Exploitation de l'isomorphisme de graphes et de l'alignement d'ontologies pour enrichir les objets logiciels de connaissances sémantiques : Application aux objets sages.

Abdelhafid Dahhani

UNIVERSITÉ SAVOIE MONT BLANC

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ SAVOIE MONT BLANC

Spécialité : **INFORMATIQUE**

Arrêté ministérial : 25 Mai 2016

Présentée par **Abdelhafid DAHHANI**

Thèse dirigée par **Ilham Alloui**, **Sébastien Monnet** et **Flavien Vernier**

préparée au sein du laboratoire **LISTIC**
et de l'école doctorale **SIE**

Exploitation de l'isomorphisme de graphes et de l'alignement d'ontologies pour enrichir les objets logiciels de connaissances sémantiques - Application aux objets sages.

Thèse soutenue publiquement le **14 Décembre 2023**,
devant le jury composé de :

Rapportrice - Sophie Ebersold
Professeur - Université Toulouse - Jean Jaurès / IRIT
Rapporteur - Jérôme Euzenat
Directeur de Recherche Inria - Université Grenoble Alpes / LIG
Examinatrice - Sylvie Despres
Professeur - Université Sobonne Paris Nord / LIMICS
Examinateur - Sylvain Giroux
Professeur - Université de Sherbrooke / DOMUS
Examinateur - Alexandre Benoit
Professeur - Université Savoie Mont Blanc / LISTIC
Directrice de thèse - Ilham Alloui
Maitresse de Conférences - Université Savoie Mont Blanc / LISTIC
Directeur de thèse - Sébastien Monnet
Professeur - Université Savoie Mont Blanc / LISTIC
Directeur de thèse - Flavien Vernier
Professeur - Université Savoie Mont Blanc / LISTIC

Résumé

Cette thèse aborde l'intégration de l'intelligence artificielle dans le monde réel, tout en respectant le concept de la technologie calme. À cette fin, le « Wise Object Framework » qui est basé sur la boucle MAPE-K, et son dérivé IAPE-K a été créé au sein du laboratoire LISTIC. La boucle IAPE-K signifie : Introspecter, Analyser, Planifier et Exécuter, sur la base des connaissances partagées.

Fondamentalement, les « Wise Object » sont des entités logicielles qui apprennent à se comporter par elles-mêmes grâce à des mécanismes d'introspection et de réflexivité pour enrichir leurs connaissances sur elle-même. Ces connaissances sont organisées sous la forme d'un graphe et représentent le comportement des « Wise Object ». Cependant, ce graphe manque d'informations sémantiques significatives, ce qui empêche les « Wise Object » de communiquer sémantiquement avec les humains. L'approche proposée pour enrichir ce graphe de connaissances avec de la sémantique peut être mise en œuvre comme un plug-in dans le Wise Object Framework, car ce dernier respecte le principe de modularité et de séparation des préoccupations.

Pour résoudre ce problème et permettre aux « Wise Object » de communiquer avec les humains, j'ai proposé un algorithme de correspondance entre le graphe de connaissances générées par le WO, et le formalisme appelé systèmes de transition symbolique d'entrée/sortie, qui représente le comportement sémantique d'une application. Ce formalisme est utilisé pendant la phase de conception du logiciel pour exprimer son comportement de manière sémantique. L'algorithme étend le graphe de connaissances généré par le « Wise Object » avec la sémantique humaine portée par le formalisme. Ce formalisme étant également représenté par un graphe, l'algorithme consiste en la mise en correspondance des deux graphes. L'une des principales limites de la première version de l'algorithme est le nombre multiple d'appariements obtenus. Cela rend le « Wise Object » confus et l'empêche de choisir la bonne sémantique pour communiquer avec l'humain. Je propose donc un second algorithme, qui étend le premier, en prenant en compte la sémantique disponible dans les deux graphes pour raffiner la mise en correspondance.

Une dernière partie présente les résultats obtenus dans le cadre d'une collaboration avec DOMUS de l'Université de Sherbrooke et le laboratoire LISTIC de l'Université Savoie Mont Blanc. Au cours de cette collaboration, j'ai travaillé avec DOMUS sur le problème inverse, la communication homme vers machine. Dans le contexte du maintien des personnes à domicile, le problème consiste à trouver tous les capteurs/effecteurs qui sont responsables d'aider une personne âgée à effectuer une activité telle que « Pauline gets out of bed, Pauline takes a glass of water ». Cette activité est décrite par les aidants/proches sous la forme d'une requête vocale transmise à la maison intelligente. Ce problème me conduit à l'alignement de

l'ontologie dans le contexte des systèmes d'assistance à l'autonomie.

Mots clés — systèmes de surveillance ; systèmes adaptatifs et contrôle ; systèmes à base de connaissances ; systèmes à événements discrets ; appariement de graphes ; sémantique ; ontologie ; maison intelligente.

Abstract

This thesis discusses the integration of artificial intelligence into real-world applications while respecting the concept of calm technology, which minimises intrusion into users' lives. A framework called "Wise Object Framework" based on the MAPE-K loop and its derivative IAPE-K is created at the LISTIC laboratory to develop such applications. IAPE-K loop stands for Introspect, Analyse, Plan and execute over a shared knowledge between its components.

Basically, "Wise Objects" are software entities that learn how they behave by themselves through introspection mechanisms, and collect data about their behaviour through self-monitoring to produce logs that I termed in this thesis "knowledge". This knowledge is organised in the form of a graph and represents the "Wise Object"'s behaviour. However, this graph lacks meaningful semantic information, which prevents "Wise Object" from communicating semantically with humans. The approach proposed to enhance a knowledge graph with semantic can be designed and implemented as a plug-in in the Wise Object Framework, as this latter respects the principle of modularity.

To address this issue and enable "Wise Object" communicate with humans (machine-human communication), I propose a matching algorithm between a graph, representing AI-generated knowledge, and a conceptual graph expressed as an Input/Output Symbolic Transition System, representing human conceptual views. This graph is given by experts to the "Wise Object" during the conceptual phase of the application and contains its semantic. The algorithm aims to extend AI-generated knowledge with human semantic by searching matches between the two types of graphs. A major limitation of the first version of the algorithm is the number of matches obtained, that leads to an inexact matching from a human point of view. It also confuses the "Wise Object" and prevents it from choosing the right match, as the challenge now lies at the semantic level. To meet this challenge, I propose to integrate ontologies into the matching computation process in order to contextualise it.

The last section presents the results obtained in collaboration with DOMUS laboratory of the University of Sherboorke and the LISTIC laboratory of Savoie Mont Blanc University. During this collaboration, I have worked with DOMUS on a dual problem of those addressed mainly in this thesis, human-machine communication, exploring the matching algorithm in the context of ambient assisted living systems for the elderly. The objective is to find all the sensors/effectors that able to help an elderly person to perform an activity such as "Pauline gets out of bed, Pauline takes a glass of water". This activity is described by the caregivers/relatives in a form of voice queries given to the smart home. This problem leads me to the ontology alignment in the context of ambient assisted living systems.

Keywords — monitoring systems ; adaptive system and control ; knowledge-based systems ; discrete-event systems ; graph matching ; semantic ; ontology ; smart home.

Remerciements

Je tiens à exprimer ma profonde gratitude envers mes encadrants.es, Ilham Alloui, Flavien Vernier et Sébastien Monnet, pour leur soutien inestimable, leurs conseils éclairés et leur patience infinie tout au long de cette aventure intellectuelle. Leurs orientations expertes et leur engagement indéfectible ont été la pierre angulaire de cette thèse. Leurs encouragements constants m'ont guidé à travers les défis et les triomphes de la recherche doctorale. Je suis convaincu que leur influence se fera sentir dans toutes mes futures entreprises académiques.

Je souhaite également exprimer ma reconnaissance envers le professeur Sylvain Giroux et Hubert Kenfack Ngankam pour leur acceptation de collaborer avec moi et leur encadrement pendant mon séjour au Canada. Leur expertise et leurs conseils ont enrichi mon expérience et ont contribué de manière significative à ma recherche. En outre, je tiens à adresser mes remerciements les plus sincères à Yannick pour ses explications détaillées sur le fonctionnement de quelques parties du projet sur lequel j'ai travaillé. Ses éclaircissements ont été précieux, m'aidant à mieux comprendre le contexte et à approfondir mes recherches. Son expertise a été une ressource inestimable.

Mes remerciements s'adressent également à Mme Cécile Pétigny pour sa relecture minutieuse de la proposition de collaboration. Grâce à son aide, j'ai pu obtenir le financement nécessaire à la réalisation de mon projet au Canada, à l'Université de Sherbrooke. Sa confiance en mon travail a été un élément essentiel dans la réalisation de ce projet. De même, je tiens à remercier chaleureusement l'organisation Mitacs pour son précieux soutien financier.

Je tiens également à rendre hommage à mes parents. Mon père, malgré son absence physique, reste une source d'inspiration constante. C'est grâce à lui que j'ai découvert le domaine fascinant de l'informatique, une passion qui a guidé chaque étape de mon parcours académique. Sa mémoire reste vivante en moi, m'incitant à persévérer même dans les moments les plus difficiles. À ma mère, mon roc, je dois un immense merci. Son amour inconditionnel, sa sagesse et son soutien ont été mes piliers. Elle a été mon modèle de force et de résilience, et chaque réussite que j'ai accomplie porte la trace de son influence positive dans ma vie.

Ce travail de thèse est le résultat d'un effort collectif, et je suis reconnaissant envers chacun d'entre vous pour votre contribution. Votre soutien inconditionnel a été un catalyseur essentiel dans la réalisation de ce rêve académique.

Table des matières

Table des figures

Introduction

Depuis 1971, le monde connaît une révolution technologique continue, de l'invention du microprocesseur à l'invention d'Internet, plus profondément, chaque solution apportée à un défi a conduit à un nouveau défi. Cela se résume par le passage à travers les trois différentes révolutions technologiques [GGM+21] : la machine à vapeur et l'industrie textile ; l'électricité, le moteur à combustion interne et la chimie ; les technologies de l'information et de la communication. Sur la base de la littérature que j'ai réalisée au cours des trois années de ma thèse, je pense que la nouvelle ère technologique qui arrivera respectera totalement la notion de technologie calme (vers la fin de la 4e révolution industrielle et numérique). Cette notion mobilise à la fois le centre et la périphérie de notre attention, a savoir, des systèmes intelligents qui fonctionnent indépendamment des humains « L'émergence de la superintelligence » comme mentionné dans [San13]. En revanche, la technologie logicielle, quant à elle, a connu une évolution exponentielle qui a conduit à l'intégration de l'intelligence artificielle (IA) dans le processus de développement d'applications intelligentes, ce qui a favorisé la fusion des deux domaines. Ces évolutions sont utilisées dans différents domaines, tels que la domotique, la médecine, la finance, sans oublier la communauté « faites-le vous-même – Do It Yourself (DIY) – », etc. Cette révolution se poursuit à un point où le monde de la recherche étudie comment intégrer l'IA dans le monde réel, le cas de la réalité augmentée.

Jusqu'à présent, il existe des systèmes qui connaissent et analysent leurs comportements, mais le résultat de cette analyse est toujours difficile à exprimer aux humains de manière compréhensible, c'est-à-dire en langage naturel, car ces systèmes n'ont pas d'idée sémantique sur leurs états. Dans cette thèse, je traite la relation entre la connaissance numérique d'un système, à savoir les états de ce

système, et la connaissance sémantique donnée par un expert au moment de la conception du système. De manière générale, deux types de communication sont traités : machine-humain et humain-machine.

Vers des objets connectés quasi-autonomes

Avec l'avènement de la technologie et de l'Internet des objets – Internet of Things (IoT) –, l'environnement informatique est de plus en plus utilisé par les entreprises et revêt une importance croissante dans la vie humaine. À cette fin, un nouveau paradigme apparait pour tenir compte de la complexité et de l'hétérogénéité de ces environnements. En 2001, Paul Horn, d'IBM, a proposé une nouvelle approche pour développer des systèmes informatiques, qu'il a baptisées « informatique autonome – autonomic computing– » [SK16]. Ce nouveau paradigme présente quatre caractéristiques majeures, à savoir l'auto-guérison, l'auto-configuration, l'auto-optimisation et l'auto-protection, connues sous le nom d'auto-CHOP [ST05]. Ce quadruple est organisé sur cinq niveaux [AD14] : basique (niveau 1), administré (niveau 2), predictif (niveau 3), adaptatif (niveau 4) et autonome (niveau 5). En général, le principal objectif des systèmes autonomes est l'autogestion, qui n'est possible que si tous les composants du système travaillent ensemble pour y parvenir [SK16]. En plus des composants logiciels et matériels, il y a les humains qui interagissent avec ces systèmes. Sans l'adaptabilité au contexte, ces systèmes risquent d'être très intrusifs dans la vie quotidienne des humains. En effet, la connaissance du contexte et l'autonomie sont deux éléments clés indispensables pour réaliser une intégration efficace des systèmes cybers-physiques modernes à forte composante fonctionnant dans des environnements ouverts et non déterministes [CFLP16], ainsi qu'une communication claire et sémantique avec l'humain.

En négligeant les informations du contexte, les systèmes autonomes ne sont pas en mesure de communiquer de manière compréhensible avec les humains, à savoir une communication sémantique, ni d'avoir une indépendance basique (selon ma vision, un pas vers des systèmes singuliers). Par exemple, dans un système domotique, si l'utilisateur a oublié d'éteindre la télévision, il doit être connecté à distance pour faire cette action s'il n'y a pas un capteur de présence dans le salon. Le principal problème dans ce scénario est que l'utilisateur doit réagir pour réadapter le système au nouveau contexte (l'utilisateur est à l'extérieur), et même s'il y a un capteur de présence dans le salon, est-ce qu'il a la capacité de comprendre les informations du contexte pour prendre la bonne décision ? Est-ce qu'il est capable de donner une sémantique à ses différents états, afin de les analyser et d'avoir une information compréhensible par l'humain ? Ainsi, pour donner à un système la capacité de réagir à une situation donnée en analysant les données (journaux du système) [ABPV19] et en respectant la notion de technologie calme présentée

par Mark Weiser et John Seely Brown en 1995 [WB96], le concept d'objet sage – Wise Object (WO) – qui peut connaître son comportement par lui-même sans aucune interaction de la part de l'utilisateur final a été proposé et est étudié au sein du laboratoire LISTIC [1][ABPV19]. En précisant que ce type d'objet fait intervenir tous les sous-domaines de l'informatique, de l'ingénierie logicielle à l'IA, et se caractérise par :

- **Se connaître par soi-même**, c'est-à-dire apprendre comment il est censé se comporter afin d'éviter toute interaction de la part de l'utilisateur. Par exemple, un volet roulant sage connaît ses différents états.
- **Connaître par soi-même son utilisation**, c'est-à-dire savoir comment il est utilisé par chaque utilisateur. Par exemple, un volet roulant sage connaît le comportement de ses utilisateurs en ce qui le concerne.
- **Améliorer ses capacités**, c'est-à-dire que tout objet/système intelligent doit pouvoir améliorer la qualité des services qu'il offre. Par exemple, un volet roulant sage adapte son comportement en fonction du comportement de ses utilisateurs.

Les travaux (partie II) que j'ai menés dans le cadre de cette thèse portent sur les WO.

> *In an ideal world, computers will blend into the landscape, will inform but not overburden you with information, and make you aware of them only when you need them [Tug04]. – Alexandru Tugui*

Contexte de la thèse

Désormais, l'informatique est ubiquitaire, elle est intégrée dans notre vie quotidienne. Par conséquent, les systèmes prenant en compte le contexte entrent en jeu pour minimiser l'intrusion des appareils connectés dans la vie des humains, d'où plus d'invisibilité et des actions silencieuses. Plus profondément, il se caractérise par le fait d'avoir une vision sur l'environnement, de réagir et de notifier les utilisateurs sans presque aucune intervention de leur part (Implicit Output [2]), mais aussi d'analyser le comportement des utilisateurs pour pouvoir s'adapter à eux. J'ai trouvé dans mon parcours de recherche plusieurs exemples contenant de nombreux problèmes, tels que l'adaptabilité du système et la conscience du

1. Laboratoire d'Informatique, Systèmes, Traitement de l'Information et de la Connaissance.
2. Implicit Output est le résultat d'un système informatique qui n'est pas directement lié à une entrée explicite et qui s'intègre de manière transparente à l'environnement et à la tâche de l'humain [Sch05].

contexte. Par exemple, des solutions qui aident les personnes souffrant de troubles cérébraux traumatiques ou de la démence d'Alzheimer. Ces solutions sont basées sur la spécification formelle de scénarios à exécuter et d'ontologies [3] [Nga19], sachant qu'un scénario est donné par les utilisateurs de ces solutions, et décrit la succession d'activités à mener pour atteindre un but (détaillé dans le chapitre 6). Ce qui surcharge les utilisateurs finaux et les placent au centre du processus de fonctionnement de ces systèmes, par conséquent, cette représentation est adaptée à un type d'application spécifique. Ce type de solution qui est axé sur l'utilisateur final se classe dans le concept de DIY [DRSC+12] est un peu éloigné des technologies calmes, puisqu'une forte configuration de nouveaux scénarios est toujours nécessaire. Un autre exemple est le Context Toolkit, un framework conceptuel, qui fournit des composants réutilisables pour soutenir le prototypage rapide d'applications contextuelles basées sur des capteurs [DAS01], ce cadre présente un certain nombre de problèmes, tels que sa capacité limitée à s'adapter à une augmentation du nombre de composants, en complément, il ne peut pas identifier la manière dont il est utilisé, de ce fait, il ne peut pas changer son comportement pour se conformer à ses nouvelles habitudes. Certains auteurs ont également utilisé des systèmes multi-agents pour mettre en œuvre la représentation du contexte [BBB05a], qui, selon ma vision, oublient les notions de « généralité moderne » qui se base sur l'analyse des données environnementales et leur usage à des fins d'adaptabilité au contexte.

> *Ubiquitous computing forces the computer to live out here in the world with people. – Mark Weiser*

Selon ma vision, la citation de Mark Weizer a besoin d'une autre partie :

> *To prevent overburdening them, computer needs a kind of consciousness, or at least a form of contextual and semantic awareness.*

La conscience – awareness – est un autre pilier de la technologie calme après avoir déduit de ce que Mark Weiser a dit à propos de la technologie calme : « L'information provenant de la technologie se déplace en douceur vers l'attention de l'utilisateur lorsque cela est nécessaire, mais reste calmement à la périphérie de l'utilisateur, l'informant ainsi, mais ne nécessitant pas l'attention ou la concentration de ce dernier ». Fondamentalement, *la conscience* chez la machine représente la capacité à collecter – à fournir des données internes – sur elle-même et par elle-même. Par exemple, lorsqu'une entité collecte des informations et des données sur ses capacités (ce qu'elle est capable de faire) et son utilisation (ce qu'on lui demande de faire), ces informations seront numériques dans la plupart des cas et,

3. Une ontologie est une spécification explicite de la structure d'un certain domaine [SPSL02].

à mesure que les systèmes évoluent vers des communications machine-humain, ces informations ont besoin d'une sémantique pour les rendre claires aux humains (ma contribution dans cette thèse, partie II).

Problématique

La sémantique permet à la machine d'interpréter les informations et de communiquer avec les humains d'une manière claire dans un environnement variable, trouvant ainsi les connaissances et les ressources nécessaires à une requête ou à une interaction implicite de la part des humains [PB08]. Tout cela se résume par la « conscience sémantique – semantic awareness – ». La notion de sémantique est utilisée dans un certain nombre de domaines, comme dans les systèmes de stockages, une organisation sémantique, connue sous le nom de « SmartStore », qui exploite la sémantique des métadonnées des fichiers pour regrouper judicieusement les fichiers corrélés dans des groupes sémantiques [HJZ+09]. Également, dans l'agriculture, grâce aux systèmes d'aide à la décision, la demande de l'agriculteur est contextualisée afin de l'aider à choisir les semences à planter, comme mentionné dans [BBZP19]. Dans un contexte plus proche de mes travaux, citons la thèse de Fabien Sartor [Sar12a] et de la thèse d'Hubert Kenfack Ngankam [Nga19]. Ces premiers travaux, ceux de Sartor, ont réussi à donner aux experts la possibilité de décrire le comportement d'un objet/système en donnant ses différents états, ce qui constitue une sémantique de tout objet/système, la signification de l'information qui circule dans l'objet/système sous la forme d'un formalisme comportemental[4]. Mais leur solution était très éloignée du principe de technologie calme, ce qui est un inconvénient majeur, de plus, les experts peuvent faire évoluer les fonctionnalités de l'objet en oubliant de mettre à jour le modèle comportemental. Les seconds travaux combinent la description des scénarios crées par des utilisateurs finaux et les ontologies pour rendre une application sensible au contexte, à savoir la modélisation de l'environnement. Par exemple, un système ontologique qui représente tous les composants d'une maison est nécessaire : les murs, les pièces et les fournitures (cette liste n'est pas exhaustive). Puis, vient la phase de description des scénarios qui contiennent des tâches à faire au quotidien « aller à la cuisine pour boire un verre d'eau ou se lever du lit pour aller aux toilettes ». Cette approche ne respecte pas la notion de technologie calme vu qu'elle implique fortement les utilisateurs dans le processus de création des scénarios et le système crée n'est pas autonome.

Le comportement autonome est une propriété essentielle des systèmes/objets auto-adaptatifs, car l'utilisateur final n'interagit pas ou rarement avec eux, puisque

4. Un formalisme comportemental permet de modéliser le comportement des dispositifs domotiques ou de tout objet connecté, quel que soit leur type, logiciel ou matériel.

certains systèmes peuvent nécessiter une mise à jour continue de leurs politiques ou procédures organisationnelles. Les WO arrivent à ce niveau grâce à leurs processus d'apprentissage, qui les aident à se connaître de manière autonome et à connaître les habitudes de leurs utilisateurs, ce processus est alimenté par l'autocollection des données. Un exemple est celui d'un système domotique qui recueille le comportement d'une personne dans une pièce et l'analyse pour pouvoir agir « silencieusement » en cas de besoin. De tels systèmes doivent requérir un *minimum d'attention* de la part de leurs utilisateurs finaux tout en étant capables de s'adapter aux changements de leurs comportements, dans la plupart des cas, l'attention requise sera la prise en compte des informations du système vers l'humain, ce qui nécessite un langage de communication compréhensible par ce dernier, à savoir la connaissance de la signification des états du système par le système (c.-à-d. la sémantique). En général, la connaissance dans les systèmes basés sur l'IA peut être fournie de deux manières : en décrivant a priori l'arrangement des activités à réaliser par le système, ou en laissant le système acquérir la connaissance requise à l'aide de mécanismes d'apprentissage.

Dans le premier cas, des ontologies ou des scénarios sont généralement utilisés pour décrire l'agencement des activités pour atteindre un objectif, comme décrit dans [BC08, KNPF+17]. Bonino et Corno [BC08] évoquent que le comportement fonctionnel ainsi que l'interopérabilité des entités du système sont décrits a priori à l'aide de diagrammes d'états. Kenfack Ngankam [KNPF+17] va un peu plus loin en combinant des ontologies pour concevoir des systèmes d'assistance à l'autonomie avec des spécifications basées sur la logique et les analyseurs pour vérifier les clauses logiques avant le déploiement du système, afin de créer des scénarios pertinents. Dans ces approches, l'utilisateur final est au cœur du processus de création de scénarios, comme décrit dans [WL15, RNP+16]. *Dans le second cas*, les connaissances sont fournies par le système assisté par l'IA dans des représentations et des vues qui ne sont pas nécessairement compréhensibles par les humains. Ceci est lié au vaste problème de la compréhensibilité de l'IA et à la distance entre les vues du domaine métier et du domaine technologique.

Objectifs

Dans cette thèse, je travaille sur un framework logiciel existant, qui offre et utilise un ensemble de mécanismes de base, par exemple, en respectant l'architecture en couches, en offrant un concept de surveillance, en utilisant l'introspection et l'analyse nécessaires pour développer des applications basées sur l'IA à travers le concept des WO. L'objectif de base du framework est de séparer les préoccupations, c'est-à-dire que les développeurs de logiciels devraient pouvoir utiliser le cadre avec un minimum de contraintes et d'intrusion dans le code source de

l'application/métier. Au final, chaque dispositif ou logiciel connecté à ce système aura trois éléments de base : *une conscience, une sagesse et une sémantique*[5]. Ces éléments sont définis dans le chapitre 1 de cette thèse.

De la machine vers l'humain : le WO acquiert par lui-même les connaissances sur ses capacités – services à fournir – et son utilisation pour modérer l'attention des utilisateurs finaux (technologie calme) [Wei91, Tug04], il analyse également ces connaissances pour en générer de nouvelles. En conséquence, il produit un graphe de connaissances. Je considère les graphes de connaissances comme la manière la plus naturelle de modéliser la dynamique d'un système. Le désavantage de ces graphes générés par les WO est que les données numériques fournies ont une signification partielle pour les humains, ainsi, une compréhensibilité insuffisante. Toutefois, comme indiqué au début de cette section et évoqué dans la littérature [CJMR08, CJMR07], les formalismes comportementaux comme les « Input/Output Symbolic Transitions System (IOSTS) » sont souvent utilisés pour modéliser le comportement des systèmes afin de les gérer en utilisant la synthèse d'oracle ou de contrôleur, et comme ce type de graphe est conceptuellement compréhensible par les humains, il possède une sémantique qui peut enrichir les graphes de connaissances générés par les WO en leur apportant cette sémantique.

La figure 1 présente les principaux défis de cette thèse, qui consistent à ajouter à un WO une sémantique afin de le faire évoluer vers un WO sémantiquement conscient, capable de comprendre le sens humain de chaque action. Par conséquent, l'objectif est de répondre à ces questions :

- Comment rendre un WO capable de communiquer avec les humains de manière compréhensible ?
- Quelles sont les limites d'un WO sémantiquement conscient ?

De l'humain vers la machine : Dans la thèse de M. Hubert Kenfack Ngankam qui a pour thème le modèle sémantique de l'intelligence ambiante pour le développement autonome d'habitats intelligents. Ce modèle présente de nombreux avantages, comme celui d'aider les personnes âgées atteintes de la maladie d'Alzheimer et de rassurer leurs proches en leur envoyant des informations récentes sur leur état de santé, leur hygiène, etc. Ce type de système fonctionne avec des scénarios décrits d'une manière explicite (par étape) par les proches de ces personnes, ce qui les place au centre du fonctionnement du système (inconvénient 1), et ce sans oublier que ce type de système n'a aucune adaptabilité au comportement de ces personnes[6] (inconvénient 2). Dans cette thèse, je traite le premier inconvénient en rendant la description des scénarios plus explicite (en langage naturel) qu'implicite,

5. La sémantique est la problématique traitée dans cette thèse.

6. Il est très difficile de concevoir un système qui s'adapte au comportement d'une personne atteinte de ce type de maladie, mais il y a toujours une solution quelque part, comme l'appren-

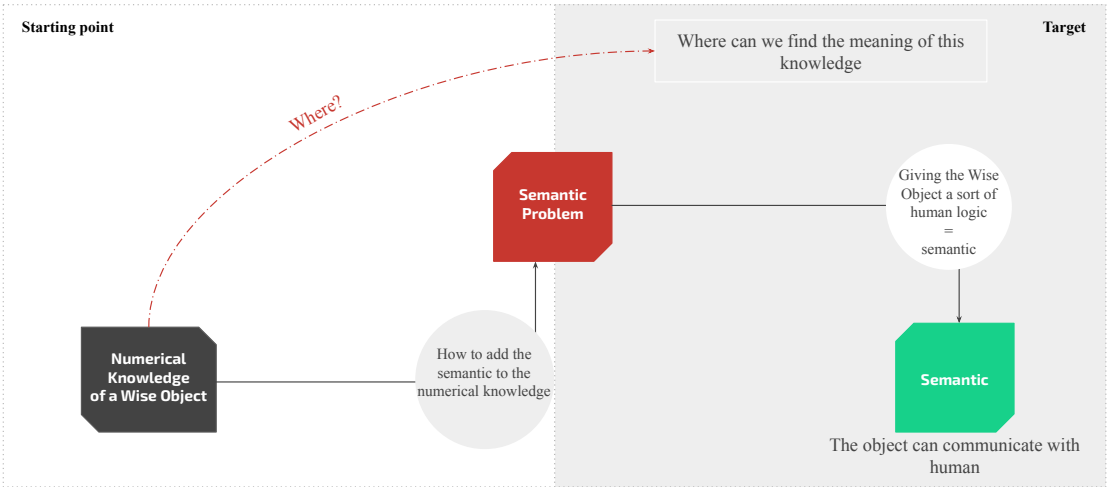


FIGURE 1 : Schéma problématique (1) – un objet de connaissance à un objet sémantique.

où j'ai collaboré avec le professeur Sylvain Giroux, chercheur au sein du laboratoire DOMUS, et M. Hubert Kenfack Ngankam, professionnel de la recherche au sein du même laboratoire.

La figure 2 présente la problématique de la communication homme-machine, partant d'un scénario exprimé en langage naturel, l'objectif consiste à le transformer en scénarios exploitables par la machine. Par conséquent, nous devons répondre à ces questions :

- Comment transformer le langage naturel en scénarios exploitables par la machine ?
- Quelles sont les limites d'une telle approche ?

Plan de la thèse

Hormis l'introduction, le contexte et l'objectif, le manuscrit de cette thèse se compose de deux parties.

La première partie : est consacrée à l'état de l'art et comprend les trois chapitres suivants :

- Le premier chapitre présente l'architecture actuelle et les limites des WO.

tissage semi-supervisé, qui interagit légèrement avec leurs proches sans les surcharger ou en faire le centre du système.

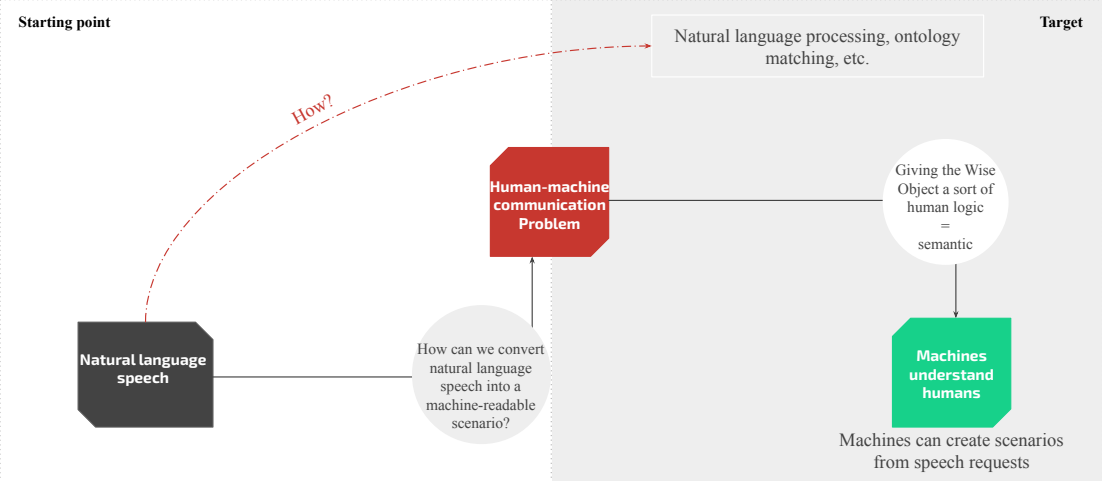


FIGURE 2 : Schéma problématique (2) – d'une requête verbale à la création de scénarios.

- Le deuxième chapitre présente de manière générale l'histoire des graphes dans différents domaines, et la définition formelle de modèles comportementaux complexes. Plus précisément, les graphes d'état de transitions (STG) et les IOSTS, en apportant des illustrations concrètes.

- Le dernier chapitre de cette partie présente les différents types d'ontologie. Plus précisément, les ontologies et les relations sémantiques telles qu'elles existent dans la base de données lexicale WordNet [7].

La seconde partie : est consacrée aux trois contributions apportées au cours de cette thèse :

- Le quatrième chapitre introduit deux nouveaux algorithmes d'appariement de graphes entre deux graphes de connaissances de nature différente (STG et IOSTS), l'un univarié et l'autre multivarié.

- Le cinquième chapitre présente la solution au problème d'appariement entre les deux formalismes en intégrant la sémantique entre les variables sous la forme d'une matrice sémantique afin de donner à l'algorithme une capacité de décision quasi humaine.

- Le dernier chapitre de mon mémoire traite de la relation entre les systèmes sages et les notions DIY. Plus particulièrement sur la combinaison de ces deux

7. WordNet est une base de données lexicale de relations sémantiques entre les mots. WordNet relie les mots en relations sémantiques, y compris les synonymes, les hyponymes et les méronymes. [Mil95a]

concepts pour aider les personnes atteintes de la maladie d'Alzheimer. Ce chapitre sera le resultat d'un projet de collaboration entre l'université Savoie Mont Blanc et l'université de Sherbrooke, précisément entre le laboratoire LISTIC et DOMUS.

I

Contexte de la thèse et état de l'art

Introduction

La réutilisabilité et l'évolution dans des systèmes à grande échelle sont deux problèmes majeurs rencontrés lors du développement d'applications basées sur l'intelligence artificielle, car ces applications sont spécifiques à des domaines particuliers. En outre, elles sont souvent développées par des communautés différentes (IA, Ingénieurs logiciels, entreprises, etc.). Le principal défi est alors de fournir un support pour le développement d'applications basées sur l'IA d'une manière similaire au développement de logiciels « classiques » en utilisant des méthodes, des outils et des langages de génie logiciel. L'ensemble du cycle de vie des applications basées sur l'IA doit être pris en charge afin de garantir la qualité et la viabilité des logiciels. Comme les systèmes logiciels sont désormais omniprésents dans notre vie quotidienne, que leur utilisation peut varier en fonction de l'utilisateur final et qu'ils peuvent évoluer dans le temps, je me concentre sur les applications basées sur l'IA qui sont capables d'acquérir des données [8] à partir de, ou sur, leur environnement et leur comportement, de les manipuler et de les analyser, soit pour aider les utilisateurs à prendre des décisions, soit pour adapter de manière autonome leur comportement aux besoins des utilisateurs, ce qui a permis aux WO de voir le jour. La première étape vers le concept des WO est de respecter la notion de « technologie de calme » revendiquée par Mark Weiser et John Seely Brown dans [WB96], en donnant à une entité la capacité de s'adapter de manière autonome à son utilisation. Le défi qui s'ensuit est de pousser le concept des WO vers la sémantique. Ce défi exige que chaque objet sage soit conscient de son contexte sémantique pour pouvoir comprendre son contexte, agir de manière fluide avec son environnement et communiquer avec l'humain d'une manière compréhensible, sans attendre une action de stimulation de la part de ce dernier. Comme la recherche sur le concept de WO avance pas à pas, la définition d'un contexte sémantique spécifique (c.-à-d. lié à un domaine/métier) d'un WO ne sera pas abordée dans cette thèse. En revanche, j'utilise la sémantique générale telle que présentée dans l'ontologie lexicale WordNet [RB90].

Cette première partie est organisée comme suit : le premier chapitre traite le point de départ de cette thèse, les WO et leur Framework. Le deuxième chapitre aborde la notion de graphes comportementaux, et le dernier chapitre est consacré aux ontologies et à leur relation avec la sémantique.

8. Données que j'appelle des connaissances et qui sont représentées sous la forme de graphes de connaissances.

Sommaire

Chapitre 1

Architecture et limitations des WO

Dans notre vie quotidienne, différents types de technologies sont utilisés, allant du suivi et de la détection des problèmes de santé, à titre d'exemple, le cas d'une montre connectée qui envoie l'état cardiaque d'une personne âgé, aux opérations les plus complexes comme l'intelligence domotique, la reconnaissance faciale, etc. Lors de l'utilisation de ce type de technologie, l'être humain est toujours le centre d'attention. Ce fait est considéré comme un inconvénient de tout dispositif qui n'arrive pas à fonctionner correctement sans intervention humaine, à savoir dans la périphérie, où les objets connus sous le nom de « WO » interviennent, ce type d'objet connaît son environnement, ses fonctions sans aucune intervention humaine.

Ce chapitre couvre tous les éléments de base pour comprendre le fonctionnement des WO, l'origine du concept et le chemin suivi pour aboutir aux questions abordées au cours de cette thèse.

1.1 Idée de base et définitions

L'idée de base qui sous-tend le concept d'un WO est de donner à une entité[1] les mécanismes de base pour apprendre son comportement par introspection et analyse. L'objectif est d'aller plus loin en permettant à tout type d'entité d'exécuter des boucles de « Surveillance, Analyse, Planification et Exécution » basées sur la « Connaissance », appelée MAPE-K [AV18]. Au cœur de ce concept, le WOF [AEV15] a été construit avec des décisions de conception principalement guidées par des exigences de réutilisabilité et de généricité : le cadre doit pouvoir être maintenu et utilisé dans différents domaines d'application avec différentes stratégies (par exemple, différentes approches d'analyse).

Dans un souci de clarté, nous avons emprunté certains termes utilisés pour les humains afin de faire référence aux capacités que possède un WO. La *conscience* et la *sagesse* reposent toutes deux sur la *connaissance*. Inspiré par [DP98], j'introduis quelques définitions de ces termes couramment utilisés par les humains [Cam22] et propose celles que j'ai choisies pour les WO.

Donnée – Data – La donnée est une mesure brute, c'est une valeur obtenue à partir d'un dispositif électronique ou d'un composant logiciel. Elle représente une observation quantifiable. Généralement collectée, stockée, traitée, transmise et analysée, la donnée n'a pas de sens prise individuellement. Pour être pertinente, elle doit être associée à un contexte spécifique. Dans ce cas, on parle *d'information* [Nga19].

Information – Information – Les informations fournissent aux données le contexte dans lequel elles peuvent être analysées et traitées. L'information devient alors compréhensible par les humains et les machines, elle donne un sens particulier à chaque donnée. Les informations sont significatives et permettent de mieux répondre aux questions Quand ? Où ? Qui ? Quoi ? etc. Par exemple, Tome a placé sa main sur une gazinière chaude. Par contre, les réponses à ces (Quand ? Où ? Qui ? Quoi ?) questions simples ne permettent pas de faire des déductions. Pour y parvenir, il faut monter en abstraction vers la connaissance [Nga19].

Connaissance – Knowledge – La connaissance est un ensemble de faits, d'informations et de compétences acquises par une personne par l'expérience ou l'éducation ; la compréhension théorique ou pratique d'un sujet [Cam22]. Dans le cas des WO, la connaissance se réfère à l'information, aux règles d'inférence et à l'information déduite de celles-ci, par exemple, allumer un radiateur provoquera un changement de température. Le problème de la connaissance seule est qu'elle ne

1. Logicielle, objet connecté, composant, sous-système, etc.

permet pas à un WO de connaître ses capacités par soi-même. Pour y parvenir, il faut aller un peu plus loin vers la conscience.

Conscience – Awareness – La conscience correspond à la connaissance de l'existence de quelque chose, ou la compréhension d'une situation ou d'un sujet au moment présent, sur la base d'informations ou d'expériences [Cam22]. Dans le cas des WO, elle représente leur capacité à collecter – à fournir des données internes – sur eux-mêmes par eux-mêmes. Par exemple, c'est lorsqu'une entité recueille des informations et des données sur ses capacités *(ce qu'il est capable de faire)* et son utilisation *(ce qu'on lui demande de faire)*. Les capacités sont les services/fonctionnalités que le WO peut rendre. La section 1.2.1 détaille comment la conscience est mise en œuvre dans un WO. Notez que la brique de la conscience dans un WO ne permet pas d'analyser les données collectées. Pour y parvenir, il faut monter d'un cran vers la sagesse.

Sagesse – Wisdom – La capacité d'utiliser ses connaissances et son expérience pour prendre de bonnes décisions et émettre des jugements [Cam22]. Dans le cas des WO, La sagesse est la capacité d'analyser les informations collectées (Awareness) et les connaissances stockées relatives à leurs capacités et à leur utilisation pour produire des informations utiles. Notez que la brique de la sagesse dans un WO ne permet pas de communiquer le résultat de l'analyse à l'humain de manière claire (vu que les données collectées sont des données numériques) en utilisant un langage compréhensible par les humains. Pour y parvenir, il faut monter d'un cran vers la sémantique.

Sémantique – Semantic – Sens donné à une notion afin qu'elle puisse être comprise par les humains [Cam22]. Cette définition s'applique également aux WO, car la sémantique est utilisée pour communiquer avec les humains. La valeur « 100 » d'une variable « Data » ne signifie rien pour un humain si on ne lui précise pas l'information qu'elle représente un pourcentage d'humidité. Il convient de noter que la sémantique d'un mot ou d'une expression peut varier en fonction du contexte dans lequel il est utilisé, de sorte que nous ne pouvons pas nous fier uniquement à la sémantique pour communiquer avec les humains. Par conséquent, il faut aller un peu plus loin vers le contexte (voir chapitre 3).

Sensation – Sensation – La sensation dans un WO est représentée par une distance entre le comportement ou l'utilisation actuel et le comportement habituel du WO lui-même. Une brève présentation avec une explication seront faites respectivement dans les sections 1.2.1 et 1.2.3 car ce concept dépasse le cadre de la thèse.

Il peut sembler que *la sagesse* soit identique à *l'intelligence* ; les figures 1.1 et 1.2 illustrent respectivement la relation entre les données externes et l'intelligence, et les données internes et la sagesse pour comprendre la principale différence entre l'intelligence et la sagesse selon le concept des WO. Comme l'indique la figure 1.2,

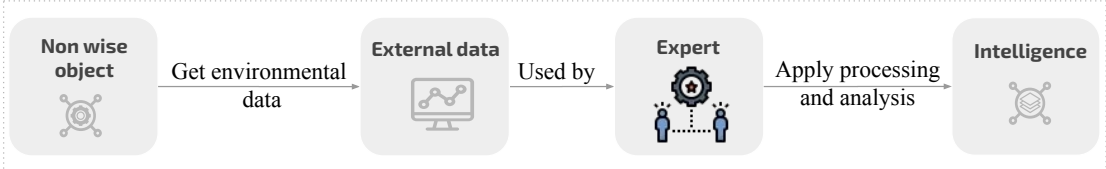


FIGURE 1.1 : Représentation du comportement d'un objet non sage

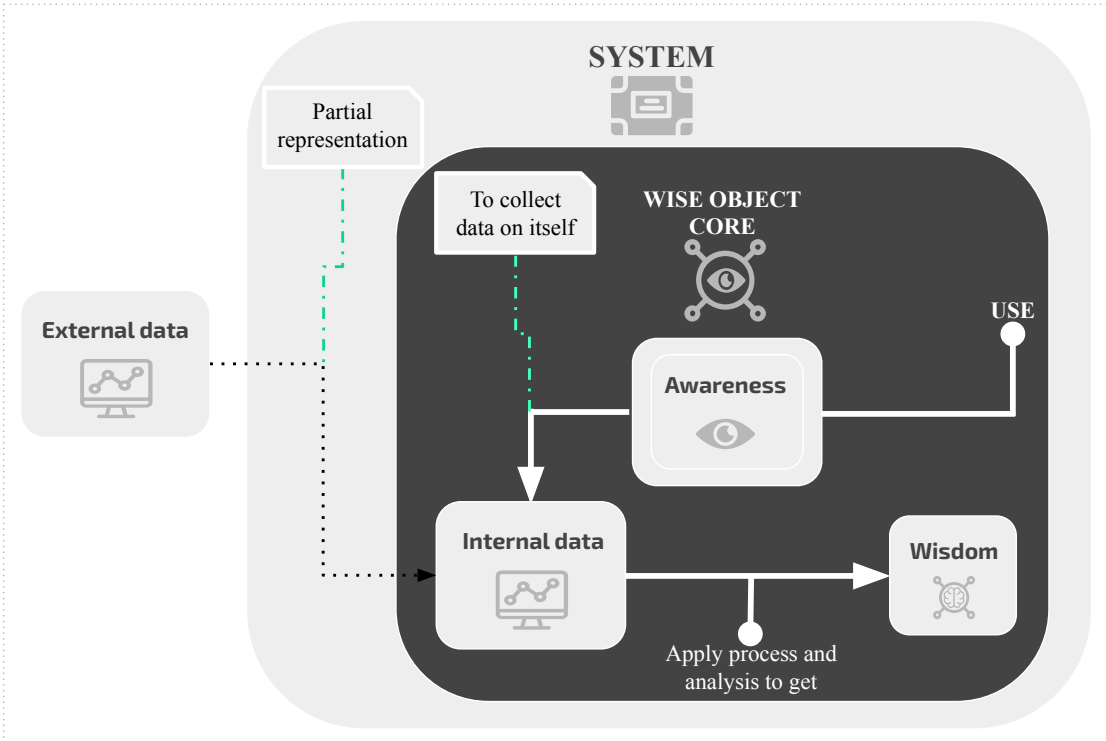


FIGURE 1.2 : Représentation du comportement d'un WO [DAMV22a]

les WO utilisent la conscience pour collecter des données internes qui seront mélangées aux données externes par un processus appelé représentation partielle. L'analyse et le traitement de ces données internes donnent la notion de sagesse, toutes ces opérations ont eu lieu à l'intérieur même du WO. Alors que l'intelligence se réfère à l'analyse, le traitement de données externes d'un objet non sage et la

construction des modèles d'IA par des experts comme le montre la figure 1.1. Les prochaines sections détaillent les six dernières définitions.

1.2 Objet sage, système sage et framework d'objet sage

1.2.1 Objet sage

Un WO est un objet qui a la capacité d'apprendre sur son comportement et également sur le comportement de ses utilisateurs en fonction de l'évolution du contexte. En général, un WO est une entité simple avec des options avancées comme mentionné ci-dessous [DAMV22a] :

- Capable d'apprendre sur lui-même (utilisations et capacités).
- Capable d'apprendre sur les autres (les autres WO).
- Capable de planifier et de prendre des décisions.

En conséquence, un WO est capable de minimiser le degré d'attention des utilisateurs finaux.

Vue d'ensemble de l'architecture

Un WO est considéré comme un avatar logiciel conçu pour être connecté à des dispositifs physiques (par exemple un chauffage, un aspirateur, une ampoule) ou à une entité logique [AV18] (figure 1.3). Dans le cas d'une ampoule, le WO peut apprendre sur la base de la durée d'allumage de l'ampoule, il permet en outre d'améliorer ses performances (un temps d'allumage réduit, moins de consommation d'énergie), ce type d'objet aide à respecter une telle contrainte afin de le classer dans le domaine des technologies calmes :

- Autonomie, il est capable de se comporter sans intervention humaine.
- Adaptabilité, il modifie son comportement lorsque les besoins ou son environnement changent.
- Intelligence, il s'observe et observe son environnement, l'analyse et utilise ses connaissances pour décider de son comportement.
- Communication, il communique avec son environnement qui comprend d'autres WO et des utilisateurs finaux de manière décentralisée (c'est-à-dire à différents endroits).

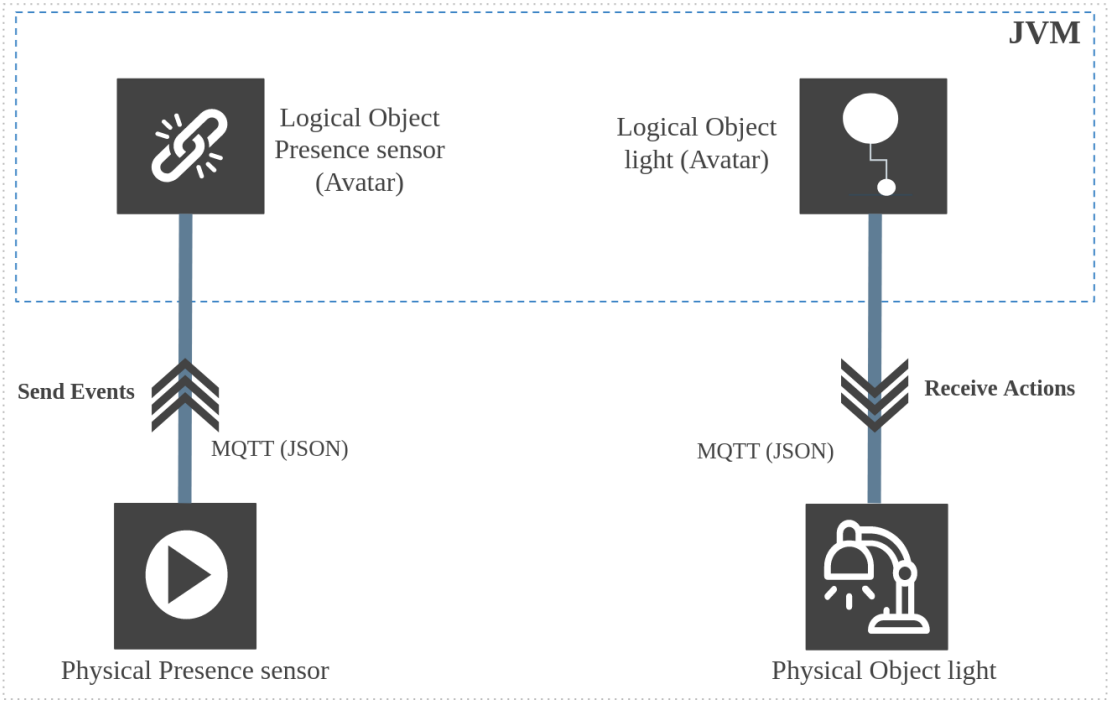


FIGURE 1.3 : Connexion entre les objets physiques et logiques (Avatars).

