Approche sémantique de la conception de services connectés : cadre d'architecture, algorithmique de composition, application à la maison connectée

Rayhana Bouali Baghli

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

TELECOM ParisTech

2017-ENST-0072

EDITE - ED 130

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

TELECOM ParisTech

Spécialité « Informatique et Réseaux »

présentée et soutenue publiquement par

Rayhana BOUALI BAGHLI

le 15 décembre 2017

Approche sémantique de la conception de services connectés - cadre d'architecture, algorithmique de composition, application à la maison connectée

Directeur de thèse : **M. Elie NAJM**
Co-encadrant de thèse : **M. Bruno TRAVERSON**

Jury

M. Gérard MEMMI, Professeur, TELECOM ParisTech	Président du jury
M. Alessandro FANTECHI, Professeur des Universités, Universita degli Studi di Firenze, Italie	Rapporteur
Mme Zoubida KEDAD, Maître de Conférences HDR, Université de Versailles Saint-Quentin-en-Yvelines	Rapporteur
Mme Nawal GUERMOUCHE, Maître de Conférences, INSA de Toulouse	Examinateur
M. Philippe FUTTERSACK, Ingénieur Chercheur, Département Icame, EDF R&D	Invité

T H È S E

TELECOM ParisTech
école de l'Institut Mines-Télécom - membre de ParisTech
46 rue Barrault 75013 Paris - (+33) 1 45 81 77 77 - www.telecom-paristech.fr

Remerciements

"Soyons reconnaissants aux personnes qui nous donnent du bonheur ;
elles sont les charmants jardiniers par qui nos âmes sont fleuries."
Marcel Proust.

Ces années de thèse ont été pour le moins intenses, intellectuellement et humainement parlant. L'aboutissement de cette thèse n'aurait pu être possible sans l'aide et le soutien de nombreuses personnes à qui je tiens à exprimer ma gratitude.

Je voudrais tout d'abord remercier Elie NAJM, mon directeur de thèse, pour le temps qu'il a consacré à mon travail de thèse. Tout au long de la thèse, il n'a eu cesse de m'accompagner et d'enrichir mon travail par son expérience et ses conseils avisés. Je le remercie pour les nombreuses relectures et corrections, même dans la phase de finalisation du rapport qui a été particulièrement intense.

Je remercie Bruno TRAVERSON, mon encadrant de thèse de m'avoir accompagnée durant tout ce travail de thèse. Je le remercie d'avoir toujours été à l'écoute de mes questions et de mes idées. Je le remercie pour toutes les fois où il m'a soutenue et motivée, même dans les moments les plus difficiles. Merci d'avoir cru en moi et de m'avoir fait confiance jusqu'au bout.

Je remercie les rapporteurs de ma thèse, Alessandro FANTECHI et Zoubida KEDAD. Je suis très honorée du temps qu'ils ont consacré à la relecture et à l'évaluation approfondie de mon travail de thèse. Je remercie mes examinateurs de thèse, Gérard MEMMI et Nawal GUERMOUCHE, qui m'honorent de leur présence et qui m'éclairent par leurs remarques constructives. Mes remerciements vont également à Philippe FUTTERSACK pour avoir accepté de participer à ce jury de thèse.

Je remercie le laboratoire SEIDO qui a financé cette thèse. Je remercie tous ces membres avec qui j'ai eu beaucoup de plaisir à travailler et à collaborer.

Je remercie l'ensemble des enseignants-chercheurs, des ingénieurs-chercheurs, des post-doctorants, des doctorants et des stagiaires des deux départements qui m'ont accueillie pour cette thèse, Infres de Télécom ParisTech et Icame d'EDF R&D. Merci pour les nombreux moments passés ensemble.

Enfin je remercie ma chère famille sans qui je n'aurais pas pu mener ce travail. Chacun de vous est pour moi une source inépuisable d'inspiration. Merci pour vos relectures. Merci pour vos encouragements et pour votre soutien sans faille. Cette thèse et moi vous devons beaucoup. Merci.

À ma famille que j'aime...

Résumé

Dans le contexte de l'Internet des Objets, la conception de services connectés – c'est-à-dire de services portés par des objets connectés – nécessite une approche de bout en bout pour non seulement répondre aux attentes des bénéficiaires de ces services mais aussi pour adapter le fonctionnement de ces services à des conditions d'exécution très variées allant de la maison à la ville connectée.

L'approche sémantique proposée par cette thèse offre un niveau d'abstraction qui permet aux concepteurs de services de se concentrer sur les aspects fonctionnels des services et des objets. Elle s'inscrit dans un cadre d'architecture plus large qui aborde, en plus de ce niveau sémantique, les aspects plus opérationnels de mise en œuvre de ces services (niveau Artefacts) dans des environnements techniques éventuellement hétérogènes (niveau Ressources).

En proposant cette approche sémantique de conception, la thèse vise plusieurs objectifs qui peuvent être regroupés en trois catégories. La première catégorie d'objectifs est de décloisonner le monde actuel des services connectés en découplant les services des objets connectés et en permettant le partage d'objets par plusieurs services connectés. L'ouverture induite par ces premiers objectifs conduit à viser une deuxième catégorie d'objectifs qui a trait à la composition des services connectés. Chaque service devra être conscient et adopter un comportement compatible avec les autres éléments de son contexte d'exécution. Ces éléments de contexte comprennent bien sûr les autres services mais aussi les phénomènes physiques et les actions des occupants des espaces concernés. Enfin, la troisième catégorie d'objectifs s'adresse plus particulièrement aux bénéficiaires des services connectés afin d'optimiser l'expérience utilisateur par des attentes mieux prises en compte et des automatismes respectueux des comportements humains.

Le fondement théorique de l'approche sémantique proposée dans cette thèse s'appuie sur un méta-modèle qui permet de définir les éléments de modélisation nécessaires pour modéliser les services, les objets connectés et les comportements des services sous forme déclarative. Tout d'abord, la notion d'état objectif permet de déclarer l'intention du service. En complément, le service permet de regrouper plusieurs états susceptibles de satisfaire une même intention. La notion de liste d'états objectif permet de déclarer des alternatives classées par ordre de priorité qui satisfont l'intention du service. Enfin, la notion d'état contextualisé prend en compte l'agent (service, humain, phénomène physique)

responsable de l'état courant d'un objet connecté. Il permet au service d'agir de manière adaptée en fonction de l'agent responsable de l'état courant d'un objet.

Pour montrer son aptitude à adresser les problématiques des services connectés, le méta-modèle sémantique a été utilisé sur un scénario de mise en place d'un service de régulation de température dans une pièce d'une maison connectée. Chaque étape est détaillée de la sélection du service et des objets connectés, à leurs instanciations dans ce contexte d'exécution, la mise en correspondance des instances et l'adaptation des règles de fonctionnement au contexte d'exécution. De plus, une formalisation du méta-modèle sémantique dans le langage de spécification formelle Alloy a permis de détecter des incohérences dans un ensemble de règles de comportement de services connectés. Enfin, une mise en œuvre du méta-modèle sous forme d'une ontologie OWL et de règles SWRL a permis une simulation de la gestion des conflits d'accès aux objets partagés dans le cas d'un bouquet de plusieurs services et l'intérêt d'un algorithme de composition fondé sur les priorités et l'optimisation de l'atteinte des objectifs de chaque service du bouquet.

Cette thèse propose une approche sémantique à la conception de services connectés sous la forme d'un méta-modèle intégrant tous les éléments de modélisation permettant d'exprimer les aspects structurels et comportementaux des services connectés en s'abstrayant des considérations plus opérationnelles et techniques. L'approche a été validée de façon expérimentale par plusieurs cas d'utilisation illustrant des situations de contextualisation d'un service, de vérification de la cohérence d'un ensemble de règles de comportement et de la gestion des conflits et des oscillations dans le cas de composition de services. Elle a permis aussi d'ouvrir des perspectives au niveau de la mise en œuvre des services connectés de façon opérationnelle, de la perception par les bénéficiaires des services vers une meilleure collaboration et sur l'application de l'approche sémantique à d'autres domaines tels que le bâtiment et la ville connectée.

Abstract

In the context of the Internet of Things, the design of connected services - that is, services supported by connected objects - requires an end-to-end approach to not only meet the expectations of the recipients of these services but also to adapt the operation of these services to a wide range of execution conditions spreading from smart homes to smart cities.

The semantic approach proposed in this thesis provides a level of abstraction that allows service designers to focus on the functional aspects of services and objects. It is part of a larger architecture framework that addresses, in addition to this semantic level, the more operational aspects of implementation of these services (Artifacts level) in potentially heterogeneous technical environments (Resources level).

By proposing this semantic design approach, the thesis aims at achieving several objectives that can be grouped into three categories. The first category of objectives is to decompartmentalize the current world of connected services by decoupling services from connected objects and allowing the sharing of objects by several connected services. The openness induced by these first objectives leads to a second category of objectives that relates to the composition of connected services. Each service will have to be aware and adopt a behavior compatible with the other elements of its execution context. These contextual elements include of course the other services but also the physical phenomena and the actions of the occupants of the spaces concerned. Finally, the third category of objectives focusses on the recipients of connected services in order to optimize the user experience through better requirement management and automatisms respectful of human behaviors.

The theoretical basis of the semantic approach proposed in this thesis is a meta-model that defines the modeling elements needed to model services, connected objects and service behaviors in a declarative form. First of all, the notion of objective state allows declaring the intention of the service. In addition, the notion of service makes it possible to gather several states capable of satisfying the same intention. The concept of a list of objective states makes it possible to declare alternatives ranked in order of priority which satisfy the intention of the service. Finally, the notion of contextualized state takes into account the agent (service, human, physical phenomenon) responsible for the current state of a connected object. It allows the service to act appropriately according to the agent responsible for the current state of an object.

To show its ability to address the problems of connected services, the semantic meta-model was used on a scenario of setting up a temperature control service in a room of a

smart home. Each step is detailed ; from the selection of the service, and the connected objects, to their instantiations in this execution context, the mapping of the instances and the adaptation of the operating rules to the execution context. In addition, a formalization of the semantic meta-model in the Alloy formal specification language allows for the detection of inconsistencies in a set of connected service behavior rules. Finally, an implementation of the meta-model in the form of an OWL ontology and SWRL rules allows for a simulation of the shared object access conflicts management in the case of a composition of several services. This implementation shows also the interest of a composition algorithm based on the priorities and the optimization of the achievement of the objectives of each service of the composition.

This thesis proposes a semantic approach to the design of connected services in the form of a meta-model integrating all the elements of modeling allowing to express the structural and behavioral aspects of the connected services in abstracting more operational and technical considerations. The approach has been experimentally validated by several use cases illustrating situations of contextualization of a service, verification of the consistency of a set of rules of behavior and the management of conflicts and oscillations in the service composition case. It has also opened perspectives for the implementation of connected services in an operational way, from the perception by the service users towards a better collaboration between services. It has opened other perspectives on the application of the semantic approach to other domains such as smart buildings and smart cities.

Table des matières

Liste des illustrations

Première partie

Introduction générale

1 Introduction

Nous introduisons notre travail de thèse en présentant, d'abord, le contexte académique et industriel dans lequel s'inscrit cette thèse. Puis, nous déclinons l'organisation du manuscrit de thèse en précisant le périmètre scientifique dans lequel s'inscrivent nos travaux de thèse.

1.1 Contexte de travail

Notre travail de thèse s'inscrit dans le cadre du laboratoire SEIDO (cybersécurité et internet des objets) [1], un laboratoire commun entre EDF R&D et Télécom ParisTech. La R&D d'EDF et Télécom ParisTech collaborent depuis 2012 dans le cadre du laboratoire SEIDO dont les travaux portent sur la thématique de l'Internet des Objets (IdO) et la Cybersécurité pour les systèmes électriques.

Dans le but de répondre aux problématiques de l'Internet des Objets et de la cybersécurité, le laboratoire a pour vocation de réunir des chercheurs du monde académique et du monde industriel disposant de compétences complémentaires.

Ce laboratoire constitue une occasion de réunir les spécialistes de domaines disciplinaires hétérogènes (sécurité, réseaux, Internet, monde énergétique, réseaux électriques, etc.) autour de finalités scientifiques et industrielles qui les rassemblent. Il apporte des avancées en matière de solutions de sécurisation des systèmes d'information, de contrôle et de commande pour les objets connectés.

Il contribue à mieux observer, contrôler et commander les objets du système électrique, plus vite, plus sûrement et en tenant compte de leur contexte d'usage, en respectant la confidentialité des échanges.

Son enjeu est de préparer et faciliter le déploiement de services de gestion de la demande énergétique et d'efficacité énergétique s'appuyant sur l'interopérabilité d'objets énergétiques et ainsi contribuer à assurer la cohérence de l'ensemble du système.

Les principaux objectifs du laboratoire sont [1] :

— de contribuer à la conception d'architectures informatiques distribuées, à large échelle, capables de surveiller et de gérer les systèmes de distribution de l'électricité jusqu'aux équipements terminaux en incluant la production locale et le stockage décentralisés ;

— d'amener les résultats d'études sur des plateformes de simulation et de démonstration ;

— de concourir à l'obtention de nouveaux concepts, de nouveaux standards de définition de nouveaux équipements, de nouveaux logiciels et développer ainsi un patrimoine intellectuel commun.

Les problématiques qui sont abordées par cette thèse s'inscrivent dans l'axe de l'Internet des Objets du laboratoire SEIDO.

1.2 Organisation du rapport

Notre travail de thèse s'inscrit dans le domaine de l'Internet des Objets. Il propose une approche sémantique de la conception et de l'orchestration de services connectés – c'est-à-dire de services portés par des objets connectés.

Pour ce faire, nous présentons notre travail dans ce manuscrit en quatre parties : introduction générale, contribution, application de la contribution et conclusions générales :

La partie I du rapport introduit le travail qui a été mené durant cette thèse en décrivant le contexte du travail, les problématiques abordées et l'état de l'art. Elle se décline en cinq chapitres :

Le chapitre 1 est une introduction à la partie I.

Le chapitre 2 décrit le contexte technique dans lequel s'inscrit notre travail de thèse. La thèse porte sur la conception et l'orchestration de services connectés pour l'Internet des Objets. Ce chapitre permet de mettre en évidence les spécificités techniques qui sont liées à notre domaine de recherches.

Le chapitre 3 décrit les différents verrous scientifiques, technologiques et industriels auxquels nous nous intéressons dans ce travail de thèse. Les verrous sont classés selon leur thématique en trois axes : le comportement cloisonné, la composition de services et l'acceptabilité des services par les utilisateurs.

Le chapitre 4 présente un état de l'art des technologies qui sont relatives aux problématiques que nous abordons dans la thèse. Nous commençons par présenter les techniques desquelles nous nous sommes inspirées et que nous avons utilisées dans la thèse. Ensuite, nous présentons les solutions existantes dans le domaine de l'IdO.

Le chapitre 5 est une synthèse du contenu de la partie I.

La partie II du rapport concerne les contributions qui ont été faites dans cette thèse. Cette partie se décline en six chapitres :

Le chapitre 6 est une introduction à la partie II.

Le chapitre 7 a pour objectif d'analyser les problématiques de définition, de conception et de composition de services dans le but de mettre en place un écosystème de services et d'objets ouvert et composable. Cette analyse prend forme à travers différents scénarios de mise en place, d'exécution et d'orchestration de services et d'objets connectés.

Le chapitre 8 présente de manière informelle une vision globale de l'architecture à trois niveaux dans laquelle s'inscrit notre contribution de thèse.

Le chapitre 9 présente l'approche sémantique proposée par cette thèse. Cette approche offre un niveau d'abstraction qui permet aux concepteurs de services de se concentrer sur les aspects fonctionnels des services et des objets connectés.

Le chapitre 10 Dans ce chapitre, nous présentons une formalisation directe du niveau sémantique. Cette formalisation fait partie d'un article [2] à paraitre et qui sera présenté à la conférence Modelward2018.

Le chapitre 11 synthétise les principales contributions apportées par cette thèse.

La partie III du rapport concerne les différentes applications qui ont été faites de notre contribution. Ces applications permettent de vérifier et d'illustrer l'applicabilité de notre solution. La partie III se décline en cinq chapitres :

Le chapitre 12 est une introduction à la partie III.

Le chapitre 13 décrit un exemple d'application de notre contribution dans le cadre d'une maison connectée. Cet exemple illustre un projet d'un utilisateur qui souhaite équiper une pièce de sa maison de technologies de l'IdO. L'utilisateur sélectionne un service générique et des objets connectés dans un catalogue. Nous illustrons les descriptions du service générique et des objets qui sont sélectionnés par l'utilisateur, puis nous décrivons le service et les objets dans leur version déployée et enfin nous illustrons la mise en correspondance entre le service et les objets.

Le chapitre 14 décrit une mise en œuvre de la contribution qui permet de vérifier la validité des règles d'un service avec le langage et l'outil Alloy. Cet outil s'appuie sur un solver SAT et permet de générer de manière automatique un grand espace (10^{60}) d'instances d'état sur lequel les vérifications de services ont lieu.

Le chapitre 15 illustre une mise en œuvre du méta-modèle sur lequel s'appuie notre contribution sous forme d'une ontologie OWL et de règles SWRL. Cette mise en œuvre a permis d'effectuer une simulation de la gestion des conflits d'accès aux objets partagés dans le cas d'un bouquet de plusieurs services et l'intérêt d'un algorithme d'orchestration fondé sur les priorités ainsi que l'optimisation de l'atteinte des objectifs de chaque service du bouquet.

Le chapitre 16 synthétise et discute les résultats obtenus suite aux différents travaux de mise en mise en œuvre décrits dans cette partie du rapport.

La partie IV synthétise et conclut les travaux qui ont été menés durant cette thèse. Elle se décline en quatre chapitres :

Le chapitre 17 est une introduction à la partie IV.

Le chapitre 18 synthétise les problématiques qui ont été adressées dans cette thèse.

Le chapitre 19 présente les questions qui ont été ouvertes par nos travaux et quelques perspectives de recherche.

Le chapitre 20 conclut la dernière partie de la thèse.

2 Objets et services connectés

"The Internet of Things has the potential to change the world,
just as the Internet did. Maybe even more so."
Kevin Ashton

SOMMAIRE

Ce chapitre décrit le contexte technique dans lequel s'inscrit notre notre travail de thèse. La thèse porte sur la conception et l'orchestration de services connectés pour l'Internet des Objets. Ce chapitre permet de mettre en évidence les spécificités techniques qui sont liées à notre domaine de recherches.

Nous commençons par présenter le domaine de l'Internet des Objets. Nous décrivons ensuite les objets connectés qui font partie de l'IdO. Puis, nous décrivons de manière générale les services et de manière spécifique les service connectés qui s'appuient sur des objets de l'IdO. Nous présentons les particularités des services connectés par rapports aux autres services.

2.1 Internet des Objets

L'expression Internet des Objets (IdO) ou Internet of Things (IoT) en anglais a été prononcée publiquement pour la première fois en 1999 par Kevin Ashton. Dans son article paru au RFiD Journal, Kevin Ashton explique que l'idée première de l'IdO est de permettre aux systèmes informatiques de collecter les données du monde réel dont ils ont besoin de manière autonome et sans intervention des utilisateurs[3]. Plus tard, l'IdO a été décrit de manière plus formelle par l'ITU (International Telecommunication Union) dans le « ITU Internet report » en 2005 [4].

En plus de récolter les données du monde réel, l'IdO permet aux systèmes informatiques d'agir de manière automatique sur les objets du monde réel. Le matériel qui permet de collecter les données du monde réel et d'agir sur le monde réel est désigné par le terme « objets connectés ».

Les auteurs de [5] définissent l'IdO comme étant ce qui permet aux personnes et aux objets d'être connectés n'importe quand, n'importe où, avec n'importe qui, idéalement en utilisant n'importe quel chemin/réseau et n'importe quel service.

L'IdO est défini par [6] comme étant une convergence de plusieurs points de vue : Un point de vue orienté Internet, un point de vue orienté objets et un point de vue orienté sémantique. La figure 2.1 [6] illustre cette convergence. Elle représente les principaux concepts, technologies et normes qui sont relatives à chaque point de vue de l'IdO. Chaque point de vue est axé autour des compétences des différents acteurs qui s'intéressent aux problématiques de l'IdO. Le point de vue orienté Internet est porté par les acteurs télécoms qui considèrent les aspects de connectivité et de communication de l'Internet des objets. Le point de vue orienté objets est porté par les pionniers des constructeurs d'objets connectés -notamment RFiD. Ce point de vue considère l'IdO comme étant des objets génériques qui sont destinés à être intégrés dans une plateforme commune. Du point de vue

orienté sémantique, l'expression Internet des Objets signifie « Un réseau mondial d'objets interconnectés, adressables de manière unique, basé sur des protocoles de communication standards » [7]. Le point de vue orienté sémantique considère les aspects d'adressage unique, de représentation, et de stockage de l'information.

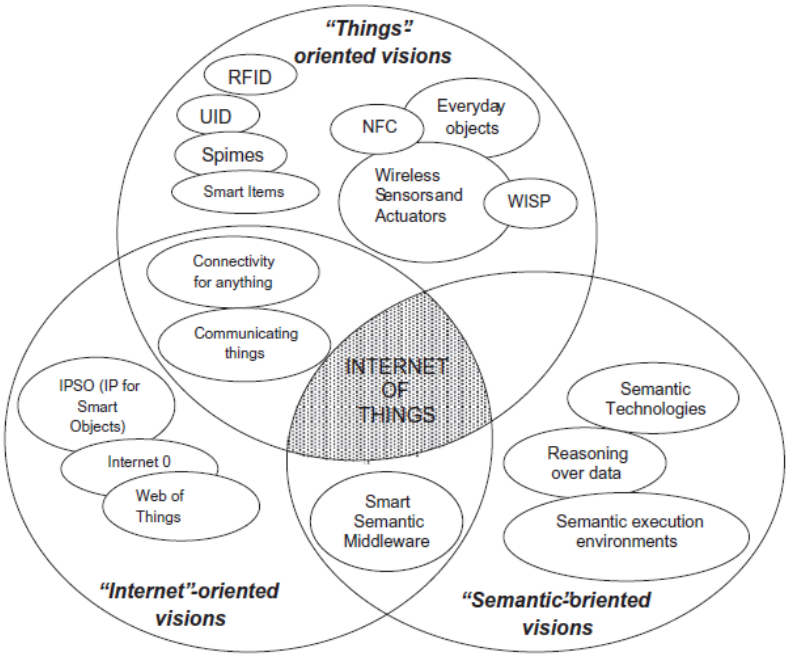


FIGURE 2.1: Paradigme de l'IdO : résultat de convergence de plusieurs points de vue.

Notre travail s'inscrit dans le cadre l'IdO sans se contraindre à un point de vue en particulier.

Nous présentons dans la section suivante les objets qui font partie de l'IdO tels qu'ils sont définis dans la littérature.

2.2 Objets connectés

Les objets de l'IdO sont définis par le Cluster of European research projects on the Internet of Things [8] comme étant des participants actifs dans les processus métier, d'information et sociaux. Il sont capables d'interagir et de communiquer entre eux et avec l'environnement, en échangeant des données et des informations perçues de l'environnement, tout en réagissant de manière autonome sur le monde physique avec ou sans intervention humaine directe. Les interfaces, sous forme de services, facilitent les interactions entre les objets et Internet.

Les objets connectés sont classés en deux catégories : les capteurs et les actionneurs.

2.2.1 Les capteurs

Les capteurs représentent les objets qui ont la capacité de recueillir les données et informations de l'environnement. Ils sont dotés d'interfaces qui permettent aux données collectées d'être transférées sur le réseau.

2.2.2 Les actionneurs

Les actionneurs représentent les objets qui ont la capacité d'agir sur le monde physique en réaction à des données qui ont été perçues.

2.3 Services connectés

De manière informelle, un service est un composant logiciel dont le but est d'offrir des fonctionnalités qui répondent à des besoins spécifiques. Un service est une collection de données et de comportements associés pour accomplir une fonction ou une caractéristique particulière.

Le W3C définit , dans [9], un service comme étant une ressource abstraite qui possède la capacité d'exécuter des tâches. Ces tâches représentent une fonctionnalité cohérente du point de vue des entités qui fournissent le service et du point de vue des entités qui le demandent. Pour être utilisé, ce service doit être réalisé par un agent fournisseur concret.

Le W3C définit un service Web comme étant un système logiciel qui est conçu pour prendre en charge l'interaction M2M (machine-to-machine) sur un réseau. Il a une interface décrite dans un format qui peut être traité par une machine, en particulier WSDL(Web Service Description Language) [10]. Les autres systèmes interagissent avec le service Web d'une manière prescrite par sa description. Ces interactions se font à l'aide de messages SOAP (Simple Object Access Protocol) [11] qui sont généralement transmis via le protocole HTTP(Hypertext Transfer Protocol) [12]. Les messages sont encodés à l'aide du langage XML (eXtensible Markup Language) [13] conjointement avec d'autres normes Web [9].

Aujourd'hui, l'industrie du logiciel s'oriente vers des approches axée sur les services. en particulier, dans le domaine des logiciels d'entreprise, les nouvelles applications complexes reposent sur la composition et la collaboration d'autres services [14].

Les auteurs de [15] présentent une visions de l'Internet des Services (IdS). Cette vision décrit l'idée que, à grande échelle, les applications complexes reposent sur des services. Ces services résident dans différentes couches du système, par ex. entreprise, réseau, ou même au niveau de l'objet connecté en étant mis en oeuvre par un logiciel embarqué.

Les services que nous considérons dans cette thèse sont des services qui s'exécutent dans le but de fournir des fonctionnalités aux utilisateurs. Ces services opèrent sur les objets de l'IdO et sont indépendants des technologies des services Web. Dans la suite de la thèse, nous désignons les services qui font l'objet de notre étude par le terme services IdO ou simplement services.

Les auteurs de [16] classifient les services connectés en deux catégories : des services primaires et des services secondaires. Un service primaire désigne les services qui exposent les fonctionnalités principales d'un objet de l'IdO. Il peut être considéré comme le composant de service de base. Ce composant peut être utilisé par un autre service. Un service secondaire fournit des fonctionnalités supplémentaires à un service primaire ou à d'autres services secondaires. Il peut inclure un ou plusieurs services primaires, il peut aussi inclure un ou plusieurs autres services secondaires.

Les services connectés sont particuliers et possèdent des spécificités par rapport aux autres services. Nous présentons, dans ce qui suit, les particularités les plus importantes des services connectés par rapport aux autres services.

2.3.1 Remontée des données du monde réel vers les services

Les services connectés s'appuient sur des objets connectés qui sont composés de capteurs et d'actionneurs.

Une différence importante entre les services connectés et les autres services informatiques est que ces derniers s'appuient sur des objets qui leur permettent de récolter des données du monde réel de manière automatique. Les services connectés sont indépendants des êtres humains lorsqu'il s'agit de collecter les données de l'environnement. Cette indépendance représente une puissance considérable qui permet aux services connectés de ne pas dépendre des données qui leurs sont fournies par les utilisateurs.

En effet, des données du monde réel représentent des entrées pour les services connectés. Grâce aux capteurs, les services connectés deviennent indépendants des êtres humains et peuvent récolter les données qui servent notamment à décrir le contexte dans lequel ils s'exécutent.

2.3.2 Action des services sur le monde réel

Grâce aux objets connectés, les services peuvent recevoir des données du monde réel et peuvent aussi agir sur le monde réel de manière automatique. Par exemple, un service connecté peut décider de fermer une porte connectée ou d'allumer une lampe connectée de manière automatique sans intervention des êtres humains.

La possibilité d'action des services sur des objets du monde réel signifie que ces services ont un impact direct sur la vie quotidienne de ses utilisateurs. De ce fait, plusieurs besoins découlent. Les services doivent par exemple agir de manière cohérente et coordonnée sur les objets qu'ils gèrent.

2.3.3 Criticité des services et des objets connectés

Les services connectés sont variés et peuvent avoir différents objectifs. Psar exemple, un service de gestion de luminosité dans une pièce a pour objectif de gérer et de régler la luminosité dans la pièce dans laquelle il est déployé. Un service de gestion d'incendie a pour objectif de s'assurer du respect des règles de sécurité en cas de déclenchement d'un incendie. Nous pouvons constater à travers ces deux exemples que les services connectés peuvent avoir différents niveaux de criticité. En cas d'incendie, le service de gestion d'incendie doit être plus prioritaire que le service de gestion de luminosité par exemple. Il est alors important de tenir compte des criticités respectives des services dans un environnement qui est destiné à accueillir plusieurs services en même temps.

Aussi, les services connectés agissent sur des objets du monde réel. Ces objets sont variés et peuvent aller de la lampe connectée au réfrigérateur connecté en passant par un appareil de dialyse. Comme pour les services, au niveau des objets, il peut y avoir des objets dont l'importance est vitale pour l'utilisateur. Il est alors important que les décisions qui sont prises par les services tiennent compte de cette importance.

2.3.4 Cohabitation des technologies et des humains

Les services connectés sont des services qui capturent des données du monde réel et qui agissent sur le monde réel. Ils ont alors un impact sur la vie des êtres humains. Ces services s'exécutent dans le but de fournir des fonctionnalités pour les utilisateurs.

Pour que les services connectés puissent être intégrés dans l'environnement des utilisateurs et être acceptés par ces derniers, ils ne doivent pas être instrusifs. Leur fonctionnement doit être le plus transparent possible aux utilisateurs. Ils doivent s'intégrer dans la vie quotidienne des utilisateurs jusqu'à en être indissocciables [17].

2.3.5 Influence de l'environnement et prise en compte du contexte

Une spécificité importante des services connectés est l'influence de l'environnement sur le comportement de ces services. Cette particularité découle du fait que les services s'appuient sur des objets du monde réel, il est alors possible que l'environnement change

l'état de ces objets. Le comportement des services connectés est fortement influencé par l'environnement et le contexte d'exécution des services.

Les auteurs de [18] définissent le contexte comme étant toute information qui peut être utilisée pour caractériser la situation d'une entité. Une entité est une personne, un lieu ou un objet considéré comme pertinent pour l'interaction entre un utilisateur et une application, y compris l'utilisateur et les applications elles-mêmes.

La prise en compte du contexte peut être identifiée à trois niveaux en fonction de l'interaction des services avec l'utilisateur [19] [20].

— **Personnalisation :** elle permet aux utilisateurs de définir leurs souhaits, leurs préférences et leurs attentes sur le système de manière manuelle. Par exemple, dans le cadre d'une maison connectée équipée d'un service de régulation de température, les utilisateurs peuvent définir une température de consigne. Le service s'exécute alors en ayant comme objectif de maintenir la température spécifiée dans la maison.

— **Prise en compte passive du contexte :** le système surveille en permanence l'environnement et offre les options appropriées aux utilisateurs afin qu'ils puissent prendre des mesures. Par exemple, lorsqu'un utilisateur entre dans un super marché, le téléphone mobile alerte l'utilisateur avec une liste de produits à prix réduit à considérer.

— **Prise en compte active du contexte :** Le système surveille en permanence l'environnement et réagit de manière autonome. Par exemple, dans le cadre d'une maison connectée qui est équipée d'un service de surveillance d'incendie, si les détecteurs de fumée et les capteurs de température détectent un incendie alors, le service avertira automatiquement les pompiers via les méthodes dont il dispose.

Nous avons présenté dans ce chapitre le contexte technique dans lequel s'inscrit notre travail de thèse. Nous avons commencé par présenter l'Internet des Objets, puis, nous avons décrit les objets qui font partie de l'Internet des Objets et enfin, nous nous sommes intéressés aux services connectés pour lesquels nous avons mis en évidence les particularités dont il faudra tenir compte tout au long de la thèse.

Dans le chapitre suivant, nous déterminons les problématiques auxquelles nous nous intéressons dans ce travail.

3 Problématiques posées

"Computing science is about how to solve, with or without machines, the problems posed by the existence of computers."
Edsger Dijkstra

SOMMAIRE

Nous décrivons, dans ce chapitre les différents verrous scientifiques, technologiques et industriels auxquels nous nous intéressons pour notre travail de thèse. Les verrous sont classés selon leur thématique en trois parties : le comportement cloisonné, la composition de services et l'acceptabilité des services par les utilisateurs.

3.1 Comportement cloisonné

La phase d'exécution de services représente les contraintes que nous considérons après le déploiement des services.

3.1.1 Services et objets en silos

En parallèle à leur fonctionnement, nous considérons les objets du point de vue de leurs usages. Un objet peut, en effet, servir pour des usages transverses. Dans le cadre de la mise en place d'un service de sécurité, un ou plusieurs capteurs de présence sont utilisés pour détecter les mouvements et réagir en fonction, en déclenchant une alarme par exemple. Un service d'allumage automatique utilise également un capteur de présence pour détecter les mouvements et allumer ou éteindre la lumière en fonction de la présence ou de l'absence de mouvements. Aujourd'hui, dans l'IdO, il n'existe pas encore de cadre standard approuvé et adopté par la communauté scientifique et par les constructeurs d'objets connectés. Au contraire, la tendance est à produire des objets qui communiquent via des protocoles et des modèles propriétaires, qui sont basés sur des architectures verticales, ce qui conduit à l'apparition de silos technologiques. Dans le modèle actuel en silos, un service de sécurité et un service d'allumage automatique ne peuvent partager le même capteur de présence s'ils ne proviennent pas du même constructeur, mais chaque service aura son propre capteur de présence comme montré en figure 3.1. Une duplication de l'objet « capteur de présence » peut être nécessaire au bon fonctionnement des deux services.

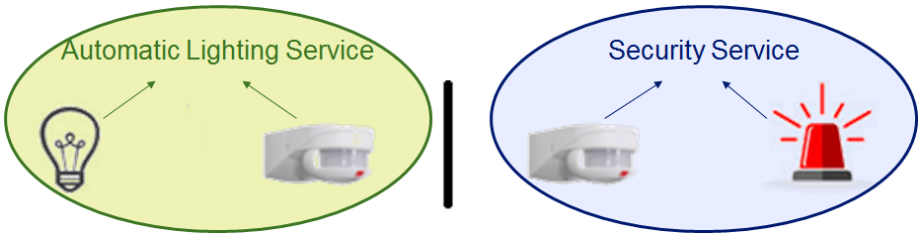


FIGURE 3.1: Redondance d'un capteur de présence dans le cadre d'architectures verticales.

L'architecture qui est présentée en figure 3.1 est une architecture en silos dans l'Internet des Objets. Du point de vue technique, ces silos sont causés par un couplage fort entre les services de l'IdO et les objets connectés. A l'heure actuelle, un service est conçu pour agir sur des objets qui lui sont préalablement connus. Souvent, ces objets sont conçus par le même fabriquant. Les services sont alors fortement liés aux objets qu'ils peuvent interroger et aux objets sur lesquels ils peuvent agir.

Un des défis majeurs de notre étude est de proposer une solution qui découple les objets connectés des services conçus pour l'Internet des Objets. Ce découplage fera en sorte que les services ne soient plus liés à des objets qui proviennent d'un certain constructeur. Un des objectifs de nos recherches est de proposer une solution qui permet de concevoir des services génériques qui sont indépendants des objets connectés. Ces services pourront être mis en correspondance, au moment du déploiement, avec des objets qui proviennent de plusieurs constructeurs et qui possèdent différentes caractéristiques.

3.1.2 Dépendance technique des services aux objets

Les services que nous déployons sont destinés à s'adapter aux objets qui sont présents dans leur contexte d'exécution. Les objets qui composent l'IdO sont très hétérogènes. Ils ont des caractéristiques différentes et communiquent via divers protocoles. A priori, les services n'ont pas connaissances des objets sur lesquels ils seront déployés ni sur leurs caractéristiques techniques.

Chaque objet connecté de l'IdO est doté de capacité, soit de capture de données de son environnement, soit d'actionnement d'un objet physique de son environnement, soit les deux en même temps. L'objet connecté peut également mettre à disposition de son environnement un service s'il possède des capacités de calcul. L'idée principale de notre étude est de faire collaborer les objets connectés afin d'utiliser les fonctionnalités et les services qu'ils offrent dans le but de produire de nouveaux services plus complexes et régis par les besoins des utilisateurs.

La figure 3.2 représente un échantillon d'objets connectés de l'Internet de Objets. Ces objets vont du thermomètre communicant à la voiture communicante, en passant par l'aspirateur communicant. Certains des objets sont dotés d'intelligence et de capacités de calculs, ils peuvent alors effectuer des calculs voire même prendre des décisions localement. Ces objets peuvent offrir, en plus de leur fonctionnalité, un service à leur environnement. D'autres objets ne possèdent aucune intelligence et exécutent uniquement les requêtes qu'ils reçoivent de la part d'autres équipements ou services.

FIGURE 3.2: Échantillon d'objets connectés.

3.2 Composition de services

Dans nos travaux, nous nous intéressons à des services de type services élémentaires mais aussi à des services de type services complexes. Nous désignons par services élémentaires les services de base. Ces services ne sont pas composés par d'autres services et ne pilotent pas eux-mêmes d'autres services.

