Contribution à la mise en œuvre d'une architecture ambiante d'interaction homme-robot-environnement. Dans le cadre de la robotique d'aide à la personne dépendante.

Nadia Touileb Djaid

université PARIS-SACLAY

NNT: 2017SACLV037

Thèse de doctorat

UNIVERSITÉ DE VERSAILLES ST-QUENTIN-EN-YVELINES

université PARIS-SACLAY

USTHB

Contribution à la mise en œuvre d'une architecture ambiante d'interaction homme-robot-environnement. Dans le cadre de la robotique d'aide à la personne dépendante.

Thèse de doctorat de l'Université des Sciences et de la Technologie Houari Boumediene d'Alger et de l'Université Paris-Saclay
préparée à l'Université Versailles Saint-Quentin en Yvelines

École doctorale n°580 Sciences et Technologies de l'Information et de la Communication (STIC)
Spécialité de doctorat : Robotique

Thèse présentée et soutenue à Alger, le 16 Décembre 2017, par

Mme Nadia Touileb Djaid

Composition du Jury :

M. R. Baba-Ali Professeur, USTHB	Président
Mme K. Akli Maître de Conférences / A, USTHB	Rapporteur
Mme N. Levy Professeure, CNAM Laboratoire Cédric	Rapporteur
M. É. Monacelli Professeur, UVSQ	Examinateur
M. M. Guiatni Maître de Conférences / A, EMP d'Alger	Examinateur
M. A. Ramdane-Cherif, Professeur, UVSQ	Directeur de thèse
Mme N. Saadia Professeure, USTHB	Co-Directeur de thèse

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Contribution à la mise en œuvre d'une architecture ambiante d'interaction homme-robot-environnement. Dans le cadre de la robotique d'aide à la personne dépendante.

Mots clés : Multimodalité, Fusion, Fission, Ontologie, Règles SWRL

Résumé : Le sujet de cette thèse de doctorat consiste à proposer une architecture ambiante d'interaction homme-robot-environnement. Dans le cadre de la robotique d'aide à la personne dépendante. Cette architecture va permettre aux robots « Ubiquitous Networked Robots » de prendre en compte le contexte évolutif pour fournir continuellement du service à l'utilisateur. L'architecture proposée utilise le concept d'Ontologie du domaine pour la déscription de l'environnement. Nous avons choisi d'utiliser l'outil open source PROTEGE qui va nous permettre de définir l'ontologie ainsi que les moteurs de fusion et de fission. Les entrées multimodales seront fusionnées puis subdivisées en tâches élémentaires et envoyées comme commandes au fauteuil roulant muni d'un bras manipulateur. Cette architecture sera validée par des spécifications et des simulations via des réseaux de Pétri temporels et stochastiques.

Title: Contribution to the implementation of an ambient architecture for the human-robot-environment interaction, as part of the robotics help for dependent people.

Keywords : Multimodality, Fusion, Fission, Ontology, SWRL rules

Abstract: The subject of this thesis is to provide an ambient architecture for the human-robot-environment interaction, as part of the dependent person robotics help. This architecture will enable the robot to take into account the changing context and continually provide a service to the user. The architecture uses the concept of ontology for the description of the environment. We have chosen to use the open source PROTEGE because it allows the definition of the ontology and the fusion and fission engines. Indeed, multimodal inputs will be merged and subdivided into elementary tasks and sent to control the wheelchair with the manipulated arm. This architecture will be validated by specifications and simulations via temporal and stochastic Petri nets.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France

Remerciements

Je tiens à adresser mes sincères remerciements à mes deux directeurs de thèse Mme Nadia Saadia et Mr Amar Ramdane-Cherif pour tout leur soutien et le temps qu'ils ont su m'accorder pour m'aider à aboutir à l'objectif de notre travail. Je ne saurais dire combien leurs conseils m'ont été précieux.

Aussi, je voudrais remercier tous les membres du jury qui ont accepté d'évaluer mon travail.

Un grand remerciement à toute l'équipe Robotique 2 du laboratoire LRPE de l'USTHB et l'équipe du laboratoire LISV de l'UVSQ pour leur aide et soutien.

Un spécial remerciement à ma sœur Katia et à ma Mère que j'ai souvent embêtées pour mes dossiers pendant mes années de Doctorat. Un grand merci à mon Père et ma sœur Samia qui m'ont souvent donné des précieux conseils. Enfin, un énorme merci à mon mari et mes filles pour avoir supporté mes heures de travail et tout le stress qui vient avec.

Enfin, je voudrais dire un grand merci à l'ensemble de ma famille qui m'a toujours encouragée tout le long de ma vie. Qui ont cru en moi et qui m'ont toujours soutenue et offert tous les moyens pour réaliser mes projets. Je tiens à leur offrir ma profonde reconnaissance pour tout leur support et conseil qui m'ont permis de finaliser ce travail.