La figure 1.3 est une représentation simple de la connexion entre les objets physiques et logiques au sein du WOF. Ces objets peuvent envoyer et recevoir des événements ou actions via des protocoles comme le protocole MQTT [2] (la figure est détaillée dans la section 1.2.2). D'un point de vue strictement architectural, chaque WO est :

- Une entité logicielle indépendante – Standalone – (objet, composant, etc.).
- Un avatar logiciel conçu pour être un proxy pour des dispositifs physiques (par exemple, un chauffage, un aspirateur, une ampoule) [ABPV19].
- Un avatar logiciel conçu pour être un proxy pour une entité logicielle existante (objet, composant, etc.).

Sachant qu'un « proxy » en termes simples, est une enveloppe qui fait passer l'invocation d'une fonction par son propre mécanisme, ce qui permet d'ajouter éventuellement des fonctionnalités [Ora]. Généralement, il est utilisé par les développeurs des frameworks.

2. Message Queuing Telemetry Transport (MQTT) est un protocole de messagerie publication/abonnement – publish/subscribe – basé sur le protocole TCP/IP.

Position d'un WO dans un système sage

Un système sage – Wise System (WS) – est un ensemble de WO qui communiquent leurs états à ce système, comme l'illustre la figure 1.4. Elle met en évidence un

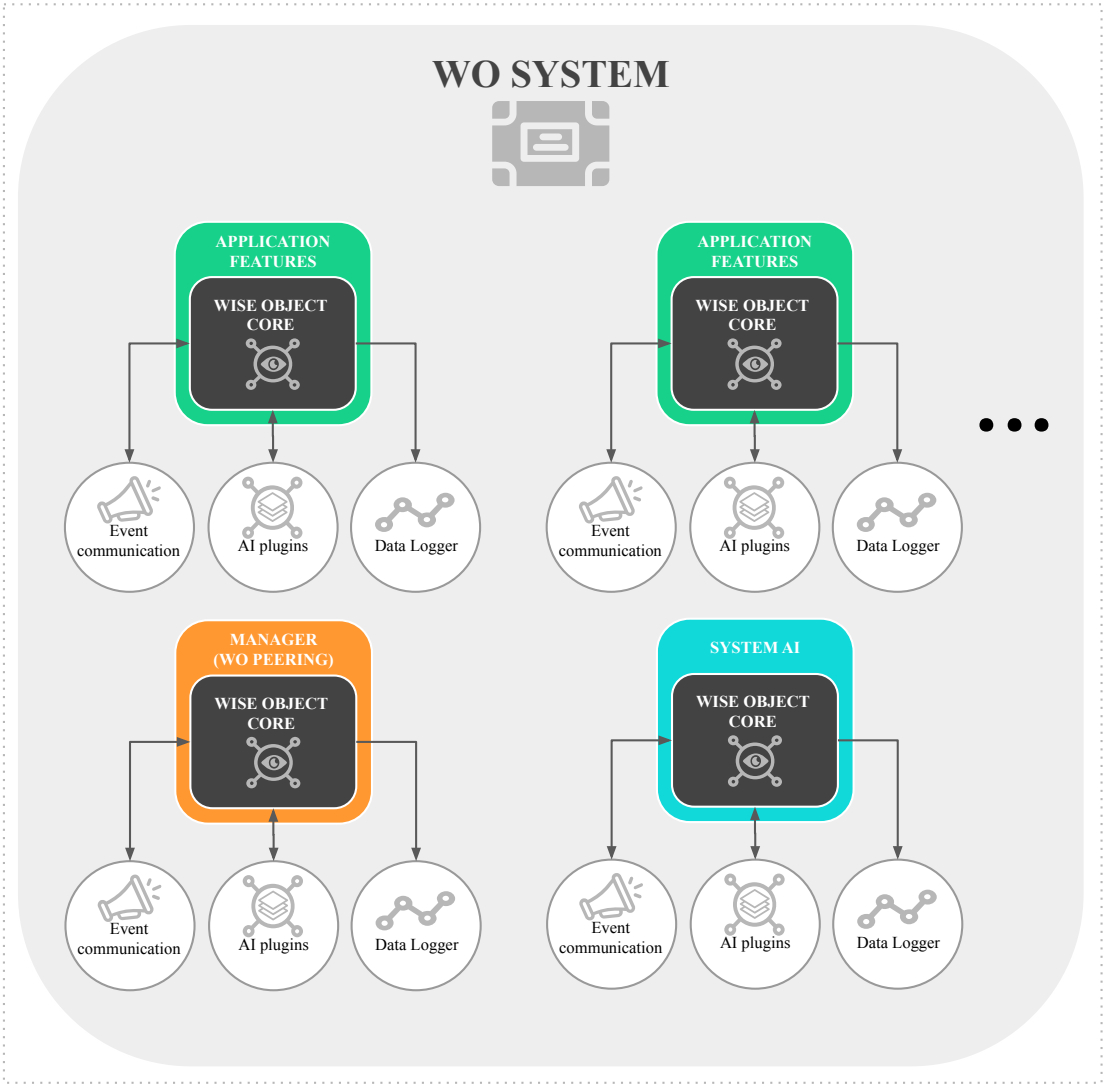


FIGURE 1.4 : Vue globale d'un système d'objets sages composé d'un ensemble de WO, d'un gestionnaire et d'un modèle d'IA [DAMV23a].

WS classique qui peut être composé d'un ensemble de WO instanciés, à savoir, des fonctionnalités d'application, des gestionnaires, des modèles d'IA. Ces WO contiennent le noyau où sont définis les mécanismes de base : l'introspection, la surveillance, l'analyse et la communication entre les instances des WO. Chaque WO a trois composants associés :

- Un support de communication d'événements pour publier son état au système (event communication).
- Un journal de données pour enregistrer toutes les interactions dans/avec l'objet (data logger).
- Un ou plusieurs plugins d'IA permettant au développeur d'ajouter des objets avec différentes politiques d'introspection, de surveillance, de décision et d'action (AI plugins).

En outre, deux types de WO supplémentaires sont définis dans le système :

- Le gestionnaire – manager – : le WS peut contenir un ou plusieurs gestionnaires (en orange) qui gèrent/assurent les actions/réactions de peering entre les WO, en utilisant par exemple les règles Event-Condition-Action (ECA).
- L'IA du système – system AI – : il gère l'ensemble des modèles de l'IA du système. Les modèles IA désignent toutes les activités nécessaires à la résolution des problèmes, à la supervision, à l'apprentissage et à l'analyse au niveau système, contrairement aux plugins qui se focalisent uniquement sur leur WO propre.

Vue du fonctionnement d'un WO

Super-états des WO : la figure 1.5 montre les trois phases différentes d'un WO (appelées dans la littérature *super-états*) [LAMV22], la phase d'*éveil*, de *rêve* et d'*Idle*[3]. Pendant la phase d'éveil, les WO répondent à différents services et demandes, à savoir l'exécution de méthodes appelées par d'autres objets/applications, surveillent leurs exécutions et leurs utilisations. La phase de *rêve* est encapsulée par la phase *Idle* en vue d'introspecter leurs comportements et d'analyser la connaissance sur leurs utilisations pour apprendre leurs comportements sans aucun effet sur les autres objets de l'application [AEV15], grâce à leur capacité à se déconnecter du reste du système. La capacité des WO à se déconnecter du monde réel est une caractéristique forte qui distingue les WS des autres systèmes auto-adaptatifs comme les systèmes multi-agent (MAS) [FM99]. En ce qui concerne la phase d'Idle, elle permet aux WO de n'effectuer aucune action et encapsule la phase de rêve, car les WO n'ont pas besoin de rêver tant qu'ils n'ont pas d'expérience passée, ainsi, sans déclencheur d'expériences passées, les WO seront toujours en phase d'Idle. Cette encapsulation pousse le WO à se rapprocher du comportement de l'humain, en effet, l'humain rêve à partir de son expérience passée et non de manière aléatoire [LAMV22]. Les trois phases sont créées pour garantir l'autonomie des WO, l'autonomie correspondant à la boucle de rétroaction MAPE-K

3. Phase d'inactivité.

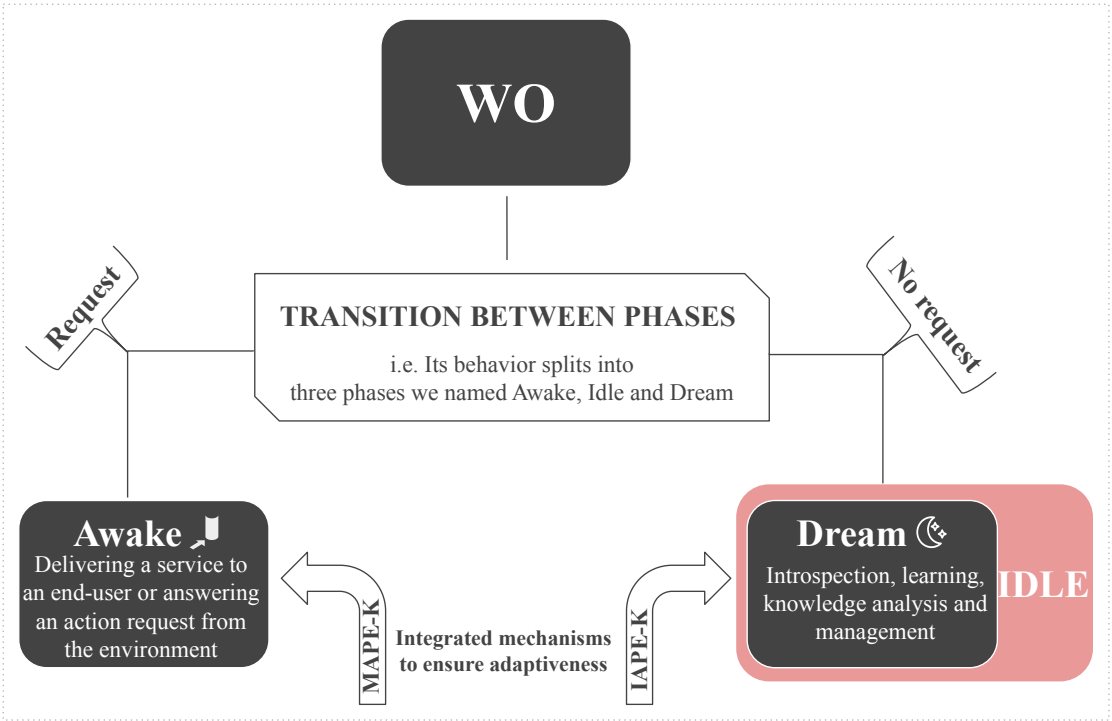


FIGURE 1.5 : Les trois phases (*super-états*) d'un WO.

(utilisée durant la phase d'éveil) et IAPE-K (utilisée durant la phase de rêve) dans lesquelles s'inscrivent ces trois phases.

La figure 1.6 est une formalisation de la figure 1.5. Le WO n'est pas autorisé

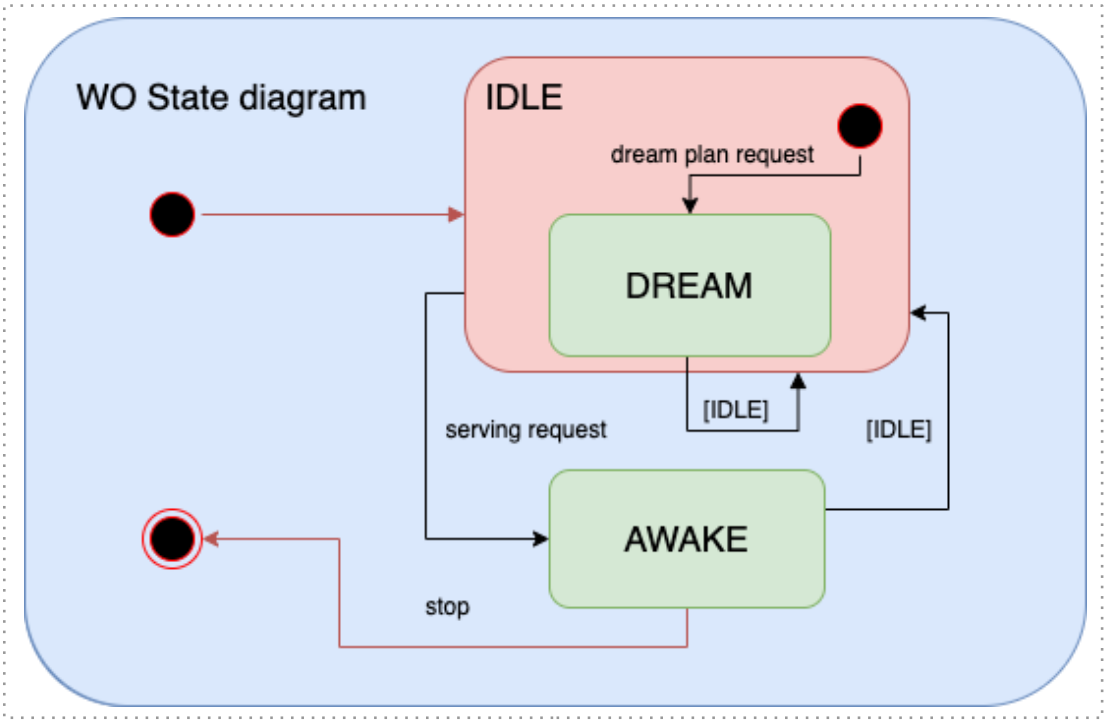


FIGURE 1.6 : Représentation formelle des trois phases (*super-états*) – awake, Idle et dream – d'un WO [LAMV22].

à rêver tant qu'il ne sait pas ce à quoi il doit rêver. Il peut rêver lorsqu'il est en état d'Idle, qu'il a une idée sur son utilisation et qu'il reçoit une demande de plan de rêve. Une fois le rêve terminé, le WO retourne directement à l'état d'Idle [LAMV22].

Conformité à la référence MAPE-K : tout système est considéré comme autonome s'il met en œuvre de manière directe ou indirecte, le modèle de boucle de rétroaction MAPE-K [BDMSG+09]. La boucle de rétroaction présente plusieurs avantages dont la séparation entre la surveillance, l'analyse, la planification, l'exécution et les connaissances utilisées pour réaliser l'*adaptation* et l'*auto-régulation*[4] – *self-regulating* – dans le système autonome. Dans la plupart des applications de MAPE-K, les systèmes auto-adaptatifs se basent sur un point de contrôle centra-

4. Dans le cas d'une entité (objet connecté, composant logiciel, application, etc.) qui possède cette capacité, elle est en mesure de contrôler son propre système sans avoir besoin d'un intervenant externe (définition adaptée de [Cam22]).

lisé, à l'exception des travaux présentés dans [IW12, WMA10, AHZ13], ce qui est loin du pattern de la séparation des préoccupations en matière d'adaptation – *separation of adaptation concerns (SoC)* – [Rus06]. Selon Arcaini et Riccobene [ARS15], ces travaux sont les premiers à présenter une approche formelle pour spécifier et vérifier les propriétés comportementales des systèmes auto-adaptatifs décentralisés, à travers les boucles de rétroaction MAPE-K. En outre, la boucle de rétroaction revêt un rôle très important dans les environnements dynamiques où les conditions opérationnelles sont très changeantes [ARS15, BDMSG+09, KC03], comme c'est le cas des WS.

- Surveillance et analyse : dans la mesure où l'objet sage ne doit pas être conscient des autres composants de la boucle MAPE-K, et ne doit se concentrer que sur le côté métier, une séparation est faite entre ces quatre composants « Surveiller-Analyser-Décider-Agir ». La figure 1.7 est une représentation implicite de ces composants : la surveillance est incluse dans le WO lui-même et représentée par la classe « WO ». L'analyse est représentée par la classe « Analyser ». Le planificateur a pour mission de planifier une décision, il est également un analyseur avec des spécifications appropriées, par conséquent, il utilise les résultats d'autres analyseurs ; par exemple, un premier analyseur transforme et organise les données, un deuxième utilise le résultat pour générer un graphe de connaissances numériques et un troisième ajoute de la sémantique à ce graphe, l'objet de ma contribution (Partie II). Cela pousse aussi les composants vers l'interopérabilité. Cette séparation est faite grâce à ces deux composants « la brique mémoire » et « le système à événements ». Le fonctionnement de ces deux composants est détaillé dans l'élément (connaissance) suivant [LAMV22].

- Connaissance : en effet, le WO a pour objectif de proxifier un objet existant (physique ou logique) du système pour lui ajouter de la sagesse (figure 1.8). Cette sagesse va permettre à l'objet proxifié d'utiliser ses connaissances pour prendre de « bonnes » décisions. Les connaissances utilisées sont envoyées à la mémoire qui écoute le WO. Cette connaissance est générée à l'aide de générateurs de connaissances – Knowledge Generator – à savoir, un WO est un générateur de connaissance vu qu'il a une idée sur son comportement et ses usages. Sans oublier le gestionnaire d'événements sage – Wise Event Manager – qui est dédié au stockage de ces connaissances en fonction de l'état « éveil et rêve » du WO (figure 1.7).

- Planificateur : afin d'assurer la séparation des préoccupations, le planificateur des WO est une entité indépendante – Standalone –. Le planificateur a deux rôles correspondant à deux phases (éveil et rêve) différentes du WO.

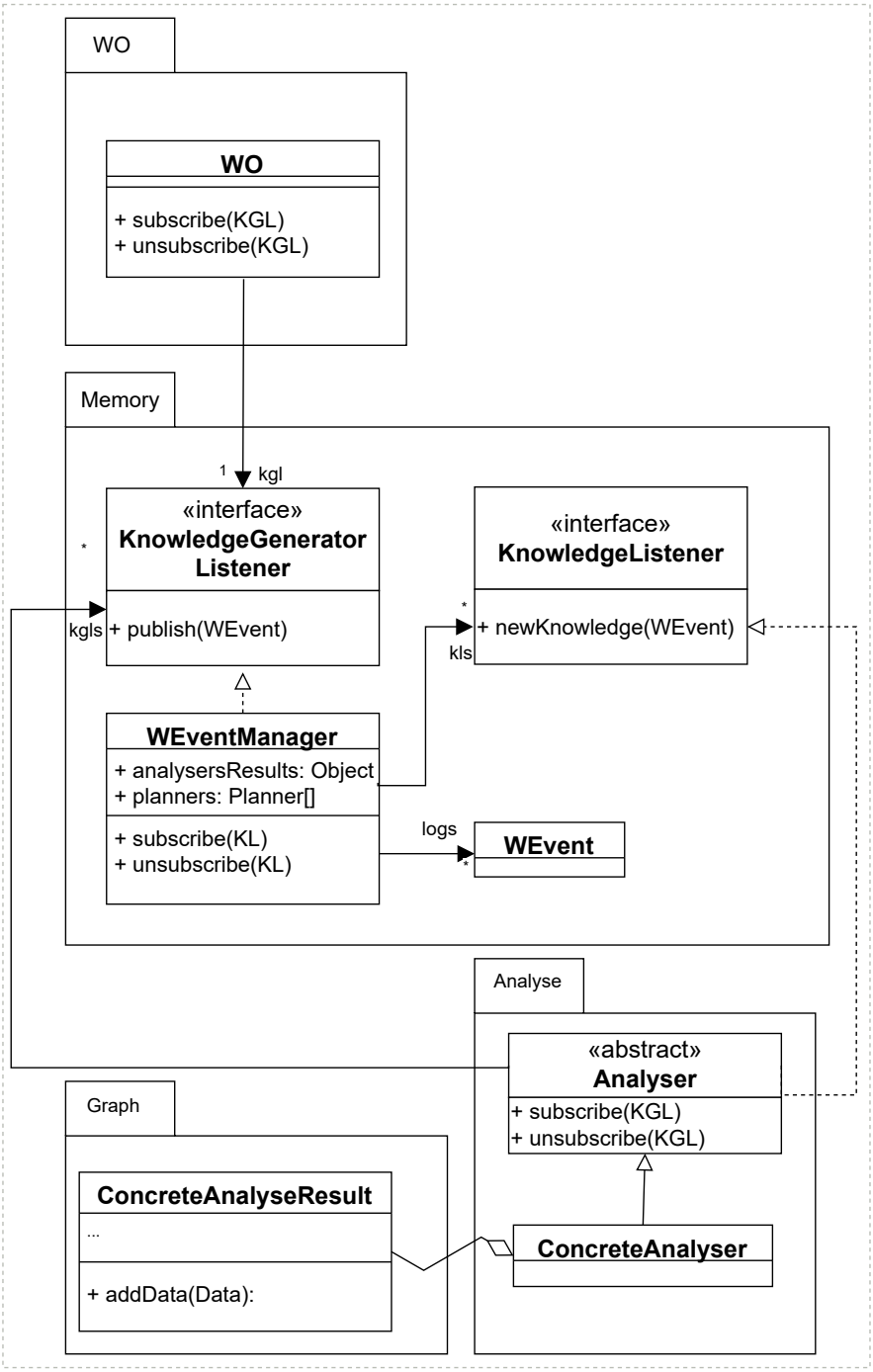


FIGURE 1.7 : Modèle statique de classes UML centré mémoire, illustrant la gestion de la surveillance, l'analyse et la connaissance [LAMV22].

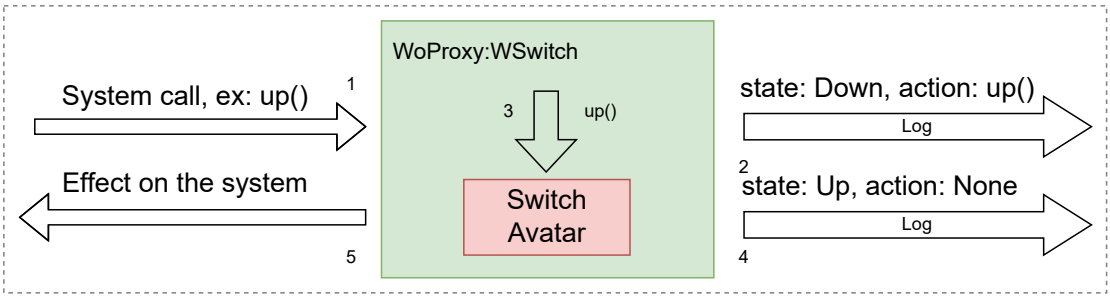


FIGURE 1.8 : Exemple d'une version sage d'un interrupteur. L'interrupteur a 2 états Up/Down et 2 fonctions up()/down() changeant l'état métier de l'objet. WO Proxy intercepte les appels (up() ou down()) (1) et les surveille (2). Il demande à l'objet d'agir (3). Lorsque l'état change, il le surveille à nouveau (4). Il renvoie la réponse en fonction de l'appel de fonction (5) [LAMV22].

(a) Au cours de la phase d'éveil, le planificateur examine le résultat de l'analyse effectuée par les analyseurs, afin de décider si une action sera planifiée pour modifier la requête d'origine ou non. (b) Durant la phase de rêve, le planificateur se sert de l'analyseur des paramètres [5] afin de déterminer les bons paramètres pour découvrir de nouveaux états de l'objet. Comme précisé dans la figure 1.9, le planificateur écoute des connaissances, celles générées par les analyseurs, s'appuie sur les politiques – Policy – des WO, et génère un plan d'action le plus approprié. Rappelons que les politiques sont une configuration de haut niveau [KC03] en fonction des différents environnements [LAMV22].

- Exécuteur : dans l'architecture des WO, l'exécuteur a une responsabilité supplémentaire qui n'est présentée dans aucun article à ce jour, selon [LAMV22], il doit appeler, modifier les appels faits pour l'objet selon les états (éveil/-rêve) du WO (voir la partie Executor figure 1.10). Puisque l'exécuteur est un écouteur – listener – du planificateur (figure 1.11), il analyse les plans créés en vue de les attribuer au bon exécuteur [LAMV22] d'éveil – Reaction Executor – ou rêve – Dreamer –. De manière plus large, l'exécuteur est le porte-parole du planificateur, de sorte que le WO puisse comprendre ce qu'il doit exécuter.

En revenant à la figure 1.5, le côté rêve prend en compte la boucle de rétroaction IAPE-K. Cette dernière n'est qu'une adaptation technique de la boucle MAKE-K

5. Comme il n'est pas possible de tester des milliers de paramètres de méthodes, l'analyseur de paramètres se base sur l'historique d'utilisation des méthodes durant la phase d'éveil pour découvrir de nouveaux paramètres.

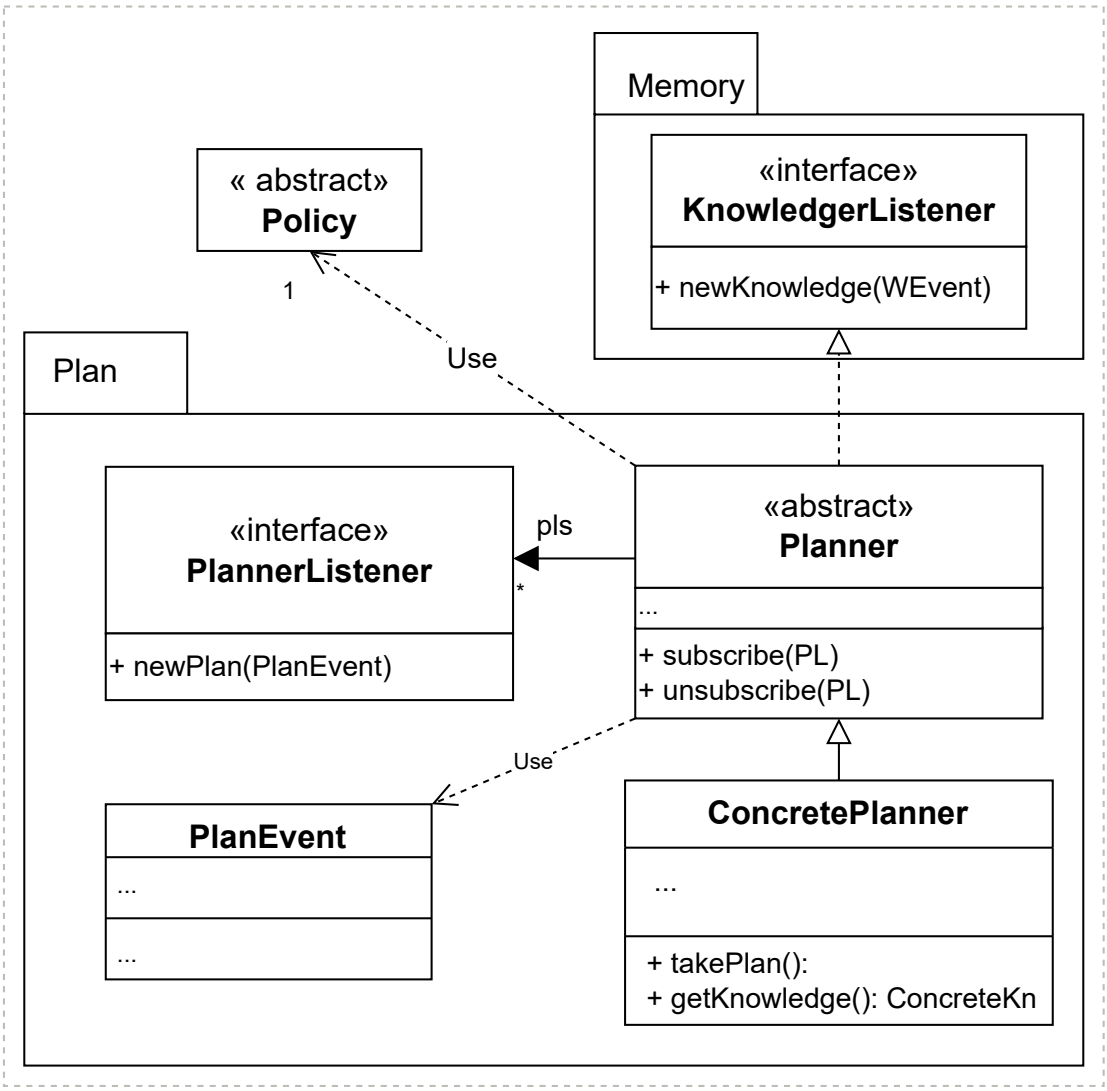


FIGURE 1.9 : Modèle statique de la connaissance et du plan d'exécution [LAMV22].

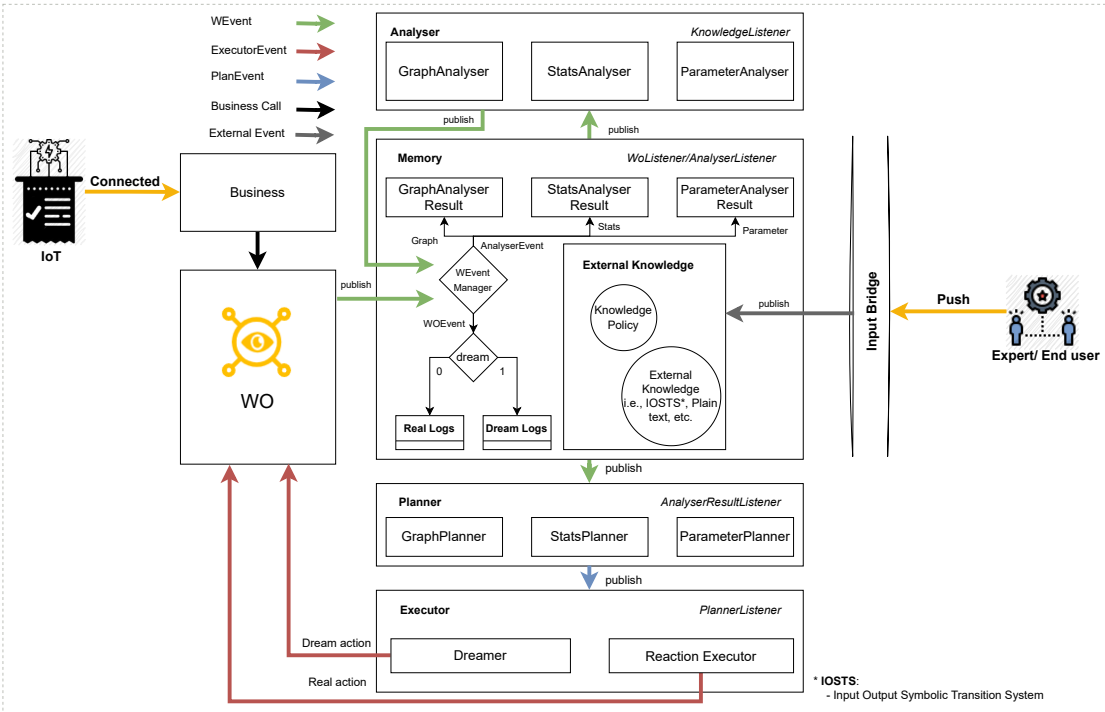


FIGURE 1.10 : Méta-modèle de l'architecture du WOF (prise de [LAMV22] avec l'ajout de la connaissance externe, l'expert, et la passerelle entre eux).

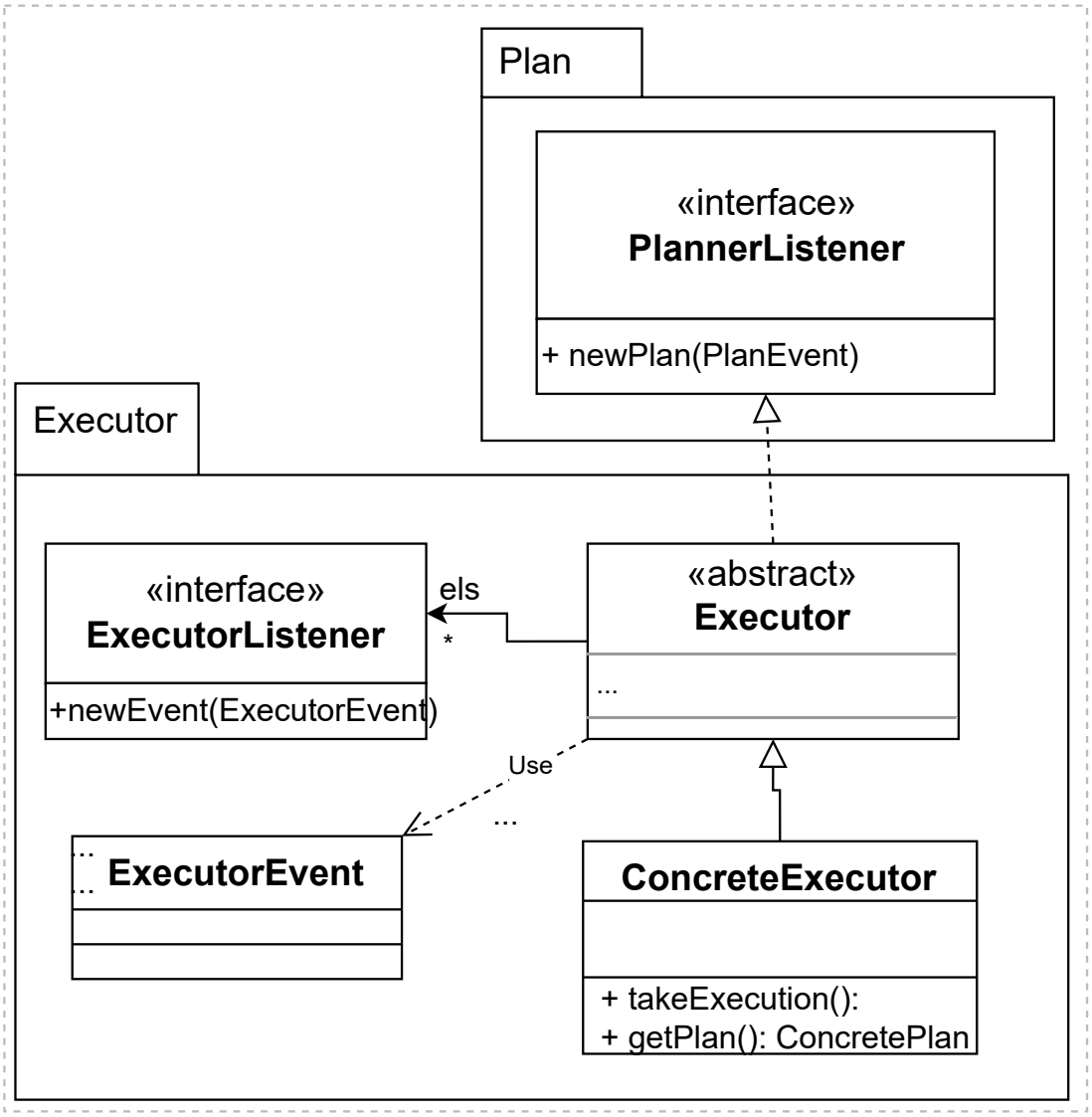


FIGURE 1.11 : Modèle statique de Plan et Exécution [LAMV22].

pour s'aligner sur les besoins du WO pendant la phase de rêve. Plutôt que de surveiller – Monitor – son (WO) activité, il utilise le mécanisme d'introspection – Introspect – pour découvrir son comportement et toute opération inhabituelle de son utilisation, d'où le « I » au lieu du « M ».

Construction progressive de la connaissance (Step-wise construction) : le cœur des boucles de rétroaction MAPE-K/IAPE-K est les connaissances, ces connaissances peuvent être représentées sous plusieurs formats. Dans le cas présent, je m'appuie sur une représentation des connaissances sous la forme d'un graphe de transition d'état (STG) qui est construit par un analyseur spécifique décrit dans la section des analyseurs 1.2.3. Plus précisément, ce graphe est construit par itérations – Step-wise construction – au cours du processus d'introspection, l'analyseur qui crée ce graphe est lancé lors de la phase de rêve, durant laquelle le WO découvre tous ses états [AV17]. Cette connaissance – le graphe – peut être analysée pour avoir de nouvelles connaissances ou une *sensation*, ce qui aide le WO à découvrir les comportements d'usage habituels et inhabituels . J'utilise le STG comme une forme de représentation de connaissances pour obtenir le comportement d'un WO en format numérique, afin de répondre à la problématique soulevée dans l'introduction.

Il est à noter qu'un WO à deux types de STG, un STG lié à ses capacités et un autre lié à son usage par ses utilisateurs. Le WO ne prend jamais en compte l'ensemble de ses états en raison d'une explosion combinatoire évidente. Seul un sous-graphe utile est stocké selon des mécanismes d'oubli des sous-parties inutiles. Ces mécanismes d'oubli font partie de la mémoire des WO et sortent du cadre de cette thèse. Le STG utilisé dans ma contribution et celui des capacités du WO et bien que ce dernier ne le stocke pas en globalité, je le considère comme tel dans la suite de mon mémoire.

Sensation

En ce qui concerne la connaissance de l'utilisation d'un WO, il faut distinguer deux situations : la sensation et l'adaptation du comportement. La sensation d'un WO est définie comme la distance entre son usage actuel et son usage courant, cette distance proche de 0 représente une sensation habituelle et un usage inhabituel quand elle s'éloigne de 0. Dans un premier test de mise en œuvre, un comportement habituel est considéré comme stationnaire, ce test est réalisé en utilisant la notion de stationnarité pour détecter les comportements inhabituels. Cette notion est une approche statistique utilisée pour définir l'usage habituel et inhabituel.

Un WO peut être stressé si l'un de ses services est utilisé plus fréquemment, ou à l'inverse, peut être ennuyé ou surpris si l'un de ses services est utilisé et que cela ne s'est jamais produit auparavant.

1.2.2 Framework d'objet sage

Une architecture en couches

Afin de regrouper tous les concepts utilisés pour créer un WO, un framework d'objet sage – Wise Object Framework (WOF) – est développé autour de ce concept (WO) [ABPV18]. Le WOF contient tout ce qui concerne la sagesse, la conscience et l'intelligence. Du point de vue du développement du système, les décisions de conception derrière le WOF sont principalement guidées par les trois exigences suivantes :

- La compréhensibilité, il est facile de comprendre la partie sage d'une application, puisque l'intrusion (le mélange) du code du framework dans le métier de l'application est évitée.
- La réutilisabilité et la généricité, afin d'être utilisé dans différents domaines d'application avec différentes stratégies.
- La flexibilité, le côté sage d'une application est facile à maintenir grâce aux patrons de conceptions utilisés comme le « décorateur ».

Les développeurs doivent pouvoir utiliser le framework avec le minimum de contraintes et d'intrusion dans le code source de celui-ci. Par conséquent, le WOF fournit la « conscience » et sépare la « logique d'intelligence » des services d'application qu'ils sont censés rendre. Comme le montre la figure 1.12, le WOF est conçu pour être utilisé dans une architecture en couches et offre la possibilité d'utiliser le concept des WO dans une application classique :

- La couche **« framework »** est composée de plusieurs blocs de construction interdépendants qui intègrent des mécanismes de base pour l'introspection, la surveillance, l'analyse et la communication entre les instances WO à travers le mécanisme de publication/abonnement en utilisant des règles d'action en fonction des événements – Event, Condition, Action (ECA) – établies par un gestionnaire – manager – (pour plus de détails [AV18]). La classe principale du framework est « WiseObject », à partir de laquelle le développeur du système peut spécialiser des classes pour quelles soient sages au niveau de l'application en utilisant seulement l'annotation *@WiseObject*, en procédant ainsi, un proxy est généré automatiquement pour envelopper l'objet original comme les classes « Interrupteur » et « Volet » dans le système domotique de la figure 1.12.
- La couche **« système logiciel »** – software system – contient le paquetage et les classes liés aux systèmes logiciels développés pour les utilisateurs finaux. Les classes représentant des objets peuvent être proxifiées en utilisant

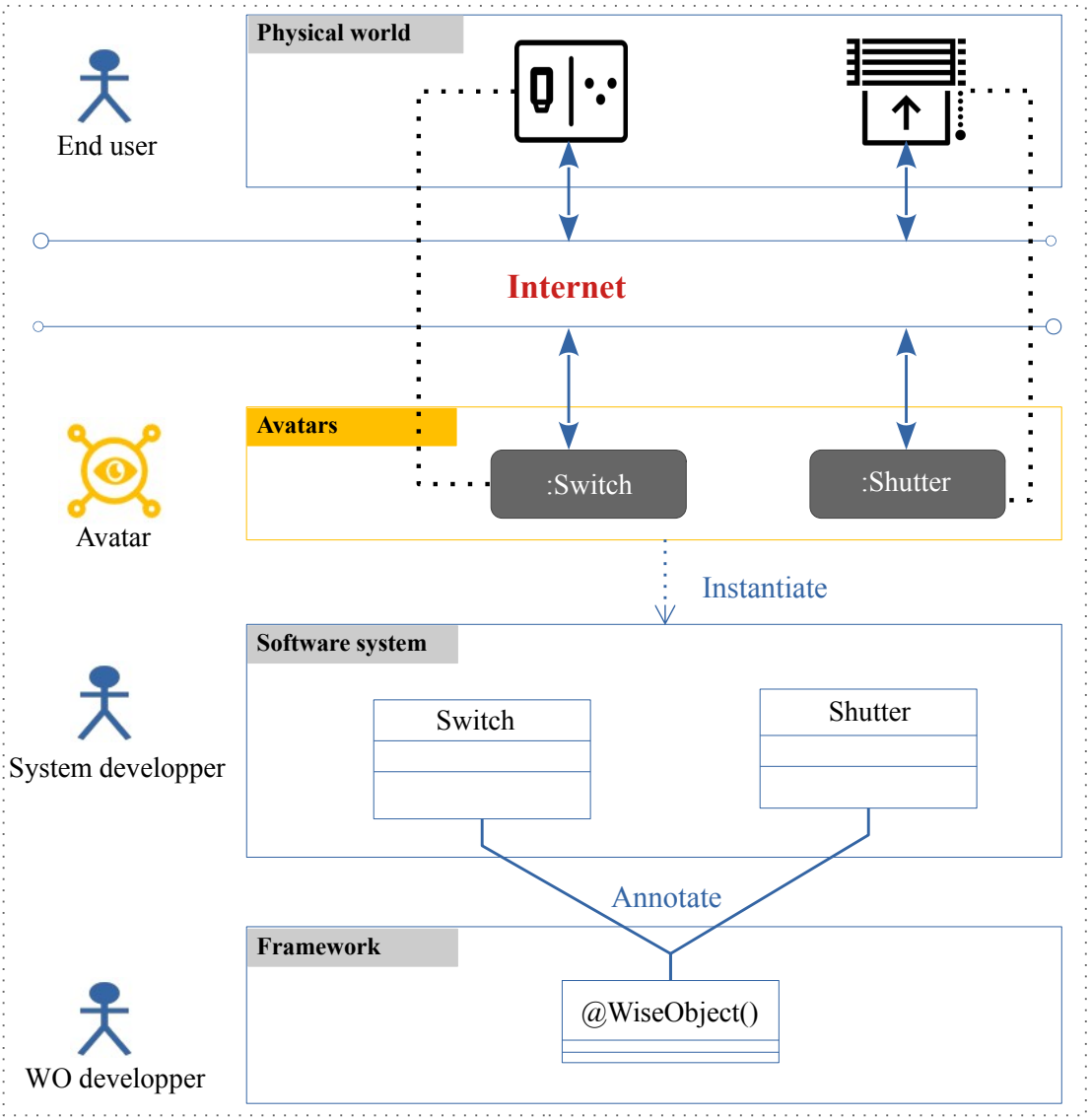


FIGURE 1.12 : Exemple d'architecture concrète du WOF en couches.

l'annotation **@WiseObject** de la couche framework pour qu'ils soient sages. Pour faire l'analogie avec la figure 1.10, le système logiciel est la partie métier – business –.

- Les **« systèmes logiciels instanciés »** – instantiated software system – ou avatars créés par le biais du proxy regroupant les systèmes logiciels applicatifs instanciés de la couche précédente. Les instances des classes liées aux applications sont les avatars d'objets physiques ou logiques – things –. Sachant qu'on peut utiliser plusieurs modèles d'implémentation.

L'approche conceptuelle proposée dans la figure 1.12 montre l'utilisation du WOF dans une architecture en couches où tous les éléments communiquent entre eux. La communication entre les WO dans le WOF est assurée par un système de communication d'événements – bus – utilisant le protocole de publication/abonnement. Ce protocole est très important dans le WOF, il permet aux WO de gérer les deux états, « éveil »et « rêve ». Pendant l'état d'éveil, le WO écoute et publie sur le système de communication afin de réagir aux actions demandées et pour informer le système de ses changements d'état, respectivement. Pendant l'état de rêve, le WO n'écoute le système de communication que pour répondre aux demandes externes. Comme un WO exécute lui-même ses méthodes (fonctionnalités) pour connaître son comportement, ses rêves ne doivent pas avoir d'effet sur l'ensemble du système. Ainsi, il ne publie pas son changement d'état pendant son état de rêve grâce à sa capacité à se déconnecter du monde réel [LAMV22].

Séparation des préoccupations dans le WOF

Comme mentionné en début de section, la compréhensibilité, la réutilisabilité et la flexibilité sont les principales exigences qui guident le développement du WOF. Ces exigences imposent avant tout le respect du socle de conception pour la production d'architectures logicielles, connu sous le nom de « SOLID », qui se résume en cinq principes :

- **Responsabilité unique – Single responsibility principle – :** Robert C. Martin[6], à l'origine de l'expression qui dit que chaque entité (classe) ne doit pas dépasser sa propre portée et doit se concentrer sur ses propres occupations sans lui ajouter plus d'une responsabilité [Mar17, Mar02].
- **Ouvert/fermé – Open/closed principle – :** chaque entité (fonction, classe, module, etc.) doit être fermée à la modification, mais ouverte à l'extension. À savoir que chaque entité testée et livrée ne doit pas être modifiée, mais seulement étendue [Mey88, Mar17].

6. Robert C. Martin est un ingénieur logiciel américain, connu pour le Manifeste Agile et les principes SOLID.

- **Substitution de Liskov – Liskov substitution principle – :** ce principe a été formulé par deux informaticiens (Barbara Liskov et Jeannette Wing) et dit que si $\phi(x)$ est une propriété démontrable pour tout objet x de type T, alors $\phi(y)$ est vraie pour tout objet y de type S tel que S est un sous-type de T [LW93]. Littéralement, si un objet A est un sous-type de B, alors tout objet de type B peut être remplacé par un objet de type A, tout en respectant les propriétés souhaitables du programme concerné.

- **Ségrégation des interfaces – Interface segregation principle – :** la philosophie de ce principe est qu'il faut toujours opter pour des interfaces multiples nommées interfaces de rôle afin de ne pas obliger les utilisateurs des classes à dépendre de fonctions qu'ils n'utilisent pas ou qu'ils n'utiliseront jamais. La raison est de maintenir un système à couplage faible [Mar17].

- **Inversion des dépendances – Dependency inversion principle – :** ce principe prend la forme d'un découplage avancé des modules dans un système et repose sur deux affirmations : a) quel que soit le niveau (haut/bas) des modèles, ceux-ci doivent dépendre d'abstractions, b) les abstractions ne doivent pas dépendre des détails, mais bien l'inverse [Mar08].

En respectant les cinq concepts de base présentés précédemment « SOLID », qui résument sommairement le fait que chaque élément du WOF a un but exclusif, le développement du WOF est poussé vers des variantes de séparation des préoccupations [Dij12], dans lequel un ensemble de limites a également été respecté, et englobant toute la variété des responsabilités en fonction des dimensions, telles que l'éveil, le rêve, les sources multiples de connaissance, etc. Ce type de séparation est connu sous le nom de « séparation horizontale multidimensionnelle » [LAMV22]. Par exemple, la séparation multidimensionnelle des préoccupations est très utile dans la gestion des analyseurs. Vu que chaque analyseur aura sûrement des dépendances différentes (statistiques, apprentissage profond, théorie des ensembles, etc.), mais communes, ils sont posés dans la même dimension des préoccupations. Cette séparation a en plus favorisé l'apparition de l'élément central, la mémoire, du WOF, dont il est question dans la sous-section qui suit ce paragraphe. Par contre, on n'entrera pas dans les détails des séparations des préoccupations, car ce concept dépasse le cadre de la thèse, pour plus d'information, voir [LAMV22, OT02, TOHS99].