3.2.1 Conflits d'accès aux objets partagés

Un des points clés de la problématique de composition de services est le partage d'objets par plusieurs services. En effet, notre étude se projette dans un cadre architectural horizontal non siloté. Dans ce contexte, Il est permis à plusieurs services de partager les mêmes objets connectés.

La propriété de partage d'objets par les services est très intéressante car elle permet de réutiliser des objets qui sont présents dans un contexte d'exécution commun à plusieurs services. Cette possibilité de réutilisation d'objets permet de palier au problème de duplication d'objets dans le cas d'un monde siloté où chaque service opère sur des objets qui lui sont propres.

Cependant, lorsque plusieurs services partagent un même objet en modification, ils peuvent vouloir agir de manière différente sur cet objet partagé, c'est ce qui est désigné par l'expression conflit d'accès des services aux objets partagés. Dans un tel cas de figure, quelle décision doit être prise ? et qui est-ce qui est apte à prendre cette décision ?

En plus de concevoir des services qui sont indépendants des objets connectés, nous avons pour ambition de permettre à plusieurs services de partager un même objet connecté s'ils en ont le besoin. Ainsi, le service d'allumage et le service de sécurité décrits par la figure 3.1 pourront partager la même donnée de capture de présence qui provient d'un même capteur de présence comme le montre la figure 3.3.

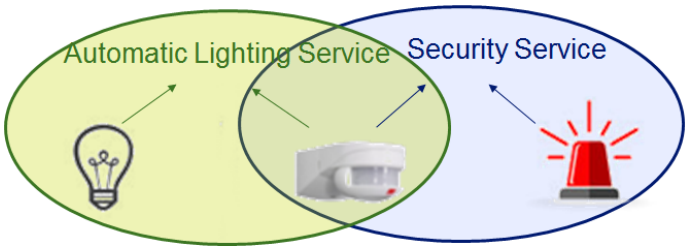


FIGURE 3.3: Partage d'un capteur de présence par deux services.

L'accès multiple aux objets permet d'éviter la redondance des objets connectés, où plusieurs services peuvent partager un même objet. Si l'objet partagé est un capteur qui mesure une variable de son environnement alors plusieurs services peuvent lire simultanément les valeurs de cette variable sans altérer le bon fonctionnement de l'objet. En revanche, si l'objet est un actionneur sur lequel les services peuvent agir alors il faut s'assurer que cet objet ne reçoive pas simultanément deux instructions contradictoires de la part de deux services différents. Une stratégie de contrôle d'accès aux actionneurs devra alors être mise en place.

3.2.2 Prise en compte de l'environnement et du contexte

Ce verrou concerne la prise en compte de l'environnement par les services. Les services que nous considérons sont des services qui sont destinés à interagir avec leur environnement, il est alors important que ces derniers tiennent compte et s'adaptent au contexte dans lequel ils s'exécutent.

3.3 Acceptabilité des services

Les services auxquels nous nous intéressons sont destinés à optimiser la vie quotidienne de leurs utilisateurs. Ces derniers peuvent être des utilisateurs experts ou des utilisateurs lambda. Dans un cadre où les services sont adressés à des êtres humains, le critère d'acceptabilité de ces services devient critique. L'acceptabilité des services est un domaine de recherche pluridisciplinaire, nous nous intéressons dans nos travaux à l'acceptabilité du point de vue informatique et nous traitons trois verrous qui nous semblent importants dans ce domaine : l'interaction des utilisateurs avec les services, l'adaptation des services aux besoins des utilisateurs et l'abstraction de la complexité technique des services.

3.3.1 Adaptation des services aux besoins des utilisateurs

Les utilisateurs auxquels les services sont destinés peuvent avoir des profils et des besoins très différents. Les services auxquels nous nous intéressons sont destinés à être déployés à une grande échelle. Il nous semble alors important que le comportement des services ne soit pas figé, mais qu'il puisse s'adapter aux besoins de chaque utilisateur.

3.3.2 Interaction des utilisateurs avec les services

L'interaction des utilisateurs avec les services est importante dans le cas où les services que nous étudions ne sont pas des services autonomes mais ils sont destinés à interagir avec les utilisateurs. Lors de la phase de conception de services, il est nécessaire de considérer les aspects d'interactions efficaces entre les utilisateurs et les services.

Pour que les services puissent adapter leur comportement aux besoins des utilisateurs, il est nécessaire qu'il y ait une interaction efficace entre ces services et leurs utilisateurs.

En somme, les problématiques que nous abordons dans la thèse sont classés en trois thématiques : le comportement cloisonné, la composition de services et l'acceptabilité de services par les utilisateurs. Le comportement cloisonné soulève les problématiques des services et objets connectés en silos et la dépendance technique des services aux objets. La composition de services aborde la problématique de conflit d'accès des services aux objets partagés ainsi que la prise en compte, par les services, de l'environnement et du contexte d'exécution. Enfin, l'acceptabilité des services par les utilisateurs aborde les aspects d'adaptation des services aux besoins des utilisateurs et les aspects des interactions des utilisateurs avec les services.

4 État de l'art

http://geek-and-poke.com, September, 2nd, 2015

SOMMAIRE

Nous présentons, dans ce chapitre, un état de l'art des technologies qui sont relatives aux problématiques que nous abordons dans la thèse. Nous commençons par présenter les techniques desquelles nous nous sommes inspirées et que nous avons utilisées dans la thèse. Ensuite, nous présentons les solutions existantes dans le domaine de l'IdO et qui concernent les problématiques que nous abordons dans la thèse.

4.1 Techniques utilisées

Cette section présente les techniques que nous avons utilisées ou dont nous nous sommes inspirés pour notre travail de thèse. Nous commeçons par décrire les cadres d'architectures puis nous présentons les approches déclaratives.

4.1.1 Cadres d'architecture

Nous présentons, dans cette section, deux approches de cadres d'architecture : l'approche MDA (Model Driven Architecture) et l'approche RM-ODP (Reference Model of Open Distributed Processing).

L'approche MDA

Les travaux qui ont été menés durant notre thèse ont pour but de produire des modèles d'architecture dans lesquels les aspects métier et les aspects technologiques sont indépendants les uns des autres. Pour définir notre cadre architectural, nous nous sommes alors inspirés de l'approche MDA (Model Driven Architecture) [21] [22] [23], qui constitue un standard qui a été proposé par l'OMG (Object Management Group) [24]. L'approche MDA, est centrée sur les modèles et permet de définir plusieurs niveaux de modèles ainsi que des transformations entre ces niveaux de modèles[25].

L'approche MDA fournit un niveau d'abstraction supérieur lors de la description d'un système. En effet, Le CIM (Computational Independent Model) regroupe des modèles de haut niveau qui sont indépendants des concepts informatiques. A partir des modèles CIM et à l'aide de transformations de modèles, il est possible de générer des modèles PIM (Platform Independent Model) qui sont indépendants des plateformes. Les modèles PSM (Platform Specific Model) sont générés à partir des modèles PIM pour une plateforme particulière à l'aide de transformations de modèles. Enfin, le code source de l'application est généré à partir des modèles PSM. La figure 4.1 représente les différents niveaux de modèles qui sont définis par le MDA.

FIGURE 4.1: Niveaux d'abstractions des architectures dirigées par les modèles.

Le lecteur qui souhaite approfondir sa connaissance des niveaux d'abstraction et des transformations de modèles de l'approche MDA est invité à consulter le standard de l'OMG [21].

RM-ODP

Le standard RM-ODP (Reference Model of Open Distributed Processing) [26] [27] [28] [29] est un modèle de référence résultant des travaux communs menés par deux organismes de standardisation : l'ISO (International Organization for Standardization) et l'ITU (International Telecommunication Union).

RM-ODP vise à fournir un cadre de coordination pour le traitement distribué ouvert en mettant en place une architecture qui supporte la distribution, l'interfonctionnement, l'interopérabilité et la portabilité [26]. Il définit cinq points de vue, représentés par la figure 4.2 : le point de vue Entreprise, Information, Traitement, Ingénierie et Technologie. Chaque point de vue représente une perspective du système via laquelle seuls les éléments qui sont relatifs au point de vue en question sont considérés.

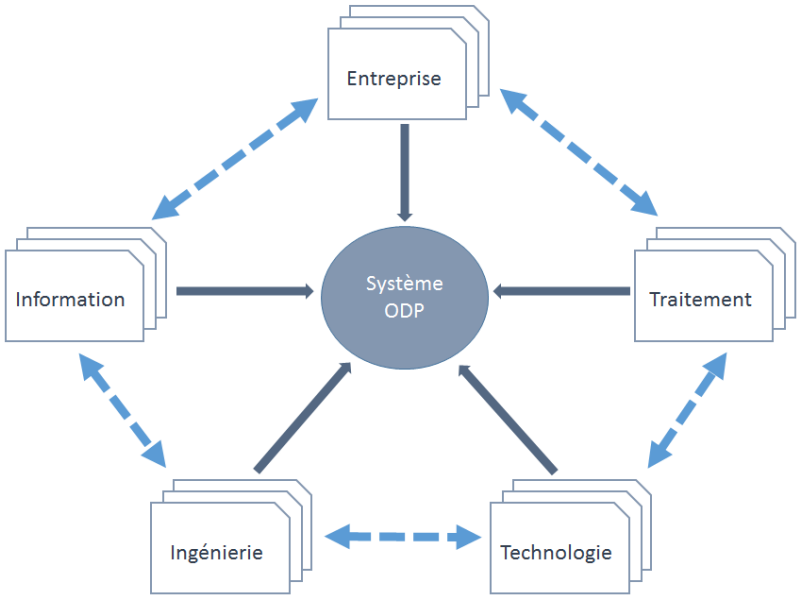


FIGURE 4.2: Les cinq points de vue du standard RM-ODP.

Le point de vue entreprise décrit les aspects économiques du système en définissant ses objectifs, sa portée et ses politiques. Ce point de vue répond aux questions : Dans quel but ? pourquoi ? qui ? et quand ?

Le point de vue information décrit les informations qui sont nécessaires au bon fonctionnement du système ODP. Il répond à la question : quoi ?

Le point de vue traitement décrit les aspects de conception de l'application et la décomposition fonctionnelle du système. Il répond à la question : Comment fonctionne chaque élément du système distribué ?

Le point de vue ingénierie décrit l'infrastructure requise pour déployer le système distribué. Il répond à la question : comment s'articulent les différents éléments du système ensembles ?

Le point de vue technologie décrit les choix technologiques pour implémenter le système distribué. Il répond à la question : avec quoi ?

Le lecteur qui souhaite approfondir ses connaissances est invité à se référer aux différents documents de l'ISO/ITU qui détaillent RM-ODP [26] [30].

4.1.2 Approches déclaratives

Les services sont des composants logiciels qui s'exécutent dans le but d'atteindre des objectifs des utilisateurs. Ils sont dirigés par des objectifs. De même, Les approches déclaratives consistent à spécifier ce qui doit être fait sans préciser comment ça doit l'être. Elles consistent à décrire des objectifs à atteindre plutôt que des procédures à suivre. Ces approches, comme les services, sont dirigées par des objectifs.

Les règles métier peuvent être utilisées dans des approches déclaratives. Elles permettent de décrire les objectifs en faisant abstraction des procédures. Ces règles métier sont des expressions qui permettent de guider et de contrôler le comportement d'un système. [31]

D'après le Business Rules Group [32], une règle métier est une déclaration qui définit ou contraint certains aspects liés aux métiers. Elle vise à guider, contrôler ou influencer le comportement d'un élément d'un Système d'Information. Une règle métier est une règle atomique qui ne peut pas être décomposée. Les règles métier représentent des éléments important de description des Systèmes d'Information. Plusieurs travaux de spécification, de classification, de formalisation ou d'implémentation ont été menés autour des règles métier [33] [34].

Notre travail de thèse est inspiré par les approches déclaratives qui se basent sur des règles métier pour décrire et contrôler le comportement de services.

4.2 Solutions existantes

Nous présentons dans cette section le contexte technologique dans lequel s'inscrit ce travail de thèse. L'état de l'art est divisé en trois axes : l'exécution de services, la composition de services pour l'IdO et l'acceptabilité de services.

4.2.1 Exécution de services

Les utilisateurs des technologies de l'IdO sont réfractaires à l'idée d'être contraint à s'équiper de la totalité des objets et des services de l'IdO chez un même fournisseur. Au contraire, lors de l'acquisition d'objets ou de services, les utilisateurs ont tendance à privilégier les solutions qui ne posent pas de contraintes fortes sur leurs futures acquisitions.

De ce fait, plusieurs efforts ont émergé dans le but de converger vers une interopérabilité entre les services et les objets connectés. Nous distinguons trois approches : les partenariats, les standardisations et les médiateurs. Les partenariats visent à mettre en place des contrats de collaborations entre quelques constructeurs d'objets connectés et quelques fournisseurs de services afin d'assurer la compatibilité entre les services et les objets. Les standardisations visent à fournir des standards et des plateformes qui permettent d'abstraire l'hétérogénéité des objets et des protocoles qu'ils utilisent afin de les mettre à disposition des services de manière homogène et assurer alors un certain degré d'interopérabilité. Les médiateurs fournissent un moyen pragmatique de décrire un comportement de services pour l'IdO.

Solutions propriétaires et partenariats industriels

Dans le but d'offrir à l'utilisateur un large panel d'objets et de services tout en conservant leur position de pionniers du marché de l'IdO, les gros industriels adoptent une approche de solutions propriétaires et de partenariats industriels. En effet, chaque « gros » fabricant d'objets connectés, établit des partenariats avec des fournisseurs de services et avec d'autres constructeurs d'objets afin d'assurer une interopérabilité avec les objets qu'il met sur le marché. Chacun d'entre eux tend à mettre en place un standard et des outils qui l'accompagnent.

Samsung SmartThings [35] sont les objets connectés qui sont fabriqués par le constructeur coréen. Sur son site, le constructeur affiche une liste des objets qui sont compatibles avec SmartThings [36]. Cette compatibilité est par ailleurs désignée par l'expression « Works with SmartThings » présente sur les objets qui le sont. De plus, Samsung encou-

rage les utilisateurs qui le souhaitent à développer leurs propres services en interopérabilité avec SmartThings en fournissant une plateforme de développement SmartThings Developers [37].

Apple se positionne sur le marché de l'IdO par Apple HomeKit [38] et Apple HealthKit [39]. HomeKit fournit des objets et des services pour la maison et HealthKit concerne des objets et des services autour de la santé et le maintien en bonne forme des utilisateurs. Le logo « Works with HomeKit », représenté par la figure 4.3 [40], est présent sur les objets compatibles avec la technologie IdO d'Apple. Apple Developer [41] fournit une plateforme de développement de services compatibles avec les produits Apple.

FIGURE 4.3: Logo : Works With Apple HomeKit.

Google Nest [42] et Google Home [43] [44] sont les solutions via lesquelles Google aborde le marché de l'IdO. Comme les autres constructeurs, Google passe des partenariats pour que le plus grand nombre d'objets sur le marché soit compatible avec ses solutions.

Solutions ouvertes et standards

Dans le but de favoriser l'interopérabilité et de casser les silos de l'IdO, une deuxième approche, portée par des consortiums, vise à définir des solutions et des standards ouverts. Nous citons trois approches pertinentes : OneM2M, AllJoyn et Iotivity.

Dans le but de standardiser les communications dans l'IdO, le consortium OneM2M a été formé [45]. C'est un consortium international important qui est constitué de plusieurs organismes de standardisation dont l'ETSI (European Telecommunications Standards Institute). OneM2M a proposé en 2015 une architecture Restful [46] basée sur des ressources afin de faciliter l'interopérabilité entre les objets de l'IdO [47]. Le projet OM2M [48], initié par LAAS-CNRS, est une implémentation Open Source du standard OneM2M. Il fournit une plateforme de services M2M horizontale pour le développement des services indépendant du réseau sous-jacent, dans le but de faciliter le déploiement des applications verticales et des dispositifs hétérogènes.

L'Alliance AllSeen qui regroupe plus de deux cents (200) compagnies [49] a mis en place le standard et la plateforme Open Source AllJoyn [50]. Cette plateforme permet de faciliter l'implémentation des communications via des paradigmes pair-à-pair entre les objets sur un réseau local, de bénéficier de services de haut niveau fournis par les AllJoyn Service Framework pour accélérer et faciliter le développement des applications. Elle offre aussi des API via les AllJoyn Core Libraries pour permettre aux développeurs d'implémenter leurs propres services tout en évitant le couplage fort entre les services et les objets.

Par ailleurs, l'Open Connectivity Foundation (OCF) [51] propose Iotivity [52], une spécification, une plateforme de communication et une implémentation open source. Cette plateforme permet d'abstraire l'hétérogénéité des objets et leur offre la possibilité de communiquer indépendamment des protocoles qu'ils utilisent.

Solutions pragmatiques et médiateurs

Une autre approche, plus récente que les deux précédentes, consiste à mettre en oeuvre des services médiateurs. Par exemple IFTTT (If This Then That) [53] permet à l'aide de règles conditionnelles de créer des connexions entre services et entre des services et des objets connectés. La figure 4.4 représente une recette IFTTT. Chaque recette est composée de deux parties, un déclencheur et une action. La partie « this » représente le déclencheur qui peut être un service comme un service de météo par exemple ou un objet connecté comme une porte connectée par exemple. De même, la partie « that » représente l'action à effectuer et concerne un service ou un objet connecté.

FIGURE 4.4: IFTTT.

4.2.2 Composition de services pour l'IdO

Nous nous intéressons dans nos travaux à la composition de services d'un point de vue de gestion de conflits d'accès de services aux objets connectés.

Dans le domaine de la résolution de conflits pour Internet of Things, les auteurs de [54] ont proposé Diasuite, un outil pour développer des applications SCC (Sense Compute

Control). Le paradigme SCC permet de collecter des données à partir de capteurs, de les calculer et de commander l'actionneur pour agir sur l'environnement.

Pour gérer les conflits, d'autres travaux proposent dans [55] une approche basée sur une extension DSL (Domain Specific Language). Cette approche génère au moment de l'exécution, un code pour gérer les conflits. La principale limitation de cette approche est que la gestion des conflits n'est pas entièrement automatique et les développeurs doivent agir sur le système pour les résolutions de conflits.

Dans [56], les auteurs décrivent une approche qui vise à gérer les conflits de partage d'objets. Ils proposent d'ajouter une couche d'accès autonome dans une plate-forme ubiquitaire appelée iCASA [57]. Cette plate-forme fournit un modèle orienté services pour développer des applications ubiquitaires. Ces efforts semblent être intéressants mais sont encore dans une phase préliminaire. Il n'existe pas encore de modèle formel qui est proposé pour répondre à la problématique de gestion de conflits.

4.2.3 Acceptabilité de services

Dans le but d'améliorer l'acceptabilité des services par les utilisateurs, plusieurs travaux mettent -plus ou moins- l'utilisateur au centre de leurs approches [58].

Les travaux parus dans [59] mettent les utilisateurs au centre des développements IdO, en les mettant dans un environnement d'objets connectés. Ces travaux mettent à disposition des utilisateurs une plateforme matérielle extensible et reconfigurable afin de leur permettre d'imaginer de nouvelles formes d'interactions et de personnaliser les fonctionnalités.

Dans le même registre, Freedomotic, offre une plateforme de développement IdO, Open Source et flexible [60]. Il fournit un environnement de simulation d'objets connectés et de contexte d'exécution. Il permet aux utilisateurs de définir les règles de comportement du système dans un langage proche du langage naturel.

Les auteurs de [61] s'inspirent des théories de la psychologie cognitive pour concevoir des systèmes IdO adaptables qui améliorent l'expérience des utilisateurs.

4.2.4 Modélisation sémantique des concepts de l'IdO

De nombreux travaux s'intéressent à la manière de représenter sémantiquement les différents concepts de l'IdO. L'ontologie SSN (Semantic Sensor Network) [62] [63] [64] proposée par le W3C se concentre sur la modélisation de capteurs de l'IdO. SSN propose un modèle qui décrit les capteurs et les propriétés physiques que ces derniers permettent de mesurer.

Les auteurs de [65] [66] se basent sur l'ontologie SSN pour proposer en suivant la même approche, l'ontologie SAN (Semantic Actuator Network) qui permet de modéliser les actionneurs de l'IdO en mettant l'accent sur les propriétés physiques sur lequels peuvent agir ces actionneurs.

Dans [67], les auteurs proposent une ontologie, nommée DogOnt, qui permet de modéliser les objets de l'IdO dans le contexte d'une maison connectée. Cette ontologie permet de décrire les objets connectés impliqués dans des cas d'usages de la domotique. Elle décrit aussi l'environnement d'exécution de ces objets en offrant la possibilité de définir différentes topologies.

Nous avons présenté, dans ce chapitre, un état de l'art qui concerne les problématiques que nous abordons dans cette thèse. Nous avons commencé par présenter deux cadres d'architecture desquels notre travail s'est inspiré. Ensuite, nous avons présenté les différentes solutions qui sont décrites dans la littératures ou qui sont mises à disposition sur le marché de l'IdO. Nous avons classé ces solutions selon quatre axes : L'exécution de services, la composition de services, l'acceptabilité des services par les utilisateurs et la modélisation sémantique des concepts de l'IdO.

5 Conclusion

La première partie de ce manuscrit de thèse décrit le contexte de recherche dans lequel s'inscrivent nos travaux.

Nous décrivons en premier lieu le domaine de l'Internet des Objets en précisant les notions d'objets connectés et de services tels qu'ils sont décrits dans la littérature et tels qu'ils sont abordés dans cette thèse. Les particularités des services connectés par rapport aux autres services informatiques sont mises en évidence.

Nous abordons ensuite les problématiques qui sont considérées dans cette thèse. Elles se déclinent en trois axes : le comportement cloisonné des services et objets connectés, la composition de services et l'acceptabilité de services par les utilisateurs.

Enfin, nous présentons un état de l'art qui décrit les différentes solutions présentes dans la littérature et sur le marché de l'IdO et qui sont relatives aux problématiques que nous abordons. Nous présentons également des approches et des techniques qui n'ont pas de relation directe avec le domaine de l'IdO mais qui ont influencé la manière dont nous avons abordé les problématiques posées.

Maintenant que le domaine de recherche, les problématiques et l'état de l'art sont posés, nous présentons, dans la partie II, les contributions apportées par cette thèse.

Deuxième partie

Contribution

6 Introduction

Nous présentons, dans la partie II, les différentes contributions qui ont été faites dans cette thèse.

Nous commençons par présenter une analyse informelle des problématiques qui ont été posées dans le chapitre 3. D'abord, les problématiques de définition, de conception et de composition de services dans le but de mettre en place un écosystème ouvert et composable de services et d'objets sont analysées et étudiées. Cette analyse prend forme à travers différents scénarios de mise en place, d'exécution et d'orchestration de services et d'objets connectés.

Ensuite, nous présentons le cadre d'architecture plus large dans lequel s'inscrit l'approche sémantique proposée par cette thèse. Cette approche sémantique offre un niveau d'abstraction qui permet aux concepteurs de services de se concentrer sur les aspects fonctionnels des services et des objets. Le cadre d'architecture aborde, en plus de l'approche sémantique, des aspects plus opérationnels de mise en oeuvre de ces services (niveau Artefacts) dans des environnements techniques éventuellement hétérogènes (niveau Ressources).

Puis, nous décrivons de manière formelle l'approche sémantique de conception et d'orchestration de services proposée par cette thèse. Dans cette approche, les services sont orientés par des besoins qui sont exprimés sous forme d'objectifs. Nous adoptons une approche déclarative, qui permet de spécifier des objectifs à atteindre plutôt que des procédures à suivre, pour décrire les comportements de services. Un méta-modèle est proposé pour concevoir la structure des objets et des services ainsi que leurs comportements. En plus de ce méta-modèle, nous proposons des règles de validation qui permettent de vérifier la conformité de modèles de services par rapport au méta-modèle sémantique que nous proposons. Enfin, nous décrivons des stratégies flexibles d'orchestration de services et de résolution de conflits en s'appuyant sur l'approche sémantique.

7 Analyse des services de l'IdO et de leur problématique

"The joy of success lies in the process or journey – not so much on the destination, because one destination opens the door for another. It's like the beauty of watching a blossoming flower ... a bird taking flight for the first time or a child taking the first step ... Enjoy your process today as I am."

Val Uchendu

SOMMAIRE

L'objectif de ce chapitre est d'analyser les problématiques de définition et de composition de services dans le but de mettre en place un écosystème de services ouverts et composables.

7.1 Scénario de mise en place des services et des objets connectés

Nous proposons dans cette section un scénario de première mise en place de services et d'objets connectés dans une maison. Ce scénario est un exemple qui tend à concrétiser un projet de l'utilisateur pour équiper sa maison de services et d'objets connectés.

Nous considérons, dans un premier temps, les objets connectés et les services tels qu'ils sont décrits dans la littérature (cf. chapitre 2). Puis, au fur et à mesure de notre analyse, nous préciserons les définitions qui doivent l'être.

Nous décrivons le scénario en plusieurs étapes. Nous considérons que l'état initial de la maison correspond à son état sans aucun objet connecté ni aucun service. En partant de cet état initial, chaque étape va apporter des enrichissements jusqu'à aboutir à une maison connectée et fonctionnelle qui contient des objets connectés et des services qui gèrent ces objets.

Plusieurs des étapes que nous décrivons peuvent avoir lieu simultanément ou se précéder les unes les autres. Les particularités liées à chaque étape sont décrites dans une sous section. Les règles de précédence sont également représentées par la figure 7.1.

Le scénario de première mise en place de services et d'objets connectés décrit un état initial « Maison nue » et un état final « Maison connectée fonctionnelle ». Il décrit six étapes la « Sélection des services », la « Sélection des objets connectés », la « Description de la topologie de la maison », la « Mise en place des objets connectés », le « Déploiement des services » et enfin la « Mise en correspondance entre les services et les objets ». La « Sélection des services », la « Description de la topologie de la maison » et la « Sélection des objets connectés » peuvent avoir lieu directement après l'état initial « Maison nue ». Ce sont des étapes indépendantes les unes des autres et peuvent avoir lieu simultanément ou se précéder les unes les autres. Le « Déploiement des services » dépend de la « Sélection des services » et de la « Description de la topologie de la maison », il doit avoir lieu après ces deux étapes. De même, la « Mise en place des objets connectés » dépend de la « Sélection des objets connectés » et de la « Description de la topologie de la maison », elle doit avoir lieu après ces deux étapes. La « Mise en correspondance entre les services et les objets » est la dernière étape qui précède l'état final « Maison connectée fonctionnelle » et qui dépend

de la « Mise en place des objets connectés » et du « Déploiement des services ». A la fin de cette étape, l'état final de la maison est une « Maison connectée fonctionnelle ».

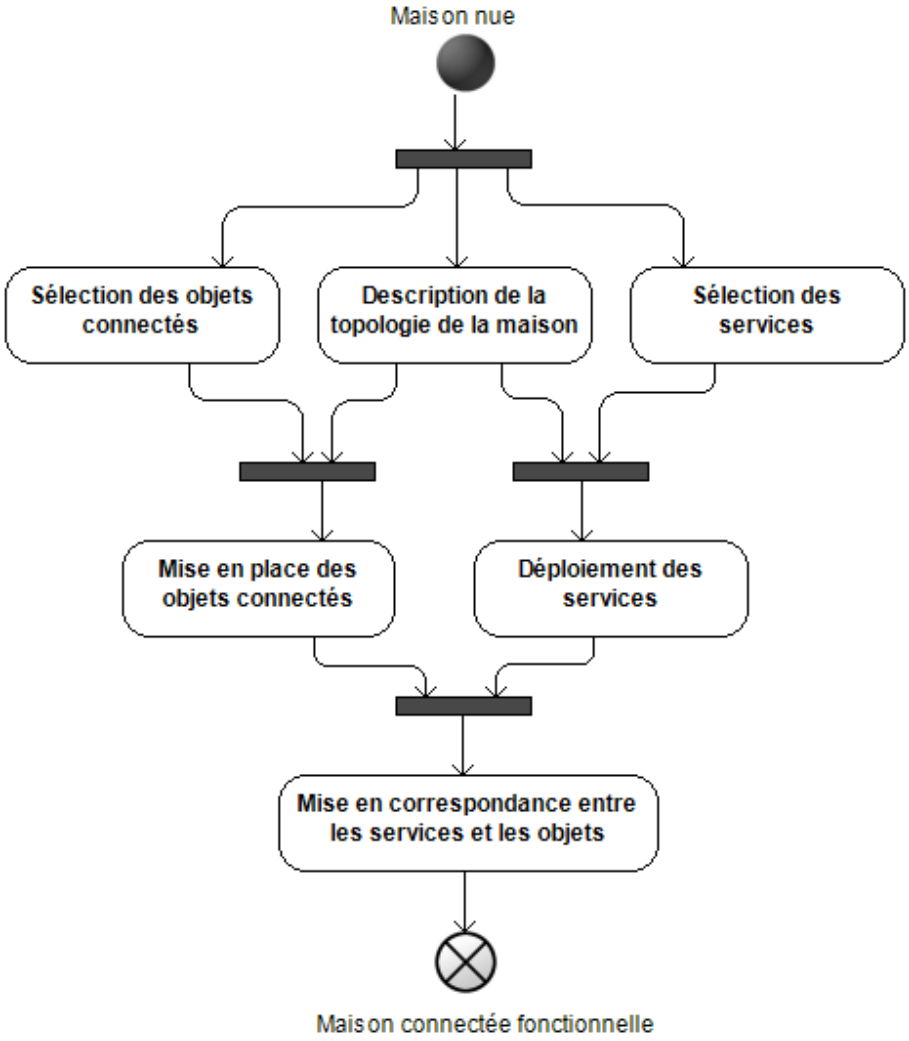


FIGURE 7.1: Scénario de première mise en place de services et d'objets connectés.

Nous avons décrit les règles de précédence entre les étapes qui composent le scénario de première mise en place de services et d'objets connectés. Nous allons à présent décrire chaque étape en précisant ses besoins et les résultats obtenus après qu'elle ait eu lieu.

7.1.1 Etat initial : Maison nue

Nous considérons que l'état initial de la maison est une maison nue. Une maison nue ne contient aucun objet connecté ni aucun service. Cette maison sera enrichie au fur et à mesure du déroulement des étapes du scénario jusqu'à arriver à l'état final qui représente une maison connectée fonctionnelle.

7.1.2 Sélection des services

Cette étape consiste à sélectionner un bouquet de services par l'utilisateur en fonction de son projet initial pour équiper sa maison. Les services sont présentés à l'utilisateur par le fournisseur de services sous forme d'un catalogue de services génériques. Les services sont dits « génériques » parce qu'ils ne sont encore liés à aucun objet connecté ni à aucune topologie de maison. Il s'agit alors d'une version générique des services qui sont destinés à être déployés dans l'habitat des utilisateurs finaux.

Si l'utilisateur souhaite équiper sa maison d'un service de régulation de température qui s'exécute à la granularité d'une pièce, lors de cette étape une version générique du service de régulation de température est sélectionnée par l'utilisateur dans le catalogue de services.

Cette étape peut avoir lieu avant, après ou en même temps que les étapes de sélection des objets connectés et de description de la topologie de la maison.

7.1.3 Sélection des objets connectés

L'utilisateur sélectionne les objets connectés, en adéquation avec son projet d'équipement initial. La sélection se fait sur un catalogue qui est mis à disposition à l'utilisateur par le fournisseur d'objets connectés. Les objets seront installés lors d'une prochaine étape du scénario. La sélection des objets connectés est indépendante des étapes de sélection des services et de description de la topologie de la maison. Si le projet de l'utilisateur est d'équiper sa maison d'objets pour la régulation de température, il sélectionnera des objets en lien avec son projet, tels qu'un thermomètre connecté, un appareil de chauffage connecté ou une chaudière connectée.

7.1.4 Description de la topologie de la maison

Cette étape consiste à représenter la topologie de la maison de façon organisée. Il s'agit de décrire les informations relatives à la topologie de la maison telles que les pièces et les cloisons (murs, plafonds, planchers, ... etc.). La description des cloisons permet de spécifier la mitoyenneté des pièces. Des services du type du service de régulation de température peuvent considérer la mitoyenneté des pièces d'une maison afin de prendre en considérations les transferts thermiques possibles entre les différentes pièces [68] [69]. La topologie de la maison décrit aussi toutes les ouvertures (fenêtres ou portes) en spécifiant les cloisons sur lesquelles elles se trouvent.

7.1.5 Déploiement des services

Cette étape consiste à déployer les services génériques qui ont été sélectionnés par l'utilisateur. Ce déploiement tient compte de la topologie de la maison, il dépend alors de cette étape et doit s'exécuter après elle. Si nous reprenons le service générique de régulation de température qui s'exécute par pièce ; lors de cette étape, il sera déployé dans toutes les pièces à réguler. Dans le cas où le service générique décrit un orchestrateur qui a pour but de coordonner le comportement des services qui s'exécutent par pièce, cet orchestrateur sera également déployé lors de cette étape, cela permettrait d'avoir une régulation de température globale au niveau de la maison. Les services qui s'exécutent par pièces peuvent aussi être autonomes et ne pas nécessiter la présence d'un orchestrateur. Ces caractéristiques dépendront du service générique qui a été sélectionné par l'utilisateur.

Lors du déploiement des services génériques, le contexte d'exécution est défini pour chaque service. Le contexte d'exécution d'un service autonome qui s'exécute dans une pièce est cette même pièce. Cette définition de contexte permet de restreindre le champ d'action et au champ de «vision» du service.

7.1.6 Mise en place des objets connectés

Cette étape consiste à introduire les objets connectés dans la maison. L'introduction des objets connectés est indépendante de la mise en place des services, mais dépend de la description de la topologie de la maison. Lors de l'introduction des objets, un lien est créé entre l'objet et sa localisation physique. Grâce à la localisation de chaque objet, les services pourront dans un second temps être mis en correspondance avec les seuls objets qui se trouvent dans leur contexte d'exécution. Reprenons l'exemple de l'utilisateur qui souhaite équiper sa maison de technologies de l'IdO afin de réguler la température. Il aura déjà sélectionné les objets connectés et lors de cette étape, tous les objets seront installés dans la maison en fonction du projet de l'utilisateur.

7.1.7 Mise en correspondance entre les services et les objets

Une fois que les services ont été déployés et que les objets ont été installés, cette étape consiste à faire la correspondance entre les objets et les services qui les gèrent.

Le processus de mise en correspondance entre les services et les objets connectés dépend de plusieurs critères, notamment :

— Le contexte d'exécution de chaque service détermine l'emplacement des objets qui intéressent ce service. Un service de régulation de température qui s'exécute dans

le contexte d'une seule pièce en faisant abstraction des données provenant de l'extérieur de cette pièce, sera intéressé uniquement par les objets qui se trouvent physiquement dans cette pièce ou qui mesurent ou agissent sur des propriétés qui appartiennent à cette même pièce. Le contexte d'exécution des services permet d'effectuer une première sélection des objets connectés qui peuvent intéresser le service.

— Chaque service, en fonction de l'objectif pour lequel il a été conçu, s'intéresse à certains types d'objets. Un service de régulation de température s'intéressera par exemple à des capteurs de type « thermomètre ». Une deuxième sélection d'objets connectés est effectuée en fonction des types de chaque objet connecté qui se trouve dans le contexte d'exécution du service.

Lorsque les étapes de sélection sont effectuées, chaque service est mis en correspondance avec les objets connectés qui l'intéressent en fonction de leurs localisations et de leurs types. Un lien de correspondance est créé entre chaque service et les objets qu'il peut gérer.

7.1.8 Etat final : Maison connectée fonctionnelle

L'état final correspond à une maison où la topologie de la maison a été décrite ; les services génériques ont été sélectionnés et déployés ; les objets connectés ont été installés ; et les liens de correspondance entre les services et les objets ont été créés. L'état final correspond à une maison connectée fonctionnelle où un certain nombre de services gèrent automatiquement les objets qui sont présents dans cette maison.

Cet état final représente le premier état fonctionnel de la maison où les services gèrent de manière automatique les objets connectés. Il s'agit d'un état où il peut encore y avoir des modifications de configuration. Des objets peuvent être retirés ou de nouveaux objets peuvent aussi être installés. Des services peuvent être retirés ou de nouveaux services peuvent être mis en place puis mis en correspondance avec des objets connectés de la maison. Aussi, il est moins fréquent mais il peut arriver que la topologie de la maison soit modifiée suite à des travaux qui modifient la topologie de l'habitat.

En somme, l'état final que nous décrivons ici représente le premier état de la maison connectée et fonctionnelle mais dont la composition en termes de services et d'objets connectés peut encore évoluer.

Le scénario de première mise en place de services et d'objets connectés décrit un ensemble d'étapes nécessaires pour une première installation de services et d'objets connectés en partant d'une maison nue sans aucun service ni objet connecté. D'autres scénarios

peuvent également être décrits, tels qu'un scénario de maintenance qui décrirait un processus permettant d'ajouter ou de retirer des objets ou des services. Il décrirait aussi les étapes qui permettent de mettre à jour la topologie d'une maison. A titre d'exemple illustratif, nous considérons en détail toutes les étapes qui constituent le scénario de mise en place de services et d'objets connectés.