Dédicaces

Je dédie ce travail à mes deux adorables filles Imene et Yasmine et mon neveu Ilyes qui sont ma fierté. A mon mari Daoud qui m'a toujours encouragé et soutenu.

Je le dédie aussi à mes chers parents qui se sont dévoués toute leur vie pour notre bien-nêtre, à mes merveilleuses sœurs Katia et Samia, mon beau-frère Mehdi et à toute ma famille et belle-famille.

Table des matières

Table des figures

Liste des tableaux

Introduction Générale

Ce mémoire présente nos travaux de recherche qui portent sur l'élaboration d'une architecture ambiante d'interaction homme-robot-environnement, dans le cadre de la robotique d'aide à la personne dépendante. Notre travail consiste en l'élaboration d'une architecture de commande pour un fauteuil roulant muni d'un bras manipulateur utilisé par des personnes à mobilité réduite.

Avec l'émergence croissante de l'intelligence ambiante ([1], [2], [3]), de l'informatique ubiquitaire ([4] [5],[6]), des réseaux de capteurs et des technologies de réseau sans fil ([7], [8]), la robotique ubiquitaire et les robots en réseau (ubiquitous networked robotics) ([9], [10], [11],[12],) deviennent un domaine de recherche très actif dans les systèmes autonomes intelligents. En effet, ce domaine vise des applications novatrices dans lesquelles les systèmes robotiques seront intégrés dans des environnements ubiquitaires comme des entités autonomes, logicielles ou physiques.

Ces robots sont capables d'interagir de manière autonome avec l'environnement ambiant et de fournir des services à valeur ajoutée comme l'aide aux personnes dans des maisons intelligentes (smart homes), des bureaux, des bâtiments et des espaces publics. Les robots présentés comme étant des entités cognitives seront donc en mesure de coordonner leurs activités avec les autres entités physiques ou logiques, de se déplacer, d'explorer l'environnement ambiant, de décider et d'agir pour répondre aux situations qu'ils peuvent rencontrer. Ces opérations cognitives feront parties de ces réseaux intelligents capables de fournir, individuellement ou collectivement, de nouvelles fonctionnalités et divers services d'assistance, n'importe où et n'importe quand. Ce paradigme permet de construire un pont entre la robotique et l'informatique ubiquitaire ([4], [13]), c'est-à-dire

la création d'architectures flexibles et extensibles, basées sur Internet, capables de soutenir toutes sortes de services robotiques intelligents et autonomes et d'interagir avec des objets virtuels ou réels.

Dans ce contexte, nos travaux seront orientés vers la conception d'une architecture ambiante d'interaction homme-robot-environnement. Cette architecture va permettre au robot de fournir des services à l'utilisateur dans un environnement dynamique en prenant en compte un contexte évolutif [14]. Le paradigme de Ubiquitous Networked Robotics (UNR) soulève un certain nombre de défis de recherche importants tels que :

1. La Communication en temps réel en utilisant des technologies hétérogènes, des réseaux sans fil et filaires, tout en assurant la qualité et la continuité des services de communication ;
2. L'interopérabilité entre les différentes technologies matérielles et logicielles utilisées pour garantir une interaction continue entre l'UNR (Ubiquitous Networked Robotics) et les dispositifs et systèmes environnants ;
3. Les nouveaux paradigmes d'interaction homme-robot-environnement, comprenant les mécanismes de communication implicite, de perception artificielle et de raisonnement ;
4. L'adaptabilité omniprésente à l'environnement et le management de l'évolutivité. Les systèmes UNR nécessitent plus de flexibilité, de mobilité, de sécurité, de fiabilité et de robustesse grâce à des mécanismes de middleware tels que la découverte automatique des entités et des services, la composition et l'orchestration de services, l'auto-adaptation et la sensibilisation du contexte avec la gestion de l'incertitude.

Les êtres humains interagissent en permanence avec leur environnement en utilisant leurs capacités de communications naturelles telles que la parole ou le geste. Même lors de la présence d'un handicap, le cerveau humain est capable de s'adapter à l'état de santé de la personne et de développer un moyen de communication avec son environnement. Les chercheurs tentent d'imiter les fonctions cérébrales des êtres humains et de créer

des machines intelligentes capables de communiquer avec les personnes en utilisant des moyens naturels.

Nous visons dans notre travail présenté dans ce mémoire, à construire un système de communication entre un fauteuil roulant muni d'un bras manipulateur et son utilisateur, tout en prenant en considération les informations contextuelles sémantiques. En effet, les personnes sont capables de comprendre et de communiquer entre eux en stockant les informations dans leur cerveau et en les réutilisant. Ainsi, pour la création d'une machine intelligente , nous devons trouver un moyen de stockage d'informations ainsi qu'un moyen qui va nous permettre de les réutiliser. En outre, les informations doivent être présentées de telle sorte qu'elles soient comprises par la machine et l'utilisateur en même temps. La définition de l'environnement et de la relation sémantique entre ses entités doivent être claires et accessibles à tout moment par la machine.