La mémoire et le WOF

Comme évoqué plus haut dans la sous-section 1.2.1, le WO analyse ses connaissances, celles-ci sont stockées par *lui-même* dans une entité appelée : la mémoire, par le mécanisme de publication – publish – comme illustré dans la figure 1.10. Cette entité représente l'élément central du WOF dans la mesure où elle englobe

tous les types de connaissances qui se résument au moment de la rédaction de cette thèse en quatre types principaux :

- Configuration initiale – Policy – : dans tout système auto-adaptatif qui respecte la boucle de rétroaction MAPE-K, il existe une connaissance initiale qui va jouer le rôle de *repère* dans ce système, elle est souvent appelée en anglais « policy » ou « initial configuration » [KC03]. Sachant que ces connaissances initiales sont établies par des utilisateurs finaux ou des experts.

- Connaissance comportementale – Knowledge log – : dans notre cas, la connaissance comportementale est la première connaissance générée par le WO et sauvegardée en mémoire sous forme de journaux – logs –. Ces journaux contiennent l'historique d'utilisation du WO et les journaux créés lors de l'introspection, respectivement sauvegardés dans les deux entités « Real logs » et « Dream logs » comme le montre la figure 1.10. Sachant que le mélange des deux types de journaux est une autre question qui entre dans le cadre de la *fusion de la connaissance* et sort du cadre de cette thèse.

- Connaissances accumulées – Output/Result knowledge – : une fois que le système a acquis les connaissances comportementales, ces connaissances peuvent être analysées par des analyseurs en fonction des besoins du WO, la figure 1.10 montre l'exemple de trois analyseurs génériques : GraphAnalyser, StatisticsAnalyser et ParametersAnalyser, dans le cas de cette thèse, un STGAnalyser et un STGAnalyserResult. Ces analyseurs ont un résultat [7], le STG est le résultat – STGAnalyserResult – de l'analyseur de la connaissance comportementale – STGAnalyser –, ce résultat est une nouvelle connaissance qui peut être utilisée, par exemple, par un planificateur pour générer un plan d'action ou par un autre analyseur pour générer de nouvelles connaissances.

- Connaissance externe – External Knowledge – : cette connaissance peut être fournie par un expert, un utilisateur final ou un objet connecté.

Comme décrit auparavant, chaque interaction avec le WO provoque un événement pour stocker la connaissance – logs – dans la mémoire, qui, à son tour, déclenche un autre événement, qui peut permettre aux analyseurs d'analyser cette connaissance. Ces analyseurs sont créés par différents types d'experts (développeurs, expert en science des données, mathématiciens, etc.). Dans la section qui suit, la base de ces analyseurs est abordée avec une illustration à travers un exemple dans la section 1.2.3

7. Au moment de la rédaction de cette thèse, les analyseurs peuvent avoir un seul resultat – output –

Types de stratégies de mémoire pour un WO

Pour connaître et distinguer les comportements habituels des inhabituels, il faut analyser le passé, ce qui implique de faire appel à la notion de stratégie de mémoire. Plusieurs paramètres sont à prendre en compte lors de la modélisation de la mémoire, l'événement e, l'action effectuée a, sa source s_e et le moment t_e où il s'est produit, ainsi que la fonction poids f et l'âge $\Delta_{t_e} = now - t_e$, où now est l'instant présent. L'importance d'un événement est donnée par la fonction $fs(\Delta_t)$, qui définit le poids d'un événement passé. Ainsi, trois types de mémoire sont présentés, définis comme suit [MVM20] :

- S_{IT} signifie que tous les événements ont le même poids (qu'ils soient récents ou très anciens).
- S_{FT} signifie que la notion de fenêtre temporelle – time window – est utilisée, qui ne prend en compte que les événements récents (le terme « récent » dépendant de la taille de la fenêtre).
- S_{EE} signifie qu'un modèle de mémoire évanescente est utilisé ; le poids d'un événement diminuant avec le temps.

Après avoir analysé les connaissances, selon l'un des analyseurs qui utilise une stratégie de mémoire, une autre connaissance – output – sera générée et sauvegardée dans l'entité mémoire (figure 1.10). Cette nouvelle connaissance sera utilisée, par exemple, par un autre analyseur basé sur la notion de stationnarité pour détecter l'inhabituel/habituel dans le comportement d'un WO.

L'internet des objets et le WOF

Comme le montre la figure 1.12, le « monde physique » contient tous les appareils connectés au système via Internet ou un protocole [Sar12b, Chapter 1] de communication tel que ZigBee [Alla], Z-Wave [Allb] ou tout autre système moderne de communication. Un objet WIoT [ABPV18] est défini comme une paire composée d'une entité physique (ampoule, télévision, volet roulant, etc.) et d'un objet logique (avatar, WO).

La figure 1.13 illustre l'interconnexion respective entre les deux entités logiques et leurs objets physiques (capteur de présence et lampe de chevet). Les objets logiques des objets physiques sont appelés « avatars ». (L.Event, L.Action) et (P.Event, P.Action) sont des messages envoyés de/à l'avatar et à/par l'objet physique sur le bus respectivement. Par ailleurs, l'interconnexion se révèle fluide et facile à mettre en œuvre grâce à la séparation des préoccupations qui fait l'objet de la prochaine sous-section.

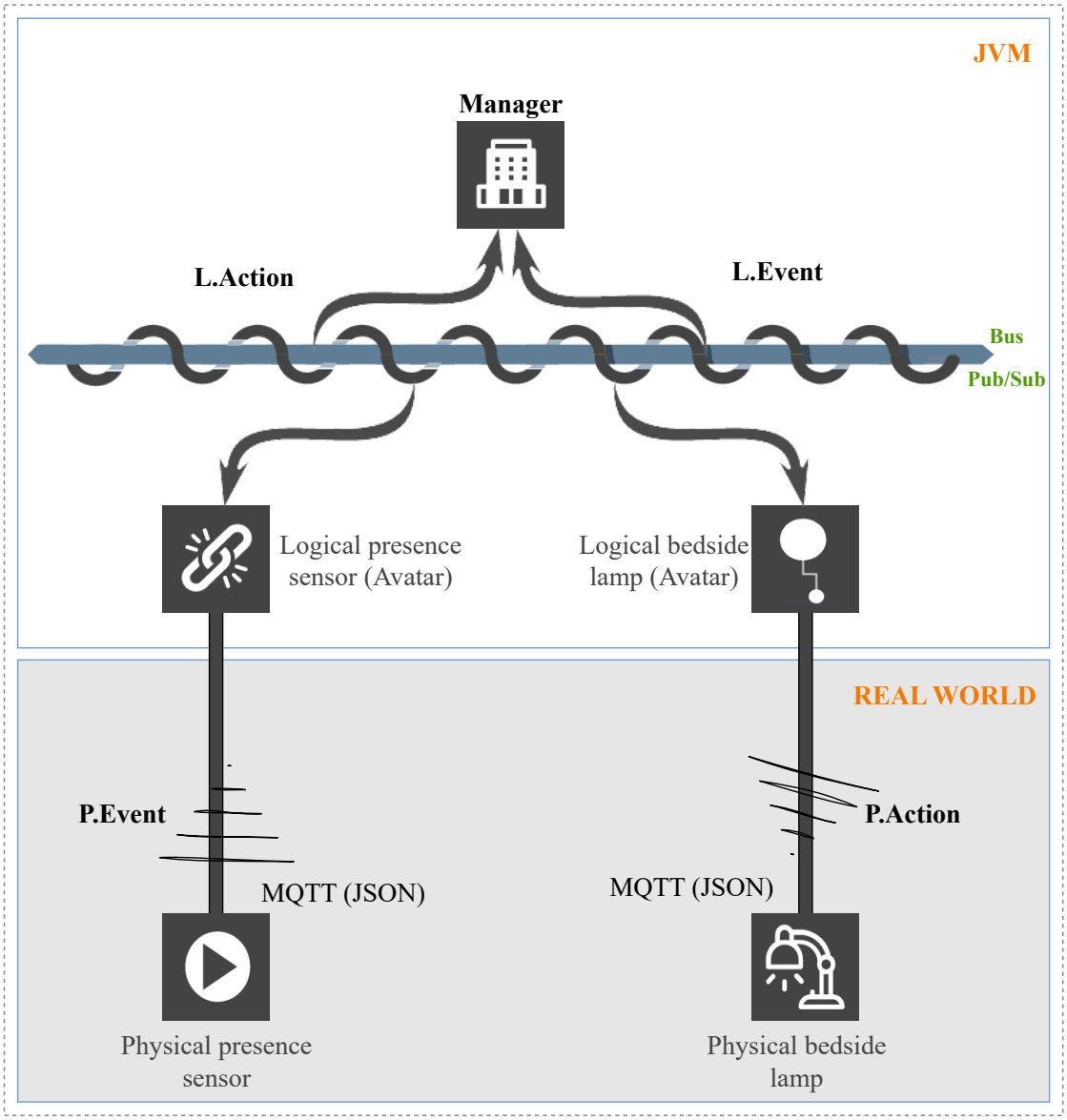


FIGURE 1.13 : Flux de communications entre le gestionnaire, les objets logiques et leurs objets physiques.

Lorsqu'un objet connecté, tel qu'un volet roulant, est connecté au WOF, il a deux possibilités : (a) son avatar correspondant, à savoir, une instance de la classe Shutter est automatiquement instanciée, cette paire formant alors un nouvel objet WIoT (voir, figure 1.12). Cela signifie que la classe avatar de l'objet existe, et comme il n'est pas souhaitable et même pas pertinent de fournir à tous les éléments du système la capacité d'apprendre et d'analyser, la création d'un WO générique est prévue dans des travaux futurs sans la capacité d'introspection. (b) Son avatar n'existe pas dans le WOF, le développeur doit donc créer une classe sage portant le nom de cet objet et l'annoter avec l'annotation « @WiseObject() ».

1.2.3 Exemples d'analyseurs de connaissances

Les analyseurs de connaissances peuvent être basés sur différents types de modèles, à savoir, statistique, apprentissage profond, construction par étape (le cas d'un STG), etc., et sur une stratégie de mémoire pour produire en sortie de nouvelles connaissances qui seront stockées dans la mémoire du WO. Par exemple, un analyseur utilise les logs pour générer une nouvelle connaissance : un STG. Un second analyseur utilise les logs et le STG, pour définir un graphe de Markov. Un troisième analyseur utilise ce graphe et l'action en cours pour déterminer sa distance à l'habitude. Les analyseurs ont un lien fort avec la mémoire, vu que cette dernière contient les informations – la connaissance – dont ils ont besoin pour fonctionner, et stocke le résultat de leurs analyses.

Cette section décrit deux types d'analyseur présent dans le WOF au moment de la rédaction de cette thèse, le premier est basé sur la notion de stationnarité au sens faible pour analyser le comportement d'une entité, cette notion sera combinée avec différents types de stratégies (section 1.2.2) pour analyser les événements passés, chaque stratégie étant associée à un analyseur. Le deuxième est un analyseur de graphes – STGAanalyser – qui transforme le comportement d'une entité en un graphe.

Stationnarité au sens faible pour un WO

La notion de stationnarité au sens faible – Weak-Sense Stationarity (WSS) – a été choisie comme processus stochastique pour déterminer si le comportement change dans le temps ou non. Littéralement, un processus est stationnaire si et seulement si ses propriétés ne sont pas affectées par un changement dans le temps. Par conséquent, le même comportement est obtenu que l'on choisisse les variables t_i ou $t_j = t_i + k$ sachant que t est le temps et k un décalage sur la série temporelle.

Le premier analyseur implémenté pour tester le WOF est basé sur la notion WSS. L'étude de la stationnarité se concentre sur les occurrences de l'événement.

Un événement est l'appel d'une méthode à partir d'un état/configuration donné, à savoir l'exécution d'une transition du graphe de connaissances (STG). WSS permet de distinguer un comportement habituel d'un comportement inhabituel en étudiant le processus aléatoire continu $\{X_t\}$, sachant que $\{X_t\} = \{x_{t_i}, x_{t_{i+1}}, x_{t_{i+2}}, ..., x_{t_n}\}$. Donc $\{X_t\}$ est un processus aléatoire temporel stationnaire, si et seulement si, la moyenne et la covariance sont constantes indépendamment du temps (t). Pour les détails théoriques et formelles de la WSS, voir [Il18, VER21].

L'analyseur STG

L'analyseur STG analyse le comportement du WO en itérations et produit un graphe de connaissance/comportement appelé STG, ce processus est introduit dans la sous-section 1.2.1 (construction progressive de la connaissance). De manière plus précise, cet analyseur traite les logs acquis en phase d'éveil pour générer le STG. De plus, il génére des plans de rêve pour explorer des états du STG, non atteints en phase d'éveil. Ces derniers génèrent des logs de rêve qui sont utilisés par l'analyseur pour enrichir la STG.

1.3 Conclusion

La revue de la littérature a permis de présenter le concept principal du WO, le WOF et l'avantage de cette structure, ainsi que la communication entre le monde physique et le monde logique. Jusqu'à présent, le concept du WOF a quatre parties fortes :

- le WOF donne la capacité à tout WO à se connaître par lui-même,
- le WOF offre à tout WO la capacité de se déconnecter du monde réel,
- le WOF peut proxifier [LAMV22] toute entité (logicielle, objet connecté, composant, sous-système, etc.) capable de communiquer son état en utilisant l'architecture et le protocole de communication définis dans 1.2.2,
- le WOF utilise une architecture en modules, qui lui permettra d'ajouter des analyseurs selon le besoin de l'entreprise, des experts ou simplement de l'application.

Le point clé du WOF et qui le distingue des systèmes multi-agents [FM99] est que ce cadre donne au WO la capacité d'apprendre sur lui-même par lui-même, en utilisant les notions d'introspection et de surveillance, et de générer des connaissances qui seront analysées par des analyseurs en fonction des différents types de mémoire pour les structurer, par exemple, sous la forme d'un STG. Un STG contient tout le comportement d'un WO et entre dans la catégorie des graphes comportementaux ; l'objet du chapitre suivant.

Sommaire

Chapitre 2

Modèles comportementaux complexes, définitions et illustrations

Les modèles comportementaux complexes sont des outils qui décrivent le fonctionnement d'un système afin de savoir tout ce qui se passe en arrière-plan (son fonctionnement interne). Ces modèles sont utilisés pour simplifier des problèmes complexes et se fondent sur une théorie purement mathématique et formelle, découlant de la théorie des graphes initiée par le mathématicien et physicien suisse Leonhard Euler au XVIIIe siècle face à la fameuse problématique des sept ponts de Königsberg [HMR06].

Ce chapitre aborde l'origine de la théorie des graphes, leurs types, leurs fonctionnements et leurs définitions formelles. Il est un élément important pour la compréhension des chapitres suivants.

2.1 Historique

Le premier article traitant une problématique à l'aide d'une structure qui sera plus tard appelée « graphe » a été présenté à l'académie des sciences de Saint-Pétersbourg par le scientifique Leonhard Euler le 26 août 1735 [NER86, HMR06]. La problématique est nommée : les sept ponts de Königsberg (figure 2.1). Le même auteur publiera l'année suivante l'article « La solution d'un problème relatif à la géométrie de la position » [Eul36]. À la suite d'Euler, de nombreux grands scientifiques ont suivi la voie en formalisant leurs problèmes sous forme de graphes comme le fit le mathématicien français Alexandre-Théophile Vandermonde avec la problématique des chevaliers [Bro17, HW96] – knight problem –. Enfin, la naissance de la topologie en tant que branche à part entière des mathématiques est due à la généralisation faite par Augustin-Louis Cauchy [Cau09] et Simon Antoine Jean L'Huilier [Cau13]. En 1936 et 1969, le mathématicien hongrois Dénes Kőnig et le mathématicien américain Frank Harary écrivent les deux premiers livres sur la théorie des graphes [Tut01] et ouvriront la voie à l'utilisation de ces derniers dans différents domaines scientifiques tels que (liste non exhaustive) : chimie, physique, électricité, sciences sociales et l'informatique dans tous ses sous-domaines.

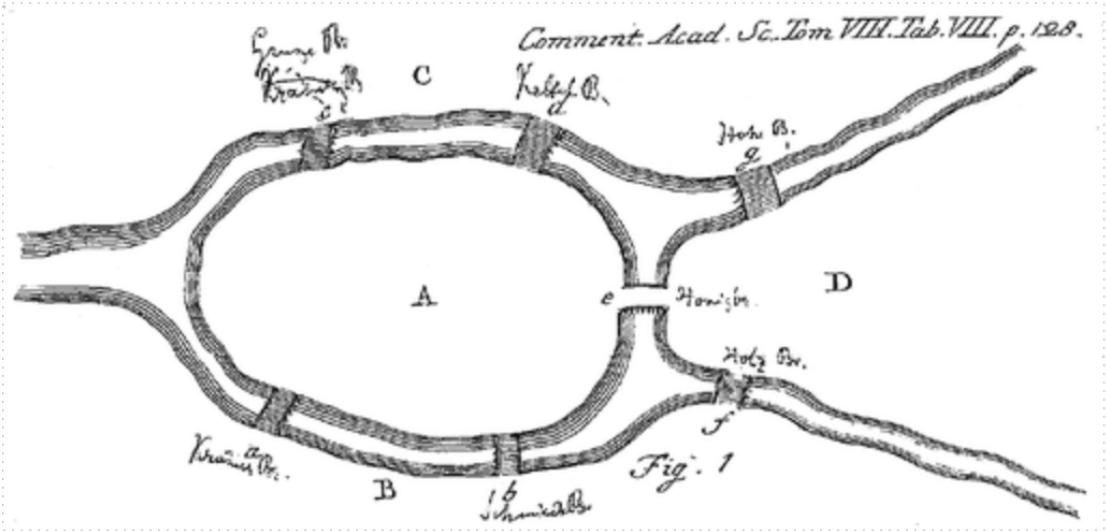


FIGURE 2.1 : Le pont Königsberg sous forme de graphe dessiné par Leonhard Eurler [Eul36]

Depuis de nombreuses années, plusieurs types de graphe (graphes non orientés, graphes orientés, graphes bipartis [Cha77], graphes étiquetés [Gal00], graphes d'Hadamard, etc.) sont utilisés dans de nombreux domaines pour représenter des problèmes complexes de manière descriptive (par exemple, les cartes, les relations

entre les profils de personnes, les transports publics, la biologie moléculaire, la recherche de voies évolutives conservées, l'alignement protéine-protéine, etc.) à des fins diverses comme : l'analyse, l'exploitation, la modélisation des connaissances, la détection de modèles, etc. Bien qu'elle ait été initiée au XVIIIe siècle comme mentionné dans le paragraphe précédent, la théorie des graphes reste un outil puissant pour le développement de systèmes à forte intensité logicielle et un moyen efficace de représenter des objets, comme cela a été prouvé dans [EF86]. Étant dans le domaine de la science de données en informatique, je me concentre sur certains types de graphe avec une formulation rigoureuse, les STG et les IOSTS qui sont utilisés dans cette thèse.

2.2 Types de graphe

Dans le domaine de la science de l'informatique, les graphes sont utilisés dans de nombreuses applications, de la description du comportement des systèmes à la représentation d'objets pour les modèles de Machine Learning. Si cette utilisation est massive, c'est en raison de leur facilité à représenter des structures complexes de manière simple.

2.2.1 Définitions et notations de base

Definition 2.2.1 (Graphe) *:* Un graphe G est considéré comme un triplet (V, E, ψ) [Wil10] [Ben02], où :

- $V = \{v_1, v_2, \ldots, v_n\}$ est l'ensemble des sommets du graphe G dont les éléments sont des nœuds.
- $E = \{e_1, e_2, \ldots, e_m\}$ est l'ensemble des arêtes du graphe G.
- ψ est appelée la relation d'adjacence définie par $\psi \colon E \to V \times V$ qui définit l'association entre chaque arête et les paires de nœuds de G.

Généralement, l'ensemble des sommets (nœuds) V et l'ensemble des arêtes E du graphe G sont exprimés respectivement par $V(G)$ et $E(G)$. En outre, toute arête d'un graphe qui relie un nœud à lui-même est appelée boucle, formellement, $\psi \colon e \to u \times v \mid (u, v) \in V^2 \wedge u = v$. Notez que la figure géométrique d'un graphe peut être représentée sous une infinité de formes, elle n'est donc ni unique ni un critère de comparaison [Jp68].

De cette définition générale découlent les définitions de nombreux types de graphes. Dans la présente thèse, je mets l'accent sur deux types de graphes seulement : Graphe d'état de transition – State Transition Graph (STG) – et Systèmes

de transition symboliques d'entrée et de sortie – Input Output Symbolic Transition Systems (IOSTS) – qui sont présentés dans les sections 2.2.2 et 2.2.3.

2.2.2 Graphes STG

STG générique

Un STG générique est un graphe orienté où les sommets représentent les états d'un système de manière générale, et les transitions représentent les changements d'état déclenchés par des événements. Considérons un objet défini par son ensemble d'attributs A et son ensemble d'événements, sachant qu'un événement déclenche une ou plusieurs transactions T. En fonction de ces informations (A et T), la définition du STG est donnée dans la définition 2.2.2.

Definition 2.2.2 (STG) *:* Un STG est défini par le triplet $G(V, E, L)$, où V et E sont, respectivement, les ensembles de sommets et d'arêtes, et L un ensemble d'étiquettes [DAMV22b].

- V est l'ensemble des sommets, avec $|V| = n$ où chaque sommet représente un état unique de l'objet, et inversement, chaque état de l'objet est représenté par un sommet unique. Par conséquent, $v_i = v_j \Leftrightarrow i = j$ avec $v_i, v_j \in V$ et $i, j \in [0, n[$.

- E est l'ensemble des arêtes orientées où $\forall e \in E$, e est défini par le triplet $e = (v_i, v_j, t_k)$, tel que $v_i, v_j \in V$ et $t_k \in T$. Ce triplet est appelé une transition étiquetée par t_k. La survenance de l'événement t_k à partir de l'état v_i bascule l'objet dans l'état v_j, cette transition est nommée transition de sortie pour son origine v_i et transition d'entrée pour sa destination v_j.

- L représente l'ensemble des étiquettes des sommets, où toute étiquette $l_i \in L$ est associée à v_i. Un étiquette l_i est l'ensemble des paires $(att_j, value_{i,j})$ $\forall att_j \in A$, avec $value_{i,j}$ la valeur de att_j dans l'état v_i. $Dom(att_j)$ est le domaine de valeurs de att_j, à savoir, l'ensemble de $value_{i,j}$ pour tout i. Par définition, deux états v_i et v_j sont différents $v_i \neq v_j$, *ssi* $\exists att_k \in A$, tel que $valeur_{i,k} \neq valeur_{j,k}$. Inversement, si $\forall k \in [0, |A|[$ et $value_{i,k} = value_{j,k}$, alors les états v_i et v_j sont considérés comme identiques, c'est-à-dire que $v_i = v_j$, et donc $i = j$.

La figure 2.2 illustre un STG générique. Ce STG est défini par les attributs « att_1 et att_2 » ($A = \{att_1, att_2\}$) et six transitions « $t_1, t_2, \ldots, t_6$ » ($T = \{t_1, \ldots, t_6\}$). Les valeurs des attributs A changent quand l'un des évènements T se déclenchent, ce qui change l'état du système d'un v_i vers v_j.

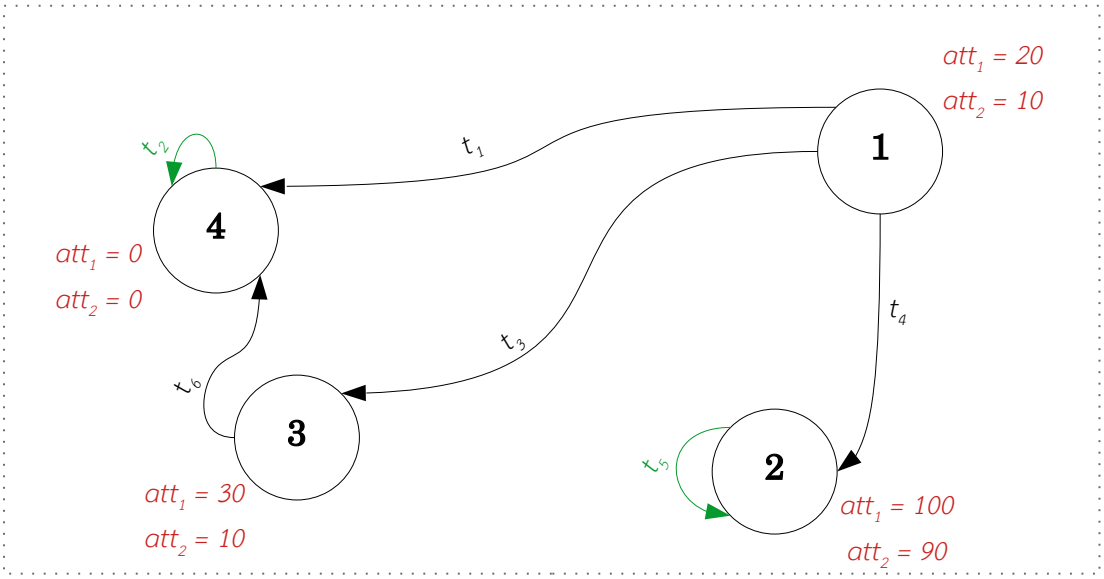


FIGURE 2.2 : STG générique d'un comportement abstrait.

Le STG d'un WO

Dans le contexte des WO, un STG est un graphe orienté où les sommets représentent les différents états d'un objet et les transitions représentent l'exécution de ses méthodes M. Ce qui transforme légèrement la définition des arêtes [DAMV23b] :

- E est défini par l'ensemble des triplets $e = (v_i, v_j, m_k)$, tel que $v_i, v_j \in V$ et $m_k \in M$. Ce triplet est appelé transition (c'est-à-dire, l'exécution de la méthode m_k) étiquetée par m_k. L'appel de la méthode m_k à partir de l'état v_i bascule l'objet dans l'état v_j.

La figure 2.3 illustre le STG d'une ampoule couleur connectée. Pour simplifier l'illustration, seules deux couleurs sont prises en compte et l'état éteint de l'ampoule n'est pas considéré. Ce STG est défini par l'attribut « specter » ($A = \{specter\}$) et deux méthodes « upFrequence » et « downFrequence » ($M = \{downFrequence(), upFrequence()\}$). Les méthodes « upFrequence » et « downFrequence » augmentent et diminuent le spectre de 40nm, respectivement.

Ce STG sera enrichi par des caractéristiques spéciales liées au concept des WO (définition 4.1.1), est présenté dans la partie II, et est utilisé pour répondre à la problématique évoquée dans l'introduction, voir le paragraphe « de la machine vers l'humain ».

L'objectif du STG généré par le WO est de lui donner la capacité de se connaître et de se manipuler « l'autonomie ». Mais, il ne lui permet pas de communiquer ses états d'une manière sémantique avec l'humain. La question est désormais de

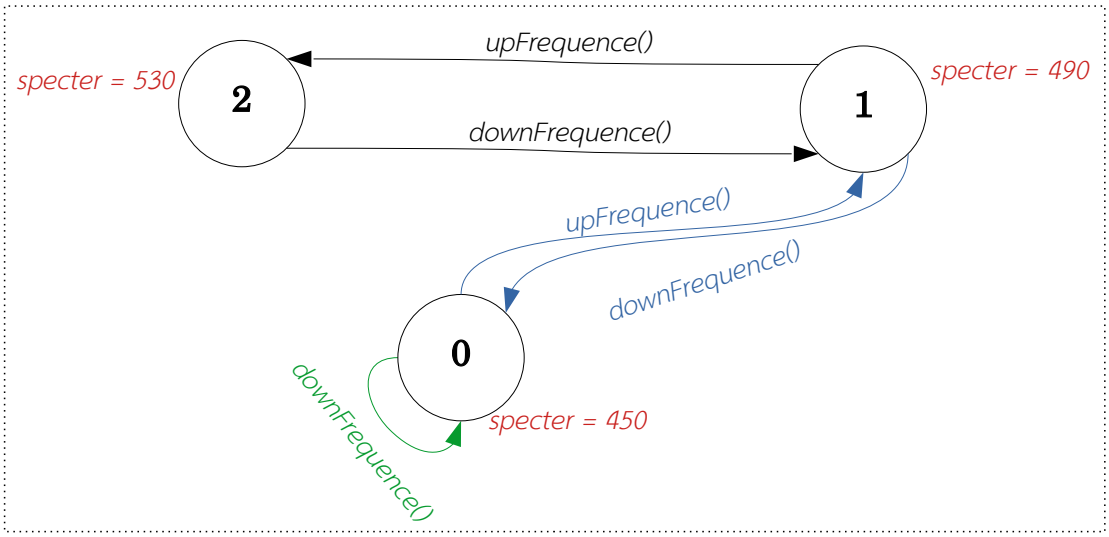


FIGURE 2.3 : STG du comportement d'une ampoule connectée.

savoir où le WO peut trouver de la sémantique pour enrichir son STG. La réponse est présentée dans la section qui suit.

2.2.3 Automates IOSTS

Un automate IOSTS est un graphe orienté dont les sommets, appelés localités, représentent différents états d'un système (dans le contexte de cette thèse, le système est un objet logiciel) et dont les arêtes sont des transitions. Les localités sont reliées par des transitions déclenchées par des actions. Un IOSTS permet de définir de manière finie un système de transitions d'états infini, contrairement à un STG qui définit tous les états du système de manière explicite. Dans la littérature, les IOSTS sont utilisés pour vérifier, tester et contrôler les systèmes. La vérification et le test sont des techniques formelles permettant de valider et de comparer deux vues d'un système, tandis que le contrôle est utilisé pour contraindre le comportement du système [CJMR07, RMT+04]. La définition d'un IOSTS donnée dans la définition 2.2.3, est tirée de [RMJ05, CJMR07] et notamment du cas d'utilisation donné dans [MSV12].

Definition 2.2.3 (IOSTS) *:* Un IOSTS est un tuple de six éléments $\langle D, \Theta, Q, q_0, \Sigma, T \rangle$ tel que :

- D est un ensemble fini de données typées composé de deux ensembles disjoints : les variables X et les paramètres des actions P. Soit un élément $d \in D$, $Dom(d)$ détermine le domaine de valeurs de d.

- Θ une condition initiale exprimée sous la forme d'un prédicat sur les variables X.

- Q est un ensemble fini non vide ($Q \neq \emptyset$) de localités avec $q_0 \in Q$ la localité initiale. Une localité q est un ensemble d'états tel que : $statesOf(q) \subseteq Dom(X)$, avec $Dom(X)$ le produit cartésien des domaines $Dom(x)$ de chaque $x \in X$ [DAMV23a] :
$$Dom(X) = \prod_{x \in X} Dom(x). \tag{2.1}$$
Il est important de noter qu'un état est défini par un ensemble unique de valeurs pour l'ensemble des variables.

- Σ est l'alphabet, un ensemble fini et non vide ($\Sigma \neq \emptyset$) d'actions. Il s'agit de l'union disjointe de l'ensemble $\Sigma^?$ des actions d'entrée, de l'ensemble $\Sigma^!$ des actions de sortie et de l'ensemble $\Sigma^\mathcal{T}$ des actions internes. Pour chaque action a dans Σ, sa signature $sig(a) = \langle p_1, \ldots, p_k \rangle | p_i \in P$ est un tuple de paramètres. La signature des actions internes est toujours un tuple vide.

- T est un ensemble fini de transitions, tel que chaque transition est un tuple de cinq éléments $t = \langle q_o, a, G, A, q_d \rangle$ défini par :

  - la localité $q_o \in Q$, l'origine de la transition,
  - l'action $a \in \Sigma$, l'action déclenchant la transition,
  - une expression booléenne G sur $X \cup Sig(a)$ liée aux variables et aux paramètres de l'action, appelée garde de transition. Les gardes de transition permettent de distinguer les transitions qui ont la même origine et la même action, mais des conditions disjointes à leur déclenchement,
  - une affectation A de l'ensemble des variables, de la forme $(x := A^x)_{x \in X}$ telle que pour chaque $x \in X$, A^x est une expression sur $X \cup Sig(a)$, qui définit le changement des valeurs des variables pendant la transition,
  - q_d est la localité destinataire de la transition.

Conformément à la définition 2.2.3, chaque variable possède un sous-domaine dans chaque localité. La fonction $dom(q, x)$ renvoie le domaine de définition de la variable $x \in X$ dans la localité $q \in Q$, donc $dom(q, x) \subseteq Dom(x)$. Par extension, $dom(q, X)$ est le produit cartésien des domaines de tous les x dans q [DAMV23a] :

$$\begin{gathered} dom(q, X) = \textstyle\prod_{x \in X} dom(q, x), \\ dom(q, X) \subseteq Dom(X). \end{gathered} \tag{2.2}$$

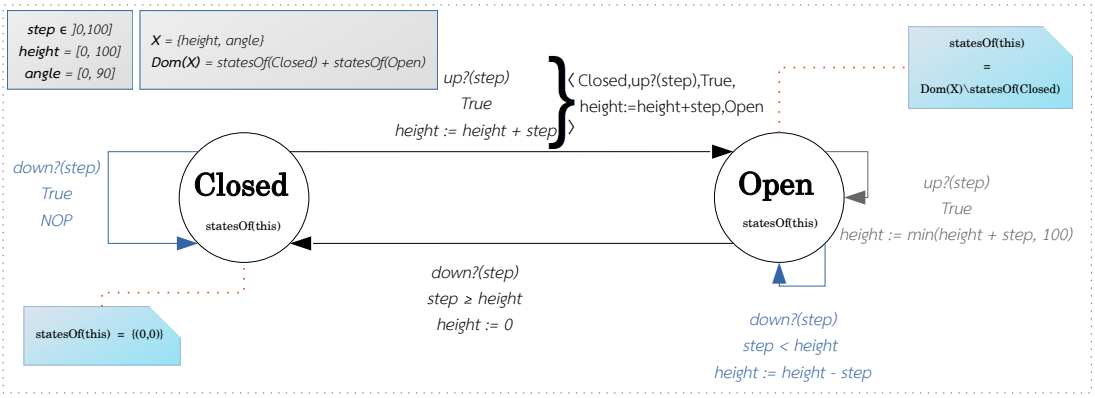


FIGURE 2.4 : Une représentation IOSTS d'un volet roulant.

Example 2.2.1 *:* La figure 2.4 illustre l'IOSTS donné par un développeur pour contrôler un volet roulant à lamelles orientables [DAMV23a]. Cet IOSTS indique que le volet roulant attend une entrée $up?/down? \in \Sigma^?$ portant le paramètre $step \in]0, 100]$, l'élévation relative pour accroître ou décroître le niveau $height$ du volet. En outre, les lames du volet roulant peuvent être tournées de 0 à 90 degrés pendant l'élévation. Ceci est exprimé par la variable $angle$. Précisons que le volet roulant utilisé dans cet exemple adapte automatiquement son $angle$, alors que l'élévation $height$ est comprise entre 0 et 100%. Ainsi, les utilisateurs finaux ne contrôlent que l'élévation. L'IOSTS contient deux localités :

- La localité où le volet est fermé (c'est-à-dire que les deux variables sont égales à 0, voir équation 2.3). Dans cette localité, si le système reçoit la commande $up?(step)$, la transition sera effectuée de la localité $Closed$ à la localité $Open$ en augmentant la valeur de la variable $height$ de $step$. Si le système reçoit l'action $down?(step)$, il n'effectuera aucune opération – no operation (NOP) –.

- La localité où le volet est ouvert (c'est-à-dire que l'une des variables est différente de 0, voir équation 2.4). Dans cette localité, si le système reçoit l'action $up?(step)$, la transition sera réflexive de $Open$ vers elle-même et calculera la valeur de la variable $height$ en exécutant cette affectation $height := min(height + step, 100)$, l'élévation du volet roulant ne peut pas être augmentée plus que le maximum de l'élévation. Si le volet roulant reçoit l'action $down?(step)$ et que l'action ferme le volet moins qu'il n'est ouvert $(step < height)$, $height$ est diminué de $step$, sinon la transition se fera de la localité $Open$ à la localité $Closed$ en attribuant 0 à la variable $height$.

Comme l'illustre la figure 2.4 et conformément à la définition 2.2.3, l'IOSTS se compose des éléments suivants :

- $Q = \{Closed, Open\}$, l'ensemble des localités.
- $X = \{height, angle\}$, l'ensemble des variables.
- $P = \{step\}$, l'ensemble des paramètres.
- $\Sigma = \{up?, down?\}$, l'ensemble des actions dont les signatures sont : $Sig(up?) = Sig(down?) = \langle step \rangle$.
- $Dom(height) = [0, 100]$, $Dom(angle) = [0, 90]$ et $Dom(step) =]0, 100]$ sont, respectivement, les domaines des variables *height*, *angle* and *step*.
- Conformément à l'équation 2.1, $Dom(X)$ est définie de la manière suivante :

$$Dom(X) = [0, 100] \times [0, 90].$$

- Les états de la localité « Closed », définis par l'équation 2.2, sont :

$$\begin{gathered} statesOf(Closed) = \{(0, 0)\}, \\ \{(0, 0)\} \subset Dom(X). \end{gathered} \tag{2.3}$$

- Les états de la localité « Open », définis par l'équation 2.2, sont :

$$\begin{gathered} statesOf(Open) = Dom(X) \backslash statesOf(Closed), \\ Dom(X) \backslash statesOf(Closed) \subset Dom(X). \end{gathered} \tag{2.4}$$

En conséquence :

$$statesOf(Closed) \cap statesOf(Open) = \emptyset. \tag{2.5}$$

En ce qui concerne la liste des transtions T, l'IOSTS illustré dans la figure 2.4 modélise un système d'états infini basé sur 5 transitions :

$$\begin{aligned} T = \quad \langle \quad & t_{Close-Open}, \\ & t_{Open-Close}, \\ & t^1_{Open-Open}, \\ & t^2_{Open-Open}, \\ & t_{Close-Close} \quad \rangle \end{aligned}$$

À savoir :

$$
\begin{aligned}
t_{Close-Open} = \ & \langle \ Open, up?(step), \\
& True, height := height + step, \\
& Open \rangle \\
t_{Open-Close} = \ & \langle \ Open, down?(step), \\
& step \geq height, height := 0, \\
& Close \rangle \\
t^1_{Open-Open} = \ & \langle \ Open, down?(step), \\
& step < height, height := height \\
& -step, Open \rangle, \\
t^2_{Open-Open} = \ & \langle \ Open, up?(step), \\
& True, min(height + step, 100), \\
& Open \rangle, \\
t_{Close-Close} = \ & \langle \ Close, down?(step), \\
& True, NOP, \\
& Close \rangle.
\end{aligned}
$$

Il est évident qu'il existe une infinité de chemins et d'états représentés par la variable *height* puisque son domaine est l'intervalle $[0, 100]$.

Les automates IOSTS sont un formalisme de référence classique pour les tests basés sur des modèles de systèmes réactifs [GLGRT06], ils fournissent une abstraction efficace des comportements de ces systèmes.

2.3 Conclusion

La revue de la littérature a permis de présenter les concepts de graphes génériques simples et de modèles comportementaux plus complexes dans leurs représentations formelles. Deux modèles sont considérés dans cette thèse :

- Les STG, pour représenter les connaissances liées aux comportements générées par les systèmes sages.
- Les automates IOSTS, pour représenter la modélisation conceptuelle donnée par les développeurs/experts.

Les STG sont des graphes générés automatiquement par les WO et représentent leur comportement, ce qui leur donne la capacité de se connaître eux-mêmes. Les IOSTS, quant à eux, sont donnés par les développeurs/experts pour décrire de manière formelle et sémantique le comportement d'un tel système. La connaissance de l'utilisation de ces derniers ne se limite pas à cela, et remonte un peu plus loin jusqu'à la vérification formelle, le contrôle et le test fonctionnel qui valident un système. En outre, la vérification/test d'un système ne se fait pas seulement à l'aide

de formalismes mathématiques tels que l'IOSTS, elle prend également d'autres formes plus simples, telle que l'évaluation automatisée des systèmes interactifs à l'aide d'ontologies développées sur la base de la « programmation pilotée par le comportement – Behaviour Driven Development – ». La simplicité réside dans l'utilisation de frameworks tels que PANDA, qui prennent en compte les scénarios « User Stories » des testeurs exprimés en langage naturel, afin d'effectuer ce type de vérification et de test d'un tel système [SHW17]. Dans le chapitre suivant, le concept d'ontologie est abordé.

Sommaire

Chapitre

3

Approche sémantique contextuelle - ontologie

L'utilisation des réseaux sémantiques pour la représentation des connaissances remonte à plusieurs siècles dans le domaine de la logique. Mais, sur le plan informatique, le chercheur en linguistique informatique Richard Hook Richens a été le premier à mettre en œuvre les réseaux sémantiques dans le domaine de la traduction en langue naturelle en 1956 [LR92]. Une trentaine d'années plus tard, les chercheurs ont commencé à se rendre compte que la représentation des connaissances – knowledge representation – combinée à des systèmes intelligents est une invention qui donnera un atout majeur à ces systèmes [Pow91], en particulier pour les systèmes sensibles au contexte [NPL+20].

Ce chapitre aborde la présentation du concept d'ontologie, ses définitions linguistique et formelle, ainsi que ses types avant de donner quelques exemples de son utilisation.

3.1 Présentation du concept d'ontologie

Le terme « Ontologie » a été inventé au début du XVIIe siècle. Selon [GG95], ce terme a plusieurs définitions dans différentes communautés. Aristote a traité ce sujet dans la Métaphysique et a défini l'ontologie comme la science de l'être, c'est-à-dire l'étude des attributs appartenant aux choses en raison de leur nature. En informatique, une ontologie est un type particulier d'objet d'information ou d'artefact informatique, et selon [Gru93] et [Gru95], pour les systèmes d'IA, ce qui « existe » est ce qui peut être représenté.

3.1.1 Définition linguistique

En 1993, Gruber Thomas a défini à l'origine la notion d'ontologie comme une « spécification explicite d'une conceptualisation » [Gru93]. En 1997, Willem Borst a défini une ontologie comme une « spécification formelle d'une conceptualisation partagée » [BB97]. Sur la base de la fusion des deux définitions [SBF98], une ontologie peut être définie comme une structure représentative (spécification) d'un domaine donné (indépendamment de la technologie utilisée). Elle est fondée sur un ensemble de connaissances représentées de manière formelle, et basée sur une conceptualisation explicite (non ambiguë) de ce domaine. Pour mieux comprendre cette définition, il est nécessaire de commencer par clarifier certains termes :

- Structure représentative (spécification), signifie que les sujets de l'ontologie s'agencent entre eux en un système ordonné de règles, décrivant à la fois les sujets (généralement les objets du monde réel) et les relations (prédicats) qu'elles entretiennent entre eux.
- Conceptualisation explicite, signifie que les concepts, les individus, les attributs et leurs relations sont explicitement définis [SBF98, Nga19]. Sur un plan plus général, la conceptualisation est une vue abstraite, simplifiée et suit une structure de base du domaine représenté [GOS09].
- Formelle, signifie que la spécification doit être explicite et sans ambiguïté, de sorte qu'elle soit lisible par une machine [SBF98, Nga19].

La figure 3.1 illustre une vue globale de l'ontologie d'une maison connectée, appelée OntoDomus[1], créée au sein du laboratoire DOMUS de l'université de Sherbrooke. Elle est utilisée au chapitre 6 dans le cadre de ma collaboration avec le laboratoire DOMUS, afin de traiter la problématique décrite dans l'introduction « de l'humain vers la machine ».

1. OntoDomus est une ontologie pour les systèmes ambiants d'assistance à la vie quotidienne [NPG22].

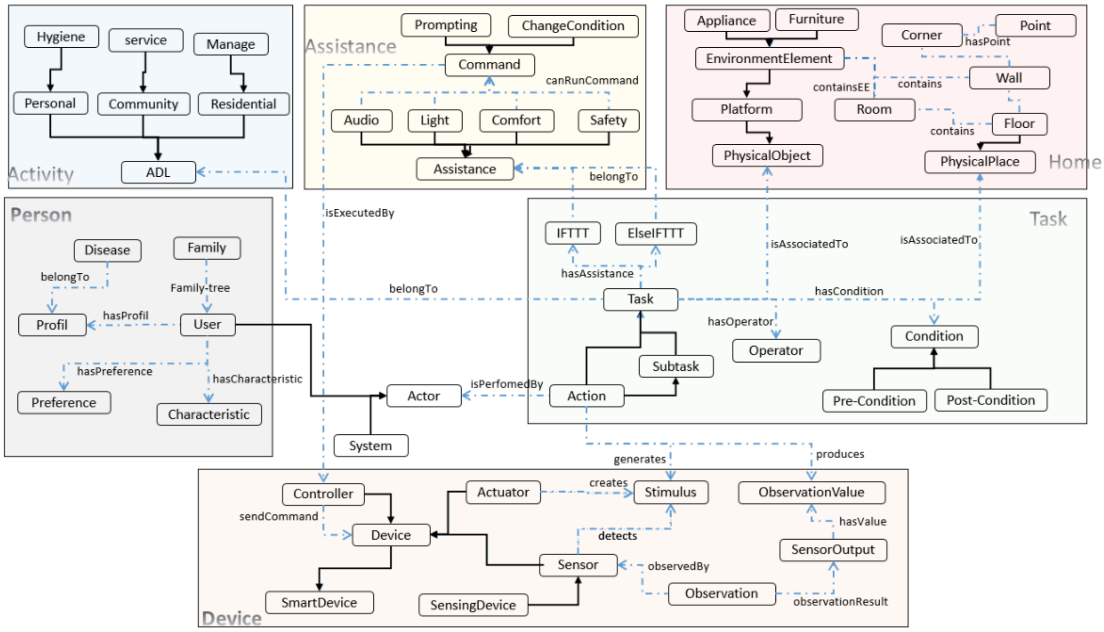


FIGURE 3.1 : Vue globale de l'ontologie OntoDomus [NPG22].

3.1.2 Définition formelle

Definition 3.1.1 (Ontologie) *:* une ontologie O est l'énoncé d'une théorie logique [Gru95] définie comme un tuple de cinq éléments $O(C, P, I, \Lambda, \Omega)$ [SEH+03, ZXY+21] où C et P sont, respectivement, les ensembles non vides de classes et de propriétés. L'ensemble I (il peut être vide) représente les individus qui sont des instances $(I : C)$ de classes qui représentent les concepts du monde réel. Λ est un ensemble non vide d'axiomes (assertions sous une forme logique) qui sont utilisés pour vérifier la cohérence de l'ontologie ou déduire de nouvelles informations, Ω est un ensemble d'annotations qui fournissent des métadonnées afin que l'humain/expert puisse avoir une vision claire de l'ontologie. Les éléments importants d'une ontologie sont définis de manière formelle comme suit :

$$
\begin{aligned}
C =& \{c_1, c_2, c_3, \ldots, c_n\} \mid C \neq \emptyset, \\
P =& \{p_1, p_2, p_3, \ldots, p_s\} \mid P \neq \emptyset, \\
I =& \{i_1, i_2, i_3, \ldots, i_r\} \mid \forall i, i : c \wedge c \in C.
\end{aligned}
$$

En général, les classes, les propriétés et les individus sont appelés entités, concepts ou termes.

3.1.3 Sensibilité au contexte et ontologie

Dans les systèmes interactifs avancés, la notion de contexte sera toujours présente puisqu'elle est la référence pour que ces systèmes fonctionnent correctement, d'autant plus si ces systèmes interagissent directement avec un être humain [KDA98]. Le développement de l'informatique omniprésente a donné naissance à de nouveaux modes d'utilisation de la technologie dans la vie quotidienne, tels que la décentralisation des unités de calcul, l'utilisation de capteurs multiples, d'actionneurs, etc. Ces technologies communiquent continuellement entre elles et avec les humains à chaque instant. Pour qu'une entité/objet/dispositif soit efficace et cohérente dans sa communication, elle doit être une technologie calme, ce qui implique sa capacité à être sensible au contexte – context-aware –.