7.2 Gestion de conflits d'accès des services aux objets

Dans cette section, nous considérons comme actée l'existence d'un écosystème peuplé d'objets connectés aux interfaces ouvertes et pouvant être liés à des services qui les requièrent dès lors que les interfaces les mettant en correspondance en relation sont compatibles.

Nous explorons, de manière progressive, les différentes formes de composition d'objets et services pour en dégager les problématiques et pour définir les modes de coopérations et de résolution de conflit.

7.2.1 Définition des services

En première approximation, dans un contexte IdO, un service est une entité qui agit sur un actionneur en fonction des données captées par un capteur. Ce mode d'action peut être représenté par un échange d'évènements : le service reçoit des données du capteur et réagit par un envoi de commandes vers l'actionneur. La figure 7.2 illustre deux exemples de services.Le premier utilise un détecteur de présence et agit sur une lampe connectée, le second utilise un (autre) détecteur de présence et agit sur une alarme.

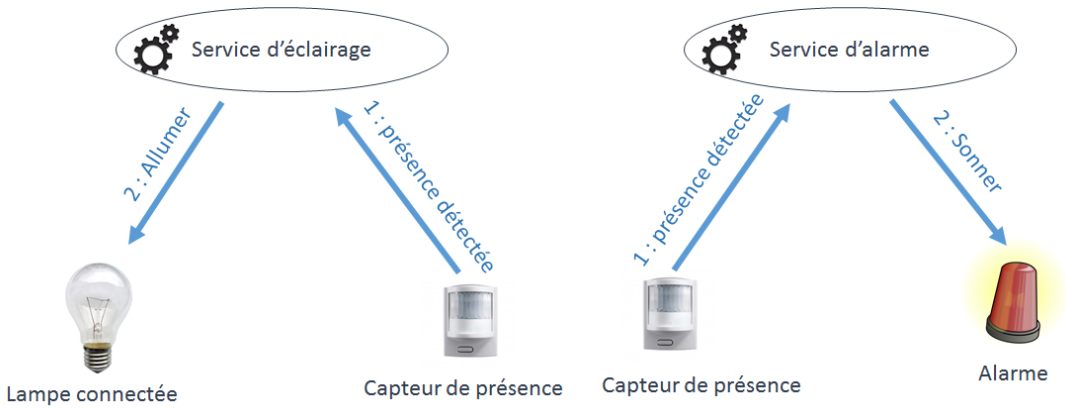


FIGURE 7.2: Deux services utilisant deux détecteurs de présence.

Une première forme de composition souhaitable est le partage de capteurs entre plusieurs services. En effet, les deux services présentés par la figure 7.3 peuvent être déployés avec un seul détecteur de présence.

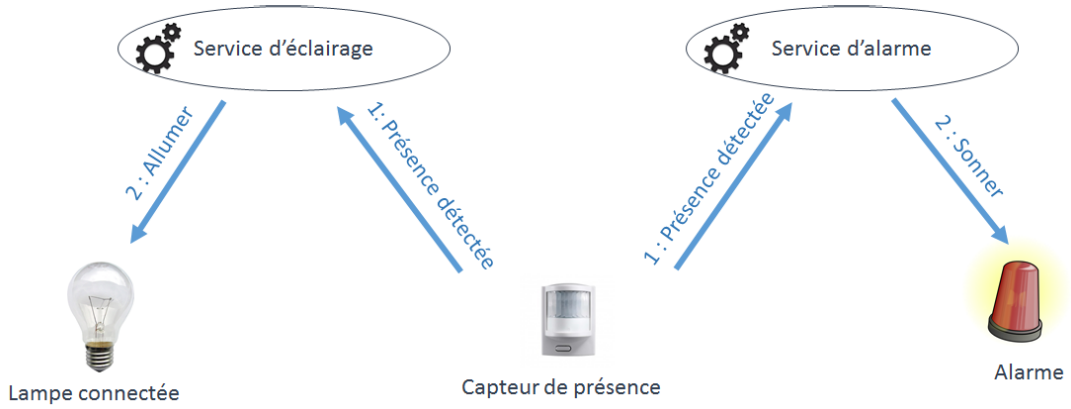


FIGURE 7.3: Deux services partageant un détecteur de présence.

Cette composition, n'implique pas une coopération particulière entre ces deux services. Ils reçoivent les données du capteur, mais chacun est ensuite libre d'agir sur son actionneur.

Considérons une deuxième forme de partage d'objets connectés entre services. La figure 7.4 représente deux services ayant chacun son propre capteur mais agissant sur le même type d'actionneur. Le premier service réagit à une mauvaise qualité de l'air dans la pièce en commandant l'ouverture de la fenêtre. Le second réagit à une baisse de la température dans la pièce en commandant la fermeture de la fenêtre. Si ces deux services sont déployés dans la même pièce, leur action sur la fenêtre connectée doit être coordonnée.

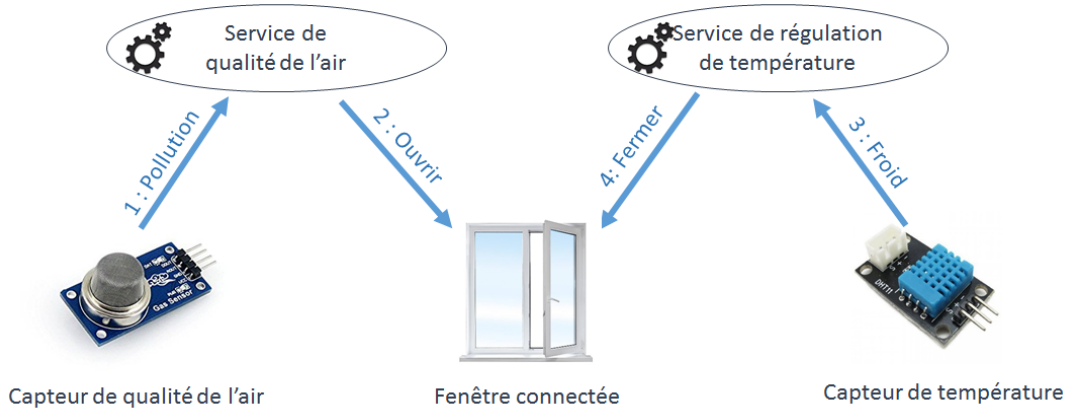


FIGURE 7.4: Oscillation entre service qualité de l'air et service réglage de température.

Reprenons la représentation des services par échange d'évènements. Comme chaque service réagit d'une manière séparée aux évènements et au rythme de son propre capteur, les réactions sont susceptibles d'être alternées provoquant un phénomène indésirable d'oscillation entre services. Plaçons nous dans le contexte d'une pièce froide et ayant son air vicié. Les évènements annonçant la froidure feront réagir le premier service commandant la fermeture de la fenêtre, et ce, en alternance, avec les évènements déclarant la mauvaise qualité de l'air qui provoquerons l'ouverture de la fenêtre.

Par ailleurs, si l'on couple ces deux services en les forçant à coopérer, il en résultera un conflit qui doit être tranché avant de décider de l'action à appliquer sur l'objet connecté fenêtre. De l'examen de l'exemple illustré par la figure 7.4 nous tirons deux conséquences. La première est un raffinement de la définition du concept de services dans le contexte IdO. La deuxième a trait à la manière de définir et de composer des services.

Définition du concept de service A la lumière de l'exemple illustré par la figure 7.4, le concept de service élémentaire doit être revisité et précisé. Reprenons l'exemple du service de qualité de l'air et considérons la possibilité que cette qualité peut être obtenue soit en ouvrant la fenêtre soit, alternativement, en démarrant une ventilation. Ce service est illustré par la figure 7.5.

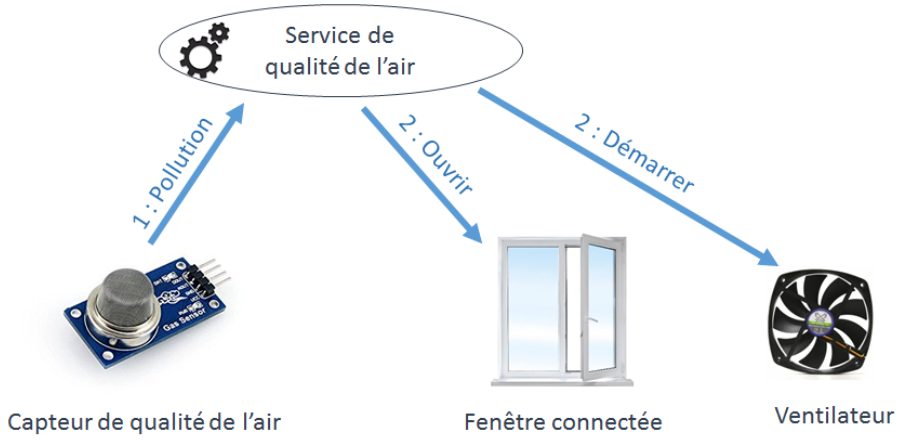


FIGURE 7.5: Service de qualité de l'air agisssant sur une Fenêtre et un ventilateur.

On voit tout l'intérêt de réunir ces deux modes d'action en un seul service plutôt que de les considérer comme deux services élémentaires. En effet, l'important pour un service est d'atteindre l'objectif qui lui est consigné (par l'utilisateur) et l'utilisation de différentes modalités permettra, à d'autres services avec lesquels il est amené à être composé, d'atteindre aussi le leur. Ceci amène à poser la définition suivante pour le service dans un contexte IdO :

> Un service est une entité qui peut être déployée dans un environnement physique dont elle peut acquérir l'état grâce à un ensemble de capteurs et sur lequel elle peut agir selon différentes modalités grâce à un ensemble d'actionneurs. Cette entité s'exécute de manière autonome avec pour mission l'atteinte d'un objectif consigné par l'utilisateur.

7.2.2 Définition des compositions de services

L'oscillation constatée dans l'exemple de la figure 7.4 incite à adopter une approche coopérative entre services partageant des actionneurs. De ce fait, ces services doivent être tous consultés lorsque survient un évènement d'un quelconque de leurs capteurs. Si nous reprenons l'exemple de la figure 7.4, l'occurrence de l'évènement « il fait froid » sera suivi par la séquence suivante : le service régulation de température informe le service de qualité de l'air « je propose de fermer la fenêtre », le service qualité de l'air se doit de vérifier avec son capteur si la qualité de l'air est suffisante pour fermer la fenêtre. Si oui, il donne son accord. Si non, il y a un conflit qui doit être tranché. Imaginons, par exemple, que le service de qualité de l'air est prioritaire, alors la commande résultante qui sera envoyée sera d'ouvrir la fenêtre.

Cet exemple, qui n'implique que deux services, montre immédiatement la complexité potentielle qui peut résulter lorsqu'on assemble des services partageant des actionneurs. Cette complexité est principalement due au fait que les services sont décrits de manière proche d'un modèle d'exécution, et en se basant sur les évènements échangés. Cette complexité rend les services et leur composition peu lisibles par les concepteurs et les utilisateurs finaux. En outre, elle rend difficile l'analyse des propriétés attendues des services et de leur composition. D'où l'idée de définir un niveau d'abstraction dans lequel les services et leur composition peuvent être définis, et d'articuler ce niveau avec d'autres niveaux, plus opérationnels, faisant émerger les concepts d'évènements et de middleware ou de ressources (l'architecture multi-niveaux proposée fera l'objet d'une autre section de la thèse).

Dans ce niveau d'abstraction, que nous appellerons niveau sémantique, et qui sera formellement défini dans les chapitres suivants, un service est une entité qui est associée à un ensemble de variables qui représentent l'ensemble des objets connectés avec lesquels le service est mis en relation. Les variables sont partitionnés en 3 catégories, les variables en lecture, et qui représentent les capteurs, les variables en écriture, et qui représentent les actionneurs, et les variables en lecture/écriture et qui représentent les captionneurs, c'est-à-dire, des actionneurs ayant la capacité de révéler leurs états.

Au niveau d'abstraction sémantique, un service est une entité qui est liée à un ensemble de variables en lecture (correspondant à ses capteurs et captionneurs) et un ensemble de variables en écriture (correspondant à ses actionneurs et captionneurs). Cette entité est réactive : elle réagit en lisant l'état des variables en lecture et propose un nouvel état aux variables en écriture.

Le comportement d'un service est donné par un ensemble de règles logiques : $R_1, ..., R_n$ où, chaque R_i est de la forme Pre $\Longrightarrow$ EtatSuivant.

Où *Pre* une condition sur les variables en lecture et EtatSuivant une(ou plusieurs – voir par la suite) proposition d'affectation de valeurs aux variables en écriture.

Les Prémisses établissent le contexte sur l'état des capteurs (et captionneurs) où une réaction du service est requise, et EtatSuivant est une proposition pour un état des actionneurs souhaité par le service.

Au niveau d'abstraction sémantique, le comportement d'un service S est donné par un ensemble de règles
$\{R_1, ..., R_k\}$ avec
$R_i = \text{Pre}_i \Rightarrow \text{EtatSuivant}_i$ où
Pre_i est une condition sur les variables en lecture de S et EtatSuivant_i est une proposition d'affectation de valeurs aux variables en écriture de S.

Remarque : la représentation précédente est une simplification. Elle correspond au concept de service déployé. Un service peut être décrit aussi de manière générique. Dans ce cas, il est donné comme agissant sur des rôles typés (au lieu de variables). L'opération de déploiement consistant à remplacer les rôles par des variables de même type.

Bouquets de services Nous avons vu que deux services partageant un actionneur doivent être associés afin de coordonner leurs réactions sur cet actionneur. D'une manière générale, l'on peut définir le concept de bouquet de services comme étant un ensemble de services dont les actions doivent être coordonnées. Si on définit la relation $\underline{part}$: $S \ \underline{part} \ S'$ ssi S et S' partagent un actionneur, notons $\underline{part}^*$ la fermeture transitive de la relation $\underline{part}$, alors on définit un bouquet comme étant tout ensemble, $\{S_1, ..., S_n\}$, de services vérifiant la propriété : $\forall \ i, j, S_i \ \underline{part}^* S_j$

La fermeture transitive se définit ainsi : $S \ \underline{part}^* \ S'$ ssi $\exists \ i_0, ..., i_k$, satisfaisant $S = S_{i_0}, S' = S_{i_k}$, *et* $\forall \ p, S_{i_p} \ \underline{part} \ S_{i_{p+1}}$.

> Un bouquet B est un ensemble $\{S_1,...,S_n\}$ de services partageant deux à deux - directement ou transitivement - un actionneur <nl> $B = \{S_1,...,S_n\}$ avec $\forall\ i,\ j:\ i \neq j \Rightarrow\ S_i\ \underline{part}^*\ S_j$

Orchestration des bouquets de services Les services d'un même bouquet peuvent être en conflit concernant l'état de certains de leurs actionneurs partagés. La résolution des conflits consiste à les ordonner selon la priorité souhaitée par l'utilisateur. D'où le mécanisme suivant qui définit la réaction d'un bouquet de services (qui sera formalisé dans les chapitres suivants).

> Un bouquet orchestré de services, B, est un bouquet de service totalement ordonné par la relation de priorité. $B = \{S_1,...,S_n\}$ avec $\forall\ i,\ j:\ i < j\ \Rightarrow S_j$ plus prioritaire S_i

Lorsque l'état des capteurs liés au bouquet change, chaque service évalue l'ensemble de ses prémisses (un service est bien formé ssi une seule prémisse peut être vraie à la fois) sur l'état des capteurs. Cette évaluation permet de dégager la sous-liste des services ayant une prémisse vraie. A cette liste correspond une liste de propositions d'états à appliquer sur les actionneurs. Cette liste est ordonnée selon la même relation de priorité qui ordonne les services. Un algorithme de type **sat**(satisfaisabilité booléenne) cherchera alors l'assignation de valeurs aux variables d'actionneurs qui satisfait le maximum de services prioritaires.

A la vue logique de ce mécanisme qui vient d'être décrit au niveau d'abstraction sémantique peut correspondre une vue opérationnelle où on verra apparaitre une nouvelle entité qui est l'orchestrateur. De manière informelle, les rôles sont ainsi répartis entre les services et leur orchestrateur : chaque service évalue ses prémisses et envoie ses propositions d'état suivant à l'orchestrateur lequel applique l'algorithme de **sat** et notifie les services de l'état suivant choisi.

Exemple de bouquets de services La figure 7.6 représente un exemple de configuration où cinq services partagent des objets connectés. Le service S1 est un exemple de service d'allumage automatique de lumière de la cuisine. Il est en relation avec un capteur de présence et une lampe connectée. Le service S2 est un service de contrôle de consommation électrique de la cuisine. Il est en relation avec les objets électriques se trouvant dans son contexte d'exécution, dans cet exemple, une lampe et une machine à café. De plus, il possède une consigne de consommation qui concerne la cuisine et qui provient du compteur

électrique. Le service S3 est un service de réveil qui met en marche la machine à café à l'heure du réveil des utilisateurs. Il s'appuie sur une machine à café et un capteur d'horloge. Le service S4 est un service de régulation de température de la pièce qui, en fonction de l'heure de réveil des utilisateurs et d'un capteur de température, régule la température. Il s'appuie sur un capteur de température, un capteur d'horloge, un radiateur et une fenêtre. Le service S5 est un service de contrôle de la qualité de l'air de la pièce qui s'appuie sur un capteur de qualité de l'air et une fenêtre.

Les services qui partagent des actionneurs sont groupés en bouquets. Deux bouquets de services sont formés à partir de ces services. Un premier bouquet regroupe les services S1, S2 et S3 et un deuxième bouquet regroupe les services S4 et S5. Chaque bouquet de services est géré par un orchestrateur.

7.2.3 Résolution de conflits

L'algorithme de résolution de conflit présenté ci-dessus possède trois variantes qui permettent d'augmenter la satisfaction des services impliqués. Nous présentons ces variantes en mettant en évidence le rôle joué par l'entité « orchestrateur ».

Proposition 1 : résolution de conflits par priorités

La première proposition d'algorithme de résolution de conflits est de le résoudre en s'appuyant que des priorités de service. Une priorité est associée à chaque service et lorsqu'il y a un conflit entre deux services, l'orchestrateur choisit de satisfaire le service le plus prioritaire des deux. Le deuxième service se voit refuser sa demande de changement d'état de l'objet connecté.

Les services qui font l'objet de cette thèse sont destinés à être utilisés par des humains. Dans le but de favoriser l'acceptabilité des services par les utilisateurs finaux, il est important que les services tiennent compte des souhaits des utilisateurs. Tous les utilisateurs de service n'ont pas les mêmes besoins. Par exemple, il y en a qui préfèreraient favoriser un service de confort par rapport à un service d'économie d'énergie et il y en a qui préfèreraient l'inverse.

Nous proposons alors que la gestion de conflits ait lieu en fonction de priorités qui ont été définies par les utilisateurs.

La figure 7.7 reprend l'exemple de conflit entre un service de régulation de température et un service de qualité de l'air. Ce conflit concerne l'objet fenêtre. Ici, lorsqu'un service souhaite modifier l'état d'un objet, il envoie d'abord sa requête à l'orchestrateur. Si un conflit est détecté entre deux services, alors l'orchestrateur, en suivant un algorithme de

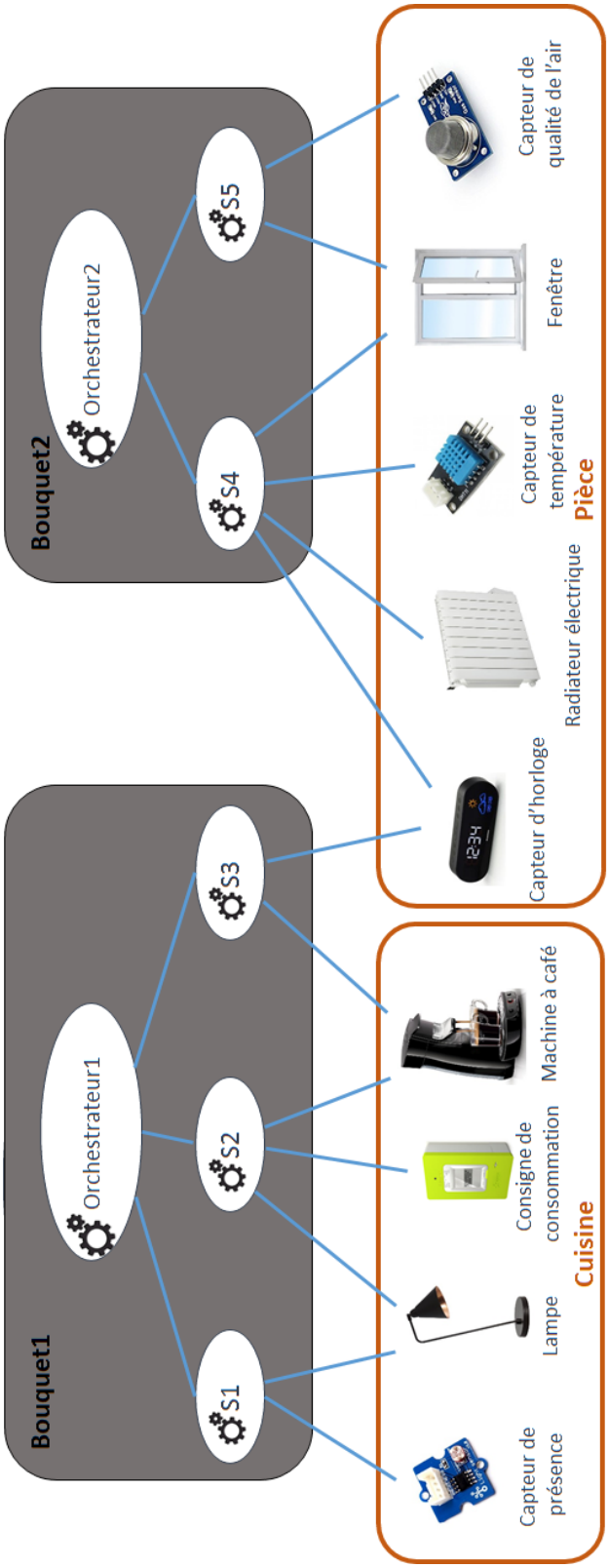


FIGURE 7.6: Deux bouquets de services formés à partir de cinq services.

résolution de conflits décide de favoriser un des deux services. Dans notre exemple, le résultat de résolution de conflits favorise le service de régulation de température. L'état de la fenêtre sera alors « fenêtre fermée » à l'issue de ce conflit d'accès.

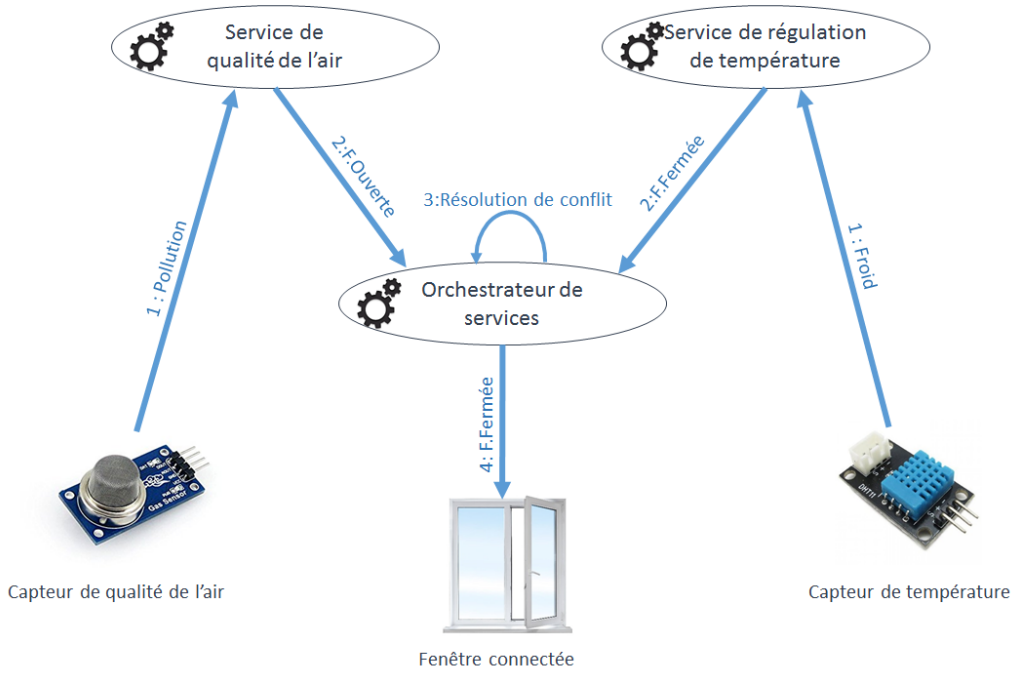


FIGURE 7.7: Scénario de résolution de conflits par priorités.

L'avantage de cette proposition est qu'elle est simple à mettre en oeuvre puisque l'algorithme d'orchestration prend les décisions en fonction des priorités de chaque service.

En revanche, cette méthode n'est pas flexible, parce que lorsqu'il y a un conflit entre deux services, l'un des deux est satisfait et l'autre ne l'est pas. L'orchestrateur, dans ce cas n'a pas le moyen de trouver un compromis qui convient aux deux services.

Proposition 2 : résolution de conflits par compromis

Pour pallier à la rigidité de la première proposition qui consiste à résoudre les conflits en se basant sur les priorités de services, nous proposons d'enrichir cette proposition pour permettre à l'orchestrateur de résoudre les conflits par compromis.

Nous proposons que les services soient plus « flexibles » lorsqu'ils proposent des changements d'états pour des actionneurs. Au lieu de proposer un seul état d'un objet, un service proposera plusieurs états qui peuvent lui convenir.

La figure 7.8 reprend l'exemple de conflit entre un service de régulation de température et un service de qualité de l'air sur l'objet fenêtre. Elle décrit un scénario de résolution de conflit par compromis où chaque service propose plusieurs états qui peuvent lui convenir

pour l'objet fenêtre. Le service de qualité de l'air propose les états {fenêtre ouverte ou fenêtre entre-ouverte} et le service de régulation de température propose les états {fenêtre fermée ou fenêtre entre-ouverte}. Lors de la résolution de conflits, l'orchestrateur choisit l'état fenêtre entre-ouverte qui est commun aux deux propositions et qui satisfait donc les deux services. Ce compromis permet de maximiser le nombre de services prioritaires qui sont satisfaits.

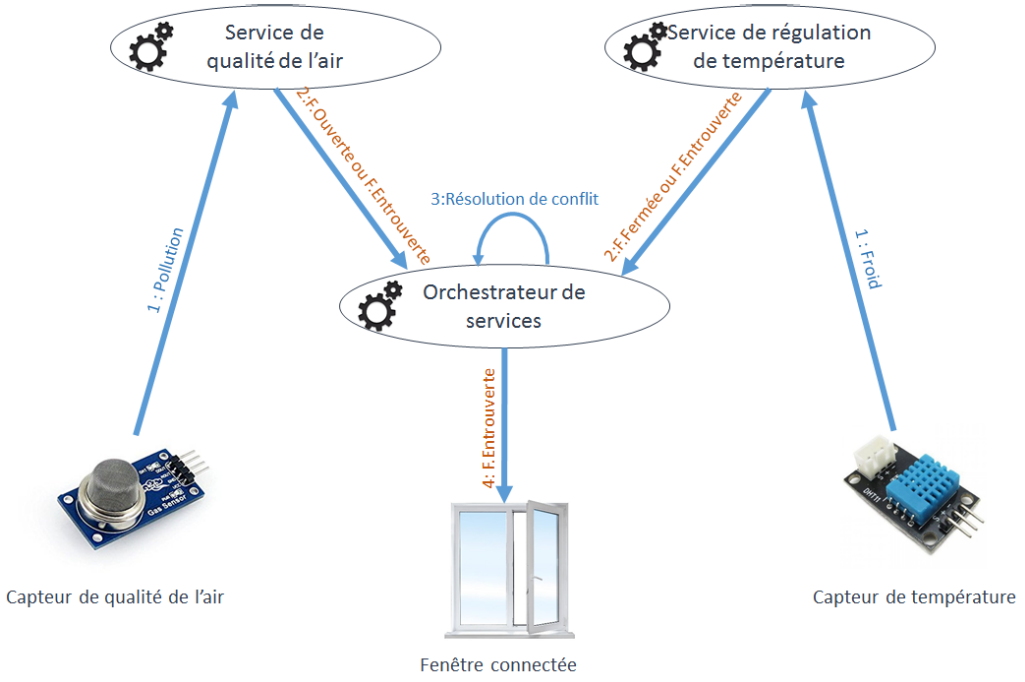


FIGURE 7.8: Scénario de résolution de conflit par compromis.

Il peut y avoir des cas où les ensembles d'états qui sont proposés par les services en conflit ne contiennent aucun état commun. Dans ce cas, l'orchestrateur favorise le service le plus prioritaire des services qui sont en conflit.

La question qui se pose à présent est : Que fait un service qui se voit refuser un changement d'état d'un objet ?

Nous proposons d'enrichir cette approche pour permettre aux services dont le souhait d'état n'est pas satisfait d'adapter leur comportement aux choix de l'orchestrateur.

Proposition 3 : Adaptation du comportement des services

La troisième proposition a pour but d'augmenter la flexibilité des services et de garantir le maintien du fonctionnement d'un service même lorsque l'état objectif d'un objet qui le concerne n'est pas satisfait. Nous proposons que les services soient « conscients » du

processus de gestion de conflit. Chaque service a conscience que les états objectifs qu'il propos sont sujets à négociation et qu'il est possible que des états autres que ceux qu'il a proposé soient choisis.

Nous proposons ainsi que chaque service décrive deux modes de fonctionnement : Fonctionnement en mode optimal et fonctionnement en mode dégradé. Chaque service peut adapter son comportement en fonction des choix d'états qui résultent du processus de négociation. Un service qui se ne voit pas ses états objectifs satisfaits n'est pas bloqué mais peut continuer à fonctionner dans un mode dégradé.

Le scénario représenté par la figure 7.9 enrichit le scénario de prise de décision de l'orchestrateur. Lorsque le service de qualité de l'air détecte la pollution, il envoie deux groupes d'états objectifs à l'orchestrateur. Le premier groupe représente les états objetcifs pour un fonctionnement en mode optimal (MO) et le deuxième groupe concerne les états objectifs pour un fonctionnement en mode dégradé (MD). En mode optimal, le service souhaite que la fenêtre soit ouverte. Si la fenêtre ne peut être ouverte, en mode dégradé, le service souhaite que le ventilateur soit allumé.

Dans cet exemple, le service de régulation de température est plus prioritaire que le service de qualité de l'air. La résolution de conflits se passe en deux temps.

D'abord, l'orchestrateur examine pour la fenêtre, s'il y a un état qui peut satisfaire les deux services en conflit. Dans cet exemple, il n'y a aucun état qui permet de satisfaire les deux services. L'orchestrateur décide de favoriser le service le plus prioritaire des deux c'est-à-dire le service de régulation de température. L'état fermé est choisi pour la fenêtre.

Ensuite, puisque les états objectifs du service de qualité de l'air en mode optimal n'ont pas pu être satisfaits. L'orchestrateur examine les états objectifs qu'il a défini pour son mode dégradé. Le mode dégradé du service de qualité de l'air souhaite que le ventilateur soit allumé. Dans cet exemple, le ventilateur est un objet auquel seul le service de qualité de l'air accède. Il n'y a donc pas de conflit d'accès au ventilateur. L'orchestrateur décide de satisfaire l'état objectif souhaité par le service de qualité de l'air et ainsi s'assurer que le ventilateur soit allumé.

L'adaptation du comportement des services via la description d'un mode de fonctionnement optimal et d'un mode de fonctionnement dégradé permet aux services d'être flexibles. Ainsi, lors d'un conflit, si les états objectifs d'un service ne sont pas satisfaits, ce dernier n'est pas bloqué et peut continuer à fonctionner en utilisant d'autres objets qui ne sont pas concernés par le conflit.

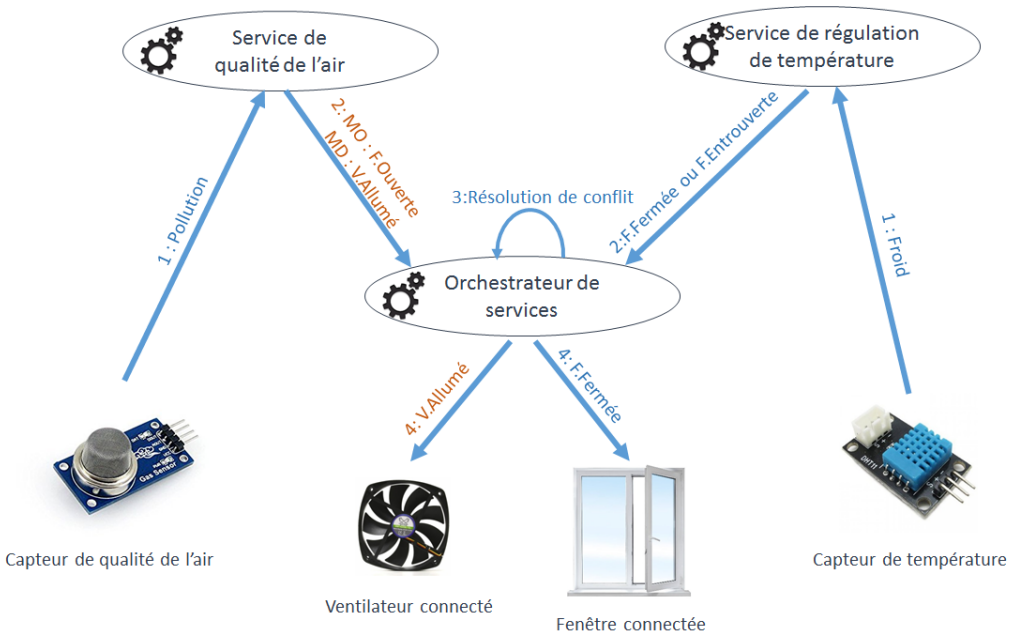


FIGURE 7.9: Scénario de prise de décision avec adaptation de comportement de services.

8 Présentation globale de l'architecture

"What is now proved was once only imagined"
William Blake

SOMMAIRE

Aujourd'hui dans l'Internet des Objets, les services sont fortement couplés aux objets pour lesquels ils sont dédiés. De ce fait, l'entreprise qui fournit les objets connectés fournit aussi les services qui gèrent ces objets. Dans le but de définir une stratégie de mise en place de services qui sont indépendants des objets connectés, nous redéfinissons la configuration des rôles sur le marché. Les différents rôles que nous avons identifiés sont présentés dans la section suivante.

8.1 Ecosystème d'un IdO ouvert

Dans le but de définir une stratégie de mise en place de services qui sont indépendants des objets connectés, nous redéfinissons la configuration des acteurs sur le marché de l'Internet des Objets. Sur le marché actuel de l'IdO, un seul acteur fournit les services et les objets connectés. Afin de permettre de répartir les rôles entre différents acteurs du marché, nous pensons qu'il est nécessaire d'avoir deux acteurs principaux au lieu d'un seul. Le premier acteur fournit les objets connectés et le deuxième acteur fournit les services qui opèrent sur les objets. Le troisième acteur que nous décrivons est l'utilisateur, il est présent également dans le marché actuel. Il s'agit des personnes physiques ou morales qui souhaitent s'équiper de technologies de l'Internet des Objets, à savoir des services et des objets connectés.

8.1.1 Fournisseur d'objets connectés

Nous définissons le fournisseur d'objets connectés comme étant l'acteur du marché de l'IdO qui met à disposition des utilisateurs les objets connectés. Le fournisseur d'objets connectés peut produire lui-même ces objets ou il peut aussi être un intermédiaire entre le producteur d'objets connectés et l'utilisateur. L'étude économique du marché de l'Internet des Objets n'étant pas le but de cette thèse, nous nous permettons de simplifier l'aspect économique en considérant que le fournisseur d'objets connectés est le producteur de ces objets.

8.1.2 Fournisseur de services

Le deuxième acteur du marché de l'IdO qui est important pour notre étude est le fournisseur de services. Le fournisseur de services met à disposition des utilisateurs un certain nombre de services de l'Internet des Objets. Il conçoit, développe et déploie ces services dans le but de satisfaire des besoins des utilisateurs. Le fournisseur de services peut également représenter une entité intermédiaire entre l'acteur qui conçoit et développe

les services et l'utilisateur. Dans ce cas, il ne s'occupera que du déploiement des services. Comme pour le fournisseur d'objets, nous simplifions la configuration économique du marché en considérant que le fournisseur de services s'occupe lui-même de la conception et du développement des services qu'il propose. Quant au déploiement de ces services, il peut être fait par un installateur ou par l'utilisateur lui-même.

Le fournisseur de services met à disposition des utilisateurs des services qui ne dépendent pas d'une certaine marque d'objets connectés. Les services sont destinés à être mis en place sur une configuration d'objets connectés qui ne soit pas préalablement connue par le fournisseur de services.

