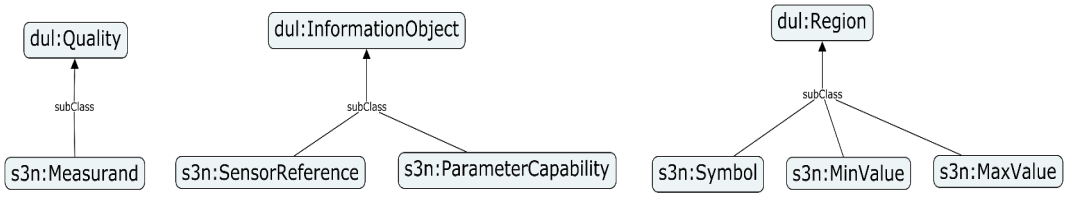

FIGURE 4.5 – Alignement des concepts du module S3N-Datasheet avec Dolce UL

Nous réutilisons également le scénario 4 ou nous réutilisons l'ontologie SOSA/SSN. La figure 4.6 illustre les différents concepts du module S3N-Datasheet.

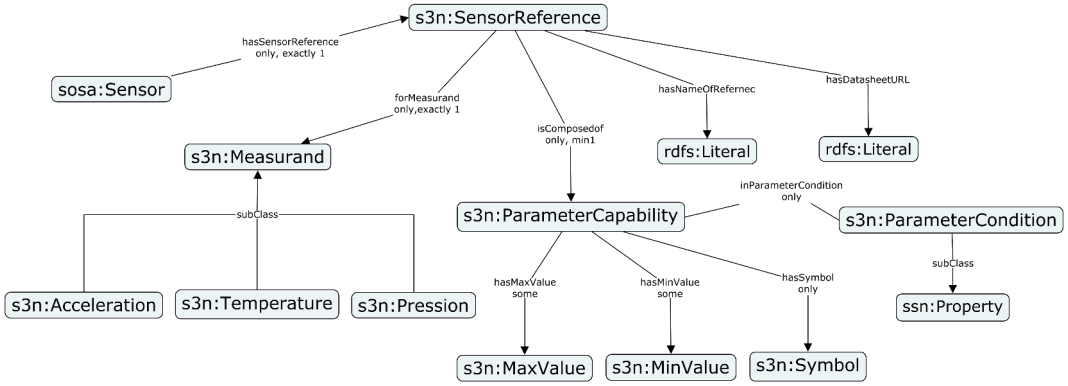

FIGURE 4.6 – Le module S3N-Datasheet

Le module "S3N-System"

Tout composant électronique (capteur basique, micro-contrôleur, etc.) possède un ensemble de capacités caractérisant son fonctionnement. L'ontologie SOSA/SSN au niveau du module SSN-System spécifie un ensemble de ces capacités. Le module S3N-System, que nous spécifions ici, est une extension de SSN-System avec l'ajout de nouveau concepts pour les capacités liées aux composants intégrés dans un capteur intelligent, à savoir le micro-contrôleur et le système de communication. Nous spécifions ainsi le module S3N-System comme suit :

Objectif. Ce module décrit les capacités de fonctionnement liées aux composants déployés au niveau d'un capteur intelligent.

Acteurs. Les acteurs cibles à utiliser ce module sont : Les électroniciens et les développeurs des traitements logiques à installer sur ces périphériques.

Glossaire. Le tableau 4.8 regroupe les différents termes utilisés dans la conception du module S3N-System.

Les questions de compétences. Pour ce module nous avons dégagé les questions de compétences que nous résumons dans le tableau 4.9.

Modèle ontologique de S3N-System. Pour la conception de ce module nous avons suivi le scénario 4 de Neon, ou nous avons étendu le module SSN-System. La figure 4.7 montre cette extension.

Termes	Définitions
Memory	Afin de stocker un programme, des données temporaires ou des données persistantes, le micro-contrôleur dispose de mémoires, ayant des caractéristiques particulières.
MaximumBandwidth	La bande passante maximale désigne le débit binaire maximal d'une voie de transmission.

TABLEAU 4.8 – Glossaire pour le module S3N-System

N°	Questions de Compétences
QCS1	Quelle est la capacité de la mémoire du micro-contrôleur d'un capteur intelligent donné ?
QCS2	Quel est le débit de la bande passante du système de communication d'un capteur intelligent donné ?

TABLEAU 4.9 – Questions de compétences pour le module S3N-System

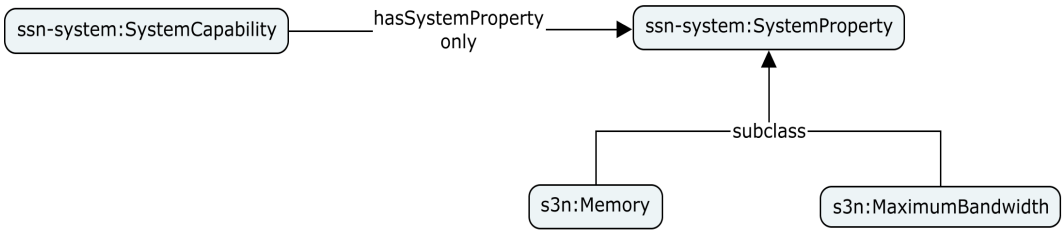

FIGURE 4.7 – Le module S3N-System

Le module "S3N-Procedure"

Pour qu'un traitement logique soit exécuté sur une unité de calcul certaines caractéristiques doivent être prises en considération par les développeurs surtout pour une installation et une reprogrammation de systèmes à contraintes tels que les micro-contrôleurs intégrés dans les capteurs intelligents. Ainsi nous spécifions le module S3N-Procedure comme suit :

Objectif. Ce module décrit les différentes caractéristiques des traitements qui peuvent être implémentés dans un capteur intelligent et spécifiquement au niveau de son micro-contrôleur.

Acteurs. Les électroniciens ainsi que les développeurs informatiques.

Glossaire. Le tableau 4.10 regroupe les différents termes utilisés dans la conception du module S3N-Procedure.

Les questions de compétences. Pour ce module nous présentons dans le tableau 4.11, les types de questions de compétences que lequel module S3N-Procedure a été.

Modèle ontologique de S3N-Procedure. Pour la conception de ce module nous avons réutilisé l'ontologie SOSA/SSN. Ainsi, pour sa conceptualisation nous avons adopté le scénario 3 de Neon tout en s'appuyant sur la documentation des experts du projet, (utilisation du scénario 2). La figure 4.8, illustre ce module.

Termes	Définitions
ProcedureFeature	caractéristique d'une procédure donnée dans des conditions spécifiques.
ProcedureProperty	une propriété déterminant une capacité d'une procédure
ProcedureCondition	une condition pour laquelle une caractéristique d'une procédure est définie
ComputationalCost	le temps nécessaire à l'exécution d'un programme (qui exécute l'algorithme), soit le nombre d'étapes que l'algorithme effectue avant de terminer.
TimeComplexity	temps nécessaire à une fonction de programme pour traiter une entrée donnée.
SpaceComplexity	mesure de la quantité de mémoire de travail nécessaire à un algorithme.

TABLEAU 4.10 – Glossaire pour le module S3N-Procedure

N°	Questions de Compétences
QCP1	Pour un algorithme donné, quel est l'espace de stockage nécessaire pour l'installer ?
QCP2	Quel est la complexité temporelle d'un algorithme ?
QCP3	Quel est la taille en mémoire d'un snippet ?

TABLEAU 4.11 – Questions de compétences pour le module S3N-Procedure

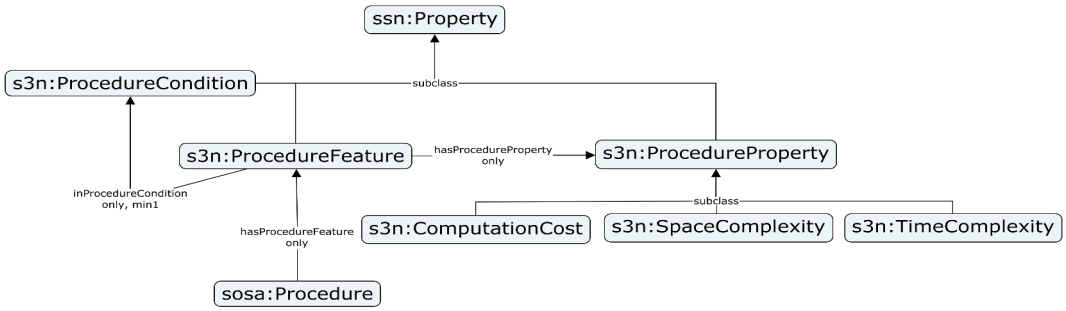

FIGURE 4.8 – Le module S3N-Procedure

Le module "S3N-Thing"

Ce module permet de fournir des descriptions de la manière dont on peut interagir avec les différents composants d'un capteur intelligent. Ce module permet d'aligner des concepts de S3N-Core et de l'ontologie TD. Une description de cet alignement est proposée dans la section 4.4.6.

4.3.2 Spécification et conceptualisation des modules spécifiques de l'ontologie SMS

Les modules spécifiques sont les modules qui décrivent les concepts liés à l'objet primaire et au domaine applicatif. Dans notre cas et dans le cadre du projet SmartSensing l'objet primaire est un vêtement et le domaine est le sport.

Pour la modélisation de vêtements, nous pouvons envisager la réutilisation de l'ontologie modulaire du textile VetiVoc [100], qui fournit un vocabulaire riche pour l'habillement.

Pour la modélisation du domaine applicatif à savoir le sport et les indicateurs de performances associés, nous avons dû développer deux modules : le module "Sport" et le module "Measurement and Indicator". Après un certain nombre d'investigations, nous n'avons pas pu trouver une ontologie de sport répondant à nos besoins. Les ontologies existantes telle que BBC Sport Ontologie [3] ne décrivent pas la vision telle que définie par les expert du projet SmartSensing.

Le module "Sport"

Objectif. Ce module spécifie, de façon générique, les connaissances relatives aux activités sportives. Il permet de décrire les sports et de les classifier (collectif ou individuel). Il est destiné à être utilisé pour relier chaque sport à l'ensemble de ses pratiques et de ses indicateurs. Il offre un modèle pour référencer les sports et décrit les contraintes qui leurs sont associées.

Acteurs. Les acteurs cibles utilisant ce module sont : les sportifs aussi bien amateurs que professionnels, les entraineurs (coach), les concepteurs et fabricants d'un VI ou d'une PI.

Glossaire. Le tableau 4.12 regroupe les différents termes utilisés dans la conception du module Sport.

Termes	**Définitions**
Sport	Tout genre d'exercices ou d'activités physiques ayant pour but la réalisation d'une performance.
CollectiveSport	Sport qui oppose des équipes entre elles.
IndividualSport	Sport qui oppose des individus.
SportingPractice	Un sport peut être exercé sous différentes pratiques tel qu'un entraînement, une compétition, etc.
PhysicalActivity	Ensemble des mouvements corporels produits par la mise en action des muscles squelettiques.
Duration	Espace de temps qui s'écoule entre le début et la fin.
Locale	Espace géographique.

TABLEAU 4.12 – Glossaire pour le module Sport

Les questions de compétences. Pour ce module nous avons dégagé 8 questions de compétences que nous résumons dans le tableau 4.13.

3. https ://www.bbc.co.uk/ontologies/sport

N°	Questions de Compétences
QCSport1	Quelles sont les pratiques d'un sport donné ?
QCSport2	Quels sont les indicateurs de performances pour un sport donné ?
QCSport3	Quels sont les indicateurs de performances pour une pratique dans le cadre d'un sport donné ?
QCSport4	Quelles sont les activités sportives exercées dans le cadre d'une pratique donnée ?
QCSport5	Quelle est la durée d'une activité sportive ?
QCSport6	Quelle est la durée d'une séance de pratique donnée ?
QCSport7	Quels sont les sports individuels ?
QCSport8	Quels sont les sports collectifs ?

TABLEAU 4.13 – Questions de compétences pour le module Sport

Modèle ontologique de Sport. Pour la conception de ce module nous avons suivi le scénario 2 de Neon, où nous nous sommes appuyés sur les documents fournis par les experts dans le cadre du projet SmartSensing. Nous avons également suivi le scénario 3 ou nous avons réutilisé l'ontologie de haut niveau DOLCE UL. L'alignement entre le module "Sport" et l'ontologie Dolce UL est illustré dans la figure 4.9.

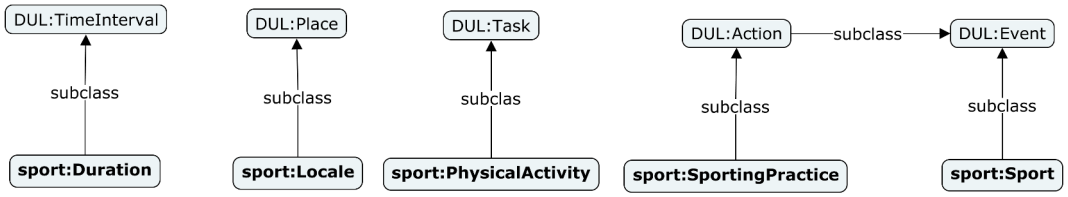

FIGURE 4.9 – Alignement des concepts du module Sport avec Dolce UL

La figure 4.10, montre le modèle conceptuel du module "Sport". Chaque sport a un ensemble de pratiques et à chaque pratique est associé un ensemble d'indicateurs de performances. Les indicateurs de performances sont importés du module " Measurement et Indicator ". Une classification des sports peut étendre ce module par l'ajout des différentes catégories de sports (Running, Tennis, Football, etc.). Nous ne présentons pas dans la Figure 4.10 ces catégories de sports. Ces dernières peuvent être ajoutées comme une spécialisation des concepts "sport :CollectiveSport" ou "sport :IndividualSport". Elles peuvent être également ajoutées comme des instances d'un de ces concepts.

Le module "Measurement et Indicator" (MI)

Objectif. Ce module permet de de donner une classifier des mesures et des indicateurs définis dans le cadre du projet.

Acteurs. Les acteurs cibles à utiliser ce module sont : les sportifs aussi bien amateurs que professionnels, les entraineurs (coach), les concepteurs et fabricants d'un VI ou d'une PI.

Glossaire. Le tableau 4.14 regroupe les différents termes utilisés dans la conception du module MI.

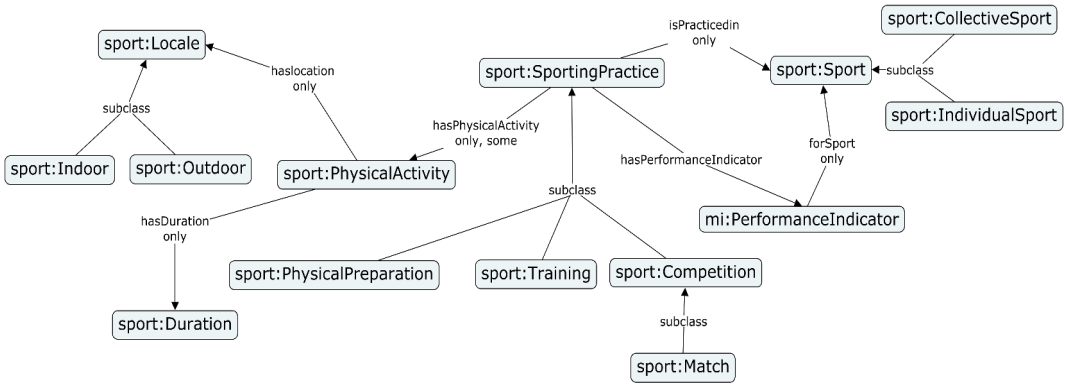

FIGURE 4.10 – Le module Sport

Termes	**Définitions**
PerformanceIndicator	regroupe toutes les qualités mesurées ou calculées.
Indicator	Un indicateur est toute donnée résultant d'un calcule. C'est une dérivée du traitement d'une ou de plusieurs mesures et/ou indicateurs qui fait sens pour une ou des personnes en fonction de son ou de leurs activités. Il sert à évaluer une situation et aide à prendre une décision.
Measurement	Une mesure est toute donnée produite par un capteur suite à une procédure de détection. Toute valeur d'une mesure est décrite dans une unité spécifique. Valeur physique renvoyée par un capteur dans une unité de mesure.

TABLEAU 4.14 – Glossaire pour le module MI

Modèle ontologique MI. Pour la conception de ce module nous avons suivi le scénario 2 de Neon, où nous nous sommes appuyés sur les documents fournis par les experts dans le cadre du projet SmartSensing. Nous avons également suivi le scénario 3 ou nous avons réutilisé l'ontologie de haut niveau DOLCE UL. L'alignement entre le module "MI" et l'ontologie Dolce UL est montré dans la figure 4.11. Une classification des mesures et des indicateurs peut être créée comme une spécialisation de "mi :Measurement" et de "mi :Indicator".

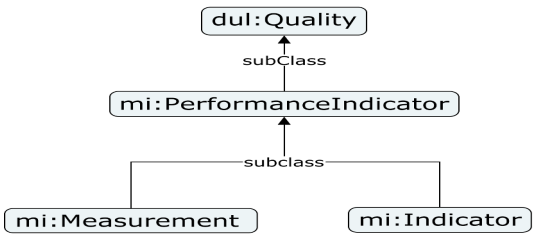

FIGURE 4.11 – Le module Measurement et Indicator

4.4 Développement de l'ontologie SMS

L'objectif de notre travail est de fournir une représentation sémantique permettant l'interprétation et la compréhension des opérations de capteurs intelligents. Dans cette section, nous implémentons la partie générique de l'ontologie SMS, à savoir le module S3N (*Semantic Smart Sensor Network*). Ce dernier est une ontologie basée sur SOSA/SSN et l'ontologie TD pour modéliser les capteurs intelligents. Similaire à l'ontologie SSN et à ses différents modules, S3N est divisé en six modules qui décrivent sémantiquement l'aspect spécifique des capteurs intelligents (*Smart-Sensors*) et répondent aux différents ensembles de questions de compétences. Ces modules peuvent être utilisés (ou importés) séparément ou ensemble, et ils définissent tous des termes dans le même espace de noms ; `http://w3id.org/s3n/`. Ils sont publiés conformément aux meilleures pratiques de publication et de métadonnées en utilisant le système de publication innovant SEAS [99].

Nous raccourcissons l'espace de noms avec `s3n:`.

```
@prefix s3n: <http://w3id.org/s3n/>.
```

4.4.1 Le Module s3n :S3NAlgorithm

Le module **s3n:S3NAlgorithm (S3N Algorithm),** décrit les traitements logiques et a pour URL `http://w3id.org/s3n/S3NAlgorithm`. IL permet de représenter le flux de données et les dépendances entre ces différents données. S3NAlgorithm permet également la description des algorithmes en terme d'ensemble d'opérations. Un s3n:Algorithm est ainsi liée à ses opérations par le lien s3n:hasOperation. Un algorithme est une suite successive bien déterminée d'opérations, ainsi chaque opération (s3n:Operation) ait un ordre bien déterminé pour l'exécution de l'algorithme est l'obtention du résultat attendu. Ce qui implique que pour tout opération a un ordre spécifique pour un algorithme spécifique. Un algorithme est composé d'au moins une opération, une opération peut être réutilisée par plusieurs algorithme (pas d'inverse à la propriété s3n:hasOperation). Mais pour un algorithme, une opération se voit affecter un numéro de séquence de son ordre d'apparition dans le traitement.

$$\text{s3n:Algorithm} \sqsubseteq \forall \text{s3n:hasOperation.s3n:Operation} \tag{4.1}$$

$$\text{s3n:Algorithm} \sqsubseteq \geq 1.\text{s3n:hasOperation.s3n:Operation} \tag{4.2}$$

$$\text{s3n:SequenceNumber} \sqsubseteq \forall \text{s3n:ofOperation.s3n:Operation} \tag{4.3}$$

$$\text{s3n:SequenceNumber} \sqsubseteq \forall \text{s3n:forAlgorithm.s3n:Algorithm} \tag{4.4}$$

$$\text{s3n:SequenceNumber} \sqsubseteq = 1.\text{s3n:forAlgorithm}.\text{s3n:Algorithm} \tag{4.5}$$

$$\text{s3n:forAlgorithm} \circ \text{s3n:hasOperation} \sqsubseteq \text{s3n:ofOperation} \tag{4.6}$$

Les propriétés s3n:ofOperation et s3n:forAlgorithm sont alors définies comme suit :

```
s3n:ofOperation
      schema:domainIncludes s3n:SequenceNumber ;
      schema:rangeIncludes s3n:Operation .

s3n:forAlgorithm
      schema:domainIncludes s3n:SequenceNumber ;
      schema:rangeIncludes s3n:Algorithm .
```

Tout traitement logique est spécifié par ses pré et post conditions que nous implémentons comme suit :

$$\text{s3n:PreCondition} \sqsubseteq \text{s3n:Parameter} \tag{4.7}$$

$$\text{s3n:PostCondition} \sqsubseteq \text{s3n:Parameter} \tag{4.8}$$

$$\text{s3n:PostCondition} \sqcap \text{s3n:PreCondition} \sqsubseteq \bot \tag{4.9}$$

$$\text{s3n:Algorithm} \sqsubseteq \forall \text{ssn:hasPreCondition}.\text{s3n:PreCondition} \tag{4.10}$$

$$\text{s3n:Algorithm} \sqsubseteq \forall \text{ssn:hasPostCondition}.\text{s3n:PostCondition} \tag{4.11}$$

$$\text{s3n:Operation} \sqsubseteq \forall \text{ssn:hasPreCondition}.\text{s3n:PreCondition} \tag{4.12}$$

$$\text{s3n:Operation} \sqsubseteq \forall \text{ssn:hasPostCondition}.\text{s3n:PostCondition} \tag{4.13}$$

D'un autre côté, une opération peut être implémentée par plusieurs codes sources, qui ne sont rien d'autre que des fragments de programme, ils sont représentés par le concept s3n:Snippet.

$$\text{s3n:Operation} \sqsubseteq \forall \text{s3n:hasImplementation}.\text{s3n:Snippet} \tag{4.14}$$

```
s3n:hasImplementation
```

```
schema:domainIncludes s3n:Operation ;
schema:rangeIncludes s3n:Snippet .
```

4.4.2 Le Module s3n :S3NCore

Le module **s3n:S3NCore (S3N Core),** décrit les constituants (composants) d'un capteur intelligent (*Smart-Sensor*) et son adaptabilité. Ce module importe et étend l'ontologie SOSA/SSN et a pour URL `http://w3id.org/s3n/S3NCore`. Nous modélisons la composition des *Smart-Sensors* et leur adaptabilité en termes d'algorithmes qu'ils peuvent exécuter.