Plusieurs définitions sont proposées concernant la notion de contexte et la sensibilité au contexte telles qu'elles sont spécifiées dans [KDA98, SAW94] et [SD03]. En ce qui concerne mes travaux de thèse, les deux définitions qui reviennent le plus souvent dans la littérature sont celles qui sont les plus appropriées [DAS01, ADB+99].

Definition 3.1.2 (Contexte – Context –) *:* il s'agit de toute information qui peut être utilisée pour caractériser la situation d'une entité. Dans ce cas, une entité est une personne, un lieu ou un objet qui est considéré comme pertinent pour l'interaction entre un humain et une application ou entre deux applications [ADB+99]. Comme les ontologies sont utilisées pour construire le contexte, il existe différents niveaux de sensibilité au contexte en fonction des quatre types d'ontologies [Roc03] (section 3.2).

Definition 3.1.3 (Sensibilité au contexte – Context-aware –) *:* un système est sensible au contexte s'il utilise le contexte, à savoir un lieu, le comportement d'une personne, des données liées aux températures, etc., pour fournir des informations ou des services pertinents à une entité (une personne, un lieu ou un objet), où la pertinence dépend de la tâche de l'entité qui l'utilise [ADB+99]. Par exemple, l'utilisation de la sensibilité au contexte pour déterminer l'inhabituel en utilisant la technique de *la sensation (Définition 1.1)*.

Definition 3.1.4 (Sensibilité à la sémantique – Semantic-aware –) *:* conformément à ce que j'ai compris de la littérature, un système est sensible à la sémantique – semantic awareness – s'il utilise une ontologie lexicale, par exemple WordNet, pour comprendre le sens des mots/documents et les relations entre eux, afin de fournir des informations ou des services pertinents à cette compréhension.

Les chercheurs ont proposé de nombreuses approches pour mettre en place des entités sensibles au contexte. Ces approches se distinguent selon les paradigmes de programmation et les modèles de développement. Sur cette base, les approches peuvent se retrouver dans les catégories suivantes [KPTV09] :

- solutions d'intergiciels et plates-formes de services spécialisées,

- utilisation d'ontologies,
- raisonnement basé sur des règles,
- programmation au niveau du code source/extensions du langage,
- approches fondées sur des modèles.

Étant donné que ce chapitre concerne la notion d'ontologie, citons quelques approches utilisant la représentation ontologique pour définir le contexte, comme le « Framework Context Toolkit » [DAS01], « Context Fabric Infrastructure » [HSK09], « systèmes multi-agents pour la représentation du contexte » [BBB+05b, SBPZ13]. L'idée qui sous-tend cette représentation est le formalisme posé par le « World Wide Web Consortium (W3C) » qui repose sur deux grands piliers, l'identification et l'adressage de toute ressource (objet, attribut, script, relation, etc.) à l'aide d'un unique identificateur de ressources uniformes (URI). Les ressources sont généralement composées des principaux éléments suivants : d'instances (individus), de concepts (classes), d'attributs et de relations. Les ressources sont représentées par une relation ternaire via les triplets RDF (Resource Description Framework) [MM04] associant un sujet, un prédicat et un objet (figure 3.2) [Nga19] :

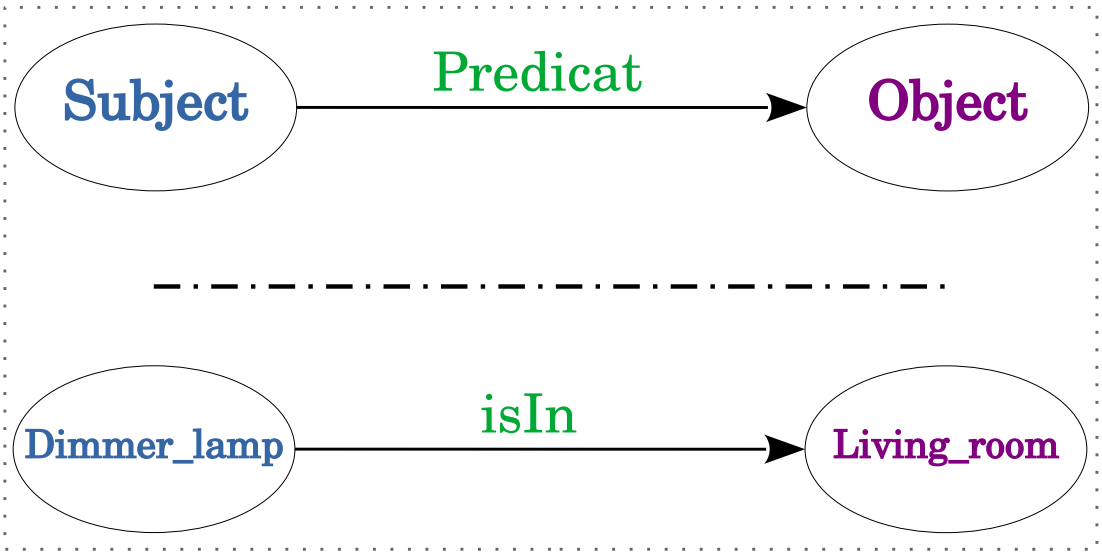


FIGURE 3.2 : Modèle de données RDF (partie supérieure) et son application concrète (partie inférieure).

- le sujet correspond à la ressource à décrire,

- le prédicat correspond à un type de propriété applicable à cette ressource,
- l'objet correspond à une donnée ou à une autre ressource.

À partir du modèle RDF, le langage de représentation des connaissances – Web Ontology Language (OWL) – est construit [Gro11]. Un exemple complet de l'ontologie d'une lampe à intensité variable – dimmer lamp – dans un environnement domotique intelligent est donné dans la figure 3.3. Comme le montre la relation

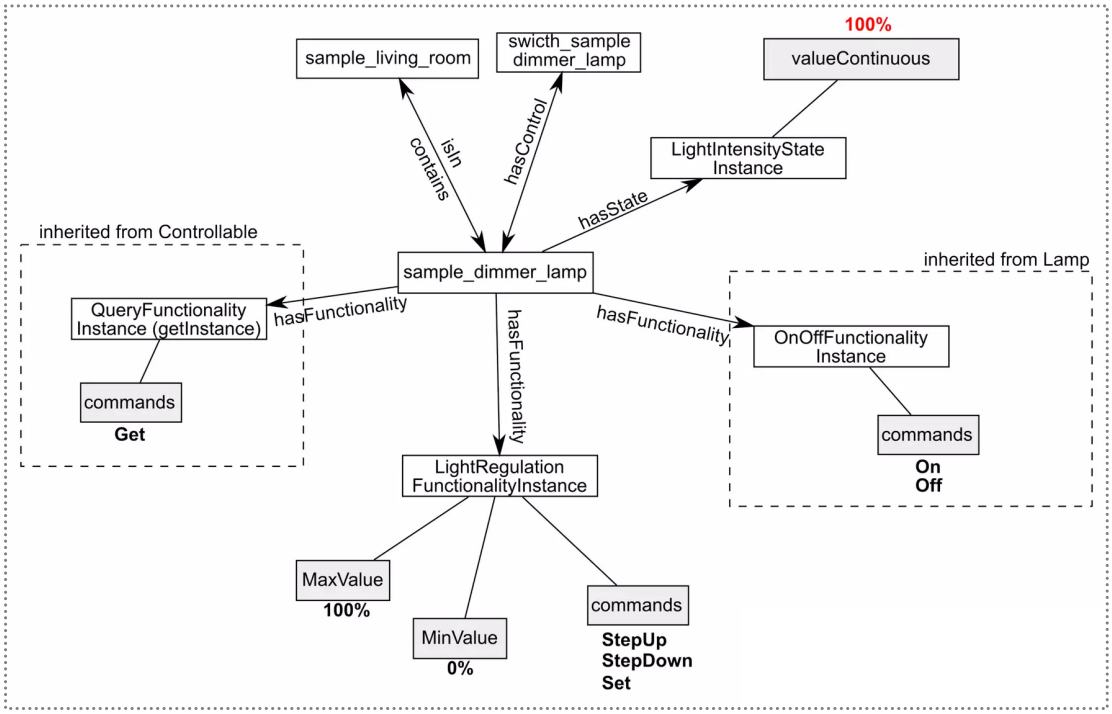


FIGURE 3.3 : Ontologie de l'instance d'un luminaire à intensité variable [BC08].

« hasFunctionality » entre « sample_dimmer_lamp » et « OnOffFunctionality », il peut en être déduit que l'instance « sample_dimmer_lamp » est contrôlable par des commandes de type « On » et « Off ». Cette information est déduite après une étape de raisonnement qui calcule la finalité transitive du modèle, transformant ainsi des connaissances implicites en informations explicites [BC08].

Pour revenir à la sensibilité au contexte, si l'ontologie de la lampe à intensité variable illustrée dans la figure 3.3 est intégrée dans l'ontologie d'une maison intelligente (par exemple, OntoDomus), un lien entre l'instance de la lampe à intensité variable et la cuisine avec une relation « isIn » pourra être fait. Du côté de la lampe, ce lien donnera à celle-ci un contexte – existence dans la cuisine – d'autre part, la cuisine « saura » qu'elle a une lampe à intensité variable. En outre, cette

technique permettra la sélection proximale – Proximate Selection –, si un utilisateur demande « allumer la lumière » près de la cuisine, la lampe à intensité variable située dans la cuisine sera choisie pour satisfaire la demande. Pour plus de catégories d'applications contextuelles, voir [SAW94].

3.2 Types d'ontologie

Selon [VHSW97, vH95] et [Abe93], les ontologies se divisent selon quatre différents niveaux de généralité :

- Les ontologies représentationnelles sont des ontologies qui ne se limitent pas à un domaine particulier. Ces ontologies fournissent des entités de représentation sans préciser ce qui doit être représenté [Gru93]. L'exemple qui suit ce modèle est l'éditeur d'ontologie « Protégé ». Il utilise ce type d'ontologie pour donner à ses utilisateurs la capacité de créer des ontologies. Exemple : l'entité class, les relations type, subtype, hasAttribute, etc.

- Les ontologies génériques sont des ontologies valables dans plusieurs domaines. Par exemple, une ontologie sur les mesures, les capteurs, les personnes, etc., sont des ontologies applicables à de nombreux domaines. Les ontologies génériques sont également appelées des ontologies centrales [BAT97] [VHSW97].

- Les ontologies d'un domaine sont des ontologies qui représentent les connaissances valables pour un type particulier de domaine. Par exemple, Maison intelligente, électronique, médical, etc.

- Les ontologies d'application sont des ontologies qui contiennent toutes les connaissances nécessaires à la représentation d'un domaine particulier. Généralement, elle est une combinaison d'ontologies de domaine et de méthode [2] [HWEA94]. Contrairement à l'ontologie du domaine, elle est plus spécifique et non réutilisable. Par exemple : OntoDomus.

Dans cette thèse, j'utilise la base de données lexicale Wordnet dans mon travail de recherche comme un point de départ vers la sensibilisation des WO au contexte. La base de données lexicale WordNet n'est pas une ontologie contextualisée, j'utilise donc un contexte général [3], un contexte plus restreint sera traité dans des travaux futurs.

2. Les ontologies méthodologiques ou ontologies de méthodes contiennent des connaissances sur la manière de résoudre un problème ou d'accomplir une tâche [HWEA94].

3. Un contexte général d'une application sage ne prend pas en compte le domaine de cette application.

3.3 Utilisation des ontologies

Les ontologies sont principalement utilisées dans le domaine de l'ingénierie de la connaissance en raison de leur capacité à fournir un modèle compréhensible par une machine, sans ambiguïté si celui-ci respecte les normes et les critères de conception [Gru95] tels que :

- La précision : l'ontologie doit communiquer le sens des termes définis sans aucune ambiguïté et en toute objectivité. En ce qui concerne la définition des concepts, (1) la motivation doit résulter des exigences informatiques, (2) la définition doit se présenter sous une forme logique (3) toutes les définitions doivent être documentées à l'aide d'un langage naturel.

- La cohérence : les définitions doivent être logiquement cohérentes. La cohérence doit également s'appliquer aux concepts définis de manière informelle, tels que ceux décrits dans la documentation en langage naturel.

- L'extensibilité : l'ontologie doit accepter de nouvelles définitions pour des utilisations spéciales basées sur le vocabulaire existant, de manière à ne pas nécessiter la révision des définitions existantes.

- Le biais d'encodage minimal : la conception de l'ontologie doit être définie sur la base des connaissances sans tomber dans l'erreur du biais d'encodage. Un biais d'encodage se produit lorsque les choix de représentation sont faits uniquement pour des raisons de facilité de notation ou d'implémentation.

- Un engagement ontologique minimal, l'ontologie doit faire le moins de déclarations possible sur le monde modélisé. Ce qui permet aux parties engagées dans l'ontologie de se spécialiser et d'instancier l'ontologie en fonction des besoins. Étant donné que l'engagement ontologique repose sur l'utilisation cohérente du vocabulaire, il peut être minimisé en spécifiant la théorie la plus faible – Weakest theory – (permettant le modèle général) et en définissant uniquement les termes qui sont essentiels à la communication de connaissances conformes à cette théorie [4].

Comme les ontologies contiennent des termes, leurs significations et leurs relations, elles permettent la communication entre la machine et les humains, entre deux machines ou entre deux humains, sachant qu'il existe plusieurs approches

4. Une ontologie a un objectif différent de celui d'une base de connaissances et, par conséquent, une notion différente d'adéquation de la représentation s'applique [MH69]. Une ontologie partagée ne doit décrire qu'un vocabulaire permettant de parler d'un domaine, alors qu'une base de connaissances peut inclure les connaissances nécessaires pour résoudre un problème ou répondre à des questions arbitraires sur un domaine[Gru95].

de l'ontologie [Roc03], à savoir l'ontologie web, l'ontologie d'ingénierie, l'ontologie linguistique, etc. Ce qui donne une utilisation dans plusieurs domaines d'application, comme la gestion de la connaissance d'une entreprise [Obe14] tel qu'illustré dans le projet TOVE [FCF93], le suivi et l'évaluation des situations et des événements susceptibles d'affecter la santé publique, grâce au système d'information sanitaire SAPPHIRE basé sur la sémantique, ou encore rassembler des données provenant de différentes sources pour tirer plus d'avantages des systèmes d'information géographique. Par ailleurs, l'utilisation qui est considérée dans cette thèse est le calcul de la similarité entre deux concepts (mots, textes), ce calcul est basé en général sur des ontologies générales telles que WordNet [RB90, Mil95b] en utilisant des méthodes comme le comptage des arêtes [RMBB89], le contenu des informations [MJB15], les méthodes basées sur les caractéristiques (propriétés concrètes) [Tve77] et méthodes hybrides [RSM94].

Les ontologies sont également utilisées dans les systèmes de Machine Learning, certaines d'entre elles nécessitent le travail collaboratif d'experts de différents domaines. C'est le cas de l'ontologie « QMM-Core » du contrôle de la qualité dans l'industrie automobile, comme expliqué dans [SZP+20]. Ce type d'ontologie a sa place dans les modèles de Machine Learning, puisqu'elle facilite leurs inspections et explique leurs résultats. Par ailleurs, de nombreuses opérations sont apparues entre ontologies comme l'appariement d'ontologies [LTLQ21] – ontology matching – en tant que solution au problème de l'hétérogénéité sémantique dans l'intégration et le partage de l'information. Il en est de même pour la création de scénarios d'assistance dans une maison intelligente exprimés par un utilisateur humain en langage naturel (l'objet du chapitre 6 de cette thèse).

3.4 Conclusion

La revue de la littérature a permis de présenter les concepts de base de l'ontologie, sa définition linguistique et formelle. La conception d'une ontologie doit toujours commencer par la recherche d'une ontologie existante qui couvre le domaine/application en question, et ce afin d'étendre cette ontologie en respectant les critères mentionnés dans [Gru95]. Les ontologies sont largement utilisées pour dépasser les problèmes liés à la sensibilité au contexte, elles permettent également de surmonter un certain nombre de limitations, telles que celles du langage de balisage hypertexte traditionnel (HTML), qui ne permettait pas d'obtenir des informations sémantiques détaillées sur une page web avant l'apparition du web sémantique. Le web sémantique est une combinaison du HTML avec les ontologies. Enfin, l' utilisation des ontologies n'est pas limitée à un domaine/application précis, mais à plusieurs domaines allant de la communication entre humains à l'amélioration des modèles de Machine Learning.

Conclusion

Cette première partie de la thèse introduit le concept des WO, qui sont caractérisés par une connaissance autonome de leurs comportements sans besoin d'un intervenant/humain externe, ainsi que le framework (WOF) pour faire face aux problèmes de développement d'applications basées sur l'IA, en utilisant des méthodes et des outils similaires à ceux utilisés pour les applications « traditionnelles ». Ce cadre fournit des mécanismes de base pour la surveillance et l'analyse du comportement des WO, tout en donnant aux développeurs la possibilité de les étendre, et d'avoir des analyseurs concurrents grâce à l'utilisation de plugins d'IA. Bien qu'un WO puisse connaître son comportement de manière autonome, il est limité dans la communication avec les humains, puisque la connaissance du comportement déclenche la génération automatique d'un modèle comportemental, un STG dans notre cas, qui n'a pas de sémantique. Le modèle IOSTS entre en jeu pour l'enrichir par le biais de la sémantique, c'est l'objet des deux chapitres 4 et 5 des contributions. Le modèle IOSTS est un outil mathématique permettant de modéliser des systèmes infinis au moyen de variables internes, ces variables permettent de représenter les états de ce système. Le choix d'un IOSTS est tout d'abord dû au fait qu'il représente la même chose qu'un STG d'une manière contractée (finie) et sémantique. Ensuite, il est également présent lors de la phase conceptuelle, ce qui signifie que ce qui a été fait lors de la phase de conception peut être réutilisé.

Dans le dernier chapitre, le concept d'ontologie est abordé, ce dernier est au centre de nombreux défis dans le secteur des technologies de l'information. Dans le cadre de cette thèse, ce concept est utilisé pour contextualiser le lien entre un STG et un IOSTS. C'est l'objet du chapitre 6 de la deuxième partie de cette thèse (contributions).

Comme indiqué dans la première partie de cette conclusion, l'IOSTS est utilisé pour enrichir le STG par le biais de la sémantique. Comme il s'agit de deux graphes dont l'un est un graphe sémantique, je suis confronté au problème d'appariement de graphes d'une part, et de la contextualisation de cet appariement d'autre part. La mise en contexte est réalisée en incluant l'ontologie dans le calcul d'appariement.

II

Contributions

Introduction

Au fur et à mesure que l'utilisation de l'informatique ubiquitaire évolue dans notre vie quotidienne, il est devenu essentiel de respecter la notion de technologie calme. Pour parvenir à ce type de technologie, la sémantique est, de mon point de vue, l'un des principaux piliers de la communication entre machines et humains. Elle est construite à l'aide de plusieurs techniques, notamment, les ontologies, certains types de graphes comportementaux complexes ou encore les graphes de connaissances.

En outre, et comme détaillé dans le chapitre 1, le mécanisme d'introspection aide les WO à découvrir leurs comportements en utilisant la conscience pour collecter des données sur eux-mêmes. Ces données sont utilisées par les analyseurs, par exemple, le « STG Generator » pour construire une nouvelle connaissance pour les WO. Cette nouvelle connaissance peut être utilisée par d'autres analyseurs, comme le « Graph Matcher », qui étend le STG en lui ajoutant de la sémantique, via les IOSTS donnés par des experts/développeurs aux WO. Cette extension est basée sur trois types d'algorithmes d'appariement de graphes, qui sont l'objet principal de la contribution.

Sommaire

Chapitre 4

Vers un modèle sémantique pour les systèmes sages

Au cours de la conception d'une application, les experts/développeurs peuvent fournir un modèle conceptuel décrivant et spécifiant la façon dont ils voient le comportement des entités de l'application associées aux WOs, autrement dit, les entités qui seront sages. Ces modèles sont représentés à l'aide de plusieurs formalismes, et dans cette thèse, nous avons choisi les IOSTS. Ces formalismes contiennent la sémantique donnée par les développeurs aux WO. Le formalisme IOSTS est principalement connu pour simplifier la modélisation des systèmes en permettant une

représentation symbolique des paramètres et des valeurs de variables, au lieu d'une énumération de valeurs de données concrètes [MSV12]. Le formalisme IOSTS est utilisé dans ce chapitre pour représenter la connaissance sémantique d'un système. Ce formalisme appartient aux types de graphes conceptuels et capture la sémantique dans la représentation des données.

Ce chapitre détaille mon approche pour donner de la sémantique aux connaissances d'un WO en utilisant la notion d'appariement de graphes de différentes natures. Deux versions d'appariement sont présentées, la version simple et sa version générique, respectivement, univariée et multivariée.

4.1 Appariement de graphes (travaux connexes)

De manière simple, la notion d'appariement de graphes est le problème de la recherche d'une similarité entre les graphes. Plusieurs approches ont été développées, la première formulation du problème de l'appariement de graphes a été proposée par Wen-Hsiang Tsai et King-Sun Fu dans [TF79], en 1979. Par la suite, plusieurs formulations sont apparues, telles que la programmation convexe-concave, le sous-graphe commun maximal – maximum common subgraph (MCS) –, l'utilisation de la norme de Frobenius qui utilise les matrices d'adjacence des graphes correspondants pour exprimer la maximisation ou la minimisation des arêtes qui ne se chevauchent pas entre deux graphes, l'appariement de graphes utilisant des nœuds fictifs – dummy vertices – [ARE98] qui consiste à trouver un appariement à l'exception de certains sommets du graphe de données, qui n'ont aucune correspondance. En général, il existe deux formulations principales du problème de l'appariement des graphes [Zas10, Ben02] : l'appariement exact et l'appariement inexact, présentés dans les sous-sections suivantes.

4.1.1 Types d'appariement

Appariement exact

L'appariement exact se divise en deux catégories : (a) l'isomorphisme de graphes, qui vérifie si deux graphes sont identiques. (b) l'isomorphisme de sous-graphes, qui vérifie si le plus petit graphe est un sous-graphe du plus grand. Les deux techniques sont très complexes et reposent sur l'isomorphisme de graphes/sous-graphes. Un isomorphisme entre deux graphes/sous-graphes (G, H) est une fonction $f\colon V(G) \rightarrow V(H)$ qui fait correspondre les nœuds V du premier graphe à ceux du second graphe, en respectant la structure $(\forall(u,v) \in E(G), (f(u), f(v)) \in E(H) \mid (u,v) \in V^2(G))$ entre leurs nœuds [HN04]. $V(G)$ et $E(G)$ sont respectivement les ensembles de nœuds et d'arêtes du graphe G. Pour un isomorphisme de sous-graphes, $E(G) \subseteq E(H)$.

Appariement inexact

L'appariement inexact est un terme utilisé lorsqu'il est impossible de trouver un isomorphisme entre deux graphes [Ben02]. Dans ce cas, la technique d'appariement des graphes consiste à déterminer le meilleur appariement entre eux, à l'aide de plusieurs techniques :

- Le sous-graphe commun maximal (MCS), utilisé pour rechercher la similarité entre les graphes afin de connaître leur degré de différence au lieu d'une réponse binaire [Ull76].

- La formulation des moindres carrés – least square formulation –, utilisée dans le cas des graphes pondérés pour rechercher une correspondance qui minimise la différence totale entre toutes les arêtes alignées en utilisant la norme de Frobenius par exemple [Ull76].

- La distance entre graphes par édition – graph edit distance –, utilisée pour trouver à faible coût la séquence d'opérations (c'est-à-dire la suppression, l'insertion et la substitution de sommets et d'arêtes) qui transforment un graphe en un autre [BA83]. Étant donné que cette procédure est un problème combinatoire difficile, une autre solution appelée « recherche par faisceau – beam search – » a été proposée, elle est expliquée en détail dans [NRB06].

4.1.2 Quelques techniques alternatives du problème de l'appariement de graphes

Appariement de graphes : étiquettes de sommets

Étant donné qu'il existe des graphes avec des étiquettes de sommets – vertex labels –, un type d'appariement de graphes se fonde sur la comparaison de ces étiquettes, en particulier sur la mesure de la similarité entre les étiquettes. Le défi consiste à choisir le bon modèle mathématique pour mesurer la similarité, car ces étiquettes peuvent être des variables discrètes (catégorielles) ou continues (numériques). Ce type de graphes est également utilisé dans plusieurs domaines comme : la vision par ordinateur [SRT06], la bio-informatique [PML05]. Ces méthodes sont connues pour leur simplicité et leur rapidité, avec l'inconvénient que la structure du graphe n'est pas prise en compte sauf dans quelques travaux comme illustré dans [CSC01] et [SXB08]. La thèse de Zaslavskiy [Zas10] présente plus de détails sur ce sujet.

Appariement de graphes : algorithmes génétiques

L'algorithme génétique est une technique permettant de formuler le problème d'appariement de graphes comme une optimisation combinatoire. Dans ce contexte, certaines recherches se concentrent sur le type d'opérateurs de création de populations, de croisement et de mutation, comme illustré dans [KS02]. D'autres utilisent les mesures bayésiennes – Bayesian measures – en exploitant la mesure de l'aptitude – fitness measure – [CWH97, WH96]. Ce type d'algorithme est utilisé pour la reconnaissance d'images du cerveau humain comme illustré dans [CCI04, BPBR99, Ben02]. Les inconvénients de ces algorithmes sont qu'ils sont parfois difficiles à comprendre et à débugger, et qu'ils ne sont pas déterministes (plusieurs solutions) avec une grande population ; ce qui donne des résultats ambigus [MH00].

Appariement de graphes : méthodes probabilistes

De nombreuses techniques appliquent la théorie des probabilités aux problèmes d'appariement des graphes. La relaxation probabiliste est utilisée pour résoudre le problème d'appariement des graphes sous certaines conditions [YZL13] pour l'attribution d'étiquettes contextuelles, en combinant plusieurs mesures de distance relationnelle. Cette technique est utilisée par exemple dans des applications d'appariement des réseaux routiers (données géospatiales) [YZL13]. L'appariement spectral permet d'approximer la solution de problèmes d'appariement entre les paires de nœuds. L'appariement probabiliste spectral est utilisé dans des applications d'analyse d'images [AYH13]. On trouve également quelques autres techniques basées sur la théorie des probabilités décrites dans la thèse de Bengoetxea [Ben02] et détaillées dans [FC01, Mar00, RS21].

Problème d'affectation quadratique

Initié par les deux économistes Tjalling C. Koopmans et Martin Beckmann en 1957 dans [KB57], ce problème est l'un des problèmes d'optimisation combinatoire fondamentaux dans la branche de l'optimisation en mathématiques, appartenant à la classe des problèmes NP-difficiles [SG76]. Le problème de l'appariement des graphes peut être formulé comme un problème d'affectation quadratique qui consiste à trouver une allocation optimale des sommets d'un graphe sur ceux de l'autre graphe en utilisant des normes comme la norme l_1 ou son alternative l_2 (aussi nommée Frobenius). Pour la formulation du problème, voir [Zas10] et [LdBN+07].

De nombreuses autres techniques d'appariement existent dans la littérature et ne sont pas citées dans cette thèse pour en faciliter la compréhension aux lecteurs, comme les réseaux neuronaux qui sont utilisés pour la reconnaissance d'objets en 3D [LP02] dans le domaine médical [RMPO+02], le domaine de la météo et climat [LL00]. Le concept des arbres de décision a également sa place dans le domaine de l'appariement de graphes : ils sont créés à l'aide d'un ensemble de graphes modèles connus a priori, générés à partir de la détection d'isomorphismes exacts de sous-graphes. Comme expliqué dans [SBV01], les arbres de décision sont utilisés pour résoudre le problème du plus grand sous-graphe commun au lieu d'appliquer des requêtes à une base de données de modèles.

4.1.3 Complexité du problème d'appariement de graphes

Dans les applications réelles, nous voulons souvent faire correspondre des graphes de tailles différentes, ce qui donne lieu à de nouvelles techniques et normes, comme indiqué dans [Zas10]. En outre, le problème est plus vaste qu'on ne pourrait l'imagi-

ner, car les graphes sont utilisés pour représenter des objets, des images, des régions dans des cartes géographiques, ce qui a donné de nombreux types de correspondance, comme la correspondance entre différentes représentations de la connaissance en tant que STG et IOSTS, qui, à notre connaissance, n'ont fait l'objet d'aucune publication. Jusqu'à présent, les opérations les plus connues sur les graphes sont la comparaison de deux ou plusieurs représentations de graphes qui nécessitent de nombreux concepts théoriques complexes [GJ83], comme l'appariement de graphes, qui est une version plus contrainte du problème de l'isomorphisme de graphes (la base de cette thèse). Enfin, nous mentionnons que l'isomorphisme et l'homomorphisme de graphes/sous-graphes sont considérés comme les problèmes les plus complexes de l'appariement de graphes, ils sont NP-complets. Ces problèmes ont été étudiés dans [Bas94, GJ83, Abd98]. Il convient de mentionner que pour certains types de graphes soumis à certaines contraintes, il a été prouvé que la complexité de l'isomorphisme était de type polynomial avec un coût important [HW74].

4.1.4 Un STG ayant des caractéristiques particulières

Je rappelle tout d'abord qu'un STG générique est défini par le triplet $G(V, E, L)$ où V et E sont, respectivement, les ensembles de sommets et d'arêtes, et L un ensemble d'étiquettes. La définition 2.2.2 contient plus de détails sur la structure des ensembles V, E et L.

Dans cette thèse, j'utilise un STG ayant des caractéristiques particulières que je nomme « l'exhaustivité ». La définition d'un « STG exhaustif » que j'introduis est donnée dans la définition 4.1.1.

Definition 4.1.1 (STG exhaustif) *:* Un STG exhaustif est un STG dont chaque sommet v_i comporte $|M|$ transitions de sortie, où chacune de ses transitions est étiquetée par une méthode m_k de M différente :

$$\forall v_i \in V, \forall m_k \in M, \exists v_j \in V | (v_i, v_j, m_k) \in E.$$

Il convient de noter que v_i et v_j peuvent être des états différents ou identiques ($v_i \neq v_j$ ou $v_i = v_j$).

Par conséquent, un STG exhaustif est déterministe, c'est-à-dire qu'à partir de n'importe quel état, sur n'importe quel appel de méthode, l'état de destination est connu. De plus, le nombre de transitions $|E|$ dans un STG exhaustif dépend du nombre de sommets $|V|$ et de méthodes $|M|$ tel que :

$$|V| \times |M| = |E|. \tag{4.1}$$

Chaque sommet représente un état et un ensemble unique de valeurs d'attributs, et chaque attribut est défini selon un domaine discret et borné. L'ensemble

des attributs est naturellement fini, le nombre de sommets |V| est borné et défini comme suit :

$$|V| \leq \prod_{i=1}^{|A|} |Dom(att_i)|. \tag{4.2}$$

L'inégalité est due au fait que certaines combinaisons de valeurs d'attributs ne sont pas possibles. Le nombre d'états possibles est $\prod_{i=1}^{|A|} |Dom(att_i)|$, mais ceux-ci ne sont pas tous nécessairement atteignables.

Example 4.1.1 *:* La figure 4.1 illustre un STG exhaustif du comportement d'un volet roulant simplifié à lames orientables. Il est défini par les attributs « level » et

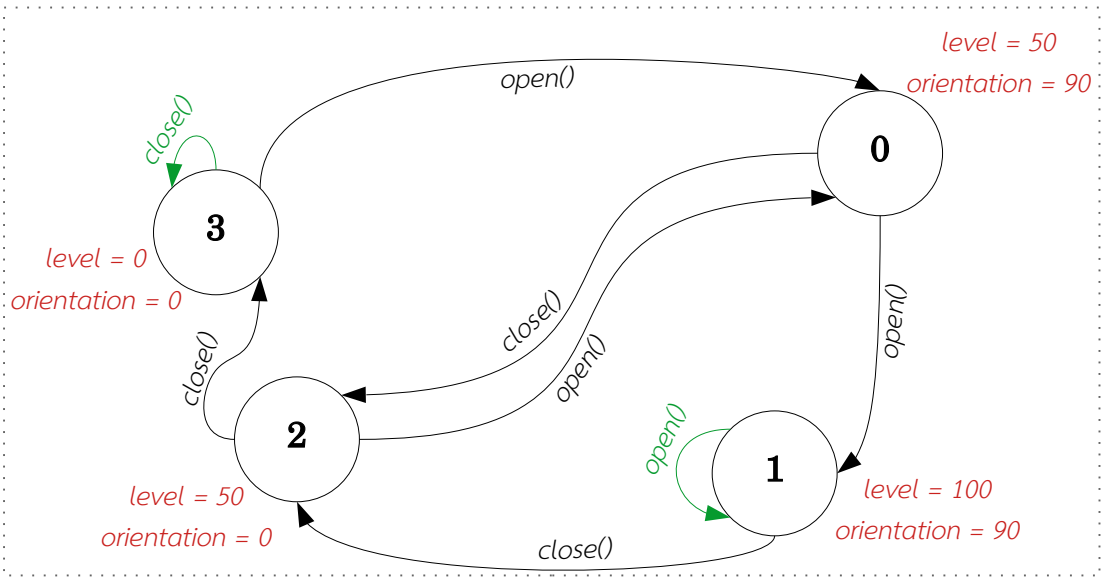


FIGURE 4.1 : STG exhaustif du comportement d'un volet roulant.

« orientation » $(A = \{level, orientation\})$ et deux méthodes « open » et « close » $(M = \{open(), close()\})$. Les méthodes « open » et « close » incrémentent et décrémentent la valeur de l'attribut « level » par 50 quand un utilisateur ouvre ou ferme le volet roulant, respectivement. L'orientation des lamelles est ajustée automatiquement pour prendre les valeurs 0 ou 90. Ce STG a été généré automatiquement par un WO et l'état 0 correspond au premier état découvert, l'état 1 au deuxième, etc. Par conséquent, à l'exception des méthodes, open() et closed(), qui donnent une sémantique aux transitions, les états n'ont pas de sémantique. En effet, les méthodes sont créées par les développeurs d'applications, mais les états sont découverts par le WO. Selon les domaines des attributs « level » et « orientation », et la propriété 4.2, le STG a 4 états atteignables. Par ailleurs, le nombre d'arêtes est de 8 selon la propriété 4.1. Précisons que l'état 0 est indépendant des valeurs initiales du volet roulant, il correspond au premier état trouvé lors de la génération automatique du STG.

L'exemple du STG de l'ampoule connectée, donné dans la figure 2.3, se présente comme suit (figure 4.2) dans sa version exhaustive :

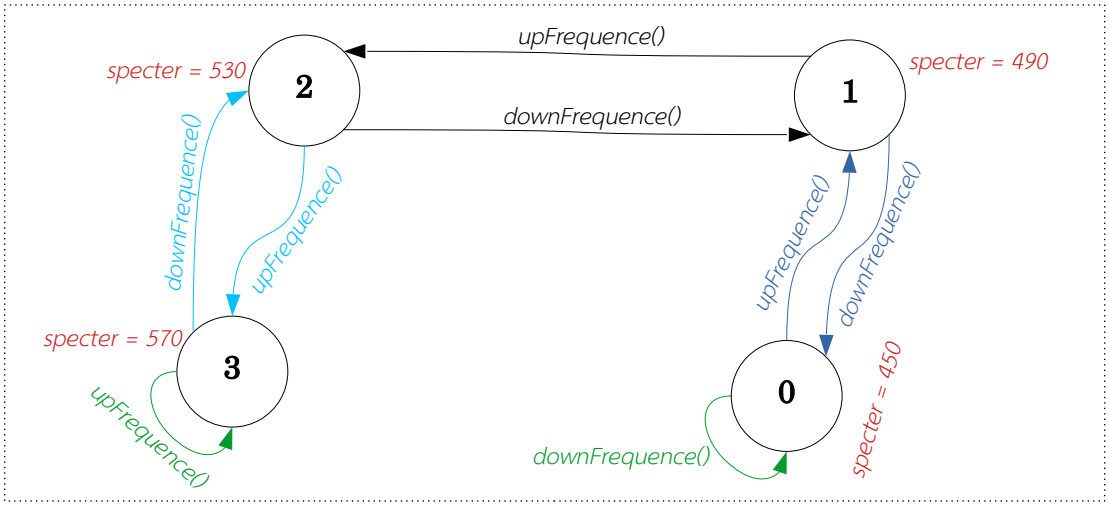


FIGURE 4.2 : STG exhaustif du comportement d'une ampoule connectée.

L'architecture du WOF étant basée sur une approche modulaire « programmation orientée composant », j'ai proposé que la contribution qui consiste à faire correspondre deux graphes (STG et IOSTS) de natures différentes prenne la forme d'un plugin appelé « Graph Matcher ». Il convient également de noter que dans cette thèse, je ne considère que les STG exhaustifs de comportements pour mettre en évidence cette contribution.

4.2 Position du plugin d'appariement de graphes dans le WOF

La figure 4.3 présente la source de chaque représentation de la connaissance. Plus précisément, le métier lié à l'application (exemple : application domotique, application bancaire, etc.) est la partie principale. Une fois cette partie encapsulée par le WO, ce dernier va commencer l'introspection dans l'application pour avoir les données internes que je nomme connaissances, et qui seront représentées sous la forme d'un **STG**. D'autre part, nous avons les informations données par le développeur dans un modèle conceptuel représenté sous la forme d'un **IOSTS**, ces données portent la sémantique. En utilisant le morphisme de graphes, je vais relier les deux représentations (celle générée par le WO et celle donnée par le développeur) pour étendre le WO avec la sémantique. Pour rappel, en théorie des graphes, un morphisme entre deux graphes (G, H) est une fonction $f : V(G) \rightarrow V(H)$ qui

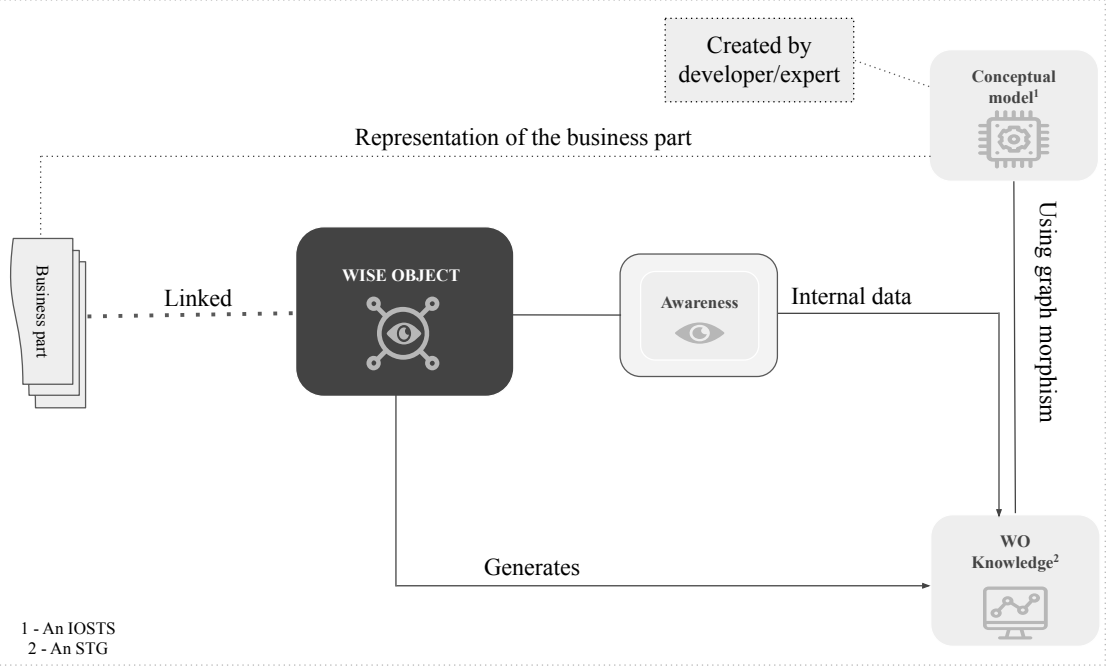


FIGURE 4.3 : Processus à suivre pour obtenir la sémantique.

fait correspondre les nœuds V du premier graphe au second graphe en respectant la structure $(\forall(u,v) \in E(G), (f(u), f(v)) \in E(H))$ entre leurs nœuds [HN04]. $E(G)$ est l'ensemble des arêtes du graphe G.

D'un point de vue plus large et plus abstrait, nous avons la figure 4.4 qui représente la même chose que la figure 4.3, avec le focus sur le STG généré par le WO par l'intermédiaire du plugin [1] [CEM03] appelé générateur de graphes comportemental – Behavioral Graph Generator – et le modèle conceptuel créé par le développeur. Le modèle conceptuel (IOSTS) est donné au WOF pour obtenir la sémantique en utilisant le morphisme de graphe avec le STG, par le biais du plugin Graph Matcher. Plus précisément, ma proposition est d'étendre la connaissance générée avec la connaissance conceptuelle en utilisant un algorithme d'appariement de graphes – Graph Matching Algorithm (GMA) –, univarié et multivarié basé sur le morphisme des graphes [GJ83, Cic99]. Ceci permet de rendre la connaissance générée par les systèmes sages compréhensible par les humains et de permettre l'évaluation humaine des résultats des systèmes sages.

1. Les plugins sont des composants qui nous permettent d'explorer des techniques de conception pour isoler les caractéristiques spécifiques à une plateforme. Ils sont également dédiés à l'extension des applications pour fournir des fonctionnalités supplémentaires. Dans le WOF, un analyseur est un plugin.

4.3 Algorithme d'appariement univarié

Dans cette section, je présente l'algorithme d'appariement de graphes que je propose pour relier le STG généré par un WO à la sémantique donnée par les développeurs/experts exprimée dans un IOSTS. Pour illustrer le processus global, examinons la construction d'un échantillon d'appariement de graphes sur un volet roulant simple et celui d'une lampe simple à intensité variable. Le mot « simple » signifie ici que nous nous baserons sur un seul couple attribut/variable (dans le STG/IOSTS) pour effectuer l'appariement [DAMV22b].

4.3.1 Contraintes et algorithme d'appariement

Contraintes : le STG et l'IOSTS doivent répondre à certains critères pour appliquer correctement l'algorithme.

1. Il existe deux caractéristiques équivalentes : une variable x_e appartenant à l'ensemble des variables X de l'IOSTS et un attribut att_e appartenant à l'ensemble des attributs A du STG. Comme cette section est destinée à simplifier la solution qui sera présentée complètement dans la section 4.4, considérons qu'ils sont uniques :

$$\exists! x_e \in X, \exists! att_e \in A | x_e \equiv att_e, \tag{4.3}$$

   $x_e \equiv att_e$ signifie que les deux représentent la même information, avec :

$$Dom(att_e) \subseteq Dom(x_e). \tag{4.4}$$

   Notons que $Dom(att_e)$ est un sous-ensemble de $Dom(x_e)$ car x_e est théoriquement défini dans l'IOSTS, et att_e est partiellement découvert par le WO au moment de son exécution – runtime –. En complément, $Dom(x_e)$ est défini comme un domaine continu et $Dom(att_e)$ est défini comme un ensemble discret et borné de valeurs.

2. Les domaines de x_e dans les différentes localités de l'IOSTS sont disjoints :

$$\begin{gathered} \forall q, q' \in Q, q \neq q' \\ \Leftrightarrow \\ dom(q, x_e) \cap dom(q', x_e) = \emptyset. \end{gathered}$$

Algorithme : selon les définitions du STG et de l'IOSTS, et les deux contraintes, un état (nœud) $v_i \in V$ correspond à une localité $q_j \in Q$ (noté $v_i \implies q_j$), si

et seulement si, $value_{i,e} \in dom(q_j, x_e)$, avec $value_{i,e}$ la valeur de att_e dans l'état v_i :

$$\begin{array}{c} \forall v_i \in V, \exists! q_j \in Q \\ value_{i,e} \in dom(q_j, x_e) \Leftrightarrow v_i \implies q_j. \end{array} \tag{4.5}$$

Comme l'algorithme d'appariement est un morphisme de graphes, ce dernier doit respecter la structure des graphes appariés [GC97]. Dans notre contexte, chaque état correspond à une localité et une localité est appariée à au moins un état. De plus, les relations d'adjacence doivent être respectées par l'appariement ; si deux états sont reliés par une transition dans le STG, leurs localités appariées sont reliées par une transition équivalente dans l'IOSTS. L'appariement STG $\rightarrow$ IOSTS est un homomorphisme surjectif $\mathcal{S}$, c'est-à-dire un épimorphisme [GC97] comme l'illustre l'équation ci-dessous [DAMV23a] :

$$\begin{array}{rl} \mathcal{S} \colon STG & \rightarrow IOSTS \\ V & \twoheadrightarrow \mathcal{S}(V) = Q, \end{array} \tag{4.6}$$

ce qui implique :

$$\mathcal{S}_E \colon E \rightarrow T,\ \mathcal{S}_E((v_i, v_j)) = (\mathcal{S}(v_i), \mathcal{S}(v_j)) \subset T. \tag{4.7}$$

Pour toute transition $(v_i, v_j) \in E$ du STG, $(\mathcal{S}(v_i), \mathcal{S}(v_j)) \in T$ est une transition de l'IOSTS.

4.3.2 Algorithme d'appariement par étapes

Comme l'illustre la figure 4.5, l'algorithme que je propose est divisé en deux étapes : la première consiste à faire correspondre les variables-attributs en fonction de leur domaine, défini par la fonction « *compatibleDomain* ». La seconde étape est consacrée au calcul de l'appariement entre les états et les localités. La section qui suit décrit les détails de l'implémentation de la fonction « *compatibleDomain* » dont le résultat fera partie des données d'entrée de l'algorithme principal. En détail, les équations (4.3) et (4.4) sont développées dans la fonction « *compatibleDomain* » exposée dans l'algorithme 1. L'équation (4.5) est exposée dans l'algorithme 2.

Le pseudo-code (algorithmes 1 et 2) est la première version de l'algorithme d'appariement que nous avons développé et testé en tant que première étape vers la sémantique humaine.

4.3.3 Illustrations

Comme indiqué dans le début de ce chapitre et dans le chapitre 2, nous avons considéré un STG/IOSTS à plusieurs attributs/variables (figure 4.1) pour approcher le

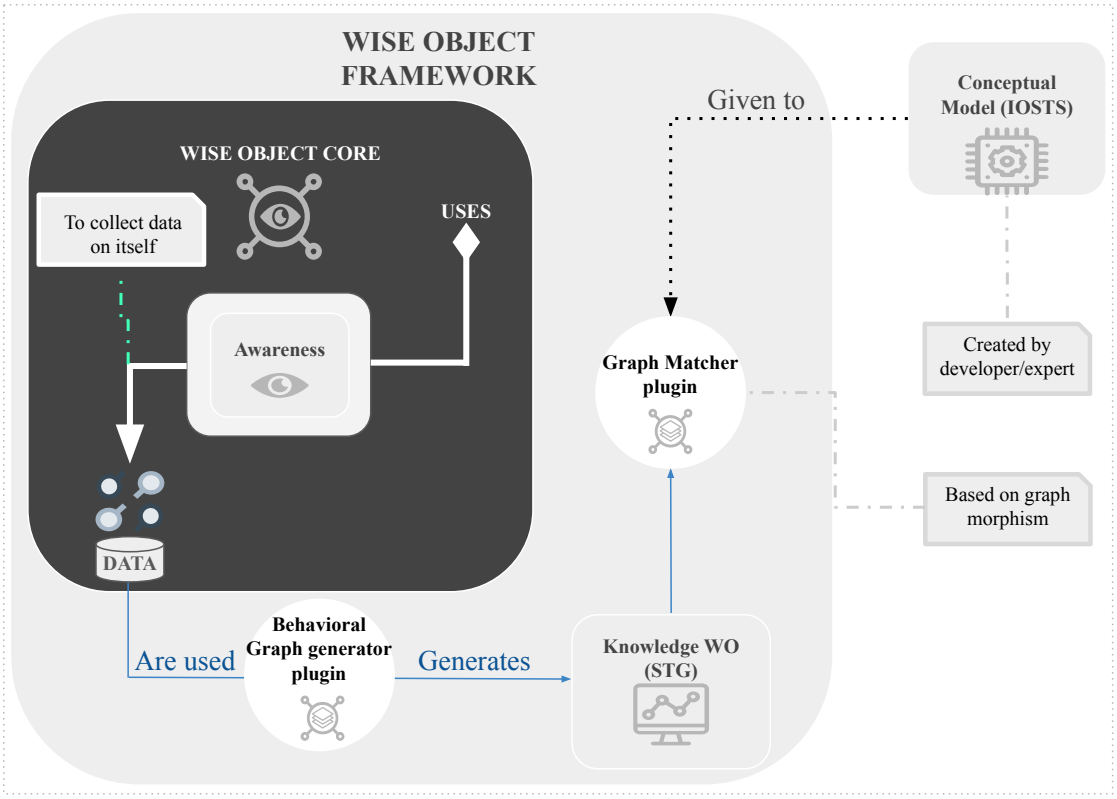


FIGURE 4.4 : Vue d'ensemble du processus de la figure 4.3.