De plus, le fournisseur de services ne connait pas à priori les topologies sur lesquelles il déploiera les services. Le but est de fournir des services personnalisables en fonction de chaque lieu que l'utilisateur souhaite équiper.

8.1.3 Utilisateur

Couramment appelé consommateur en jargon économique, l'utilisateur est l'acteur qui souhaite s'équiper de technologies de l'Internet des Objets et bénéficier de services bâtis au dessus d'objets connectés. Nous l'appelons utilisateur et non consommateur car du point de vue de notre étude informatique, l'aspect qui est important est l'usage que fait l'utilisateur des technologies de l'IdO et des services associés et non son profil de consommateur. Au-delà de l'usage de ces technologies et services, l'utilisateur peut aussi être acteur dans le processus d'installation et de configuration des objets connectés et des services de l'IdO.

8.2 Les trois niveaux d'architecture

Nous décrivons dans cette section, notre proposition d'architecture qui s'appuie trois niveaux d'abstraction : un niveau sémantique, un niveau d'artefacts et un niveau de ressources. Cette architecture est inspirée des points de vue RM-ODP [27] [28] ainsi que des modèles des approches MDA [25]. Elle vise à servir de base à la conception et au développement de services pour l'IdO. Elle prescrit l'existence de référentiels types pour les objets connectés. Par conséquent, il permet la conception de services génériques qui sont indépendants des objets connectés et qui peuvent être liés à des objets spécifiques lors de la phase de déploiement, à condition que les types respectifs correspondent. Cette architecture fournit également une base de raisonnement sur l'orchestration du service. Elle permet de définir des orchestrateurs qui détectent, gèrent et résolvent les conflits pouvant survenir lorsque deux ou plusieurs services partagent un même actionneur. Comme

présenté par la figure 8.1, le niveau de ressources est le niveau le plus proche des objets connectés et le niveau sémantique est le niveau le plus proche des utilisateurs.

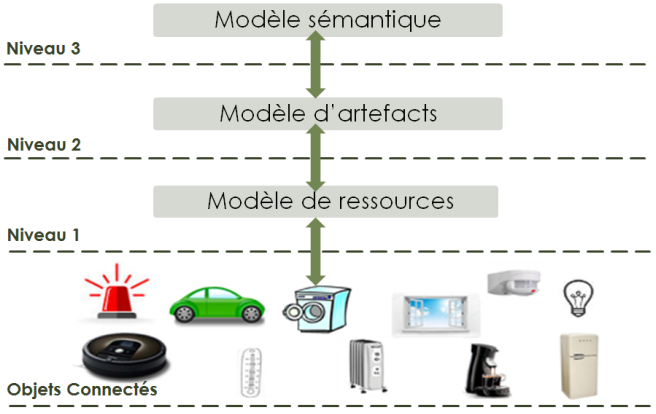


FIGURE 8.1: Architecture à trois niveaux de modèles.

Nous présentons une vue globale de chaque niveau d'architecture dans une sous-section en spécifiant les aspects techniques qu'il prend en charge. Nous procédons par une description du niveau le plus haut en allant vers le niveau le plus bas de l'architecture.

8.2.1 Niveau sémantique

Le niveau le plus haut de notre architecture est le niveau sémantique. Il s'agit du niveau le plus proche de l'utilisateur.

Simplement dit, dans ce niveau, le système est dirigé par des invariants. Il est considéré comme une collection de variables qui sont lues et écrites par les services. Les capteurs sont abstraits en tant que variables en lecture. Les actionneurs sont abstraits en tant que variables que les services peuvent uniquement écrire. Nous présentons également une troisième catégorie d'objets que nous appelons les captionneurs et qui correspondent aux actionneurs qui ont la capacité de capter et de communiquer leur état actuel. Au niveau de la sémantique, les captionneurs sont abstraits en tant que variables en lecture et écriture. L'état actuel du système est reflété par les valeurs actuelles des variables composant le système. L'environnement physique a la capacité de changer l'état des capteurs, tandis que les services se comportent comme des agents qui réagissent à ces changements en fournissant les valeurs que doivent prendre les variables d'actionneurs et de captionneurs.

A ce niveau d'architecture, nous traitons uniquement des notions d'états. Tous les aspects opérationnels, comme les échanges de messages entre les services ou les évènements, ne sont considérés que dans les niveaux d'artefacts et de ressources. Le niveau

sémantique abstrait toutes ces notions et fournit un moyen de haut niveau pour modéliser, organiser et résoudre les conflits entre les services.

8.2.2 Niveau d'artefacts

Le deuxième niveau de notre architecture s'articule autour d'un modèle d'artefacts. Ce niveau précise les aspects opérationnels du système. Il transforme les services qui ont décrits par des variables au niveau sémantique en transfert de messages et évènements. Les choix technologiques d'échanges synchrones ou asynchrones entre les objets et les services sont effectués à ce niveau.

Aussi, les changements d'état du système qui ont été décrits par des états de variables seront remplacés par les actions qui leur correspondent à ce niveau de l'architecture.

Le modèle d'artefacts s'appuie sur les (Business) Artefacts qui sont une approche émergente proposée par IBM durant la fin des années 2000 pour la modélisation de processus métier dans une entreprise [70] [71].

Les approches traditionnelles de modélisation de processus et flux métier sont centrées sur les flux d'activité qui sont couplés avec des données souvent pensées après coup ; ou elles sont basées sur des documents avec un traitement souvent pensé après coup. La plupart des Frameworks BPM (Business Process Management) [72] [73] utilisent des méta-modèles centrés sur les flux d'activités, et les données manipulées par ces processus sont considérées avec une moindre importance. D'autres approches définies par Glushko et al. [74] sont centrées sur les documents, avec un méta-modèle de processus généralement plus pauvre [71].

L'approche des Business Artefacts vise à combiner les aspects de données et de processus dans un même bloc de construction de base, ce bloc est appelé « Artefact ». Seules les entités métier clés pour le processus métier sont modélisées sous forme d'artefacts. Ces entités évoluent au fur et à mesure de l'exécution des opérations métier.

La figure 8.2 représente la structure d'un Artefact. Ainsi, l'artefact est formé de deux parties, un modèle d'information qui regroupe les données importantes pour le métier et un modèle de cycle de vie qui décrit l'évolution de ces données au fur et à mesure de l'exécution du processus métier.

Le modèle que nous proposons pour notre architecture s'appuie sur les travaux qui ont été menés autour des Business Artefacts. Les artefacts, dans le cadre de nos travaux permettent, dans un premier temps, de résoudre la problématique de pléthore de ressources. Ce modèle va permettre d'avoir des vues plus abstraites et plus générales du modèle des ressources.

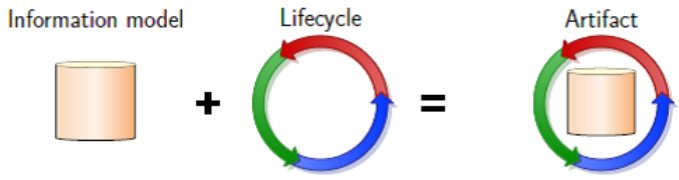


FIGURE 8.2: Business Artefact.

Ensuite, dans un second temps, les cycles de vie des artefacts enrichissent le modèle de ressources en apportant une dimension dynamique aux ressources. Ce modèle permet alors de décrire, non seulement les données liées aux objets mais aussi leurs évolutions dans le temps.

8.2.3 Niveau de ressources

Le niveau le plus bas de notre architecture s'appuie sur un modèle de ressources. Ce modèle a pour but de répondre à la problématique d'hétérogénéité des objets et de leurs communications dans l'internet des objets. Dans ce modèle, quel que soit l'objet connecté, sa typologie et ses caractéristiques, il est représenté sous forme d'une ressource. Les ressources sont adressables et disponibles sur internet. Le modèle de ressources permet donc d'uniformiser la nature des objets en les encapsulant dans une ressource. Ce modèle permet aussi d'uniformiser l'accès aux ressources. Contrairement aux architectures à base de services SOA (Service Oriented Architectures) où les méthodes d'accès aux services diffèrent d'un service à un autre, les architectures à base de ressources ROA (Resource Oriented Architectures) uniformisent l'accès aux ressources. Ces ressources sont alors accessibles et interrogeables de manière unique sur internet. Quelle que soit la ressource sur le Web, elle est toujours interrogée en utilisant les mêmes méthodes CRUD (Create, Retrieve, Update et Delete).

Le modèle de ressources s'appuie sur le style d'architecture REST (REpresentation State Transfer). REST a été proposé par Roy T. Fielding en 2000 lors de sa thèse de doctorat [46] et est destiné pour le développement d'applications distribuées. Il s'appuie sur la création de services faiblement couplés qui peuvent facilement être réutilisés. REST est un style d'architecture, il n'est donc pas lié à un ensemble spécifique de technologies. Pour nos travaux, nous nous focalisons sur les technologies spécifiques qui implémentent le Web comme un système Restful. L'élément de base des architectures Restful est la ressource. Ainsi, chaque donnée importante est représentée sous forme de ressource. REST reprend les principes fondamentaux du web, ceux qui ont contribué au succès, à l'évolutivité et à la modularité du web traditionnel [75]. A savoir, l'identification et l'adressabilité

des ressources, l'interface uniforme, l'hypermédia comme moteur d'état de l'application et les interactions sans état. Dans les architectures Restful, chaque ressource est identifiée à l'aide d'un identifiant unique.

Identification des ressources Sur le Web, les ressources sont identifiées en utilisant les URI (Uniform Resource Identifier). Chaque URI est composé d'une adresse URL (Uniform Resource Locator) et d'un nom unique URN (Uniform Resource Name). La contrainte d'identification des ressources assure le principe d'adressabilité qui stipule que toutes les données importantes sur le Web doivent posséder une adresse.

Interface uniforme La deuxième contrainte est la celle de l'interface uniforme. Afin de satisfaire cette contrainte, les ressources doivent être manipulées dans un format standardisé plutôt que dans un format spécifique aux besoins d'une application. Sur le Web, les ressources sont disponibles via les quatre verbes http : GET, PUT, POST et DELETE pour l'interrogation, la modification, la création et la suppression. Ainsi, les mises en œuvre sont découplées des services qu'elles fournissent.

L'hypermedia comme moteur de l'état de l'application La troisième contrainte dans les architectures Restful est la contrainte du HATEOAS (Hypermedia As The Engine Of Application State). Derrière cette contrainte, se cache un élément fondamental de la conception des API Restful. En effet, le moteur de l'état de l'application est l'hypermédia, il est alors possible de naviguer entre les ressources en parcourant les liens hypertextes qui se trouvent au niveau de ces dernières. L'HATEOAS garantit le principe de connectivité des architectures Restful. Ainsi, chaque ressource qui fait partie du Web est accessible via l'hypermédia depuis d'autres ressources.

Architecture sans état Enfin, la communication entre le client est le serveur est sans état, et l'état de l'application est rappelé par le client à chaque interaction. Autrement dit, chaque requête envoyée au serveur contient toutes les données nécessaires pour l'exécution de l'application par le serveur. Ainsi, les architectures Restful améliorent considérablement la possibilité de montée en charge. Le serveur n'a pas besoin de stocker l'état entre deux requêtes du même client. A la fin de l'exécution d'une requête, toutes les ressources du serveur sont libérées jusqu'à l'arrivée d'une autre requête.

Dans les architectures Restful, les interactions entre ressources se font par le mécanisme de requête/réponse. Afin de compléter ces échanges avec des échanges évènementiels asynchrones, indispensables dans le cadre de l'Internet de Objets, les communications

de type Requête / Réponse sont enrichies par le mécanisme publish/subscribe qui permet de souscrire à l'état d'une ressource et d'être informé au moment où la ressource change d'état sans avoir à l'interroger de nouveau. Les objets peuvent communiquer en utilisant le protocole HTTP en tant que protocole de la couche application et non uniquement comme un protocole de transport. Ainsi, les mêmes verbes HTTP servent pour l'exécution des mêmes actions quels que soient l'objet et la ressource concernés par ces actions et ce, via l'interface uniforme.

Le modèle de ressource vient directement au-dessus des objets connectés et répond à deux de nos problématiques : la disparité de la nature des objets et leur interconnectivité. Premièrement, dans le modèle de ressources, quelle que soit la nature des objets, ils sont représentés de manière uniforme sous forme d'une ressource. Ce modèle permet alors de lisser l'hétérogénéité des objets qui composent l'internet des objets. Deuxièmement, le modèle permet aussi d'uniformiser et de simplifier l'accès à ces ressources. En effet, dans ce modèle, l'accès aux ressources se fait uniquement en utilisant les quatre verbes HTTP. Cette propriété est très importante car notre système est destiné à s'adapter à son environnement, il doit être capable d'intégrer de nouveaux objets lorsqu'ils sont disponibles sur le réseau. Le fait d'avoir connaissance de la manière de communiquer avec ces objets avant même de les retrouver sur le réseau facilite grandement au système la découverte et l'intégration de ces nouveaux objets.

9 Niveau sémantique

"Il s'agit moins de penser davantage que de penser autrement."
Jean-Marie Domenach

SOMMAIRE

Ce niveau d'architecture adopte une approche déclarative qui est axée sur les besoins des utilisateurs. Ces besoins sont exprimés sous forme d'objectifs. Ainsi, les services que nous définissons sont décrits en termes d'objectifs à atteindre plutôt que d'actions à exécuter. A ce niveau de notre architecture, les services décrivent des règles déclaratives qui s'appuient sur des états du système. Le niveau sémantique de notre architecture s'articule autour d'un méta-modèle sémantique.

Dans les sections suivantes, nous décrivons de manière formelle le méta-modèle sur lequel s'appuie notre l'approche sémantique de conception et d'orchestration de services. Ce méta-modèle permet de concevoir la structure des objets et des services ainsi que les comportements de services. Puis, nous décrivons notre proposition de règles de validation qui permettent de vérifier la conformité de modèles de services par rapport au méta-modèle sémantique que nous proposons. Enfin, nous décrivons des stratégies flexibles d'orchestration de services et de résolution de conflits en s'appuyant sur l'approche sémantique.

9.1 Le méta-modèle sémantique

Le travail de notre thèse porte sur la conception et la composition de services génériques pour l'IdO. Comme nous l'avons décrit précédemment, les services génériques est une première version de services qui n'est pas fortement couplée avec des objets connectés. Cependant, au cours de nos recherches, nous avons pris conscience que certaines informations relatives aux objets sont nécessaires lors de la description des services génériques. Un service générique décrit ses objectifs et la manière de les atteindre en fonction des objets connectés. Aussi, il est nécessaire que les services expriment leurs besoins et leurs attentes relatives aux objets connectés dans leur version générique afin d'assurer une mise en correspondance cohérente entre les objets et les services déployés.

Si le service générique décrit de manière très précise les objets dont il a besoin pour fonctionner alors l'effort de produire des services génériques n'aura fait que maquiller le fort couplage entre les services et les objets. Si la description des objets par les services est trop précise, alors il ne sera plus possible de substituer un objet à un autre, ce qui conservera le fort couplage actuel entre les services et les objets.

Si, au contraire, la description des objets par le service générique n'est pas assez précise alors le risque est que lors de la phase de mise en correspondance, les services soient liés à des objets qu'ils ne sont pas en mesure de gérer.

La difficulté est de trouver un juste milieu qui permet d'assurer une mise en correspondance efficace entre les services et les objets tout en laissant assez de marge pour pouvoir substituer un objet à un autre.

Nous proposons de décrire les services génériques en deux parties : une partie structurelle et une partie comportementale. La partie structurelle décrit le service, son type, ainsi que ses besoins en termes d'objets connectés. La partie comportementale décrit de manière déclarative le comportement du service en fonction de ses objectifs.

9.1.1 Modélisation structurelle

La modélisation structurelle des services génériques vise à les décrire du point de vue de la structure. Chaque service générique possède ses propres caractéristiques en fonction des objectifs pour lesquels il est conçu. Dans cette section, nous représentons des caractéristiques qui sont communes aux services de l'IdO.

Lors de la phase de mise en correspondance entre les objets et les services, il y a une comparaison entre les éléments de structure qui sont décrit par les services et ceux décrits par les objets. Il est alors important que les mêmes types d'information soient présents sur les descriptions de services et celles des objets.

Dans ce qui suit, nous présentons le méta-modèle d'objets connectés. Ce dernier décrit les concepts qui doivent être exprimés par les objets au niveau sémantique pour que ces derniers puissent être mis en correspondance avec les services. Nous présentons ensuite le méta-modèle structurel des services génériques.

Modélisation des objets connectés

Dans cette section, nous modélisons les concepts sémantiques qui doivent être décrits par les objets connectés pour que ces derniers soient « compatibles » et puissent être mis en correspondance avec les services que nous décrivons. Ce méta-modèle décrit uniquement les concepts sémantiques des objets connectés. D'autres concepts opérationnels pourront compléter la description opérationnelle des objets.

Tel que décrit dans le chapitre 4, un objet connecté peut soit mesurer une grandeur variable de son environnement, soit agir sur son environnement ou les deux en même temps. La figure 9.1 représente les échanges de données entre l'environnement, les objets connectés et les services. Nous distinguons entre les capteurs, les actionneurs et les captionneurs. Les capteurs ont la capacité de mesurer des variables de l'environnement et de les transmettre aux services. Dans ce cas, les données vont de l'environnement au capteur puis aux services. Les actionneurs ont la capacité d'agir sur leur environnement mais ne peuvent communiquer sur l'état de l'environnement. Les données vont alors des services aux actionneurs puis se transforment en actions sur le monde réel. Les captionneurs quant à eux ont la capacité de mesurer une grandeur variable de l'environnement et d'agir dessus. Les

données transitent alors dans les deux sens entre les captionneurs et l'environnement et entre les captionneurs et les services.

Un exemple de capteur est un capteur de température qui a la capacité de mesurer la température, une grandeur variable de son environnement mais il ne peut pas agir dessus. Un exemple de captionneur est une fenêtre connectée, elle peut être actionnée à distance et peut aussi communiquer sur son état. Un exemple d'actionneur est un objet qui peut agir sur son environnement mais qui ne communique pas sur son état.

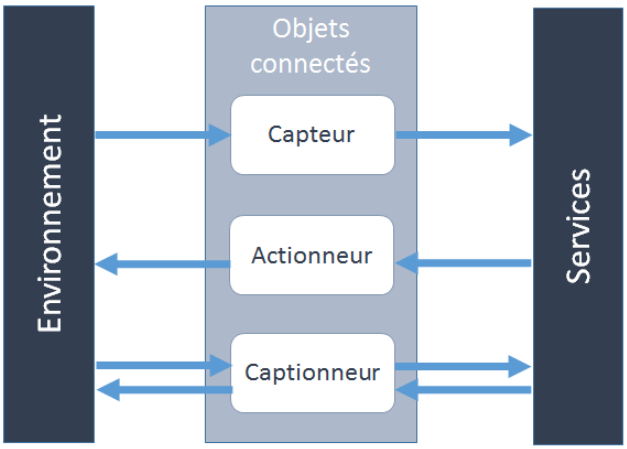


FIGURE 9.1: Echange de données entre l'environnement, les objets connectés et les services.

La figure 9.2 représente le méta-modèle sémantique des objets connectés. Il décrit les concepts qui doivent être décrits par les objets connectés pour permettre de les mettre en correspondance avec les services. Comme ce qui est décrit dans l'état de l'art 4, les descriptions des objets connectés sont souvent bien plus complexes que ce méta-modèle mais les concepts que nous décrivons sont souvent présents dans les modèles de description d'objets. Cette inclusion permet aux services d'être compatibles avec une large gamme d'objets connectés. Nous décrivons dans la suite chaque concept du méta-modèle ainsi que les relations qui lient ces concepts.

Le concept d'objet connecté (Connected Object) représente un objet physique ou une partie d'un objet physique connecté. Sur le marché de l'IdO, il peut exister des objets physiques complexes qui sont composés de plusieurs objets connectés. Par exemple, nous pouvons trouver plusieurs capteurs (par exemple : capteur de température, de qualité de l'air et de présence) sur un même dispositif physique. Lors de la conception de ce type d'objets complexes, chaque objet connecté est modélisé séparément. Trois objets connec-

tés seront représentés, un pour le capteur de température, un deuxième pour le capteur de qualité de l'air et un troisième pour le capteur de présence.

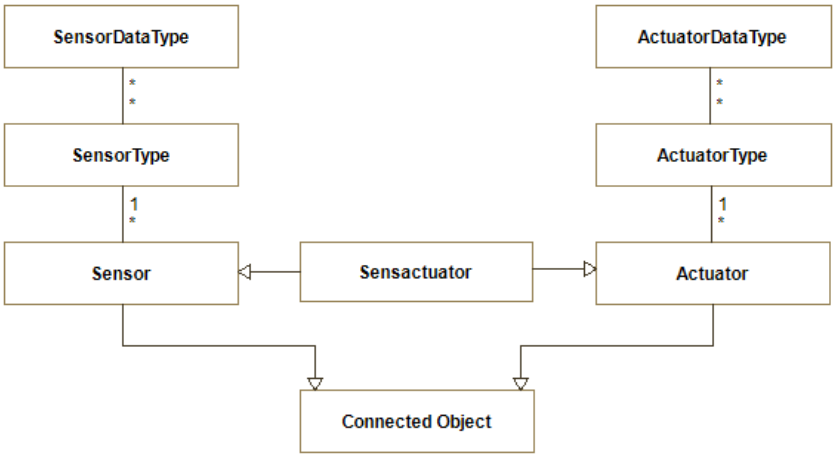


FIGURE 9.2: Méta-modèle d'objets connectés.

Le concept de capteur (Sensor) est un objet connecté. Il est en relation avec un concept type de capteur. Sémantiquement, cette relation représente une relation de classification des capteurs. Chaque capteur est lié au type de capteur auquel il appartient. La classification se fait en fonction des données qui sont mesurées par le capteur. « Capteur de température » et « capteur de luminosité » sont deux exemples de types de capteurs. Par exemple, le thermomètre Samsung est de type capteur de température.

Le concept type de capteur (SensorType) peut être en relation avec plusieurs capteurs. Il est en relation avec le concept Type de données de capteur.

Le concept de type de données de capteur (SensorDataType) représente le domaine de valeurs dans lequel varient les données mesurées par le capteur. Ce domaine de valeur peut être discret ou continu, fini ou infini. Le type de base « Integer » est un exemple de type de données de capteur discret et infini.

Le concept d'actionneur (Actuator) est un objet connecté et il est en relation avec un type d'actionneur.

Le concept de type d'actionneur (ActuatorType) un type d'actionneur peut être en relation avec plusieurs actionneurs. Il est aussi lié au concept Type de données d'actionneur. Un exemple de type d'actionneur peut être « actionneur de fenêtre ».

Le concept de type de données d'actionneur (ActuatorDataType) représente le domaine de valeurs sur lequel peut agir l'actionneur. Il s'agit des différents états que peut prendre l'objet physique.

Le concept de captionneur (Sensactuator) est un capteur et un actionneur. Dans ce cas le type de données d'actionneur est inclus dans le type de données de capteur. Ces deux ensembles peuvent contenir les mêmes éléments, mais le type de données de capteur peut aussi contenir plus d'éléments que le type de données d'actionneur. Le type de données de capteur qui contient des éléments supplémentaires au type de données d'actionneur signifie que le capteur peut détecter des états de l'objet qui ne sont pas atteignables via l'actionneur. Un exemple illustratif de captionneur est une fenêtre connectée qu'il est possible d'actionner pour la mettre en état « ouverte » ou « fermée ». le type de données d'actionneur est alors formé des deux états « ouverte » et « fermée ». Le capteur lié à cette fenêtre peut non seulement détecter les deux états « ouverte » et « fermée » mais aussi lorsque la fenêtre est cassée. Le type de données de capteur est alors formé de trois états « ouverte », « fermée » et « cassée ».

En somme, dans le but de faciliter l'interopérabilité entre les objets et les services génériques, il est important que les objets décrivent leurs types ainsi que les types de données qu'ils supportent. Il est aussi important pour les services de savoir si l'objet est un capteur, un actionneur ou un captionneur.

Passons à présent à la description structurelle des services génériques.

Modélisation structurelle des services génériques

Au moment de la conception du service, le concepteur n'a pas encore connaissance des objets réels sur lesquels le service sera déployé. Cette approche favorise à casser les dépendances technologiques qui existent entre les services et les objets connectés. Il contribue à la mise en place de services et d'objets dans un contexte de faible couplage.

La figure 9.3 représente le méta-modèle que nous proposons pour modéliser la partie structurelle des services génériques. Ce méta-modèle s'articule autour de treize concepts.

Le concept de service (Service) représente le service générique à modéliser. Il est décrit tout d'abord par un ensemble de rôles de capteurs, de rôles d'actionneurs et de rôles de captionneurs. Les concepts de rôles représentent les éléments qui sont nécessaires au bon fonctionnement du service. Au moment du déploiement, chaque rôle sera mis en correspondance avec un objet connecté qui est apte à jouer ce rôle. Le concept de Service est en

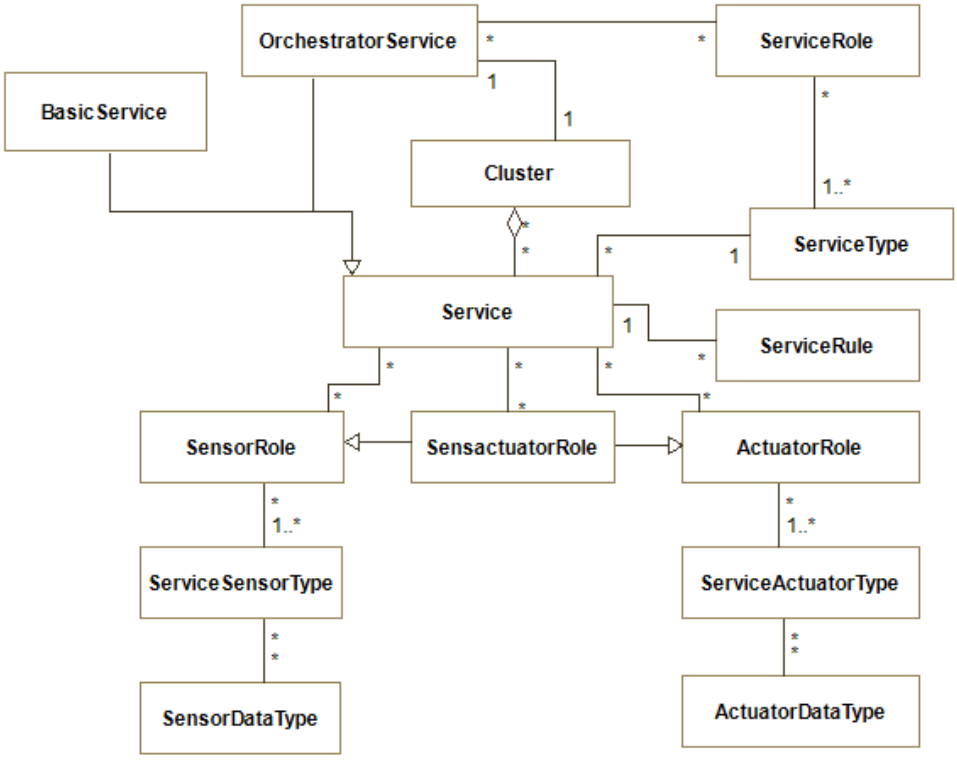


FIGURE 9.3: Méta-modèle structurel de services.

relation avec des règles de service. Ces règles représentent la description du comportement de service. Chaque service peut être en relation avec un type de service.

Nous représentons, de manière formelle un service comme étant un ensemble de rôles et un ensemble de règles.

$$S = (\mathcal{R}o\ ,\ \mathcal{R}e)$$

Avec :

« $\mathcal{R}o$ » l'ensemble des rôles du service S

« $\mathcal{R}e$ » l'ensemble des règles du service S

L'ensemble des rôles de service représente la description structurelle du service et l'ensemble des règles représente la description comportementale du service.

Les rôles que décrit un service sont soit des rôles de capteurs, soit des rôles d'actionneurs soit des rôles de captionneurs.

$$\mathcal{R}o = \mathcal{R}\mathcal{C} \cup \mathcal{R}\mathcal{A}\ \cup \mathcal{R}\mathcal{C}\mathcal{A}$$

Avec :

« $\mathcal{R}\mathcal{C}$ » l'ensemble des rôles de capteurs d'un service

« $\mathcal{RA}$ » l'ensemble des rôles d'actionneurs d'un service

« $\mathcal{RCA}$ » l'ensemble des rôles de captionneurs d'un service

L'ensemble de rôles de service est l'union des rôles de capteurs, des rôles d'actionneurs et des rôles de captionneurs.

Le concept de rôle de capteur (SensorRole) décrit les rôles de capteurs nécessaires au bon fonctionnement du service. Il est en relation avec un ou plusieurs types de capteur de service . Par exemple, le rôle « Thermomètre intérieur » est de rôle de capteur pour un service de régulation de température.

Le concept de rôle d'actionneur (ActuatorRole) décrit les rôles d'actionneurs nécessaires au bon fonctionnement du service. Il est en relation avec un ou plusieurs types d'actionneur de service. Cette relation signifie qu'un rôle d'actionneur peut être joué par plusieurs types d'actionneurs. « Actionneur de source d'apport de chaleur » est un exemple de rôle d'actionneur qui peut être défini par un service de régulation de température. Un rôle d'actionneur est en relation avec un ou plusieurs types d'actionneur de service ;

Le concept de rôle de captionneur (SensactuatorRole) décrit les rôles de captionneurs nécessaires au bon fonctionnement du service. Ce concept hérite des concepts de rôle de capteur et de rôle d'actionneur. Un rôle de captionneur hérite uniquement de rôles de capteur et d'actionneur qui sont compatibles. Par exemple, le rôle de captionneur « ouverture vers extérieur » hérite d'un rôle de capteur « Capteur d'ouverture vers extérieur » et d'un rôle d'actionneur « Actionneur d'ouverture vers extérieur ».

Le concept de type de capteur de service (ServiceSensorType) décrit le type de capteur attendu par le service pour remplir un rôle de capteur donné. Par exemple, le rôle « Thermomètre intérieur » peut être rempli par un capteur de type « Thermomètre ». Chaque type de capteur de service est en relation avec un type de données de capteur.

Le concept de type d'actionneur de service (ServiceActuatorType) les rôles de capteurs, le service décrit des rôles d'actionneurs qui seront, au moment du déploiement liés à des actionneurs correspondants à ces rôles. Par exemple, le rôle de « Actionneur d'ouverture vers extérieur » peut être joué par un actionneur de type « Actionneur de fenêtre ».

Le concept de type de données de capteur (SensorDataType) , tout comme dans le méta-modèle des objets connectée présenté en section 9.1.1, est un ensemble de valeurs

qui représente le domaine de variation des états du type de capteur. La différence avec le type de données de capteur du méta-modèle des objets connectés est qu'au niveau du méta-modèle de service, le type de donnée de capteur représente le domaine de variation des états du capteur qui est attendu par le service. L'ensemble des valeurs comprises dans le type des données de capteurs décrites par le service sont les valeurs qui peuvent être prises en compte par le service.

Le concept de type de données d'actionneur (ActuatorDataType) représente le type d'actionneur qui peut être pris en compte par le service pour remplir le rôle d'actionneur auquel il est relié. Chaque type d'actionneur de service est en relation avec un ou plusieurs types de données d'actionneur.

Nous définissons l'ensemble $\mathcal{V}al$ qui représente l'union de tous les types de données de capteurs et d'actionneurs.

$$\mathcal{V}al = \cup_{t \in \mathcal{T}} Dom(t)$$

Avec :

« $Dom(t)$ » le domaine de variation des valeurs d'un type de données de capteurs ou d'actionneurs « t ». L'ensemble des valeurs est l'union des domaines de valeur de chaque élément de $\mathcal{T}$

Le concept de type de service (ServiceType) est en relation avec un ensemble de services qui possèdent des caractéristiques communes.

Le concept de bouquet de services (Cluster) représente un ensemble de services qui sont orchestrés par un orchestrateur. Les services sont groupés par bouquet de services mais ces derniers peuvent exister en dehors d'un bouquet. Chaque service joue un rôle de service dans un bouquet. Les services qui font partie d'un même bouquet sont les services qui partagent deux à deux (directement ou par farmeture transitive) des actionneurs.

Le concept de service orchestrateur (OrchestratorService) représente les services qui possèdent la capacité d'orchestrer d'autres services. il définit un ensemble de rôles de services.

Le concept de service de base (BasicService) représente un service qui ne peut pas orchestrer d'autres services.

Dans la suite de ce document, nous utilisons le terme de service pour désigner les services de base et le terme orchestrateur pour désigner les services qui orchestrent d'autres services.

Le concept de rôle de service (ServiceRole) décrit l'ensemble des rôles de services qui peuvent être pris en compte par l'orchestrateur. Le rôle que joue un service pour l'orchestrateur est impliqué dans la prise de décision sur l'état objectif d'un objet. « Service prioritaire » est un exemple de rôle de service défini par un orchestrateur. Le rôle de service est en relation avec les types de service qui peuvent remplir ce rôle. Un exemple d'implémentation de rôles de services consiste à considérer chaque rôle comme étant une priorité de service.

En somme, un bouquet de services est un ensemble de services qui sont associés à des rôles. ces rôles peuvent représenter des priorités entre services. Nous formalisons les bouquets de services comme étant une liste ordonnée de services.

$$B = (S_1, ..., S_n) \text{ tel que } \forall\, i,\ j\ :\ S_i \ \underline{part^*}\ S_j$$

avec :
$(S_1, ..., S_n)$ les services du bouquet B classés par priorité croissante.

Le concept de règle de service (ServiceRule) décrit les aspects comportementaux du service. Ce concept sera décrit en détail dans la section 9.1.2.

Nous avons présenté le méta-modèle qui permet de décrire la partie structurelle des services génériques indépendamment des objets connectés. Le méta-modèle permet de définir des rôles de capteurs, d'actionneurs et de captionneur ainsi que les types de capteurs et d'actionneurs qui sont attendus par le service pour remplir ces rôles. Le service explicite aussi les types de données de capteurs et les types de données d'actionneurs qu'il peut traiter. La distinction entre un service orchestrateur et un service de base se situe dans le fait que le service orchestrateur définit, en plus du service de base, le concept de rôle de service. Le rôle de service est quant à lui en relation avec le type de service qui peut remplir ce rôle.

Nous allons, dans la section suivante, présenter le méta-modèle qui permet de décrire la partie comportementale du service.

9.1.2 Modélisation comportementale

Nous proposons dans ce chapitre, un méta-modèle comportemental, qui permet de décrire les aspects comportementaux de service. Nous proposons une approche pour répondre aux problématiques posées au niveau sémantique, en faisant abstraction des aspects opérationnels des services et des objets, tels que les échanges d'évènements par exemple. Ici, les services et objets sont considérés d'un point de vue sémantique. Nous considérons alors que le système est géré par des invariants.

Les approches déclaratives permettent de décrire des objectifs à atteindre plutôt que des procédures à suivre. Notre choix, pour le niveau sémantique, s'est alors naturellement porté sur une approche déclarative qui permet de décrire les services en un ensemble de règles de service qui spécifient des objectifs, en faisant abstraction sur la manière d'atteindre ces objectifs. Les aspects opérationnels des services et des objets ainsi que les procédures à suivre pour atteindre les objectifs sont de l'ordre des niveaux inférieurs de l'architecture.

Un autre avantage que présentent les approches déclaratives par rapport aux approches impératives est qu'elles sont plus dynamiques que les approches impératives. Si nous souhaitons modifier le comportement d'un service décrit en termes de règles de service, il suffit de modifier une partie de ces règles ou d'en rajouter d'autres. Tandis que si nous souhaitons modifier le comportement d'un service qui est décrit par une approche déclarative, en termes de workflow par exemple. Il faudra rééditer le workflow en entier. Les approches déclaratives présentent alors l'avantage d'être plus dynamiques que les approches impératives.

Le comportement des services est alors décrit sous forme déclarative en s'appuyant sur des règles de service. Nous décrivons dans cette section, le méta-modèle de règles que nous avons défini pour exprimer les règles de services au niveau sémantique en se basant sur les états du système. Ce méta-modèle est représenté par la figure 9.4.

Le concept de règle de service (ServiceRule) représente une règle de service. Chaque service peut mettre en oeuvre deux types de règles. Des règles de fonctionnement en mode normal et des règles de fonctionnement en mode dégradé. Les règles de fonctionnement en mode normal représentent le comportement d'un service lorsqu'il s'exécute seul dans et qu'il n'est pas orchestré par un orchestrateur. Les règles de fonctionnement en mode dégradé opèrent lorsqu'un service est orchestré. Ces règles permettent au service d'adapter son comportement dans le cas où un ou plusieurs objets sont verrouillés par d'autres services. Chaque règle est composée d'une prémisse et d'une conclusion.