Implémentation des composants d'un Capteur Intelligent. Nous décrivons un capteur intelligent comme un système d'assemblage de plusieurs composants électroniques en particulier des capteurs basiques, un micro-contrôleur et un système de communication. Pour la modélisation de l'architecture matérielle d'un capteur intelligent, il est important de noter que l'ontologie SOSA/SSN permet de décrire le concept "capteur" de manière générique. Un capteur est défini comme un dispositif, un agent (y compris les humains) ou un logiciel (simulation) qui réponde à un stimulus, ou des données d'entrée composées à partir des résultats d'observations antérieures, et génèrent un résultat. La description d'un capteur dans SOSA/SSN est définie par le concept sosa:Sensor en tant que sous-classe du concept ssn:System. Cette représentation de capteur donnée par SOSA/SSN répond aux exigences de notre approche pour décrire les capteurs basique. Ainsi, aucun nouveau concept n'est défini pour décrire ces capteurs et le concept sosa:Sensor est parfaitement adapté. Dans l'ontologie SOSA/SSN, un ssn:System est composé de plusieurs sous-systèmes. Un capteur intelligent est un assemblage de plusieurs composants électroniques, en particulier d'un ensemble de sosa:Sensor, d'un s3n:MicroController et d'un s3n:CommunicatingSystem. Pour représenter cette spécificité, l'ontologie proposée étend l'ontologie SOSA/SSN en introduisant les classes pour les trois autres concepts principaux :

s3n:MicroController : est un sous-système d'un capteur intelligent. Il implémente certaines procédures et exécute des ProcedureExecutions. Il permet d'exécuter un ensemble d'algorithmes adaptés au domaine applicatif.

s3n:CommunicatingSystem : est un sous-sytème intégré dans un capteur intelligent pour échanger des informations avec un autre système de communication sur un réseau.

s3n:SmartSensor : un Smart-Sensor est composé d'un ou plusieurs capteurs basiques assemblés à un micro-contrôleur qui implémente différentes procédures et effectue des exécutions de ces procédures sur le résultat des observations faites par ces capteurs basiques pour produire une valeur résultante pour certaines propriétés calculables. Cette valeur peut alors être communiquée par un "CommunicatingSystem".

Un Smart-Sensor héberge différents composants et peut donc être considéré comme a sosa:Platform. En reprenant le schéma axiomatique de SOSA, S3N affirme que le domaine de sosa:hosts inclut également s3n:SmartSensor, tandis que son range comprend également s3n:MicroController, s3n:CommunicatingSystem et s3n:SmartSensor fidèlement au modèle de SSN.

```
sosa:hosts
      schema:domainIncludes s3n:SmartSensor ;
      schema:rangeIncludes s3n:Microcontroller ;
      schema:rangeIncludes s3n:CommunicatingSystem;
      schema:rangeIncludes s3n:SmartSensor .
```

La propriété sosa:ishostsby est l'inverse de sosa:hosts, par la suite nous incluons au domaine de cette propriété les classes de type s3n:MicroController, s3n:CommunicatingSystem et s3n:SmartSensor. Le range de la propriété inclut aussi s3n:SmartSensor.

```
sosa:ishostsby
      schema:domainIncludes s3n:SmartSensor ;
      schema:domainIncludes s3n:Microcontroller ;
      schema:domainIncludes s3n:CommunicatingSystem ;
      schema:rangeIncludes s3n:SmartSensor .
```

La propriété sosa:hosts est utilisée lorsque l'emplacement de l'entité hébergée dépend de celui de l'entité hôte, mais l'entité hébergée n'est pas nécessairement un composant de l'entité hôte. Par exemple, dans le scénario G, Beth peut héberger le capteur intelligent et les quatre capteurs basiques, mais le capteur intelligent héberge uniquement le micro-contrôleur et le composant de communication WiFi.

Exemple :

```
<sensor1> a sosa:Sensor . <sensor2> a sosa:Sensor .
<sensor3> a sosa:Sensor . <sensor4> a sosa:Sensor .
<micro-controller> a s3n:MicroController .
<communicatingcomp> a s3n:CommunicatingSystem .

<Beth> a foaf:Person , sosa:Platform ;
  sosa:hosts <smartsensor>, <sensor1>, <sensor2>, <sensor3>, <sensor4> .

<smartsensor> a s3n:SmartSensor ;
 sosa:hosts <micro-controller>, <communicatingcomp> .
```

Bien que, dans ce scénario G, le capteur intelligent n'héberge pas directement chacun des capteurs basiques, on peut considérer qu'ils sont ses composants avec le micro-contrôleur et le composant de communication (exemple. WiFi), en utilisant la propriété ssn:hasSubSystem. Cette propriété ne peut pas être utilisée pour lier Beth à ces appareils électroniques, sauf si elle est un cyborg.

```
<smartsensor> ssn:hasSubSystem <sensor1>, <sensor2>, <sensor3>, <sensor4>,
    <micro-controller>, <communicatingcomp> .
```

Profitant de l'axiomatisation plus riche de SSN, s3n:SmartSensor est défini comme une sous-classe de s3n:Platform et ssn:System, tandis que les deux concepts s3n:MicroController et s3n:CommunicatingSystem sont définis comme des sous-classes de ssn:System. Enfin, nous modélisons le fait qu'un s3n:SmartSensor contient au moins un de chaque composant. Nous obtenons ainsi les axiomes suivants :

$$\text{s3n:SmartSensor} \sqsubseteq \text{sosa:Platform} \tag{4.15}$$

$$\text{s3n:SmartSensor} \sqsubseteq \text{ssn:System} \tag{4.16}$$

$$\text{s3n:MicroController} \sqsubseteq \text{ssn:System} \tag{4.17}$$

$$\text{s3n:CommunicatingSystem} \sqsubseteq \text{ssn:System} \tag{4.18}$$

$$\text{s3n:SmartSensor} \sqsubseteq \exists \text{ssn:hasSubSystem.sosa:Sensor} \tag{4.19}$$

$$\text{s3n:SmartSensor} \sqsubseteq \exists \text{ssn:hasSubSystem.s3n:MicroController} \tag{4.20}$$

$$\text{s3n:SmartSensor} \sqsubseteq \exists \text{ssn:hasSubSystem.s3n:CommunicatingSystem} \tag{4.21}$$

Les algorithmes et leurs exécutions. Le module S3N-Core de l'ontologie SMS permet également de décrire le fonctionnement d'un capteur intelligent. Pour ce faire, nous remarquons que l'ontologie SOSA/SSN suit des modèles de conception similaires pour la description des sous-types de ssn:System comme les capteurs (sosa:Sensor) qui implémentent des procédures et font des observations, les actionneurs (sosa:Actuator) qui implémentent des procédures et des déclenchements et les échantillonneurs (sosa:Sampler) qui implémentent des procédures et des échantillonnages. Comme nous introduisons de nouveaux sous-types de ssn:System, il est justifié de réutiliser ce même modèle et de modéliser ce que fait le s3n:MicroController.

Contrairement aux capteurs et aux actionneurs qui peuvent implémenter différents types de procédures, s3n:MicroController est spécifiquement conçu pour implémenter des algorithmes liés à la détection ainsi que des algorithmes de calcul spécifique à des cas d'usage qui peuvent être reflashés sur le capteur intelligent pour son adaptation et sa reconfiguration. Nous spécialisons donc sosa:Procedure et proposons l'instanciation du pattern suivant : les s3n:MicroController(s) implémentent des s3n:Algorithm(s) et exécutent des s3n:AlgorithmExecution(s) (utilisation de la propriété objet s3n:madeAlgorithmExecution). Parallèlement à SSN, le module S3N-Core spécifie que s3n:MicroController ne fait que des exécutions d'algorithmes, implémente au moins une chose, et que cette chose est un algorithme. Par contre, nous n'explicitons pas la relation inverse de la propriété ssn:implements, à savoir ssn:implementedBy, entre s3n:Algorithm et s3n:MicroController, car nous supposons qu'un algorithme peut être implémenté sur n'importe quel système (unité de calcule).

L'axiomatisation de ce pattern serait alors :

$$\text{s3n:Algorithm} \sqsubseteq \text{sosa:Procedure} \tag{4.22}$$

$$\text{s3n:MicroController} \sqsubseteq \forall \text{s3n:madeAlgorithmExecution.s3n:AlgorithmExecution} \tag{4.23}$$

$$\text{s3n:MicroController} \sqsubseteq \geq 1 \text{.ssn:implements} \tag{4.24}$$

$$\text{s3n:MicroController} \sqsubseteq \forall \text{ssn:implements.s3n:Algorithm} \tag{4.25}$$

Nous choisissons de ne pas imiter l'ontologie SSN pour définir une relation inverse nommée *madeByMicrocontroller* car d'autres systèmes pourraient faire des s3n:AlgorithmExecution(s). La propriété s3n:madeAlgorithmExecution est alors définie comme suit :

```
s3n:madeAlgorithmExecution
      schema:domainIncludes s3n:MicroController ;
      schema:rangeIncludes s3n:AlgorithmExecution .
```

Le lien entre un s3n:AlgorithmExecution et l'algorithme spécifique qu'il utilise peut être rendu explicite en utilisant la propriété sosa:usedProcedure. Ainsi, nous obtenons :

$$\text{s3n:AlgorithmExecution} \sqsubseteq \forall \text{sosa:usedProcedure.s3n:Algorithm} \tag{4.26}$$

Nous reprenons le schéma axiomatique de SOSA, S3N-Core introduit au domaine de la propriété sosa:usedProcedure le type s3n:AlgorithmExecution et au range le type s3n:Algorithm.

```
s3n:usedProcedure
      schema:domainIncludes s3n:AlgorithmExecution ;
      schema:rangeIncludes s3n:Algorithm .
```

De plus, notons que l'ontologie SSN contient l'axiome 4.27, ce qui implique que pour toute sosa:Observation effectuée par un sosa:Sensor utilise toutes les sosa:Procedure(s) implementées par le capteur. Dans nos cas d'utilisation, un s3n:MicroController peut implémenter différents algorithmes, donc nous ne transposons pas cet axiome aux micro-contrôleurs.

$$\text{sosa:madeBySensor} \circ \text{ssn:implements} \sqsubseteq \text{sosa:usedProcedure} \tag{4.27}$$

Nous définissons également une propriété s3n:forContext, qui est destinée à lier un s3n:Algorithm à son contexte d'utilisation.

```
s3n:forContext
      schema:domainIncludes s3n:Algorithm .
```

Les Caractéristiques d'intérêt et les propriétés. La hauteur d'un arbre ou la température d'une surface sont des exemples de propriétés observables, ce sont des qualités qui peuvent être observées *via* des stimuli par un certain type de capteurs basiques. Ces propriétés sont représentées dans l'ontologie SOSA via le concept sosa:ObservableProperty. Un autre type de propriétés est spécifié dans l'ontologie SOSA est celui des qualités qui peuvent être actionnées par des actionneurs. Ces propriétés sont définies par le biais du concept sosa:ActuatableProperty. C'est propriétés (*Observable* ou *Actuatable*) sont inhérentes aux caractéristiques d'intérêt (sosa:FeatureOfInterest) et ne sont pas indépendants. Les *FeatureOfInterest(s)* sont des entités du monde réel, par exemple, un arbre ou une surface.

Par ailleurs, d'autres propriétés sont liées aux caractéristiques d'interêt, par exemple dans le Scénario F, le nombre de pas calculé pendant la couse d'Abdel. Nous enrichissons ainsi la classe ssn:Property par une sous-classe s3n:ComputableProperty permettant de décrire les propriétés calculables par le micro-contrôleur. L'exécution d'un algorithme par le micro-contrôleur calcule une propriété d'un *FeatureOfInterest*. Un *AlgorithmeExecution* permet de tisser les liens vers un micro-contrôleur pour décrire ce qui a fait l'exécution et comment ; des liens vers un objet *ComputableProperty* pour décrire le résultat du calcul et un objet *FeatureOfInterest* pour détailler à quoi cette propriété était associée.

Nous obtenons ainsi les axiomes suivants :

$$\text{s3n:AlgorithmExecution} \sqsubseteq \forall \text{sosa:hasFeatureOfInterest}.\text{sosa:FeatureOfInterest} \tag{4.28}$$

$$\text{s3n:AlgorithmExecution} \sqsubseteq = 1.\text{sosa:hasFeatureOfInterest}.\text{sosa:FeatureOfInterest} \tag{4.29}$$

$$\text{s3n:AlgorithmExecution} \sqsubseteq \forall \text{s3n:CalculatedProperty}.\text{s3n:ComputableProperty} \tag{4.30}$$

$$\text{s3n:MicroController} \sqsubseteq \forall \text{s3n:Calculates}.\text{s3n:ComputableProperty} \tag{4.31}$$

$$\text{s3n:ComputableProperty} \sqsubseteq \text{ssn:Property} \tag{4.32}$$

$$\text{s3n:ComputableProperty} \sqsubseteq \forall \text{s3n:isCalculatedBy}.\text{s3n:MicroController} \tag{4.33}$$

$$\text{s3n:ComputableProperty} \sqsubseteq \forall \text{s3n:CalculatedProperty}^{-}.\text{s3n:AlgorithmExecution} \tag{4.34}$$

Nous introduisons à la propriété sosa:hasFeatureOfInterest et à son inverse sosa:isFeatureOfInterestOf les objets de type s3n:AlgorithmeExecution, respectivement, au niveau du domaine et du range :

```
s3n:hasFeatureOfInterest
      schema:rangeIncludes s3n:AlgorithmeExecution .
```

```
s3n:isFeatureOfInterestOf
      schema:domainIncludes s3n:AlgorithmeExecution .
```

Le lien entre un *AlgorithmeExecution* et une propriété calculée pour un *FeatureOfInterest* est défini par la propriété s3n:CalculatedProperty.

```
s3n:CalculatedProperty
      schema:domainIncludes s3n:AlgorithmeExecution ;
      schema:rangeIncludes s3n:ComputableProperty .
```

La propriété ssn:forProperty est une relation entre certains aspects d'une entité et une propriété. Exemple, d'un capteur aux propriétés qu'il peut observer ou encore un micro-contrôleur à la *ComputableProperty* qui peut calculer. Nous créons ainsi la propriété s3n:Calculates et son inverse comme suit :

$$\text{s3n:Calculates} \sqsubseteq \text{ssn:forProperty} \tag{4.35}$$

```
s3n:Calculates
      schema:domainIncludes s3n:MicroController ;
      schema:rangeIncludes s3n:ComputableProperty .
```

$$\text{s3n:isCalculatedBy} \equiv \text{s3n:Calculates}^{-} \tag{4.36}$$

```
s3n:isCalculatedBy
      schema:domainIncludes s3n:ComputableProperty ;
      schema:rangeIncludes s3n: MicroController .
```

Les résultats. Enfin, le résultat d'un s3n:AlgorithmExecution peut être un littéral (en utilisant sosa:hasSimpleResult) ou une instance de sosa:Result (en utilisant sosa:hasResult), auquel cas il peut aussi être une instance de s3n:Error et potentiellement a un s3n:hasCause. Une axiomatisation des résultats est :

$$\text{s3n:AlgorithmExecution} \sqsubseteq \forall \text{sosa:hasResult.sosa:Result} \tag{4.37}$$

$$\text{s3n:AlgorithmExecution} \sqsubseteq \geq 1 \text{.sosa:hasResult} \tag{4.38}$$

$$\text{s3n:AlgorithmExecution} \sqsubseteq = 1 \text{.sosa:resultTimet.xsd:dateTime} \tag{4.39}$$

$$\text{s3n:Error} \sqsubseteq \text{sosa:Result} \tag{4.40}$$

Nous reprenons le schéma le schéma axiomatique de SOSA, S3N-Core affirme que le domaine de sosa:hasResult inclut également s3n:AlgorithmExecution, ainsi que le range de la propriété inverse sosa:isResultOf. De même pour les *Data Properties* sosa:resultTimet et sosa:hasSimpleResult dont le domaine inclut s3n:AlgorithmExecution.

```
sosa:hasResult
      schema:domainIncludes s3n:AlgorithmExecution .
sosa:isResultOf
      schema:rangeIncludes s3n:AlgorithmExecution .

sosa:resultTime
      schema:domainIncludes s3n:AlgorithmExecution .
sosa:hasSimpleResult
      schema:domainIncludes s3n:AlgorithmExecution .
```

Nous définissons également une propriété s3n:hasCause, qui est destinée à lier un s3n:Error à la cause de cette erreur.

```
sosa:hasCause
      schema:domainIncludes s3n:Error .
```

Ci-dessous un exemple de Smart-Sensor <smartSensor1> dont le micro-contrôleur a effectué des exécutions de deux algorithmes différents, l'un aboutissant à 17,0 degrés Celsius et l'autre à une erreur.

```
<avgTemp#24H> a s3n:Algorithm . <avgTemp#1Y> a s3n:Algorithm .

<smartSensor1> sosa:hasSubSystem <uc1> .
<uc1> a s3n:MicroController ;
  sosa:implements <avgTemp#24H>, <avgTemp#1Y> ;
  sosa:madeAlgorithmExecution <exec1>, <exec2> .

<exec1> a s3n:AlgorithmExecution ;
  sosa:usedProcedure <avgTemp#24H> ;
  sosa:hasSimpleResult "17.0 DEG"^^cdt:ucum .

<exec2> a s3n:AlgorithmExecution ;
  sosa:usedProcedure <avgTemp#1Y> ;
  sosa:hasResult [ a s3n:Error ; s3n:cause <insufficientMem> ] .
```

4.4.3 Le Module s3n :S3NSystem

Le module S3N-System a pour URL http://w3id.org/s3n/S3NSystem, il importe et étend le module SSN-System. Contrairement à l'ancienne version de SSN qui limitait la portée de la description des capacités à la seule classe sosa:Sensor, la nouvelle version étend la portée à tout objet de type ssn:System. Ainsi,tout ssn:System peut avoir ses capacités décrites dans le nouveau module *SSN Capability* spécifié par SSN-System. Ce module propose une liste prédéfinie de propriétés

systèmes importantes pour les capteurs et les actionneurs, et peut être réutilisée pour d'autres types de systèmes tels que s3n:MicroController et s3n:CommunicatingSystem. Par exemple, la définition de ssn-system:Frequency peut être applicable pour ces systèmes.