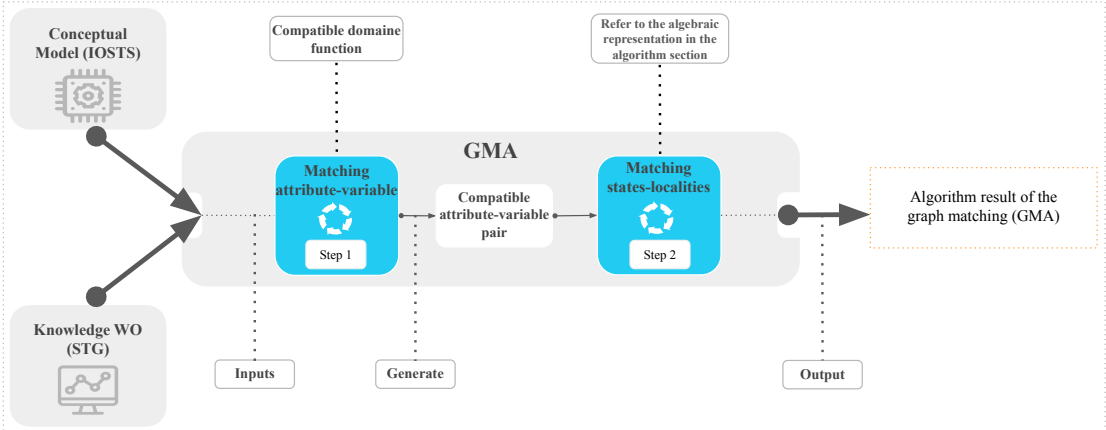


FIGURE 4.5 : Illustration de l'algorithme d'appariement par étapes appliqué à un volet roulant (résultat illustré dans la figure 4.6).

Algorithm 1 Variable matching algorithm (« compatibleDomain » function)

Inputs :
  $iosts$: IOSTS,
  stg : Exhaustive STG
Outputs :
  # Set of possible equivalent attribute/variable pairs such that $Dom(attribute) \subseteq Dom(variable)$
  Eq : Set Of Tuples< (attribute, variable) >
# Build all possible equivalent attributes-variables using the cartesian product between A and X
for $cartesian \in product(stg.A, iosts.X)$ **do**
# Keep attribute-variable pairs, such that $Dom(a) \subseteq Dom(x)$
  if $Dom(cartesian.getAttribute()) \subseteq Dom(cartesian.getVariable())$ **then**
    $Eq.add(cartesian)$
  end if
end for
return Eq

cas réel, et étant donné que nous avons fixé des contraintes d'unicité (un seul attribut et une seule variable) dans cette section, nous n'utilisons qu'un STG/IOSTS avec un seul couple attribut/variable [DAMV23a].

Première illustration d'appariement : volet roulant

L'objet connecté est un volet roulant qui a un mouvement simple, l'ouverture et la fermeture, représenté respectivement par les fonctions open() et close(). Il nécessite un seul attribut/variable (level/height) sans tenir compte de l'orientation de ses lamelles. Par ailleurs, le STG utilise des valeurs discrètes avec un niveau d'ouverture de 50% – level –, tandis que l'IOSTS utilise des intervalles continus pour la variable hauteur – height –, sans aucune contrainte sur le pas – step – qui est une valeur réelle (figure 4.6).

La figure 4.6 illustre le résultat de la mise en correspondance des deux graphes à l'aide de l'API d'appariement de graphes implémenté en Python. Les localités de l'IOSTS sont *Closed* et *Open*, chacune contenant des variables avec des domaines disjoints, dans notre exemple, une seule variable nommée *height* qui prend des valeurs différentes en fonction de sa localité. Conformément aux contraintes de l'algorithme d'appariement :

1. Il existe des caractéristiques équivalentes entre le STG et l'IOSTS, l'attribut « level » et la variable « height », respectivement.

Algorithm 2 Main algorithm : Graph matching algorithm

Inputs :
  $iosts$: IOSTS,
  stg : Exhaustive STG
Outputs :
  $match$: Dictionary<state, locality>
Locales :
  # Set of possible equivalent attribute/variable pairs
  Eq : Set Of Tuples< (attribute, variable) >
  # Possible matches for each possible equivalent pair
  M : Dictionary< (attribute, variable), <state, locality>>
Initialize :
  # Build possible equivalent attribute/variable pairs,
  # such that $dom(a) \subseteq dom(x)$ (Algorithm 1)
  $Eq \leftarrow compatibleDomain(stg.A, iosts.X)$
for $(a, x) \in Eq$ **do**
  for $v_i \in stg.V$ **do**
# Get the locality where the domain of variable x contains
# the value of a in v_i, according to the second constraint, q_i is unique
    $q_i \leftarrow iosts.getLocality(x, v_i.getValue(a))$
    $M((a, x))(v_i) \leftarrow q_i$
  end for
# As the matching is a surjective application,
# remove the pair if it does not generate surjective matching
  if $M((a, x)).getKeys() \neq stg.V$
  or $M((a, x)).getValuesAsSet() \neq iosts.Q$ **then**
    $Eq.remove((a, x))$
    $M.remove((a, x))$
  else
# If the application is surjective (Equation (4.6)),
# check the transitions' consistency (Equation (4.7))
    for $e \in stg.E$ **do**
      $v_1 \leftarrow e.getSource()$
      $v_2 \leftarrow e.getDestination()$
      $q_1 = M((a, x))(v_1)$
      $q_2 = M((a, x))(v_2)$
      if $iosts.getTransition(q_1, q_2)$ **is null then**
        $Eq.remove((a, x))$
        $M.remove((a, x))$
      end if

```
        end for
    end if
end for
# Checking that just only one matching exists according to constraints
# defined in Section 4.3.1
if M.getKeys().size() == 1 then
    match ← M.getValues()[1]
else
    exception('Requiredconditionsnotsatisfied')
end if
return match
```

2. Les domaines de la variable « height » dans les différentes localités sont disjoints les uns des autres : dans la localité *Closed*, la variable ne peut prendre que la valeur 0, et dans la localité *Open*, la variable peut prendre n'importe quelle valeur dans l'intervalle $]0, 100]$.

Du côté du STG, il y a trois états, chacun étiqueté avec un ensemble de couples attribut-valeur (att, value). Dans notre cas, l'attribut unique *level* prend les valeurs $(0, 50, 100)$, respectivement, pour (v_0, v_1, v_2). Par conséquent, pour établir un lien entre les deux graphes, il faut comparer le domaine de définition de l'attribut *level* dans chaque état du STG avec le domaine de définition de la variable *height* dans chaque localité de l'IOSTS. Cette comparaison m'a conduit à un appariement de l'état v_0 avec la localité *Closed*, signifiant que le volet roulant est fermé, et un appariement des états v_1 et v_2 avec la localité *Open*, signifiant que le volet roulant est ouvert.

Deuxième illustration d'appariement : ampoule de couleurs RGB

La deuxième illustration concerne une ampoule connectée avec des couleurs RGB. Pour simplifier l'illustration, seules deux couleurs sont prises en compte et l'état éteint de l'ampoule n'est pas considéré. La figure 4.7 montre les deux graphes comportementaux simplifiés de l'ampoule : STG et IOSTS. Le premier est défini par l'attribut « specter » $(A = \{specter\})$ et 2 méthodes « upFrequence » et « downFrequence » $(M = \{downFrequence(), upFrequence()\})$. Les méthodes « upFrequence » et « downFrequence » augmentent et diminuent le spectre de 40nm, respectivement. Le second graphe contient deux localités *Green* et *Blue*, chacune contenant des variables avec des domaines disjoints, dans notre exemple, une seule variable nommée *wavelength* qui prend des valeurs différentes en fonction de sa localité. Conformément aux contraintes de l'algorithme d'appariement :

1. Il existe des caractéristiques équivalentes entre le STG et l'IOSTS, l'attribut

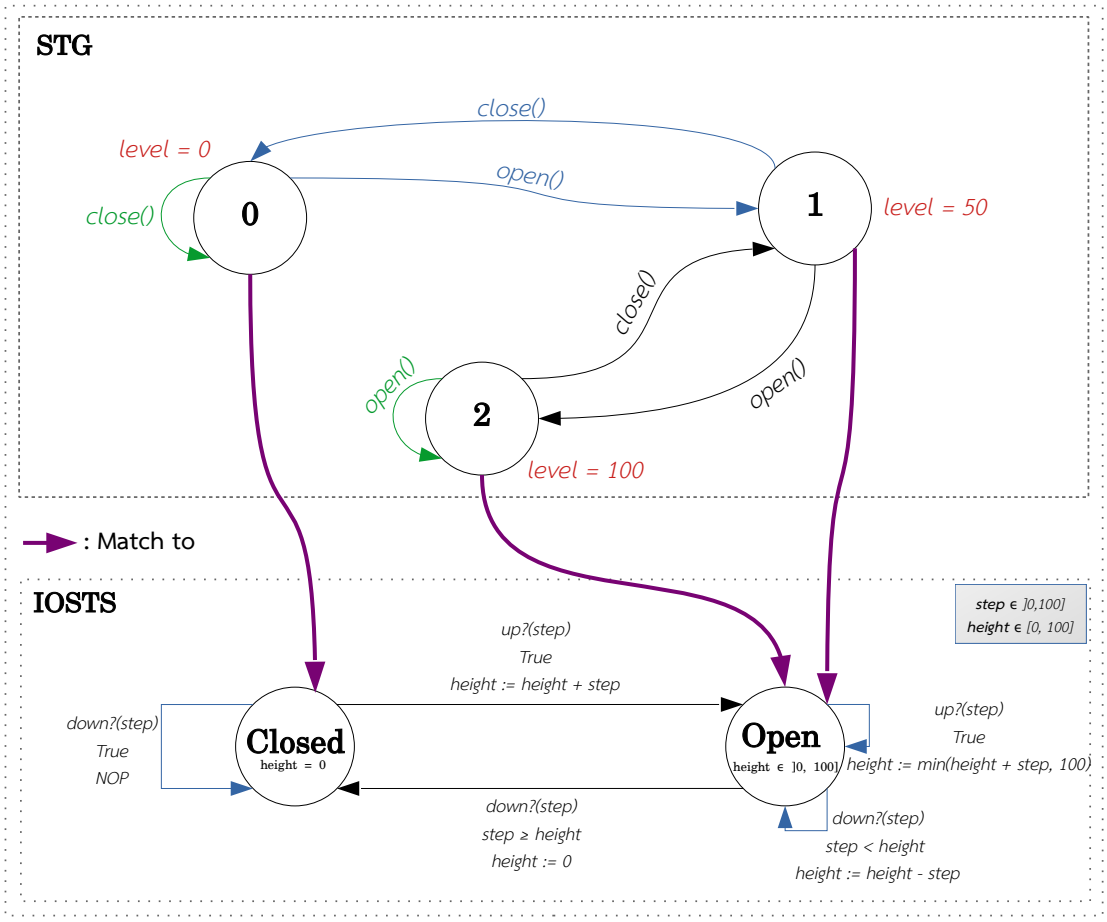


FIGURE 4.6 : Résultat de l'algorithme d'appariement de graphes (volet roulant).

« specter » et la variable « wavelength », respectivement,

2. Les domaines de la variable « wavelength » dans les différentes localités sont disjoints les un des autres : dans la localité *Green*, la variable peut prendre n'importe quelle valeur dans l'intervalle $]495, 570]$ et dans la localité *Blue*, la variable peut prendre n'importe quelle valeur dans l'intervalle $[450, 495]$.

L'algorithme conduit à un appariement d'états v_0, v_1 avec la localité *Blue* signifiant que l'ampoule est de couleur bleue, et un appariement d'états v_2, v_3 avec la localité *Green* signifiant que l'ampoule est de couleur verte.

Dans sa première version, l'algorithme présente manifestement plusieurs limites, la plus importante concerne le nombre limité d'attributs/variables équivalents dans les STG/IOSTS, à savoir un seul attribut-variable. Une autre limitation est la contrainte de l'existence d'une seule correspondance entre un STG et un IOSTS (ce qui est très loin de la réalité). Dans la section suivante, nous levons ces restrictions en généralisant l'algorithme pour qu'il soit un appariement à plusieurs variables.

4.4 Algorithme d'appariement multivarié

Dans cette section, je généralise l'algorithme présenté dans la section précédente et l'étends davantage pour lever les restrictions citées ci-dessus. Ainsi, de nombreux attributs/variables des deux graphes (STG et IOSTS) seront pris en compte cette fois-ci, ce qui donnera lieu à des appariements différents. Pour illustrer le processus complet, je montrerai la construction d'un exemple d'appariement de graphes sur un exemple plus concret.

4.4.1 Contraintes et algorithme d'appariement

Contraintes : comme nous généralisons l'algorithme, le STG et l'IOSTS doivent répondre à de nouveaux critères pour appliquer correctement le nouvel algorithme d'appariement.

1. Considérant que nous avons dans l'ensemble des attributs A un sous-ensemble non vide appelé $A_e \neq \emptyset$ et dans l'ensemble des variables X un sous-ensemble non vide appelé $X_e \neq \emptyset$. De plus, considérons $\mathcal{R}$, la relation binaire de A_e dans X_e, qui est une bijection ($\rightarrowtail\!\!\!\!\!\rightarrow$) [Mas06] :

   - Chaque élément de A_e doit être lié exactement à un élément de X_e,
   - Chaque élément de X_e doit être lié exactement à un élément de A_e.

$$\begin{array}{c} \exists! A_e \subseteq A, \exists! X_e \subseteq X | A_e \rightarrowtail\!\!\!\!\!\rightarrow X_e \\ \Leftrightarrow \\ (A_e, X_e), \end{array} \tag{4.8}$$

où (A_e, X_e) signifie que A_e et X_e représentent la même information. La solution d'appariement est donnée par $(A_e, X_e, \mathcal{R})$. Par conséquent, tout couple att_e, x_e tel que $att_e \mathcal{R} x_e$ noté (att_e, x_e) exprime le fait que att_e et x_e représentent la même information, ce qui signifie que :

$$Dom(att_e) \subseteq Dom(x_e). \tag{4.9}$$

En outre, l'équation 4.9 nous permet de déduire ce qui suit :

- Dans le cas où $Dom(att_e) \subset \mathbb{R}$:

$$\begin{array}{c} (att_e, x_e) \\ \Leftrightarrow \\ min(Dom(att_e)) \geq min(Dom(x_e)) \wedge \\ max(Dom(att_e)) \leq max(Dom(x_e)). \end{array}$$

- Dans le cas où $Dom(att_e) \subset \mathbb{S} \mid \mathbb{S}$ est l'ensemble des chaînes de caractères :

$$\begin{array}{c} (att_e, x_e) \\ \Leftrightarrow \\ \forall value_i \in Dom(att_e), value_i \in Dom(x_e). \end{array}$$

2. Chaque localité de l'IOSTS doit être unique en ne prenant en compte que les variables de l'ensemble X_e. Ainsi, les domaines de X_e dans les différentes localités de l'IOSTS sont disjoints :

$$\begin{array}{c} \forall q, q' \in Q \mid q \neq q' \\ \Leftrightarrow \\ dom(q, X_e) \cap dom(q', X_e) = \emptyset. \end{array}$$

Algorithme : l'algorithme se compose de plusieurs étapes utilisant la théorie des ensembles. L'algorithme est exponentiel et va parcourir tous les attributs et variables, états et localités, ce qui va nous donner plusieurs résultats d'appariement qui ne sont pas tous corrects du point de vue humain. La réduction du coût d'appariement par le choix automatique d'un seul appariement correct sans calculer les autres appariements est prévue dans le chapitre suivant, par l'utilisation des ontologies et de la structure matricielle des graphes. Notez que ce problème est classé NP-difficile.

- P est l'ensemble qui contient tous les couples attribut-variable potentiellement équivalents en fonction de leurs domaines :

$$P = \{(att, x) \mid att \in A, x \in X, Dom(att) \subseteq Dom(x)\}. \tag{4.10}$$

- $\mathcal{P}(P)$ est l'ensemble des parties – power set – de P [SSSK15], il contient tous les sous-ensembles c de P :

$$\forall c \subseteq P, c \in \mathcal{P}(P), \tag{4.11}$$

sachant que $|\mathcal{P}(P)| = 2^{|P|}$, et que A_e et X_e ne sont pas vides, l'ensemble vide peut être retiré de $\mathcal{P}(P)$:

$$\mathcal{P}_{\not\emptyset} = \mathcal{P}(P) \backslash \{\emptyset\}.$$

- $\mathcal{P}_v$ est l'ensemble des combinaisons c valides de couples attribut-variable dans $\mathcal{P}_{\not\emptyset}$. Une combinaison c est valide, si et seulement si, elle représente une bijection entre ses attributs et ses variables.

$$\mathcal{P}_v = \{c \mid c \in \mathcal{P}_{\not\emptyset}, \forall (att_i, x_j) \in c, (att_i, x_k) \notin c \wedge (att_l, x_j) \notin c\},$$

avec $i \neq l$ et $k \neq j$.

- $\mathcal{P}_m$ est l'ensemble des combinaisons c maximisées des couples attribut-variable dans $\mathcal{P}_v$:

$$\mathcal{P}_m = \{c \mid c \in \mathcal{P}_v, \forall c_j \in \mathcal{P}_v, c \not\subset c_j\}. \tag{4.12}$$

sachant que les combinaisons maximisées sont les plus longues combinaisons telles que tout sous-ensemble d'une combinaison n'existe pas dans l'ensemble des combinaisons $\mathcal{P}_m$.

$\mathcal{P}_m$ regroupe les combinaisons maximisées en fonction du domaine de définition des attributs et des variables.

Comme il existe pour chaque état une localité unique, tel que pour tout couple d'attribut-variable d'une combinaison, les valeurs de l'attribut sont incluses dans le domaine de la localité, nous ne conservons dans $\mathcal{P}_m$ que les combinaisons qui vérifient cette propriété. Par conséquent, $\mathcal{P}_{VQ}$ garde les combinaisons possibles qui correspondent à un appariement valide entre l'ensemble des états V et l'ensemble des localités Q. Formellement :

$$\begin{aligned} \mathcal{P}_{VQ} = \{c \mid c \in \mathcal{P}_m, \exists! v \in V, \exists! q \in Q, \forall (att, x) \in c, \\ dom(v, att) \subseteq dom(q, x)\}. \end{aligned} \tag{4.13}$$

De $\mathcal{P}_{VQ}$, on peut déduire tous les appariements possibles $\mathcal{M}_{VQ}$ qui regroupent les ensembles de couples d'état-localité :

$$\begin{aligned}\mathcal{M}_{VQ} = \{m_i \mid \forall c_i \in \mathcal{P}_{VQ}, \exists! v \in V, \exists! q \in Q, \\ \forall (att, x) \in c_i, dom(v, att) \subseteq dom(q, x), \\ (v, q) \in m_i\}.\end{aligned} \tag{4.14}$$

Ces appariements $\mathcal{M}_{VQ}$ sont valides en ce qui concerne les couples attribut att variable x, et les couples état v localité q. L'algorithme d'appariement étant basé sur le morphisme des graphes, il doit respecter la structure des graphes appariés [GC97]. Dans notre contexte, les images des états du STG dans l'IOSTS - les localités - doivent respecter les relations d'adjacence présentes dans le STG (c'est-à-dire les transitions). En d'autres termes, deux états adjacents doivent correspondre à la même localité (vu les relations réflexives dans l'IOSTS entre une localité et elle-même) ou à deux localités adjacentes. Considérons $\mathcal{S}$ l'application surjective du STG dans l'IOSTS, respectivement, entre les états V et les localités Q (voir l'équation 4.6), c'est-à-dire que chaque état correspond à une localité et qu'une localité correspond à au moins un état. Pour toute transition $(u, v) \in E$ de STG, $(\mathcal{S}(u), \mathcal{S}(v)) \in T$ est une transition dans l'IOSTS (voir l'équation 4.7). L'appariement STG $\rightarrow$ IOSTS est un homomorphisme surjectif, c'est-à-dire un épimorphisme [GC97].

Conformément à l'équation 4.7, si l'état v' est le voisin de l'état v et que la localité q correspond à v, alors v' doit correspondre à q ou à un voisin de q :

$$\begin{aligned}\mathcal{M}_{\mathcal{S}} = \{m \mid m \in \mathcal{M}_{VQ}, \forall (v, q) \in m, \forall v' \in neighbors(v), \\ \exists q' \in neighbors(q) \cup \{q\}, (v', q') \in m\}.\end{aligned} \tag{4.15}$$

4.4.2 Algorithme d'appariement par étapes

L'algorithme d'appariement recevra deux types de représentation de la connaissance (STG et IOSTS) et se déroule automatiquement en trois étapes résumées dans la figure 4.8. La première détermine tous les couples d'appariement attribut-variable possibles P (équation 4.10). Le résultat sera utilisé dans la deuxième étape pour construire toutes les combinaisons maximums $\mathcal{P}_m$ (équation 4.12), qui seront utilisées pour construire l'appariement entre les états et les localités $\mathcal{M}_{\mathcal{S}}$ à la troisième étape (équation 4.15).

4.4.3 Illustration

Pour cette illustration, le même volet roulant que dans l'illustration précédente est utilisé, sans omettre l'orientation de ses lamelles, mais en gardant un comportement

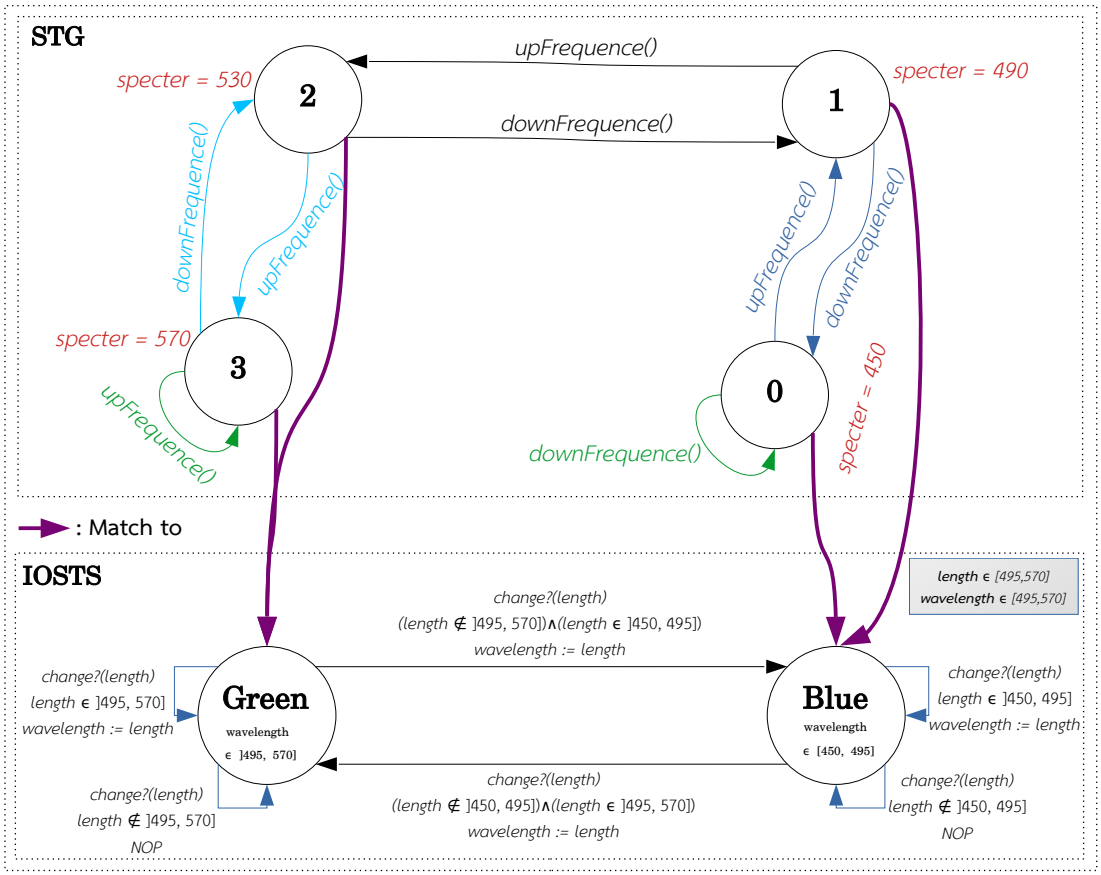


FIGURE 4.7 : Résultat de l'algorithme d'appariement de graphes (ampoule).

simplifié pour mettre en évidence l'algorithme, sachant que l'implémentation peut traiter des comportements complexes. Le STG est généré par un WO et 'IOSTS est donné par un développeur/expert. Les deux graphes représentent le même comportement du volet roulant. Le STG utilise des valeurs discrètes avec un niveau d'ouverture de 50% et une orientation des lamelles de 90° tandis que l'IOSTS utilise des intervalles continus, sans aucune contrainte sur le pas – step – qui est une valeur réelle.

Les figures 4.9a et 4.9b illustrent tous les résultats d'appariement possibles des deux graphes (STG et IOSTS) obtenus à partir de l'API d'appariement/Matching que j'ai développé. Les localités de l'IOSTS sont *Closed* et *Open*, chacune contient des variables avec des domaines disjoints (voir équation 2.5), dans notre exemple, les deux variables *height* et *angle*.

Conformément aux contraintes de l'algorithme d'appariement données dans la section 4.4.1 :

1. il existe des attributs/variables potentiellement équivalents entre le STG et l'IOSTS, plus précisément, entre les attributs « level, orientation » et les variables « height, angle ». Selon l'équation 4.9, les couples suivants représentent des équivalences potentielles d'attributs/variables :

$$(level, height),$$
$$(orientation, angle),$$
$$(orientation, height),$$

2. les domaines des deux localités *Closed* et *Open* vérifient l'équation 2.5. Ainsi, *Closed* et *Open* sont disjoints.

En ce qui concerne le STG, il y a quatre états, chacun étiqueté avec un ensemble de couples attribut-valeur $(att, value)$. Dans notre cas, la paire $(level, orientation)$ prend les valeurs $[(50, 90),\ (100, 90),\ (50, 0),\ (0, 0)]$, respectivement pour $(v_0,\ v_1,\ v_2,\ v_3)$. Par conséquent, pour établir un appariement entre les deux graphes, pour toutes les combinaisons $\mathcal{P}_m$, le domaine de définition du couple $(level, orientation)$ dans chaque état du STG doit être comparé au domaine de définition du couple de variables $(height, angle)$ dans chaque localité de l'IOSTS. Cette comparaison donne $\mathcal{P}_{VQ}$, ce qui implique implicitement $\mathcal{M}_{\mathcal{S}}$.

Concrètement, on obtient les ensembles suivants :

$$P = \{(level, height), (orientation, angle),$$
$$(orientation, height)\}.$$

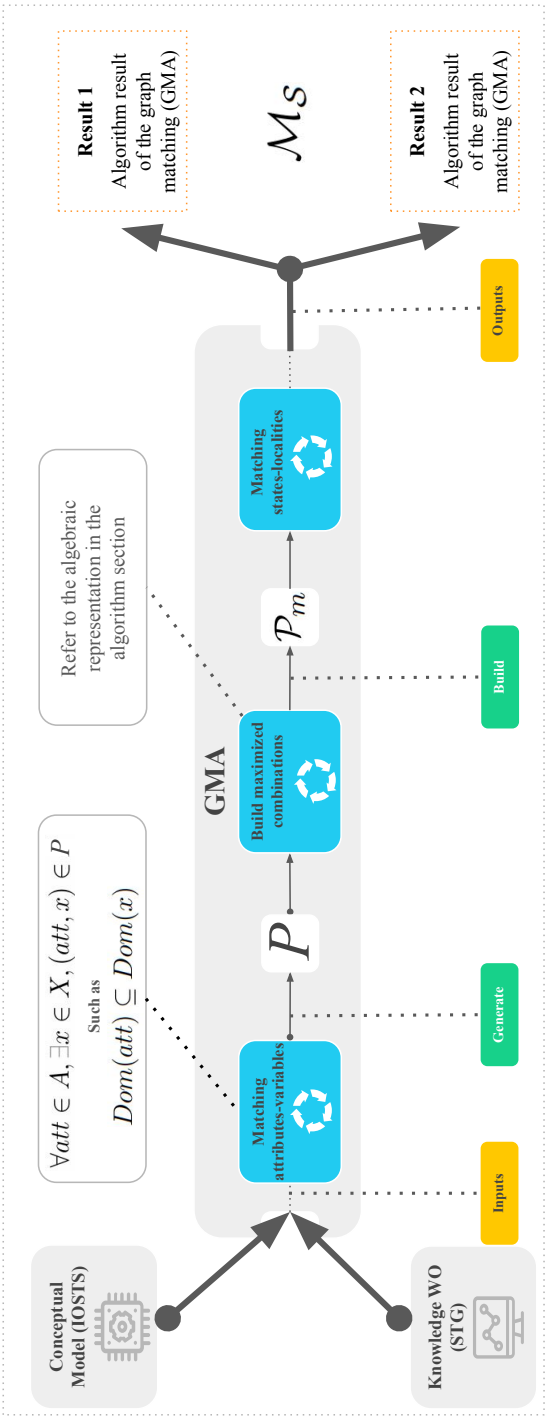
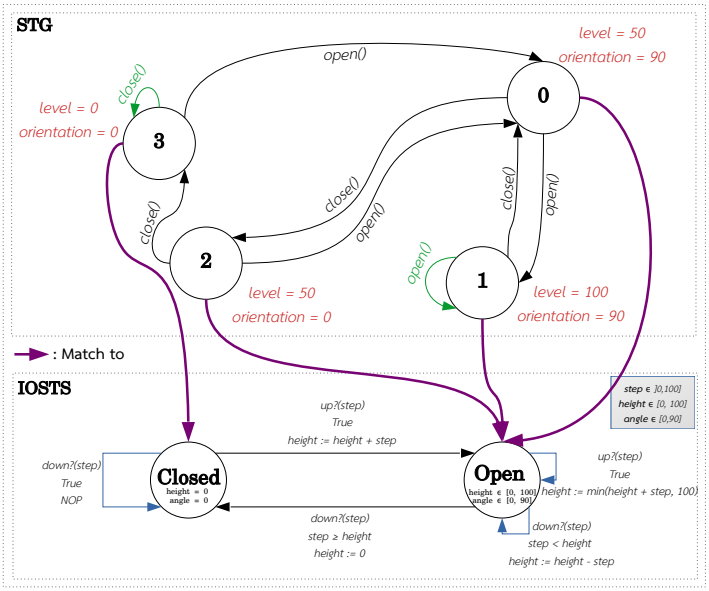
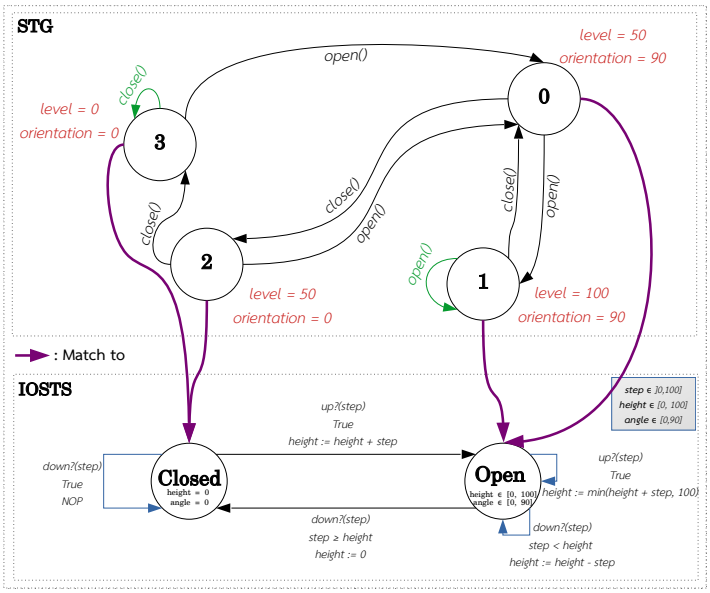


FIGURE 4.8 : Illustration de l'algorithme d'appariement par étapes (résultats dans les figures 4.9a et 4.9b).



(a) Résultat de l'algorithme d'appariement $\mathcal{P}_m^1 = \{(level, height), (orientation, angle)\} \in \mathcal{P}_{VQ}$.



(b) Résultat de l'algorithme d'appariement $\mathcal{P}_m^2 = \{(orientation, height)\} \in \mathcal{P}_{VQ}$.

FIGURE 4.9 : Illustration de tous les appariements $\mathcal{M}_{VQ}$ résultats de l'algorithme.

L'ensemble des parties de P :

$$\begin{aligned}\mathcal{P}_{\emptyset}(P) = \{&\{(level, height)\}, \{(orientation, angle)\},\\ &\{(orientation, height)\},\\ &\{(level, height), (orientation, angle)\},\\ &\{(level, height), (orientation, height)\},\\ &\{(orientation, angle), (orientation, height)\},\\ &\{(level, height), (orientation, angle),\\ &(orientation, height)\}\}.\end{aligned}$$

Ainsi, l'ensemble des combinaisons valides :

$$\begin{aligned}\mathcal{P}_v = \{&\{(level, height)\}, \{(orientation, angle)\},\\ &\{(orientation, height)\},\\ &\{(level, height), (orientation, angle)\}\}.\end{aligned}$$

Par conséquent, l'ensemble des combinaisons maximisées :

$$\begin{aligned}\mathcal{P}_m = \{&\{(orientation, height)\},\\ &\{(level, height), (orientation, angle)\}\}.\end{aligned}$$

Puisque toutes les combinaisons maximisées sont conformes à l'équation 4.13, $\mathcal{P}_{VQ} \equiv \mathcal{P}_m$, ce qui donne deux appariements

$$\begin{aligned}&\mathcal{M}_{VQ} = \{\\ &\ \{(v_0, Open), (v_1, Open), (v_2, Open), (v_3, Closed)\},\\ &\ \{(v_0, Open), (v_1, Open), (v_2, Closed), (v_3, Closed)\}\\ &\}.\end{aligned}$$

Comme les deux appariements sont surjectifs dans cette illustration, $\mathcal{M}_{VQ} \equiv \mathcal{M}_{\mathcal{S}}$.

Cet exemple fournit deux appariements possibles et l'algorithme ne peut pas déterminer lequel correspond à (A_e, X_e). Des informations supplémentaires sont nécessaires pour déterminer la bonne correspondance. Ces informations peuvent être fournies par l'utilisateur final ou, comme je le présente dans le prochain chapitre, déterminées à partir de la signification des attributs et des variables à l'aide d'une ontologie.

4.5 Conclusion

Dans ce chapitre, nous avons présenté quelques types d'appariement de graphes pris de la littérature avant de passer au sujet principal, à savoir l'appariement de

deux graphes de nature différente sur la base de leurs attributs et de leurs variables. Ce chapitre divise la problématique en deux sujets, à savoir l'appariement univarié et l'appariement multivarié, deux techniques qui semblent identiques dans leur forme, mais très différentes dans leur formalisme. Comme nous le savons déjà, il existe deux types d'appariement de graphes, exact et inexact, et dans notre cas, nous avons les deux dans un seul résultat. Pour bien comprendre la situation, il faut considérer l'appariement des deux points de vue : celui de la machine (c'est-à-dire numérique et structurel) et celui de l'homme (c'est-à-dire sémantique). Selon la machine, et puisque l'appariement préserve la structure et les transitions entre les deux formalismes, l'appariement est toujours exact entre les « états » et les « localités », ce qui donne un épimorphisme (équations 4.6 et 4.7). Cependant, du point de vue humain, dans la plupart des cas réels, il y aura au plus un appariement exact conforme à la sémantique. Comme le montrent les illustrations (figures 4.9a et 4.9b), un seul appariement sera exact. Le problème d'appariement exact est un grand défi, mes travaux dans le chapitre suivant se concentrent sur ce problème en prenant en compte, en plus de la perspective numérique, la perspective sémantique, à savoir, la conscience sémantique.

Sommaire

Chapitre

5

Vers une capacité de décision quasi humaine : GMA et ontologie

La prise de décision humaine est depuis toujours fortement influencée par des processus mentaux inconscients qui produisent parfois de bons résultats rapidement, mais qui nous amènent parfois à faire des choix subjectifs. Pour prendre une décision, les êtres humains peuvent implicitement rechercher et utiliser des structures visuelles, des expériences, des apprentissages et des émotions [Joh21]. D'un autre côté, de nombreux scientifiques issus d'un large éventail de domaines (mathématiques, psychologie, ingénierie, économie et sciences politiques) ont compris ce schéma et ont commencé à discuter de la possibilité de créer un cerveau

artificiel [Kap22]. En 1956, le terme intelligence artificielle (IA) a été officiellement inventé par McCarthy [McC79] lors de la conférence du projet de recherche d'été de Dartmouth sur l'IA. Ceci donnera plus tard naissance à l'IA en tant que domaine de recherche autonome.

Dans ce chapitre, j'aborde la prise de décision du point de vue de la machine (WO). Pour ceci, l'algorithme d'appariement de graphes – Graph Matching Algorithm (GMA) – (chapitre 4) est étendu pour permettre aux WO de décider quel appariement est correct.

5.1 Ontologies

L'IA est basée sur un processus de pensée humaine qui peut être mécanisé en utilisant le raisonnement formel établi par les anciens philosophes [Ber01], tels qu'Aristote, Euclide, Al-Khawarizmi, etc. Les informaticiens ont utilisé les outils inventés par les mathématiciens (par exemple, les lois statistiques, la recherche opérationnelle, les méthodes de classification, etc.) pour créer de nombreux systèmes (par exemple, l'exploration de données, l'analyse textuelle, l'apprentissage automatique, etc.). Mais l'IA moderne présente de nombreuses lacunes, car d'une manière ou d'une autre, les humains sont principalement impliqués dans son processus de fonctionnement, ce qui les charge d'informations inutiles pour eux, mais importantes pour que les systèmes d'IA prennent les bonnes décisions en fonction du contexte. C'est pourquoi les WO et WS dotés de capacités spécifiques ont été créés, ils ont pour objectif d'avancer d'un cran vers les systèmes d'IA calmes (voir chapitre 1).

Jusqu'à présent, tout WO peut se connaître, connaître son utilisation et améliorer ses capacités par lui-même. Le défi qui a suivi cette approche comme mentionné dans le contexte se résume en deux questions :

- Comment rendre un WO capable de communiquer avec les humains ?
- Quelles sont les limites d'un tel WO ?

La première question est résolue dans nos premières propositions [DAMV22b, DAMV23a, DAMV23b], généralement intitulées, « Algorithme d'appariement de graphes », et détaillées dans le chapitre 4. Mais la deuxième question reste un grand défi, puisque la limite illustrée précédemment est que les WO ne sont pas conscients du contexte sémantique – semantic-awareness (définition 3.1.4) – et n'arrivent pas à se décider sur le seul appariement correct. En identifiant la situation/domaine d'une application, il sera facile de définir un contexte précis (la sémantique selon le contexte) pour permettre aux WO de choisir le bon appariement.

La définition du contexte sémantique précis d'une application ne sera pas abordée dans cette thèse, en revanche, je commence par utiliser la sémantique lexicale

telle qu'elle est présentée dans l'ontologie WordNet [Mil95a]. Dans les travaux futurs, je considérerai un contexte sémantique plus spécifique, afin de calculer de manière plus précise l'appariement correct.

5.2 Appariement de domaines

Dans cette étude, nous ne considérons que les attributs et les variables numériques. Les domaines des attributs et des variables peuvent être déduits respectivement du STG et de l'IOSTS. Le défi consiste à trouver les sous-ensembles d'attributs équivalents aux sous-ensembles de variables, respectivement, parmi tous les variables et attributs, afin de modéliser des combinaisons permettant d'établir des appariements. L'équivalence signifie que chaque couple attribut/variable équivalent représente la même information.

5.2.1 Matrice de compatibilité entre les domaines

La première étape consiste à vérifier la compatibilité entre chaque attribut et chaque variable en fonction de leur domaine. Un attribut a est compatible avec une variable x si, et seulement si, $\mathrm{dom}(att) \subseteq \mathrm{dom}(x)$ (cette inclusion sera transformée en matrice de compatibilité). Deux cas sont possibles :

1. Cas 1 : aucune variable et aucun attribut ne sont compatibles, dans ce cas, le STG et l'IOSTS ne peuvent pas être appariés. Les deux modèles ne sont pas compatibles et le processus s'arrête, aucune sémantique ne peut être ajoutée au STG avec l'IOSTS fourni.

2. Cas 2 : un ou plusieurs ensembles d'attributs sont compatibles avec de nombreux ensembles de variables. Dans ce cas, le WO doit passer en mode semi-supervisé et proposer à l'utilisateur les appariements les plus probables

Nous définissons le degré de compatibilité du domaine comme le pourcentage de chevauchement entre le domaine de l'attribut et celui de la variable, sous la condition principale, $\mathrm{dom}(att) \subseteq \mathrm{dom}(x)$. L'équation 5.1 donne le degré de compatibilité entre un attribut att et une variable x. Notons que nous avons séparé la situation où $\mathrm{dom}(att) = \mathrm{dom}(x)$ de la situation où $\mathrm{dom}(att) \subset \mathrm{dom}(x)$ afin d'éviter la division par 0 dans le cas d'un domaine avec une seule valeur. Par exemple, $dom(att) = [i, i]$ et $dom(x) = [i, i]$, où les domaines de att et x ont tous les deux une seule valeur i.

$$d(att, x) = \begin{cases} \frac{\max(\mathrm{dom}(att)) - \min(\mathrm{dom}(att))}{\max(\mathrm{dom}(x)) - \min(\mathrm{dom}(x))}, & \text{iff} \quad \mathrm{dom}(att) \subset \mathrm{dom}(x). \\ 1, & \text{iff} \quad \mathrm{dom}(att) = \mathrm{dom}(x). \\ 0, & \text{otherwise.} \end{cases} \tag{5.1}$$

De l'équation 5.1, la matrice des degrés – Compatibiliy Matrix – entre les ensembles A et X se déduit comme suit :

$$D_{(A,X)} = \begin{bmatrix} d(att_1, x_1) & d(att_1, x_2) & \dots & d(att_1, x_{|X|}) \\ d(att_2, x_1) & d(att_2, x_2) & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ d(att_{|A|}, x_1) & d(att_{|A|}, x_2) & \dots & d(att_{|A|}, x_{|X|}) \end{bmatrix} \tag{5.2}$$

Si D est une matrice nulle, D fait référence au cas 1. Sinon, D représente le cas 2, et contiendra des entrées entre 0 et 1 $\iff (0 \leq D_{ij} \leq 1)$. Notons $D_{ij} = d(att_i, x_j)$.

En partant de la matrice D, l'objectif est de trouver tous les sous-ensembles (combinaisons) de couples attribut/variable qui peuvent être des appariements valides, à savoir, qui respectent la propriété 4.9. Pour cela, la première étape consiste à trouver tous les arrangements possibles entre les lignes ou colonnes de la matrice D selon les conditions expliquées dans la section qui suit.

5.2.2 Tous les arrangements possibles

Comme évoqué précédemment, l'objectif est de calculer toutes les combinaisons possibles. À cette fin, nous avons d'abord besoin de l'ensemble de tous les arrangements possibles. Pour calculer toutes les combinaisons possibles, en utilisant la matrice D, il convient simplement de calculer tous les arrangements d'indices, de lignes ou de colonnes et de les utiliser de la manière suivante. Considérons l'ensemble $\mathcal{A}_n^r$ comme l'ensemble des arrangements – All r-subsets – d'indices de colonnes dans le cas où $|A| \geq |X|$, ou de lignes dans le cas contraire. Sachant qu'un arrangement $\mathcal{A} \in \mathcal{A}_n^r$ se compose de r éléments. Pour résumer, deux cas peuvent se présenter, le nombre d'attributs $|A|$ est supérieur au nombre de variables $|X|$, ou l'inverse. Le premier cas est calculé avec les arrangements de lignes, donnés par l'équation (5.3a), le second est calculé par les arrangements de colonnes, donné par l'équation (5.3b).

$$|\mathcal{A}_n^r| = \frac{n!}{(n-r)!} \quad : \begin{cases} n = |A|, r = |X|, & \text{pour } |A| \geq |X| \quad (5.3a) \\ n = |X|, r = |A|, & \text{sinon .} \quad (5.3b) \end{cases}$$

Exemple générique 5.2.1 *:* Pour illustrer la matrice des degrés et les arrangements (5.2.1 et 5.2.2) en vue d'une meilleure compréhension. Considérons l'exemple générique suivant : le STG d'une entité avec 3 attributs att_1, att_2 et att_3 ($|A| = 3$), et un IOSTS qui modélise le comportement de cette entité avec 2 variables ($|X| =$

2), appelées x_1 et x_2. La matrice des degrés correspondante D est donnée par :

$$D_{(A,X)} = \begin{bmatrix} d(att_1, x_1) & d(att_1, x_2) \\ d(att_2, x_1) & d(att_2, x_2) \\ d(att_3, x_1) & d(att_3, x_2) \end{bmatrix}.$$

Comme $|A| \geq |X|$ dans l'exemple générique, le nombre d'arrangements est calculé selon l'équation 5.3. Ainsi, les arrangements possibles $\mathcal{A}_3^2$ sont opérés sur les indices des lignes [Pia38] et présentés par indices comme suit :

$$\mathcal{A}_3^2 = \{(1,2),(1,3),(2,3),(3,2),(3,1),(2,1)\},$$

ou bien en représentation explicite (nom des attributs) comme suit :

$$\mathcal{A}_3^2 = \{(att_1, att_2),(att_1, att_3),(att_2, att_3),(att_3, att_2),(att_3, att_1),(att_2, att_1)\},$$

sachant que tous les arrangements dans $\mathcal{A}_3^2$ sont différents. Cela donne une symétrie dans l'ensemble des arrangements, qui m'a permis d'éviter l'opération « Power set » sur l'ensemble de tous les couples attribut-variable que j'ai effectuée au chapitre 4 (équation 4.11), compte tenu de son énorme coût.

Après avoir construit la matrice D et trouvé tous les arrangements possibles $\mathcal{A}_n^r$, nous devons maintenant construire toutes les combinaisons possibles de couples d'attribut-variable à partir de ces arrangements.

5.2.3 Toutes les combinaisons possibles

L'ensemble des arrangements précédents représente tous les ensembles possibles de couples attribut/variable sous une forme condensée et implicite. Cette forme s'apparente à la notation de Cauchy utilisée pour les matrices de permutations [Wus07].

Une matrice de permutation est une matrice binaire dont tous les coefficients sont égaux à 0, à l'exception d'un coefficient égal à 1 dans chaque ligne et chaque colonne. Dans notre cas, une combinaison peut être représentée par une matrice binaire dont chaque ligne et chaque colonne contient au plus un coefficient égal à 1. Ce dernier représentant un couple (att, x) conservé (c'est-à-dire un couple équivalent) dans la combinaison. La notation de Cauchy à deux lignes se définit comme suit :

$$\begin{pmatrix} 1 & 2 & \dots & n \\ \sigma(1) & \sigma(2) & \dots & \sigma(n) \end{pmatrix}$$

Où (1, 2, . . . , n) représentent les lignes de la matrice des degrés et ($\sigma(1)$, $\sigma(2)$, . . . , $\sigma(n)$) représentent les lignes de permutation dans l'approche initiale (c'est-à-dire de Cauchy), ou les colonnes dans mon cas.