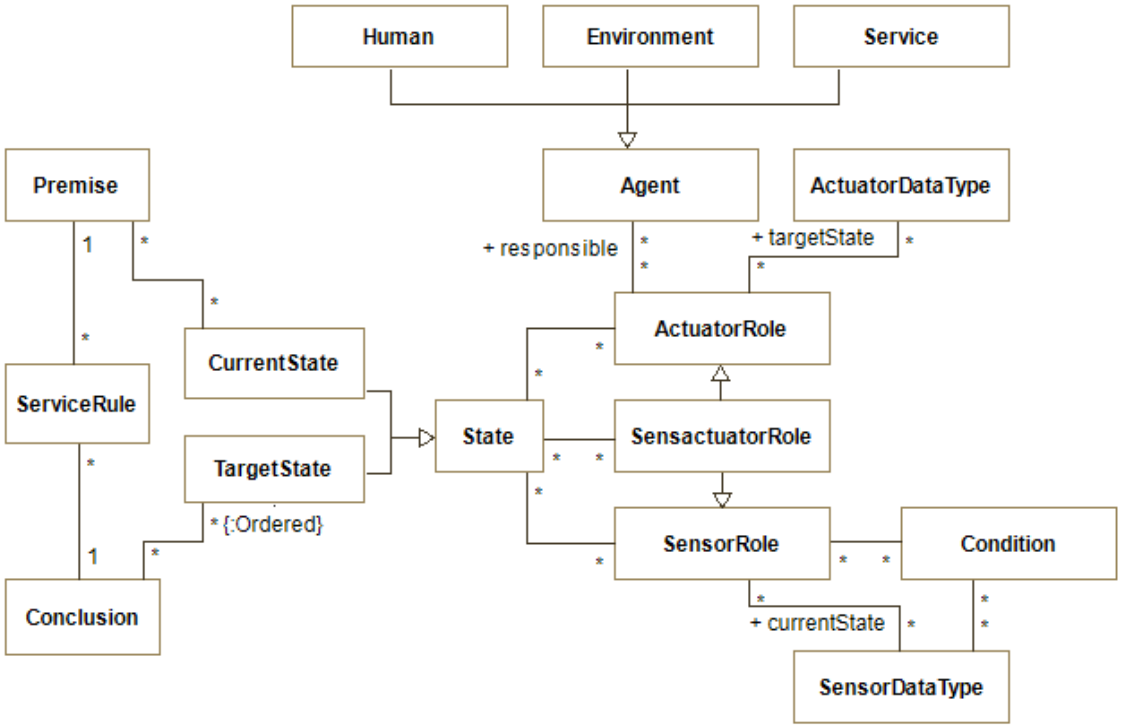


FIGURE 9.4: Méta-modèle comportemental de services.

$$R = (Pre, EtatSuivant)$$

Avec :
« *Pre* » la prémisse de la règle « *R* »
« *EtatSuivant* » est la conclusion de la règle « *R* »

Le concept de prémisse (Premise) représente la partie qui déclenche une règle de service. Elle dépend de l'état actuel du système.

Le concept de conclusion (Conclusion) représente la conséquence au déclenchement d'une règle de service. La conclusion est en relation avec un ensemble ordonné d'états objectifs. L'ensemble ordonné des états objectifs représente un choix d'états objectifs, du plus souhaité au moins souhaité par le service. Le fait de proposer plusieurs choix d'états objectifs en conclusion offre la possibilité à un éventuel orchestrateur de services de trouver un compromis entre les différents choix exprimés par des services en conflit. Le processus d'orchestration sera alors plus souple. L'orchestrateur pourra satisfaire plusieurs services en choisissant un état objectif qui paraît être le meilleur compromis globalement mais qui n'est pas nécessairement le premier choix pour chaque service.

Chaque conclusion de règle est composée d'un ensemble de couples (EtatObjectif, Priorité)

$$EtatSuivant = NS_1, ..., NS_k$$

Le concept d'état courant (CurrentState) est une spécialisation du concept d'état. Il représente l'état courant dans lequel se trouvent les objets qui peuvent influencer le comportement du service. L'état courant décrit uniquement des valeurs qui peuvent être lues. Il décrit donc uniquement des rôles de capteurs et de captionneurs.

Le concept d'état objectif (TargetState) représente les états des objets tels qu'ils sont souhaités par le service lorsqu'une règle est déclenchée. Ces objectifs concernent uniquement les objets sur lesquels le service peut agir, ils concernent alors uniquement les actionneurs et les captionneurs.

Le concept d'état (State) décrit un état du système. L'état du système, au moment de la conception d'un service générique est décrit en termes de rôles et de valeurs associées à ces rôles. Un état du système est décrit par un ensemble de rôles de capteurs, de rôles d'actionneurs et de rôles de captionneurs. Il s'agit des mêmes concepts de rôles de capteurs et de rôles d'actionneurs qui sont décrits dans la partie structurelle du méta-modèle de services représenté par la figure 9.3.

Le concept de rôle de capteur (SensorRole) décrit les rôles de capteurs qui font partie de la prémisse d'une règle. Il est en relation avec les concepts de type de données de capteurs et de condition.

Le concept de type de données de capteur (SensorDataType) représente les valeurs que peut prendre le rôle de capteur avec lequel il est en relation. Les types de données de capteurs est le même concept que celui décrit par le méta-modèle structurel de services représenté par la figure 9.3.

Le concept de rôle d'actionneur (ActuatorRole) décrit les rôles d'actionneurs qui font partie de la conclusion d'une règle. Il est en relation avec le concept de type de données d'actionneurs.

Le concept de type de données d'actionneur (ActuatorDataType) représente les valeurs que peut prendre le rôle d'actionneur avec lequel il est en relation. Les types de données d'actionneur est le même concept que celui décrit par le méta-modèle structurel de services représenté par la figure 9.3.

Le concept de condition (Condition) est une expression logique qui peut être évaluée à« vrai » ou « faux ». Cette expression est formée à partir d'un rôle de capteur et d'une valeur du type de données de capteur. Par exemple « thermomètre.valeur < 15 » est un exemple d'expression qui peut être évalué à vrai ou faux en fonction de la valeur renvoyée par le capteur thermomètre.

Le concept d'agent (Agent) représente l'entité qui est responsable du changement d'état d'un actionneur. Le responsable de changement d'état peut être soit un humain, soit un service soit l'environnement. Un service réagira de manière différente et adaptée à un même changement d'état, en fonction de l'entité qui a déclenché ce changement d'état. La prise en compte de l'agent qui déclenche le changement d'état permet de concevoir des services qui tiennent compte des actions de l'utilisateur, des actions des autres services et de l'environnement.

Le concept d'humain (Human) représente les êtres humains qui font partie du contexte d'exécution des services. Ce concept est une spécialisation du concept Agent, qui est responsable de l'état d'un actionneur. Il est important pour les services de tenir compte du responsable des changements d'états des objets dans la mesure où notre souhait est d'augmenter l'acceptabilité des services par les utilisateurs. Ainsi, ce modèle offre la possibilité aux services d'adapter leur comportement et de l'aligner avec celui désiré par les utilisateurs lorsque ces derniers agissent sur les objets.

Le concept de service (Service) est une spécialisation du concept Agent. Il peut être responsable de l'état d'un actionneur.

Le concept d'environnement (Environment) est une spécialisation du concept Agent. Il est, par défaut, responsable de tout changement d'état des actionneurs qui n'est causé ni par un humain ni par un service.

Le concept de rôle de captionneur (SensactuatorRole) est une spécialisation des concepts de rôle de capteur et de rôle d'actionneur. Il représente les rôles d'objets qui peuvent communiquer sur leurs états et sur lesquels le service peut agir.

Pour résumer, nous avons présenté la description des règles qui spécifient le comportement de services au niveau sémantique lors de la description des services génériques. Chaque règle est décrite par une prémisse et une conclusion. La prémisse décrit des conditions sur les états actuels des rôles de capteurs et de captionneurs. La conclusion décrit un ensemble ordonné d'états objectifs des rôles d'actionneurs et de captionneurs. Cet ensemble est une liste de choix qui peuvent satisfaire le service.

Pour illustrer notre approche reprenons l'exemple du conflit entre un service de régulation de température et un service de qualité de l'air. Si la température intérieure est trop élevée et que ce service souhaite ouvrir la fenêtre pour rafraichir la pièce, alors un exemple d'un ensemble ordonné d'états objectifs peut être (« fenêtre fermée » ou « fenêtre entre-ouverte »). Cet ensemble signifie que le service souhaite que la fenêtre soit en état « fermée », si ce n'est pas possible, alors il peut se satisfaire de l'état « entre-ouverte ». Si le service de qualité de l'air souhaite que cette même fenêtre soit en état « fenêtre ouverte » ou « fenêtre entre-ouverte » alors le choix qui satisfait les deux services est l'état « fenêtre entre-ouverte ». Même si cet état n'est pas le choix optimal pour chaque service séparément mais il représente un état acceptable pour les deux services. En somme, exprimer les conclusions en termes d'un ensemble d'états objectifs permet, lorsque plusieurs services opèrent en même temps, de satisfaire un nombre plus important de services en cas de conflit et de réduire le nombre de conflits bloquants entre services.

Il est à noter que ces règles subiront quelques transformations lorsque les services seront mis en correspondance avec les objets connectés à l'étape 7.1.7 du scénario de première mise en place de services et d'objets connectés (cf. chapitre 7). Elles sont exprimées en termes de rôles, lors de la mise en correspondance, chaque rôle sera remplacé par l'objet qui sera affecté à ce rôle.

Nous avons présenté les descriptions structurelle et comportementale des services et des orchestrateurs de services, nous allons à présent décrire ces méta-modèles en langage formel. Dans la section suivante, nous présentons des parties de la traduction de notre modèle en langage formel. Nous mettons en annexe la formalisation complète du méta-modèle.

Après avoir présenté le méta-modèle du niveau sémantique de notre proposition, nous reprenons le scénario de première mise en place de services et d'objets connectés que nous avons décrit dans le chapitre 7. Nous reprenons ce scénario en le rapprochant du méta-modèle et en décrivant, pour chaque étape, la partie du méta-modèle qui est concernée.

9.2 Règles de validation de services

Une contribution importante de notre thèse est la validation de services. Le niveau sémantique de notre architecture offre un moyen de concevoir des services pour l'Internet des Objets. Le comportement de ces services est décrit par des règles. Cependant, une des points critiques qui est souvent rencontré dans les systèmes à base de règles est de s'assurer que l'ensemble des règles est cohérent (cf. [33] [76]). Il est important de s'assurer qu'il n'y a pas de règles qui se contredisent entre elles. La vérification de cohérence des règles est importante lors de la mise en oeuvre du système de règle et aussi à chaque mise à jour d'une ou de plusieurs règles du système.

Pour valider le système de règles des services présenté en section 9.1.2, nous proposons deux vérifications de cohérence : la détection de contradictions et la détection d'oscillations.

9.2.1 Détection de contradictions

La détection de contradictions consiste à vérifier que les règles d'un même service sont exclusives. L'exclusivité des règles signifie que l'intersection des prémisses des règles est égale à l'ensemble vide. Cette vérification permet d'assurer que pour un état actuel donné, une seule règle du service est déclenchable. Cette condition permet de garantir que le service puisse prendre une décision sur les états objectifs de manière déterministe.

Prenons un exemple de deux règles d'un service de régulation de température qui ne sont pas exclusives. Supposons que ce service modifie l'état d'un radiateur en fonction de la température qu'il reçoit de la part d'un thermomètre via le paramètre thermomètre.étatActuel. Dans cet exemple, le radiateur est de type captionneur, son état peut être lu sur le paramètre radiateur.étatActuel, il peut être écrit via le paramètre radiateur.étatObjectif. Ses états objectifs peuvent prendre une valeur dans l'ensemble éteint, alluméFaible, alluméFort. Dans un but de schématisation, nous considérons la version déployée du service et non sa version générique. La différence entre ces deux versions est que dans la version déployée, les services ont déjà été mis en correspondance avec les objets connectés. Dans cette version, les règles sont écrites en fonction des objets connectés qui sont présents dans le contexte d'exécution du service.

— R1 : Si thermomère.étatActuel < 15 alors radiateur.étatObjectif = alluméFort.

— R2 : Si thermomètre.étatActuel > 13 alors radiateur.étatObjectif = alluméFaible.

Si nous considérons les règles R1 et R2 séparément, elles sont correctes. Mais la conjonction des deux est problématique. Dans le cas où la température est égale à 14°C

alors les deux règles sont déclenchables, en revanche, ces deux règles donnent deux états objectifs contradictoires pour l'objet radiateur. Ce dernier doit être alluméFort et allumé-Faible en même temps ! Ce qui est absurde.

Nous avons pris un exemple simple pour illustrer la problématique des règles non exclusives. Dans cet exemple, la contradiction peut être facilement détectée par le concepteur de services. En réalité, les règles de service peuvent être bien plus complexes que dans l'exemple. Les prémisses peuvent dépendre de plusieurs objets connectés et les conclusions aussi. Par ailleurs les conclusions des règles peuvent définir plusieurs choix possibles d'états objectifs pour un objet. Dans ce cas, si deux règles sont déclenchables en même temps, la problématique de construction et de classement des états objectifs du services se pose.

Prenons un deuxième exemple de règles contradictoires qui illustre la problématique de construction d'états objectif pour le service. Considérons un service de régulation de température qui régule la température en fonction de la température intérieure à la pièce, de la température extérieure et d'une consigne de température. Ce service agit sur un radiateur électrique et sur une fenêtre.

La première règle a pour objectif d'allumer le radiateur si la température intérieure est inférieure à la consigne de température. Dans le but d'être flexible, cette règle propose deux états objectifs classés par priorité pour le radiateur : alluméFort ou alluméFaible.

La deuxième règle considère la fenêtre comme étant une source potentielle d'apport de chaleur. Dans le cas où il fait plus chaud à l'extérieur qu'à l'intérieur, cette règle permet d'ouvrir la fenêtre. Dans le but d'être flexible, la conclusion de la règle décrit deux états objectifs classés par priorité pour la fenêtre : ouverte ou entre-ouverte.

— R1 : Si thermomètreIntérieure.étatActuel < consigneDeTemperature.étatActuel alors radiateur.étatObjectif $\in$ (alluméFort, alluméFaible).

— R2 : Si thermomètreExtérieure.étatActuel > thermomètreIntérieure.étatActuel alors fenêtre.étatObjectif $\in$ (ouverte, entre-ouverte).

Dans le cas où la consigne de température est de 19°C, la température intérieure est de 15°C et la température extérieure est de 17°C alors les prémisses des deux règles sont vraies, donc elles sont toutes les deux déclenchables. De plus, il n'y a pas de priorité entre les règles d'un même service, les deux règles ont alors la même importance. Dans ce cas, comment seront classés les états objectifs du service ? Deux choix sont possibles :

— ({radiateur.étatObjectif = alluméFort et fenêtre.étatObjectif = ouverte},
{radiateur.étatObjectif = alluméFaible et fenêtre.étatObjectif = ouverte},

{radiateur.étatObjectif = alluméFort et fenêtre.étatObjectif = entre-ouverte},
{radiateur.étatObjectif = alluméFaible et fenêtre.étatObjectif = entre-ouverte})

— ({radiateur.étatObjectif = alluméFort et fenêtre.étatObjectif = ouverte},
{radiateur.étatObjectif = alluméFort et fenêtre.étatObjectif = entre-ouverte},
{radiateur.étatObjectif = alluméFaible et fenêtre.étatObjectif = ouverte},
{radiateur.étatObjectif = alluméFaible et fenêtre.étatObjectif = entre-ouverte})

Le service a deux possibilités pour construire son état objectif. Ce qui n'est pas déterministe. Pour pallier à cette problématique, notre proposition s'appuie sur des règles qui sont exclusives. Pour un état donné, une seule règle d'un service est déclenchable.

Sur quel principe se base la détection de contradiction ? La vérification des règles pour détecter les contradictions permet de s'assurer que les règles d'un service sont exclusives, c'est à dire que l'intersection des prémisses des règles est égale à l'ensemble vide.

Les règles d'un service sont exclusives : pour un état actuel donné, une seul règle du service est déclenchable.

9.2.2 Détection d'oscillations

La détection d'oscillations consiste à détecter à priori les oscillations possible entre les règles d'un même service et entre les règles de plusieurs services qui sont gérés par un orchestrateur. Une oscillation se produit lorsque deux services demandent le changement d'état d'un objet de manière cyclique. Un service modifie l'état d'un objet, cette modification d'état déclenche un deuxième service qui modifie de nouveau l'état de l'objet, cette deuxième modification déclenche de nouveau le premier service qui modifie de nouveau l'état de l'objet et ainsi de suite. Une boucle infinie peut s'établir et l'état de l'objet n'arrive pas à atteindre un état stable.

Les oscillations peuvent survenir entre deux services mais aussi entre plusieurs services. Par exemple, un service A est déclenché, l'application de ses modifications d'états des objets déclenche un service B, dont les modifications d'états déclenchent un service C, dont les modifications d'états déclenchent de nouveau le service A et ainsi de suite. Le système n'arrivera alors pas à un état stable des objets.

La gestion des oscillations est assurée par la fonction d'orchestration. L'orchestrateur décide de l'état objectif d'un objet en tenant compte des états objectifs candidats de tous les services accédant à cet objet.

Aussi, ce type d'oscillations peut aussi survenir au sein d'un seul service où l'on peut avoir plusieurs règles qui se déclenchent mutuellement. Dans ce cas, les règles de validation de service peuvent permettre de vérifier qu'un service est valide et qu'il ne peut pas provoquer d'oscillation d'états d'un objet.

9.3 Stratégie d'orchestration

Nous décrivons dans cette section, la stratégie de prise de décision par l'orchestrateur lorsqu'un conflit arrive entre plusieurs services sur un même objet. Nous nous plaçons dans le contexte d'un bouquet qui comporte plusieurs services.

9.3.1 Analyse

Pour prendre sa décision, l'orchestrateur a besoin de connaître :

— Les priorités associées aux services qu'il orchestre.

— Pour les objets sur lesquels il y a un conflit, les états objectifs candidats de chaque service.

L'algorithme d'orchestration prend alors en entrée les priorités des services et les états objectifs candidats proposés par chaque service.

En sortie, l'algorithme d'orchestration fournit un état objectif sélectionné pour chaque objet concerné par le conflit.

L'algorithme d'orchestration définit la satisfaction de services. Cette satisfaction correspond à un tableau de chiffres binaires composées d'autant de cases que de services dans le bouquet. Chaque case correspond à un service et les services sont classés du plus prioritaire au moins prioritaire. Ainsi, la case la plus à gauche correspond au service le plus prioritaire et la plus à droite correspond au service le moins prioritaire.

Pour chaque état candidat, s'il satisfait un service i, alors la valeur 1 sera affectée à la case satisfaction[i]. Sinon, l'algorithme affecte la valeur 0 à la case satisfaction[i]. Ce tableau sera ensuite converti en nombre binaire.

La condition d'arrêt de cet algorithme est de trouver l'état candidat qui maximise la satisfaction de services.

Lorsque le tableau est converti en nombre binaire. La satisfaction associée à un état candidat qui satisfait uniquement le service le plus prioritaire est plus grande que la satisfaction associée à un état candidat qui satisfait tous les services sauf le service le plus prioritaire. Par exemple, si nous avons quatre services en conflits sur un objet avec deux

états objectifs candidats pour l'objet. Le premier état satisfait uniquement le service le plus prioritaire et le deuxième satisfait tous les services sauf le plus prioritaire.

— Satisfaction1 : 1000 = 8

— Satisfaction2 : 0111 = 7

Satisfaction1 > Satisfaction2 alors le premier état qui satisfait uniquement le service le plus prioritaire est sélectionné par l'orchestrateur.

9.3.2 Algorithme

```
EtatSélectionné : Etat ;
Sat : Tableau de chiffres binaires ;
Satisfaction, i : Entier ;
Fonction CalculerSatisfaction(E : Etat, Bouquet Tableau de Services classés par priorité) : Entier
    i ← 0 ; Satisfaction ← 0 ;
    Redim(Sat, Bouquet.NbElements) ;
    Pour (i←Bouquet.NbElements ; i ⩾ 1 ; i–) faire
        Si (E in Bouquet[i].EtatsCandidats) Alors
            Sat[i] ← 1 ;
        Sinon
            Sat[i] ← 0 ;
        Fin Si
    Fin Pour
    Pour (i←Bouquet.NbElements ; i ⩾ 1 ; i–) faire
        Satisfaction ← Satisfaction + Sat[i] x 2^i ;
    Fin Pour
    Retourner Satisfaction ;
Fin
```

Algorithme 1: Algorithme de calcul de satisfaction

```
Sat, Sat1, i : Entier ;
Fonction SélectionEtatObjectif(EtatsCandidats : Tableau d'Etats, Bouquet Tableau
de Services classés par priorité) : Etat
    Sat ← 0 ;
    Sat1 ← 0 ;
    Pour (i←1 ; i ≤ EtatsCandidats.NbElements ; i++) faire
        Sat1 ← CalculerSatisfaction (EtatsCandidats[i], Bouquet) ;
        Si (Sat1 <Sat) Alors
            EtatSélectionné ← EtatsCandidats[i] ;
            Sat ← Sat1 ;
        Fin Si
    Fin Pour
    Retourner EtatSélectionné ;
Fin
```

Algorithme 2: Algorithme de sélection d'état objectif

Pour résumer, ce chapitre décrit de manière formelle notre approche sémantique de conception et d'orchestration de services axée sur les besoins des utilisateurs. Cette approche s'appuie sur un méta-modèle sémantique qui permet de modéliser les objets connectés en décrivant leurs types et les types de données qu'ils peuvent fournir aux services. Le méta-modèle permet de modéliser des services génériques en décrivant leurs structure et leurs comportements en fonction de rôles d'objets connectés et de types de données. Ces rôles et types de données sont destinés à être mis en correspondance avec les types qui sont décrits par les objets connectés au moment du déploiement des services.

En plus de fournir le méta-modèle sémantique qui permet de concevoir des services génériques, nous proposons de mettre en place des règles de validation qui permettent de vérifier la conformité de modèles de services par rapport au méta-modèle.

Ce chapitre décrit enfin, des stratégies flexibles d'orchestration de services et de résolution de conflits via une approche algorithmique.

10 Formalisation

SOMMAIRE

Dans ce chapitre, nous présentons une formalisation directe du niveau sémantique. Cette formalisation fait partie d'un article [2] à paraitre et qui sera présenté à la conférence Modelsward 2018.

10.1 SmartHome

A smart home is defined as a triple made of a set of devices, a collection of services and a set of bindings that allow to deploy services on the devices of the smart home.

SmartHome = (DEV, Serv, Bindings)

10.2 Devices

The set of devices is a disjoint union of sensors (SE), actuators (AC) and sensactuators (SA).

$$DEV = SE \uplus AC \uplus SA$$

We will use d to denote any device and we will assume the existence of a mapping

$$kind : DEV \Rightarrow \{S, A, SA\}$$

with $d.kind = S \Leftrightarrow d \in SE(respectively\ A,\ SA)$.

Actually, as seen by services, devices represent read (sensors), write (actuators) and read/write (sensactuators) variables.

The set $RD = SE \cup SA$ is the subset of variables that can be read by services and $WD = SA \cup AC$ those that can be written.

We denote by Val_{DEV}, Val_{SE}, Val_{AC}, Val_{SA}, Val_{RD}, Val_{WD} the set of values taken respectively by all devices, sensors, actuators, sensactuators, readable and writable variables.

We let se denote any sensor, ac, any actuator, sa, any sensactuator, rd, any readable device (sensor or sensactuator) and wd any writable device (actuator or sensactuator).

10.3 State of a smart home

The running state of smart home is any function that maps devices to values. States can be global on all devices or partial on a subset of devices :

global state : $GS : DEV \rightarrow Val_{DEV}$
read state : $RS : RD \rightarrow Val_{RD}$
write state : $WS : WD \rightarrow Val_{WD}$

10.4 Services

Informally, a service is an active entity that acts on a set of associated roles , reading from the read roles and reacting by writing on the write roles.

More formally,

A service S is a couple $S = (Ro, Ru)$ with

Ro a set of roles

Ru a set of rules : $Ru = R_1, ..., R_n$

Ro is equipped with the *kind* function similar to DEV.

Hence, for a role r, $r.kind \in \{S, A, SA\}$ is its *kind* : sensor, actuator or sensactuator.

$Ro.RR = \{r \in Ro | r.kind \in \{S, SA\}\}$ is the subset of readable roles and $Ro.WR = \{r \in Ro | r.kind \in \{A, SA\}\}$ is the subset of writable roles.

Each role has an associated domain of values (which depends on its type).

We let $Val_{RR}(Ro)$ the set of values of all readable roles in Ro and $Val_{WR}(Ro)$ the set of all writable roles in Ro.

A rule $R \in Ru$ is of the format $R = (Pre, NS)$

with

Pre is the precondition of the rule. It is a predicate that evaluates on readable roles. The concrete syntax depends on the types of readable roles.

If rs is a readable state on roles i.e. $rs : Ro.RR \rightarrow Val_{RR}(Ro)$ then $Pre(rs) \in \{true, false\}$

NS is a priorized list of next states of write roles with

$NS = (ns_1, ..., ns_k)$

A next state ns is a partial function $ns : Ro.WR \rightarrow Val_{WR}(Ro)$

We denote $NS.WR$ the set of write roles appearing in NS.

A service is well formed iff the following condition holds

$$\forall\, R_i,\, R_j \in Ru \text{ with } i \neq j,\, R_i = (Pre_i,\, NS_i),\, R_j = \left(Pre_j,\, NS_j\right),\, \forall\, rs,$$

$$Pre_i(rs) = True \Rightarrow Pre_j(rs) = False$$

10.5 Bindings

Given a smart home equipped with a set of devices and a collection of independently defined services ready to be deployed, we need to define the notion of deployable and deployed services. To do so, we use, for each service, a bind function that maps roles to devices.

10.5.1 Valid binding

Given a well formed service, $S = (Ro, Ru)$ and $Bind$, a map, $Bind : Ro \rightarrow DEV$

Bind is said to be a valid deployment map with regard to S iff it satisfies the following two conditions :

(i) Kind compatibility : $\forall r \in Ro, Bind(r).kind = r.kind$

(ii) Injectivity on writable devices : $\forall\ r,\ r' : r \neq r'$ with $r\ \in_R o.WR,\ r'\ \in Ro.WR$: $Bind(r) \neq Bind(r')$

Condition (i) stipulates that roles are mapped on devices of the same kind. We will assume also that there is a compatibility between the types of roles and the types of devices they are bound to and that the domains of values are identical between roles and the bound devices.

Condition (ii) enforces that different wr writable roles cannot be bound to the same device.

10.6 Deployed services

A deployed service is a couple (DD, DRu) where DD is a set of devices and DRu is a set of rules acting on the set of devices. A deployed service is similar to a service, except that roles are replaced with devices.

Given a couple $(S, Bind)$, with S a well formed service and $Bind$ a valid deployment map, with $S = Ro, Ru$ and $Ru = \{(Pre_1, NS_1), ..., (Pre_n, NS_n)\}$

We can define a deployed service DS where roles are substituted with devices as follows :

$$Bind(S) = (Bind(Ro), Bind(Ru))$$

Where $Bind\,(Ro)$ is the codomain of Bind and $Bind\,(Ru)$ is the set of rules obtained by replacing in each rule of *Ru* roles by their corresponding bound devices.

Claim : if *S* is well formed then $Bind\,(S)$ is well formed.

10.7 Deployed smart home

Considering a smart home with devices *DEV*, services *Serv*, and *BIND*, a set of valid deployable maps, one can define a deployed smart home where services are replaced by deployed services. To that end, since different services can act on shared actuators, we need to introduce a mechanism to resolve conflicts between services accessing the same actuators.

A deployed smart home is a couple $(DEV, DServ)$ where *DEV* is a set of devices and *DServ* a list of prioritized deployed services, i.e. a totally ordered set of deployed services.

10.7.1 Cluster

A cluster is the unit for orchestration. Services of the same cluster are orchestrated based on their priorities given a read state.

A cluster of services *C* is an ordered list of deployed services compatible with the order in *DServ*, $(DS_1, ..., DS_n)$ and such that $\forall\ i,\ j\ :$
$DS_i\ sh^*\ DS_j$

where *sh* is a binary relation on deployed services defined by

$$DS\ sh\ DS' \leftrightarrow \exists\ wd\ such\ that\ wd\ \in\ DS\ and\ wd\ \in\ DS'$$

and sh^* is the transitive closure of *sh* i.e.

$$DS\ sh*\ DS'\ iff\ \exists\ \{DDS_1, ..., DDS_n\}\ \subset\ DSERV\ such\ that$$
$$DDS_1\ =\ DS\ \wedge\ DDS_k\ =\ DS'\ \wedge\ \forall\ i\ <\ k-1\ :\ DDS_i\ sh\ DDSi+1$$

10.7.2 Conflict resolution within a cluster of orchestrated services

Given a cluster $C\ =\ (DS_1, ..., DS_n)$ with $DS_i\ =\ (DD_i, DRui)$ where
$DRui\ =\ (Pre_i^1, NS_i^1), ...,\ (Pre_i^k,\ NS_i^k)$

For given read state RS, let $DS_{i_1}, ..., DS_{i_m}$ the subset of services for which one premise is true on RS.

Let us call Pre_{i_p} the premise true for service DS_{i_p} and NS_{i_p} the associated next state.

$NS_{i_1} = ns_{i_1}^1, ..., ns_{i_1}^{q_{i_1}}$

.

.

.

$NS_{i_m} = ns_{i_m}^1, ..., ns_{i_m}^{q_{i_m}}$

The next state that maximizes satisfaction of high priority service is given as follows :

1) We define the set Max_1, the set of next state that maximize the function f

$$f = 2^m(ns_{i_m}^1 \vee ... \vee ns_{i_m}^{q_{i_m}}) + ... + 2(ns_{i_1}^1 \vee ... \vee ns_{i_1}^{q_{i_1}})$$

2) for $ns \in Max_1$ we define function f'

$$f'(ns) = 2^m(2^{q_{i_m}}(ns \to ns_{i_m}^{q_{i_m}}) + ... + 2(ns \to ns_{i_m}^1)) + ... + 2(2^{q_{i_1}}(ns \to ns_{i_1}^{q_{i_1}}) + ... + 2(ns \to ns_{i_1}^1))$$

Let Max_2 the set of states ns' that maximizes function f'

Max_2 contains exactly one element that we call NS_{sat} which is selected as the next state corresponding to read state RS, that maximizes satisfaction.

11 Conclusion

La partie II décrit les contributions apportées par cette thèse.

D'abord, nous commençons par présenter une analyse informelle des problématiques que nous considérons dans la thèse. Nous étudions et analysons les problématiques de cloisonnement du marché de IdO dans le but de mettre en place un écosystème de services et d'objets ouvert et composable. Cette analyse prend forme à travers différents scénarios de mise en place, d'exécution et d'orchestration de services et d'objets connectés.

Nous définissons ensuite le cadre d'architecture à trois niveaux dans lequel s'inscrit l'approche sémantique proposée par cette thèse. Cette approche sémantique offre un niveau d'abstraction qui permet aux concepteurs de services de se concentrer sur les aspects fonctionnels des services et des objets. Le cadre d'architecture aborde, en plus de l'approche sémantique, des aspects plus opérationnels de mise en oeuvre de ces services (niveau Artefacts) dans des environnements techniques éventuellement hétérogènes (niveau Ressources).

Nous décrivons ensuite de manière formelle l'approche sémantique qui s'appuie sur un méta-modèle permettant de modéliser les objets connectés, la structure et le comportement des services génériques. En plus de décrire le méta-modèle sémantique, nous proposons de mettre en place des règles de validation qui permettent de vérifier la conformité de modèles de services par rapport au méta-modèle. En nous basant sur le méta-modèle sémantique, nous proposons des stratégies flexibles d'orchestration de services et de résolution de conflits qui sont décrites via une approche algorithmique.

Enfin, nous présentons une description formelle directe de l'approche sémantique. Cette formalisation présente les versions génériques des services et des objets connectés dans le cadre d'une maison connectée. Elle décrit les fonctions (bindings) qui font correspondre un service avec les objets connectés. Puis, les services et les objets sont représentés dans leurs formes déployées et mises en correspondance dans le cadre d'une

maison connectée. Les services sont regroupés par bouquets et la résolution de conflits dans un bouquet est formalisée.

Troisième partie

Application de la contribution

12 Introduction

La partie III du rapport concerne les différentes applications et mises en oeuvre qui ont été faites de notre contribution. Ces applications permettent de vérifier et d'illustrer l'applicabilité de notre approche. Trois mises en œeuvre sont décrites dans cette partie.

La première mise en œuvre s'inscrit dans le cadre d'une maison connectée et illustre le projet d'un utilisateur qui souhaite équiper une pièce de sa maison de technologies de l'IdO. L'utilisateur sélectionne un service générique et des objets connectés dans un catalogue. Nous illustrons les descriptions du service générique et des objets qui sont sélectionnés par l'utilisateur, puis nous décrivons le service et les objets dans leur version déployée et enfin nous illustrons la mise en correspondance entre le service et les objets déployés.

La deuxième mise en œuvre permet de vérifier la validité de services avec le langage et l'outil Alloy. Alloy s'appuie sur un solver SAT et permet de générer de manière automatique un grand espace (10^{60}) d'instances d'état sur lequel les vérifications de services ont lieu. Les instances d'états sur lesquelles Alloy effectue ses validations correspondent aux contextes d'exécution dans lesquels le service peut être amené à s'exécuter. Ces validations se font sur un service de régulation de température déployé. Nous considérons d'abord un service de régulation de température non valide, nous montrons le résultat de l'analyse avec Alloy. Puis, nous corrigeons le problème qui cause la non validité du service et enfin nous montrons le résultat de validation du service en Alloy.

La troisième mise en œuvre illustre une implémentation du méta-modèle sémantique sur lequel s'appuie notre contribution sous forme d'une ontologie OWL et de règles SWRL. Cette mise en œuvre permet d'effectuer une simulation de la gestion des conflits d'accès aux objets partagés dans le cas d'un bouquet de services et montre l'intérêt d'un algorithme d'orchestration fondé sur les priorités ainsi que l'optimisation de l'atteinte des objectifs de chaque service du bouquet.

13 Scénario de mise en place des services et des objets connectés

"Essentially, all models are wrong, but some are useful."
George Box.

SOMMAIRE

Nous décrivons dans ce chapitre un exemple d'application de notre contribution dans le cadre d'une maison connectée. Cet exemple illustre un projet d'un utilisateur qui souhaite équiper une pièce de sa maison de technologies de l'IdO.

Il sélectionne un service de régulation de température générique, puis il sélectionne les objets connectés, puis ces derniers sont déployés dans le contexte d'exécution et enfin le service est mis en correspondance avec les objets.

Nous illustrons les descriptions du service générique et des objets qui sont sélectionnés par l'utilisateur, puis nous décrivons le service et les objets dans leur version déployée et enfin nous illustrons la mise en correspondance entre le service et les objets.

13.1 Description d'un service de régulation de température générique

Le service de régulation de température est un service de base qui a pour but de réguler la température dans son contexte d'exécution. Le contexte d'exécution du service de régulation de température peut être une pièce de la maison, un ensemble de pièces, un étage ou la maison en entier. Ce service s'intéresse aux objets qui produisent de la chaleur et aux objets via lesquels il peut y avoir une perte de chaleur comme les ouvrants (Portes et fenêtres). Il s'intéresse également aux objets qui mesurent la température dans son contexte d'exécution (température intérieure) et en dehors de son contexte d'exécution(température extérieure). Il s'appuie sur une conigne de température qui représente un objectif de température à atteindre dans le contexte d'exécution.

Nous présentons, dans les deux sections suivantes, la description structurelle et comportementale du service de régulation de température.

13.1.1 Description structurelle

La description structurelle du service de régulation de température générique s'appuie sur les rôles de capteurs et de captionneurs suivants :

Les rôles de capteurs

- Mesure de température intérieure : Ce rôle a pour but de mesurer la température intérieure au contexte d'exécution du service.
  - ServiceSensorType1 : Capteur de température en degrés Celsius.
  - SensorDataType1 : [-50, +50].

— Mesure de température extérieure : Ce rôle a pour but de mesurer la température extérieure au contexte d'exécution du service.
  — ServiceSensorType1 : Capteur de température en degrés Celsius.
  — SensorDataType1 : [-50, +50].

Les rôles de captionneurs

— Source d'apport de chaleur : Ce rôle constitue un moyen d'apport de chaleur au contexte d'exécution du service.
  — ServiceSensorType1 : Capteur de radiateur électrique.
  — SensorDataType1 : { hors service, éteint, allumé }.
  — ServiceActuatorType1 : Actionneur de radiateur électrique.
  — ActuatorDataType1 : { éteint, allumé }.
  — ServiceSensorType2 : Capteur de chaudière électrique.
  — SensorDataType2 : { hors service, éteinte, allumée }.
  — ServiceActuatorType2 : Actionneur de chaudière électrique.
  — ActuatorDataType2 : { éteinte, allumée }.