Le module S3N-System exploite cette extensibilité et ajoute deux nouvelles propriétés aux systèmes : (1) s3n:Memory pour les s3n:MicroController(s) : La mémoire du micro-contrôleur dans les conditions définies ; et (2) s3n:MaximumBandwidth pour les s3n:CommunicatingSystem(s) : La bande passante maximale de l'équipement communicant dans les conditions définies. Une axiomatisation de cette extension serait la suivante :

$$\text{s3n:Memory} \sqsubseteq \text{ssn-system:SystemProperty} \tag{4.41}$$

$$\text{s3n:Memory} \sqsubseteq \forall \text{ssn-system:hasSystemProperty}^{-}.(\forall \text{ssn-system:hasSystemCapability}^{-}.\text{s3n:MicroController}) \tag{4.42}$$

$$\text{s3n:MaximumBandwidth} \sqsubseteq \text{ssn-system:SystemProperty} \tag{4.43}$$

$$\text{s3n:MaximumBandwidth} \sqsubseteq \forall \text{ssn-system:hasSystemProperty}^{-}.(\forall \text{ssn-system:hasSystemCapability}^{-}.\text{s3n:CommunicatingS} \tag{4.44}$$

4.4.4 Le Module s3n :S3NProcedure

Le module S3N-Procedure a pour URL `http://w3id.org/s3n/S3NProcedure`, et étend SOSA/SSN en ajoutant un module permettant de décrire les caractéristiques des procédures implémentées par les systèmes. La conception du module S3N-Procedure est fortement inspirée par le module SSN-System et S3N-System.

Le module S3N-Procedure est conçu pour décrire les propriétés de procédures telles que la durée, le coût de calcul, le coût de stockage, etc. d'une procédure dans certaines conditions spécifiées telles qu'une taille d'entrée. Cette extension utilise les termes s3n:ProcedureFeature, s3n:hasProcedureFeature, s3n:hasProcedureProperty et s3n:ProcedureProperty. Pour définir un lien entre une entité et une de ses propriétés, l'ontologie SSN définie la propriété ssn:hasProperty (et son inverse ssn:isPropertyOf). Nous définissons les axiomes suivants :

$$\text{sosa:Procedure} \sqsubseteq \forall \text{s3n:hasProcedureFeature}.\text{s3n:ProcedureFeature} \tag{4.45}$$

$$\text{s3n:ProcedureFeature} \sqsubseteq \text{ssn:Property} \tag{4.46}$$

$$\text{s3n:ProcedureProperty} \sqsubseteq \text{ssn:Property} \tag{4.47}$$

$$\text{s3n:ProcedureFeature} \sqsubseteq \forall \text{s3n:hasProcedureProperty}.\text{s3n:ProcedureProperty} \tag{4.48}$$

$$\text{s3n:hasProcedureProperty} \sqsubseteq \text{ssn:hasProperty} \tag{4.49}$$

$$\text{s3n:hasProcedureFeature} \sqsubseteq \text{ssn:hasProperty} \tag{4.50}$$

La classe s3n:ProcedureFeature décrit les caractéristiques normales telles que la complexité, le coût de calcul, etc. d'un processus dans certaines conditions spécifiées telles qu'une exception de calcul. La classe s3n:ProcedureProperty décrit une propriété identifiable et observable qui représente la caractéristique d'un processus à effectuer un résultat attendu. Nous réutilisons la propriété ssn:forProperty pour relier une caractéristique d'une procédure à la propriété pour laquelle la caractéristique a été décrite. Nous définissons ainsi les axiomes suivants :

$$\text{s3n:ProcedureFeature} \sqsubseteq \forall \text{ssn:forProperty}.\text{ssn:Property} \tag{4.51}$$

$$\text{s3n:ProcedureCondition} \sqsubseteq \text{ssn:Property} \tag{4.52}$$

$$\text{s3n:ProcedureFeature} \sqsubseteq \forall \text{s3n:inProcedureCondition}.\text{s3n:ProcedureCondition} \tag{4.53}$$

$$\text{s3n:ProcedureFeature} \sqsubseteq \geq 1.\text{s3n:inProcedureCondition} \tag{4.54}$$

Nous ajoutons ensuite des propriétés de procédures comme sous-classes de s3n:ProcedureProperty. La propriété de procédure s3n:ComputationalCost est proposé pour toute procédure, et les propriétés de procédure s3n:TimeComplexity et s3n:SpaceComplexity sont spécifiquement définies pour les s3n:Algorithm(s).

$$\text{s3n:ComputationalCost} \sqsubseteq \text{s3n:ProcedureProperty} \tag{4.55}$$

$$\text{s3n:TimeComplexity} \sqsubseteq \text{s3n:ProcedureProperty} \tag{4.56}$$

$$\text{s3n:SpaceComplexity} \sqsubseteq \text{s3n:ProcedureProperty} \tag{4.57}$$

$$\text{s3n:SpaceComplexity} \sqsubseteq \forall \text{s3n:hasProcedureProperty}^{-}.(\forall \text{s3n:hasProcedureFeature}^{-}.\text{s3n:Algorithm}) \tag{4.58}$$

$$\text{s3n:TimeComplexity} \sqsubseteq \forall \text{s3n:hasProcedureProperty}^{-}.(\forall \text{s3n:hasProcedureFeature}^{-}.\text{s3n:Algorithm}) \tag{4.59}$$

Ces termes peuvent également être utilisés pour modéliser le fait qu'un algorithme peut avoir des capacités différentes en fonction de l'activité pour laquelle il est utilisé, en utilisant s3n:inProcedureCondition. Une requête SPARQL ASK peut être écrite pour vérifier si la mémoire d'un micro-contrôleur est supérieure à la somme des coûts de calcul de chacune des procédures que l'on veut y charger (les snippets).

4.4.5 Le Module s3n :S3NDatasheet

Le module S3N-Datasheet a pour URL `http://w3id.org/s3n/S3NDatasheet`, et étend SOSA/SSN en ajoutant un module permettant de décrire les fiches techniques des capteurs à embarquer dans un capteur intelligent. Ce module est très important aussi bien durant la phase de fabrication du capteur intelligent que durant la phase de sa réutilisation et sa reconfiguration. Une datasheet est définie pour un type de capteurs pour une mesure particulière. Nous avons défini les axiomes suivants :

$$\text{sosa:Sensor} \sqsubseteq \forall \text{s3n:hasSensorReference.s3n:SensorReference} \tag{4.60}$$

$$\text{sosa:Sensor} \sqsubseteq = 1.\text{s3n:hasSensorReference.s3n:SensorReference} \tag{4.61}$$

$$\text{s3n:SensorReference} \sqsubseteq \forall \text{s3n:hasMeasurand.s3n:Measurand} \tag{4.62}$$

$$\text{s3n:SensorReference} \sqsubseteq = 1.\text{s3n:hasMeasurand.s3n:Measurand} \tag{4.63}$$

Une Datasheet est généralement accessible sur le Web, nous avons ainsi :

$$\text{s3n:SensorReference} \sqsubseteq = 1.\text{s3n:hasNameReference} \tag{4.64}$$

$$\text{s3n:SensorReference} \sqsubseteq = 1.\text{s3n:hasDatasheetURL} \tag{4.65}$$

Une datasheet est une description du fonctionnement du capteur dans certaines conditions, ceci est

décrit par les axiomes suivants :

$$\text{s3n:SensorReference} \sqsubseteq \forall \text{s3n:isComposedof.s3n:ParameterCapability} \tag{4.66}$$

$$\text{s3n:SensorReference} \sqsubseteq \geq 1.\text{s3n:isComposedof.s3n:ParameterCapability} \tag{4.67}$$

$$\text{s3n:ParameterCondition} \sqsubseteq \forall \text{ssn:Proprerty} \tag{4.68}$$

$$\text{s3n:ParameterCapability} \sqsubseteq \forall \text{s3n:inParameterCondition.s3n:ParameterCondition} \tag{4.69}$$

4.4.6 Le Module s3n :S3NThing

Le module S3N Thing a pour URL `http://w3id.org/s3n/S3NThing`, il décrit les interactions possibles avec un Smart-Sensor. Ce module importe l'ontologie S3N-Core et TD. Il contient les alignements proposés entre les concepts SSN et TD et un seul axiome logique supplémentaire définissant s3n:SmartSensor en tant que sous-classe de td:Thing.

De cette façon, le s3n:SmartSensor contient la description de la façon dont on peut interagir avec ses différents composants. L'exemple suivant illustre l'utilisation de l'ontologie TD sur le scénario F : le capteur intelligent de Abdel utilise peut-être (1) un modèle d'interaction de propriété décrivant comment vérifier la liste des algorithmes implémentées par son micro-contrôleur ; (2) un modèle d'interaction d'action pour appairer un nouveau dispositif Bluetooth avec le système de communication ; (3) les modèles d'interaction d'événement pour recevoir les valeurs d'indicateurs et les notifications lorsque la batterie est faible.

Bien que la TD subisse toujours des modifications structurelles et de dénomination, il est probable que son cœur (les éléments ayant des schémas d'interaction décrits) resteront pratiquement inchangés. Dans la version actuelle de TD, (3) peut être partiellement décrit comme suit :

```
<smartsensor01> a s3n:SmartSensor ;
  td:interaction [ a td:Property ;
    rdfs:comment "How to observe the algorithm execution results."@en ;
    td:observable true ;
    td:link [ td:href <wss://192.168.24.75/> ; td:mediaType "application/json"
       ] ] .
```

4.5 Conclusion

Dans de ce chapitre, nous avons présenté l'ontologie modulaire SMS pour la fabrication et l'exploitation d'un Objet Intelligent, en général, et aux vêtements intelligents de sport en particulier. Cette

ontologie représente le noyau du Framework FSMS. Elle permet de représenter les trois composants structurels d'un OI, à savoir le composant d'objet primaire, le composant du domaine d'application et le composant des parties électroniques intégrés (matériels et logiciels). Chaque composant est décrit dans l'ontologie SMS par un module spécifique. Le module ontologique des composants électroniques est générique du fait qu'il est indépendant du domaine d'application et de l'objet considéré. Ce module nommé S3N est ainsi réutilisé pour tout OI. Comme cas particulier d'OI et dans le cadre applicatif des vêtements intelligents de sport, nous avons défini les ontologies de domaine pour le sport et pour la description des indicateurs de performances.

Pour la construction des différents modules de l'ontologie SMS nous avons adopté la méthodologie Neon avec un ensemble de directives de bonnes pratiques. Le développement des modules a été fondé sur la réutilisation de ressources ontologiques de références à savoir, DOLCE UL, TD et SOSA/SSN. Basé sur un cadre méthodologique, nous avons présenté les modules mis en œuvre par le Framework FSMS de modélisation d'OI et leur structuration en termes de relations sémantiques dans l'ontologie SMS. La validation de ces modules fait l'objet du chapitre 6.

Troisième partie

Validation : L’écosystème D-shirt

Cette thèse a abouti à plusieurs propositions théoriques visant à concevoir/fabriquer et exploiter des objets et des systèmes intelligents connectés et ce, *via* le Framework sémantique FSMS fondé sur l'ontologie SMS. Ces propositions ont été mises en œuvre de manière opérationnelle à travers l'écosystème D-SHIRT. Dans cette troisième partie de la thèse, nous entamons l'évaluation de nos contributions théoriques par :

— Une architecture fondée sur l'ontologie SMS appliquée dans le contexte du projet SmartSensing pour supporter le déploiement de vêtements intelligents et reconfigurables. Les détails de l'écosystème D-SHIRT sont présentés au niveau du chapitre 5 ;

— L'exécution de scénarios pour la conception et l'exploitation d'un vêtement intelligent de sport ainsi que pour la reconfiguration/adaptation d'un capteur intelligent. Ces scénarios sont détaillés dans le chapitre 6.

Chapitre 5

Architecture de l'écosystème D-SHIRT

Sommaire du présent chapitre

5.1 Introduction et contexte

Nous avons présenté dans les chapitres précédents notre première contribution qui consiste en un Framework sémantique générique de modélisation d'objets intelligents connectés. Nous avons également décrit en détails les modèles ontologiques que nous considérons comme les éléments constitutifs principaux de ce Framework. Dans cette dernière partie de la thèse, nous présentons notre troisième et quatrième contribution relatives à la spécification et développement d'écosystèmes auto-adaptables et re-configurables en vue de prendre en considération les différents scénarios d'usage. Un écosystème n'est autre que la plateforme et l'application IdO destinées à l'intégration et l'exploitation d'un objet intelligent connecté.

L'écosystème D-SHIRT est relatif au projet SmartSensing. Ce dernier est composé d'un ensemble d'artefacts matériels et logiciels qui ont été définis lors du montage du projet SmartSensing. Chaque partenaire a eu pour mission de piloter ou de participer à la spécification et la réalisation de plusieurs artefacts.

Il est toutefois nécessaire de signaler un fait survenu un an après le démarrage du projet. Au lancement, les partenaires industriels avaient vu et pris conscience de l'intérêt du Web sémantique pour la modélisation des connaissances et des données du projet. Nous les avons sensibilisés à cette technologie *via* des présentations que nous avons assurées pendant les premières réunions de travail. Ils ont adhéré à ce choix et l'ont approuvé. Après la phase d'expression des besoins et l'élaboration du cahier des charges, l'entreprise pilotant le projet s'est opposée à cette orientation et a procédé à un changement stratégique. Elle a abandonné, dans un premier temps, l'idée de constituer une communauté de développeurs autour de ces technologies. Elle a justifié ce choix par la nécessité de garder contrôle des technologies qu'elle va déployer. Elle a aussi abandonné l'approche fondée sur le Web sémantique sous prétexte qu'aucun de ses collaborateurs n'a de compétences dans cette discipline. Elle a argumenté sa décision par son incapacité à trouver le temps et le financement nécessaires pour former ses collaborateurs. Il s'est avéré, même que lors du montage financier du budget, la phase de conception logicielle a été largement minorée. La priorité était devenue de fabriquer un prototype au plus vite afin de faire des démonstrations dans les salons technologiques et trouver des bailleurs de fonds.

Nous commençons donc par présenter une méthodologie générique de conception et développement d'écosystèmes d'objets intelligents connectés ou plus généralement d'écosystèmes IdO. Nous mettons particulièrement l'accent sur la conceptualisation et modélisation des mécanismes d'adaptation et de re-configuration aux contextes d'usage ; étape que nous jugeons fondamentale à la conception développement d'écosystèmes IdO et qui, à notre connaissance, n'a pas du tout été considérée dans les

méthodologies existantes. Nous discutons ensuite les différentes architectures possibles à proposer pour ce type d'écosystème en présentant des modèles alternatifs d'architectures accompagnés d'un ensemble de compromis et de décisions rationnelles guidant à la proposition de notre propre architecture de référence. Afin de valider notre proposition, nous présentons également notre conceptualisation de l'approche de validation, et nous terminons ce chapitre par une conclusion. Les principales questions de recherche auxquelles nous essayons de répondre sont les suivantes :

1. En quoi les approches existantes présentent-elles des limites pour ce type de système ?
2. Comment exploiter les modèles ontologiques voire la base de connaissances élaborée dans le Framework, en vue d'assister et/ou d'automatiser la méthodologie et le processus de conception d'un écosystème IdO ?
3. Quel modèle architectural proposer pour l'écosystème D-SHIRT en vue de favoriser la reconfiguration des objets intelligents intégrés ?

5.2 Vers une méthodologie de conception d'écosystèmes fondée sur le Web sémantique

Différentes méthodes et processus de conception/développement d'écosystèmes existent dans la littérature. Ceux-ci sont fortement inspirés des méthodes et processus classiques de développement logiciel. Les résultats ou livrables de ces méthodes sont essentiellement basés sur les diagrammes de conception classiques comme ceux offerts par UML. De plus certains processus sont itératifs ou suivent un modèle en cascade, d'autres plus récents sont agiles, et d'autres enfin sont agiles mais s'orientent vers un développement/livraison continues. Un processus commun à ces approches est réparti entre les phases « PLAN », « BUILD » et « RUN », dans lesquelles les étapes ou activités suivantes sont réalisées et que nous résumons dans la table 5.1 :

5.2.1 Identification et spécification des buts, des besoins et des exigences

Tout d'abord, il est indispensable de définir les buts et les exigences de l'application IdO. Pour cela nous optons pour une démarche guidée par les scénarios d'usages, le contexte d'usage et le domaine. Des exigences et des risques sont également spécifiés en faisant participer toutes les parties prenantes dans cette activité.

5.2.2 Spécification des processus métiers ou logique d'exécution de l'application

Pour chaque scénario d'usage, et pour chacun des cas d'utilisation spécifiques identifiés, un ou plusieurs processus métiers sont spécifiés pour guider le développement des interactions entre les

différentes entités de l'application ou de la plateforme IdO ainsi que toute la logique de traitement requises.

5.2.3 Spécification des modèles du Domaine

Dans cette étape touts les concepts, les entités physiques ou virtuelles (logiques) seront décrits avec leurs caractéristiques, propriétés, attributs, comportements (opérations) et relations. Ces modèles incluent également les périphériques (devices), objets, ainsi que les services et les ressources requises pour élaborer le comportement global de l'application et de la plateforme IdO.

5.2.4 Spécification des modèles d'information

Cette étape s'attache à définir les modèles d'information pour caractériser les flux de données et les données véhiculées dans le système. Elle a pour but de préciser davantage les attributs de chaque entité, leurs types et voire leurs plages de valeurs.

5.2.5 Spécification des Services

Dans cette étape, une spécification détaillée des services offerts par l'application ou la plateforme est proposée. Cette spécification permet de dresser une hiérarchie de services, leurs types (services d'infrastructures ou services fonctionnels), leur Input/Outputs, leurs points d'entrée (ou endpoints), leurs protocoles, les orchestrations/chorégraphies de services si elles existent, les pré et post-conditions et leurs effets sur les états du système.

5.2.6 Spécification de la vue fonctionnelle

Dans cette étape un regroupement des fonctionnalités de l'écosystème ou de l'application est réalisé, en vue de réaliser une correspondance entre ces groupes fonctionnels et l'architecture logique du système. Cette spécification permet également de préciser les fonctionnalités d'interaction avec les entités spécifiées dans le modèle du domaine (entités virtuelles et physiques, services, ressources, devices, etc).

5.2.7 Spécification du niveau ainsi que de l'architecture de déploiement de l'écosystème IdO

Cette étape concerne la spécification de l'architecture physique de déploiement des différents éléments de l'écosystème.

5.2.8 Spécification de la vue opérationnelle (ou d'opérationnalisation)

Cette étape spécifie comment sera développé chaque groupe fonctionnel avec quelle technologies et avec quelles plateformes de déploiement ou d'exécution comme par exemple le choix des middlewares d'intégration ou ceux de services Web, le choix des systèmes de gestion de données, leurs administration, etc.

5.2.9 Spécification de l'intégration de composants électroniques dans les devices

Cette étape concerne la définition technique du device et de la manière dont il va intégrer les composants physiques comme les capteurs, les cartes de communication etc.

5.2.10 Développement de l'Application d'Interaction Homme Écosystème

Il s'agit ici surtout de définir l'interface qui permettre à l'utilisateur d'interagir avec l'écosystème. C'est un logiciel qui pourrait être embarqué dans son SmartPhone ou Smart Watch.

5.3 Enrichissement sémantique de la méthode et processus de conception

Bien que la méthodologie et le processus présentés précédemment semblent supporter les concepteurs et leur servent de guide pratique, le problème terminologique entre les différentes parties prenantes responsables de la conception de l'Ecosystème d'intégration reste entier.

De plus cette méthodologie préconise la définition de modèles d'exigences, de scénarios d'usages, de domaines, d'information, de logiques de traitements, et même d'objets intelligents connectés en partant de zéro et qu'aucun modèle ou méta-modèle n'est utilisée en guise de support d'appui ou de patron réutilisable. De plus, ces étapes de conception, même dans le cas où elles sont appliquées correctement par des personnes expertes, elles peuvent omettre de considérer certains cas d'usage, ou certaines logiques de traitement possibles.

Par ailleurs dans le souci d'appliquer le principe de découplage entre la spécification et le développement d'un objet intelligent ou de l'Ecosystème qui l'intègre, et leurs exploitations et usages, nous avons élaboré un Framework sémantique générique à part entière pour la modélisation des objets intelligents connectés. Nous notons également qu'il existe une forte synergie et interaction entre les modèles ontologiques décrits dans le Framework et le rôle et la sémantique des modèles de conception préconisés par la méthodologie de conception de l'écosystème ou de la plateforme IdO (voir figure5.1. De plus, mise à part que ces modèles ontologiques permettent de définir ou de remplacer les modèles de conception de la méthodologie, leur aptitude à supporter des mécanismes de raisonnement et d'inférences, permet d'inférer automatiquement des éléments omis ou non considérés par les concepteurs.