En utilisant les arrangements précédemment déterminés, l'ensemble des combinaisons recherchées peut être déterminé de la manière suivante. Considérons la relation σ qui associe à chaque ensemble d'attributs un ensemble de variables de taille identique. Cette relation permet de modéliser toutes les combinaisons possibles $\mathcal{C}$ de couples attribut/variable. Les équations 5.4 et 5.5 définissent, respectivement, σ comme une application de $\mathcal{A}_n^r$ à X^r et σ^{-1} comme une application de $\mathcal{A}_n^r$ à A^r, afin de définir pour chaque liste d'attributs sa liste de variables correspondante. L'utilisation de σ ou σ^{-1} est conditionnée aux tailles de A et X puisque $\mathcal{A}$ en dépend, comme introduit dans l'équation 5.3. Ainsi, grâce à l'approche par arrangements, l'image de tout $\mathcal{A}$ est constante et correspond à l'ensemble des indices de colonnes (variables) de la matrice D ordonnées si $|A| \geq |X|$, conformément à l'équation 5.3b, ou bien l'antécédent de tout $\mathcal{A}$ est constant et correspond à l'ensemble des indices de lignes (attributs) de la matrice D ordonnées, sinon conformément à l'équation 5.3a. De manière plus formelle, si $|A| \geq |X|$:

$$\begin{gathered}\sigma : \mathcal{A}_n^r \longrightarrow X^r \\ \forall \underbrace{(att_i, att_j \ldots, att_k)}_{\mathcal{A}} \in \mathcal{A}_n^r, \sigma((att_i, att_j \ldots, att_k)) = (x_1, x_2, \ldots x_r),\end{gathered} \tag{5.4}$$

sinon :

$$\begin{gathered}\sigma^{-1} : \mathcal{A}_n^r \longrightarrow A^r \\ \forall \underbrace{(x_i, x_j \ldots, x_k)}_{\mathcal{A}} \in \mathcal{A}_n^r, \sigma^{-1}((x_i, x_j \ldots, x_k)) = (att_1, att_2, \ldots att_r).\end{gathered} \tag{5.5}$$

À partir de σ ou σ^{-1}, $\mathcal{C}$ est défini de la manière suivante :

$$\forall (att_i, att_j \ldots, att_k) \in \mathcal{A}_n^r, ((att_i, x_1), (att_j, x_2) \ldots (att_k, x_r)) \in \mathcal{C}, \tag{5.6}$$

si $|A| \geq |X|$, ou

$$\forall (x_i, x_j \ldots, x_k) \in \mathcal{A}_n^r, ((att_1, x_i), (att_2, x_j) \ldots (att_r, x_k)) \in \mathcal{C}, \tag{5.7}$$

sinon.

Notons que chaque combinaison $c \in \mathcal{C}$ contient r couples, soit le nombre d'éléments dans un arrangement $\mathcal{A} \in \mathcal{A}_n^r$:

$$\forall c \in \mathcal{C}, |c| = |\mathcal{A}| = r,$$

et que la taille de $\mathcal{C}$ est égale à la taille de $\mathcal{A}_n^r$:

$$|\mathcal{C}| = |\mathcal{A}_n^r| = \frac{n!}{(n-r)!} \mid n \geq r.$$

Cette approche est équivalente à celle proposée en section 4.4, mais elle apporte l'avantage de ne pas se baser sur un produit cartésien, plus lourd à gérer en taille et temps de traitement.

Exemple générique 5.2.2 (Suite) *:* Comme $|A| \geq |X|$, l'application 5.4 sur l'exemple générique 5.2.1 $\mathcal{A}_3^2$, donne le résultat suivant :

$$\begin{array}{lll} \mathcal{A}_3^2 & : & \Big\langle \underbrace{(att_1, att_2)}_{\mathcal{A}_1} \; \underbrace{(att_1, att_3)}_{\mathcal{A}_2} \; \underbrace{(att_2, att_3)}_{\mathcal{A}_3} \; \underbrace{(att_3, att_2)}_{\mathcal{A}_4} \; \underbrace{(att_3, att_1)}_{\mathcal{A}_5} \; \underbrace{(att_2, att_1)}_{\mathcal{A}_6} \Big\rangle \\ \sigma(\mathcal{A}_3^2) & : & \Big\langle (x_1, x_2) \; (x_1, x_2) \; (x_1, x_2) \; (x_1, x_2) \; (x_1, x_2) \; (x_1, x_2) \Big\rangle \end{array}$$

Le premier arrangement $\mathcal{A}_1$ associe les attributs att_1 et att_2, respectivement, aux variables x_1 et x_2, à savoir, $\sigma((att_1, att_2)) = (x_1, x_2)$; le deuxième arrangement, $\mathcal{A}_2$, associe les attributs att_1 et att_3, respectivement, aux variables x_1 et x_2, et ainsi de suite. Ainsi, en s'appuyant sur l'équation 5.7, l'ensemble des combinaisons possibles $\mathcal{C}$ est défini par :

$$\mathcal{C} = \left\langle \begin{array}{l} \{(att_1, x_1), (att_2, x_2)\}, \\ \{(att_1, x_1), (att_3, x_2)\}, \\ \{(att_2, x_1), (att_3, x_2)\}, \\ \{(att_3, x_1), (att_2, x_2)\}, \\ \{(att_3, x_1), (att_1, x_2)\}, \\ \{(att_2, x_1), (att_1, x_2)\} \end{array} \right\rangle. \tag{5.8}$$

Nous disposons à présent de toutes les combinaisons possibles de couples attribut/variable. La question est de savoir quelles sont les combinaisons les plus susceptibles d'aboutir à un appariement, ce qui est l'objet de la section suivante.

5.2.4 Choix de combinaisons

Degré de compatibilité : différentes méthodes de calcul

Conformément au degré de compatibilité des domaines de chaque couple donné dans la matrice D (matrice 5.2), Le degré de compatibilité des domaines d'un ensemble de couples, c'est-à-dire une combinaison, se calcule en fonction de la matrice D. Cette valeur doit satisfaire deux contraintes :

- Dans une combinaison $c \in \mathcal{C}$, si le degré de compatibilité des domaines d'un couple n'est pas nul, le degré de compatibilité des domaines de la combinaison c ne peut pas être nul. Cela signifie que si au moins un couple $p_{ik} = (a_i, x_k)$ dans une combinaison est compatible ($d(p_{ik}) \neq 0$), la combinaison est possible.

- Les couples p_{ik} avec un degré nul $d(p_{ik}) = 0$ sont pris en compte dans le calcul du degré d'une combinaison, en revanche, ils ne sont pas pris en compte dans les combinaisons possibles. Effectivement, si le degré d'un couple est nul, le couple n'est pas valide.

Il existe un certain nombre de manières pour fusionner l'ensemble des degrés des couples formant une combinaison. Une moyenne des degrés de compatibilité sur l'ensemble des couples d'une combinaison peut être considérée :

$$deg(c) = \frac{1}{r} \cdot \sum_{\forall p_{ik} \in c} d(p_{ik}) \mid c \in \mathcal{C} \tag{5.9}$$

L'avantage de cette moyenne est qu'elle prend en compte le nombre total initial r de couples. Ainsi les degrés de compatibilité des combinaisons sont naturellement comparables. Par exemple, une combinaison de deux couples ayant un degré égal à 1 sera privilégiée à une autre combinaison avec un seul couple ayant un degré égal à 1. Supposons que $d(p_{12}) = d(p_{23}) = 1, d(p_{11}) = 1$ et $d(p_{22}) = 0$ alors : $deg(\{p_{12}, p_{23}\}) = \frac{1+1}{2} = 1$ et $deg(\{p_{11}, p_{22}\}) = \frac{1+0}{2} = 0,5$. L'inconvénient de cette moyenne est que si l'un des degrés d'un couple p_{ik} est très petit, cela n'affectera pas le résultat final. De plus, pour certaines combinaisons clairement différentes, elle ne sera pas discriminante. Par exemple, on obtient le même degré pour les combinaisons $\{p_{12}, p_{23}\}$ et $\{p_{11}, p_{22}\}$ avec $(d(p_{12}) = 0,5$ et $d(p_{23}) = 0,6)$ et $(d(p_{11}) = 1$ et $d(p_{22}) = 0,1)$.

Une moyenne sur l'ensemble des couples compatibles d'une combinaison peut être aussi utilisée : les couples p_{ik} ayant un degré nulle $d(p_{ik}) = 0$, dont les domaines ne sont pas compatibles peuvent être ignorés. Considérons $\mathcal{C}'$ l'ensemble $\mathcal{C}$ sans les p_{ik} dont $d(p_{ik}) = 0$:

$$\forall c \in \mathcal{C}, \forall p_{ik} \in c, d(p_{ik}) \neq 0, p_{ik} \in c', c' \in \mathcal{C}',$$

ainsi, le degré d'une combinaison peut se calculer de la manière suivante :

$$deg(c) = deg(c') = \frac{1}{|c'|} \cdot \sum_{p_{ik} \in c'} d(p_{ik}) \mid c' \in \mathcal{C}'$$

L'avantage de cette moyenne est qu'elle a une approche logique en ne prenant en compte que les couples qui ont un sens dans la combinaison, de ce fait elle favorise les petites combinaisons à forts degrés. Cependant, le fait de ne pas avoir le même nombre de paires dans les différents c' rend difficile la comparaison de leurs degrés. Supposons qu'il y ait une combinaison avec un couple parfait (c'est-à-dire $deg(p_{ik}) = 1$) et une seconde combinaison avec 2 couples parfaits, elles auront le même degré. Les couples de degrés faibles tirent les degrés des combinaisons vers le bas à nombre de couple parfait égal. Par exemple, la combinaisons $c_1 = \{p_{12}, p_{23}\}$ avec $d(p_{12}) = 1$ et $d(p_{23}) = 0)$ a un degré de $deg(c_1) = 1$ nettement avantageux par rapport à $c_2 = \{p_{11}, p_{22}\}$ avec $(d(p_{11}) = 1$ et $d(p_{22}) = 0,1$ qui a un degré $deg(c_2) = 0,55$. Cette moyenne présente un problème majeur de non normalisation.

Il peut également être intéressant de ne considérer que les compatibilités de domaines parfaites :

$$deg(c) = \sum_{\forall p_{ik} \in c} d(p_{ik}) \mid c \in \mathcal{C}, d(p_{ik}) = 1$$

L'avantage de cette moyenne est qu'elle favorise les plus longues combinaisons parfaites. Cependant, elle ne prend pas du tout en compte, les couples aussi proches soient-ils de la perfection.

La moyenne quadratique peut également être utilisée pour calculer les degrés des combinaisons :

$$deg(c) = \sqrt{\frac{1}{r} \cdot \sum_{p_{ik} \in c} d(p_{ik})^2} \mid c \in \mathcal{C}. \tag{5.10}$$

L'avantage de cette moyenne est qu'elle prend le nombre total initial r de couples, ainsi les degrés des combinaisons est naturellement comparables. De plus, elle ne donne jamais le même résultat lorsque la somme des degrés des couples d'une combinaison est égale à la somme des degrés des couples d'une autre combinaison. Donc, elle est discriminante. Supposons que $d(p_{12}) = 0,9, d(p_{23}) = 0,3, d(p_{11}) = 0,8$ et $d(p_{22}) = 0,4$ alors : $deg(\{p_{12}, p_{23}\}) = \sqrt{\frac{0,9^2+0,3^2}{2}} = 0,67$ et $deg(\{p_{11}, p_{22}\}) = \sqrt{\frac{0,8^2+0,4^2}{2}} = 0,63$. L'inconvénient de cette moyenne est que si le degrés d'un couple p_{ik} d'une combinaison est très faible, cela n'affecte pas le résultat final. De plus, elle a un coup de traitement plus élevé que les précédentes.

Enfin, les degrés des combinaisons peuvent être considérés comme une distance entre les domaines des attributs et variables. Différents types de distance (de Manhattan, Euclidienne, Minkowski), qui n'ont pas été étudiés dans le cadre de la thèse, peuvent également être utilisés pour fusionner les distances des couples en une distance de combinaison. Le choix d'une méthode pour fusionner les degrés est indispensable et dépendant du type de combinaison qui sera favorisé. Dans la suite de cette thèse, je ne m'appuierai que sur l'équation 5.9 pour calculer les degrés des combinaisons. Dans le cadre plus général des WO, où ce calcul se fait à l'exécution sans forcément connaître l'ensemble des domaines des attributs qui eux aussi sont découverts petit à petit au cours de l'exécution. L'équation 5.9 ne présente qu'un seul élément variable, les $d(p_{ik})$, puisque r est constant et connu dès le départ. Cependant, cette problématique de découverte en cours d'exécution sort du cadre de cette thèse et ne sera pas approfondie ici.

Minimisation du nombre de combinaisons

La première solution pour minimiser le nombre de combinaisons consiste à placer un seuil δ qui permettra d'éliminer toutes les combinaisons dont le degré est inférieur à ce dernier, et en particulier, toutes les combinaisons dont le degré est égal à zéro. Indépendamment de ce seuil, toutes les combinaisons dont le degré égal à 0 doivent être supprimées, ainsi la valeur par défaut de δ est 0.

$$^{v}\mathcal{C} = \{c \mid c \in \mathcal{C}, deg(c) > \delta\} \tag{5.11}$$

Du fait que certains couples dont le degré est nul sont ignorés, certaines combinaisons peuvent être incluses l'une dans l'autre. Pour cette raison, elles peuvent

être fusionnées afin de **maximiser** les combinaisons, c'est-à-dire avoir les plus grandes combinaisons possibles avec le plus grand degré de compatibilité. Donc, si une combinaison de couples d'attribut-variable $c \in {}^{v}\mathcal{C}$ apparaît dans d'autres combinaisons , celle-ci sera ignorée, si elle a un degré de compatibilité de domaines inférieur à la combinaison à laquelle elle appartient, ce qui donne des combinaisons maximisées. L'ensemble de toutes les combinaisons ${}^{v}\mathcal{C}_{max}$ maximisées est donné par :

$$ {}^{v}\mathcal{C}_{max} = \{c \mid c \in {}^{v}\mathcal{C}, \forall c' \in {}^{v}\mathcal{C}, deg(c) > deg(c') \vee c \not\subset c'\}. \tag{5.12} $$

5.2.5 Illustration : extraction des combinaisons

À partir du STG (figure 4.1) et de l'IOSTS (figure 2.4), on obtient les ensembles A et X, respectivement, les ensembles d'attributs et de variables :

$$ \begin{array}{rcl} A & = & \{Level, Orientation\}, \\ X & = & \{Height, Angle, Step\}. \end{array} $$

Les domaines de chaque élément des ensembles sont définis comme suit :

$$ \begin{array}{rl|rl} dom(Height) & = [0, 100] & dom(Level) & = [0, 100] \\ dom(Angle) & = [0, 90] & dom(Orientation) & = [0, 90] \\ dom(Step) & = [50, 50] & & \end{array} $$

Ainsi, la matrice D sera définie comme suit :

$$ D_{(A,X)} = \begin{array}{r} \\ Level \\ Orientation \end{array} \begin{array}{c} \begin{array}{ccc} Height & Angle & Step \end{array} \\ \begin{bmatrix} 1 & 0 & 0 \\ 0,9 & 1 & 0 \end{bmatrix} \end{array}. \tag{5.13} $$

D'après les équations 5.3 et 5.5, les arrangements $(\mathcal{A}_3^2)$ et leurs images $(\sigma(\mathcal{A}_3^2))$ sont définis comme suit :

$$ \begin{array}{rcl} \mathcal{A}_3^2 & : & \{(1,2),(1,3),(2,3),(3,2),(3,1),(2,1)\}, \\ \sigma(\mathcal{A}_3^2) & : & \{(1,2),(1,2),(1,2),(1,2),(1,2),(1,2)\}. \end{array} \tag{5.14} $$

Ce qui donne d'une manière explicite ce qui suit :

$$ (\mathcal{A}_3^2, \sigma(\mathcal{A}_3^2)) = \left\langle \begin{array}{ll} \{(Height, Level)^1, & (Angle, Orientation)^1\}^{\mathbf{1}}, \\ \{(Height, Level)^1, & (Step, Orientation)^0\}^{\mathbf{0,5}}, \\ \{(Angle, Level)^0, & (Step, Orientation)^0\}^{\mathbf{0}}, \\ \{(Step, Level)^0, & (Angle, Orientation)^1\}^{\mathbf{0,5}}, \\ \{(Step, Level)^0, & (Height, Orientation)^{0,9}\}^{\mathbf{0,45}}, \\ \{(Angle, Level)^0, & (Height, Orientation)^{0,9}\}^{\mathbf{0,45}} \end{array} \right\rangle = \mathcal{C}. \tag{5.15} $$

En appliquant les équations 5.11 et 5.12 au résultat 5.14, les combinaisons maximisées possibles sont définies comme suit :

$$ {}^{v}\mathcal{C}_{max} = \left\langle \begin{array}{c} \{(Height, Level)^{1}, (Angle, Orientation)^{1}\}^{\mathbf{1}}, \\ \{(Height, Orientation)^{0,9}\}^{\mathbf{0,45}} \end{array} \right\rangle . \tag{5.16} $$

Théoriquement, la matrice D présentée dans cette illustration présente toutes les combinaisons en fonction du degré de compatibilité entre les domaines des attributs et des variables, et après quelques opérations, on obtient l'ensemble final de toutes les combinaisons maximisées. Je mentionne également qu'un expert doit donner un seuil de décision (δ) basé sur le contexte de l'application, pour pousser l'algorithme à déterminer les combinaisons logiques pour les humains à partir de toutes les combinaisons ${}^{v}\mathcal{C}_{max}$. Mais dans certains cas, comme dans cette illustration, l'attribut *Level* correspond à la variable *Angle* selon le domaine de valeur, cette correspondance est logique pour la machine en raison de la matrice D (c'est-à-dire, $deg((Angle, Level)) = 0,9 \neq 0$), alors que ce n'est pas le cas pour les humains, puisque *Angle* est sémantiquement différent de *Level*. Par conséquent, le fait de se fier uniquement à la matrice D pour effectuer tous les calculs perturbera l'expert lorsqu'il s'agira de déterminer le seuil δ, puisque le contexte des éléments de la matrice est omis. Le principal défi consiste à présent à calculer une matrice qui présente à la fois une similarité de domaine et une similarité sémantique entre les couples d'attribut/variable, afin de donner à l'algorithme une vision quasi-humaine des couples correctes d'attribut/variable.

5.3 Application de la compatibilité sémantique

Pour mesurer la similarité entre deux concepts (mots, phrases, documents), il existe de nombreuses mesures de similarité, telles que les mesures syntaxiques, celles linguistiques/sémantiques et celles basées sur la taxonomie [ZXY+21]. Dans cette étude, j'utilise la similarité sémantique comme mesure, car elle représente le caractère commun de deux concepts, en s'appuyant sur leurs relations hiérarchiques. Cette mesure utilise le dictionnaire « thésaurus [1] », afin de trouver le degré de similarité en faisant correspondre des mots qui sont conceptuellement liés, mais pas nécessairement syntaxiquement similaires [AAA18]. En outre, dans ce cas, elle contribuera à générer la matrice de similarité domaine-sémantique – domain-semantic similarity matrix (DSS) –, en injectant la mesure de similarité sémantique entre les couples d'attribut/variable dans le processus de calcul de la matrice D 5.2, de manière à pousser la machine vers une capacité de décision quasi-humaine.

1. Thésaurus est un dictionnaire analogique de référence dans lequel les mots sont organisés par champ lexical, où l'on peut trouver des synonymes et antonymes de mots.

5.3.1 Distance sémantique

Les chercheurs ont proposé il y a plusieurs années diverses méthodologies prometteuses (méthodes de comptage des arêtes [RMBB89], méthodes de contenu de l'information [MJB15], méthodes basées sur les caractéristiques [Tve77], méthodes hybrides [RSM94]) pour calculer la distance de correspondance (similarité linguistique/sémantique) entre les mots à l'aide de différentes mesures de similarité sémantique conventionnelles, telles que Path, Wu & Palmer, Li, Resnik, Lin, Jiang & Conrad et WPath [LC98, WP94, Res99]. Ces méthodologies sont basées sur des ontologies générales telles que WordNet [RB90, Mil95b], qui est une base de connaissances sous la forme d'une base de données lexicale contenant la signification des mots et les relations entre eux dans une hiérarchie conceptuellement organisée (voir figure 5.1), ainsi que les relations entre les synsets [2] telles que l'hyperonymie/l'hyponymie.

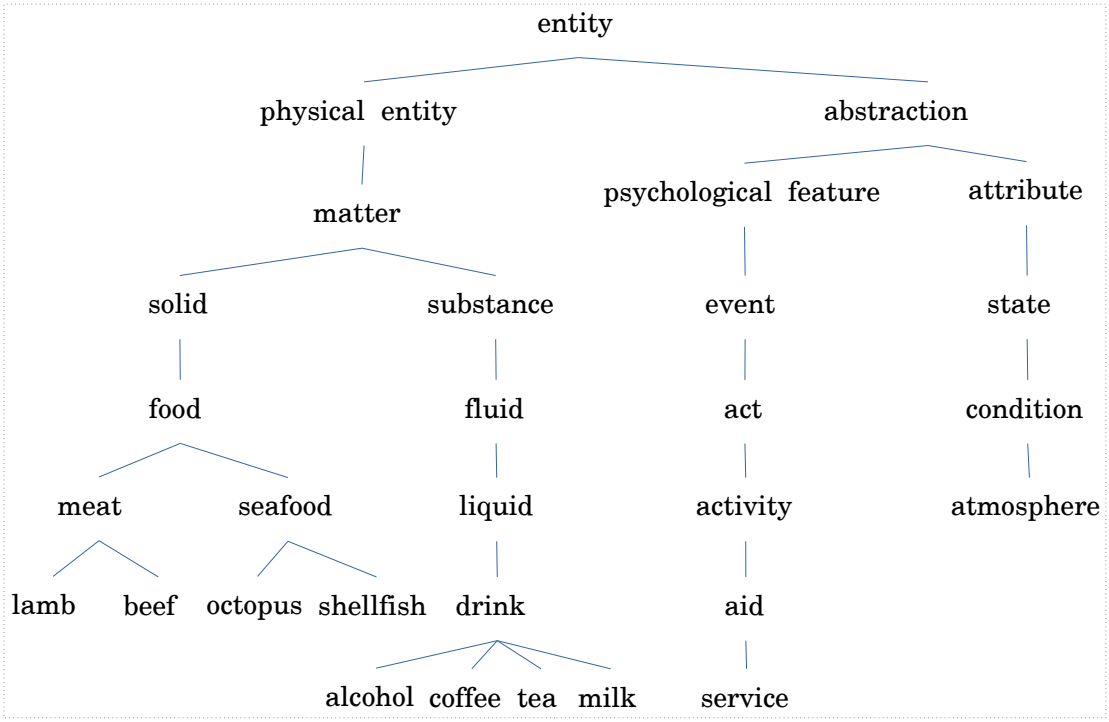


FIGURE 5.1 : Un fragment de la taxonomie des concepts de WordNet [ZI17].

Dans cette thèse, le framework SEMATCH [3] [dSI15] permet de calculer la mesure de similarité sémantique entre les mots afin de l'utiliser dans le processus

2. Les synsets sont un ensemble de mots ayant un sens commun (synonymes).
3. SEMATCH est un framework utilisé pour développer et évaluer les mesures de similarité sémantique pour les concepts, les mots, les entités et leurs applications.

de calcul de la matrice *DSS*. Le choix du framework SEMATCH est fondé sur sa capacité à mesurer la similarité sémantique entre les concepts, représentés par des taxonomies. La similarité sémantique est calculée sur la base de la proximité entre les synsets issus de WordNet, plus précisément, sur la relation hiérarchique (Is-A). En outre, SEMATCH peut calculer la similarité multilingue des mots basée sur WordNet, avec diverses métriques de similarité sémantique [ZI15]. Ce framework présente des résultats prometteurs illustrés dans [ZI17]. L'objectif de mes mesures étant d'être aussi précis que possible, j'ai choisi la métrique « wpath », car elle est une combinaison de plusieurs métriques et a fait état d'une amélioration des performances « Accuracy, Precision, Recall et F-Measure » par rapport à d'autres métriques (tableau 5.1). Sans entrer dans les détails de tous les concepts illustrés dans le tableau, considérons les paires de concepts *(beef, lamb)* et *(beef, octopus)*. Pour calculer la similarité entre ces concepts, les applications exigent que les métriques de similarité donnent une valeur plus élevée à sim(beef, lamb) qu'à sim(beef, octopus), parce que les concepts de *beef* et de *lamb* sont des types de viande tandis que le concept *d'octopus* est un type de fruit de mer. Ainsi, ce tableau montre que la ligne de la paire de concepts *(beef, lamb)* a un score plus élevé que la ligne de concepts *(beef, octopus)* pour toutes les métriques. J'ajoute que je n'ai pas choisi la métrique *path* ou *lch*, parce que les lignes de concepts *(beef, lamb)*, *(meat, seafood)* et *(octopus, shellfish)* ont un score égal, et c'est parce que ces paires de concepts ont la même longueur de chemin le plus court dans l'ontologie utilisée pour le calcul de la similarité. Voir [ZI17] pour plus d'information sur les autres métriques.

Concept Pairs	path	lch	wup	li	res	lin	jcn	wpath
beef-octopus	0.200	2.028	0.714	0.442	6.109	0.484	0.071	0.494
beef-lamb	0.333	2.539	0.857	0.667	6.725	0.591	0.097	0.692
meat-seafood	0.333	2.539	0.833	0.659	6.109	0.760	0.205	0.662
octopus-shellfish	0.333	2.539	0.857	0.667	9.360	0.729	0.125	0.801
beef-service	0.071	0.999	0.133	0.000	0.000	0.000	0.050	0.071
beef-atmosphere	0.083	1.153	0.154	0.000	0.000	0.000	0.052	0.083
beef-coffee	0.111	1.440	0.429	0.168	3.337	0.319	0.066	0.208
food-coffee	0.143	1.692	0.500	0.251	3.337	0.411	0.095	0.260

TABLE 5.1 : Illustration des méthodes de similarité sémantique sur quelques exemples de couples de concepts [ZI17].

5.3.2 Matrice de compatibilité et distance sémantique

Pour calculer la matrice *DSS* intégrant à la fois distance de domaine et distance sémantique, il faut d'abord construire la matrice sémantique ($SM_{|A|\times|X|}$). Comme

illustré dans le listing 5.1, la fonction « WordNetSimilarity() » retourne le degré de similarité sémantique en comparant les mots deux à deux.

```
# 1. Install the scientific computing libraries numpy and scipy.
# 2. Install Sematch framework: pip install sematch

from sematch.semantic.similarity import WordNetSimilarity

def similarity_distance(word_1, word_2, metric="wpath") -> float:
    wns = WordNetSimilarity()
    return wns.word_similarity(word_1, word_2, metric)
```

Listing 5.1 : Interaction avec le framework SEMATCH en utilisant Python.

Ainsi la matrice SM se définit comme suit :

$$SM_{(A,X)} = \begin{bmatrix} sm(att_1, x_1) & sm(att_1, x_2) & \dots & sm(att_1, x_{|X|}) \\ sm(att_2, x_1) & sm(att_2, x_2) & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ sm(att_{|A|}, x_1) & sm(att_{|A|}, x_2) & \dots & sm(att_{|A|}, x_{|X|}) \end{bmatrix}, \tag{5.17}$$

où chaque $sm(att_i, x_j)$ est donnée par la fonction « WordNetSimilarity() ».

La détermination de la matrice DSS nécessite la fusion des deux matrices, D et SM. Il existe un certain nombre de manières pour fusionner les deux matrices, D et SM. Une moyenne simple peut être considérée :

$$DSS_{ij} = \frac{D_{ij} + SM_{ij}}{2} \tag{5.18}$$

L'avantage de cette moyenne est qu'elle résume l'information d'une manière simple dans un point d'équilibre. En général, l'inconvénient de cette moyenne est qu'elle tire le résultat vers le haut en cas de valeurs élevées. Cet inconvénient n'est pas considéré comme négatif dans le contexte de cette thèse, puisque je veux tirer la moyenne vers le haut, c'est-à-dire que si la valeur de SM_{ij} est plus importante, cela aura un impact sur le résultat à choisir.

La fusion peut également être effectuée en utilisant la moyenne quadratique, également appelée, moyenne de Hölder :

$$DSS_{ij} = \sqrt{\frac{D_{ij}^2 + SM_{ij}^2}{2}} \tag{5.19}$$

Cette distance est avantageuse dans le cas de domaines peu compatibles, ou d'attributs et de variables présentant une faible compatibilité sémantique. La distance quadratique peut être utilisée pour favoriser les fortes compatibilités.

La fusion peut également être effectuée à l'aide de la fonction MAX, mais dans ce cas, l'un des degrés est complètement ignoré, ce qui n'est généralement pas souhaitable.

Dans le contexte de cette thèse, le type de fusion utilisé est celui décrit dans l'équation 5.18. Elle présente l'avantage de gérer naturellement le comportement des WO, c'est à dire la découverte des domaines des attributs à l'exécution, puisqu'elle se calcule facilement en flux de données. Cette problématique de découverte en cours d'exécution associée aux méthodes de fusion des degrés sera approfondie dans des travaux futurs. Avec cette approche, l'expert peut déterminer un seuil de décision δ à partir de la vision humaine, qui couvre à la fois le degré de compatibilité entre les domaines et la similarité sémantique, pour pousser l'algorithme vers une décision quasi-humaine. Il convient de noter que le nouveau degré d'un couple $\underbrace{(a_i, x_k)}_{p_{ik}}$ sera $d(p_{ik}) = DSS_{ik}$. Ainsi, l'équation 5.12 devient plus restrictive en appliquant une décision humain représentée par la fonction « hum-dec() » :

$$\text{hum-dec}({}^{v}\mathcal{C}_{max}) = \{c_i \mid \forall c_i \in {}^{v}\mathcal{C}_{max}, deg(c_i) \geq \delta\}. \tag{5.20}$$

5.3.3 Illustration : une décision quasi-humaine

L'illustration 5.2.5 nous a révélé la matrice D et les possibles combinaisons ci-dessous :

$$D_{(A,X)} = \begin{array}{r} \\ Level \\ Orientation \end{array} \begin{array}{c} \begin{array}{ccc} Height & Angle & Step \end{array} \\ \begin{bmatrix} 1 & 0 & 0 \\ 0,9 & 1 & 0 \end{bmatrix} \end{array}. \tag{5.21}$$

$${}^{v}\mathcal{C}_{max} = \left\langle \begin{array}{c} \{(Height, Level)^1, (Angle, Orientation)^1\}^{\mathbf{1}}, \\ \{(Height, Orientation)^{0,9}\}^{\mathbf{0,45}} \end{array} \right\rangle. \tag{5.22}$$

En s'appuyant sur le framework SEMATCH, on construit la matrice SM présentée ci-dessous :

$$SM_{(A,X)} = \begin{array}{r} \\ Level \\ Orientation \end{array} \begin{array}{c} \begin{array}{ccc} Height & Angle & Step \end{array} \\ \begin{bmatrix} 0,87 & 0,23 & 0,66 \\ 0,32 & 0,49 & 0,18 \end{bmatrix} \end{array} \tag{5.23}$$

En appliquant l'équation 5.18, la matrice DSS sera la moyenne des deux matrices, D et SM, comme indiqué ci-dessous :

$$DSS_{(A,X)} = \begin{array}{r} \\ Level \\ Orientation \end{array} \begin{array}{c} \begin{array}{ccc} Height & Angle & Step \end{array} \\ \begin{bmatrix} 0,94 & 0,55 & 0,33 \\ 0,16 & 0,75 & 0,09 \end{bmatrix} \end{array} \tag{5.24}$$

Comme l'ensemble ${}^{v}\mathcal{C}_{max}$ (voir 5.22) contient toutes les combinaisons maximisées reposant sur la matrice 5.13, la matrice DSS est appliquée directement sur celle-ci,

car je ne souhaite pas inclure les combinaisons qui ont des couples avec un degré de compatibilité de domaine égal à 0. Ainsi, le résultat présente les deux concepts, compatibilité selon les domaines et compatibilité selon la mesure de similarité sémantique.

$$ {}^{v}\mathcal{C}_{max} = \left\langle \begin{array}{c} \{(Height, Level)^{0,94}, (Angle, Orientation)^{0,75}\}^{\mathbf{0,85}}, \\ \{(Height, Orientation)^{0,16}\}^{\mathbf{0,08}} \end{array} \right\rangle \tag{5.25} $$

Il est important de noter que la discrimination est maintenant plus facile, puisque la prise en compte de la sémantique a réduit (avant : 0,45) le taux de compatibilité de la deuxième combinaison qui était la moins probable, et augmenté celui de la première, qui est plus logique d'un point de vue humain. Il est donc plus facile de choisir un seuil qui tient compte à la fois de la compatibilité du domaine et de la compatibilité sémantique. La détermination de ce seuil reste une problématique d'expert. Cependant, dans l'exemple sur lequel s'appuie ce travail, et que l'on prenne une valeur moyenne ($\delta = 0,5$), la moyenne de degrés ou un médian, une seule solution sortira.

$$ \text{hum-dec}({}^{v}\mathcal{C}_{max}) = \left\langle \begin{array}{c} \{(Height, Level)^{0,94}, (Angle, Orientation)^{0,75}\}^{0,85 > \delta}, \\ \cancel{\{(Height, Orientation)^{0,16}\}^{0,08 < \delta}} \end{array} \right\rangle $$

Les résultats précédents fournissent l'ensemble des combinaisons intégrant à la fois un aspect numérique et sémantique pour faire correspondre les deux représentations des connaissances, à savoir un STG et un IOSTS. Comme mentionné dans le chapitre précédent, l'algorithme d'appariement étant basé sur le morphisme de graphes, il doit respecter la structure des graphes appariés, ce qui n'est pas vérifié lors de la construction des combinaisons de couple d'attribut/variable. La construction d'appariement état-localité et la vérification de l'aspect structurel font l'objet de la section suivante.

5.4 Appariement de graphes utilisant des données sémantiques

L'appariement entre un STG et un IOSTS se fait entre chaque état du STG et son unique localité correspondante dans l'IOSTS. L'algorithme d'appariement étant basé sur le morphisme de graphes, il doit respecter la structure des graphes appariés [GC97] pour toute combinaison $c \in \text{hum-dec}({}^{v}\mathcal{C}_{max})$ 5.20. La dernière section de ce chapitre, est consacrée à la construction de la matrice d'appariement des deux représentations selon les combinaisons restantes dans hum-dec($ {}^{v}\mathcal{C}_{max}$).

5.4.1 Épimorphisme

Comme mentionné précédemment au chapitre 4, les images des états du STG dans l'IOSTS doivent respecter les *relations d'adjacence* présentes dans le STG (c'est-à-dire les transitions). Ainsi, $\mathcal{S}$ est l'application surjective du STG dans l'IOSTS, respectivement, entre les états V et les localités Q. Pour toute transition $(v, v') \in E$ du STG, $(\mathcal{S}(v), \mathcal{S}(v')) \in T$ est une transition de l'IOSTS. L'appariement STG $\rightarrow$ IOSTS est un homomorphisme surjectif appelé épimorphisme [GC97], formellement décrit dans ce qui suit :

$$\begin{array}{rl} \mathcal{S}\colon STG & \rightarrow IOSTS, \\ V & \twoheadrightarrow Q = \mathcal{S}(V), \end{array} \tag{5.26}$$

Ainsi :

$$\begin{array}{c} \mathcal{S}_E\colon E \rightarrow T, \\ \mathcal{S}_E((v, v')) = (\mathcal{S}(v), \mathcal{S}(v')), \\ (\mathcal{S}(v), \mathcal{S}(v')) \subset T. \end{array} \tag{5.27}$$

Sachant que $\mathcal{S}(v) = \mathcal{S}(v')$ (transition réflexive) dans le cas ou les deux états, v et v', sont liés à la même localité. En d'autres termes, tout état du STG est lié à une localité de l'IOSTS et chaque localité de l'IOSTS est liée à au moins un état du STG. S'il existe une transition entre deux états, il existe une transition entre les deux localités auxquels ils sont liés, sachant que ces deux localités peuvent être identiques.

5.4.2 Matrice d'appariement de graphes

Afin d'obtenir la correspondance entre les états et les localités, le même raisonnement logique est utilisé (c'est-à-dire utilisation des matrices), la matrice binaire $G2M_{n\times m}$ – Graph Matching Matrix – est appliquée sur la base de l'équation suivante 5.28. n et m sont les normes des ensembles d'états V et de localités Q, respectivement. Rappelons qu'un état $v_i \in V$, correspond à une localité $q_j \in Q$ ($v_i \implies q_j$) si, et seulement si, le domaine de chaque attribut dans l'état v_i, appartient au domaine de sa variable correspondante dans la localité q_j :

$$\begin{array}{c} \forall c_k \in \text{hum-dec}({}^{v}\mathcal{C}_{max}), \forall (att, x) \in c_k, \exists! v_i \in V, \exists! q_j \in Q \\ \mid dom(v_i, att) \subseteq dom(q_j, x) \\ \Longleftrightarrow \\ v_i \implies q_j \end{array} \tag{5.28}$$

En conséquence, la valeur de la case $g2m(v_i, q_j)^{c_k}$ dans la matrice $G2M_{n\times m}^{c_k}$ sera déterminée comme suit :

$$g2m(v_i, q_j)^{c_k} = \begin{cases} 1, & \text{iff} \quad v_i \implies q_j. \\ 0, & \text{otherwise.} \end{cases} \tag{5.29}$$

En fonction de la compatibilité des couples attribut/variable dans chaque combinaison $c_k \in \text{hum-dec}({}^{v}\mathcal{C}_{max})$ donnée par l'équation 5.28, l'appariement entre le STG et l'IOSTS est défini par la matrice $G2M^{c_k}_{n\times m}$ résultant de l'équation 5.29 :

$$g2m(v_i, q_j)^{c_k} = y^{c_k}_{ij} \mid c_k \in \text{hum-dec}({}^{v}\mathcal{C}_{max}). \tag{5.30}$$

Par ailleurs, la matrice $G2M$ présente certaines caractéristiques :

- Si la matrice $G2M$ est conforme à l'équation 5.31, à savoir que la somme des valeurs de chaque ligne est égale à 1, alors, l'appariement est correct. On peut dire aussi que l'appariement est une *bijection* dans le cas ou le nombre de lignes n est égal au nombre de colonnes m. Autrement dit, chaque localité de l'IOSTS est associée à un seul état dans le STG, ce qui est très rare dans un cas réel vu que l'IOSTS encapsule plusieurs états dans une seule localité. La matrice $G2M$ prend la forme d'une matrice de permutation.

$$\forall i \in \{1, \ldots, n\}, \sum_{j=1}^{m} y_{ij} = 1. \tag{5.31}$$

- Si la matrice $G2M$ est conforme à l'équation 5.32, à savoir si la somme des valeurs de chaque colonne est supérieure ou égale à 2, cela signifie que l'appariement est correct, ce qui implique également une *surjection*. Autrement dit, chaque localité dans l'IOSTS est appariée à un ou plusieurs états dans le STG.

$$\forall j \in \{1, \ldots, m\}, \sum_{i=1}^{n} y_{ij} \geq 2 \tag{5.32}$$

- Si la matrice $G2M$ est conforme à l'équation 5.33, cela signifie qu'il existe un état dans le STG sans image correspondante dans l'IOSTS, ou qu'un état du STG a deux images dans l'IOSTS. Il y a dans ce cas une incohérence entre les deux formalismes.

$$\forall i \in \{1, \ldots, n\}, \exists i' \mid \textstyle\sum_{j=1}^{m} y_{i'j} = 0 \vee \sum_{j=1}^{m} y_{i'j} \geq 2. \tag{5.33}$$

- Si la matrice $G2M$ est conforme à l'équation 5.34, cela signifie qu'il existe une localité orpheline dans l'IOSTS sans état qui lui corresponde dans le STG. Il y a dans ce cas une incohérence entre les deux formalismes.

$$\forall j \in \{1, \ldots, m\}, \exists j' \mid \textstyle\sum_{i=1}^{n} y_{ij'} = 0. \tag{5.34}$$

5.4.3 Algorithme d'appariement par étapes

L'algorithme d'appariement reçoit deux types de représentation de connaissances (STG et IOSTS), et se déroule automatiquement en trois étapes résumées dans la figure 5.2. La première détermine toutes les combinaisons possibles qui se résument dans l'équation 5.12. On dit des combinaisons possibles parce qu'elles peuvent générer un appariement numériquement logique entre les deux représentations de connaissances. Le résultat est utilisé dans la deuxième étape pour y appliquer la mesure sémantique, afin de ne laisser que les combinaisons qui satisfont une compatibilité de domaine et une sémantique humaine correcte, qui se résume dans l'équation 5.20. Généralement, il n'y a qu'une seule combinaison. Enfin, l'algorithme utilise cette combinaison pour calculer la matrice d'appariement des deux représentations de connaissances (équations 5.28 et 5.29).

5.4.4 Illustration : appariement de graphes

À partir des illustrations 5.2.5 et 5.3.3, nous avons trouvé toutes les combinaisons maximisées possibles qui sont à la fois logiques pour l'humain et la machine, elles sont définies comme suit :

$$\text{hum-dec}({}^{v}\mathcal{C}_{max}) = \Big\langle \ \{(Height, Level)^{0,94}, (Angle, Orientation)^{0,75}\}^{0,85 > \delta} \ \Big\rangle .$$

Comme chaque illustration est le complément de celle qui la précède, j'ai pris les mêmes STG et IOSTS que dans l'illustration 5.2.5 de ce chapitre. Pour rappel, le STG possède 4 états V (v_0, v_1, v_2, v_3) et l'IOSTS possède 2 localités Q $(Closed, Open)$. Ainsi, selon les deux équations 5.28 et 5.29, la matrice $G2M^{c_1}_{n\times m}$ est calculée comme suit :

$$G2M^{c_1}_{n\times m} = \begin{array}{c} \\ v_0 \\ v_1 \\ v_2 \\ v_3 \end{array} \begin{array}{c} \begin{array}{cc} \textit{Open} & \textit{Closed} \end{array} \\ \left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{array}\right] \end{array} \left\| \begin{array}{c} \text{Verification} \\ \sum_j^{|Q|} y_{0j} = 1 \\ \sum_j^{|Q|} y_{1j} = 1 \\ \sum_j^{|Q|} y_{2j} = 1 \\ \sum_j^{|Q|} y_{3j} = 1 \end{array} \right. \tag{5.35}$$

La matrice $G2M^{c^1}_{n\times m}$ respecte les équations 5.31 et 5.32, ce qui signifie que l'appariement obtenu est correct et surjectif.

5.5 Conclusion

Dans ce chapitre, j'ai donné une introduction générale à l'utilisation des ontologies pour fournir aux systèmes un contexte, une sensibilité sémantique ou les

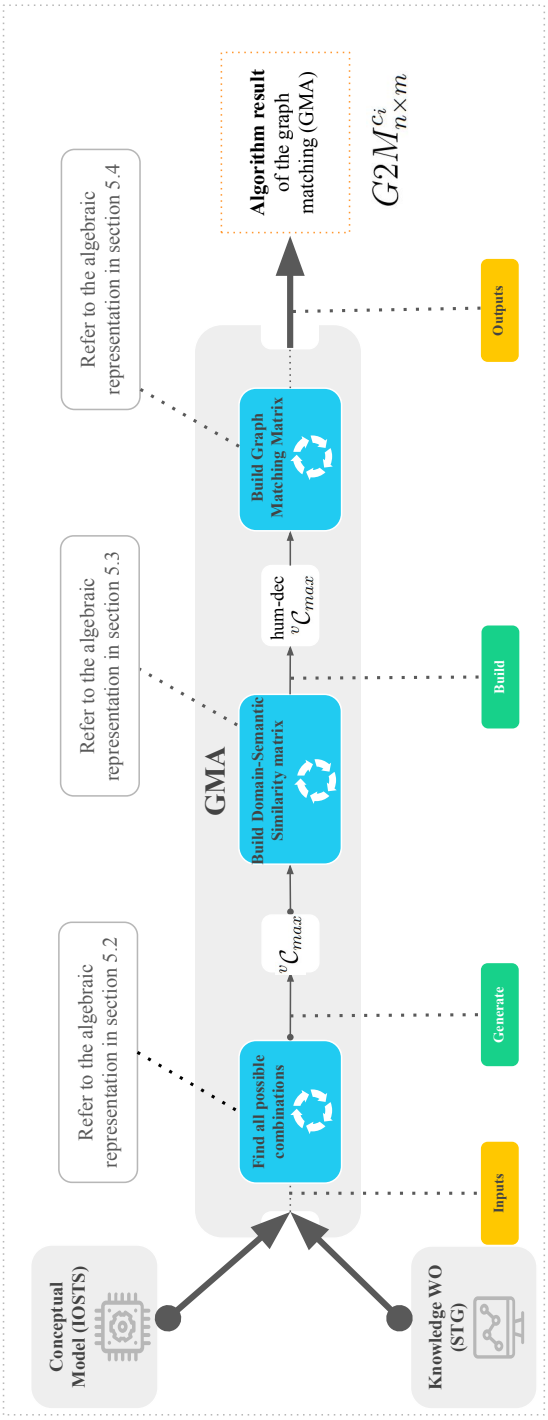


FIGURE 5.2 : Illustration de l'algorithme d'appariement par étapes (le résultat est dans la figure 4.9a).

deux à la fois, avant de passer au sujet principal, à savoir l'appariement de deux graphes de nature différente sur la base de leurs attributs, de leurs variables et de leur sémantique. Ce chapitre divise la problématique en trois parties principales, à savoir le calcul de la matrice de distance numérique entre les attributs et les variables, la fusion de la distance sémantique et de la distance numérique en une seule mesure, et enfin le calcul de la matrice d'appariement entre les deux graphes. Cette dernière matrice respecte la distance numérique, le sens sémantique et les contraintes structurelles des deux graphes, puisque l'appariement des graphes est basé sur l'épimorphisme. Cela permet d'obtenir toujours un appariement généralement unique dans le cas où les deux formalismes sont cohérents, aussi bien du point de vue de la machine que de celui de l'homme.

Les travaux principaux de cette thèse partent de la connaissance d'origine de la machine, des WO. Cette connaissance est purement numérique et nous avons vu comment lui apporter de la sémantique pour une inter-compréhension entre la machine et l'humain. Le problème peut être abordé dans l'autre sens, c'est-à-dire en partant de la connaissance humaine pour communiquer avec le système. Le chapitre suivant s'intéresse à cette approche. L'appariement d'un graphe de connaissance extrait d'une phrase, ou une commande vocale, avec une ontologie dans le contexte des maisons intelligentes.

Sommaire

Chapitre

6

Alignement des ontologies : Première étape vers la commande vocale pour les scénarios d'assistance

Après l'ère de l'ordinateur personnel vient celle de l'informatique omniprésente [Thé13], qui fait désormais partie intégrante de la vie des utilisateurs, grâce à des appareils connectés. Ces appareils communiquent entre eux et avec leurs utilisateurs de manière discrète, ne nécessitant parfois qu'un minimum d'attention de la part de ces derniers. C'est le cas des systèmes d'assistance personnelle intelligents – Smart Personal Assistance (SPA) – tels qu'Amazon Alexa, Siri et Google Home. IBM Shoebox, a été le premier assistant à commande vocale présenté en 1961 par William Dersch, un ingénieur d'IBM [Der]. Ces systèmes peuvent conduire à des difficultés majeures s'ils sont combinés avec la notion de « Do It Yourself (DIY) ». En effet, les utilisateurs doivent être pro-actifs en définissant des scénarios de fonctionnement pour ces systèmes, ce qui nécessite à la fois une connaissance du fonctionnement des SPA et la capacité de combiner des composants de DIY pour construire un système personnalisé.

Ce chapitre détaille ma collaboration avec le laboratoire DOMUS de l'université de Sherbrooke au Canada, et présente une nouvelle approche de la création de scénarios d'assistance pour favoriser le vieillissement à domicile. Cette approche combine les notions de DIY et de SPA, et traitera la communication **de l'humain vers la machine**, soit le chemin inverse de ce que j'ai présenté dans les chapitres 4 et 5.

6.1 Contexte

Le concept de « DIY », catégorisé sous le terme « d'activités de consommation », est fortement adopté par les consommateurs/utilisateurs pour co-créer de la valeur. Comme illustré dans [XBT08], les utilisateurs ont commencé à fabriquer des produits pour leur propre consommation en utilisant leurs propres paradigmes. Ils ont également commencé à combiner la notion de DIY avec les systèmes SPA. En général, et à titre d'exemple simple, les utilisateurs peuvent combiner le concept DIY avec les systèmes SPA pour créer un système de notification qui les alerte par téléphone lorsqu'il n'y a plus de nourriture pour le chat dans sa gamelle. Les composants DIY présentent ces avantages suivants :

- ils sont abordables pour tout le monde, car ils sont moins chers qu'un système domotique traditionnel (bon marché),
- ils ne nécessitent pas l'aide d'un expert pour l'installation et la configuration, une brève documentation est suffisante (facilité),
- leurs utilisateurs peuvent choisir parmi une large gamme de produits existants sur le marché (flexibilité).