— Source de perte de chaleur : Ce rôle constitue un éventuel moyen de perte de chaleur du contexte d'exécution du service.
  — ServiceSensorType1 : Capteur de fenêtre.
  — SensorDataType1 : { cassée, ouverte, fermée }.
  — ServiceActuatorType1 : Actionneur de fenêtre.
  — ActuatorDataType1 : { ouverte, fermée }.

— Consigne de température en degrés Celsius : elle représente la température objectif en degrés Celsius que le service doit maintenir dans son contexte d'exécution.
  — ServiceSensorType1 : Capteur de consigne de température.
  — SensorDataType1 : [+15, +30].
  — ServiceActuatorType1 : Actionneur de consigne de température.
  — ActuatorDataType1 : [+15, +30].

La figure 13.1 illustre la description structurelle du service de régulation de température. Ce service décrit deux rôles de capteurs et trois rôles de captionneurs.

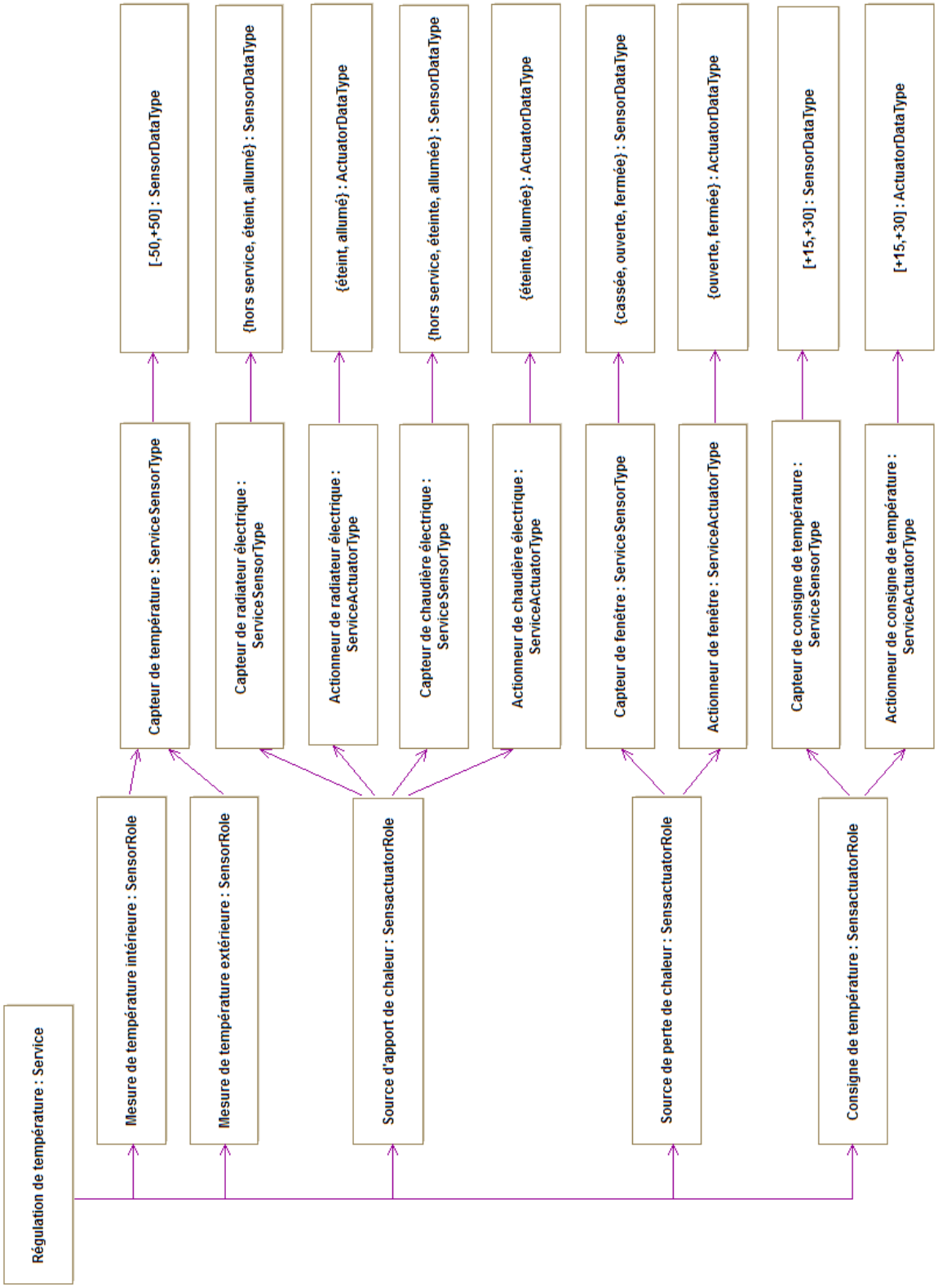


FIGURE 13.1: Description structurelle du service générique de régulation de température.

Les rôles de capteurs sont un rôle de mesure de température intérieure et un rôle de mesure de température extérieure. Les deux rôles peuvent être joué par des objets de même type : Capteur de température. Le type de capteur de température renvoie des valeurs comprises entre -50 et +50.

Les rôles de captionneurs sont : un rôle de source d'apport de chaleur, un rôle de source de perte de chaleur et un rôle de consigne de température.

Le rôle de source d'apport de chaleur peut être joué par deux types de captionneurs : un radiateur électrique ou une chaudière électrique. Comme nous l'avons décrit dans notre contribution (cf. chapitre 9), un objet de type captionneur est décrit par un type de capteur et un type d'actionneur. Chaque rôle de captionneur défini par le service décrit un type de capteur et un type d'actionneur. Le rôle de source d'apport de chaleur décrit alors un capteur et un actionneur de radiateur électrique et un capteur et un actionneur de chaudière électrique.

Les valeurs que renvoie le capteur de radiateur électrique varient dans l'ensemble {hors service, éteint, allumé} . Les valeurs de l'actionneur de radiateur électrique varient dans l'ensemble {éteint, allumé}.

Les valeurs que renvoie le capteur de chaudière électrique varient dans l'ensemble {hors service, éteinte, allumée} . Les valeurs de l'actionneur de chaudière électrique varient dans l'ensemble {éteinte, allumée}.

Le rôle de perte de chaleur peut être joué par un captionneur de type fenêtre qui décrit un capteur et un actionneur de fenêtre. Les valeurs renvoyées par le capteur de fenêtre varient dans l'ensemble {cassée, ouverte, fermée} et les valeurs de l'actionneur de fenêtre varient dans l'ensemble {ouverte, fermée}.

Le rôle de consigne de température est joué par un objet virtuel qui décrit un capteur de consigne de température et un actionneur de consigne de température. Les valeurs de capteur et d'actionneur de consigne de température varient dans l'intervalle [+15, +30].

Nous avons décrit la structure du service générique de régulation de température, nous passons à présent à la description comportementale de ce service.

13.1.2 Description comportementale

Nous décrivons dans cette section le comportement du service de régulation de température par un ensemble de règles génériques. Ces règles dépendent des rôles qui ont été définis en section 13.1.1.

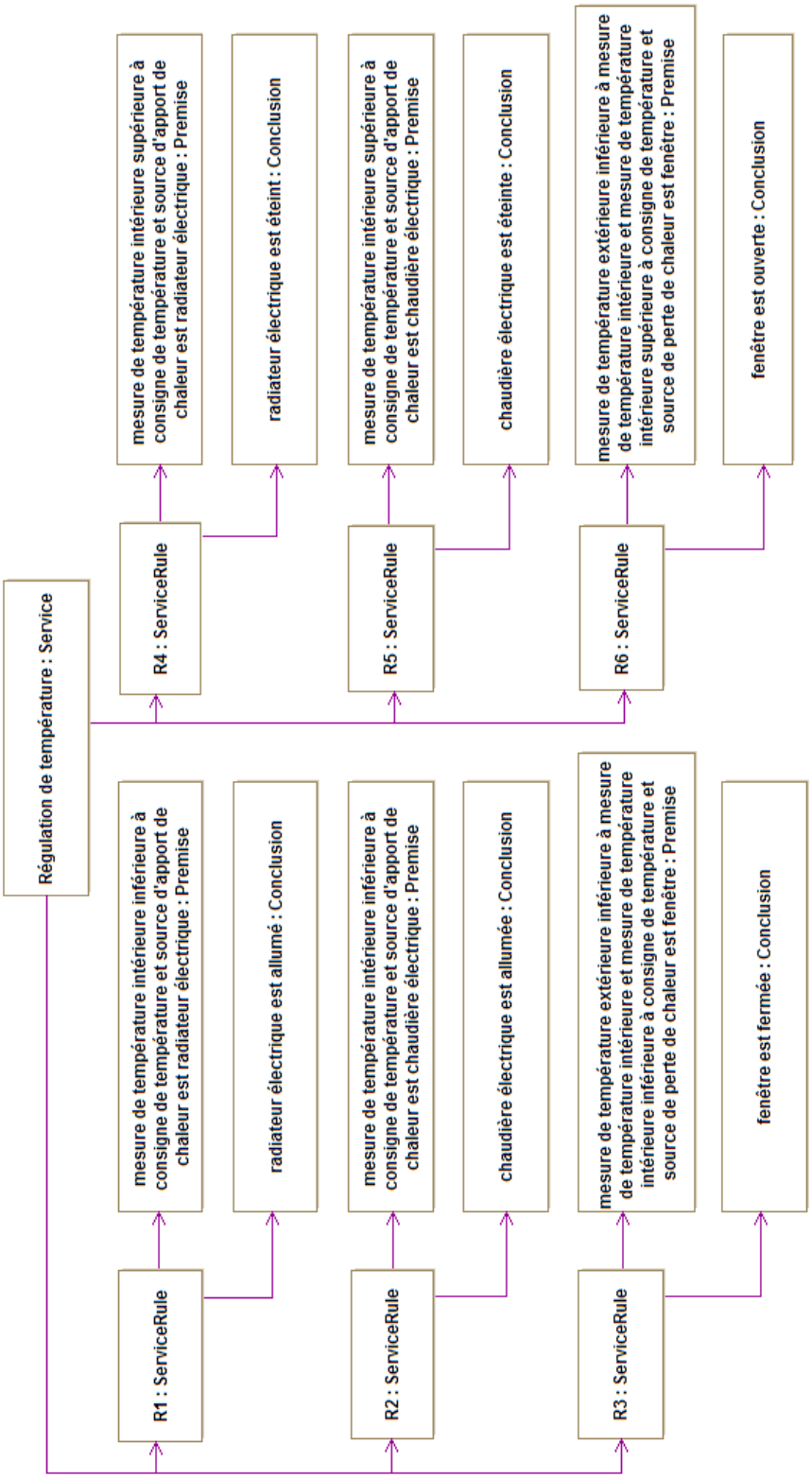


FIGURE 13.2: Description comportementale du service générique de régulation de température.

— Si la mesure de température intérieure est inférieure à la consigne de température et si la source d'apport de chaleur est un radiateur électrique alors le radiateur électrique doit être allumé.

— Si la mesure de température intérieure est inférieure à la consigne de température et si la source d'apport de chaleur est une chaudière électrique alors la chaudière électrique doit être allumée.

— Si la mesure de température extérieure est inférieure à la mesure de température intérieure et la mesure de température intérieure inférieure à consigne de température et la source de perte de chaleur est une fenêtre alors la fenêtre doit être fermée.

— Si la température intérieure est supérieure à la consigne de température et la source d'apport de chaleur est un radiateur électrique alors le radiateur électrique doit être éteint.

— Si la température intérieure est supérieure à la consigne de température et la source de chaleur est une chaudière électrique alors la chaudière électrique doit être éteinte.

— Si la mesure de température extérieure est inférieure à la mesure de tempéraure intérieure et la mesure de température intérieure est supérieure à la consigne de température et la source de perte de chaleur est une fenêtre alors la fenêtre doit être ouverte.

La figure 13.2 illustre les règles comportementales du service de régulation de température. Le service décrit six règles et chaque règle est décrite par sa prémisse et sa conclusion.

13.2 Description de types d'objets

Nous décrivons dans cette section quatre types d'objets : un capteur de température, un captionneur de type radiateur électrique décrivant un capteur et un actionneur de radiateur électrique, un captionneur de type fenêtre décrivant un capteur et un actionneur de fenêtre et un captionneur de type consigne de température décrivant un capteur et un actionneur de consigne de température.

Le capteur de température mesure des valeurs qui sont comprises dans l'intervalle [-50,+50]. Le capteur de radiateur électrique mesure des états du radiateurs électrique compris dans l'ensemble {hors service, éteint, allumé}. Les états objectifs possibles pour l'actionneur de radiateur électrique sont dans l'ensemble {éteint, allumé}. Le capteur de fenêtre peut détecter des états de la fenêtre qui sont compris dans l'ensemble {cassée, ouverte, fermée}. L'actionneur de fenêtre peut recevoir des états objectifs qui sont compris

dans l'ensemble {ouverte, fermée}. Les valeur de capteur et d'actionneur de consigne de température sont comprises dans l'intervalle [+10, +35].

Ces objets sont destinés à être déployés dans un contexte d'exécution puis mis en correspondance avec le service de régulation de température générique que nous avons décrit en section précédente.

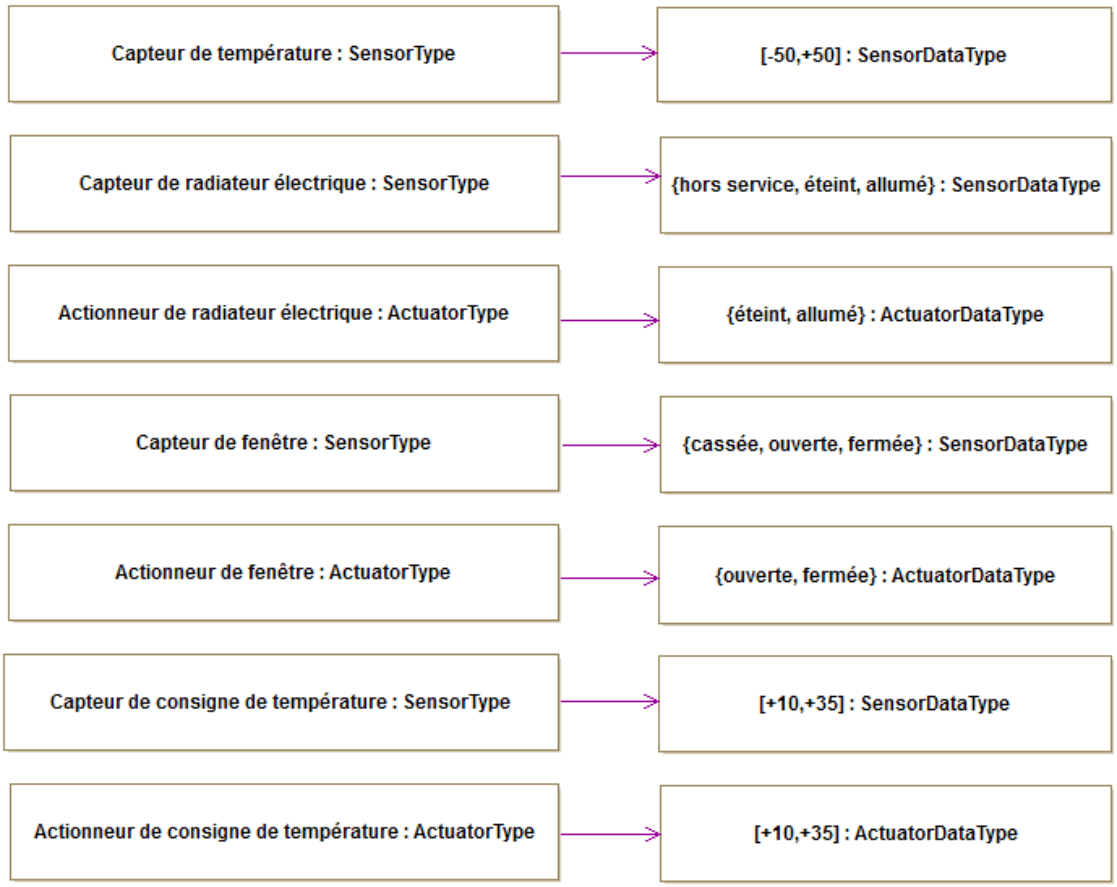


FIGURE 13.3: Description de types d'objets génériques.

Nous décrivons dans la section suivante, les objets et le service de régulation de température dans leurs versions déployées après le processus de déploiement.

13.3 Déploiement de services et d'objets

Nous reprenons dans cette section, l'exemple du service générique de régulation de température et les types d'objets décrits précédemment. Nous considérons un exemple de contexte d'exécution dans lequelle le service et les objets sont déployés. Puis, en nous

appuyant sur le processus de déploiement qui est décrit en chapitre 9, nous présentons la version déployée des objets et du service de régulation de température.

Nous commençons par décrire le contexte d'exécution puis le service de régulation de température et les objets déployés dans ce contexte d'exécution.

13.3.1 Contexte d'exécution

Nous considérons que le service de régulation de température est destiné à s'exécuter dans le contexte d'une pièce (pièce1) d'une maison connectée.

Les objets qui seront déployés dans la pièce1 sont : un capteur de température, un radiateur électrique et une fenêtre.

Un capteur de température extérieur à la pièce1 est aussi accessible au service de régulation de température.

La section suivante décrit, en langage naturel, les caractéristiques de chacun des objets présents dans la pièce.

13.3.2 Description du service et des objets déployés

Nous considérons, dans cette section les objets installés dans la pièce. C'est à dire, l'étape de sélection et d'installation des objets a déjà eu lieu et chaque objet possède un attribut de localisation en fonction de l'endroit physique dans lequel il se trouve.

La figure 13.4 illustre le service et les objets qui sont déployés dans le contexte d'exécution Pièce1. Ainsi, chaque entité est désormais instanciés et liée à son contexte d'exécution. Cette information apparaît au niveau du nom de chaque instance. Par exemple, « Radiateur électrique de pièce1 » est une instance du captionneur Radiateur électrique et cette instance est déployée dans la pièce1.

Chaque instance de capteur est liée à son type de capteur. Chaque instance de captionneur est liée à soon type de capteur et son type d'actionneur. Par exemple, l'instance « Radiateur électrique de pièce1 » est liée à un capteur de radiateur électrique qui est un type de capteur et elle est liée à un actionneur de radiateur électrique qui est un type d'actionneur.

Nous avons illustré le déploiement du service de régulation de température et des objets connectés dans le contexte de la pièce1. Nous présentons dans la section suivante la mise en correspondance entre le service et les objets déployés.

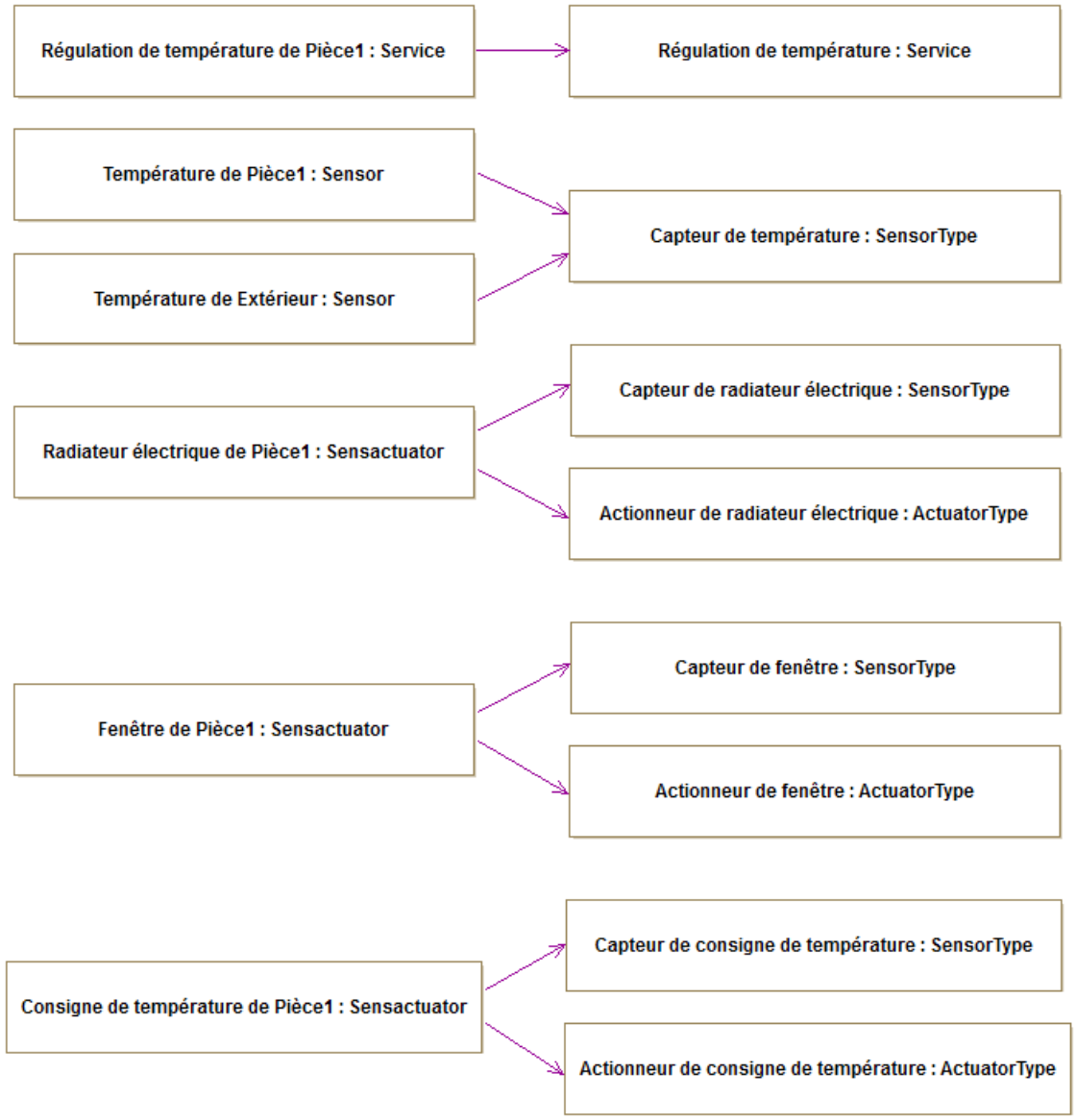


FIGURE 13.4: Description du service et des objets déployés dans la pièce1.

13.4 Mise en correspondance

En nous appuyant sur les règles de mise en correspondance entre les services et les objets connectées qui ont été décrites en chapitre 9, nous présentons la nouvelle version du service de régulation de température qui est déployé et lié aux objets connectés. Nous commençons par la description structurelle puis nous décrivons son comportement.

13.4.1 Description structurelle après la mise en correspondance

Après la mise en correspondance entre les services et les objets connectés, la description structurelle relie chaque rôle à l'objet qui le joue. Par exemple, le rôle de mesure de température intérieure est lié au capteurDeTempérature1 qui assure ce rôle.

La figure 13.5 illustre la description structurelle du service de régulation de température après sa mise en correspondance avec les objets présents dans la pièce1.

Ainsi, chaque rôle défini par le service est lié à un objet déployé. Le processus de mise en correspondance entre les services et les objets connectés est détaillé dans le chapitre 9. Pour rappel, chaque rôle qui est décrit par un service déployé est mis en correspondance avec un objet déployé.

Une première étape de sélection se fait en fonction du contexte d'exécution du service et des objets déployés. Ainsi, notre exemple de service de régulation de température aura accès aux objets qui se trouve dans son contexte d'exécution -c'est-à-dire les objets qui se trouvent dans la pièce1- et les objets qui se trouvent à l'extérieur.

Une deuxième étape de sélection se fait en fonction du type des objets déployés. Pour chaque rôle, un objet est sélectionné si son type est identique au type de capteur d'actionneur ou de captionneur qui est décrit par ce rôle.

Une troisième étape de sélection se fait parmi les objets retenus. Cette sélection se fait en comparant les types de données qui sont pris en charge par le service et ceux qui sont pris en charge par les objets. Un objet est sélectionné et mis en correspondance avec le service si son type de donnée est compatible avec celui qui est décrit par le service.

Nous avons présenté un exemple de description structurelle du service de régulation de température après la de mise en correspondance avec les objets. Nous présentons à présent la description comportementale de ce service.

13.4.2 Description comportementale après la mise en correspondance

Dans la version des règles du service de régulation de température mis en correspondance avec les objets, les prémisses et les conclusions des règles dépendent des objets qui jouent effectivement les rôles décrits par les objets.

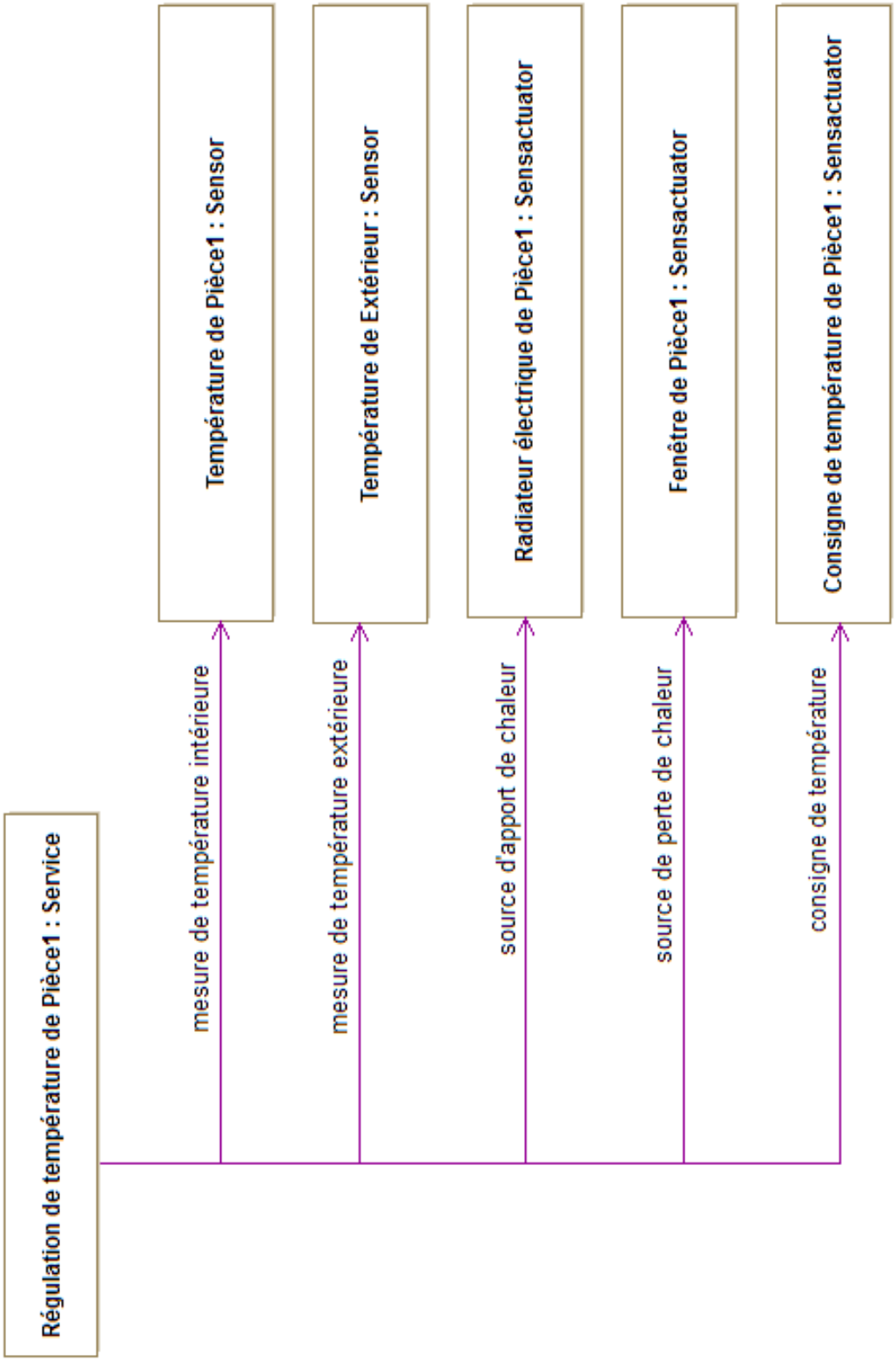


FIGURE 13.5: Description structurelle du service mis en correspondance avec les objets dans la pièce1.

Aussi, il y a des règles qui sont supprimée. Dans la version générique du service de régulation de température, le rôle de source d'apport de chaleur pouvait être joué par un objet de type radiateur électrique ou un objet de type chaudière électrique. Lors de la sélection d'objets, un radiateur électrique a été sélectionné pour jouer le rôle de source d'apport de chaleur. Les règles génériques qui décrivaient le comportement du service avec une chaudière électrique comme source d'apport de chaleur sont supprimées dans la version déployée et mise en correspondance du service. Dans notre exemple, le comportement du service générique est décrit par six règles et le comportement du service déployé et mis en correspondance est décrit par uniquement quatre règles.

De plus, les règles du service déployé et mis en correspondance décrivent directement les objets déployés. Par exemple, la conclusion de la règle1 considère le radiateur électrique de la pièce 1 c'est-à-dire l'instance de radiateur électrique qui est déployée.

Nous avons présenté dans ce chapitre un projet d'un utilisateur qui souhaite équiper une pièce de sa maison de technologies de l'IdO. Nous avons commencé par présenter un exemple de service de régulation de température générique qui est destiné à réguler la température d'une pièce d'une maison connectée. Nous avons décrit la version générique de ce service en précisant sa structure et son comportement. Dans sa version générique, le service décrit des rôles d'objets et des types d'objets qui peuvent jouer ces rôles.

Nous avons ensuite présenté une liste des objets que l'utilisateur a sélectionné pour son projet d'équipement. Les descriptions des objets ont été illustrés.

Nous avons ensuite présenté les versions déployées du service et des objets connectés. Enfin, nous avons décrit le service mis en correspondance avec les objets connectés.

Cet exemple de projet d'équipement nous a permis d'illustrer le scénario de mise en place de services et d'objets connectés qui a été présenté en section 7.1. Cet exemple montre la faisabilité de notre approche qui propose de décrire des services génériques indépendamment des objets connectés, puis de les déployer dans un contexte d'exécution et enfin de les mettre en correspondance en adaptant les descriptions structurelle et comportemmentale du service en fonction des objets qui sont effectivement déployés.

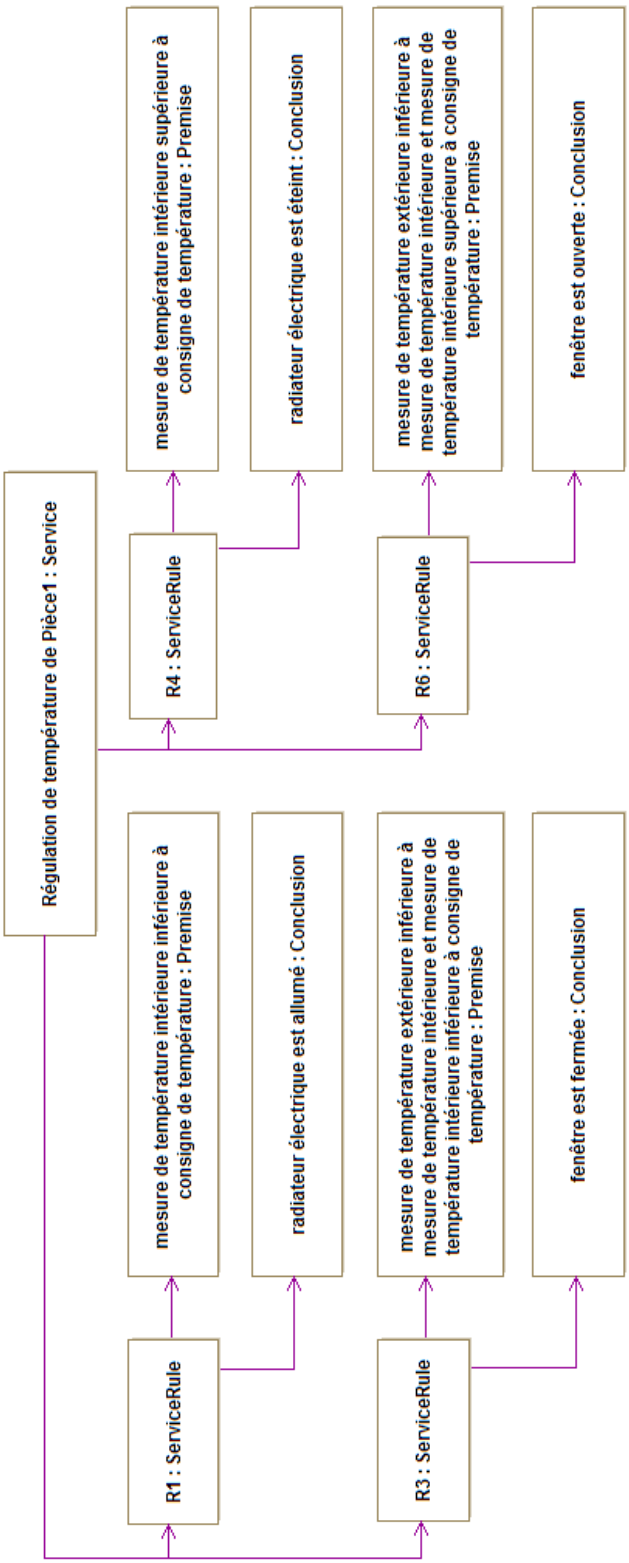


FIGURE 13.6: Description comportementale du service mis en correspondance avec les objets dans la pièce1.

14 Validation de services dans le contexte d'une maison connectée

Hind Bouali.

SOMMAIRE

Nous présentons dans ce chapitre une mise en œuvre qui permet de vérifier la validité de services en s'appuyant sur des règles de validation. Cette mise en œuvre concerne les règles de validation de services qui sont proposées dans la section 9.2.

Nous commençons par présenter le langage et outil Alloy qui nous permet de mettre en œuvre les services et la validation de service. Nous expliquons la méthodologie que nous avons utilisée pour traduire les modèles UML vers Alloy. Ensuite, nous présentons une implémentation de services et de règles de validation de services. Nous considérons un service de régulation de température non valide. Puis, nous démontrons que Alloy détecte bien la non validité du service à l'aide des règles de validation. Enfin, nous corrigeons le problème qui cause la non validité du service et enfin nous montrons le résultat de validation du service en Alloy.

Pour les besoins de simulations, nous considérons un service de régulation de température dans sa version déployée et mise en correpondance avec les objets connectés.

14.1 Alloy : Langage et outil de validation

Nous présentons dans cette section le langage et l'outil Alloy. Puis, nous décrivons la méthodologie que nous avons suivie pour traduire les modèles UML vers Alloy.

14.1.1 Présentation du langage et de l'outil

Alloy est un langage qui a été proposé par D.Jackson, le langage, sa logique et ses analyses sont décrits en détails dans le livre intitulé « Software Abstractions : Logic, Language, and Analysis » et paru en 2006 au MIT press [77]. Il permet de décrire les structures et fournit un outil pour les explorer. Il a été utilisé dans une large gamme d'applications allant de la découverte de failles dans les mécanismes de sécurité jusqu'à la conception de réseaux téléphoniques [78].

Le langage Alloy est profondément inspiré de la spécification Z [79], il permet de décrire toute la structure d'un système avec des notions mathématiques très réduites. Alloy est également fortement influencé par les notations de modélisation d'objets tels que UML, il facilite ainsi la classification des objets, l'association des propriétés aux objets et les expressions de navigation.

Alloy reprend les notions les plus simples de la théorie des ensembles et modélise toute la structure comme étant des ensembles et des relations. Il aussi des contraintes sur ces modèles en tant que faits, prédicats, fonctions ou assertions. Les faits correspondent à des contraintes qui doivent à tout moment être vérifiées par le système, elles correspondent

à des invariants. Les prédicats, les fonctions et les assertions sont des contraintes qui permettent de faire des vérifications (check) et des simulations (run) sur le modèle décrit en Alloy[80].

Alloy Analyzer offre deux types d'analyse : les vérifications (check) et les simulations (run). Les vérifications se font en se basant sur des prédicats ou des fonctions qui doivent être vérifiées. L'analyse se fait sur une portée (scope) définie par l'utilisateur. Cette portée décrit le nombre d'instances que l'analyseur va générer et sur lesquelles il effectuera ses vérifications et simulations. L'analyseur d'Alloy peut vérifier un nombre très important d'instances (soit 10^{60} cas ou plus) [78].

Pour faire des vérifications (check), l'analyseur génère des instances de modèle sur une portée (scope) qui a été définie par l'utilisateur. Ces instances sont générées dans le but de trouver un contrexemple qui ne vérifie pas la contrainte décrite par le prédicat ou la fonction qui est en train d'être vérifié. Si un contrexemple est trouvé alors l'analyseur peut affirmer que la contrainte n'est pas satisfaite. En revanche, si aucun contrexemple n'est trouvé par l'analyseur, alors il conclut que la contrainte est « probablement » vérifiées par le modèle. Plus la portée d'analyse est grande plus la probabilité de véracité de la contrainte est grande.