Etape	**Livrable**	**Exemples d'éléments**
Identification et spécification des buts, des besoins et des exigences	Diagrammes de uses case, taxonomies de buts, etc	Collecte et analyse de données, Gestion du système, Reconfiguration au design time , Reconfiguration au runtime, Déploiement local de l'application, Accessible pour des utilisateurs enregistrés
Spécification des processus métiers ou logique d'exécution de l'application	Cas d'utilisation, diagrammes d'activités, processus métiers (en BPMN, ou autre), algorithmes, etc	Capture d'une mesure, Conversion et envoie d'une mesure
Spécification des modèles du Domaine	Diagrammes d'entité associations, diagrammes de classes UML, Diagramme de services incluant les interfaces de services,	Entités physiques : Objet intelligent,Capteur, Entités logiques : représentation des objets physiques, etc
Spécification des modèles d'information	Diagrammes d'entité associations, diagrammes de classes UML, Diagramme de services incluant les interfaces de services,	Fréquence de mesure d'un capteur, Valeurs et unités, etc
Spécification de la vue fonctionnelle	Schémas d'architectures fonctionnelles	Groupe fonctionnels de sécurité, Groupe fonctionnel de gestion, Groupe fonctionnel de gestion de données, etc
Spécification du niveau ainsi que de l'architecture de déploiement de l'écosystème IdO	Diagrammes de déploiement, Modèles d'architecture technique	Serveurs, Caractéristiques techniques, Protocoles, Devices, Capteurs
Spécification de la vue opérationnelle (ou d'opérationnalisation)	Manuels de configuration, d'administration	Services Web de type Soap, Protocole http, etc
Spécification de l'intégration de composants électroniques dans les devices	Schéma techniques de devices et de capteurs, etc	Capteur, Carte Arduino, etc
Développement de l'Application d'Interaction Homme Ecosystème	Maquettes d'écrans, Schémas d'interaction	Dialogue en langage naturel, Interface graphique, etc

TABLEAU 5.1 – Etapes de la méthodologie de conception

Notre vision étant donc d'enrichir les étapes de la méthodologie de conception de l'écosystème par l'ensemble des modèles ontologiques, en vue d'une génération semi-automatique de l'ensemble des livrables de la méthodologie. Nous résumons cet enrichissement sémantique au niveau de la table 5.2.

Nous montrons ainsi le rôle des modèles ontologiques dans la proposition d'une approche séman-

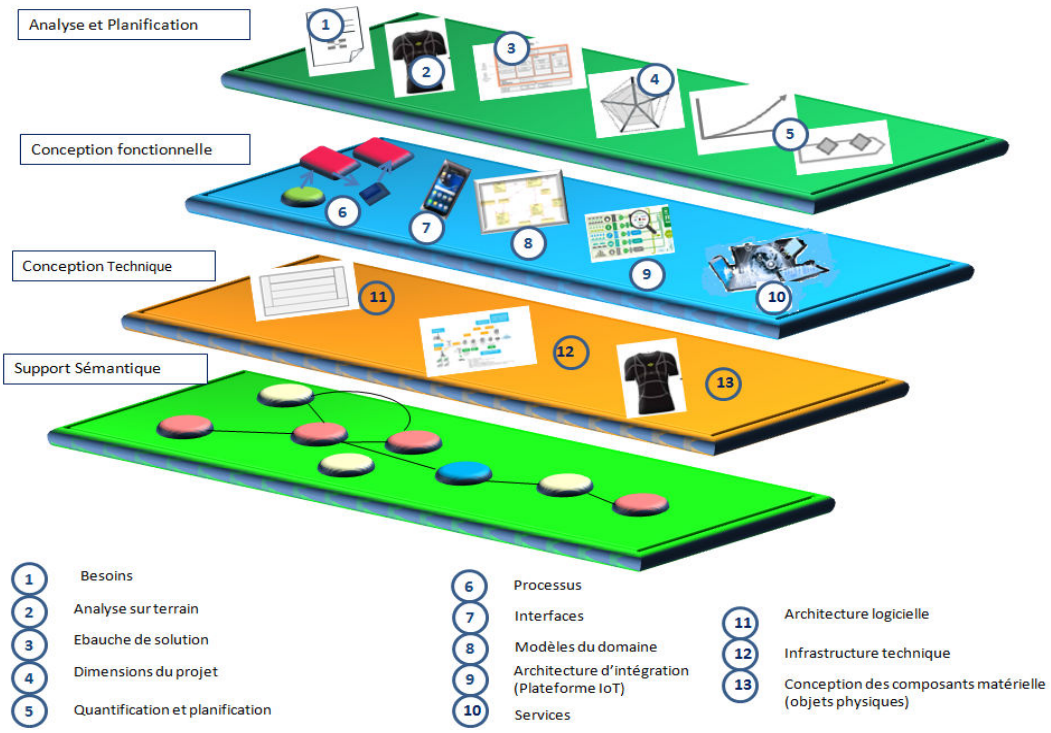

FIGURE 5.1 – Méthodologie de conception de l'écosystème

tique de conception d'écosystèmes IdO. En effet, l'identification des scénarios initiaux qui ont guidé le développement du framework sémantique de modélisation peuvent être réutilisés pour inférer de nouveaux scénarios d'usages à partir du domaine, des mesures et indicateurs et du VI ou PI utilisé. Ces scénarios recommandés par le système peuvent être améliorés validés, etc et enrichir une base de scénarios d'usages. Il est d'autant possible sémantiquement d'opérer des fusions et/ou extensions des modèles de domaines, des profils utilisateurs et des mesures et indicateurs en vue d'inférer des modèles d'usage non prévus initialement par les concepteurs. Cela permettra de générer de nouvelles possibilités quant à l'amélioration de l'expérience de l'utilisateur. De plus la gestion sémantique des scénarios d'usage permettra non seulement de réutiliser ce qui a été conçu dans d'autres contextes mais aussi de contribuer à cibler des communautés de personnes partageant les mêmes intérêts et les mêmes préoccupations. En fin d'autres modèles sémantiques peuvent être définis et supporter les autres étapes du processus de conception comme par exemple les ontologies de services ou celles liées aux architectures logicielles ou encore celles liées aux Interactions Homme Machines.

5.4 Proposition d'une architecture générique et adaptable de l'Ecosystème D-SHIRT

Dans le but de définir l'architecture de l'écosystème, nous nous basons sur une architecture de référence établie à partir d'une étude de l'art réalisée dans [101] Cette étude montre que bien

Etape	**Livrable**	**Enrichissement sémantique**
Identification et spécification des buts, des besoins et des exigences	Diagrammes de uses case, taxonomies de buts, etc	Inférer de nouveaux scénarios d'usage à partir du domaine et des usages définis dans le framework, Evolution automatique des usages, Inférer des exigences à partir du domaine
Spécification des processus métiers ou logique d'exécution de l'application	Cas d'utilisation, diagrammes d'activités, processus métiers (en BPMN, ou autre), algorithmes, etc	Inférer les logiques de traitement à partir du domaine, Composition sémantique de fragments de codes atomiques
Spécification des modèles du Domaine	Diagrammes d'entité associations, diagrammes de classes UML, Diagramme de services incluant les interfaces de services,	Les ontologies de domaines, Les ontologies de services (OWL-S, WSMO)
Spécification des modèles d'information	Diagrammes d'entité associations, diagrammes de classes UML, Diagramme de services incluant les interfaces de services,	Les propriétés de données (Data properties), Les instances ou triplet RDF
Spécification de la vue fonctionnelle	Schémas d'architectures fonctionnelles	Groupe fonctionnels de sécurité, Groupe fonctionnel de gestion, Groupe fonctionnel de gestion de données, etc
Spécification du niveau ainsi que de l'architecture de déploiement de l'écosystème IdO	Diagrammes de déploiement, Modèles d'architecture technique	Il est possible de prévoir des modèles sémantiques d'architectures (en dehors du contexte de ce travail)
Spécification de la vue opérationnelle (ou d'opérationnalisation)	Manuels de configuration, d'administration	Services Web de type Soap, Protocole http, etc
Spécification de l'intégration de composants électroniques dans les devices	Schéma techniques de devices et de capteurs, etc	Ontologie S3N, Ontologie SMS
Développement de l'Application d'Interaction Homme Ecosystème	Maquettes d'écrans, Schémas d'interaction	Il est possible de prévoir des modèles sémantiques d'IHM (en dehors du contexte de ce travail)

TABLEAU 5.2 – Etapes de la méthodologie de conception sémantiquement enrichie

que fonctionnellement ces architectures offrent pratiquement des fonctionnalités équivalentes, elles s'appuient sur des technologies différentes. Elles utilisent également des terminologies différentes et distribuent ou déploient les composants fonctionnels, matériels ou logiciels de manières aussi différentes.

Certaines sont orientées tout juste vers un traitement local et donc elles sont entièrement embarquées, d'autres distribuent les fonctionnalités entre un client et un serveur, d'autres encore sont entièrement distribuées et adoptent une vision de sytème de systèmes (system of systems). Un choix de technologies, ensuite un choix de l'architecture s'avèrent nécessaires. Plusieurs alternatives peuvent se présenter, des compromis et des raisons rationnelles permettront de favoriser une architecture par rapport aux autres. Selon [101], plusieurs architectures existantes offrent des vues détaillées des plateformes IdO. Plus les architectures sont détaillées, plus elles gagnent globalement en termes d'hétérogénéité.

Une architecture de référence abstraite comme l'illustre la figure 5.2 peut servir comme source de connaissances de base ou minimale pour s'en inspirer à concevoir et développer d'autres architectures. Cette architecture permet de définir entre autres les différents éléments qui la composent ainsi que leurs rôles respectifs.

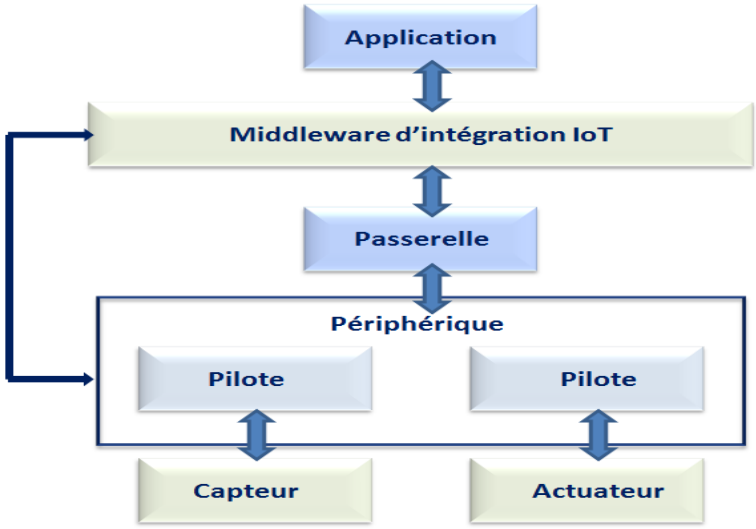

FIGURE 5.2 – Architecture de référence d'une Plateforme IdO

Un ensemble de règles régit la composition de cette architecture de référence : Si l'objectif du système (ou écosystème) est uniquement de mesurer des paramètres de l'environnement physique, alors le système sera formé uniquement de **capteurs**. Le cas échéant il pourra aussi comporter des **actionneurs**.

Si nécessaire, un capteur peut être configuré au moyen d'un logiciel mais il serait incapable d'exécuter lui-même un logiciel sauf pour le cas où ce capteur respecte la définition que nous avons spécifié pour un capteur intelligent (Smart sensor).

En général les capteurs sont connectés à ou sont intégrés dans un **périphérique**, auquel les données collectées sont envoyées. Comme exemples de périphériques on peut citer une carte RaspBerry Pi ou Arduino. On peut aussi considérer un smart phone comme étant un périphérique auquel des capteurs peuvent être connectés. La communication entre le capteur et le périphérique pourrait être filaire ou sans fil (c-à-d. via des émissions réceptions radio).

Pour traiter les données ou pour contrôler les actionneurs, des **pilotes** (Drivers en Anglais) sont nécessaires. Il s'agit d'un type particulier de logiciel permettant de jouer le rôle d'interface ou de middleware entre le composant physique (capteur ou actionneurs) et les applications du système. Donc c'est grâce aux pilotes que d'autres éléments logiciels peuvent communiquer avec les capteurs et les actionneurs. Le périphérique est considéré comme étant le point d'entrée de l'environnement physique au monde virtuel. Ces périphériques peuvent être soit autonomes et donc ils embraquent toute la logique de l'application soit ils sont connectés à d'autres systèmes d'où le rôle du **middleware d'intégration**.

La passerelle (Gateway en Anglais) joue le rôle d'intermédiaire entre les périphériques et d'autres systèmes surtout si ces devices n'offrent pas de possibilités ou de protocoles de communication avec d'autres systèmes. Elle supporte des technologies de communication bi-directionnelles depuis les périphériques vers d'autres systèmes et vice versa. En général, la passerelle transforme le message reçu à partir d'un protocole IdO à un format commun comme XML ou Json. Elle permet aussi de traduire les commandes venant d'autres systèmes vers le format de commandes des capteurs ou des actionneurs. Enfin, elle est capable d'embarquer certaines logiques de traitement et peut jouer même le rôle d'une unité de traitement à part entière.

Le middleware d'intégration IdO est responsable en général de la médiation des données reçues à partir des périphériques connectés et leurs routages vers les applications concernées ou les périphériques de contrôle. Il est en général orienté messages et implémente le modèle du "Publish and subscribe" assurant ainsi un faible couplage entre les capteurs et les applications ou les périphériques. Ces derniers peuvent communiquer directement avec le middleware ou alors ils le font à travers la passerelle. Le middleware sert donc surtout comme une couche d'intégration aux différents périphérique, capteurs, acteurs et applications et intègre en conséquence toute la logique d'interopérabilité (technique, syntaxique et également sémantique). Cette couche d'intégration peut également exposer son API comme étant des services Web REST ou SOAP permettant ainsi une interopérabilité accrue avec d'autres systèmes hétérogènes.

Application représente le logiciel qui va utiliser le middleware d'intégration comme un point d'entrée à l'environnement physique de l'écosystème. Elle se charge d'envoyer des requêtes de demandes de données ou de commandes d'actionneurs. Elle peut dans certains cas embarquer le puits de données (Well en Anglais). Ce composant sera présenté dans l'architecture D-SHIRT.

Cette architecture de référence que nous venons de présenter peut être appariée avec n'importe quelle plateforme IdO sauf que dans certains cas un composant de celle-ci soit représenté par plusieurs autres composants de l'architecture et vice versa. De plus le concept de device, de Gateway ou de middleware d'intégration peuvent changer de rôle ou de manière d'être déployé.

5.5 L'écosystème D-SHIRT comme plateforme réutilisable et générique d'intégration de VI et d'algorithmes

Dans cette section, nous présentons le système D-SHIRT en tant que plateforme IdO. Son architecture matérielle et logicielle est commune aussi bien pour un usage amateur que professionnel. La différence pour les professionnels, est située à deux niveaux. Premièrement, au niveau du choix des composants électroniques qui devraient présenter de meilleures performances et de meilleures précisions avec des logiques de traitement (algorithmes) à granularité plus fine et des services additionnels. Deuxièmement, au niveau de l'architecture de la solution et des usages qui sont complètement différents. Pour être plus explicite, dans un usage amateur, les besoins de monitoring des activités sportives sont destinés exclusivement au sportif. Les indicateurs sont affichés sur le smartphone du sportif ou sur tout autre dispositif assimilé qu'il porte telle qu'une montre connectée. Par contre, en sport professionnel, il y a les sportifs nécessairement sans équipements (smartphone ou montre connectée), l'équipe médicale, les entraîneurs, les agents des sportifs, les managers, ... constituant les différents utilisateurs de l'écosystème D-SHIRT. Chacun d'entre eux a des besoins de consultation spécifique concernant un seul sportif dans le cas des sports individuels ou plusieurs sportifs dans le cas des sports collectifs. Les indicateurs collectés simultanément sur plusieurs joueurs sont centralisés, synthétisés et ensuite distribués vers différents destinataires selon leur droit d'accès et leurs besoins. Dans la figure 5.3, nous montrons cette architecture dans une configuration de sport amateur individuel.

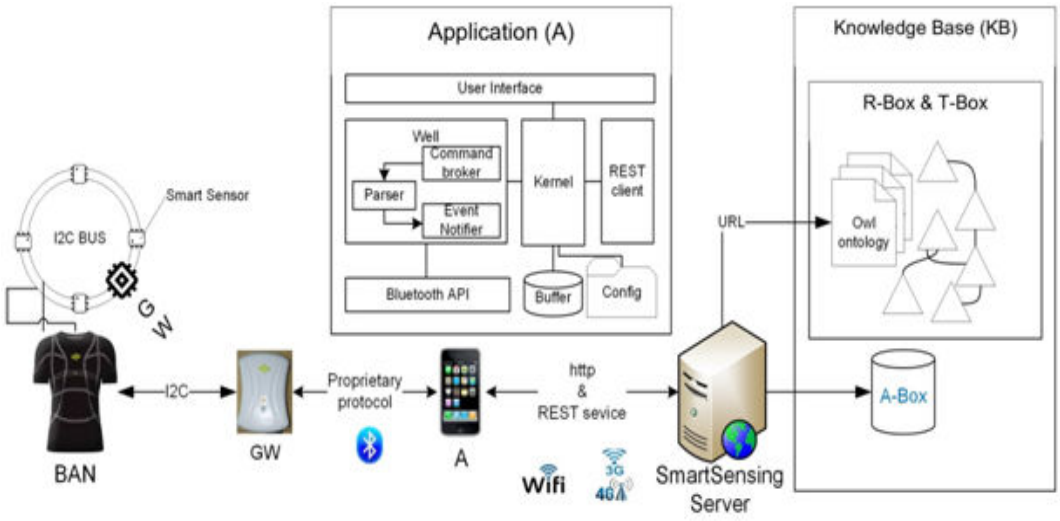

FIGURE 5.3 – Architecture de la plateforme D-SHIRT

5.5.1 Les éléments matériels de l'architecture

L'écosystème D-SHIRT est structuré en quatre niveaux. Au plus près du sportif, il y a le VI illustré dans la figure 5.3 par le T-Shirt BAN sur lequel sont localisés les capteurs intelligents. Ce premier niveau de l'écosystème est nommé BAN (*Body Area Network*). Il correspond au niveau des "capteurs actuateurs" dans l'architecture de référence d'une Plateforme IdO. Au deuxième niveau de l'écosystème, se trouve la Gateway un dispositif électronique fabriqué par un des partenaires du projet SmartSensing. Elle correspond au niveau de la "passerelle" (dans l'architecture de référence d'une Plateforme IdO) assurant le lien entre le BAN et le monde extérieur au T-shirt. Le troisième niveau de l'écosystème D-SHIRT, correspondant au niveau middleware d'intégration IdO (dans l'architecture de référence d'une Plateforme IdO), il est illustré par le Smartphone. Il s'agit d'un équipement disposant de capacité de communication Bluetooth situé à proximité radio de la(les) Gateway(s) et capable de dialoguer avec elle(s). Au quatrième et dernier niveau de l'écosystème D-SHIRT, correspondant au niveau "Application" (dans l'architecture de référence d'une Plateforme IdO) est illustré par le Smartphone et les serveurs situés dans le *cloud*. Sur ces différents équipements divers applications métiers sont hébergés et mises à disposition des utilisateurs.

Capteur intelligent

Les capteurs intelligents se présentent physiquement sous forme de patch. Nous y trouvons, par exemple, le capteur cardiaque (photo "a" de la figure 5.4) ou les accéléromètres (photo "b" de la figure 5.4) pour capturer les mouvements des membres du corps humain.

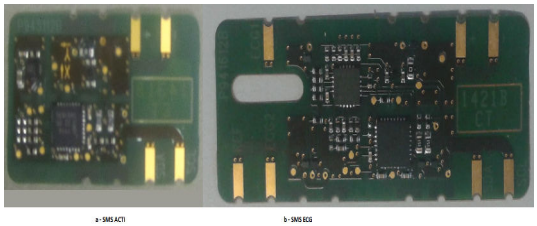

FIGURE 5.4 – Photo du capteur cardiaque et du capteur accéléromètre

Le textile du VI intègre un bus I2C formé de 3 fils sur lesquels sont branchés tous les smart-sensors. Le premier fil, SDA (*Signal DAta*), est utilisé pour transmettre les données. Le deuxième fil, SCL (*Signal CLock*) est utilisé pour transmettre un signal d'horloge synchrone. Le dernier fil est la masse permettant d'unifier les références des capteurs connectés au bus. Ces derniers sont tous alimentés par la même source d'énergie la Gateway. Le bus I2C est intégré dans le VI par une procédure de tissage particulière assurant son isolation et sa souplesse. Les capteurs intelligents sont imprimés sur du FPCB (*Flexible Printed Circuit Board*) un support d'impression de circuit imprimé flexible. Comme le montre la figure 5.5, le capteur intelligent est constitué :

— d'un ou plusieurs capteurs analogiques mesurant les grandeurs physiques à capturer,

— d'une unité de Conversion Analogique Numérique (CAN, en anglais ADC : *Analog to Digital Converter*),

— d'un module d'alimentation capturant l'énergie véhiculant sur le bus I2C et la distribuant au différents éléments du Smartsensor,

— d'un micro-contrôleur, d'une superficie granulaire (0.11mm²), de type Arm Cortex-M0, chargé d'effectuer les traitements de calcul et d'exécuter les algorithmes spécifiques à l'activité du sportif ;

— d'un module de communication I2C esclave assurant la communication avec la Gateway par le bus I2C.