Bien qu'il y ait de nombreux avantages, la notion de DIY présente également des inconvénients [WL15] :

- les utilisateurs doivent être pro-actifs, par exemple, ils doivent chercher les composants et les mettre en place selon un scénarios qu'ils doivent définir a priori,
- les utilisateurs doivent décider des caractéristiques des dispositifs, par exemple, capteurs, ampoules, actionneurs, dont ils ont besoin,
- les utilisateurs doivent s'autoformer à travers la documentation pour comprendre comment et où installer ces dispositifs, car il existe une grande variété de dispositifs pouvant être impliqués dans un système [ZBC+14].

Ces inconvénients sont surmontés grâce à un ensemble de ressources disponibles en ligne.

Avec le vieillissement de la population mondiale, le DIY et le SPA sont combinés pour aider ces personnes dans leur vie quotidienne, notamment en favorisant leur autonomie et en promouvant le vieillissement à domicile, même lorsque ces personnes ont des handicaps cognitifs ou des déficiences perceptives [RM13] tels que la maladie « d'Alzheimer ». Différents capteurs, des glucomètres, des tensiomètres, des gyroscopes et d'autres appareils connectés sont installés dans les pièces ou portés par la personne âgée pour détecter leurs activités. En fonction de certaines politiques (données par les soignants ou les proches), la maison fournit des indices visuels ou verbaux pour assister la personne âgée dans ses activités, pour avertir la personne elle-même ou pour envoyer un message à ses proches ou à ses soignants. De la technologie d'assistance à la vie quotidienne découle les systèmes ambiants d'assistance à la vie – Ambiant Assistance Living (AAL) – pour les personnes âgées [RNP+16] développés au laboratoire DOMUS.

Les systèmes AAL peuvent améliorer la vie des personnes âgées et de leurs proches, en agissant dans la vie quotidienne de ces personnes comme un rappel de médicaments, en améliorant leur sommeil grâce à des indices oraux/visuels, en permettant à leurs proches de définir des scénarios d'assistance basés sur plusieurs règles pour les aider pendant les épisodes d'errance nocturne [RNG+17]. Mentionnons qu'un scénario d'assistance aux personnes âgées décrit les comportements du système, de l'utilisateur, de l'environnement et les séquences de tâches exécutées par l'utilisateur et le système pour atteindre un objectif [KNPF+17]. Du côté des proches, les systèmes AAL peuvent les aider à avoir une certaine tranquillité d'esprit concernant les personnes âgées dont ils s'occupent, en leur envoyant des informations récentes sur leur état de santé, en leur indiquant si elles ont passé la nuit dans leur chambre, si elles s'occupent correctement de leur hygiène, et ainsi de suite. En général, l'efficacité des systèmes AAL a été prouvée par plusieurs études de recherche basées sur des expériences en situation réelle, comme indiqué dans [NPL+20, RNP+16, RNG+17].

6.2 L'existant

Les systèmes AAL sont très intéressants, en particulier lorsqu'ils favorisent le vieillissement sur place [NPL+20, RNP+16, RNG+17]. Ces systèmes sont désormais conscients du contexte, c'est-à-dire capable de comprendre le contexte des données collectées et de fournir des services personnalisés adaptés aux besoins de l'environnement et de l'utilisateur [FKT14, SM14, BMB+16]. D'une part, ils peuvent réduire la charge des soignants/proches en leur donnant beaucoup plus d'informations sur l'état de santé et la situation des personnes âgées à domicile à un moment précis. D'autre part, les systèmes AAL permettent à toute personne (par exemple, les professionnels de la santé, les proches, etc.) de pratiquer le concept

de DIY avancé pour co-concevoir l'assistance par le biais de la réalité augmentée – Augmented Reality (AR) – et les ontologies [HNGP20].

Pour personnaliser les systèmes AAL pour les personnes âgées souffrant de démence d'Alzheimer et d'errance nocturne, les soignants/proches doivent créer une assistance à la main grâce à un outil appelé conseiller virtuel – Virtual Adviser (VA) –. Au sein du laboratoire DOMUS, ils utilisent le casque de réalité mixte Microsoft Hololens pour déployer cet outil (VA), et selon [HPG20] : « les soignants/proches ont généralement perçu l'expérience comme très stimulante et avec un fort sentiment de connectivité et de jeu, améliorant ainsi l'engagement ». Dans [Hay18] et [HBHN21], l'AR est utilisée pour aider les personnes atteintes de démence en leur offrant des expériences apaisantes, des aides à la mémoire et une stimulation cognitive, l'inconvénient majeur est que les outils d'AR sont portés par la personne, ce qui les rend invasifs et non souhaitables compte tenu de mon objectif de ne pas surcharger l'utilisateur. Par ailleurs, la création d'une assistance manuelle étape par étape place les proches et les soignants au centre du processus et les submerge d'informations et de travail. Par conséquent, dans ma collaboration avec DOMUS, je m'intéresse côté soignants/proches à alléger leur travail en remplaçant la création de l'assistance du VA par des commandes vocales, grâce à l'alignement d'ontologies.

En ce qui concerne l'alignement des ontologies, les chercheurs ont proposé, il y a quelques années, diverses méthodologies prometteuses pour calculer l'alignement entre deux ontologies, à savoir : Machine Learning [AJ18, AMF17], techniques de calcul parallèle [APdN16], techniques du Cloud Computing [AKH+16], arbres de décision [Gaj11] et ainsi de suite. L'alignement des ontologies occupe une place cruciale dans certaines opérations [SE13] telles que : l'évolution des ontologies [NCLM06], l'intégration des ontologies [IWZ+09], l'intégration des données [TIP10], les entrepôts de données [DHW+08]. Désormais, il apparaît également dans le contexte de la maison intelligente, le cas présent, en particulier pour les demandes vocales visant à créer des scénarios d'assistance pour les personnes âgées atteintes de la démence d'Alzheimer. Étant donné que cette thématique de recherche a été lancée récemment, lors de mon stage de recherche au laboratoire DOMUS de Université de Sherbrooke au Canada, j'utilise une technique simple de similarité linguistique pour aligner les ontologies en vue de la création des scénarios d'assistance.

6.3 Définitions préliminaires des concepts clés

Il est nécessaire, avant d'entrer dans le vif du sujet, de clarifier certaines définitions et concepts afin d'éviter toute ambiguïté. Comme il existe plusieurs définitions et concepts, j'ai opté pour celles et ceux qui correspondent le mieux à mes travaux

de thèse.

6.3.1 Tâche - Action - Scénario

Definition 6.3.1 (Tâche) *:* Une tâche est un flux de travail organisé pour la réalisation d'une activité. Elle peut être décomposée en sous-tâches, également appelées comportements. Par exemple, prendre un verre pour boire de l'eau, prendre une douche, etc. [KNPF+17].

Definition 6.3.2 (Action) *:* Une action est une tâche atomique, elle conduit à un état qui peut être compris par les machines et détecté par les capteurs. Par exemple, prendre un verre, ouvrir le robinet, se déplacer dans la cuisine, etc. [KNPF+17].

Definition 6.3.3 (Scénario) *:* Un scénario d'assistance décrit la manière dont les activités sont organisées pour atteindre un objectif particulier « un état de satisfaction à atteindre » et les interactions entre toutes les entités (capteurs, personnes, meubles, etc.) impliquées dans ces activités. Pour lever toute ambiguïté, avant de créer un scénario, plusieurs questions doivent être posées par les soignants/-proches : où cela se passera-t-il, comment cela se passera-t-il et quelles sont les entités impliquées. Par exemple, Pauline se lève le matin et se dirige vers la cuisine pour préparer le petit-déjeuner. [KNPF+17].

6.3.2 Concepts d'ontologie et d'alignement d'ontologies

Definition 6.3.4 (Concept d'ontologie) *:* Une ontologie est l'énoncé d'une théorie logique [Gru95] définie de manière formelle, comme donné par la définition 3.1.1.

Definition 6.3.5 (Concept d'alignement d'ontologies) *:* Un alignement d'ontologies $\mathcal{M}$ entre deux ontologies O et O' est un ensemble d'éléments de mise en correspondance entre les entités de ces ontologies. Un élément de mise en correspondance est un tuple de cinq éléments $\mathcal{M}(id, e, e', n, R)$ [ES+07, BL11], où :

- id est un identifiant unique de l'alignement ;
- e et e' sont des entités telles que : $e \in C(O)$ et $e' \in C(O')$, avec $C(O)$ la fonction qui retourne les classes de l'ontologie O ;
- n est une mesure de confiance, $n \in [0, 1]$, qui se base sur des mesures de similarité normées entre les entités e et e' [AAA18] telles que : les mesures syntaxiques [XW16, Hee04], les mesures linguistiques [XY18] et les mesures basées sur la taxonomie [ZXY+21] ;
- R est une relation, par exemple : la relation de l'équivalence ($e = e'$), celle la plus générale ($e \sqsupseteq e'$) où e est plus général que e' et la relation disjointe ($e \perp e'$).

Definition 6.3.6 (Processus d'alignement d'ontologies) *:* Le processus d'alignement des ontologies ϕ peut être considéré comme une fonction $\mathcal{M}' = \phi(O, O', \gamma, r, \Omega)$, où :

- O and O' sont les ontologies à aligner.
- γ est un ensemble de paramètres, par exemple un seuil.
- r est un ensemble de ressources qui seront utilisées au cours du processus d'alignement.

  Par exemple, des préfixes (fred, dul, etc.). Notons que r, l'ensemble des préfixes, ne peut pas être vide $r \neq \emptyset$ et ne contient pas le mot vide $\epsilon \notin r$. Le préfixe « $fred$ » signifie que la classe se trouve dans l'espace de noms local par défaut du lecteur automatique pour le web sémantique « FRED » [AGP+17], le préfixe « *dul* » est utilisé pour spécifier qu'une classe est de type événement.
- Ω est un ensemble de conditions présentées sous la forme de prédicats permettant d'extraire (et d'aligner) uniquement les classes/individus de O' qui satisfont ces prédicats, par exemple, « subClassOf », « type », etc.

Notez que dans cette thèse, l'alignement est considéré comme un axiome d'équivalence pour deux entités alignées (correspondantes), et est basé sur le concept de similarité des entités. Dans un premier temps, seules les mesures linguistiques sont prises en compte pour trouver l'alignement entre les entités et les instances. Ces mesures sont obtenues grâce au framework SEMATCH dont le fonctionnement est détaillé dans la section 5.3.1 du chapitre 5.

6.3.3 Extraction de connaissances à l'aide de FRED

Une action d'un scénario est convertie en données liées à l'aide du framework FRED [DGPN13, AGP+17, PDG12]. FRED est un lecteur automatique pour le web sémantique créé par le CNR (Italie), il est mis à la disposition du public sous la forme d'une API REST [1]. FRED est capable d'analyser des textes en langage naturel dans 48 langues différentes et de les transformer en données liées – Linked data –. Dans cette thèse, j'ai choisi un graphe de connaissances formelles structurées en RDF [2] – Resource Description Framework – comme données liées, car il offre deux avantages principaux :

- Il permet de capturer des schémas de métadonnées qui sont à la fois lisibles par un humain et traitables par une machine.

1. API du framework FRED : http ://wit.istc.cnr.it/stlab-tools/fred/
2. Resource Description Framework est un modèle de graphe destiné à décrire formellement les ressources Web et leurs métadonnées, ce qui permet de les traiter d'une manière automatique.

- Il est conçu pour permettre la réutilisation et l'extension des schémas existants pour un ensemble de métadonnées sémantiques en constante évolution.

La figure 6.1 illustre le graphe de connaissances formelles structurées de l'action « Pauline se lève de son lit », généré par le framework FRED. Les classes sont représentées par un cercle jaune et leurs individus/instances par un losange violet. L'exemple met l'accent sur le style de représentation n-aire, axé sur les événements, qui utilise les rôles sémantiques et les relations entre les événements pour maintenir la connexité des entités extraites. Le listing 6.1 est un exemple d'accès à l'API REST de FRED.

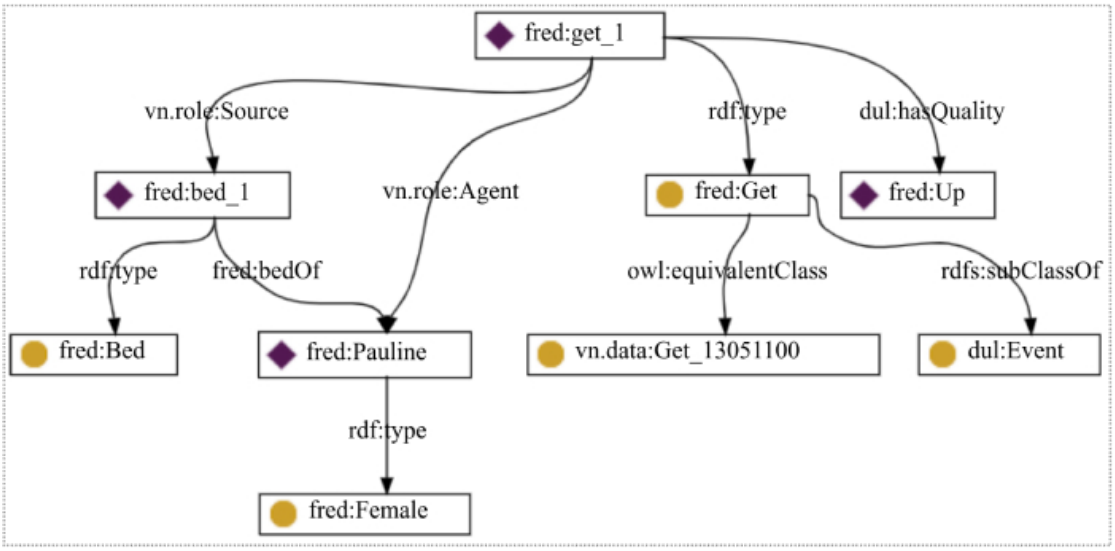


FIGURE 6.1 : Action : « Pauline gets up from her bed ».

```
# ----- FIRST METHOD ----
# When executing the curl request bellow, an RDF triples will be send as response (You can visualize it by using an RDF/OWL visualizer)

curl -G -H "Accept: application/rdf+xml" -H "Authorization: Bearer <request the CNRS to give tou a TOKEN from the form bellow" --data-urlencode text="Pauline gets up from her bed" http://wit.istc.cnr.it/stlab-tools/fred

# The form: https://docs.google.com/forms/d/e/1FAIpQLScPO_xL_F6yw9Cf9p5rNyKpOZDsHXY1fs6C0zo8jv4NDK_EvQ/viewform

# ----- SECOND METHOD ----
```

```
9 # Interact FRED framework throughtout the online sandbox via the
    link: http://wit.istc.cnr.it/stlab-tools/fred_api/
```

Listing 6.1 : Interaction avec le framework FRED à l'aide de CURL.

La figure 6.2 est identique à la première sauf que j'ai mis en évidence les classes en Vert et les individus en Bleu qui sont pris en compte par l'algorithme d'alignement. Les individus sont considérés comme importants s'ils respectent la condition

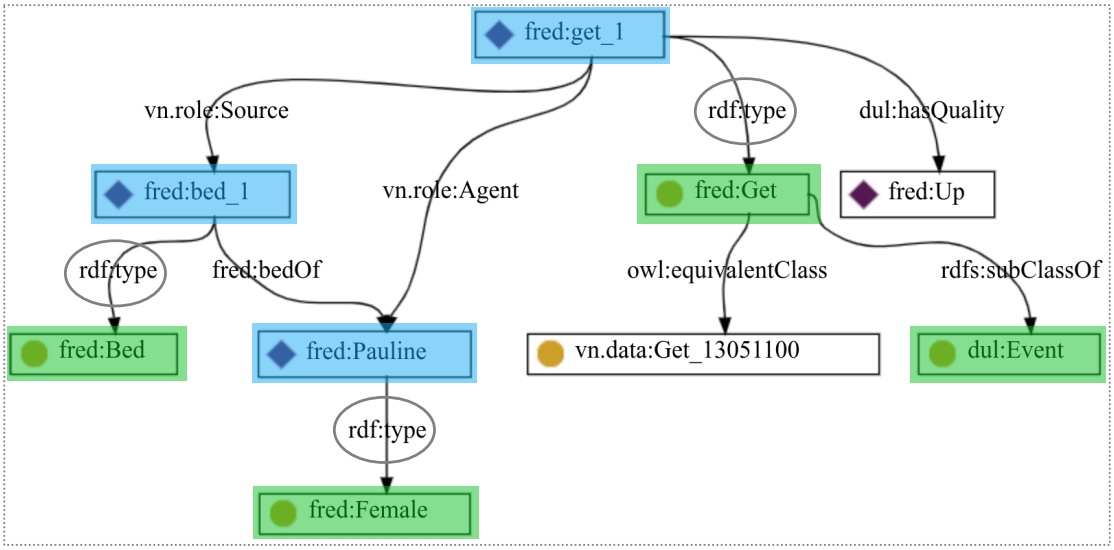


FIGURE 6.2 : Les classes et les individus importants sont mis en évidence en couleurs Vert et Bleu, respectivement.

(prédicat Ω) « $\Omega = \{type\}$ » des classes préfixées par le préfixe « $r = \{fred\}$ ». Par exemple, la classe « Female » est prise en compte, car elle est préfixée par le préfixe « fred ». L'instance/individu « Pauline » est prise en compte du fait qu'elle est de type « Female » et respecte le prédicat « type » avec une classe préfixée par le préfixe « fred ». Cependant, l'instance « fred:Up » semble importante puisqu'elle est préfixée par « fred », mais elle n'est pas d'un type préfixé par « fred », elle ne sera donc pas prise en compte. Notez que j'ai fixé ces conditions après avoir analysé plusieurs graphes d'action et conclu que dans le contexte de ces travaux, les classes importantes sont celles préfixées par les préfixes « fred » et « dul », et les individus importants sont ceux qui respectent le prédicat « type » avec les classes préfixées par le préfixe « fred ».

6.4 Alignement d'ontologies

Dans le cadre de ma collaboration avec le laboratoire DOMUS, et afin de présenter une preuve de concept, je m'appuie sur une étude de cas simple. Les problématiques liées à la grande complexité du problème seront étudiées dans de futurs travaux. L'objectif de ces travaux est de transférer la création de scénarios d'assistance du VA vers des commandes vocales, afin de simplifier cette tâche pour les soignants/proches, en gardant la notions de DIY. Les soignants/proches exprimeront une requête vocale pour créer un scénario d'assistance qui peut être décrit comme suit :

> Scénario 1 : It is 4 am, Pauline gets up from her bed, because, she is going to drink a glass of water, when she leaves her bed, the luminous path of her room lights up.

Le scénario 1 est très complexe à traiter en une seule fois, c'est pourquoi il sera décomposé en une série d'actions élémentaires :

- Pauline gets up from her bed.
- Detection in the bedroom.
- Turns on the light path.

Notons que le mécanisme utilisé pour décomposer le scénario dépasse le cadre de cette thèse, il s'agit de travaux de recherche en cours au sein du laboratoire DOMUS. Concernant mon travail de recherche, je ne me focalise que sur l'alignement automatique des actions : l'algorithme d'alignement met en relation les graphes de connaissances de ces actions avec l'ontologie de la maison intelligente, appelé OntoDomus (figure 3.1), pour extraire toutes les entités impliquées dans la réalisation de ces actions.

6.4.1 Contraintes d'alignement

L'ontologie OntoDomus et les graphes de connaissances des actions doivent répondre à certains critères pour que l'algorithme d'alignement soit correctement appliqué.

1. L'ontologie OntoDomus O doit être complète et explicite dans la mesure du possible. Ainsi, une description complète de l'environnement comprend des objets physiques connectés (par exemple, chaise, table, robinet, etc.), des lieux physiques (par exemple, des chambres, des salles de bains, des cuisines, etc.), des éléments de l'environnement (par exemple, des meubles, des appareils électroménagers, des sanitaires, etc.). Ces listes ne sont pas exhaustives.

2. L'action extraite du scénarios doit être aussi concise et explicite que possible.

3. Le graphe de connaissances de l'action A doit être obtenu en utilisant l'API FRED, car certaines caractéristiques de FRED sont actuellement utilisées dans le processus d'alignement.

Inspiré de ces recherches [XY18, ZXY+21, XW15], l'algorithme que je propose est basé sur des ressources statiques r, un seuil γ et un ensemble de conditions Ω données par un expert. L'algorithme a comme entrée deux types de connaissances différentes, la première est l'ontologie OntoDomus O et la seconde est le graphe de connaissances d'une action A, comme « Pauline gets up from her bed », un paramètre γ, une ressource r et quelques conditions Ω sous forme de prédicats qui seront pris en compte pour aligner seulement les instances qui les satisfont. En l'occurrence, il s'agit de :

- r un ensemble de préfixes, $r = \{fred, dul\}$ qui est utilisé pour ne prendre en compte que les classes pertinentes.

- γ un seuil (mesure de confiance) qui est utilisé à la fin du processus d'alignement pour décider si le résultat sera pris en compte ou non.

- Ω un ensemble de conditions (prédicats) $\Omega = \{type\}$ qui est utilisé pour ne prendre que les individus pertinents.

6.4.2 Algorithme d'alignement

Cette section présente l'algorithme d'alignement entre le graphe de connaissances d'une action et l'ontologie OntoDomus. L'action est extraite de la requête vocale « scénario » des soignants/proches puis convertie en graphe de connaissances. Seules les classes et les instances qui respectent les ressources r et les conditions (prédicats) Ω sont prises en compte par l'algorithme d'alignement. Enfin, un seuil γ est utilisé pour effectuer une évaluation finale de l'alignement.

Alignement des classes

Le framework FRED renvoie toutes les classes dans un format unique : $prefix : class\_name$. Pour commencer, A_p est un graphe de connaissances dérivé du graphe de connaissances de l'action A et contient toutes les classes préfixées par r et leurs individus :

$$A_p = \{c \mid c \in C(A), pref(c) \in r\}, \tag{6.1}$$

où $pref(c)$ est une fonction qui renvoie le $prefix$ approprié de la classe c. Rappelons que l'écriture $C(A)$ signifie l'ensemble des classes de A.

Afin d'obtenir l'alignement le plus probable entre les classes $C(O)$ de l'ontologie O et les classes $C(A_p)$ du graphe dérivé A_p, la matrice de similarité des classes « CSM » est calculée :

$$CSM_{(C(A_p),C(O))} = \begin{array}{c} \\ c_1^{A_p} \\ \vdots \\ c_m^{A_p} \end{array} \begin{array}{c} \begin{array}{ccc} c_1^O & \dots & c_n^O \end{array} \\ \begin{bmatrix} csm(c_1^{A_p}, c_1^O) & \dots & csm(c_1^{A_p}, c_n^O) \\ \vdots & \ddots & \vdots \\ csm(c_m^{A_p}, c_n^O) & \dots & csm(c_m^{A_p}, c_n^O) \end{bmatrix} \end{array}. \tag{6.2}$$

Notons $CSM_{ij} = csm(c_i^{A_p}, c_j^O)$. La valeur de chaque cellule $csm(c_i^{A_p}, c_j^O)$ dans CSM est obtenue par l'interaction avec le framework SEMATCH. L'ensemble $C_e^{A_p,O}$ contient les équivalences $(c_i^{A_p}, c_j^O)$ de la classe $c_i^{A_p}$ avec la classe c_j^O et est défini par ce qui suit :

$$\begin{aligned} C_e^{A_p,O} = \{(c_i^{A_p}, c_j^O) \mid \forall c_i \in C(A_p), \exists! c_j \in C(O), \\ CSM_{ij} = \max_{\{1 \le k \le n\}} CSM_{ik} \wedge CSM_{ij} \ge \gamma\}. \end{aligned} \tag{6.3}$$

$C_e^{A_p,O}$ contient toutes les classes avec leurs équivalences uniques qui satisfont le seuil γ. L'unicité est garantie par la recherche du maximum ; cette contrainte peut être corrompue s'il y a deux maximums.

Dorénavant, je ne considère que les classes $C(A_p)$ de l'action A_p et leur équivalent dans l'ontologie OntoDomus O. Elles sont représentées respectivement par l'équivalence $(c_i^{A_p}, c_j^O) \in C_e^{A_p,O}$, considérant que l'ontologie OntoDomus est une ontologie complète pour les systèmes AAL et que l'action demandée est dans la portée des scénarios AAL. Ainsi, il y aura toujours une injection de $C(A_p)$ dans $C(O)$ et une bijection $(\rightarrowtail\!\!\!\!\!\rightarrow)$ de $C_e^{A_p}$ dans C_e^O. Ce qui donne :

$$\begin{aligned} \exists! C_e^{A_p} \subseteq C(A), \exists! C_e^O \subset C(O) \mid C_e^{A_p} \rightarrowtail\!\!\!\!\!\rightarrow C_e^O \\ \Leftrightarrow \\ (C_e^{A_p}, C_e^O). \end{aligned} \tag{6.4}$$

En d'autre termes, $C_e^{A_p}$ et C_e^O ne contiennent que les classes qui sont équivalentes entre elles.

Alignement des individus

Considérons $I_\Omega^{C_e^{A_p}}$ l'ensemble des individus qui respectent les relations R données dans Ω avec les classes alignées $C_e^{A_p}$:

$$I_\Omega^{C_e^{A_p}} = \{i \mid i \in I(C_e^{A_p}), \exists c \in C_e^{A_p}, \exists R \in \Omega, iRc\}, \tag{6.5}$$

où $I(C_e^{A_p})$ représente les individus I des classes alignées $C_e^{A_p}$.

Considérons $I_\Omega^{C_e^O}$ l'ensemble qui contient les individus qui respectent les relations R données dans Ω avec les classes alignées C_e^O :

$$I_\Omega^{C_e^O} = \{i \mid i \in I(C_e^O), \exists c \in C_e^O, \exists R \in \Omega, iRc\}, \tag{6.6}$$

où $I(C_e^O)$ représente les individus I des classes alignées C_e^O.

En conservant le même processus que dans la section précédente pour obtenir l'alignement le plus probable entre les ensembles d'individus, $I_\Omega^{C_e^{A_p}}$ et $I_\Omega^{C_e^O}$, on obtient la matrice de similarité des individus (ISM) suivante :

$$ISM_{(I_\Omega^{C_e^{A_p}}, I_\Omega^{C_e^O})} = \begin{array}{c} \\ i_1^{C_e^{A_p}} \\ \vdots \\ i_{m'}^{C_e^{A_p}} \end{array} \begin{array}{c} \begin{array}{ccc} i_1^{C_e^O} & \dots & i_{n'}^{C_e^O} \end{array} \\ \begin{bmatrix} ism(i_1, i_1) & \dots & ism(i_1, i_{n'}) \\ \vdots & \ddots & \vdots \\ ism(i_{m'}, i_1) & \dots & ism(i_{m'}, i_{n'}) \end{bmatrix} \end{array}. \tag{6.7}$$

L'ensemble $I_e^{A_p,O}$ contient les équivalences $(i_l^{C_p^A}, i_k^{C_e^O})$ de l'individu $i_l^{C_e^{A_p}}$ avec l'individu $i_k^{C_e^O}$ et est défini par ce qui suit :

$$\begin{aligned} I_e^{A_p,O} = \{(i_l^{C_p^A}, i_k^{C_e^O}) \mid \forall i_l \in I_\Omega^{C_e^{A_p}}, \exists! i_k \in I_\Omega^{C_e^O}, \\ ISM_{lk} = \max_{\{1 \le j \le n'\}} ISM_{lj} \wedge ISM_{lk} \ge \gamma \wedge c(i_l) = c(i_k)\}, \end{aligned} \tag{6.8}$$

où $c(i_l)$ et $c(i_k)$ représentent respectivement les classes d'individus i_l et i_k.

Une fois $C_e^{A_p,O}$ et $I_e^{A_p,O}$ construits, les interactions peuvent être réalisées avec l'ontologie ontoDomus via SPARQL [3] [STE09], afin d'obtenir les éléments de l'environnement (capteurs, haut-parleurs, etc.) qui sont impliqués lors de l'exécution de l'action.

6.4.3 Algorithme d'alignement par étapes

L'algorithme d'alignement s'exécute automatiquement en deux phases résumées respectivement dans les figures 6.3 et 6.4. Chaque phase est divisée en plusieurs étapes. En supposant que l'étape 0 « step 0 » de la première phase (figure 6.3) a déjà été exécutée, dans sa première étape « step 1 », l'algorithme interagit avec l'API du service REST FRED en lui envoyant des actions sous format de texte naturel pour récupérer leurs graphes de connaissances. Durant la deuxième phase

3. SPARQL est un langage de requête RDF capable d'extraire et de manipuler des données stockées au format RDF.

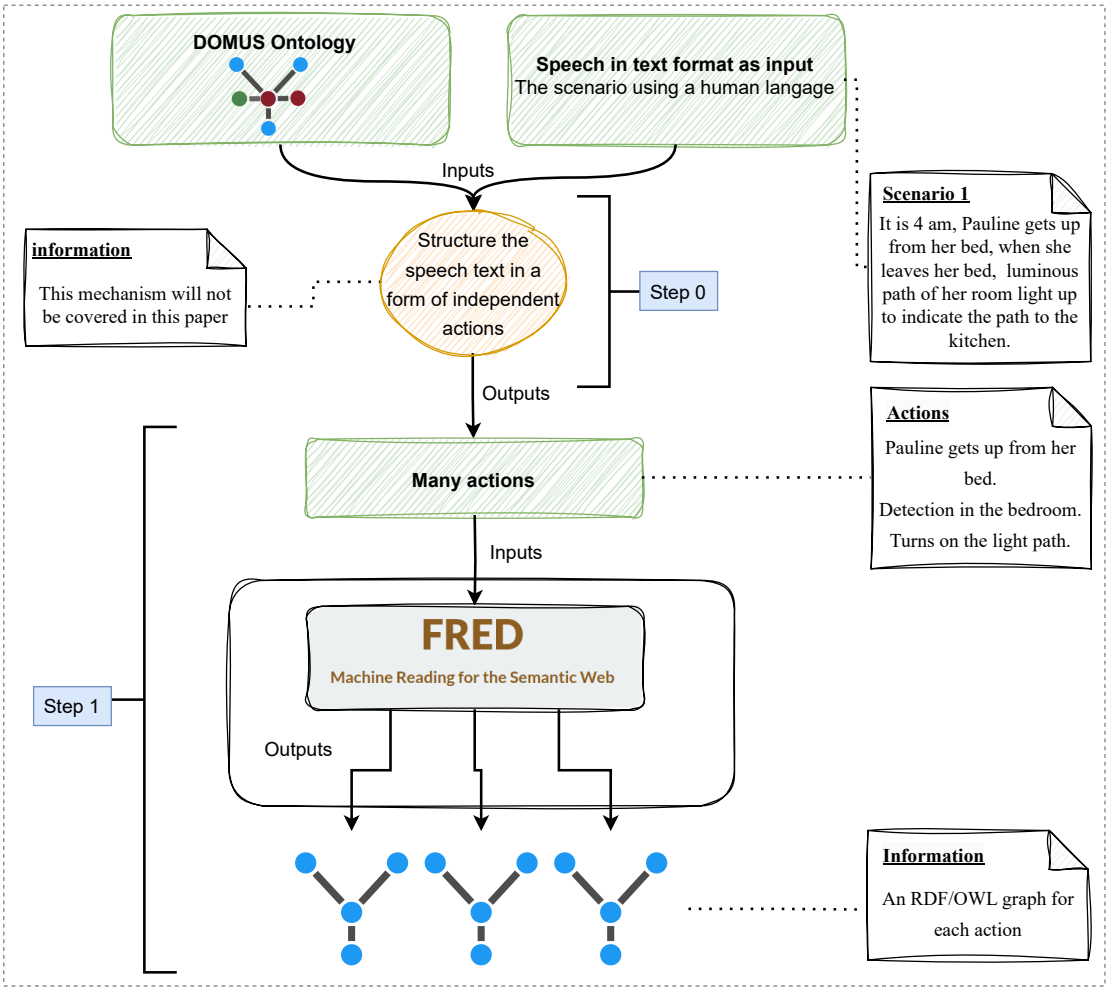


FIGURE 6.3 : Illustration de la transformation par étapes des actions en graphe de connaissances RDF.

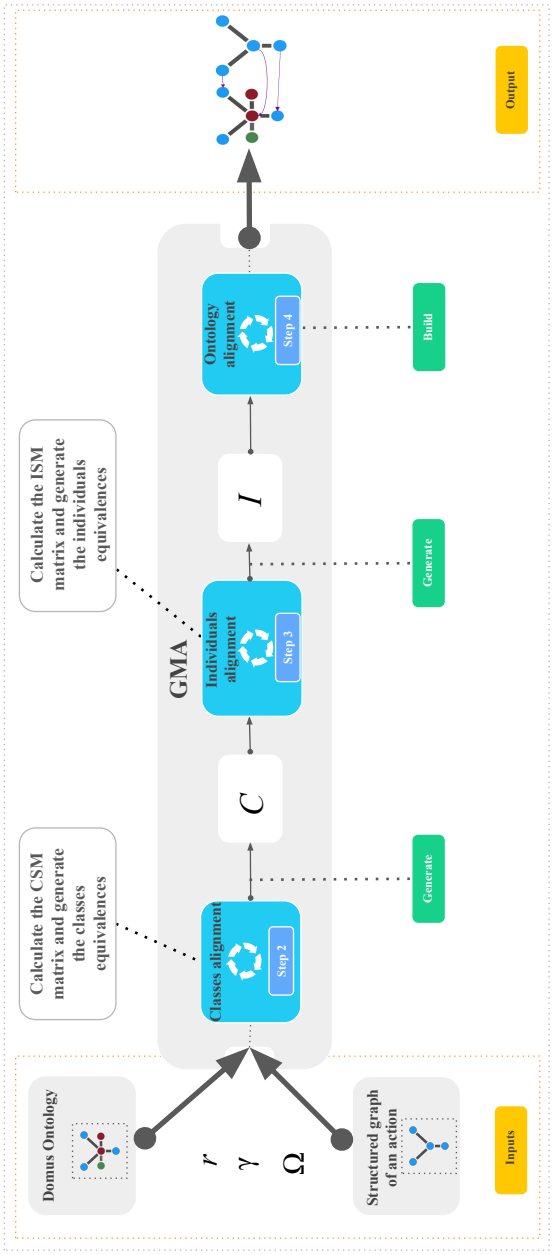


FIGURE 6.4 : Illustration de l'algorithme d'alignement par étapes.

(figure 6.4), les graphes d'actions sont utilisés comme entrées pour le modèle d'alignement d'ontologies. La deuxième et la troisième étapes « step 2 » et « step 3 » sont consacrées à l'alignement des classes et des individus à l'aide des matrices de similarité sémantique (CSM, ISM). La dernière étape « step 4 » vise à relier l'ontologie OntoDomus aux graphes d'actions, à savoir l'alignement des classes et leurs individus.

6.4.4 Illustration

En prenant le même exemple « Pauline gets up from her bed » que précédemment, la figure 6.2 illustre son graphe de connaissances généré par FRED. Les classes et les individus pris en compte sont respectivement marqués en Vert et en Bleu. Ainsi, les entrées de l'algorithme sont :

- L'ontologie OntoDomus [NPG22], brièvement illustrée dans la figure 6.5 et qui est une version partielle de celle donnée en exemple au chapitre 3, figure 3.1.

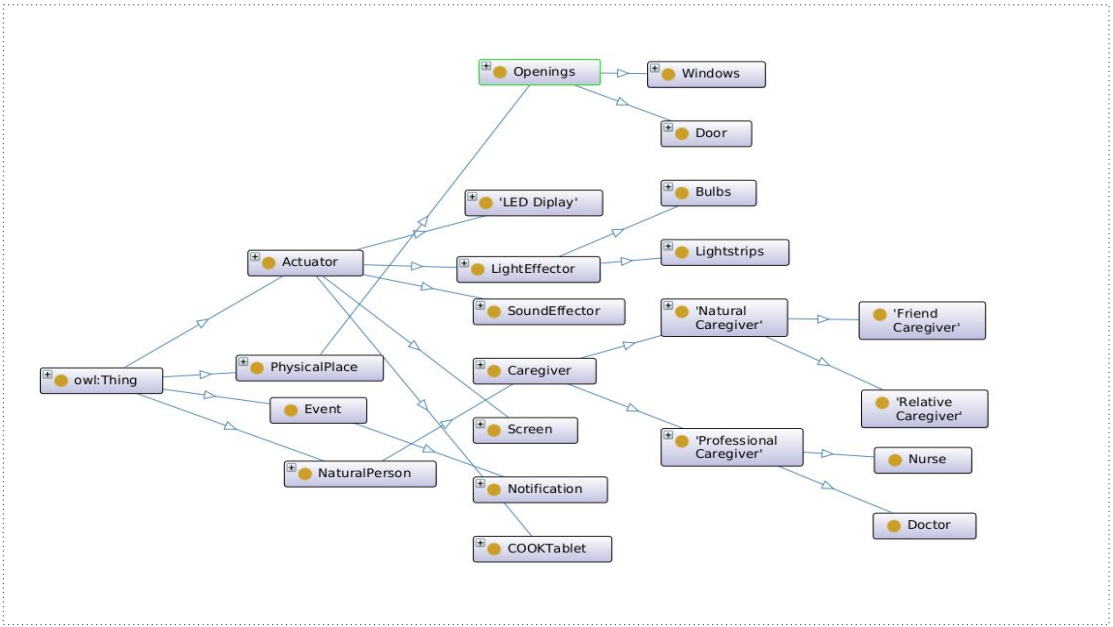


FIGURE 6.5 : Une partie des classes de premier niveau de l'ontologie OntoDomus.

- Le graphe de connaissances de l'action A « Pauline gets up from her bed », illustré dans la figure 6.2.

- L'ensemble r de ressources (préfixes) utilisé pendant le processus d'alignement est défini comme suit :

$$r = \{fred, dul\}.$$

- L'ensemble Ω des conditions (prédicats) utilisé au cours du processus d'alignement est défini comme suit :

$$\Omega = \{type\}.$$

Alignement des classes

Seules les classes qui respectent la propriété 6.1 sont prises en compte pour calculer la matrice de similarité de classes « CSM ». Ainsi les classes prises en compte sont :

$$A_p = \{Get, Bed, Female, Event\}.$$

L'ontologie OntoDomus est représentée par O, la matrice CSM est définie comme suit :

$$CSM_{(C(A_p),C(O))} = \begin{array}{c} \\ Get \\ Bed \\ Female \\ Event \end{array} \begin{array}{c} \begin{array}{ccccccc} Get & Bed & Bathroom & Person & Event & \dots \end{array} \\ \begin{bmatrix} 1 & 0.055 & 0.047 & 0.666 & 0.148 & \dots \\ 0.055 & 1 & 0.257 & 0.219 & 0.145 & \dots \\ 0.058 & 0.190 & 0.135 & 0.627 & 0.166 & \dots \\ 0.148 & 0.145 & 0.106 & 0.193 & 1 & \dots \end{bmatrix} \end{array}$$

Par conséquent, l'application de l'équation 6.3 avec un seuil de $\gamma = 0.60$ sur la matrice CSM donne le résultat suivant :

$$C_e^{A_p,O} = \{(Get, Get), (Bed, Bed), (Female, Person), (Event, Event)\}.$$

Il est à noter que le seuil $\gamma = 0.60$, dans ce contexte, a été choisi de manière à obtenir au moins une équivalence dans chaque ligne.

Alignement des individus

Les individus pris en compte sont ceux qui respectent le prédicat Ω avec les classes alignées $C_e^{A_p,O}$. L'application des équations 6.5 et 6.6 sur $C_e^{A_p,O}$ donne le résultat suivant :

- $I_\Omega^{C_p}$ l'ensemble des individus qui respectent les prédicats Ω avec les classes alignées de A_p, défini comme suit :

$$I_\Omega^{C_p} = \{get, Pauline, bed\}$$

- I_{Ω}^{O} l'ensemble des individus qui respectent les prédicats Ω avec les classes alignées de O, défini comme suit :

$$I_{\Omega}^{O} = \{getup, get, bed, chair, Pauline\}$$

La matrice ISM est définie comme suit :

$$ISM_{(I_{\Omega}^{C_e^{A_p}}, I_{\Omega}^{C_e^{O}})} = \begin{array}{c} \\ get \\ Pauline \\ bed \end{array} \begin{array}{c} \begin{array}{cccccc} getup & get & bed & chair & Pauline & Jack & \dots \end{array} \\ \begin{bmatrix} 0,047 & 1 & 0,555 & 0,122 & 0 & 0,070 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0,250 & 0.055 & 1 & 0,522 & 0 & 0,250 & \dots \end{bmatrix} \end{array}$$

Par conséquent, l'application de l'équation 6.8 avec un seuil de $\gamma = 0.60$ sur la matrice ISM donne le résultat suivant :

$$I_{e}^{A_p,O} = \{(get, get), (bed, bed)\}.$$

A partir de $C_{e}^{A_p,O}$ et $I_{e}^{A_p,O}$, on peut obtenir les composants (capteurs, haut-parleurs, etc.) de l'environnement provenant de l'ontologie OntoDomus impliqués dans la réalisation de l'action A. L'interaction avec l'ontologie OntoDomus peut être réalisée à l'aide du langage SPARQL.

En revanche, il existe des similarités étranges dans la matrice des individus, comme l'individu *bed* avec l'individu *Jack*, malgré la faiblesse de cette similarité, qui ne devrait normalement pas avoir lieu selon mon point de vue. Selon la documentation officielle [Zhu], SEMATCH s'appuie sur des jugements humains concernant les paires de mots qui peuvent ne pas être identiques dans des applications réelles, ce qui explique la similarité entre les deux instances *bed* et *Jack*. De plus, la similarité entre les individus *Pauline* est nulle, ce qui les élimine du résultat $I_{e}^{A_p,O}$, car SEMATCH ne dispose pas de l'ontologie pour tous les noms propres. En outre, SEMATCH ne peut pas donner de similarité entre des mots composés tels que « get up » et « get », pour cela, je les ai attachés. Pour surmonter ces problèmes, SEMATCH doit être entraîné sur des ensembles de données relatives au domaine d'application, dans notre cas la maison intelligente.

6.5 Conclusion

Dans ce chapitre, j'ai posé la première pierre des travaux de recherche sur l'alignement d'ontologies dans le contexte du système AAL. La contribution est un algorithme d'alignement entre deux graphes de connaissances : un graphe de connaissances structurées des actions et l'ontologie OntoDomus d'une maison intelligente.

L'algorithme calcule la similarité entre les classes et les individus en utilisant la mesure de similarité sémantique comme preuve de concept pour générer deux

matrices : la première est la matrice de similarité de classes, utilisée pour calculer la similarité entre les classes, la seconde est la matrice de similarité des individus, utilisée pour calculer la similarité entre les individus des classes alignées. À la fin de ce processus, une interaction peut être établie avec l'ontologie OntoDomus pour récupérer les objets connectés qui seront impliqués dans l'exécution de l'action. Cette interaction peut être faite via le langage de requête RDF SPARQL.

Cette première étape comporte de nombreuses limites, la plus importante étant celle du graphe généré par FRED, car le processus de travail de FRED est général (sur un domaine ouvert basé sur différentes ontologies telles que DBpedia, FramNet, WordNet, VerbNet, etc.) et n'est pas encore contextualisé pour être exécuté dans le contexte de la maison intelligente. Une autre limite concerne la connaissance des performances de l'algorithme, à savoir, sa précision. Les travaux futurs mesureront d'abord la précision de l'algorithme en testant des centaines d'actions, ainsi qu'en expérimentant des techniques de Machine Learning, en particulier des approches semi-supervisées. La contextualisation des graphes de connaissances des actions générés par FRED sera également étudiée.

Conclusions et perspectives

Conclusion

La problématique étudiée lors de ma thèse porte, de manière générale, sur la capacité d'un système logiciel de communiquer avec ses utilisateurs. Avec la 4e révolution industrielle et numérique que nous vivons, l'informatique est omniprésente et nous sollicite en permanence au travers des multiples objets connectés qui nous entourent. Cette sur-sollicitation est une problématique de recherche connue, dont la solution se nomme la calme technologie. C'est-à-dire une technologie présente, mais qui ne sollicite les utilisateurs que lorsque c'est réellement nécessaire ou selon leurs besoins propres. Pour aller vers cette calme technologie, les objets sages ont été créés avec comme objectif d'apprendre en totale autonomie leur fonctionnement et la manière dont ils sont utilisés. Ainsi ils seront capables d'être proactifs et de s'adapter à leurs utilisateurs. Un premier framework a été développé et permet l'implémentation de tels objets en langage Java. Ce framework se base sur une boucle MAPE-K où le « Monitoring » correspond à l'acquisition de la connaissance sur l'objet lui-même et son utilisation, l'« Analyze » correspond au traitement de la connaissance pour en extraire de nouvelles, et « Plan » et « Execute » correspond à la capacité de réaction. En phase d'analyse, l'objet est capable de créer son propre graphe de comportement sous la forme d'un graphe état/transition (STG). Ce dernier lui permet de prédire l'état dans lequel il se trouvera avant l'exécution d'une de ses méthodes, où calculer la suite de méthodes qu'il doit exécuter pour arriver dans un état défini. Le problème majeur est que ce graphe est purement numérique avec presque aucune sémantique. Ceci rend l'objet sage incapable de

communiquer avec ses utilisateurs et de leur expliquer pourquoi il a réagi de telle ou telle manière.

Pour résoudre ce problème, je propose une première approche qui consiste à aller chercher la sémantique absente du STG dans un graphe de conception, l'IOSTS. Ce dernier étant conceptuel, il porte beaucoup de sémantique et du fait qu'il représente le comportement d'un système, il est proche du STG généré par les WO. La problématique consiste donc à proposer un appariement entre ces deux graphes. Pour ce faire, j'ai choisi de m'appuyer sur les domaines des attributs et variables caractérisant chacun des graphes. La première solution que je propose s'apparente à une preuve de concept simplifiée puisqu'elle est univariée et ne prend en compte que le minimum d'information liant les deux graphes. En d'autres termes, elle ne prend en compte qu'un seul attribut et une seule variable dans chaque graphe respectif. Naturellement, dans un second temps, je généralise cette première solution en proposant une approche multivariée qui prend en compte l'ensemble des attributs et variables des graphes et cherche à maximiser les couvertures de domaine dans l'appariement de ces graphes. Cette seconde solution propose un ensemble réduit de solutions très cohérentes d'un point de vue numérique et utilisables en l'état. Cependant, elle nécessite une décision d'expert afin de déterminer la solution qui, en plus d'une cohérence numérique, a un sens pour l'utilisateur.

Afin d'aller plus loin dans la solution précédente et de faciliter l'intervention d'un expert, j'ai introduit une contrainte supplémentaire dans mon algorithme d'appariement. En plus des contraintes numériques sur les domaines, je propose un raffinement de l'appariement en prenant en compte la sémantique des noms d'attributs et variables. En effet, les deux étant définis par un humain, concepteur et développeur, leur sens devrait être proche. Pour ce faire, je m'appuie sur le framewok SEMATCH qui me fournit une mesure de distance sémantique entre deux termes. Ainsi je peux déterminer la matrice de distance entre chaque attribut et variable des deux graphes correspondants. Enfin, ayant deux matrices de distances, l'une sémantique et l'autre numérique se basant sur les domaines, la problématique consiste alors à les fusionner. Je présente donc différentes solutions avec leurs avantages et inconvénients.

Cette première partie de mes travaux de thèse m'a amené à chercher à augmenter le niveau sémantique d'une entité logicielle opérationnelle, un WO, afin qu'il puisse communiquer avec ses utilisateurs. J'ai pu mettre en place une collaboration avec le laboratoire DOMUS à Sherbrooke, afin de travailler sur le problème inverse. C'est-à-dire : « comment, à partir une expression à haut niveau sémantique, contrôler un système opérationnel ? ». De manière plus précise, la problématique est de réaliser une suite d'actions et réactions d'un système opérationnel – une maison connectée pour le maintien à domicile – à partir d'un scénario exprimé verbalement. Cette problématique pose le problème de l'alignement d'ontologie, en

l'occurrence celle pouvant être extraite du scénario et celle de la maison connectée. Dans le cadre de ma thèse, je me suis appuyé sur le framework FRED pour extraire l'ontologie de la description verbale du scénario, puis de SEMATCH, comme dans les travaux précédents, pour déterminer la distance sémantique entre les classes et individus de chaque ontologie. Ce travail est un travail préliminaire que j'ai effectué en vue de potentielles futures collaborations entre les laboratoires DOMUS et LISTIC.