Les simulations (run) permettent d'explorer le modèle. Elles prennent en entrée les contraintes du modèle et essaient de trouver des instances qui satisfont ces contraintes. Les simulations se font aussi sur une portée qui est définie par l'utilisateur.

14.1.2 Traduction des modèles UML vers Alloy

Alloy est un langage qui est compatible avec les approche orientée objet et en a été inspiré. Néanmoins, ces deux approches ne se basent pas sur les mêmes concepts. Plusieurs travaux ont été menés dans le but de mettre en place des techniques et des outils pour transformer des modèles UML vers Alloy.

Par exemple, UML2Alloy [81] [82] [83] est une approche qui permet de transformer des diagrammes de classe UML et les contraintes de classe OCL vers le langage Alloy afin d'identifier les incohérences de conception. Les techniques de transformation qui ont été décrites ont été utilisées dans différents domaines dont la sécurité [84] ou le contrôle d'accès [81].

Ces travaux ont mené à l'établissement d'un manuel de référence [85] qui regroupe les règles de transformation des diagrammes de classe UML et leurs contraintes OCL vers Alloy. En nous appuyant sur ce manuel, nous avons transformé nos modèles UML et OCL vers Alloy afin de pouvoir les valider.

14.2 Implémentation et résultats

Lors de l'implémentation des services et règles sous Alloy, nous considérons les services dans leurs versions déployées et mises en correspondance avec les objets connectés. Les règles des services décrivent des états d'objets et non des états de rôles d'objets.

Aussi, étant donné la logique ensembliste du langage Alloy, nous représentons directement les modèles de services et d'objets sans décrire les méta-modèles desquels ils découlent. Toutefois, nous veillons à ce que les modèles de services soient conformes au méta-modèle sémantique.

Nous commençons par décrire la structure d'exemples d'objets et de services en Alloy. Ensuite, nous décrivons des exemples de comportements valides et non valides et des exemples de vérifications qui y sont faites. Nous donnons à titre illustratif, quelques fragments de code Alloy et des copies d'écran des résultats obtenus.

14.2.1 Description structurelle

Nous commençons par décrire en Alloy les types d'objets. La figure 14.1 illustre la descrption de trois types d'objets : un capteur de température, un radiateur et une fenêtre.

Le capteur de température possède un attribut : état actuel qui varie dans l'ensemble des entiers (Int). Le radiateur et la fenêtre sont des captionneurs, ils possèdent donc deux attributs : état actuel et état objectif. L'état actuel et l'état objectif du radiateur sont soit à Allumé (On) soit à Eteint (Off). Les états actuel et objectif de la fenêtre sont soit à Ouverte (Open) soit à Fermée (Close).

```
//Connected objects Data types
one sig Open, Close, On, Off {}
//Connected Objects
sig TemperatureSensor{
   currentState : Int
}
sig Heater {
   currentState : On+Off,
   targetState : On+Off,
}
sig Window{
   currentState : Open+Close,
   targetState : Open+Close,
}
```

FIGURE 14.1: Description d'objets connectés en Alloy.

Nous modélisons ensuite, un service de régulation de température qui est déployé dans un contexte d'exécution. Ce service est en relation avec un capteur de température intérieur, un capteur de température extérieur, un radiateur électrique et une fenêtre. La figure 14.2 représente le code Alloy de description structurelle du service de régulation de température.

```
//Temperature control service structural description
sig TemperatureControlService {
   insideTemperatureSensor : one TemperatureSensor,
   outsideTemperatureSensor : one TemperatureSensor,
   heater : one Heater,
   window : one Window
}
//Constraint on the Temperature control service
//The inside temperature sensor and outside temperature sensor
//of the same temperature control service must be different
fact {
   all x: TemperatureControlService, y, z:TemperatureSensor |
   (x.insideTemperatureSensor = y and x.outsideTemperatureSensor = z)
   implies
   (y & z = none)
}
```

FIGURE 14.2: Description structurelle du service de régulation de température en Alloy.

En plus de la description structurelle, nous décrivons une contrainte qui dit que le capteur de température intérieur et le capteur de température extérieur doivent être deux objets distincts.

14.2.2 Description comportementale d'un service non valide

Nous décrivons dans cette section le comportement de notre exemple de service de régulation de température. Pour des fins de démonstration, nous considérons un service où les règles peuvent se contredire dans un contexte donné.

Le comportement du service est décrit par un ensemble de règles dont deux qui peuvent se contredire car elles ne sont pas exclusives. Pour des fins de démonstration, nous décrivons uniquement ces deux règles qui peuvent se contredire :

— R1 : Si l'état actuel du capteur de température est inférieur ou égal à 17 alors l'état objectif du radiateur doit être égal à Allumé.

— R2 : Si l'état actuel du capteur de température est supérieur ou égal à 17 alors l'état objectif du radiateur doit être égal à Eteint.

Les règles R1 et R2 provoquent un comportement non valide du service dans le cas où la température intérieure est égale à 17. Dans ce cas les prémisses des deux règles sont vraies. Le radiateur doit alors être Allumé et éteint.

La figure 14.3 décrit la représentation de ces deux règles en Alloy. Elles sont décrites par des faits (fact). Les faits sont des contraintes qui sont toujours vérifiées pour le système.

```
//Temperature control service behavioral description : invalid behavior
//Rule 1 : if the inside temperature is less than or equal to 17, then turn on the heater
fact {
   all x: TemperatureControlService, y:TemperatureSensor, z:Heater |
   (y in x.insideTemperatureSensor and z in x.heater and y.currentState <= 17)
   implies
   (On in z.targetState)
}
//Rule 2 : if the inside temperature is greater than or equal to 17, then turn off the heater
fact {
   all x: TemperatureControlService, y:TemperatureSensor, z:Heater |
   (y in x.insideTemperatureSensor and z in x.heater and y.currentState >= 17)
   implies
   (Off in z.targetState)
}
```

FIGURE 14.3: Exemple de règles d'un service de régulation de température qui peuvent se contredire.

La figure 14.4 représente un exemple de règle de validation du service de régulation de température. Cette règle écrite sous forme d'une assertion permet de vérifier que ce service affecte, quel que soit son contexte d'exécution, un seul état objectif à l'objet Radiateur.

```
//Assertion that checks that the heater has exactly one target state
assert HeaterTargetState{
   all x : TemperatureControlService, y : Heater|
   x.heater = y implies   #y.targetState = 1
}
//Check command
check HeaterTargetState for 3 but 1 TemperatureControlService, 1 Heater, 1 Window
```

FIGURE 14.4: Assertion en Alloy pour détecter le nombre d'états objectifs d'un objet.

L'assertion n'est pas vraie dans le cas où le radiateur a deux états objectifs. Typiquement, elle n'est pas vraie lorsque la température intérieure est égale à 17°C. Dans ce cas, la radiateur est en relation avec deux états objectifs. La figure 14.5 représente un contre exemple qui a été trouvé par le solver SAT et qui pour la non validité du service de régulation de température. Cette figure est celle qui a été générée par l'outil Alloy. Dans la figure 14.5, nous avons mis en évidence uniquement les liens des états objectifs du radiateur. Ce dernier est relié par un lien « état objectif » à l'état Allumé (On) et à l'état Eteint (Off).

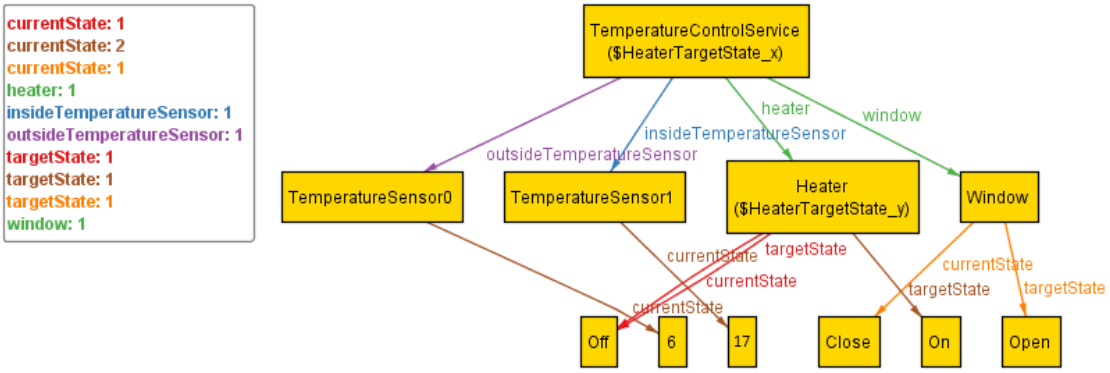


FIGURE 14.5: Contre-exemple d'instance de modèle qui invalide l'assertion.

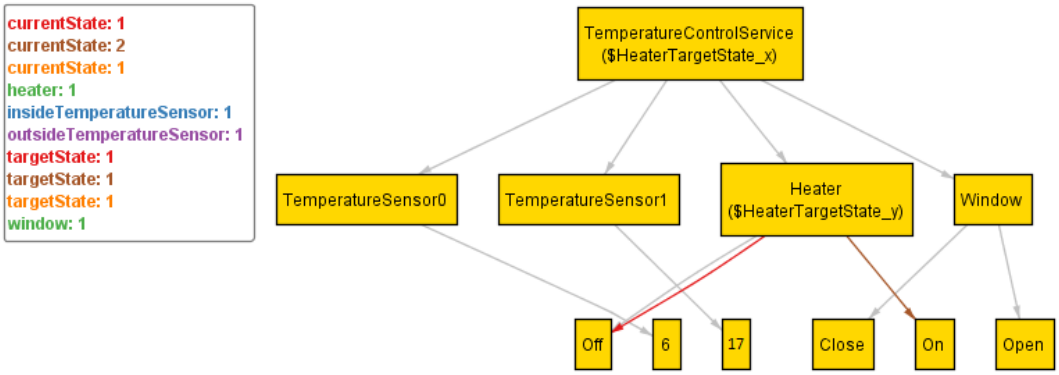


FIGURE 14.6: Contre-exemple d'instance de modèle qui invalide l'assertion : Mise en évidence de la non validité.

14.2.3 Description comportementale d'un service valide

Nous montrons dans cette section, les corrections que nous avons apportées au service de régulation de température non valide. Puis, nous présentons les résultats de validation par Alloy.

Le problème de non validité du service de régulation de température était dû à ses deux règles non exclusives. Nous avons alors modifié ces règles pour qu'elles soient exclusives et que leurs prémisses ne puissent pas être vraies en même temps.

La nouvelle version des règles du service est la suivante :

— R1 : Si l'état actuel du capteur de température est inférieur ou égal à 17 alors l'état objectif du radiateur doit être égal à Allumé.

— R2 : Si l'état actuel du capteur de température est strictement supérieur à 17 alors l'état objectif du radiateur doit être égal à Eteint.

La figure 14.7 illustre les corrections apportées aux règles du service en Alloy.

```
//Temperature control service behavioral description : valid behavior
//Rule 1 : if the inside temperature is less than or equal to 17, then turn on the heater
fact {
   all x: TemperatureControlService, y:TemperatureSensor, z:Heater |
   (y in x.insideTemperatureSensor and z in x.heater and y.currentState <= 17)
   implies
   ((On in z.targetState) and !(Off in z.targetState))
}
//Rule 2 : if the inside temperature is greater than 17, then turn off the heater
fact {
   all x: TemperatureControlService, y:TemperatureSensor, z:Heater |
   (y in x.insideTemperatureSensor and z in x.heater and y.currentState > 17)
   implies
   ((Off in z.targetState) and !(On in z.targetState))
}
```

FIGURE 14.7: Règles valides : correction des règles service de régulation de température en Alloy.

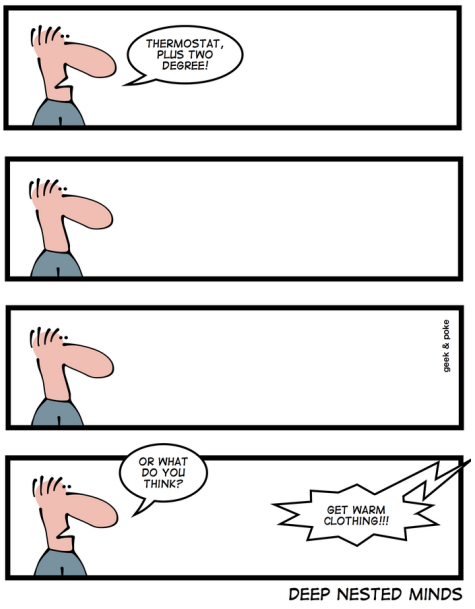
Le résultat de validation du service est montré par la figure 14.8. Le solveur ne trouve pas de contre-exemple qui invalide l'assertion. Celle-ci est alors probablement vraie. Le solveur ne peut affirmer qu'elle est vraie car, même s'il teste un très grand nombre d'instances, il ne peut pas tester tout l'espace de tous les états possibles pour les objets.

```
Executing "Check HeaterTargetState for 3 but 1 TemperatureControlService, 1 Heater, 1 Window"
  Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
  1037 vars. 72 primary vars. 2385 clauses. 15ms.
  No counterexample found. Assertion may be valid. 16ms.
```

FIGURE 14.8: Résultat de validation du service de régulation de température corrigé.

Nous avons décrit dans ce chapitre, une manière de mettre en œuvre la validation des services en utilisant le langage Alloy et l'outil Alloy Analyzer. Nous avons commencé par présenter la logique sur laquelle s'appuie Alloy. Nous avons ensuite décrit la démarche que nous avons menée pour traduire les modèles UML vers le langage Alloy. Ensuite nous avons présenté des exemples de services et de règles de validations implémentés sous Alloy et les résultats de ces validations.

15 Simulation de services et d'orchestrateur de services



http://geek-and-poke.com, January, 31st, 2014

SOMMAIRE

Nous décrivons dans ce chapitre un travail de mise en œuvre de notre proposition dans le but d'effectuer des simulations sur des cas d'usages. Ce travail a été réalisé en collaboration avec Adel Ouhabi Ouhabi, lors de son stage au sein du Labo SEIDO. Nous présentons ici les choix technologiques et architecturaux qui ont été faits lors de cette implémentation ainsi qu'une synthèse des résultats qui ont été obtenus. Les détails d'implémentation sont décrits dans le rapport de stage intitulé « Description et formalisation de règles pour le Smart Building » [86].

15.1 Choix architecturaux

Nous décrivons dans cette section les choix architecturaux qui ont été faits lors de la mise en œuvre de notre solution.

La solution met en œuvre des services de l'IdO qui partagent des objets connectés. Le partage d'objets peut causer des interactions conflictuelles. Il a alors été décidé de mettre en place un orchestrateur pour gérer les conflits entre les services.

15.1.1 La partie de la contribution qui est mise en œuvre

L'implémentation illustre un processus d'orchestration d'un bouquet de services. Les services qui sont implémentés sont des services qui sont déployés, mis en correspondance avec les objets connectés et regroupés dans un bouquet de services. Un orchestrateur est mis en place pour orchestrer les services du bouquet en utilisant un processus d'orchestration.

15.1.2 Les cas d'usage

Cinq services et un orchestrateur de services ont été implémentés pour effectuer les simulations. Les cinq services sont : un service de contrôle de luminosité, un service de régulation de température, un service de contrôle d'énergie, un service de qualité de l'air et un service de sécurité. Tous ces services opèrent dans un contexte d'exécution commun. Ils partagent deux à deux des objets en écriture. La figure 15.1 représente les cinq services que nous avons implémentés et les objets qu'ils utilisent. Chaque service est mis en correspondance avec les objets qu'il utilise. Les cinq services font partie d'un même bouquet de services qui est géré par un orchestrateur.

Chaque service possède un objectif à atteindre. Par exemple, le service de contrôle de luminosité a pour objectif de gérer la luminosité dans son contexte d'exécution. Il s'appuie sur deux capteurs (un capteur de luminosité intérieure, un capteur de luminosité

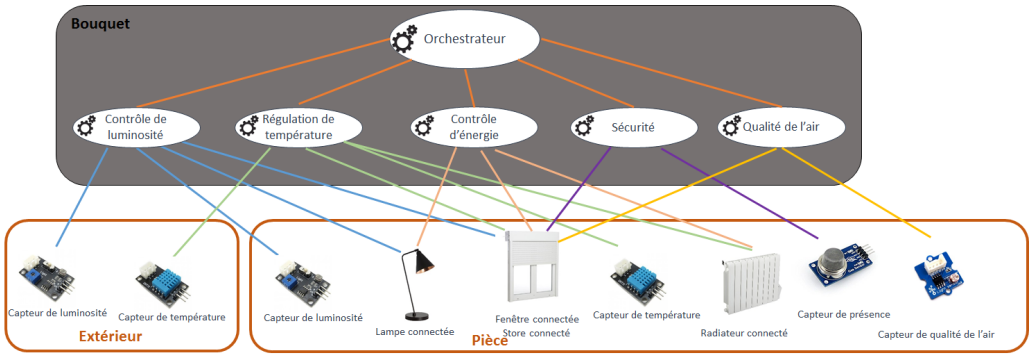


FIGURE 15.1: Cas d'usage de la simulation.

extérieure) et deux captionneurs (un store de fenêtre et une lampe). Si le service de contrôle de luminosité détecte qu'il fait sombre dans son contexte d'exécution, il a trois moyens qui peuvent lui permettre d'atteindre son objectif : lever le store s'il y a assez de luminosité à l'extérieur et que le store est baissé, allumer la lampe si elle est éteinte ou lancer les deux actions.

Lorsque nous implémentions nos cas d'usage, nous avons relevé une particularité importante des services de l'IdO. Chaque service peut avoir plusieurs manières d'atteindre son objectif. Par exemple, pour augmenter la luminosité, le service de contrôle de luminosité peut soit ouvrir le store soit allumer la lampe soit faire les deux. Nous avons exploité cette particularité afin de rendre les services plus flexibles. Par exemple, si le store est verrouillé par le service de sécurité et qu'il fait sombre dans la pièce, le service de contrôle de luminosité utilisera plutôt la lampe.

Ce procédé permet à un service d'avoir plusieurs moyens d'atteindre son objectif et d'en choisir un selon le contexte dans lequel il se trouve. Ainsi, les services les moins prioritaires peuvent continuer à fonctionner même dans le cas où un des objets dont ils ont besoin est verrouillé par un service plus prioritaire.

15.1.3 Stratégie d'orchestration

Le processus d'orchestration des services du bouquet s'appuie sur les priorités entre services. Ces priorités sont définies par l'utilisateur. Lorsqu'il y a un conflit d'accès de deux services sur un objet en écriture (de type actionneur ou captionneur), l'orchestrateur décide de l'état objectif de l'objet en fonction des prirorités des services qui sont en conflit. L'orchestrateur pose alors un verrou sur l'objet du conflit. Ce verrou garde l'objet dans l'état souhaité par le service le plus prioritaire des services qui sont en conflit.

15.2 Choix technologiques

Nous présentons dans cette section les choix technologiques qui ont été faits lors de la mise en œuvre.

15.2.1 Services et objets connectés

Le modèle que nous avons décrit dans le chapitre 9 considère les services et les objets d'un point de vue logique. Les choix technologiques que nous avons fait pour implémenter ces objets et services sont en accord avec cette vision. Notre choix s'est porté sur les technologies du Web sémantique pour représenter les modèles de services et d'objets connectés.

Les modèles structurels des services ont été représentés sous forme d'ontologies en utilisant le langage OWL (Web Ontology Language) [87]. Les modèles comportementaux des services ont été décrits sous forme de règles en utilisant le langage SWRL (Semantic Web Rule Language) [88].

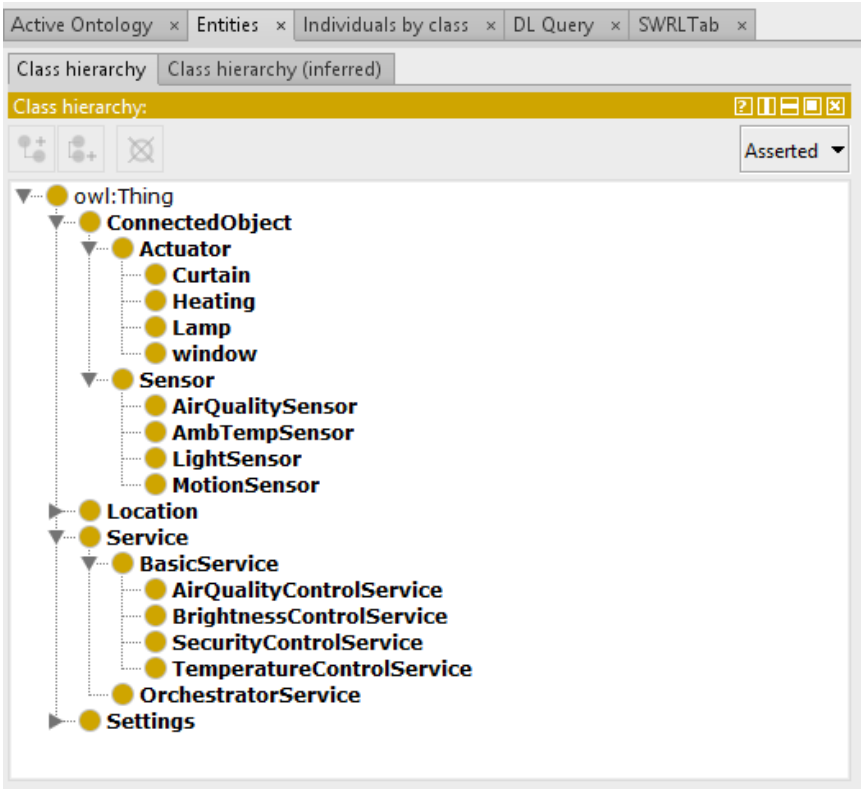


FIGURE 15.2: Extrait de l'ontologie de simulation.

La figure 15.2 illustre un extrait de l'ontologie que nous avons développée pour la mise en œuvre. Cet extrait est visualisé avec le logiciel Protégé [89]. Il représente les classes d'objets connectés, de services, de localisations et de paramètres. Les objets connectés décrivent les capteurs et les actionneurs. Les services décrivent les différents services qui ont été définis pour la simulation. Les localisations décrivent les lieux dans lesquels les objets et les services peuvent être déployés. Les paramètres représentent des données de réglage.

En plus des classes, l'ontologie définit des propriétés d'objets pour décrire les relations entre classes et des propriétés de données pour décrire les attributs de classes.

Nous avons utilisé le langage SPARQL (SPARQL Protocol and RDF Query Language) [90] pour requêter les résultats d'inférence des ontologies OWL.

Notre contexte industriel de travail a influencé notre choix pour représenter les modèles des objets connectés et des types d'objets. En effet, notre contexte géographique de travail s'inscrit dans le cadre d'un bâtiment connecté d'EDF R&D. Lors de la mise en place de ce bâtiment, plusieurs objets et services ont été déployés. Nous avons intégré ces objets et ces services et nous les avons représentés sous forme d'une ontologie.

15.2.2 Environnement de simulation

Nous avons mis en œuvre cette solution en utilisant l'environnement de simulation IdO Freedomotic [60]. Cet envitronnement permet d'avoir une vision graphique du contexte d'exécution et des objets connectés qui s'y trouvent. Ainsi tous les objets qui se trouvent dans ce contexte d'exécution sont visibles et manipulables via l'interface graphique.

La figure 15.3 représente l'environnement de simulation graphique Freedomotic sur lequel nous avons déployé quelques objets (deux capteurs de luminosité dont un intérieur et un extérieur, un capteur de présence, une lampe et un store). Ces objets et leurs états sont graphiquement représentés. Un utilisateur est également présent dans le contexte d'exécution. Le capteur de présence détecte la présence de l'utilisateur dans son contexte d'exécution.

15.2.3 Orchestration

Le processus d'orchestration de services et de résolution de conflits a été implémenté en utilisant le langage Java sous l'environnement de développement Eclipse.

Nous avons implémenté des procédures qui permettent de mettre en relation les services et l'orchestrateur de services.

FIGURE 15.3: Environnement de simulation Freedomotic.

Nous avons également implémenté un projet Web qui permet à l'utilisateur de définir les priorités des services. Il permet également de fournir des données de simulation pour les capteurs.

Nous expliquons dans la section suivante la méthodologie de développement et de simulation que nous avons adoptée.

15.3 Méthodologie de développement et de simulation

En utilisant les outils qui ont été décrits en section précédente, la mise en œuvre a été faite de telle manière à ce que l'exécution se déroule en suivant les étapes suivantes :

1. Récupérer les états des captionneurs depuis l'environnement Freedomotic
2. Récupérer les états des capteurs depuis une page web. C'est l'utilisateur qui entre les données de simulation.
3. Récupérer les priorités des services depuis une page web. Les priorités sont décidées par l'utilisateur.
4. Peupler l'ontologie avec les données récupérées.
5. Inférer l'ontologie avec les règles de services SWRL.
6. Interroger le résultat d'inférence afin de récupérer les nouveaux états des objets à l'aide de requêtes SPARQL.

7. Lancer les actions adéquates pour modifier les états des objets sur l'environnement de simulation Freedomotic. Les visuels des objets sont également modifiés sur Freedomotic.

Nous présentons dans la section suivante une synthèse des résultats que nous avons obtenus des simulations.

15.4 Résultats et discussion

Nous avons présenté dans ce chapitre, la mise en œuvre et les simulations que nous avons réalisés. nous avons effectué les simulations dans le contexte d'un bouquet de services qui est composé de cinq services qui sont orchestrés par un orchestrateur.

Les services partagent des objets en lecture et en écriture. L'orchestrateur met en place une stratégie de gestion de conflits entre services en s'appuyant sur des priorités. Ces priorités sont définies par l'utilisateur via une page web.

Nous avons utilisé un environnement graphique de simulation (Freedomotic) afin de représenter visuellement les objets connectés et les utilisateurs dans le contexte d'exécution.

Les simulations que nous avons menées nous ont permis d'illustrer notre approche d'orhestration de services dans le cadre d'un bouquet de services.

De plus, ces simulations ont révélé des points importants qui sont relatifs aux services de l'IdO. Pour un service donné, nous avons mis en évidence le fait qu'il peut y avoir plusieurs manières d'atteindre l'objectif. Ces différentes manières de faire peuvent concerner des objets différents. Par exemple, le service de contrôle de luminosité peut augmenter la luminosité d'une pièce en ouvrant le store ou en allumant la lampe. Ainsi, si la lampe est verrouillée par un autre service, il peut décider d'ouvrir le store afin d'atteindre son objectif.

Ainsi, lorsqu'un service ne peut pas modifier l'état d'un objet comme il le souhaite, il n'est pas bloqué et il peut atteindre son objectif en utilisant d'autres objets.

16 Conclusion

Nous avons présenté dans cette partie trois mises en œuvre des contributions proposées par cette thèse.

La première mise en œuvre illustre le projet d'un utilisateur qui souhaite équiper une pièce de sa maison de technologies de l'IdO. Les descriptions génériques d'un service de régulation de température ont été définies en se basant sur des rôles d'objets. Une liste d'objets pouvant jouer ces rôles a été présentée. Le processus de déploiement et de mise en correspondance des objets et du service ont été présentés en illustrant à chaque étape la version des descriptions structurelles et comportementales. Au final, nous avons présenté les versions déployées et mises en correspondance du service de régulation de température et des objets connectés.

Cette mise en œuvre nous a permis d'illustrer le scénario de mise en place de services et d'objets connectés qui a été présenté en section 7.1. Cet exemple montre la faisabilité de notre approche qui propose de décrire des services génériques indépendamment des objets connectés, puis de les déployer dans un contexte d'exécution et enfin de les mettre en correspondance en adaptant les descriptions structurelle et comportementale du service en fonction des objets qui sont effectivement déployés.

La deuxième mise en œuvre illustre la vérification de validité de services avec le langage et l'outil Alloy. Cette vérification a été effectuée sur un service de régulation de température non valide. La non validité du service a été détectée par Alloy puis, une version corrigée a de nouveau fait l'objet de vérification et validation.

Cette mise en œuvre nous a permis de vérifier la validité d'un service en phase amont grâce à des simulations de contexte d'exécution. Cette vérification peut se faire lors de simulation pour s'assurer que le service est bien formé. Si des incohérences sont détectées, elles peuvent être corrigées avant l'installation du service et sa mise en marche.

La troisième mise en œuvre illustre une implémentation du méta-modèle sémantique sur lequel s'appuie notre contribution sous forme d'une ontologie OWL et de règles SWRL.

Cette mise en œuvre permet d'effectuer une simulation de la gestion des conflits d'accès aux objets partagés dans le cas d'un bouquet de services et l'intérêt d'un algorithme d'orchestration fondé sur les priorités ainsi que l'optimisation de l'atteinte des objectifs de chaque service du bouquet.

Ces trois mises en œuvre ont permis de vérifier et d'illustrer l'applicabilité de notre approche sur des exemples de cas d'usage autour de la maison connectée.

Quatrième partie

Conclusion générale

17 Introduction

La dernière partie du manuscrit permet de faire le bilan et de synthétiser les travaux de recherche que nous avons menés durant cette thèse et les résultats que nous avons obtenus.

Nous commençons d'abord par synthétiser les différentes problématiques qui ont été adressées ainsi que les solutions qui ont été proposées pour répondre à chacune d'entre elles.

Nous discutons ensuite des limites de notre contribution de thèse et des ouvertures scientifiques et industrielles qui sont soulevées par notre thèse. Nous donnons des orientations pour les futurs travaux qui s'appuieront sur les résultats des recherches de cette thèse.

18 Problématiques adressées

"One day, in retrospect, the years of struggle will strike you as the most beautiful."
Sigmund Freud.

SOMMAIRE

Ce chapitre synthétise les problématiques qui ont été adressées dans cette thèse. Nous présentons chaque problématique dans une section en précisant la solution qui a été proposée pour y répondre.

18.1 Ecosystème ouvert des services et objets connectés

Pour que les services ne soient pas fortement couplés aux objets qu'ils gèrent, nous proposons, dans cette thèse, de mettre en place des services génériques qui décrivent des rôles d'objet. Chaque rôle décrit un ensemble de types d'objets qui peut jouer ce rôle. Le service décrit aussi, pour chaque type d'objet les types de données qu'il peut prendre en charge.

La description de rôles permet au service de ne pas dépendre d'un objet en particulier. Il décrit des types d'objets. Par exemple, un service de régulation de température qui est fortement couplé aux objets connectés, décrit le capteur de température qu'il peut prendre en charge avec les caractéristiques propres à ce capteur. Le service ne peut alors fonctionner avec un autre capteur de température que celui qu'il décrit.

En revanche, un service de régulation de température qui est faiblement couplé avec les objets, décrit plutôt un rôle de thermomètre intérieur qui peut être joué par un objet de type capteur de température.

Cette approche complexifie quelque peu l'écriture des règles de comportement du service générique. Pour être efficaces, les règles de service doivent, pour chaque rôle, balayer une gamme d'objets connectés et s'adapter lors de la phase de mise en correspondance en fonction de l'objet qui jouera effectivement le rôle décrit.

En revanche, cette approche préconise des services qui ne sont pas dépendants d'un objet particulier mais qui adaptent leur comportement en fonction des objets que possède l'utilisateur.

18.2 Partage d'objets par plusieurs services

Nous avons proposé dans la section 9 un modèle qui permet de casser les silos de l'IdO. Ainsi, un même objet peut être utilisé par plusieurs services. L'avantage de cette approche est que les services peuvent utiliser les mêmes objets quels que soient leurs founisseurs.

Le partage d'objets par plusieurs services a été illustré dans le chapitre 15. Ainsi, un même objet peut être utilisé par plusieurs services. Le partage d'objets permet pour une même fonctionnalité d'avoir un seul objet, même si cette fonctionnalité est amenée à être utilisée par des services qui proviennent de fournisseurs différents. Le partage d'objets

permet donc d'éviter la duplication d'objets qui ont la même fonctionnalité. Par exemple, si un service de sécurité et un service d'allumage automatique sont déployés dans un même contexte d'exécution et que les deux services utilisent un capteur de présence alors les deux services peuvent partager le même objet capteur de présence. Ce partage n'est alors pas conditionné par les fournisseurs chez lesquels les objets et les services ont été acquéris.

Dans notre démarche, nous distinguons entre les services qui partagent des objets en lecture (c'est-à-dire des capteurs) et les services qui partagent des objets en écriture (c'est-à-dire des actionneurs ou de captionneurs). Le partage d'objets en lecture peut se faire sans causer de conflits d'accès entre les services sur les objets partagés, ce sont des interactions harmonieuses. En revanche, lorsque plusieurs services partagent un objet en écriture, alors il peut y avoir un conflit d'accès de ces services aux objets. Nous appelons les interactions qui peuvent causer des conflits entre les services, des interactions conflictuelles. En plus du partage d'objets par plusieurs services, nous adressons dans la thèse, la problématique des interactions conflictuelles lorsque des services partagent des objets en écriture. Les propositions qui sont liées à la gestion de conflits entre services sont décrites dans la section suivante.

18.3 Gestion des conflits

Cette thèse adresse la problématique de conflits d'accès de plusieurs services à un objets connecté. Dans ce cas, les services peuvent vouloir changer l'état d'un objet de manière contradictoire. Nous proposons de mettre en place des services « évolués » qui, lorsqu'ils souhaitent modifier l'état d'un objet, commencent par proposer un ou plusieurs états objectifs candidats à une négociation ; puis reçoivent l'état objectif qui a été décidé suite à la négociation et adaptent leurs comportements en fonction de cet état décidé.

La phase de négociation et de prise de décision se fait au niveau d'un orchestrateur de services. Le processus de négociation et de prise de décision a pour objectif de satisfaire le plus grand nombre de services prioritaires. Dans le meilleur des cas, le processus de sélection d'état objectif aboutit à un état qui satisfait tous les services qui sont en conflit et dans le pire des cas, ce processus aboutit à un état qui satisfait uniquement le service le plus prioritaire des services conflictuels.

Plus le nombre d'états objectifs candidats proposé par chaque service est grand plus la probabilité de satisfaire un grand nombre de services conflictuels est grande. Les services ont alors intérêt à adopter une approche comportementale flexible et proposer un grand nombre d'états objectifs candidats pour avoir de grandes chances de voir un de ces états objectifs candidats sélectionné.

18.4 Gestion des oscillations

Nous désignons par oscillations les changements d'états successifs d'un même objet et qui sont provoqués dans le même contexte.

Ces oscillations surviennent lorsque, dans un contexte donné, plusieurs règles se déclenchent les unes les autres et que ces règles agissent de plusieurs manières sur un même objet. Ces règles peuvent décrire le comportement d'un même service ou de plusieurs services d'un cluster de services.

Nous proposons de traiter cette problématique en deux phases. D'abord en phase de conception, en vérifiant que les règles d'un même service ne provoquent pas d'oscillations dans un contexte donné. Puis en phase d'exécution, en vérifiant que plusieurs règles de différents services ne provoquent pas d'oscillation dans le contexte courant.

18.4.1 En phase de conception

La détection d'oscillations en phase de conception consiste à vérifier la validité de service. Un service valide est un service dont les règles, quel que soit leur contexte d'exécution, ne provoquent pas d'oscillations sur un objet.

La vérification se fait en détectant les règles qui se déclenchent mutuellement. Un cycle est détecté lorsque plusieurs règles se déclenchent les unes les autres à partir d'un même contexte d'exécution. Si un cycle est détecté, l'algorithme de validation vérifie qu'un même objet n'apparait pas avec deux états différents. Si tel est le cas alors le service n'est pas valide.

Nous avons montré en section 14 un exemple de service non valide où plusieurs règles d'un même service se déclenchent et agissent de manière différente sur un même objet.

L'avantage de détecter les oscillations en phase de conception est de pouvoir corriger les erreurs de conception avant de mettre en oeuvre les services.

Aussi, la complexité de l'algorithme de détection d'oscillations n'a pas d'impact sur la bonne exécution des services puisqu'il est lancé en phase amont, avant le déploiement des services.

18.4.2 En phase d'exécution

De la même manière qu'en phase de conception, il est possible de vérifier la validité d'un cluster de services dans la mesure où, dans un contexte donné, des règles de plusieurs services peuvent se déclencher les unes les autres et provoquer des oscillations d'états sur un objet.

Il est alors nécessaire de vérifier sa validité à chaque fois qu'un cluster de services est créé ou qu'un service est ajouté au cluster.

La complexité de l'algorithme de validation a un impact minime sur le fonctionnement du cluster parce que cette vérification n'est pas effectué à chaque changement d'état. Elle est effectuée une fois à chaque changement de configuration du cluster grâce au solver SAT qui permet de générer et tester un grand nombre de contextes d'exécution.