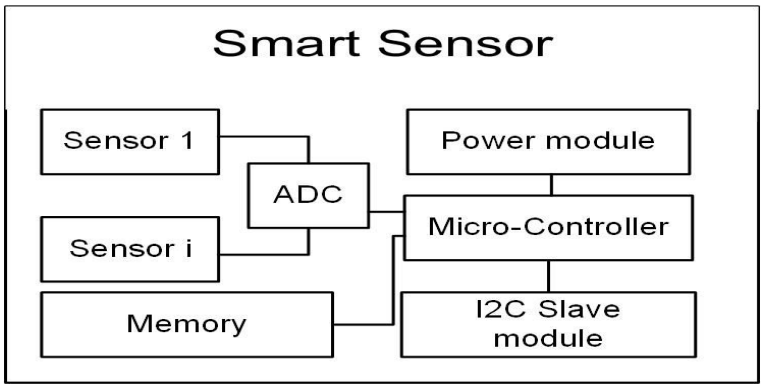

FIGURE 5.5 – Architecture d'un capteur intelligent

Les caractéristiques techniques des composants d'un smartsensor sont récupérées de leur datasheet (fiche technique) respective. Elles sont décrites avec le module S3N-Datasheet. La compilation de ces caractéristiques techniques aboutit à la construction de la fiche signalétique du smartsensor. Cette dernière recense les capacités de détection, de traitement et de communication du smartsensor. Elle est décrite avec le module S3N-System. La description de plusieurs composants électroniques dans une base de connaissances à partir des informations fournies dans les catalogues des fabricants d'électronique permet à l'écosystème de fournir un service d'assistance aux membres du consortium dans la sélection des constituants d'un smartsensor. L'écosystème peut ensuite fournir un service de compilation de la fiche signalétique de ce smartsensor.

Gateway

La gateway est un boîtier fabriqué par un partenaire du consortium 5.6. Elle se connecte au vêtement intelligent *via* un connecteur micro-usb. Elle se loge dans une pochette du VI située entre les omoplates. La gateway alimente et pilote le BAN et les smartsensor qui y sont connectés. Elle est imprimée sur PCB rigide.

Elle embarque un ensemble de capteurs et une batterie rechargeable 5.7. Les capteurs qu'y sont intégrés ne font pas partie du périmètre de la fonctionnalité Gateway, mais pour des raisons d'opti-

FIGURE 5.6 – Gateway du projet SmartSensor

misation spatiale et de réduction de coût, ils ont été imprimés sur le même PCB que la gateway. Ces capteurs sont :

— un GPS pour localiser le sportif en plein air ;

— un altimètre pour mesurer avec plus de précision l'altitude du sportif ;

— une centrale inertielle à 9 axes afin d'estimer l'orientation du sportif, sa vitesse linéaire et sa position.

FIGURE 5.7 – Contenu de la Gateway

Ces capteurs communiquent avec la Gateway soit via le bus I2C ou via des ports de communication RS-232 dédiés afin de soulager le trafic de données sur le bus I2C. Quant à la partie de la gateway assurant le rôle de passerelle, elle est constituée des éléments suivants :

— d'un module de gestion de la batterie et de distribution de l'énergie aux différents capteurs du BAN ;

— d'un microcontrôleur de type ARM Cortex-A5, chargé de piloter le VI et de calculer les indicateurs déduits à partir des mesures ou indicateurs issues de plusieurs capteurs ;

— d'un module de pilotage de la mémoire de masse pour stocker les fichiers de configuration, le profil du sportif, les codes applicatifs au format binaire des smartsensors et de la gateway ainsi que les fichiers de logs spatio-temporels contenant les valeurs des mesures et indicateurs collectés lors des activités sportifs ;

— d'un module de communication I2C maitre assurant le pilotage des dispositifs esclaves branchés sur le bus I2C ;

— d'un module de communication Bluetooth Low Energy (BLE) pour communiquer avec le monde extérieur au VI.

Comme pour les Smartsensor, la Gateway peut être décrite avec les mêmes modules ontologiques. L'utilité d'une telle description est de permettre à l'écosystème de comparer les capacités fournies par la Gateway avec celles exigées par les algorithmes à exécuter par la Gateway.

Middleware d'intégration

Dans le sport amateur, le middleware d'intégration n'est rien d'autre qu'un smartphone grand public ayant un module de communication BLE, un module de communication 4G et/ou WIFI pour communiquer avec les serveurs de l'écosystème hébergés dans le cloud. Les capacités du smartphone doivent permettre l'exécution d'une application Android dédiée au pilotage du VI et au monitoring du sportif. Dans le sport professionnel, le middleware est un ordinateur connecté d'une part à un réseau intranet et d'autre part à plusieurs gateway simultanément via plusieurs modules de communication Bluetooth. Cet ordinateur héberge une application chargée de piloter en parallèle plusieurs gateway et de collecter leurs données pour les distribuer vers les différentes applications clients du réseau intranet.

Application

Pour le sport amateur, une partie de l'application est localisée chez le sportif, nous l'appellerons partie locale. L'autre partie est hébergée chez des partenaires du projet dans le cloud. Pour le sport professionnel, l'application est éclatée entre des terminaux à disposition des membres du club sportif et des serveurs localisés dans son infrastructure intranet. Nous aurons ici aussi les deux mêmes nominations.

La partie locale. Pour le sport amateur, l'application joue le rôle d'interface entre le sportif et le VI et d'intermédiaire entre la gateway et les serveurs de l'écosystème. Dans un premier sens, elle récupère les mesures et indicateurs remontés par la Gateway, les visualise au sportif et les transfère au serveur de l'écosystème. Dans un autre sens, elle fournit au sportif, une interface graphique de pilotage de la Gateway à qui elle envoie les instructions et elle récupère des serveurs, les codes applicatifs à installer dans le VI et le paramétrage à effectuer sur le VI. Pour le sport professionnel, les applications dédiées aux différents utilisateurs de l'écosystème sont de type tableau de bord présentant des récapitulatifs de données physiologiques des sportifs. Ces données proviennent du middleware d'intégration pour les mesures et indicateurs collectés en temps réel et des serveurs pour les données historiques.

Le cloud. Pour le sport amateur, le nombre de sportifs peut être assez grand. La masse de données qu'ils engendrent peut devenir gigantesque et se mesurer en péta-octets. Elle nécessite, par conséquent, une infrastructure de stockage lourde avec un temps d'accès performant. Une architecture de type grille de serveurs échelonnable est nécessaire. Sur ce type d'architecture le consortium a opté pour

stocker les mesures et indicateurs sur la plateforme Hadoop. Pour ce qui est des données de gestion des sportifs, des profils, des configurations et des équipements, d'autres serveurs classiques sont prévues. Ces serveurs hébergent les applications métiers et des bases de données. Pour le sport professionnel, la même architecture est préconisée mais avec une dimension plus réduite pour le stockage des données physiologiques et des applications professionnelles sur mesure développées à la demande.

5.5.2 Les éléments logiciels de l'architecture

Dans cette partie nous mettons essentiellement l'accent sur les constituants du système et leurs liens avec le système de gestion de connaissances déployant toutes les composantes sémantiques (ontologies, règles et requêtes).

Avant d'arriver à proposer l'architecture D-SHIRT, plusieurs solutions alternatives ont du être étudiées.

1. Tout d'abord, en raison de l'usage des ontologies et des technologies du Web sémantique durant les deux phases du cycle de vie d'un objet intelligent et en particulier durant la phase d'exploitation, une décision architecturale devrait être prise quant au composant ou au noeud du système sur lequel seraient déployées les modules ontologiques ainsi que les programmes dédiés aux interrogations et aux inférences.Trois possibilités peuvent être considérées : Le noeud capteur intelligent Le noeud passerelle Le smart phone Le noeud serveur
2. Ensuite, il était question de décider du format d'échange et des modalités de conversion depuis les données brutes issues du matériel électronique (les capteurs) et les modèles ontologiques, phase que nous avons nommée sémantisation-désémantisation.

Les réponses à ces interrogations ont permis de décider de l'architecture suivante : Le système est constitué d'un serveur (S) déployé dans une infrastructure Cloud, d'une application pour smartphone (A), d'une Gateway (GW) qui se branche au vêtement où un réseau BAN (Body Area Network) est tissé. Sur ce réseau sont branchés les capteurs intelligents (SS). Le serveur (S) référence les six modules de l'ontologie SMS. Il contient une base de triplets RDF où sont stockées les connaissances de configuration et les données collectées par les capteurs des vêtements de tous les utilisateurs (après sémantisation). L'insertion de triplets RDF ou l'interrogation de la A-Box se fait via des requêtes SPARQL déclarées dans le serveur. Elles sont lancées de façon agnostique via des invocations de services REST.

Le smart phone embarquant l'application (A), dont l'usage durant l'activité sportive peut être facultatif, contient une application « SmartSensing » qui assure la communication avec la Gateway et le serveur (S). Elle gère le profil du sportif, ses sports et ses pratiques. Elle joue aussi le rôle d'intermédiaire bidirectionnel entre (S) et (GW). C'est au niveau de (S) que les processus de sémantisation et de dé-sémantisation sont réalisés. Puisque le traitement des données sémantiques a une empreinte trop importante sur la mémoire, le processeur et la consommation énergétique pour de l'électronique embarquée, nous avons limité la frontière sémantique au niveau de (S). L'application se charge alors de

réagir aux interactions du sportif pour piloter le vêtement (en guise de command broker) ; de récupérer les indicateurs en provenances de la GW (Event notifier), les afficher à l'utilisateur, et les envoyer à (S) via le client REST. Les données de configuration sont stockées dans des fichiers sauvegardés sur le serveur. Ces fichiers sont générés à partir de la A-Box via un processus de dé-sémantisation. En cas d'absence de connexion avec le serveur (S), l'application stocke provisoirement le log des indicateurs collectés dans une mémoire temporaire locale. Lors de l'établissement de la connexion, ce log est envoyé au serveur qui assure sa sémantisation.

La GW récupère tous les mesures et indicateurs envoyés par les SS, calcule les indicateurs dérivés et communique via Bluetooth avec l'environnement extérieur. Un SS peut être reprogrammé à la volée par la GW pour exécuter des algorithmes différents. Pour y parvenir, il intègre une infrastructure logicielle reconfigurable et un jeu de services de reconfiguration dédiés. En particulier pour le cas D-SHIRT, le framework OSGi a été choisi pour jouer le rôle de middleware reconfigurable et offrir les services de gestion du cycle de vie de Fragments de code (logiques de traitement embarquées). Dans le cadre d'un sport Running par exemple, un maillot doté d'un capteur intelligent ECG (électrocardiographe) et d'une Gateway suffit pour fournir l'ensemble des indicateurs souhaités par les pratiquants de Running. Faire fonctionner le système exige de configurer ses différents éléments (SS, GW et A) par rapport au profil du sportif et au sport pratiqué. Cette opération doit être transparente pour le sportif. Elle doit, donc, être réalisée automatiquement par le système à partir de la base de connaissances du serveur. Un « puits de données » (sink en Anglais) est utilisé pour permettre à un système informatique de communiquer avec un réseau de capteurs sans fil constitué de plusieurs nœuds (voir figure 5.8. Ces nœuds peuvent être fixes, nomades ou mobiles. Leur position n'est pas définie à l'avance. Pour véhiculer les informations, un nœud peut communiquer directement en mode point à point avec le puits de données ou grâce à un routage multi-saut.

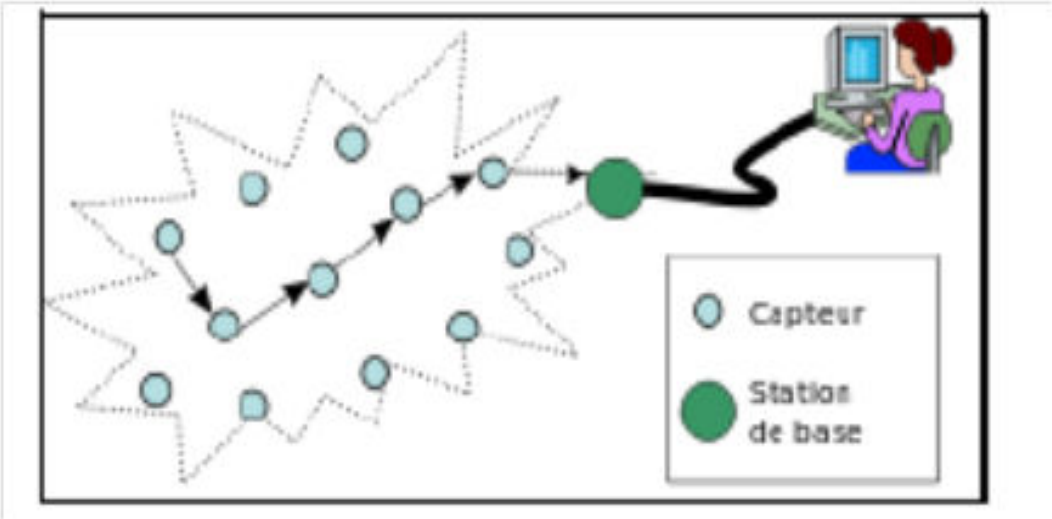

FIGURE 5.8 – Puits de données

Dans le cadre du projet SmartSensing et pour les applications hosts, le puits de données se présente

sous forme d'une librairie. Les applications destinées aux utilisateurs finaux de l'écosystème ont besoin de communiquer avec plusieurs Gateway. Le puits assure dans ce cas l'interfaçage entre cette ou ces applications et l'ensemble des Gateway(s). Il les relie tout en veillant à masquer l'hétérogénéité et la complexité des liaisons avec les Gateways.

Un utilisateur peut se connecter à une Gateway en mode Host ou en mode Configurateur. Le mode Host est basé sur une liaison Radio et le mode Configurateur sur une liaison filaire USB.

Le puits de données offre une API (API Well) pour permettre au système de réaliser toutes les opérations avec une Gateway quelle que soit la situation et le contexte d'usage. Elle permet de :

— Se connecter et se déconnecter d'une Gateway

— Récupérer en temps réel les données (mesures et indicateurs) des différentes Gateways

— Piloter à distance les différentes Gateways

— Envoyer des commandes à une ou plusieurs Gateways

5.6 Adaptation et reconfiguration

L'objectif du système D-SHIRT est de permettre avec le même vêtement d'opérer différentes configurations possibles ; ceci en le reconfigurant au niveau hard et soft. Hard en désactivant ou en activant au besoin un capteur, et soft en changeant à la volée sa logique de traitement. La logique de traitement pourrait être un simple fragment de code atomique (nommé ici Snippet), un code plus ou moins complexe ou encore le code d'un client de services Web qui pourrait envoyer des requêtes distantes à des services Web distants. Différents types d'accès à cette logique de traitement sont possibles (comme un Web service, ou en téléchargement et installation ou en invocation à courte distance via des protocoles spécifiques).

Les capteurs embarqués peuvent augmenter le potentiel du vêtement intelligent à travers leur coopération et collaboration. La vision ultime est qu'un VI ou alors l'écosystème dans lequel il est embarqué et/ou intégré pourrait offrir n'importe quelle type d'assistance ou traitement grâce à sa capacité de changer de logique de traitement au besoin et en fonction des programmes qui se trouvent par ailleurs. A défaut, et dans le cas où la logique de traitement n'est pas disponible, des techniques de génération de code peuvent à terme pouvoir être envisagées. Donc n'importe quelle tâche souhaitée pourrait être déléguée aux fragments de code téléchargés (en fonction des capacités de traitement disponibles). Le VI devient un "code Requester", il peut en fonction du coût, de la fréquence, de la vitesse, etc choisir lequel qui lui convient le mieux.

Dans un contexte où plusieurs VI sont à proximité, il leur serait également possible d'échanger à la fois des informations et des logiques de traitement. Ils peuvent fonctionner en mode peer to peer ou en mode client serveur. Comme exemple le VI du coach peut recommander des traitements aux VIs des sportifs coachés. Des politiques de sélection et d'installation de fragments de code peuvent être instaurées à terme. Cette logique d'adaptation et d'échange ou de téléchargement de fragments de code nécessite une infrastructure adéquate. Cette infrastructure requiert la gestion de tous les modèles

sémantiques et des répertoires de fragments de code, en gérant leurs métadonnées, en comparant leurs fonctionnalités via les paramètres d'entrées/sorties.

5.7 Mécanismes sémantiques d'adaptation

Soit (C) la configuration d'un capteur pour un Sport (S) donné et une pratique (Pr) donnée. Elle se compose de la séquence de snippets (Seq(Sn)) à installer ou déjà installée dans un capteur. Cette configuration (C) fait référence à un plan (P) d'exécution de ces Snippets pour le cas d'une activité donnée (A) pour la pratique (Pr). Lorsque l'utilisateur décide de changer l'usage de son VI pour un autre sport ou une autre activité ou une autre pratique, la configuration de celui-ci doit changer. L'ensemble des étapes suivantes est alors exécuté :

1. L'utilisateur signale au D-SHIRT le changement d'activité
2. Une requête SPARQL est envoyée au serveur KMS avec l'ensemble des paramètres du nouveau contexte d'usage.
3. Le moteur de raisonnement exécute la requête pour sélectionner les nouveaux snippets requis.
4. Le résultat de la sélection est affiné en considérant les contraintes du smart sensor quant à l'espace mémoire restant, la taille des Snippets nécessaires au nouvel usage, les snippets qui sont déjà installés, etc.
5. La liste finale des snippets est identifiée avec un nouveau plan (P) d'exécution
6. Le smart sensor compare les snippets disponibles localement par rapport à ce qui est requis.
7. si l'intersection entre ce qui est disponible (D) et ce qui est requis (R) est non vide alors l'ensemble des snippets à télécharger est égal à (R-D) le cas échéant il faut télécharger l'ensemble (R).
8. Le smart sensor déclenche les opérations requise pour dé-installer, et installer les nouveaux snippets manquants et l'écosystème poursuit son fonctionnement.
9. Le smart configure les snippets (initialisation d'un mapping entre les variables du snippet et les propriétes des ontologies)
10. exécution des snippets et remonter les résultats pour les enregistrer au niveau de la base (ABox) sur le serveur.

Ce mécanisme de reconfiguration a été en partie conçu lors de la construction des modules ontologiques (en particulier les modules S3NAlgorithm, S3NCore, MI et Sport). Un ensemble de règles d'inférences a été pour cela concu et développé en vue de réaliser le mappning entre snippets et propriétés. Les tests de ce mécanisme seront réalisés dans le chapitre 6

5.8 Conclusion

Nous avons traité dans ce chapitre la problématique liée à la conception et développement d'écosystèmes ou de plateformes IdO dédiées aux VIs ou plus précisément aux VIs de sport. Nous avons considéré pour cela des approches existantes. Celles-ci sont inspirées des approches utilisées pour le développement classique de logiciel, elle ne considèrent pas de manière spécifiques les problématiques d'intégration entre le monde physique des capteurs et le monde virtuel, et ne considèrent pas non plus les problèmes issus du manque d'interopérabilités et de généricité des plateformes offertes. Nous avons donc proposé dans un premier temps d'enrichir sémantiquement l'une des ces méthodologies, que nous avons suivi par la suite pour la spécification de l'architecture de l'écosystème D-SHIRT. Ce dernier se base sur le cadre sémantique développé au préalable (notamment l'ontologie SMS) et se distingue par sa capacité à s'adapter et à reconfigurer ses fonctionnalités à différents scénarios d'usage grâce à une spécification sémantique de mécanismes de reconfiguration qui opère sur des fragments de codes déployés sur le ou les Smarts Sensors de la plateforme IdO. Dans le cadre de cette thèse nous avons considéré l'hypothèse où le smart sensor (notamment celui qui embarque la logique de traitement) est conçu au dessus d'un Framework auto-reconfigurable et intégrant des mécanismes de reconfiguration de bas niveau permettant d'offrir des services de gestion de cycles de vie des fragments de code.

Chapitre 6

Validation des principales contributions de la thèse

Sommaire du présent chapitre

6.1 Introduction

Ce chapitre a pour objectif de valider les principales contributions de la thèse. Nous commençons donc par proposer notre conception de l'approche de validation. Nous décrivons par la suite l'exécution de cette approche de validation, nous concluons vers la fin par les résultats obtenus.