Perspectives

Concernant directement ma thèse, une partie de mes travaux peuvent être approfondis. Les matrices de degrés de compatibilités présentées dans le chapitre 5 sont utilisées de manière « brute », c'est-à-dire sans considérer « a priori » l'importance des degrés forts et la négligeabilité des degrés faibles ou nuls. Cette approche oblige à calculer toutes les combinaisons en se basant sur tous les couples. Il serait plus judicieux, dans des travaux futurs, d'étudier des jeux de permutations sur ces matrices de degrés afin de faire remonter les degrés les plus forts et ainsi traiter en premier lieu ceux qui donneront les combinaisons les plus probables. Ainsi, en s'appuyant sur le seuil défini par l'expert, un certain nombre de combinaisons pourrait être écarté avant même d'être calculé.

Mes travaux présentent différentes méthodes afin de combiner les deux approches de mesure des degrés – sur domaine et sémantique – ainsi que pour calculer les degrés de compatibilité des combinaisons. Le choix de la méthode utilisée dans le cadre de ma thèse s'est effectué de manière logique selon le contexte particulier des WO. Une étude plus approfondie des avantages et inconvénients de chacune d'entre elles est envisagée afin de déterminer de manière factuelle leurs impacts respectifs sur le processus de décision.

Mes travaux de thèses m'ont amené, au sein du laboratoire LISTIC, à augmenter le niveau sémantique des WO et, au sein du laboratoire DOMUS, à descendre d'une expression orale vers un scénario de commandes à exécuter. La suite naturelle de mes travaux est de lier ces deux axes afin que les WO soient capables de communiquer verbalement avec un humain et de comprendre et d'exécuter les requêtes d'un utilisateur. De plus, comme indiqué précédemment, les travaux que j'ai débutés au laboratoire DOMUS sont préliminaires. Ils mériteraient d'être approfondis par des études de scénarios plus complexes, en intégrant des phases d'apprentissage afin que les frameworks génériques que j'ai utilisés se spécialisent plus dans cette tâche particulière et donnent des résultats avec une plus grande fiabilité.

Ces travaux m'ont amené à utiliser plusieurs ontologies à différents niveaux pour lier un comportement purement numérique à des requêtes humaines. Les on-

tologies ont un grand apport dans mes travaux et peuvent également être utilisées à d'autres fins, notamment dans le cadre du contexte général de cette thèse, les WO. Puisqu'un WO peut se connaître lui-même et par lui-même, il peut découvrir le comportement d'un système à travers la documentation (Docstring Python, JavaDoc, etc.). Cette documentation est écrite en langage naturel, qui peut être transformé en graphe de connaissances grâce au framework FRED. En utilisant mes travaux sur les ontologies, on peut envisager un alignement entre le graphe de connaissances de la documentation d'une application et une ontologie du domaine de cette application, pour une interaction en langage naturel.

L'une des perspectives des WO est de rendre accessible au logiciel lui-même l'ensemble de ses éléments de conception. Dans ma thèse, c'est ce qui est fait avec l'IOSTS, mais il serait également intéressant de rendre accessibles des éléments comme les documentations, les commentaires, les tests, etc. Ceci permettrait d'augmenter la sagesse et l'intelligence du logiciel, c'est-à-dire sa connaissance de lui-même et sa capacité à interagir de manière intelligente et calme avec l'utilisateur. Les approches ontologiques peuvent s'avérer utiles pour donner la capacité au logiciel d'utiliser tous ces éléments qui lui sont annexes. Outre l'intégration de nouveaux modèles ou diagrammes, je pense que la future étape la plus importante sera de donner la capacité au logiciel d'utiliser lui-même sa documentation et la sémantique qu'elle renferme pour une communication naturelle avec l'humain.

Bibliographie personnelle

Journaux :

[DAMV23a] **Abdelhafid Dahhani**, Ilham Alloui, Sébastien Monnet and Flavien Vernier. A Graph Matching Algorithm to Extend Software Wise Systems with Human Semantic. International Journal On Advances in Intelligent Systems, 16(1 & 2), 2023.

Conférences avec actes et comité de lecture :

[DAMV22a] **Abdelhafid Dahhani**, Ilham Alloui, Sébastien Monnet and Flavien Vernier. Towards a Semantic Model for Wise Systems - A Graph Matching Algorithm ADVCOMP 2022, The Sixteenth International Conference on Advanced Engineering Computing and Applications in Sciences, November 2022, ("Best Paper Award").

[DAMV23b] **Abdelhafid Dahhani**, Ilham Alloui, Sébastien Monnet and Flavien Vernier. A Graph Matching Algorithm to extend Wise Systems with Semantic. 18th Conference on Computer Science and Intelligence Systems FedCSIS 2023. Track : 29th Conference on Knowledge Acquisition and Management, September 2023.

[Submited] **Abdelhafid Dahhani**, Kenfack Ngankam Hubert, Sylvain Giroux, Ilham Alloui, Sébastien Monnet and Flavien Vernier. Ontology Alignment : First Step towards Voice Control for Smart-Home Assistive Scenarios. International Conference on Cooperative Information Systems, (submitted).

[InProgress] **Abdelhafid Dahhani**, Ilham Alloui, Sébastien Monnet and Flavien Vernier. Distance maching : Towards near-human decisiveness. (Writing in progress).

Bibliographie

[AAA18] Ashraf Ali, Fayez Alfayez, and Hani Alquhayz. Semantic similarity measures between words : A brief survey. Sci. Int.(Lahore), 30(6) :907–914, 2018.

[Abd98] Abdulrahim Mohammad Abdulkader. Parallel algorithms for labeled graph matching. Colorado School of Mines1500 Illinois St. Golden, CO, 1998.

[Abe93] Manfred Aben. Formally specifying reusable knowledge model components. Knowledge Acquisition, 5(2) :119–141, 1993.

[ABPV18] Ilham Alloui, Eric Benoit, Stéphane Perrin, and Flavien Vernier. WIoT : Interconnection between wise objects and iot. In ICSOFT 2018, the 13th International Conference on Software Technologies, Porto, Portugal, July 2018.

[ABPV19] Ilham Alloui, Eric Benoit, Stephane Perrin, and Flavien Vernier. Wise Objects for IoT (WIoT) : Software Framework and Experimentation. Communications in Computer and Information Science, pages 349–371, Section 2/2.3, August 2019.

[AD14] Kitty Ahuja and Heena Dangey. Autonomic computing : An emerging perspective and issues. In 2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT), pages 471–475, February 2014.

[ADB+99] Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In HUC '99 : Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing, pages 304–307, London, UK, 1999. Springer-Verlag.

[AEV15] Ilham Alloui, David Esale, and Flavien Vernier. Wise objects for calm technology. In Proceedings of the 10th International Conference on Software Engineering and Applications - ICSOFT-EA, (ICSOFT 2015), pages 468–471. INSTICC, SciTePress, 2015.

[AGP+17] Harith Alani, Aldo Gangemi, Valentina Presutti, Diego Reforgiato Recupero, Andrea Giovanni Nuzzolese, Francesco Draicchio, and Misael Mongiovì. Semantic web machine reading with fred. Semant. Web, 8(6) :873–893, January 2017.

[AHZ13] Dhaminda B. Abeywickrama, Nicklas Hoch, and Franco Zambonelli. Simsota : Engineering and simulating feedback loops for self-adaptive systems. In C3S2E '13 : Proceedings of the International C* Conference on Computer Science and Software Engineering, page 67–76, New York, NY, USA, 2013. Association for Computing Machinery.

[AJ18] Nadia Alboukaey and Ammar Joukhadar. Ontology matching as regression problem. Journal of Digital Information Management, 16(1), 2018.

[AKH+16] Muhammad Bilal Amin, Wajahat Ali Khan, Shujaat Hussain, Dinh-Mao Bui, Oresti Banos, Byeong Ho Kang, and Sungyoung Lee. Evaluating large-scale biomedical ontology matching over parallel platforms. IETE Technical Review, 33(4) :415–427, 2016.

[Alla] Connectivity Standards Alliance. Zigbee : zigbee aliance. http://www.zigbee.org, https://csa-iot.org/. Accessed : 2021-06-11.

[Allb] Z-Wave Alliance. Z-Wave : z-wave aliance. https://z-wavealliance.org/. Accessed : 2021-06-11.

[AMF17] Siham Amrouch, Sihem Mostefai, and Muhammad Fahad. Decision trees in automatic ontology matching. Int. J. Metadata Semant. Ontologies, 11(3) :180–190, January 2017.

[APdN16] Tiago Brasileiro Araújo, Carlos Eduardo Santos Pires, Thiago Pereira da Nóbrega, and Dimas C. Nascimento. A fine-grained load balancing technique for improving partition-parallel-based ontology matching approaches. Knowledge-Based Systems, 111 :17–26, 2016.

[ARE98] Finch A M, Wilson R C, and Hancock E R. Symbolic graph matching with the em algorithm. Pattern Recognition, 31(11) :1777–1790, November 1998.

[ARS15] Paolo Arcaini, Elvinia Riccobene, and Patrizia Scandurra. Modeling and analyzing mape-k feedback loops for self-adaptation. In 2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, pages 13–23, 2015.

[AV17] Ilham Alloui and Flavien Vernier. A Wise Object Framework for Distributed Intelligent Adaptive Systems. In ICSOFT 2017, the 12th International Conference on Software Technologies, Madrid, Spain, July 2017.

[AV18] Ilham Alloui and Flavien Vernier. WOF : Towards Behavior Analysis and Representation of Emotions in Adaptive Systems. Communications in Computer and Information Science, 868 :244–267, 2018.

[AYH13] Egozi Amir, Keller Yosi, and Guterman Hugo. A probabilistic approach to spectral graph matching. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(1) :18–27, 2013.

[BA83] Horst Bunke and Gudrun Allermann. Inexact graph matching for structural pattern recognition. Pattern Recognition Letters, 1(4) :245–253, 1983.

[Bas94] David A. Basin. A term equality problem equivalent to graph isomorphism. Information Processing Letters, 51(2) :61–66, 1994.

[BAT97] Pim Borst, Hans Akkermans, and Jan Top. Engineering ontologies. International journal of human-computer studies, 46(2-3) :365–406, 1997.

[BB97] Willem Nico Borst and W.N. Borst. Construction of Engineering Ontologies for Knowledge Sharing and Reuse. PhD thesis, University of Twente, Netherlands, September 1997.

[BBB05a] Oana Bucur, Philippe Beaune, and Olivier Boissier. Representing context in an agent architecture for context-based decision making. CEUR Workshop Proceedings, 136, January 2005.

[BBB+05b] Oana Bucur, Philippe Beaune, Olivier Boissier, et al. Representing context in an agent architecture for context-based decision making. In Proceedings of the Workshop on Context Representation and Reasoning (CRR'05), Paris, France, volume 5, 2005.

[BBZP19] Ghada Besbes, Hajer Baazaoui-Zghal, and Yann Pollet. Semantic-based collaborative decisional system integrating fuzzy reasoning in an iot context. In 8th International Conference on Information Systems Development (ISD2019), August 2019.

[BC08] Dario Bonino and Fulvio Corno. Dogont - ontology modeling for intelligent domotic environments. In The Semantic Web - ISWC 2008, pages 790–803. Springer Berlin Heidelberg, 2008.

[BDMSG+09] Yuriy Brun, Giovanna Di Marzo Serugendo, Cristina Gacek, Holger Giese, Holger Kienle, Marin Litoiu, Hausi Müller, Mauro Pezzè, and Mary Shaw. Engineering Self-Adaptive Systems through Feedback Loops, pages 48–70. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[Ben02] Endika Bengoetxea. Inexact Graph Matching Using Estimation of Distribution Algorithms. PhD thesis, Ecole Nationale Supérieure des Télécommunications, Paris, France, December 2002.

[Ber01] D. Berlinski. The Advent of the Algorithm : The 300-Year Journey from an Idea to the Computer. Harvest book. Harcourt, 2001.

[BL11] Moussa Benaissa and Yahia Lebbah. Optimisation de l'extraction de l'alignment des ontologies avec la contrainte de différence. In Extraction et gestion des connaissances (EGC'2011), pages 401–406, 01 2011.

[BMB+16] Stephanie Blackman, Claudine Matlo, Charisse Bobrovitskiy, Ashley Waldoch, Mei Lan Fang, Piper J. Jackson, Alex Mihailidis, Louise Nygård, Arlene J. Astell, and Andrew Sixsmith. Ambient assisted living technologies for aging well : A scoping review. Journal of Intelligent Systems, 25 :55 – 69, 2016.

[BPBR99] C Boeres, A Perchant, I Bloch, and M Roux. A genetic algorithm for brain image recognition using graph non-bijective correspondence. Unpublished manuscript, 1999.

[Bro17] Alfred James Brown. Knight's Tours and Zeta Functions. PhD thesis, San José State University, 2017.

[Cam22] Cambridge Dictionary Online, 2022.

[Cau13] Augustin-Louis Cauchy. Géométrie. Mémoire sur la polyédrométrie, volume 3, page 169–189. Veuve Courcier, Imprimeur-Libraire pour les Mathématiques, 1812-1813.

[Cau09] Augustin-Louis Cauchy. Recherches sur les polyèdres (Premier Mémoire), volume 1 of Cambridge Library Collection - Mathematics, page 66–86. Cambridge University Press, 2009.

[CCI04] Boeres Maria C., Ribeiro Celso C., and Bloch Isabelle. A randomized heuristic for scene recognition by graph matching. In Ribeiro Celso C. and Martins Simone L., editors, Experimental and Efficient Algorithms, pages 100–113, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[CEM03] Robert Chatley, Susan Eisenbach, and Jeff Magee. Modelling a framework for plugins. SAVCBS 2003 Specification and Verification of Component-Based Systems, page 49, 2003.

[CFLP16] Francesco Chiti, Romano Fantacci, Michele Loreti, and Rosario Pugliese. Context-aware wireless mobile autonomic computing and communications : research trends and emerging applications. IEEE Wireless Communications, 23(2) :86–92, 2016.

[Cha77] G. Chartrand. Introductory Graph Theory. Dover Books on Mathematics Series. Dover, 1977.

[Cic99] Vincent A. Cicirello. Survey of graph matching algorithms. Technical report, Geometric and Intelligent Computing Laboratory, Drexel University, 1999.

[CJMR07] Camille Constant, Thierry Jéron, Hervé Marchand, and Vlad Rusu. Integrating Formal Verification and Conformance Testing for Reactive Systems. IEEE Transactions on Software Engineering, 33(8) :558–574, 2007.

[CJMR08] Camille Constant, Thierry Jéron, Hervé Marchand, and Vlad Rusu. Validation of Reactive Systems. In Modeling and Verification of Real-TIME Systems - Formalisms and software Tools, pages 51–76. Hermès Science, 2008.

[CSC01] Schellewald Christian, Roth Stefan, and Schnörr Christoph. Evaluation of convex optimization techniques for the weighted graph-matching problem in computer vision. In Radig Bernd and Florczyk Stefan, editors, Pattern Recognition, pages 361–368, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[CWH97] Andrew D.J. Cross, Richard C. Wilson, and Edwin R. Hancock. Inexact graph matching using genetic search. Pattern Recognition, 30(6) :953–970, 1997.

[DAMV22a] Abdelhafid Dahhani, Ilham Alloui, Sébastien Monnet, and Flavien Vernier. Towards a semantic model for wise systems - a graph matching algorithm. In ADVCOMP 2022, The Sixteenth International Conference on Advanced Engineering Computing and Applications in Sciences, November 2022.

[DAMV22b] Abdelhafid Dahhani, Ilham Alloui, Sébastien Monnet, and Flavien Vernier. Towards a semantic model for wise systems - a graph matching algorithm. ADVCOMP 2022, 34(7) :27–34, 2022.

[DAMV23a] Abdelhafid Dahhani, Ilham Alloui, Sébastien Monnet, and Flavien Vernier. A graph matching algorithm to extend software wise systems with human semantic. International Journal On Advances in Intelligent Systems, 16(1 & 2), 2023.

[DAMV23b] Abdelhafid Dahhani, Ilham Alloui, Sébastien Monnet, and Flavien Vernier. A graph matching algorithm to extend wise systems with semantic. In 18th Conference on Computer Science and Intelligence Systems FedCSIS 2023. Track : 29th Conference on Knowledge Acquisition and Management, September 2023.

[DAS01] Anind K Dey, Gregory D Abowd, and Daniel Salber. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Human–Computer Interaction, 16(2-4) :97–166, 2001.

[Der] William Dersch. Ibm shoebox.

[DGPN13] Francesco Draicchio, Aldo Gangemi, Valentina Presutti, and Andrea Nuzzolese. Fred : From natural language text to rdf and owl in one click. In The Semantic Web : ESWC 2013 Satellite Events, volume 7955 of LNCS, pages 263–267, May 2013.

[DHW+08] Stefan Deßloch, Mauricio Hernández, Ryan Wisnesky, Ahmed Radwan, and Jindan Zhou. Orchid : Integrating schema mapping and etl. Data Engineering, International Conference on, 0 :1307–1316, April 2008.

[Dij12] Edsger W Dijkstra. Selected Writings on Computing : A personal Perspective. Monographs in Computer Science. Springer, New York, NY, 1982 edition, December 2012.

[DP98] Thomas Davenport and Laurence Prusak. Working Knowledge : How Organizations Manage What They Know, volume 1. Harvard Business School Press, 1998.

[DRSC+12] Dries De Roeck, Karin Slegers, Johan Criel, Marc Godon, Laurence Claeys, Katriina Kilpi, and An Jacobs. I would diyse for it ! a manifesto for do-it-yourself internet-of-things creation. In Proceedings of the 7th Nordic Conference on Human-Computer Interaction : Making Sense Through Design, NordiCHI '12, page 170–179, New York, NY, USA, 2012. Association for Computing Machinery.

[dSI15] Grupo de Sistemas Inteligentes. Sematch framework. https://github.com/gsi-upm/sematch, 2015. Accessed : 2023-20-07.

[EF86] M. A. Eshera and King-Sun Fu. An image understanding system using attributed symbolic representation and inexact graph-matching. IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-8 :604–618, 1986.

[ES+07] Jérôme Euzenat, Pavel Shvaiko, et al. Ontology matching, volume 18. Springer, 2007.

[Eul36] Leonhard Euler. Solutio problematis ad geometriam situs pertinentis. Commentarii Academiae Scientiarum Imperialis Petropolitanae, 8 :128–140, 1736.

[FC01] Piero Fariselli and Rita Casadio. Prediction of disulfide connectivity in proteins. Bioinformatics, 17(10) :957–964, 2001.

[FCF93] Mark Fox, John Chionglo, and Fadi Fadel. A common-sense model of the enterprise. In Proceedings of the Industrial Engineering Research Conference, page 425–429, 1993.

[FKT14] Abdur Forkan, Ibrahim Khalil, and Zahir Tari. Cocamaal : A cloud-oriented context-aware middleware in ambient assisted living. Future Generation Computer Systems, 35 :114–127, 2014. Special Section : Integration of Cloud Computing and Body Sensor Networks ; Guest Editors : Giancarlo Fortino and Mukaddim Pathan.

[FM99] Roberto A. Flores-Mendez. Towards a standardization of multi-agent system framework. XRDS, 5(4) :18–24, June 1999.

[Gaj11] Bart Gajderowicz. Using decision trees for inductively driven semantic integration and ontology matching. PhD thesis, BSc, Ryerson University, Toronto, Canada, May 2011.

[Gal00] Joseph Gallian. A dynamic survey of graph labeling. Electron J Combin DS6, 19, November 2000.

[GC97] Hahn Geňa and Tardif Claude. Graph homomorphisms : structure and symmetry. In Graph Symmetry : Algebraic Methods and Applications, pages 107–166. Springer Netherlands, 1997.

[GG95] Nicola Guarino and Pierdaniele Giaretta. Ontologies and knowledge bases. Towards very large knowledge bases, pages 1–2, 1995.

[GGM+21] C. Gomez, A. Guardia, JL Mantari, Alberto Coronado, and J. Reddy. A contemporary approach to the mse paradigm powered by artificial intelligence from a review focused on polymer matrix composites. Mechanics of Advanced Materials and Structures, 29 :1–21, March 2021.

[GJ83] Michael R. Garey and David S. Johnson. Computers and intractability. a guide to the theory of np-completeness. Journal of Symbolic Logic, 48(2) :498–500, 1983.

[GLGRT06] Christophe Gaston, Pascale Le Gall, Nicolas Rapin, and Assia Touil. Symbolic execution techniques for test purpose definition. In Testing of Communicating Systems, pages 1–18, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[GOS09] Nicola Guarino, Daniel Oberle, and Steffen Staab. What Is an Ontology ?, pages 1–17. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[Gro11] W3C OWL Working Group. Owl 2 web ontology language, December 2011. Accessed : 2023-05-05.

[Gru93] Thomas R Gruber. A translation approach to portable ontology specifications. Knowledge acquisition, 5(2) :199–220, 1993.

[Gru95] Thomas R. Gruber. Toward principles for the design of ontologies used for knowledge sharing ? International journal of human-computer studies, 43(5-6) :907–928, 1995.

[Hay18] Jason Hayhurst. How Augmented Reality and Virtual Reality is Being Used to Support People Living with Dementia—Design Challenges and Future Directions, pages 295–305. Springer International Publishing, 2018.

[HBHN21] Matthew Allan Hamilton, Anthony Paul Beug, Howard John Hamilton, and Wil James Norton. Augmented reality technology for people living with dementia and their care partners. In 2021 the 5th International Conference on Virtual and Augmented Reality Simulations, ICVARS 2021, page 21–30. Association for Computing Machinery, 2021.

[Hee04] Wilbert Heeringa. Measuring Dialect Pronunciation Differences using Levenshtein Distance. Groningen dissertations in linguistics. University Library Groningen, 2004.

[HJZ+09] Yu Hua, Hong Jiang, Yifeng Zhu, Dan Feng, and Lei Tian. Smartstore : A new metadata organization paradigm with semantic-awareness for next-generation file systems. In Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, SC '09. Association for Computing Machinery, 2009.

[HMR06] Sachs Horst, Stiebitz Michael, and Wilson Robin. An historical note : Euler's königsberg letters. Journal of Graph Theory, 12 :133 – 139, October 2006.

[HN04] Pavol Hell and Jaroslav Nesetril. Graphs and Homomorphisms. Oxford Lecture Series in Mathematics and Its Applications. Oxford University Press, London, England, August 2004.

[HNGP20] Corentin Haidon, Hubert Kenfack Ngankam, Sylvain Giroux, and Hélène Pigot. Using augmented reality and ontologies to co-design assistive technologies in smart homes. In Proceedings of the 25th

International Conference on Intelligent User Interfaces Companion, IUI '20, page 126–127. Association for Computing Machinery, 2020.

[HPG20] Corentin Haidon, Hélène Pigot, and Sylvain Giroux. Joining semantic and augmented reality to design smart homes for assistance. Journal of Rehabilitation and Assistive Technologies Engineering, 7, 2020. PMID : 34422281.

[HSK09] Jong-yi Hong, Eui-ho Suh, and Sung-Jin Kim. Context-aware systems : A literature review and classification. Expert Systems with applications, 36(4) :8509–8522, 2009.

[HW74] John Edward Hopcroft and J. K. Wong. Linear time algorithm for isomorphism of planar graphs (preliminary report). In Proceedings of the Sixth Annual ACM Symposium on Theory of Computing, STOC '74, page 172–184, New York, NY, USA, 1974. Association for Computing Machinery.

[HW96] David Hooper and Kenneth Whyld, editors. The oxford companion to chess. Oxford Paperbacks, Oxford, England, October 1996.

[HWEA94] Gennari John H., Tu Samson W., Rothenfluh Thomas E., and Musen Mark A. Mapping domains to methods in support of reuse. Int. J. Hum.-Comput. Stud., 41(3) :399–424, September 1994.

[Il18] Park Kun Il. Stochastic Process, pages 135–184. Springer International Publishing, Cham, 2018.

[IW12] M. Iftikhar and Danny Weyns. A case study on formal verification of self-adaptive behaviors in a decentralized system. Electronic Proceedings in Theoretical Computer Science, 91, August 2012.

[IWZ+09] Antoine Isaac, Shenghui Wang, Claus Zinn, Henk Matthezing, Lourens van der Meij, and Stefan Schlobach. Evaluating thesaurus alignments for semantic interoperability in the library domain. IEEE Intelligent Systems, 24(2) :76–86, 2009.

[Joh21] Jeff Johnson, editor. Designing with the Mind in Mind (Third Edition), page 290. Morgan Kaufmann, third edition edition, 2021.

[Jp68] STEEN Jean pierre. Aigorithme de Recherche d'un Isomorphisme entre deux Graphes. Theses, Faculté des Sciences de l'Université de Lille, 1968.

[Kap22] Andreas Kaplan. Artificial Intelligence, Business and Civilization. Routledge, January 2022.

[KB57] Tjalling C. Koopmans and Martin Beckmann. Assignment problems and the location of economic activities. Econometrica, 25(1) :53, January 1957.

[KC03] Jeffrey Kephart and D.M. Chess. The vision of autonomic computing. Computer, 36 :41 – 50, February 2003.

[KDA98] Dey Anind K, Abowd Gregory D, and Wood Andrew. Cyberdesk : A framework for providing self-integrating context-aware services. In Proceedings of the 3rd international conference on Intelligent user interfaces, pages 47–54, 1998.

[KNPF+17] Hubert Kenfack Ngankam, Hélène Pigot, Marc Frappier, Camila Helena Souza Oliveira, and Sylvain Giroux. Formal specification for ambient assisted living scenarios. UCAmI, pages 508–519, 2017.

[KPTV09] Georgia M Kapitsaki, George N Prezerakos, Nikolaos D Tselikas, and Iakovos S Venieris. Context-aware service engineering : A survey. Journal of Systems and Software, 82(8) :1285–1297, 2009.

[KS02] K.G. Khoo and P.N. Suganthan. Evaluation of genetic operators and solution representations for shape recognition by genetic algorithms. Pattern Recognition Letters, 23(13) :1589–1597, 2002.

[LAMV22] Sylvain Lejamble, Ilham Alloui, Sébastien Monnet, and Flavien Vernier. A new software architecture for the wise object framework : Multidimensional separation of concerns. In Proceedings of the 17th International Conference on Software Technologies - ICSOFT,, pages 567–574, 2022.

[LC98] Claudia Leacock and Martin Chodorow. Combining Local Context and WordNet Similarity for Word Sense Identification, volume 49, pages 265–. MIT Press, January 1998.

[LdBN+07] Eliane Maria Loiola, Nair Maria Maia de Abreu, Paulo Oswaldo Boaventura-Netto, Peter Hahn, and Tania Querido. A survey for the quadratic assignment problem. European Journal of Operational Research, 176(2) :657–690, 2007.

[LL00] R.S.T. Lee and J.N.K. Liu. Tropical cyclone identification and tracking system using integrated neural oscillatory elastic graph matching and hybrid rbf network track mining techniques. IEEE Transactions on Neural Networks, 11(3) :680–689, 2000.

[LP02] Yang-Lyul Lee and Rae-Hong Park. A surface-based approach to 3-d object recognition using a mean field annealing neural network. Pattern Recognition, 35(2) :299–316, 2002.

[LR92] Fritz Lehmann and Ervin Y Rodin, editors. Semantic networks in artificial intelligence, volume 24 of International Series in Modern Applied Mathematics & Computer Science. Pergamon, Amsterdam, Netherlands, July 1992.

[LTLQ21] Xiulei Liu, Qiang Tong, Xuhong Liu, and Zhihui Qin. Ontology matching : State of the art, future challenges, and thinking based on utilized information. IEEE Access, PP :1–1, February 2021.

[LW93] Barbara Liskov and Jeannette M. Wing. Family values : A behavioral notion of subtyping. Technical report, MIT Laboratory for Computer Science and Carnegie Mellon University, USA, 1993.

[Mar00] Robert Mariani. Face learning using a sequence of images. International Journal of Pattern Recognition and Artificial Intelligence, 14(05) :631–648, August 2000.

[Mar02] Robert C Martin. Agile software development, principles, patterns, and practices. Alan Apt series. Pearson, Upper Saddle River, NJ, October 2002.

[Mar08] Robert C Martin. Clean code. Prentice Hall, Philadelphia, PA, August 2008.

[Mar17] Robert C Martin. Clean architecture. Prentice Hall, September 2017.

[Mas06] Maurice Mashaal. Bourbaki. Bourbaki : A secret society of mathematicians. American Mathematical Society, June 2006.

[McC79] Pamela McCorduck. Machines Who Think : A Personal Inquiry into the History and Prospects of Artificial Intelligence. AK Peters Ltd, 1979.

[Mey88] Bertrand Meyer. Object-oriented Software Construction. Prentice Hall International series in computer science. Prentice-Hall, London, England, April 1988.

[MH69] John McCarthy and Patrick J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, Machine Intelligence 4, pages 463–502. Edinburgh University Press, 1969. reprinted in McC90.

[MH00] Richard Myers and Edwin R. Hancock. Genetic algorithms for ambiguous labelling problems. Pattern Recognition, 33(4) :685–704, 2000.

[Mil95a] George A. Miller. Wordnet : A lexical database for english. Commun. ACM, 38(11) :39–41, November 1995.

[Mil95b] George A. Miller. Wordnet : A lexical database for english. Commun. ACM, 38(11) :39–41, November 1995.

[MJB15] Mary Niles Maack Marcia J. Bates. Encyclopedia of Library and Information Sciences, Third Edition, page 6106. CRC Press, 2015.

[MM04] Frank Manola and Eric Miller. Rdf primer, January 2004. Accessed : 2023-05-05.

[MSV12] Patrice Moreaux, Favien Sartor, and Flavien Vernier. An effective approach for home services management. In 2012 20th Euromicro International Conference on Parallel, Distributed and Network-based Processing, pages 47–51, 2012.

[MVM20] Etienne Mauffret, Flavien Vernier, and Sébastien Monnet. How to use the past to face the future ? Research report, LISTIC, 2020.

[NCLM06] Natalya F. Noy, Abhita Chugh, William Liu, and Mark A. Musen. A framework for ontology evolution in collaborative environments. In The Semantic Web - ISWC 2006, pages 544–558. Springer Berlin Heidelberg, 2006.

[NER86] Biggs N., Lloyd E.K., and Wilson R.J. Graph Theory, 1736-1936. Clarendon Press, 1986.

[Nga19] Hubert Kenfack Ngankam. Modèle Sémantique d'Intelligence Ambiante pour le Développement Do-It-Yourself d'Habitats Intelligents. Theses, Faculté des Sciences, Université de Sherbrooke, January 2019.

[NPG22] Hubert Kenfack Ngankam, Hélène Pigot, and Sylvain Giroux. Ontodomus : A semantic model for ambient assisted living system based on smart homes. Electronics, 11(7) :1143, April 2022.

[NPL+20] Hubert Kenfack Ngankam, Hélène Pigot, Dominique Lorrain, Isabelle Viens, and Sylvain Giroux. Context awareness architecture for ambient-assisted living applications : Case study of nighttime wandering. Journal of Rehabilitation and Assistive Technologies Engineering, 7, 2020.

[NRB06] Michel Neuhaus, Kaspar Riesen, and Horst Bunke. Fast suboptimal algorithms for the computation of graph edit distance. In Structural, Syntactic, and Statistical Pattern Recognition, pages 163–172, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[Obe14] Daniel Oberle. How ontologies benefit enterprise applications. Semantic Web, 5 :473–491, 2014.

[Ora] Oracle. Dynamic Proxy Classes — docs.oracle.com. https://docs.oracle.com/javase/8/docs/technotes/guides/reflection/proxy.html. [Accessed 07-09-2023].

[OT02] Harold Ossher and Peri Tarr. Multi-Dimensional Separation of Concerns and the Hyperspace Approach, pages 293–323. Springer US, 2002.

[PB08] Davy Preuveneers and Yolande Berbers. Encoding semantic awareness in resource-constrained devices. IEEE Intelligent Systems, 23(2) :26–33, 2008.

[PDG12] Valentina Presutti, Francesco Draicchio, and Aldo Gangemi. Knowledge extraction based on discourse representation theory and linguistic frames. In Knowledge Engineering and Knowledge Management, pages 114–129. Springer Berlin Heidelberg, 2012.

[Pia38] H. T. H. Piaggio. Introduction to mathematical probability. by j. v. uspensky. pp. ix, 411. 30s. 1937. (mcgraw-hill). The Mathematical Gazette, 22(249) :14–18, 1938.

[PML05] O'Brien Kevin P., Remm Maido, and Sonnhammer Erik L. L. Inparanoid : a comprehensive database of eukaryotic orthologs. Nucleic Acids Research, 33(suppl_1) :D476–D480, January 2005.

[Pow91] David M. W. Powers. Goals, issues and directions in machine learning of natural language and ontology (bibliography). SIGART Bull., 2 :101–114, 1991.

[RB90] George A. Miller Richard Beckwith. Implementing a Lexical Network*. International Journal of Lexicography, 3(4) :302–312, December 1990.

[Res99] Philip Resnik. Semantic similarity in a taxonomy : An information-based measure and its application to problems of ambiguity in natural language. J. Artif. Int. Res., 11(1) :95–130, July 1999.

[RM13] Parisa Rashidi and Alex Mihailidis. A survey on ambient-assisted living tools for older adults. IEEE Journal of Biomedical and Health Informatics, 17(3) :579–590, 2013.

[RMBB89] R. Rada, H. Mili, E. Bicknell, and M. Blettner. Development and application of a metric on semantic nets. IEEE Transactions on Systems, Man, and Cybernetics, 19(1) :17–30, 1989.

[RMJ05] Vlad Rusu, Hervé Marchand, and Thierry Jéron. Automatic verification and conformance testing for validating safety properties of reactive systems. In Formal Methods 2005 (FM05), volume 3582 of Lecture Notes in Computer Science, pages 189–204, Newcastle, United Kingdom, July 2005. Springer-Verlag.

[RMPO+02] Denis Rivière, Jean-François Mangin, Dimitri Papadopoulos-Orfanos, Jean-Marc Martinez, Vincent Frouin, and Jean Régis. Automatic recognition of cortical sulci of the human brain using a congregation of neural networks. Medical Image Analysis, 6(2) :77–92, 2002.

[RMT+04] Vlad Rusu, Hervé Marchand, Valéry Tschaen, Thierry Jéron, and Bertrand Jeannet. From safety verification to safety testing. In Testing of Communicating Systems, pages 160–176, March 2004.

[RNG+17] Robert Radziszewski, Hubert Ngankam, Vincent Grégoire, Dominique Lorrain, Hélène Pigot, and Sylvain Giroux. Designing calm and non-intrusive ambient assisted living system for monitoring nighttime wanderings. International Journal of Pervasive Computing and Communications, 13, June 2017.

[RNP+16] Robert Radziszewski, Hubert Ngankam, Helene Pigot, Vincent Grégoire, Dominique Lorrain, and Sylvain Giroux. An ambient assisted

living nighttime wandering system for elderly. In Proceedings of the 18th International Conference on Information Integration and Web-Based Applications and Services, iiWAS '16, page 368–374. Association for Computing Machinery, 2016.

[Roc03] Christophe Roche. Ontology : A survey. IFAC Proceedings Volumes, 36, September 2003.

[RS21] Miklos Racz and Anirudh Sridhar. Correlated stochastic block models : Exact graph matching with applications to recovering communities. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, Advances in Neural Information Processing Systems, volume 34, pages 22259–22273. Curran Associates, Inc., 2021.

[RSM94] Raymond Richardson, Alan F. Smeaton, and J Murphy. Using wordnet as a knowledge base for measuring semantic similarity between words. In In Proceedings of AICS Conference, 1994.

[Rus06] G. Russello. Separation and adaptation of concerns in a shared data space. PhD thesis, Mathematics and Computer Science, 2006.

[San13] Anders Sandberg. An overview of models of technological singularity. The Transhumanist Reader : Classical and Contemporary Essays on the Science, Technology, and Philosophy of the Human Future, pages 376–394, 2013.

[Sar12a] Fabien Sartor. Modélisation de l'interopérabilité d'objets communicants et de leur coopération : application à la domotique. Theses, Université de Grenoble, July 2012.

[Sar12b] Fabien Sartor. Modélisation de l'interopérabilité d'objets communicants et de leur coopération : application à la domotique. (modeling of interoperability of communicating object their cooperation : implementation to home automation). In Modeling of interoperability of communicating object their cooperation : implementation to home automation, 2012.

[SAW94] Bill Schilit, Norman Adams, and Roy Want. Context-aware computing applications. In 1994 first workshop on mobile computing systems and applications, pages 85–90. IEEE, 1994.

[SBF98] Rudi Studer, V Richard Benjamins, and Dieter Fensel. Knowledge engineering : Principles and methods. Data & knowledge engineering, 25(1-2) :161–197, 1998.

[SBPZ13] Alexandru Sorici, Olivier Boissier, Gauthier Picard, and Antoine Zimmermann. Applying semantic web technologies to context modeling in ambient intelligence. In Evolving Ambient Intelligence : AmI 2013 Workshops, Dublin, Ireland, December 3-5, 2013. Revised Selected Papers 4, pages 217–229. Springer, 2013.

[SBV01] Kim Shearer, Horst Bunke, and Svetha Venkatesh. Video indexing and similarity retrieval by largest common subgraph detection using decision trees. Pattern Recognition, 34(5) :1075–1091, 2001.

[Sch05] Albrecht Schmidt. Interacting with ambient intelligence. Ambient Intelligence, 2005.

[SD03] Timothy Sohn and Anind Dey. icap : an informal tool for interactive prototyping of context-aware applications. In CHI'03 extended abstracts on Human factors in computing systems, pages 974–975, 2003.

[SE13] Pavel Shvaiko and Jérôme Euzenat. Ontology matching : State of the art and future challenges. IEEE Transactions on Knowledge and Data Engineering, 25(1) :158–176, 2013.

[SEH+03] Gerd Stumme, Marc Ehrig, Siegfried Handschuh, Andreas Hotho, Alexander Maedche, Boris Motik, Daniel Oberle, Christoph Schmitz, Steffen Staab, Ljiljana Stojanović, Nenad Stojanović, Rudi Studer, York Sure, Raphael Volz, and Valentin Zacharias. The karlsruhe view on ontologies. Technical report, Technical report, University of Karlsruhe, Institute AIFB, 2003.

[SG76] Sartaj Sahni and Teofilo Gonzalez. P-complete approximation problems. Journal of the ACM, 23(3) :555–565, July 1976.

[SHW17] Thiago Rocha Silva, Jean-Luc Hak, and Marco Winckler. A behavior-based ontology for supporting automated assessment of interactive systems. In 2017 IEEE 11th International Conference on Semantic Computing (ICSC), pages 250–257, 2017.

[SK16] Taranjeet Singh and Avadhesh Kumar. Survey on characteristics of autonomous system. International Journal of Computer Science and Information Technology, 8 :121–128, April 2016.

[SM14] Nagender K. Suryadevara and Subhas C. Mukhopadhyay. Determining wellness through an ambient assisted living environment. IEEE Intelligent Systems, 29(3) :30–37, 2014.

[SPSL02] Alexander Smirnov, Mikhail Pashkin, Nikolay Shilov, and Tatiana Levashova. Knowledge source network configuration approach to knowledge logistics. In Proceedings of the 2002 IEEE International Conference on Artificial Intelligence Systems (ICAIS'02), volume 32, pages 95 – 100, February 2002.

[SRT06] Bandyopadhyay Sourav, Sharan Roded, and Ideker Trey. Systematic identification of functional orthologs based on protein network comparison. Genome research, 16 :428–35, April 2006.

[SSSK15] Arto Salomaa, Ian Naismith Sneddon, Harold M. Stark, and Jean-Pierre Kahane. Theory of Automata : International Series of Monographs in Pure and Applied Mathematics. International series in pure and applied mathematics. - page.1-2. London : Elsevier Science, 2015, 1969-2015.

[ST05] Mazeiar Salehie and Ladan Tahvildari. Autonomic computing : Emerging trends and open problems. In Proceedings of the 2005 Workshop on Design and Evolution of Autonomic Application Software, DEAS '05, page 1–7. Association for Computing Machinery, 2005.

[STE09] Toby Segaran, Jamie Taylor, and Colin Evans. Programming the semantic web. O'Reilly Media, Sebastopol, CA, July 2009.

[SXB08] Rohit Singh, Jinbo Xu, and Bonnie Berger. Global alignment of multiple protein interaction networks with application to functional orthology detection. Proceedings of the National Academy of Sciences, 105(35) :12763–12768, 2008.

[SZP+20] Yulia Svetashova, Baifan Zhou, Tim Pychynski, Stefan Schmidt, York Sure-Vetter, Ralf Mikut, and Evgeny Kharlamov. Ontology-enhanced machine learning : A bosch use case of welding quality monitoring. In The Semantic Web – ISWC 2020 : 19th International Semantic Web Conference, Athens, Greece, November 2–6, 2020, Proceedings, Part II, page 531–550, Berlin, Heidelberg, 2020. Springer-Verlag.

[TF79] Wen-Hsiang Tsai and King-Sun Fu. Error-correcting isomorphisms of attributed relational graphs for pattern analysis. IEEE

Transactions on Systems, Man, and Cybernetics, 9(12) :757–768, 1979.

[Thé13] Pierrick Thébault. La conception à l'ère de l'Internet des Objets : modèles et principes pour le design de produits aux fonctions augmentées par des applications. PhD thesis, École Nationale Supérieure d'Arts et Métiers, 05 2013.

[TIP10] Partha Pratim Talukdar, Zachary G. Ives, and Fernando Pereira. Automatically incorporating new sources in keyword search-based data integration. In Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, SIGMOD '10, page 387–398. Association for Computing Machinery, 2010.

[TOHS99] P. Tarr, H. Ossher, W. Harrison, and S.M. Sutton. N degrees of separation : multi-dimensional separation of concerns. In Proceedings of the 1999 International Conference on Software Engineering (IEEE Cat. No.99CB37002), pages 107–119, 1999.

[Tug04] Alexandru Tugui. Calm technologies in a multimedia world. Ubiquity, 2004, March 2004.

[Tut01] W T Tutte. Cambridge mathematical library : Graph theory. Encyclopedia of mathematics and its applications. Section, Combinatorics. Cambridge University Press, Cambridge, England, January 2001.

[Tve77] Amos Tversky. Features of similarity. Psychological review, 84(4) :327, 1977.

[Ull76] Julian R. Ullmann. An algorithm for subgraph isomorphism. J. ACM, 23(1) :31–42, January 1976.

[VER21] Flavien VERNIER. De l'exécution distribuée des chaînes de traitement d'images à l'acquisition distribuée de connaissance par les traitements - Chapter : Architecture de traitement de l'habitude. Habilitation à diriger des recherches en sciences et technologie de l'information et de la communication, École Doctorale SISEO, LISTIC (EA 3703), Polytech Annecy-Chambéry, 2021.

[vH95] Gertjan van Heijst. The Role of Ontologies in Knowledge Engineering. Theses, University of Amsterdam, May 1995.

[VHSW97] Gertjan Van Heijst, A Th Schreiber, and Bob J Wielinga. Using explicit ontologies in kbs development. International journal of human-computer studies, 46(2-3) :183–292, 1997.

[WB96] Mark Weiser and John Seely Brown. Designing calm technology. Powergrid Journal, 1, 1996.

[Wei91] Mark Weiser. The computer for the 21st century. Scientific American, 265(3) :66–75, 1991.

[WH96] Richard C. Wilson and Edwin R. Hancock. A bayesian compatibility model for graph matching. Pattern Recognition Letters, 17(3) :263–276, 1996.

[Wil10] E.A.B.S.G. Williamson. Lists, Decisions and Graphs. S. Gill Williamson, 2010.

[WL15] Jong-Bum Woo and Youn-Kyung Lim. User experience in do-it-yourself-style smart homes. In Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing, page 779–790, 2015.

[WMA10] Danny Weyns, Sam Malek, and Jesper Andersson. Forms : A formal reference model for self-adaptation. In Proceeding of the 7th International Conference on Autonomic Computing, ICAC '10 and Co-located Workshops, pages 205–214, June 2010.

[WP94] Zhibiao Wu and Martha Palmer. Verbs semantics and lexical selection. In Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics, ACL '94, page 133–138, USA, 1994. Association for Computational Linguistics.

[Wus07] Hans Wussing. The genesis of the abstract group concept. Dover Books on Mathematics. Dover Publications, April 2007.

[XBT08] Chunyan Xie, Richard Bagozzi, and Sigurd Troye. Trying to prosume : Toward a theory of consumers as co-creators of value. Journal of the Academy of Marketing Science, 36 :109–122, 2008.

[XW15] Xingsi Xue and Yuping Wang. Optimizing ontology alignments through a memetic algorithm using both matchfmeasure and unanimous improvement ratio. Artificial Intelligence, 223 :65–81, 2015.

[XW16] Xingsi Xue and Yuping Wang. Using memetic algorithm for instance coreference resolution. In 2016 IEEE 32nd International Conference on Data Engineering (ICDE), pages 1450–1451, 2016.

[XY18] Xingsi Xue and Xin Yao. Interactive ontology matching based on partial reference alignment. Applied Soft Computing, 72 :355–370, 2018.

[YZL13] Bisheng Yang, Yunfei Zhang, and Xuechen Luan. A probabilistic relaxation approach for matching road networks. International Journal of Geographical Information Science, 27(2) :319–338, 2013.

[Zas10] Mikhail Zaslavskiy. Graph matching and its application in computer vision and bioinformatics. Theses, École Nationale Supérieure des Mines de Paris, January 2010.

[ZBC+14] Andrea Zanella, Nicola Bui, Angelo Castellani, Lorenzo Vangelista, and Michele Zorzi. Internet of things for smart cities. IEEE Internet of Things Journal, 1(1) :22–32, 2014.

[Zhu] Ganggao Zhu. Home - Sematch Documentation — gsi-upm.github.io. https://gsi-upm.github.io/sematch/. [Accessed 06-10-2023].

[ZI15] Ganggao Zhu and Carlos Angel Iglesias. Sematch : Semantic entity search from knowledge graph. In SumPre-HSWI@ESWC, 2015.

[ZI17] Ganggao Zhu and Carlos A. Iglesias. Computing semantic similarity of concepts in knowledge graphs. IEEE Transactions on Knowledge and Data Engineering, 29(1) :72–85, 2017.

[ZXY+21] Hai Zhu, Xingsi Xue, Chaofan Yang, Chao Jiang, Pei-Wei Tsai, and Guojun Mao. Optimizing ontology alignment through linkage learning on entity correspondences. Complexity, February 2021.

Résumé

Mes travaux de thèse portent sur la problématique de la communication entre l'humain et les systèmes intelligents. Dans le cadre de cette problématique, j'ai étudié les deux questions suivantes. Comment un système intelligent peut-il communiquer efficacement avec l'humain ? Comment l'humain peut-il communiquer efficacement avec un système intelligent ? Concernant la première question, je me suis appuyé sur le concept d'objets sages introduit au sein du laboratoire LISTIC. Les objets sages ont la capacité d'apprendre par eux-mêmes leur fonctionnement et le comportement de leurs utilisateurs. Cependant, ils sont incapables de communiquer avec ces derniers en utilisant le langage humain, c'est-à-dire la sémantique humaine. Concernant la seconde question, j'ai mis en place une collaboration avec le laboratoire DOMUS de l'université de Sherbrooke (Canada), qui développe une solution d'aide au maintien des personnes à domicile. Cette solution nécessite une maison connectée et la définition de scénarios d'activités quotidiennes pour faciliter la vie de ces personnes. Mon travail a consisté à définir les actions que la maison doit réaliser à partir d'un scénario exprimé vocalement. J'ai ainsi proposé dans ma thèse : l'augmentation sémantique d'un objet sage pour qu'il puisse communiquer avec un humain, et inversement, une technique d'extraction d'actions définies par commande vocale permettant à un humain de communiquer avec une machine.

Summary

My thesis work focuses on the issue of communication between humans and machines. I have explored this problem from both directions, through the following two questions : How can a machine effectively communicate with humans ? How can humans effectively communicate with machines ? Regarding the first question, I have leveraged the concept of 'wise objects' introduced within the LISTIC laboratory. These objects have the ability to autonomously learn their functionality and the behaviour of their users. However, they lack the ability to communicate with users using human semantics. As for the second question, I established a collaboration with the DOMUS laboratory, which is developing a solution to support aging in place. This solution involves a connected home and the definition of scenario specifications to enhance the lives of the residents. My work has entailed defining the actions that the home should execute based on orally dictated scenarios. I have thus addressed the challenges of semantic augmentation to enable clear communication from a machine to a human and the extraction of high-level semantic actions to facilitate human communication with a machine.