18.5 Adaptabilité aux utilisateurs

Les services que nous développons dans cette thèse doivent prendre en compte les souhaits des utilisateurs. Nous classons les voeux exprimés par les utilisateurs en deux catégories : les voeux explicites et les voeux implicites.

18.5.1 Voeux explicites : définition des rôles de services

Les voeux explicites s'expriment lorsque l'utilisateur définit les rôles de services. Ces rôles représentent la configuration sur laquelle s'appuie l'orchestrateur de services pour prendre les décisions. Ainsi, c'est l'utilisateur qui définit quels sont les services qui lui sont importants et ceux qui le sont moins. La configuration est définie par l'utilisateur lors de la mise en place des services et peut être modifiée par la suite.

18.5.2 Voeux implicites : Prise en compte des actions des utilisateurs

Les voeux implicites de l'utilisateur s'expriment par les actions qu'il fait sur les objets connectés. Dans le but d'améliorer l'acceptabilité des services, nous proposons dans cette thèse, que les services tiennent compte des actions des utilisateurs.

Ainsi, chaque changement d'état d'actionneur est en relation avec l'entité qui est responsable de ce changement d'état. Chaque service peut alors réagir de manière différente en fonction de l'entité qui est responsable de l'état de chaque actionneur.

Il est possible de favoriser l'action des utilisateurs, en veillant à conserver les états des objets qu'il a modifiés.

La thèse propose une manière de considérer les actions des utilisateurs sur les objets mais n'impose pas de réaction type aux actions des utilisateurs. Chaque service est libre de favoriser ou non les actions des utilisateurs par rapport à son propre fonctionnement. Ainsi, un service dont la criticité est très élevée, comme par exemple un service de gestion d'incendie, peut décider de se favoriser par rapport aux actions des utilisateurs.

18.6 Approche orientée par les objectifs

Notre travail de réflexion et d'analyse des problématiques posées qui est décrit dans le chapitre 7 a fait émerger le besoin de considérer les services du point de vue de leurs objectifs. L'approche que nous décrivons dans le chapitre 9 est une approche sémantique orientée vers les objectifs. Les services que nous considérons au niveau sémantique décrivent des objectifs à atteindre. Nous avons adopté une approche logique pour représenter ces objectifs. Cette approche fait abstraction des aspects opérationnels des services en se concentrant sur la manière d'atteindre les objectifs de services.

Les travaux de simulations que nous avons présentés dans le chapitre 15 ont permis de metrre en oeuvre des services qui sont guidés par des objectifs. De plus, ces services implémentent plusieurs manières possibles d'atteindre leurs objectifs en utilisant différents objets connecéts. Ainsi, un service peut choisir d'atteindre son objectif via un moyen ou un autre en fonction des autres services qui agissent sur les objets connectés.

19 Perspectives

"L'achèvement d'une œuvre complexe doit non dissimuler
son inachèvement, mais le révéler"
Éduquer pour l'ère planétaire - Edgar Morin

SOMMAIRE

Nous discutons dans ce chapitre des ouvertures scientifiques et industrielles qui sont soulevées par notre thèse. Nous donnons des orientations possibles pour les futurs travaux qui s'appuieront sur les résultats des recherches de notre thèse.

19.1 Vers un modèle opérationnel pour les services de l'IdO

Le niveau sémantique de notre approche considère les problématiques posées d'un point de vue logique en faisant abstraction des caractéristiques opérationnelles du système. Le but du niveau Artefacts est de préciser les aspects opérationnels du système dans le but de compléter le niveau sémantique.

Nous présentons dans ce chapitre, quelques exemples de travaux qui traitent des aspects opérationnels qui complètent l'approche sémantique décrite dans ce manuscrit.

19.1.1 Aperçu des caractéristiques du niveau Artefacts

Approche procédurale : raisonnement en termes d'actions

Le niveau d'artefacts s'appuie sur une approche procédurale. Les objectifs que doit atteindre le système ont été décrits par le niveau sémantique de l'architecture. Le niveau artefacts adopte une approche procédurale. A partir des objectifs qui ont été définis au niveau sémantique et de l'état actuel du système, il met en place une procédure à suivre pour atteindre les objectifs.

Supposons que l'état objectif de l'objet « fenêtre » qui a été décidé au niveau sémantique est « Fenêtre ouverte », dans ce cas, le niveau artefacts va décider des actions à effectuer pour atteindre l'état objectif à partir de l'état actuel de l'objet fenêtre.

Il peut y avoir plusieurs choix de procédures à suivre pour atteindre un objectif à partir de l'état actuel. Dans ce cas, le niveau d'artefacts définira des critères de performance sur lesquels l'algorithme de choix de procédure va s'appuyer.

Gestion des échanges

En plus de définir les procédures à suivre pour atteindre les objectifs qui ont été fixés au niveau sémantique, le niveau d'artefacts définit ces procédures en spécifiant les différents échanges qui doivent avoir lieu pour chaque étape de la procédure.

Les choix des échanges entre services et entre services et objets connectés sont précisés au niveau artefacts. Les échanges synchrones ou asynchrones sont précisés, la fréquence de ces échanges, les types de messages envoyés, les évènements, la fréquence d'envoi des

évènements et la synchronisation sont autant d'éléments qu'il faudra préciser au niveau des artefacts.

Système de règles

Le comportement de services a été décrit en termes de règles de service. Cependant, les choix opérationnels relatifs aux systèmes de règles n'ont pas été effectués au niveau sémantique.

Un des points qui sont à préciser au niveau artefacts est le mode opératoire du système de règles.

Le mode opératoire désigne la manière avec laquelle le système de règles est évalué. Deux modes opératoires se distinguent : le mode synchrone et le mode asynchrone.

Mode synchrone En mode synchrone, le système de règles est évalué à chaque tic de l'horloge. Lors de chaque tic d'horloge, le système est évalué en fonction de son état actuel. Plusieurs objets peuvent changer d'état entre deux évaluations de règles. Dans ce cas, le système ne réagit pas à un changement d'état mais évalue, à des instants précis, le système de règles.

Ce mode opératoire limite les oscillations qui résultent de règles qui sont déclenchées suite à des changements d'états d'objets connectés. L'inconvénient du mode synchrone est la difficulté de déterminer la valeur du tic d'horloge pour lequel il faut examiner le système de règles.

Mode asynchrone Le mode opératoire asynchrone consiste à évaluer le système de règles à l'arrivée d'un évènement. Ces évènements peuvent être relatifs à des changements d'états des objets connectés.

Lorsque le système de règles suit un mode opératoire asynchrone, il réagit à chaque changement d'état d'un objet connecté, c'est à dire, à chaque évaluation, un seul état d'objet aura changé depuis l'évaluation précédente.

L'avantage du mode asynchrone est que le système réagit directement après le changement d'état d'un objet. L'inconvénient peut résider dans le passage à l'échelle où il peut y avoir un grand nombre d'objets et de changement d'états.

Chacun des modes synchrone et asynchrone présente des avantages et des inconvénients. Le choix du mode opératoire se fait après l'étude des aspects opérationnels des services de l'IdO.

19.1.2 Exemple d'approches compatibles avec le niveau Artefacts

Nous présentons, dans cette section deux approches qui peuvent être compatibles avec les besoins du niveau artefacts de notre approche. La première s'appuie sur les Business Artefacts d'IBM et la deuxième s'appuie sur une démarche autonomique. Nous présentons ces deux approches de manière intuitive. D'autres approches qui abordent les mêmes problématiques sous des angles de vue différents sont tout à fait envisageables. Une étude poussée des problématiques liée à ce niveau d'architecture doit être effectuée avant de prendre position sur le choix d'une approche adaptée.

Approche Business Artefacts

Les Business Artefacts sont une approche émergente proposée par IBM durant la fin des années 2000 pour la modélisation de processus métier dans une entreprise [70] [71].

Les approches traditionnelles de modélisation de processus et flux métier sont centrées sur les flux d'activité qui sont couplés avec des données souvent pensées après coup ; ou elles sont basées sur des documents avec un traitement souvent pensé après coup. La plupart des frameworks BPM (Business Process Management) [72] [73] utilisent des méta-modèles centrés sur les flux d'activités, et les données manipulées par ces processus sont considérées avec une moindre importance. D'autres approches définies par Glushko et al. [74] sont centrées sur les documents, avec un méta-modèle de processus généralement plus pauvre [71].

L'approche des Business Artefacts vise à combiner les aspects de données et de processus dans un même bloc de construction de base, ce bloc est appelé « Artefact ». Seules les entités métier clés pour le processus métier sont modélisées sous forme d'artefacts. Ces entités évoluent au fur et à mesure de l'exécution des opérations métier.

La figure 19.1 représente la structure d'un Artefact. Ainsi, l'artefact est formé de deux parties, un modèle d'information qui regroupe les données importantes pour le métier et un modèle de cycle de vie qui décrit l'évolution de ces données au fur et à mesure de l'exécution du processus métier.

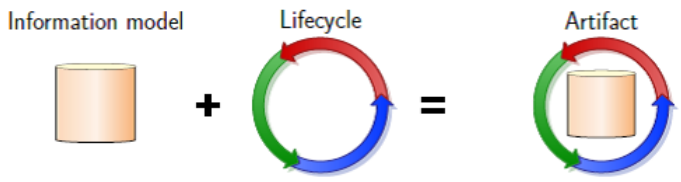


FIGURE 19.1: Business Artefact

Le modèle de cycle de vie des artefacts peut être décrit en GSM (Guard-Stage-Milestone) ou en FSM (Finite State Machine).

Les artefacts définissent les aspects opérationnels du système, tel que les échanges de messages ou les évènements. La figure 19.2 représente un système à base de trois artefacts. Les échanges entre artefacts et avec l'environnement se font via des évènements ou des messages. De plus, chaque utilisateur peut avoir une vue restreinte du système en fonction d'un rôle qui lui est assigné.

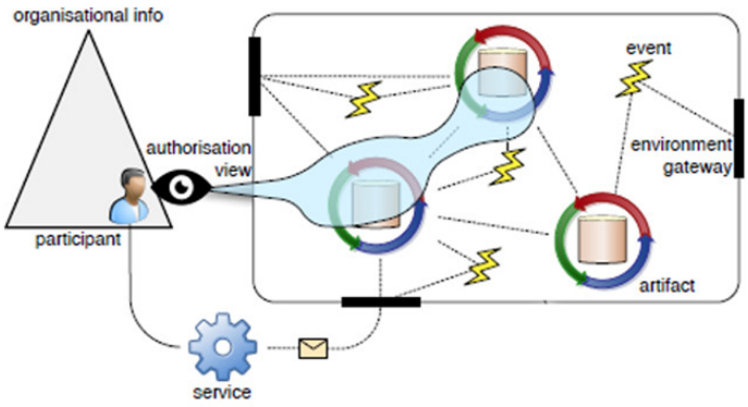


FIGURE 19.2: Système à base de Business Artefacts

Approche autonomique

Le terme « autonomique » a été inventé par IBM en 2001 et présenté par son vice-président de la recherche Paul Horn [91]. Le terme autonomique fait référence au système nerveux autonome qui régit des fonctions vitales telles que le rythme cardiaque ou la température corporelle. Ainsi, le cerveau conscient est libéré de charge de contrôler activement ces fonctions. Par analogie, les systèmes informatiques autonomiques se caractérisent par leur capacités d'autogestion, à savoir [92] :

— Auto-configuration : le système se configure automatiquement selon des politiques de haut niveau qui spécifient ce que le système doit réaliser. Il s'adapte à son contexte de manière proactive, sans intervention extérieure ;

— Auto-optimisation : le système cherche continuellement des moyens d'améliorer son fonctionnement et de mieux accomplir les tâches pour lesquelles il a été conçu ;

— Auto-réparation : le système détecte, diagnostique et répare ses problèmes internes résultant de bugs ou de défaillances logicielles et matérielles.

— Auto-protection : le système anticipe les variations de son environnement et se protège contre les menaces extérieures.

En se basant sur les approches autonomiques, les auteurs de [93] proposent d'aborder les problématiques d'un point de vue des systèmes organiques. Ils modélisent ainsi le système comme des entités holoniques qui sont indépendantes et autonomiques mais qui peuvent aussi faire partie intégrante d'un plus grand système.

19.1.3 Correspondance entre les niveaux sémantique et artefacts

La correspondance entre les niveaux sémantique et artefacts sont des techniques qui permettent de faire le lien entre l'approche déclarative du niveau sémantique et l'approche procédurale du niveau artefacts. Ces techniques permettent de décrire la manière de transformer les objectifs et les entités qui ont été définis au niveau sémantique vers des entités du niveau artefacts.

Le processus de correspondance, une fois défini, prend en entrée un modèle sémantique et génère un modèle d'artefacts en fonction d'un certain nombre de critères.

Ce processus est à définir lorsque les choix technologiques qui sont en relation au niveau d'artefacts auront été effectués. Ce processus spécifie par exemple comment transformer les états objectifs en un ensemble d'actions à exécuter.

19.2 Augmenter l'acceptabilité des services par les utilisateurs

Les services de l'IdO qui font l'objet de cette thèse sont des services qui sont destinés à être intégrés dans l'environnement des utilisateurs. Il est alors important que ces services soient acceptés par l'utilisateur. Dans le but d'augmenter l'acceptabilité des services par les utilisateurs, nous avons proposé deux moyens qui permettent de tenir compte des souhaits des utilisateurs : la définition des rôles de services et la prise en compte des actions des utilisateurs sur les objets connectés. Ainsi, c'est l'utilisateur qui définit les rôles et les priorités des services en fonction de ses besoins. De plus, nous fournissons aux services proposés dans cette thèse un moyen de connaître le resposable de l'état d'un objet. Ils peuvent alors adapter leur comportement en fonction des actions des utilisateurs sur les objets connectés.

Les moyens d'augmenter l'acceptabilité des services que nous avons proposés dans la thèse peuvent être étendus et complétés. Nous proposons des travaux futurs dans le but de renforcer l'acceptabilité des services par les utilisateurs.

19.2.1 Affiner la prise en compte des actions des utilisateurs

Notre approche prend en compte les actions des utilisateurs sur les objets connectés mais ne différencie pas entre les différents utilisateurs. Une perspective de travaux consiste à différencier entre les utilisateurs et à réagir de manière différente en fonction de l'utilisateur qui interagit avec le système. Par exemple, le service réagit de manière différente si c'est un membre de la famille qui actionne un objet ou si c'est une personne étrangère. Le service peut aussi réagir de manière différente en fonction de l'âge de l'utilisateur qui agit sur les objets.

Une première idée est de définir des rôles d'utilisateurs qui permettent de distinguer entre les différents utilisateurs qui interagissent avec le système. Ces rôles permettront aux services d'adapter leur comportement en fonction du rôle de l'utilisateur qui agit sur les objets.

Une deuxième idée s'inspire des approches centrées sur les utilisateurs [61]. Ces approches proposent de mettre l'utilisateur au coeur du système depuis les phases de conception d'un système. Le but est de concevoir des systèmes non-intrusifs qui sont mieux acceptés par l'utilisateur. Ces approches ne sont pas spécifiques à l'IdO. Un travail d'étude d'analyse et d'intégration doit être fait dans le but d'adapter ces approches au domaine de l'IdO.

A terme, le but est que la technologie se fonde complètement dans l'environnement de l'utilisateur et devienne invisible pour lui.

19.2.2 Ergonomie de services

Nous nous sommes concentrés dans la thèse sur les aspects fonctionnels des services. Cependant, un large volet de recherche autour des interactions des services avec les utilisateurs peut compléter et enrichir notre approche. Un travail qui permet de définir les interactions des services avec les utilisateurs de manière ergonomique est nécessaire pour rendre ces services acceptables par les utilisateurs.

Il est important que ces services « informatiques » soient accessibles pour des utilisateurs non experts. Des approches qui proposent de modéliser des règles métier dans un langage naturel peuvent apporter une pierre à l'édifice de services accessibles et acceptables par les utilisateurs. Nous avons proposé dans un travail antérieur de traduire des règles métier écrites en langage informatique vers des règles écrites en langage naturel [94]. Ce travail a été mis en oeuvre en s'inspirant du standard SBVR [95] défini par l'OMG. Ce travail ne concerne pas les services de l'IdO en particulier mais il est envisageable de s'en inspirer pour augmenter l'accessibilité des services aux utilisateurs.

19.2.3 Retours informatifs des services vers les utilisateurs

Dans un monde où les services et les utilisateurs cohabitent dans un même environnement et que les services agissent de manière automatique sur les objets, il peut arriver que les services prennent des décisions que les utilisateurs ne comprennent pas. Un service de qualité de l'air peut par exemple décider d'ouvrir une fenêtre alors qu'il fait froid car il a détecté que l'air intérieur est très pollué et que c'est un danger pour les utilisateurs. Dans ce cas, cette action peut déclencher une frustration chez l'utilisateur s'il ne comprend pas le motif de la décision qui a été prise.

Dans le but d'augmenter l'acceptabilité des services, nous proposons que ces derniers informent les utilisateurs des raisons pour lesquelles une décision a été prise. Il est alors important que les services puissent informer les utilisateurs des raisons de leurs décisions sans pour autant être intrusifs et solliciter trop souvent les utilisateurs. Un travail de réflexion doit alors être mené afin de délimiter cette frontière et répondre aux deux questions : Quelles sont les décisions qu'il faut justifier auprès des utilisateurs ? Quels moyens est-ce qu'il faut utiliser pour effectuer ces interactions ?

19.3 Application à d'autres domaines

Tout au long de ce manuscrit de thèse, nous avons illustré notre démarche sur des services autour de la maison connectée. Cependant, notre approche se veut indépendante des cas d'usages de la maison connectée. Un travail futur intéressant est de mettre en oeuvre notre approche sur d'autres cas d'usage qui sont issus d'autres domaines d'application. Nous décrivons deux domaines d'application connexes à la maison connectée sur lesquels il est intéressant d'appliquer notre approche afin de l'étendre et de la valider pour ces domaines d'application. L'application à ces domaines permettrait de vérifier, entre autres, la gestion de charge et le passage à l'échelle de notre approche.

19.3.1 L'immeuble intelligent (Smart Building)

Le smart building représente un domaine d'applicabilité pour lequel notre approche peut apporter quelques solutions.

Les problématiques de couplage fort entre les services et les objets de l'IdO se posent aussi lorsqu'il s'agit de services destinés aux immeubles intelligents. Plusieurs offres de services propriétaires destinées à des immeubles intelligents sont disponibles sur le marché. Intel se positionne sur ce marché avec son offre Intel Building Management Platform

[96]. D'autres compagnies proposent aussi des solutions propriétaires pour les immeubles intelligents [97] [98].

Le faible couplage entre les services et les objets connectés que préconisent nos travaux de thèse visent à donner une liberté aux acquéreurs de solutions IdO pour qu'ils puissent acquérir les objets et les services de différents fournisseurs. Cette approche est intéressante dans le cadre d'un immeuble intelligent où différents propriétaires peuvent acquérir différentes solutions IdO mais qu'il faudra composer ces solutions. Il est alors nécessaire d'avoir des services interopérables, ouverts et non propriétaires.

De plus, dans un immeuble intelligent, les orchestrations de services se font à différents niveaux de granularité. Lorsque l'immeuble est composé d'habitations, les services sont orchestrés au niveau de chaque habitation avec des résolutions de problèmes locaux. Ces mêmes services sont également orchestrés de manière globale pour répondre à des besoins généraux de l'immeuble.

Les cas d'usage d'un immeuble intelligent sont différents de ceux d'une maison intelligente. Voici quelques exemples de services :

- Service de gestion de places de stationnement partagées.

- Service de gestion de bornes de recharge de véhicules électriques dans un parking d'immeuble.

- Service de gestion de la consommation énergétique de l'immeuble.

19.3.2 La ville intelligente (Smart City)

Les services qui sont destinés aux villes intelligentes sont différents de ceux qui sont destinés aux maisons ou aux immeubles. Prenons deux exemples de services pour les villes intelligentes :

— Service de gestion dynamique des feux tricolores.

— Service de libération de voie pour les véhicules prioritaires.

Le premier service a pour but de gérer les feux tricolores de manière dynamique. Il favorise les axes les plus fréquentés, sans pénaliser les axes les moins fréquentés. Il peut tenir compte des informations de trafic et réguler en fonction de ces données collectées.

Le deuxième service est un service de libération de voie pour les véhicules prioritaires. En se basant sur l'itinéraire d'un véhicule prioritaire, il s'assure que les feux tricolores sur cet itinéraire soient au vert au moment du passage du véhicule prioritaire.

Ces deux services partagent des objets en écriture. Il peut alors y avoir un conflit entre eux. La mise en place d'un orchestrateur de services permet de gérer les conflits d'accès aux feux tricolores. Aussi, la flexibilité de chaque service permet de conserver un degré de

satisfaction important pour la globalité des services. Par exemple, le service de libération de voie pour un véhicule prioritaire pourrait avoir deux possibilités d'itinéraire. L'orchestrateur décidera, en fonction des souhaits des autres services sur les feux tricolores pour chaque itinéraire, du meilleur chemin pour satisfaire le plus grand nombre de services.

20 Conclusion

La partie IV de ce manuscrit conclut nos travaux de thèse.

Nous commençons tout d'abord en synthétisant les contributions qui ont été apportées par ce travail de recherche.

Ensuite, nous présentons les questions qui ont été ouvertes par nos recherches en décrivant les travaux que nous envisageons à la suite de cette thèse.

Bibliographie

[1] "Seido lab | cybersécurité et internet des objets." http://seido-lab.com/. Accessed on Sep. 30, 2017.

[2] R. Bouali Baghli, E. Najm, and B. Traverson, "Defining services and service orchestrators acting on shared sensors and actuators," in *Model-Driven Engineering and Software Development (MODELSWARD), 2018 6th International Conference on*, 2018.

[3] K. Ashton, "That 'internet of things' thing," *RFiD Journal*, vol. 22, no. 7, 2011.

[4] I. Peña-López *et al.*, "Itu internet report 2005 : the internet of things," 2005.

[5] O. Vermesan, P. Friess, P. Guillemin, S. Gusmeroli, H. Sundmaeker, A. Bassi, I. S. Jubert, M. Mazura, M. Harrison, M. Eisenhauer, *et al.*, "Internet of things strategic research roadmap," *Internet of Things-Global Technological and Societal Trends*, vol. 1, pp. 9–52, 2011.

[6] L. Atzori, A. Iera, and G. Morabito, "The internet of things : A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[7] A. Bassi and G. Horn, "Internet of things in 2020 : A roadmap for the future," *European Commission : Information Society and Media*, vol. 22, pp. 97–114, 2008.

[8] H. Sundmaeker, P. Guillemin, P. Friess, and S. Woelfflé, "Vision and challenges for realising the internet of things," *Cluster of European Research Projects on the Internet of Things, European Commision*, vol. 3, no. 3, pp. 34–36, 2010.

[9] "Web services glossary - w3c working group note 11 february 2004." https://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/. Accessed on Sep. 30, 2017.

[10] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana, *et al.*, "Web services description language (wsdl) 1.1," 2001.

[11] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, and D. Winer, "Simple object access protocol (soap) 1.1," 2000.

[12] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext transfer protocol–http/1.1," tech. rep., 1999.

[13] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau, "Extensible markup language (xml).," *World Wide Web Journal*, vol. 2, no. 4, pp. 27–66, 1997.

[14] D. Guinard, V. Trifa, S. Karnouskos, P. Spiess, and D. Savio, "Interacting with the soa-based internet of things : Discovery, query, selection, and on-demand provisioning of web services," *IEEE transactions on Services Computing*, vol. 3, no. 3, pp. 223–235, 2010.

[15] D. Lizcano, M. Jiménez, J. Soriano, J. M. Cantera, M. Reyes, J. J. Hierro, F. J. Garijo, and N. Tsouroulas, "Leveraging the upcoming internet of services through an open user-service front-end framework.," *ServiceWave*, vol. 8, pp. 147–158, 2008.

[16] L. Da Xu, W. He, and S. Li, "Internet of things in industries : A survey," *IEEE Transactions on industrial informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.

[17] B. Shneiderman, *Designing the user interface : strategies for effective human-computer interaction*. Pearson Education India, 2010.

[18] G. Abowd, A. Dey, P. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a better understanding of context and context-awareness," in *Handheld and ubiquitous computing*, pp. 304–307, Springer, 1999.

[19] L. Barkhuus and A. Dey, "Is context-aware computing taking control away from the user ? three levels of interactivity examined," in *UbiComp*, pp. 149–156, Springer, 2003.

[20] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things : A survey," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 414–454, 2014.

[21] R. Soley *et al.*, "Model driven architecture," *OMG white paper*, vol. 308, no. 308, p. 5, 2000.

[22] A. G. Kleppe, J. Warmer, W. Bast, and M. Explained, "The model driven architecture : practice and promise," 2003.

[23] S. Mellor, K. Scott, A. Uhl, and D. Weise, "Model-driven architecture," *Advances in Object-Oriented Information Systems*, pp. 233–239, 2002.

[24] "Omg | object management group." http://www.omg.org/index.htm. Accessed on Sep. 30, 2017.

[25] A. G. Kleppe, J. B. Warmer, and W. Bast, *MDA explained : the model driven architecture : practice and promise*. Addison-Wesley Professional, 2003.

[26] K. Raymond, “Reference model of open distributed processing (rm-odp) : Introduction,” in *Open distributed processing*, pp. 3–14, Springer, 1995.

[27] “Information technology–Open distributed processing–Reference Model - Foundations,” itu-t recommendation x.902, International Telecommunication Union, 2009.

[28] “Information technology–Open distributed processing–Reference Model - Architecture,” itu-t recommendation x.903, International Telecommunication Union, 2009.

[29] “Information technology - Open Distributed Processing - Reference Model : Architectural Semantics,” itu-t recommendation x.904, International Telecommunication Union, 1997.

[30] P. F. Linington, “Rm-odp : the architecture,” in *Open Distributed Processing*, pp. 15–33, Springer, 1995.

[31] B. Von Halle and L. Goldberg, *The business rule revolution : Running business the right way*. Happy About, 2006.

[32] “the business rules group.” http://www.businessrulesgroup.org/theBRG.htm. Accessed on Sep. 30, 2017.

[33] H. Herbst, G. Knolmayer, T. Myrach, and M. Schlesinger, “The specification of business rules : A comparison of selected methodologies.,” in *Methods and associated tools for the information systems life cycle*, pp. 29–46, 1994.

[34] G. Knolmayer and H. Herbst, “Business rules,” *Wirtschaftsinformatik*, vol. 35, no. 4, pp. 386–390, 1993.

[35] “Smartthings. add a little smartness to your things..” https://www.smartthings.com/. Accessed on Sep. 30, 2017.

[36] “Smartthings. add a little smartness to your things..” https://www.smartthings.com/products. Accessed on Sep. 30, 2017.

[37] “Overview | smartthings developers.” http://developer.smartthings.com/. Accessed on Sep. 30, 2017.

[38] “Homekit maison : Accessoires compatibles accessoires & produits compatibles avec la fonctionnalité apple homekit & siri.” http://homekit.maison/. Accessed on Sep. 30, 2017.

[39] “Healthkit - apple developer.” https://developer.apple.com/healthkit/. Accessed on Sep. 30, 2017.

[40] “Homekit - apple developer.” https://developer.apple.com/homekit/. Accessed on Sep. 30, 2017.

[41] "Apple developer." https://developer.apple.com/. Accessed on Sep. 30, 2017.

[42] "Accueil | nest." https://nest.com/fr/. Accessed on Sep. 30, 2017.

[43] "Made by google." https://madeby.google.com/intl/fr_fr/home/. Accessed on Sep. 30, 2017.

[44] "Google developers." https://developers.google.com/. Accessed on Sep. 30, 2017.

[45] "Onem2m." http://www.onem2m.org/. Accessed on Sep. 30, 2017.

[46] R. Fielding, "Fielding dissertation : Chapter 5 : Representational state transfer (rest)," *Recuperado el*, vol. 8, 2000.

[47] J. Swetina, G. Lu, P. Jacobs, F. Ennesser, and J. Song, "Toward a standardized common m2m service layer platform : Introduction to onem2m," *Wireless Communications, IEEE*, vol. 21, no. 3, pp. 20–26, 2014.

[48] "Om2m/one - eclipsepedia." https://wiki.eclipse.org/OM2M/one. Accessed on Sep. 30, 2017.

[49] "Allseen alliance." https://allseenalliance.org/. Accessed on Sep. 30, 2017.

[50] "Alljoyn™ system description | allseen alliance." https://allseenalliance.org/framework/documentation/learn/core/system-description. Accessed on Sep. 30, 2017.

[51] "Open connectivity foundation (ocf)." https://openconnectivity.org/. Accessed on Sep. 30, 2017.

[52] "Home | iotivity." https://www.iotivity.org/. Accessed : 2017-09-30.

[53] "Discover ifttt and applets - ifttt." https://ifttt.com/discover. Accessed : 2017-09-30.

[54] B. Bertran, J. Bruneau, D. Cassou, N. Loriant, E. Balland, and C. Consel, "Diasuite : A tool suite to develop sense/compute/control applications," *Science of Computer Programming*, vol. 79, pp. 39–51, 2014.

[55] H. Jakob, C. Consel, and N. Loriant, "Architecturing conflict handling of pervasive computing resources.," in *DAIS*, vol. 11, pp. 92–105, Springer, 2011.

[56] R. B. Hadj, S. Chollet, P. Lalanda, and C. Hamon, "Sharing devices between applications with autonomic conflict management," in *Autonomic Computing (ICAC), 2016 IEEE International Conference on*, pp. 219–220, IEEE, 2016.

[57] C. Escoffier, S. Chollet, and P. Lalanda, "Lessons learned in building pervasive platforms," in *Consumer Communications and Networking Conference (CCNC), 2014 IEEE 11th*, pp. 7–12, IEEE, 2014.

[58] M. A. Feki, F. Kawsar, M. Boussard, and L. Trappeniers, "The internet of things : the next technological revolution," *Computer*, vol. 46, no. 2, pp. 24–25, 2013.

[59] S. Hodges, S. Taylor, N. Villar, J. Scott, D. Bial, and P. T. Fischer, "Prototyping connected devices for the internet of things," *Computer*, vol. 46, no. 2, pp. 26–34, 2013.

[60] "Freedomotic | open iot framework." http://www.freedomotic.com/. Accessed on Sep. 30, 2017.

[61] S. W. Lee, O. Prenzel, and Z. Bien, "Applying human learning principles to user-centered iot systems," *Computer*, vol. 46, no. 2, pp. 46–52, 2013.

[62] M. Compton, P. Barnaghi, L. Bermudez, R. GarcíA-Castro, O. Corcho, S. Cox, J. Graybeal, M. Hauswirth, C. Henson, A. Herzog, *et al.*, "The ssn ontology of the w3c semantic sensor network incubator group," *Web semantics : science, services and agents on the World Wide Web*, vol. 17, pp. 25–32, 2012.

[63] L. Lefort, C. Henson, K. Taylor, P. Barnaghi, M. Compton, O. Corcho, R. Garcia-Castro, J. Graybeal, A. Herzog, K. Janowicz, *et al.*, "Semantic sensor network xg final report," *W3C Incubator Group Report*, vol. 28, 2011.

[64] "Semantic sensor network ontology." https://www.w3.org/2005/Incubator/ssn/ssnx/ssn. Accessed on Sep. 30, 2017.

[65] N. Seydoux, M. B. Alaya, K. Drira, N. Hernandez, T. Monteil, and O. Haemmerlé, "San (semantic actuator network)." https://www.irit.fr/recherches/MELODI/ontologies/SAN.html. Accessed on Sep. 30, 2017.

[66] "Linked open vocabularies." http://lov.okfn.org/dataset/lov/vocabs/SAN. Accessed on Sep. 30, 2017.

[67] D. Bonino and F. Corno, "Dogont-ontology modeling for intelligent domotic environments," *The Semantic Web-ISWC 2008*, pp. 790–803, 2008.

[68] N. Lu, "An evaluation of the hvac load potential for providing load balancing service," *IEEE Transactions on Smart Grid*, vol. 3, no. 3, pp. 1263–1270, 2012.

[69] N. Nassif, S. Kajl, and R. Sabourin, "Optimization of hvac control system strategy using two-objective genetic algorithm," *HVAC&R Research*, vol. 11, no. 3, pp. 459–486, 2005.

[70] K. Bhattacharya, C. Gerede, R. Hull, R. Liu, and J. Su, "Towards formal analysis of artifact-centric business process models," in *Business Process Management*, pp. 288–304, Springer, 2007.

[71] K. Bhattacharya, R. Hull, J. Su, *et al.*, "A data-centric design methodology for business processes," *Handbook of Research on Business Process Modeling*, pp. 503–531, 2009.

[72] F. Leymann, D. Roller, and M.-T. Schmidt, "Web services and business process management," *IBM systems Journal*, vol. 41, no. 2, pp. 198–211, 2002.

[73] W. M. van Der Aalst, A. H. Ter Hofstede, B. Kiepuszewski, and A. P. Barros, "Workflow patterns," *Distributed and parallel databases*, vol. 14, no. 1, pp. 5–51, 2003.

[74] R. J. Glushko and T. McGrath, *Document engineering*. Mit Press Cambridge, 2005.

[75] D. Guinard, V. Trifa, and E. Wilde, "A resource oriented architecture for the web of things," in *Internet of Things (IOT), 2010*, pp. 1–8, IEEE, 2010.

[76] D. L. Nazareth, "Investigating the applicability of petri nets for rule-based system verification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 5, no. 3, pp. 402–415, 1993.

[77] D. Jackson, *Software Abstractions : logic, language, and analysis*. MIT press, 2012.

[78] "Alloy." http://alloy.mit.edu/alloy/. Accessed on Sep. 30, 2017.

[79] J. Woodcock and J. Davies, *Using Z : specification, refinement, and proof*, vol. 39. Prentice Hall Englewood Cliffs, 1996.

[80] D. Jackson, "Alloy : a lightweight object modelling notation," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 11, no. 2, pp. 256–290, 2002.

[81] K. Anastasakis, B. Bordbar, G. Georg, and I. Ray, "Uml2alloy : A challenging model transformation," in *International Conference on Model Driven Engineering Languages and Systems*, pp. 436–450, Springer, 2007.

[82] K. Anastasakis, B. Bordbar, G. Georg, and I. Ray, "On challenges of model transformation from uml to alloy," *Software and Systems Modeling*, vol. 9, no. 1, pp. 69–86, 2010.

[83] B. Bordbar and K. Anastasakis, "Uml2alloy : A tool for lightweight modelling of discrete event systems.," in *IADIS AC*, pp. 209–216, 2005.

[84] G. Georg, I. Ray, K. Anastasakis, B. Bordbar, M. Toahchoodee, and S. H. Houmb, "An aspect-oriented methodology for designing secure applications," *Information and Software Technology*, vol. 51, no. 5, pp. 846–864, 2009.

[85] K. Anastasakis, "Uml2alloy reference manual," *UML2Alloy Version : 0.52 [Online] available at http ://www. cs. bham. ac. uk/bxb/UML2Alloy/files/uml2alloy manual. pdf (retrieved 01/09/2009)*, 2012.

[86] A. Ouhabi, "Description et formalisation de règles pour le Smart Building," tech. rep., Université de Versailles Saint-Quentin-en-Yvelines, 09 2017.

[87] S. Bechhofer, "Owl : Web ontology language," in *Encyclopedia of Database Systems*, pp. 2008–2009, Springer, 2009.

[88] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean, *et al.*, "Swrl : A semantic web rule language combining owl and ruleml," *W3C Member submission*, vol. 21, p. 79, 2004.

[89] H. Knublauch, R. W. Fergerson, N. F. Noy, and M. A. Musen, "The protégé owl plugin : An open development environment for semantic web applications," in *International Semantic Web Conference*, vol. 3298, pp. 229–243, Springer, 2004.

[90] E. Prud, A. Seaborne, *et al.*, "Sparql query language for rdf," 2006.

[91] P. Horn, "Autonomic computing : Ibm\'s perspective on the state of information technology," 2001.

[92] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[93] C. Muller-Schloer and S. Tomforde Edts, *Organic Computing : Technical Systems for Survival in the Real World*. Springer International Publishing, to appear in 2017.

[94] R. Baghli and B. Traverson, "Verbalization of business rules : Application to ocl constraints in the utility domain," in *Model-Driven Engineering and Software Development (MODELSWARD), 2014 2nd International Conference on*, pp. 348–355, IEEE, 2014.

[95] S. Team *et al.*, "Semantics of business vocabulary and rules (sbvr)," tech. rep., Technical Report dtc/06–03–02, Object Management Group, Needham, Massachusetts, 2006.

[96] "Smart buildings with internet of things technologies." https://www.intel.com/content/www/us/en/smart-buildings/overview.html. Accessed on Sep. 30, 2017.

[97] "Ibm watson iot - iot buildings." https://www.ibm.com/internet-of-things/iot-zones/iot-buildings/. Accessed on Sep. 30, 2017.

[98] "Smart building solutions | internet of things software." https://www.mcssolutions.com/smart-buildings/. Accessed on Sep. 30, 2017.