6.2 Conception de l'approche de validation

En vue de valider l'ensemble des composantes proposées dans le cadre de cette thèse, il nous est tout d'abord nécessaire de concevoir l'ensemble des éléments qui vont nous permettre de réaliser cette validation. Etant donné que l'une des composantes fondamentales de notre proposition est le framework de modélisation sémantique de VI ou de PI et que ce Framework est essentiellement formés

des modèles ontologiques définies dans le cadre de l'ontologie modulaire SMS, un premier niveau de validation serait de concevoir un ensemble de scénarios de validation de ces modules ontologiques et de vérifier que ces derniers permettent d'assurer convenablement la conception et la fabrication du VI ou du PI. Ensuite, il est indispensable de concevoir un prototype d'écosystème qui va permettre dans un premier lieu d'exploiter le VI dans le cadre d'un premier scénario d'usage et dans un second lieu de le reconfigurer pour l'adapter à un deuxième scénario d'usage. La mise en œuvre de ce prototype aura pour objectif de valider et de montrer encore une fois l'utilité des modules ontologiques. De plus il aura comme objectif de valider le mécanisme de reconfiguration/adaptation du VI à différents usages.

6.3 Scénario d'aide à la fabrication

Soit la SmartTextile qui utilise la base de connaissances du Framwork FSMS pour la description des composants de ses vêtements intelligents pour le sport. Nous présentons dans cette section un scénario typique d'aide à la conception/fabrication d'un capteur intelligent. Ce scénario consiste principalement à la sélection des composants, matériels (*Sensors*) et logiciels (*Algorithms*), à assembler pour concevoir un capteur intelligent. L'entreprise ayant un nouveau besoin et qui est la création d'un VI pour le sport "Running" et particulièrement pour la pratique "LongTrail". Pour ce faire ;

Étape_Fabrication 1. Nous avons besoin d'avoir la liste des pratiques du sport "Running". La requête suivante donne la liste des pratiques dont le résultat est donné dans le tableau 6.1.

```
PREFIX sport: <http://w3id.org/Sport#> .
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
PREFIX DUL: <http://www.loa.istc.cnr.it/ontologies/DUL.owl#> .
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#> .

Select ?SportingPractice
WHERE{
        ?SportingPractice rdf:type  sport:SportingPractice.
        ?Sport rdf:type  sport:Sport.
        ?SportingPractice  sport:isPracticedin ?Sport.
        ?Sport DUL:hasDataValue "Running"^^xsd:string.
}
```

Object	Property	Object
ShortTrail	sport :isPracticedin	Running
LongTrail	sport :isPracticedin	Running

TABLEAU 6.1 – Pratiques du Running

Étape_Fabrication 2. Pour chaque pratique nous devons déterminer la liste des indicateurs de performance. Sachant que la même pratique peut être pratiquée dans des autres sport il est important de spécifier les indicateurs de performance de la pratique en question (exp. "LongTrail") pour le Sport donné "Running". La requête suivante donne la liste des indicateurs pour la pratique "LongTrail", dont le résultat est donné dans le tableau 6.2.

```
PREFIX sport: <http://w3id.org/Sport#> .
PREFIX mi: <http://w3id.org/MI#> .
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
PREFIX DUL: <http ://www.loa.istc.cnr.it/ontologies/DUL.owl#> .
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#> .

Select ?PerformanceIndicator
WHERE{
?PerformanceIndicator rdf:type  mi:PerformanceIndicator.
?SportingPractice rdf:type  sport:SportingPractice.
?SportingPractice  sport:hasPerformanceIndicator ?PerformanceIndicator.
?SportingPractice  DUL:hasDataValue "Running"^^xsd:string.
?Sport rdf:type  sport:Sport.
?PerformanceIndicator  sport:forSport ?Sport.
?Sport DUL:hasDataValue "LongTrail"^^xsd:string.
}
```

Object	**Property**	**Object**
HydrationRate	sport : hasPerformanceIndicator	LongTrail
CaloriesConsumedNumber	sport : hasPerformanceIndicator	LongTrail
StepNumber	sport : hasPerformanceIndicator	LongTrail

TABLEAU 6.2 – Indicateurs pour la pratique "LongTail"

Étape_Fabrication 3. Pour chaque Indicateur de performance nous devons trouver l'algorithme qui permet de le calculer. Le mapping entre les deux modules MI et S3N-Algorithm au niveau des concepts "mi :PerformanceIndicator" et "s3n :PostCondition" Nous avons alors les faits suivants :

HydrationRate	a	s3n :PostCondition
CaloriesConsumedNumber	a	s3n :PostCondition
StepNumber	a	s3n :PostCondition

TABLEAU 6.3 – Résultat du Mapping entre le module S3NAlgorithm et le module MI

Ceci revient alors à chercher pour chaque indicateur (PreCondition du module S3N-Algorithm)

l'algorithme qui permet de le déterminer. Soit l'indicateur que nous cherchons est "HydrationRate", la requête suivante permet de retourner l'algorithme pour calculer cet indicateur.

```
PREFIX s3n: <http://w3id.org/s3n/#> .
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
PREFIX DUL: <http ://www.loa.istc.cnr.it/ontologies/DUL.owl#> .
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#> .
PREFIX ssn: <http://www.w3.org/ns/ssn/> .

Select ?Algorithm
WHERE{
    ?Algorithm rdf:type s3n:Algorithm.
    ?Indicator rdf:type s3n:PosCondition.
    ?Algorithm s3n:hasPostCondition ?Indicator.
    ?Indicator DUL:hasDataValue " HydrationRate"^^xsd:string.
}
}
```

Cette requête retourne l'algorithme "HydrationAlertAlgorithm".

Étape_Fabrication 4. Une fois l'algorithme Obtenu nous devons connaître les mesures (comme précondition d'un algorithme) qui permettent de calculer l'indicateur (requête ci-dessus). Ceci permet de déterminer les capteurs basiques nécessaires qui permettront de les détecter. Le résultat de cette requête donne dans le tableau 6.4.

```
PREFIX s3n: <http://w3id.org/s3n/#> .
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
PREFIX DUL: <http ://www.loa.istc.cnr.it/ontologies/DUL.owl#> .
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#> .
PREFIX ssn: <http://www.w3.org/ns/ssn/> .

Select ?Measurement
WHERE{
?Measurement rdf:type s3n:PreCondition .
?Algorithm rdf:type s3n:Algorithm.
?Algorithm s3n:hasPreCondition  ?Measurement .
?Algorithm DUL:hasDataValue " HydrationAlertAlgorithm"^^xsd:string.
}
```

Les précondition du module S3N-Algorithm sont équivalents aux s3n :Measurand (module S3N-DataSheet). Un mapping entre les deux modules (au niveau de l'ontologie S3N) permettra de donner

HydrationAlertAlgorithm	s3n :hasPreCondition	Temperature
HydrationAlertAlgorithm	s3n :hasPreCondition	Acceleration

TABLEAU 6.4 – Mesures pour l'indicateur "HydrationRate"

la liste des références des accéléromètres et des capteurs de température.

Un appel de la règle *addSensor* pour les deux mesures (*Tempereture* et *Acceleration*), permet de relier des capteurs basiques pour le nouveau capteur intelligent créé "SMSHydrationAlert".

```
@prefix  s3n: <http://w3id.org/s3n/#> .
@prefix  sosa: <http://www.w3.org/ns/sosa/#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

[addSensor: ("SMSHydrationAlert" rdf:type s3n:SmartSensor)
(?precondition rdf:type  s3n:PreCondition)
(?sensor rdf:type  sosa:Sensor)
(?sensor  s3n:hasSensorReference ?SensorReference)
(?SensorReference  s3n:hasMeasurand ?Measurand)
stringEqualIgnoreCase(?Measurand, ?precondition)
-> ("SMSHydrationAlert" ssn:hasSubSystem ?sensor)]
```

6.4 Phase d'exploitation d'un VI

Dans ce qui suit, nous nous intéressons exclusivement à l'échange sémantique durant deux étapes de la phase d'exploitation : L'étape de configuration automatisée et l'étape de pratique sportive. Au préalable, nous ferons un point sur l'état de la base de connaissances supposée préalablement peuplée par les fournisseurs.

Les assertions de l'A-Box après la phase de fabrication. Parmi les assertions crées par les fournisseurs, nous n'évoquons ici qu'un échantillon de celles nécessaires pour le fonctionnement du système durant la phase d'exploitation. Nous les illustrons en partie pour le sport du vélo ("Cycling") :

— Pour un sport donné, le système a besoin de connaître les pratiques qui lui sont associées et les panoplies de vêtement qui se prêtent à sa pratique.

— Pour chaque pratique de ce sport, le système doit connaître tous les indicateurs qu'il doit calculer et leur distribution sur les SS et la GW. Pour chaque indicateur, le système a besoin de connaître l'identifiant unique qui lui est associé dans les composants du système (SS, GW, A et S), l'implémentation de l'algorithme à exécuter ainsi que la cible sur laquelle il sera exécuté.

À titre d'exemple, le tableau 6.5 montre quelques assertions pour le sport "Cycling".

Subject	**Property**	**Object**
Cycling	rdf :type	sport :Sport
TrailCourt	rdf :type	sport :SportingPractice
TrailLong	rdf :type	sport :SportingPractice
TrailCourt	sport :isPracticedin	Running
TrailLong	sport :isPracticedin	Running
TrailCourt	sport :hasPerformanceIndicator	FatigueState
TrailCourt	sport :hasPerformanceIndicator	HeartRate
TrailCourt	sport :hasPerformanceIndicator	Location

TABLEAU 6.5 – Instances du module Sport pour le "Cycling"

Le module "S3N-Algorithm" permet de construire un graphe de dépendances entre les indicateurs (post-conditions) et les mesures (précondition). La sélection d'un sport permet de connaître, pour chacune de ses activités ce graphe. Par conséquent, l'utilisation d'un raisonneur et d'une requête d'interrogation permet de trouver toutes les données à traiter pour un sport. Ce graphe de dépendance sera ensuite exploité par l'ordonnanceur d'exécution des traitements, logé dans la GW, afin de déterminer enchaînement d'invocation des algorithmes.

Étape de configuration de la phase d'exploitation. Soit maintenant un sportif qui utilise l'application A pour signaler son intention de pratiquer le Running. Le système vérifie l'existence d'une configuration pour ce sport dans son maillot intelligent. En cas d'absence, c'est qu'il s'agit d'une première pratique de ce sport avec son maillot qui requiert le chargement d'une configuration à partir du serveur. C'est uniquement dans cette situation que l'étape de configuration est déclenchée. L'application A demande alors au serveur de charger une configuration du sport Running pour l'application A et pour la GW. Le serveur S, procède alors à une succession d'interrogations de la base de connaissances, à la dé-sémantisation des assertions obtenues et la génération des fichiers de configurations. Ces fichiers sont empaquetés et transmis à l'application. Cette dernière s'auto-configure pour offrir les fonctionnalités et l'IHM spécifique au Running. Elle transmet ensuite à la GW le paquetage de configuration qui lui est destinée. Dès lors, la GW s'auto-configure, déploie les algorithmes sur les micro-contrôleurs et configure l'ensemble des SS du BAN. des requêtes similaires à la section 6.3 permettront de définir comment trouver les pratiques de Running, la liste des indicateurs utilisés pour le Running, et les SS à configurer pour ce sport.

Étape de pratique de la phase d'exploitation. Au démarrage d'une pratique par le sportif via l'application A, un ordre est envoyé à la GW pour démarrer. Une session d'activité est créée. Cette session permet de connaître l'instant de démarrage de l'activité, le propriétaire de la session, la panoplie vestimentaire utilisée (le maillot dans notre cas) et l'identifiant unique de la GW utilisée. Suite au démarrage de la session d'activité, les processus de calcul des indicateurs sont activés. Périodiquement,

la GW remonte à l'application A des notifications événementielles contenant chacune une suite de trames TLV (Type, Longueur, Valeur). Le type correspond à l'identifiant de la données (mesure ou indicateur) connue dans la GW. Longueur indique la taille de la Valeur qui contient la valeur de la donnée. L'application interprète le contenu des trames TLV et prépare les données à envoyer au serveur. Pour chaque donnée, elle convertit l'identifiant connue par la GW avec l'identifiant connue par l'application, l'affiche sur l'IHM et prépare une invocation d'un service Web pour envoyer le lot de données au serveur S.

6.5 Prototype de validation

Tout d'abord, et à cause en partie de l'indisponibilité de l'ensemble du matériel et des capteurs prévus initialement dans le projet smartsensing, vu qu'il a été abandonné en cours de route, nous devons décider de l'ensemble des éléments matériels qui vont intervenir dans la conception du prototype.

Nous réalisons ceci en projetant l'architecture proposée de l'écosystème sur un modèle d'écosystème simplifié, où nous limitons le VI, l'ensemble des capteurs ainsi que la gateway à une plateforme Raspberry Pi comportant un ensemble de capteurs comme le montre la figure 6.1.

Le VI et la gateway forment alors le front end du système qui va interagir avec le serveur KMS de l'architecture initiale proposée comme le montre la figure 5.3. Le tableau 6.6 permet d'établir la correspondance entre les éléments de l'architecture de l'écosystème D-SHIRT et celle du prototype.

Architecture écosystème prototype	**Architecture proposée de l'écosystème**
Gateway embarquant les capteurs, l'application smart sensing A et le puits W	VI ou PI
	Capteurs embarqués
	Gateway
	Application mobile
	Puits
	Serveur KMS

TABLEAU 6.6 – Correspondances entre D-SHIRT et le prototype de validation

6.5.1 Architecture du Prototype de validation

L'architecture du prototype est une architecture distribuée. Pour des raisons de simplification et en vue d'illustrer l'ensemble des composants de l'architecture ainsi que leurs interactions, nous avons déployé l'ensemble des composants distribuées sur seulement deux composants à savoir un client et un serveur, fournissant ainsi une architecture de type client serveur comme c'est illustré dans la figure 6.1

Ces deux composants ainsi que leurs constituants sont décrits dans ce qui suit :

Architecture du client. Le client de l'écosystème regroupe l'ensemble des composants matériels et logiciels qui sont requis au niveau local (par rapport à l'usager final de l'écosystème). Il contient

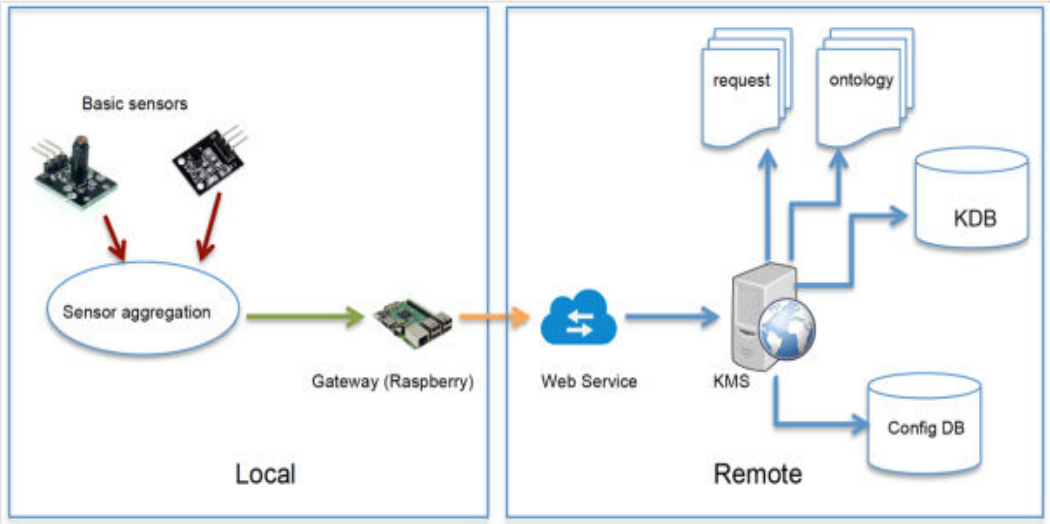

FIGURE 6.1 – Architecture simplifiée de la plateforme IdO (D-SHIRT)

donc : l'ensemble des capteurs qui ont pour rôle de détecter les données brutes requises par l'activité courante d'usage. Un capteur d'agrégation de données pour simplifier la connexion avec la passerelle. Celle-ci, en l'absence du capteur d'agrégation peut jouer un double rôle ; celui d'un capteur intelligent qui agrège et raffine les données afin de répondre aux besoins précis de l'utilisateur. Le deuxième rôle est celui d'une vraie passerelle qui agit en guise d'intermédiaire qui se charge de transformer les données dans un format convenable et de les véhiculer vers le serveur pour un traitement ultérieur spécifique. Dans le cadre de ce prototype un ensemble réduit de capteurs pouvant être connectés *via* une connexion Bluetooth ont pu être appariés avec la passerelle. De plus l'ensemble des données capturées a été simulé grâce au micro-contrôleur de type "STM32F103". Une carte Raspberry Pi B+ a été utilisée en guise de passerelle. Celle-ci représente, comme c'est exigé dans la plupart des plateformes IdO actuelles, un vrai périphérique Intelligent "Smart device", embarquant un système d'exploitation, pouvant exécuter des programmes de manière très autonome et offrant une connexion permanente à Internet. Ce type de "smart device" peut aussi être représenté par un smartPhone ou une tablette.

Architecture du serveur. Dans le cadre du présent prototype, toutes les potentialités intelligentes du système résultat des inférences et des interrogations faites sur les ontologies est déployée au niveau du serveur. Celui-ci comporte de plus les répertoires (*Repositories* en Anglais) sémantiques comprenant les modèles ontologiques et leurs instances. Il regroupe également dans ses bases de données les données de configuration, les données relatives aux indicateurs. Comme c'est illustré dans la figure 6.1, il déploie :

— la base de connaissance (l'ensemble des instances des ontologies) ;

— un moteur d'inférences (Jena dans ce cas) ;

— un ensemble d'ontologies (Ontologie du domaine sport, et mesure et indicateurs, ontologie des

bouts de codes (Snippets) ;

— le modèle de contexte d'usage de l'utilisateur final (ce sont les éléments de Configuration comme par exemple, l'activité courante, la nature du capteur intelligent, . . .).

Le middleware de services Web d'intégration entre le client et le serveur. Afin de garantir un niveau d'interaction et d'interopérabilité entre le client et le serveur qui soit indépendant de la technologie sous-jacente aux deux composants, nous avons opté pour le choix d'un middleware de services Web de type SOAP. Le serveur expose donc l' API du composant KMS sous forme de fichier WSDL. Ceci permet donc d'établir une correspondance ("binding") entre les requêtes SPARQL et leurs arguments pour interroger les ontologies du serveur KMS ainsi que sa base de connaissances.

Le Framework Lerna. L'écosystème distribué de notre prototype D-SHIRT inclue une application cliente qui est embarquée au niveau de la passerelle comme l'illustre la figure 6.1. Cependant, dans des plateformes IdO de production, cette application pourrait être embarquée dans un smart phone ou tout autre smart device. Pour cela un Framework que nous avons baptisé Lerna a été conçu et développé. L'objectif principal de ce Framework est de réaliser les reconfigurations de l'écosystème au runtime (en cours d'exécution). L'ensemble des exigences suivantes ont été donc inférées pour atteindre cet objectif :

— Ce framework client doit pouvoir être embarqué sur n'importe quel smart device en particulier la passerelle dans notre cas. Le but étant de reconfigurer le code et les programmes qu'elle embarque.

— Récupère ses instructions de reconfiguration à partir du serveur via les interfaces de services Web.

— Etre capable de reconfigurer son propre code en offrant des primitives de gestion du cycle de vie du code embarqué (dé-installation /installation, démarrage, arrêt, etc) et ce de manière dynamique en vue de répondre aux instructions de reconfiguration dictées par le serveur.

Pour le besoin du prototypage, tout ce qui a trait à des conditions réelles d'exploitation comme la consommation d'énergie, le temps de reconfiguration n'a pas été considéré. Cela n'empêcherait de les considérer en priorité pour la plateforme de production.

Notre choix pour implémenter ce Framework s'est fixé sur l'usage d'un Framework OSGi avec toutes ses capacités de reconfiguration au runtime et de gestion du cycles de vie de ses bundles. Des décisions ont du être faite quant à la granularité du code empaqueté dans les bundles et sa manière de le gérer et de le stocker sur le serveur.

Le framework Lerna est composé de trois composants principaux comme l'illustre la figure 6.2 :

— Le noyau Lerna

— Le proxy réseau

— Un ensemble de snippets (fragements de code)

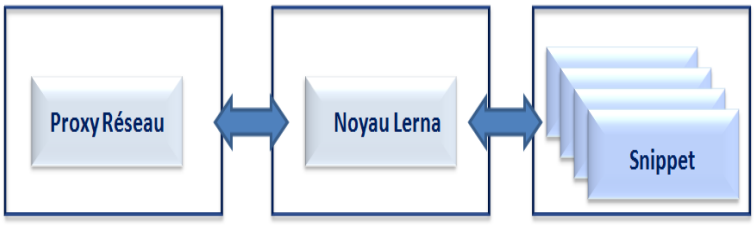

FIGURE 6.2 – Le Framework Lerna

Au démarrage, le noyau Lerna récupère sa configuration à partir du proxy, installe les bundles et services OSGi nécessaires et instancie les snippets au niveau de l'environnement d'exécution. Le noyau est aussi responsable du chargement des paramètres de configuration globaux de l'application telles que l'URL des services Web et des ports d'accès, l'URL du repository à partir duquel les snippets seront téléchargés, des fichiers de configurations des snippets qui sont spécifiques au device, etc. Le proxy réseau joue le rôle d'intermédiaire entre une application Lerna et le serveur. Les snippets ou fragements de code constituent des composants essentiels du Framework Lerna. Ces fragements de code sont capable d'être installés, dé-installés et/ ou exécutés au moment de l'exécution de l'ensemble de la plateforme. Les entrées/sorties et les logiques de traitements implémentés dans Lerna dépendent exclusivement de ces fragments de code. Donc dans une application Lerna, tout ce qui peut changer en cours d'exécution est implémenté comme un fragment de code (snippet) y compris les algorithmes, les drivers de capteurs, les filtres de données, les bibliothèques, etc. Les snippets sont packagées sous forme de bundles OSGi. Lors du chargement d'un bundle, tous les snippets qu'il contient deviennent disponibles dans le framwork Lerna. La création des instances de snippets au moment de l'exécution est faite conformément au design pattern Factory. Les références à ces instances créées sont alors enregistrées au niveau du registre de services OSGi, en vue de les rendre dynamiquement disponibles et dynamiquement manipulables au niveau de l'environnement d'exécution Lerna.

6.6 Règles d'inférence du mécanisme sémantique d'adaptation

L'adaptation d'un capteur intelligent se définit par sa reconfiguration en installant et désinstallant des fragments de code (*snippets*) pour exécuter un algorithme. Un algorithme traduit une exigence d'utilisation, par exemple pour calculer un indicateur de performance souhaité par un utilisateur de VI.

Pendant la reconfiguration d'un capteur intelligent (par exemple, "SMSACTI") deux points importants sont à vérifier :

1. Si le capteur intelligent dispose des capteurs basiques adéquats, ceci signifie que les capteurs basiques intégrés fournissent les bonnes mesures. Par exemple, nous considérons l'algorithme "HydrationAlertAlgorithm", pour calculer l'indicateur "HydrationRate" il aura besoin de deux mesures à savoir ; la température et l'accélération, ainsi "SMSACTI" devrait intégrer au moins un accéléromètre et un thermomètre ;
2. Si les capacités du micro-contrôleur sont satisfaisantes, ceci signifie qu'il possède d'au moins

une capacité de mémoire suffisante pour stocker les fragments de code, des données temporaires ou des données persistantes pour le calcul.

nous donnons ci dessous des exemples de règles utilisées par le mécanisme sémantique d'adaptation :

```
@prefix  s3n: <http://w3id.org/s3n/#>.
@prefix  sosa: <http://www.w3.org/ns/sosa/#>.
@prefix  ssn: <http://www.w3.org/ns/ssn/#>.
@prefix ssn-system: <http://www.w3.org/ns/ssn/systems/#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.

[ReconfigSmart1: (?smart rdf:type s3n:SmartSensor)
(?smart ssn:hasSubSystem ?microcontroller)
(?microcontroller ssn:implements ?algorithm)
(?algorithm s3n:hasOperation ?op)
(?op s3n:hasImplementation ?snippet)
(?snippet rdf:type sosa:Procedure)
-> (?microcontroller ssn:implements  ?snippet)]

[ReconfigSmart2: (?smart rdf:type s3n:SmartSensor)
(?smart ssn:hasSubSystem ?sensor)
(?sensor sosa:observes ?ObservableProperty)
(?algorithm rdf:type s3n:Algorithm)
(?algorithm  s3n:hasPreCondition ?precondition)
stringEqualIgnoreCase(?ObservableProperty, ?precondition)
-> (?algorithm s3n:hasPreCondition ?ObservableProperty)]

[ReconfigSmart3: ("SMSACTI" rdf:type s3n:SmartSensor)
("SMSACTI" ssn:hasSubSystem ?microcontroller)
(?microcontroller ssn:implements  ?snippet)
("SMSACTI" ssn:hasSubSystem ?sensor)
(?sensor sosa:observes ?ObservableProperty)
("PedalingSpeedAlgorithm" s3n:hasPreCondition ?ObservableProperty)
("PedalingSpeedAlgorithm" s3n:hasOperation ?opConf)
(?opConf s3n:hasImplementation ?snippetConf)
stringEqualIgnoreCase(?snippetConf, ?snippet)
-> (?microcontroller ssn:implements  "PedalingSpeedAlgorithm")]
```

6.7 Conclusion

Ce chapitre s'est focalisée sur l'approche de validation de l'ensemble des contributions de la thèse. Il a pour cela exploité les deux scénarios de fabrication et d'usage des VI pour tester l'adéquation des ontologies pour répondre aux question de compétences préalablement identifiées. L'écosystème prototype a aussi montré sa capacité à se reconfigurer à chaud en utilisant ces même modèles sémantiques. Nous tenons toutefois à souligner qu'une validation sur terrain en utilisant des un écosystème en production serait également utile pour opérer des réajustements et des raffinement éventuels des algorithmes et des traitement sémantiques déployés sur le serveur.

Chapitre 7

Conclusion et perspectives

Sommaire du présent chapitre

7.1 Conclusion

Les travaux décrits dans ce manuscrit se situent au niveau de la conjonction entre les technologies du Web sémantique et de l'Internet des Objets. En effet, les travaux réalisés dans le cadre de cette thèse nous ont permis d'aborder la problématique liée à la conception multidisciplinaire d'objets intelligents et connectés et leur exploitation dans différents contextes d'usage.
Nous rappelons que la problématique que nous nous sommes posées est la suivante : Quelles approches, méthodologies et outils peut-on proposer pour permettre à des acteurs multidisciplinaires de concevoir, fabriquer, déployer et adapter des objets et/ou des systèmes intelligents et connectés tout en tenant compte de leurs caractéristiques structurelles et comportementales, de leurs différents cas d'usage ainsi que du domaine de leur utilisation ? Cette problématique nous a guidé à poser un ensemble de questions de recherche. Pour y répondre, nous avons suivi une méthodologie fondée sur un ensemble d'investigations théoriques et pratiques. Une investigation des travaux existants nous a permis de positionner les axes de recherches traités dans cette thèse et de cadrer les solutions proposées au niveau de chaque axe. Trois axes structurent cette étude et identifient nos contributions :

— Un premier positionnement a été réalisé par rapport aux Frameworks de conception/fabrication d'OI et de systèmes intelligents connectés. Une analyse approfondie nous a permis de monter les lacunes de ces Frameworks et d'en proposer un que nous avons baptisé Framework FSMS. Le Framework FSMS proposé au niveau de cette thèse a la particularité d'être à la fois générique et fondé sur des modèles sémantiques. Il est structuré en trois composantes principales à savoir la composante sémantique, la composante méthodologique et la composante processus. Sa

composante sémantique ; principale contribution de cette thèse ; est formée d'un ensemble de modules ontologiques réutilisables permettant de décrire les OI, ces modules ontologiques forment l'ontologie SMS. La composante méthodologique offre un ensemble de directives à deux niveaux. Le premier est issue des approches en génie logiciel et celles pour la conception des plateformes IdO. Et le second se présente sous la forme d'activités d'une méthodologie de conception collaborative et modulaire d'ontologies. Et enfin, la composante processus vise à terme la possibilité d'automatiser ces processus de conception/fabrication d'OI et connectés.

— Dans l'objectif de définir la composante sémantique du Framework FSMS, une deuxième investigation a porté sur la représentation sémantique du composant clé des OI à savoir les capteurs intelligents. Au niveau de cet axe nous avons proposé la partie générique et réutilisable de l'ontologie SMS, à savoir l'ontologie S3N. Cette ontologie elle-même modulaire, elle est composée d'ensemble de modules ontologiques réutilisables et alignés sur des standards du W3C comme par exemple (SSN/SOSA et DOLCE UL).

— Un troisième positionnement a été réalisé par rapport aux plateformes IdO et en particulier leurs architectures et leurs approches de reconfiguration/adaptation. Nous précisons qu'à notre connaissance, il n'existe pas actuellement de plateforme IdO qui a exploité les technologies du Web sémantique pour permettre à ses utilisateurs d'utiliser des VI pour différents usages. Et même si ces plateformes intègrent des mécanismes d'adaptation, ceux-ci sont codés en dur dans la logique du capteur et ne sont pas du moins modifiables sans l'intervention d'acteurs avertis.

7.2 Perspectives

En guise de perspectives à ce travail, nous projetons :

— Enrichir l'ontologie SMS par un module de contexte d'usage et d'un modèle d'exigences qui permettra à terme une assistance intelligente à la conception/fabrication d'OI. D'autant plus que cela permettra de contextualiser davantage les résultats des phases de conception/fabrication préalablement réalisées.

— Considérer aussi l'enrichissement de l'ontologie SMS par un module permettant d'exprimer des logiques de collaboration et d'organisation entre les objets intelligents. Ceci servira a permettre un comportement intelligent collectif pour mieux répondre aux besoins des utilisateurs finaux et surtout pour ne pas générer des comportements d'objets individuels qui soient contradictoires.

— D'outiller le Framework FSMS de conception/ fabrication d'Objets Intelligents, de telle sorte que l'ensemble des acteurs puisse être assisté à la fois par les outils sous-jacents et par la méthodologie et le cadre sémantique préconisé.

— Automatiser les processus du Framework FSMS sous forme de services.

— Nous comptons également investiguer l'usage de l'écosystème sur terrain avec une réelle infrastructure matérielle et tissu intelligents.

— Il est important toutefois de signaler l'importance de la solution que nous avons proposée, pour tenir compte à la fois du domaine du sport et du coaching. Nous comptons dans ce contexte aborder l'expertise liée au domaine du coaching et créer ainsi des coachs assistants intelligents à embarquer dans l'écosystème et qui vont interagir avec les utilisateurs de VI et suivre le progrès des activités des utilisateurs. Il est de plus possible de considérer des co-équipiers ou des compagnons virtuels intelligents qui pourront non seulement intervenir dans le coaching mais surtout pour motiver l'utilisateur à réaliser d'avantage de progrès dans ses activités.

— Tester SMS dans le cadre de projet utilisant des plateformes IdO prouvés et ayant atteint un degré important de maturité

Bibliographie

[1] Ken PEFFERS, Tuure TUUNANEN, Marcus A ROTHENBERGER et Samir CHATTERJEE. “A design science research methodology for information systems research”. In : *Journal of management information systems* 24.3 (2007), p. 45–77.

[2] Samya SAGAR, Issam REBAI, Maha KHEMAJA et Jamel FEKI. “Ontologie modulaire pour la fabrication et l’exploitation de vêtements intelligents dédiés au sport”. In : *28es Journées francophones d’Ingénierie des Connaissances IC 2017*. 2017, p. 139–144.

[3] Samya SAGAR, Maxime LEFRANÇOIS, Issam REBAI, Khemaja MAHA, Serge GARLATTI, Jamel FEKI et Lionel MÉDINI. “Modeling Smart Sensors on top of SOSA/SSN and WoT TD with the Semantic Smart Sensor Network (S3N) modular Ontology”. In : *Emerging Topics in Semantic Technologies, ISWC 2018 Satellite Events*. Sous la dir. d’Elena DEMIDOVA, Amrapali J. ZAVERI et Elena SIMPERL. Berlin : AKA Verlag, 2018. ISBN : 978-3-89838-736-1.

[4] D. GIUSTO, Antonio IERA et Luigi Atzori (EDS.) “The Internet of Things”. In : *Springer* (2010).

[5] Luigi ATZORI, Antonio IERA et Giacomo MORABITO. “The Internet of Things : A survey”. In : *Computer Networks* 54.15 (2010), p. 2787–2805.

[6] Kevin ASHTON et al. “That ‘internet of things’ thing”. In : *RFID journal* 22.7 (2009), p. 97–114.

[7] IEEE Internet of THINGS. *Towards a definition of the internet of things (iot)*. IEEE, Tech. Rep. May, 2015.

[8] Charu C AGGARWAL, Naveen ASHISH et Amit SHETH. “The internet of things : A survey from the data-centric perspective”. In : *Managing and mining sensor data*. Springer, 2013, p. 383–428.

[9] Oscar CORCHO et Raúl GARCIA-CASTRO. “Five challenges for the semantic sensor web”. In : *Semantic Web* 1.1, 2 (2010), p. 121–125.

[10] Hoyoung JEUNG, Sofiane SARNI, Ioannis PAPARRIZOS, Saket SATHE, Karl ABERER, Nicholas DAWES, Thanasis G PAPAIOANNOU et Michael LEHNING. “Effective metadata management in federated sensor networks”. In : *Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC), 2010 IEEE International Conference on*. IEEE. 2010, p. 107–114.

[11] Danh LE-PHUOC, Hoan Nguyen Mau QUOC, Josiane Xavier PARREIRA et Manfred HAUSWIRTH. “The linked sensor middleware–connecting the real world and the semantic web”. In : *Proceedings of the Semantic Web Challenge* 152 (2011), p. 22–23.

[12] Amit SHETH, Cory HENSON et Satya S SAHOO. “Semantic sensor web”. In : *IEEE Internet computing* 12.4 (2008).

[13] Kevin A DELIN et Shannon P JACKSON. “Sensor web : a new instrument concept”. In : *Functional Integration of Opto-Electro-Mechanical Devices and Systems*. T. 4284. International Society for Optics et Photonics. 2001, p. 1–10.

[14] Xingchen CHU et Rajkumar BUYYA. “Service oriented sensor web”. In : *Sensor networks and configuration*. Springer, 2007, p. 51–74.

[15] Alasdair JG GRAY, Jason SADLER, Oles KIT, Kostis KYZIRAKOS, Manos KARPATHIOTAKIS, Jean-Paul CALBIMONTE, Kevin PAGE, Raúl GARCIA-CASTRO, Alex FRAZER, Ixent GALPIN et al. “A semantic sensor web for environmental decision support applications”. In : *Sensors* 11.9 (2011), p. 8855–8887.

[16] Michael COMPTON, Cory HENSON, Laurent LEFORT, Holger NEUHAUS et Amit SHETH. “A Survey of the Semantic Specification of Sensors”. In : *Proceedings of the 2Nd International Conference on Semantic Sensor Networks - Volume 522*. SSN’09. Washington DC : CEUR-WS.org, 2009, p. 17–32. URL : http://dl.acm.org/citation.cfm?id=2889933.2889935.

[17] Lionel M NI, Yanmin ZHU, Jian MA, Qiong LUO, Yunhao LIU, Shing Chi CHEUNG, Qiang YANG, Minglu LI et Min-you WU. “Semantic Sensor Net : an extensible framework”. In : *International Journal of Ad Hoc and Ubiquitous Computing* 4.3-4 (2009), p. 157–167.

[18] Cory A HENSON, Josh K PSCHORR, Amit P SHETH et Krishnaprasad THIRUNARAYAN. “SemSOS : Semantic sensor observation service”. In : *Collaborative Technologies and Systems, 2009. CTS’09. International Symposium on*. IEEE. 2009, p. 44–53.

[19] Arne BRÖRING, Patrick MAUÉ, Krzysztof JANOWICZ, Daniel NÜST et Christian MALEWSKI. “Semantically-enabled sensor plug & play for the sensor web”. In : *Sensors* 11.8 (2011), p. 7568–7605.

[20] Mike BOTTS, George PERCIVALL, Carl REED et John DAVIDSON. “OGC® sensor web enablement : Overview and high level architecture”. In : *GeoSensor networks*. Springer, 2008, p. 175–190.

[21] Daniele MIORANDI, Sabrina SICARI, Francesco DE PELLEGRINI et Imrich CHLAMTAC. “Internet of things : Vision, applications and research challenges”. In : *Ad hoc networks* 10.7 (2012), p. 1497–1516.

[22] Shanzhi CHEN, Hui XU, Dake LIU, Bo HU et Hucheng WANG. “A vision of IoT : Applications, challenges, and opportunities with china perspective”. In : *IEEE Internet of Things journal* 1.4 (2014), p. 349–359.

[23] Yongrui QIN, Quan Z SHENG, Nickolas JG FALKNER, Schahram DUSTDAR, Hua WANG et Athanasios V VASILAKOS. “When things matter : A data-centric view of the internet of things”. In : *arXiv preprint arXiv :1407.2704* (2014).

[24] Dominique GUINARD, Vlad TRIFA et Erik WILDE. “A resource oriented architecture for the web of things”. In : *Internet of Things (IOT), 2010*. IEEE. 2010, p. 1–8.

[25] Deze ZENG, Song GUO et Zixue CHENG. “The web of things : A survey”. In : *JCM* 6.6 (2011), p. 424–438.

[26] Dennis PFISTERER, Kay ROMER, Daniel BIMSCHAS, Oliver KLEINE, Richard MIETZ, Cuong TRUONG, Henning HASEMANN, Alexander KRÖLLER, Max PAGEL, Manfred HAUSWIRTH et al. “SPITFIRE : toward a semantic web of things”. In : *IEEE Communications Magazine* 49.11 (2011), p. 40–48.

[27] Michele RUTA, Floriano SCIOSCIA et Eugenio DI SCIASCIO. “Enabling the Semantic Web of Things : framework and architecture”. In : *2012 IEEE Sixth International Conference on Semantic Computing*. IEEE. 2012, p. 345–347.

[28] Christian BIZER, Tom HEATH et Tim BERNERS-LEE. “Linked data : The story so far”. In : *Semantic services, interoperability and web applications : emerging concepts*. IGI Global, 2011, p. 205–227.

[29] Payam BARNAGHI, Mirko PRESSER et Klaus MOESSNER. “Publishing linked sensor data”. In : *CEUR Workshop Proceedings : Proceedings of the 3rd International Workshop on Semantic Sensor Networks (SSN), Organised in conjunction with the International Semantic Web Conference*. T. 668. 2010.

[30] Josh PSCHORR, Cory Andrew HENSON, Harshal Kamlesh PATNI et Amit P SHETH. *Sensor discovery on linked data*. Rapp. tech. 2010. URL : https://corescholar.libraries.wright.edu/knoesis/780.

[31] Sasikanth AVANCHA, Chintan PATEL, Anupam JOSHI et al. “Ontology-driven adaptive sensor networks”. In : *MobiQuitous 2004* (2004), p. 194–202.

[32] David J RUSSOMANNO, Cartik KOTHARI et Omoju THOMAS. “Sensor ontologies : from shallow to deep models”. In : *System Theory, 2005. SSST’05. Proceedings of the Thirty-Seventh Southeastern Symposium on*. IEEE. 2005, p. 107–112.

[33] David J RUSSOMANNO, Cartik R KOTHARI et Omoju A THOMAS. “Building a Sensor Ontology : A Practical Approach Leveraging ISO and OGC Models.” In : *IC-AI*. 2005, p. 637–643.

[34] Mohamad EID, Ramiro LISCANO et Abdulmotaleb EL SADDIK. “A novel ontology for sensor networks data”. In : *Computational Intelligence for Measurement Systems and Applications, Proceedings of 2006 IEEE International Conference on*. IEEE. 2006, p. 75–79.

[35] Mohamad EID, Ramiro LISCANO et Abdulmotaleb EL SADDIK. “A universal ontology for sensor networks data”. In : *Computational Intelligence for Measurement Systems and Applications, 2007. CIMSA 2007. IEEE International Conference on*. IEEE. 2007, p. 59–62.

[36] Luis BERMUDEZ, Eric DELORY, Tom O’REILLY et Joaquin del RIO FERNANDEZ. “Ocean observing systems demystified”. In : *OCEANS 2009, MTS/IEEE Biloxi-Marine Technology for Our Future : Global and Local Challenges*. IEEE. 2009, p. 1–7.

[37] Holger NEUHAUS et Michael COMPTON. “The semantic sensor network ontology”. In : *AGILE workshop on challenges in geospatial data harmonisation, Hannover, Germany*. 2009, p. 1–33.

[38] Michael COMPTON, Holger NEUHAUS, Kerry TAYLOR et Khoi-Nguyen TRAN. “Reasoning about sensors and compositions”. In : *Proceedings of the 2nd International Conference on Semantic Sensor Networks-Volume 522*. Citeseer. 2009, p. 33–48.

[39] Matt CALDER, Robert A MORRIS et Francesco PERI. “Machine reasoning about anomalous sensor data”. In : *Ecological Informatics* 5.1 (2010), p. 9–18.

[40] Laurent LEFORT, Cory HENSON et Kerry TAYLOR. *Semantic Sensor Network XG Final Report*. W3C Incubator Group Report. W3C, juin 2011. URL : http://www.w3.org/2005/Incubator/ssn/XGR-ssn-20110628/.

[41] Armin HALLER, Krzysztof JANOWICZ, Simon J D COX, Danh LE PHUOC, Kerry TAYLOR et Maxime LEFRANÇOIS. *Semantic Sensor Network Ontology*. W3C Recommendation. W3C, oct. 2017. URL : https://www.w3.org/TR/vocab-ssn/.

[42] Eclipse SMARTHOME. https://www.eclipse.org/smarthome/getting-started.html. Accessed : 2018-10-07.

[43] Eclipse IoT Working GROUP et al. *The three software stacks required for iot architectures*. Accessed : 2018-10-07. URL : https://iot.eclipse.org/white-papers/iot-architectures/.

[44] Per PERSSON et Ola ANGELSMARK. "Calvin - Merging Cloud and IoT". In : *ANT/SEIT*. T. 52. Procedia Computer Science. Elsevier, 2015, p. 210–217.

[45] Luciana Moreira Sá de SOUZA, Patrik SPIESS, Dominique GUINARD, Moritz KÖHLER, Stamatis KARNOUSKOS et Domnic SAVIO. "SOCRADES : A Web Service Based Shop Floor Integration Infrastructure". In : *IOT*. T. 4952. Lecture Notes in Computer Science. Springer, 2008, p. 50–67.

[46] Hendrik BOHN, Andreas BOBEK et Frank GOLATOWSKI. "SIRENA - Service Infrastructure for Real-time Embedded Networked Devices : A service oriented framework for different domains". In : *ICN/ICONS/MCL*. IEEE Computer Society, 2006, p. 43.

[47] Xuan Thang NGUYEN, Huu Tam TRAN, Harun BARAKI et Kurt GEIHS. "FRASAD : A framework for model-driven IoT Application Development". In : *WF-IoT*. IEEE Computer Society, 2015, p. 387–392.

[48] Peter MÜLDERS, Stefan GRUNER et Xuan Thang NGUYEN. "Model-driven design plus artificial intelligence for wireless sensor networks software development". In : *SESENA@ICSE*. ACM, 2011, p. 63–64.

[49] Yuna JEONG, Hyuntae JOO, Gyeonghwan HONG, Dongkun SHIN et Sungkil LEE. "AVIoT : web-based interactive authoring and visualization of indoor internet of things". In : *IEEE Trans. Consumer Electronics* 61.3 (2015), p. 295–301.

[50] Michael MRISSA, Lionel MÉDINI, Jean-Paul JAMONT, Nicolas LE SOMMER et Jérôme LAPLACE. "An avatar architecture for the web of things". In : *IEEE Internet Computing* 19.2 (2015), p. 30–38.

[51] Jacob BEAL, Danilo PIANINI et Mirko VIROLI. "Aggregate programming for the internet of things". In : *Computer* 9 (2015), p. 22–30.

[52] Elizabeth LATRONICO, Edward A LEE, Marten LOHSTROH, Chris SHAVER, Armin WASICEK et Matthew WEBER. "A vision of swarmlets". In : *IEEE Internet Computing* 19.2 (2015), p. 20–28.

[53] Vimal SHARMA, Suvodeep DAS et Susheel KEWALEY. "Design Thing'ing : methodology for understanding and discovering Use cases in IoT scenarios". In : *Proceedings of the 7th International Conference on HCI, IndiaHCI 2015*. ACM. 2015, p. 113–115.

[54] Charith PERERA, Ciaran MCCORMICK, Arosha K BANDARA, Blaine A PRICE et Bashar NUSEIBEH. "Privacy-by-design framework for assessing internet of things applications and platforms". In : *Proceedings of the 6th International Conference on the Internet of Things*. ACM. 2016, p. 83–92.

[55] Franco ZAMBONELLI. "Toward sociotechnical urban superorganisms". In : *Computer* 45.8 (2012), p. 76–78.

[56] Sara HACHEM, Animesh PATHAK et Valerie ISSARNY. "Service-oriented middleware for large-scale mobile participatory sensing". In : *Pervasive and Mobile Computing* 10 (2014), p. 66–82.

[57] Dries HARNIE, Theo D'HONDT, Elisa Gonzalez BOIX et Wolfgang DE MEUTER. "Programming urban-area applications for mobility services". In : *ACM Transactions on Autonomous and Adaptive Systems* 9.2 (2014).

[58] Andrea SASSI et Franco ZAMBONELLI. "Coordination infrastructures for future smart social mobility services". In : *IEEE Intelligent Systems* 29.5 (2014), p. 78–82.

[59] Franco ZAMBONELLI. "Towards a general software engineering methodology for the Internet of Things". In : *arXiv preprint arXiv :1601.05569* (2016).

[60] Dirk SLAMA, Frank PUHLMANN, Jim MORRISH et Rishi M BHATNAGAR. *Enterprise IoT : Strategies and Best practices for connected products and services*. " O'Reilly Media, Inc.", 2015.

[61] Ian F AKYILDIZ, Weilian SU, Yogesh SANKARASUBRAMANIAM et Erdal CAYIRCI. "Wireless sensor networks : a survey". In : *Computer networks* 38.4 (2002), p. 393–422.

[62] Rekha KS et TH SREENIVAS. "A Review on Run-Time Reconfigurations and Code Update Mechanisms in Wireless Sensor Networks". In : *International Journal of Innovative Research in Computer and Communication Engineering* 3.2 (2015), p. 1015–1032.

[63] Amirhosein TAHERKORDI. "Programming Wireless Sensor Networks : From Static to Adaptive Models". Thèse de doct. UNIVERSITY OF OSLO, 2011.

[64] Ian F AKYILDIZ et Ismail H KASIMOGLU. "Wireless sensor and actor networks : research challenges". In : *Ad hoc networks* 2.4 (2004), p. 351–367.

[65] Tim WARK, Christopher CROSSMAN, Wen HU, Ying GUO, Philip VALENCIA, Pavan SIKKA, Peter I. CORKE, Caroline LEE, John HENSHALL, Kishore PRAYAGA, Julian O'GRADY, Matt REED et Andrew FISHER. "The design and evaluation of a mobile sensor/actuator network for autonomous animal control". In : *Proceedings of the 6th International Conference on Information Processing in Sensor Networks, IPSN 2007, Cambridge, Massachusetts, USA, April 25-27, 2007*. 2007, p. 206–215.

[66] Qing CAO, Tarek ABDELZAHER, John STANKOVIC, Kamin WHITEHOUSE et Liqian LUO. "Declarative tracepoints : a programmable and application independent debugging system for wireless sensor networks". In : *Proceedings of the 6th ACM conference on Embedded network sensor systems*. ACM. 2008, p. 85–98.

[67] Paolo COSTA, Geoff COULSON, Richard GOLD, Manish LAD, Cecilia MASCOLO, Luca MOTTOLA, Gian Pietro PICCO, Thirunavukkarasu SIVAHARAN, Nirmal WEERASINGHE et Stefanos ZACHARIADIS. "The RUNES middleware for networked embedded systems and its application in a disaster management scenario". In : *Pervasive Computing and Communications, 2007. PerCom'07. Fifth Annual IEEE International Conference on*. IEEE. 2007, p. 69–78.

[68] S BROWN et CJ SREENAN. "Updating software in wireless sensor networks : A survey". In : *Dept. of Computer Science, National Univ. of Ireland, Maynooth, Tech. Rep* (2006), p. 1–14.

[69] Stephen BROWN et Cormac J SREENAN. "Software updating in wireless sensor networks : A survey and lacunae". In : *Journal of Sensor and Actuator Networks* 2.4 (2013), p. 717–760.

[70] Xiang LI et Sangman MOH. "Middleware systems for wireless sensor networks : A comparative survey". In : *Contemporary Engineering Sciences* 7.13 (2014), p. 649–660.

[71] Adam DUNKELS, Niclas FINNE, Joakim ERIKSSON et Thiemo VOIGT. "Run-time dynamic linking for reprogramming wireless sensor networks". In : *Proceedings of the 4th international conference on Embedded networked sensor systems*. ACM. 2006, p. 15–28.

[72] Waqaas MUNAWAR, Olaf LANDSIEDEL, Muhammad Hamad ALIZAI et Klaus WEHRLE. "Remote incremental adaptation of sensor network applications". In : *Proc. of the 8th GI/ITG KuVS Fachgespräch" Wireless Sensor Networks"(FGSN'09)* (2009).

[73] Jonathan W HUI et David CULLER. "The dynamic behavior of a data dissemination protocol for network programming at scale". In : *Proceedings of the 2nd international conference on Embedded networked sensor systems*. ACM. 2004, p. 81–94.

[74] Waqaas MUNAWAR, Muhammad Hamad ALIZAI, Olaf LANDSIEDEL et Klaus WEHRLE. "Dynamic TinyOS : Modular and Transparent Incremental Code-Updates for Sensor Networks". In : *Proceedings of IEEE International Conference on Communications, ICC 2010, Cape Town, South Africa, 23-27 May 2010*. 2010, p. 1–6. DOI : 10.1109/ICC.2010.5501964. URL : https://doi.org/10.1109/ICC.2010.5501964.

[75] Chih-Chieh HAN, Ram KUMAR, Roy SHEA, Eddie KOHLER et Mani SRIVASTAVA. "A dynamic operating system for sensor nodes". In : *Proceedings of the 3rd international conference on Mobile systems, applications, and services*. ACM. 2005, p. 163–176.

[76] Jaein JEONG et David CULLER. "Incremental network programming for wireless sensors". In : *International Journal of Communications, Network and System Sciences* 2.05 (2009), p. 433.

[77] Niels REIJERS et Koen LANGENDOEN. "Efficient code distribution in wireless sensor networks". In : *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*. ACM. 2003, p. 60–67.

[78] Paul GRACE, Geoff COULSON, Gordon BLAIR, Barry PORTER et Danny HUGHES. "Dynamic reconfiguration in sensor middleware". In : *Proceedings of the international workshop on Middleware for sensor networks*. ACM. 2006, p. 1–6.

[79] Luca MOTTOLA, Gian Pietro PICCO et Adil Amjad SHEIKH. "FiGaRo : Fine-Grained Software Reconfiguration for Wireless Sensor Networks". In : *Wireless Sensor Networks, 5th European Conference, EWSN 2008, Bologna, Italy, January 30-February 1, 2008, Proceedings*. 2008, p. 286–304. DOI : 10.1007/978-3-540-77690-1_18. URL : https://doi.org/10.1007/978-3-540-77690-1%5C_18.

[80] Danny HUGHES, Klaas THOELEN, Wouter HORRÉ, Nelson MATTHYS, Javier Del CID, Sam MICHIELS, Christophe HUYGENS et Wouter JOOSEN. "LooCI : A Loosely-coupled Component Infrastructure for Networked Embedded Systems". In : *Proceedings of the 7th International Conference on Advances in Mobile Computing and Multimedia*. MoMM '09. Kuala Lumpur, Malaysia : ACM, 2009, p. 195–203. ISBN : 978-1-60558-659-5. DOI : 10.1145/1821748.1821787. URL : http://doi.acm.org/10.1145/1821748.1821787.

[81] Pedro José MARRÓN, Matthias GAUGER, Andreas LACHENMANN, Daniel MINDER, Olga SAUKH et Kurt ROTHERMEL. "FlexCup : A Flexible and Efficient Code Update Mechanism for Sensor Networks". In : *Proceedings of the Third European Conference on Wireless Sensor Networks*. EWSN'06. Zurich, Switzerland : Springer-Verlag, 2006, p. 212–227. ISBN : 3-540-32158-6, 978-3-540-32158-3. DOI : 10.1007/11669463_17. URL : http://dx.doi.org/10.1007/11669463_17.

[82] Geoff COULSON, Gordon BLAIR, Paul GRACE, Francois TAIANI, Ackbar JOOLIA, Kevin LEE, Jo UEYAMA et Thirunavukkarasu SIVAHARAN. "A Generic Component Model for Building Systems Software". In : *ACM Trans. Comput. Syst.* 26.1 (mar. 2008), 1 :1–1 :42. ISSN : 0734-2071. DOI : 10.1145/1328671.1328672. URL : http://doi.acm.org/10.1145/1328671.1328672.

[83] Jean-Philippe FASSINO, Jean-Bernard STEFANI, Julia L LAWALL et Gilles MULLER. “Think : A Software Framework for Component-based Operating System Kernels.” In : *USENIX Annual Technical Conference, General Track*. 2002, p. 73–86.

[84] Eric BRUNETON, Thierry COUPAYE, Matthieu LECLERCQ, Vivien QUÉMA et Jean-Bernard STEFANI. “The fractal component model and its support in java”. In : *Software : Practice and Experience* 36.11-12 (2006), p. 1257–1284.

[85] Frédéric LOIRET, Juan NAVAS, Jean-Philippe BABAU et Olivier LOBRY. “Component-based real-time operating system for embedded applications”. In : *International Symposium on Component-Based Software Engineering*. Springer. 2009, p. 209–226.

[86] Philip LEVIS et David CULLER. “Maté : A tiny virtual machine for sensor networks”. In : *ACM Sigplan Notices*. T. 37. 10. ACM. 2002, p. 85–95.

[87] Amirhosein TAHERKORDI, Quan LETRUNG, Romain ROUVOY et Frank ELIASSEN. “WiSeKit : A distributed middleware to support application-level adaptation in sensor networks”. In : *IFIP International Conference on Distributed Applications and Interoperable Systems*. Springer. 2009, p. 44–58.

[88] Amirhosein TAHERKORDI, Romain ROUVOY, Quan LE-TRUNG et Frank ELIASSEN. “Supporting lightweight adaptations in context-aware wireless sensor networks”. In : *Proceedings of the 1st International Workshop on Context-Aware Middleware and Services : affiliated with the 4th International Conference on Communication System Software and Middleware (COMSWARE 2009)*. ACM. 2009, p. 43–48.

[89] Amir TAHERKORDI, Frederic LOIRET, Romain ROUVOY et Frank ELIASSEN. “Optimizing sensor network reprogramming via in situ reconfigurable components”. In : *ACM Transactions on Sensor Networks (TOSN)* 9.2 (2013), p. 14.

[90] Danh LE-PHUOC et Manfred HAUSWIRTH. “Linked Open Data in Sensor Data Mashups”. In : *Proceedings of the 2Nd International Conference on Semantic Sensor Networks - Volume 522*. SSN’09. Washington DC : CEUR-WS.org, 2009, p. 1–16. URL : http://dl.acm.org/citation.cfm?id=2889933.2889934.

[91] Stephane GERVAIS-DUCOURET. “Next Smart Sensors Generation”. In : *Conference : Sensors Applications Symposium*. IEEE. 2011.

[92] Mari Carmen SUÁREZ-FIGUEROA, Asunción GÓMEZ-PÉREZ et Mariano FERNÁNDEZ-LÓPEZ. “The NeOn methodology for ontology engineering”. In : *Ontology engineering in a networked world*. Springer, 2012, p. 9–34.

[93] Busayawan ARIYATUM et Ray HOLLAND. “A strategic approach to new product development in smart clothing”. In : *Proceedings of the 6th Asian Design Conference*. T. 70. Citeseer. 2003.

[94] Qiu CHUNYAN et Yue HU. “The Review of Smart Clothing Design Research based on the Concept of 3F+ 1I”. In : *International Journal of Business and Social Science* 6.1 (2015).

[95] Willem Nico BORST et WN BORST. “Construction of engineering ontologies for knowledge sharing and reuse”. In : (1997).

[96] Mathieu D’AQUIN, Anne SCHLICHT, Heiner STUCKENSCHMIDT et Marta SABOU. “Ontology modularization for knowledge selection : Experiments and evaluations”. In : *International Conference on Database and Expert Systems Applications*. Springer. 2007, p. 874–883.

[97] Rodolfo STECHER, Claudia NIEDERÉE, Wolfgang NEJDL et Paolo BOUQUET. “Adaptive ontology re-use : finding and re-using sub-ontologies”. In : *International Journal of Web Information Systems* 4.2 (2008), p. 198–214.

[98] Sebastian KAEBISCH et Takuki KAMIYA. *Web of Things (WoT) Thing Description*. First Public Working Draft. W3C, sept. 2017. URL : https://www.w3.org/TR/2017/WD-wot-thing-description-20170914/.

[99] Maxime LEFRANÇOIS. "Planned ETSI SAREF Extensions based on the W3C&OGC SOSA/SSN-compatible SEAS Ontology Patterns". In : *Proceedings of Workshop on Semantic Interoperability and Standardization in the IoT, SIS-IoT,* juil. 2017.

[100] Xavier AIMÉ, Sophie GEORGE et Jeremy HORNUNG. "VetiVoc : a modular ontology for the fashion, textile and clothing domain". In : *Applied Ontology* 11.1 (2016), p. 1–28.

[101] Jasmin GUTH, Uwe BREITENBÜCHER, Michael FALKENTHAL, Frank LEYMANN et Lukas REINFURT. "Comparison of IoT platform architectures : A field study based on a reference architecture". In : *Cloudification of the Internet of Things (CIoT)*. IEEE. 2016, p. 1–6.

..

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Titre : Gestion intelligente de réseaux de capteurs, intégrés à des vêtements sportifs instrumentés.

Mots clés : Internet des objets, objet intelligents, Web sémantique, ontologie modulaire, Framework sémantique.

Résumé : L'Internet des Objets (IdO) intègre les réseaux de capteurs à Internet, et ouvre la voie pour des systèmes ou des écosystèmes ayant pour but d'aider les gens à vivre dans des mondes à la fois physiques et cybernétiques. L'IdO offre l'omniprésence d'objets capables d'interagir les uns avec les autres et de coopérer avec leurs voisins pour atteindre des objectifs communs. Ces objets dits "Intelligents" (OI), peuvent détecter l'environnement et communiquer avec d'autres objets. La création d'OI et de systèmes d'IdO fait intervenir des acteurs d'expertises très diverses. Ainsi, il devient indispensable d'avoir des descriptions standardisées et sémantiques pour résoudre les problèmes liés à l'interopérabilité et l'hétérogénéité sémantique entre les différentes ressources disponibles d'une part, et entre les différents intervenants à la conception/fabrication des OI, d'autre part. De ce fait, nous avons proposé le Framework sémantique et générique FSMS, structuré en un ensemble de modules ontologiques pour la conception/fabrication d'un OI. Une méthodologie de support à ce Framework a été proposée. Elle se fonde sur les mêmes modules ontologiques identifiés dans la composante sémantique du FSMS. Ces modules ontologiques forment l'ontologie SMS pierre angulaire de cette thèse. Un processus générique basé sur une description sémantique des composants structurels et comportementaux d'un OI a été également proposé en vue d'une gestion intelligente de la conception d'un OI. Ce processus a ensuite été mis en application pour des Vêtements Intelligents de sport. Un OI étant destiné à être réutilisé à différents contextes d'usage, une approche de reconfiguration/adaptation du fonctionnement de l'OI a été proposée. Celle-ci trouve à son tour son fondement dans l'ontologie modulaire SMS.

Title : Intelligent management of sensor networks, integrated into instrumented sport clothing.

Keywords : Internet of things, smart objects, semantic Web, modular ontology, semantic framework.

Abstract : The Internet of Things (IoT) integrates sensor networks with the Internet, and paves the way for systems or ecosystems to help people live in both physical and cyber worlds. IoT offers the ubiquity of objects that are able to interact with each other and cooperate with their neighbors to achieve common goals. These objects, called "Smart" (SO), can detect the environment and communicate with other objects. The creation of SO and IoT system involves actors of very diverse expertise. Hence, it becomes essential to have standardized and semantic descriptions to solve the problems related to the interoperability and the semantic heterogeneity between the different available resources on the one hand, and between the different stakeholders designing/manufacturing the SO, on the other hand. Therefore, we have proposed the FSMS semantic and generic framework, which is structured into a set of ontological modules to design/manufacture a given SO. A support methodology for this framework has been equally proposed. It is based on the same ontological modules identified in the semantic component of the FSMS. These ontological modules form the SMS ontology that is proposed and constitutes the cornerstone of this thesis. In order to intelligently manage an SO design, we proposed a generic process based on a semantic description of the structural and behavioral components of an SO. This process was thereafeter implemented for Smart Clothing of sports. This Sportswear is intended to be used in different contexts of use, an approach to reconfiguration/adaptation of the operation of the Smart Clothing has been proposed. This one is also based on the modular ontology SMS.

.