Le concept le plus adéquat pour résoudre ce problème est celui d'Ontologie ([15], [16], [17],[18]). En effet, le concept d'ontologie permet une description complète d'un environnement et la définition des relations entre les entités qui le décrivent. De ce fait, notre approche sera la modélisation d'une architecture qui a comme base de connaissance une ontologie pour les moteurs de fusion et de fission. L'ontologie utilisée décrit l'environnement changeant des systèmes multimodaux en définissant les scénarios et tous les éléments qui composent l'environnement. L'utilisation de l'ontologie assure « l'ouverture » du système grâce à la prise en compte d'un nombre important de modalités d'entrée et de sortie, la «régularité» grâce à la possibilité de définir les éléments et les scenarios les plus adéquats à la description de la composition de l'environnement, et enfin la «flexibilité» qui permet l'ajout ou la suppression d'entités à tout moment en fonction du profil de l'utilisateur ou du domaine d'utilisation. Bien que notre système soit appliqué à un fauteuil roulant muni d'un bras manipulateur dans le présent travail, en ajustant les éléments de l'ontologie, nous pouvons l'appliquer à tout autre système d'interaction homme-machine-environnement. Cette caractéristique permet à notre architecture du système d'être adaptable aux circonstances de l'utilisateur et ouverte à une multitude de domaines et aspects.

De plus, une machine qui veut imiter les compétences de communication entre personnes doit être en mesure de faire face aux différentes modalités d'entrées et de sorties. De ce fait, les systèmes multimodaux ([19], [20], [21], [22], [23]) sont les plus adéquats pour être utilisés car ils permettent la combinaison de modalités d'entrée et/ou de sortie dynamiquement. En outre, le cerveau humain est capable de comprendre et d'agir selon les informations récupérées de l'environnement entourant la personne et de décider en utilisant les informations stockées à partir d'expériences antérieures. Par exemple, si une personne entend la sonnerie du téléphone, elle saura naturellement qu'il faut se diriger vers le téléphone et y répondre. Ce qui semble être une tâche facile pour les êtres humains est en fait une action très compliquée pour un robot. En effet, le robot doit être capable de comprendre que le bruit est une sonnerie de téléphone et qu'il doit se déplacer et répondre. Pour cela, une combinaison de plusieurs modalités doit être faite en utilisant les moteurs de fusion et de fission. Le moteur de fusion est la partie du système qui est capable de comprendre une situation et peut être en mesure de décider, tandis que le moteur de fission est capable de prendre en charge les actions et d'offrir différents services en subdivisant les informations fusionnées et en envoyant des sous-tâches élémentaires aux modalités de sortie. Nous avons choisi d'utiliser le concept de règles lors de la construction des moteurs de fusion et de fission car ce dernier assure un bon alignement temporel entre les différentes modalités d'entrée et de sortie et est souvent utilisé pour mieux estimer l'état d'un objet en mouvement.

Nous proposons dans ce mémoire une nouvelle solution dans le cadre de l'interaction homme-robot-environnement qui combine un moteur de fusion et de fission dans le but d'obtenir un system complet. Le système proposé pourra prendre en charge la commande d'un utilisateur du début par la réception de la commande, jusqu'à la fin par la proposition de services. En effet, la combinaison des deux éléments fondamentaux d'un système d'interaction multimodale, que sont le moteur de fusion et de fission, assure une prise en compte totale de la demande de l'utilisateur. De plus, notre nouvelle approche utilise la représentation sémantique de l'environnement pour la fusion et la fission. Ceci assure une représentation commune des données échangées entre l'utilisateur et la ma-

chine, ce qui signifie que les données ont le même format et la même signification pour les deux (la machine et l'utilisateur).

Notre objectif est de proposer un travail innovant pour le contrôle d'un fauteuil roulant muni d'un bras manipulateur par la construction d'un système d'interaction homme-machine qui combine la fusion et le processus de fission . Nous proposons une nouvelle architecture qui facilite le travail des deux moteurs par l'utilisation du concept d'ontologie comme base de connaissance. Notre système sera en mesure de comprendre la demande, fusionner les modalités d'entrée et répondre à la demande en envoyant des tâches uni-modales aux modalités de sortie grâce aux moteurs de fission. Nous avons choisi de construire les moteurs de fusion et de fission multimodales en utilisant le concept d'ontologie comme base de connaissance tout en appliquant le modèle WWHT (What-Which-How-Then) [24]. Ce choix est guidé par le fait que l'utilisation d'une ontologie permet une description sémantique complète de l'environnement de l'utilisateur et prend en considération son contexte. En outre, ce concept offre un accès facile à l'information et permet sa réutilisation à n'importe quel moment. Aussi, le concept d'ontologie permet l'introduction de règles de fusion et de fission selon des modèles et des motifs prédéfinis dans l'ontologie. Nous avons choisi de concevoir des moteurs de fusion et de fission à base de règles car cette méthode assure un bon alignement temporel entre les différentes modalités et elle est souvent utilisée pour mieux évaluer l'état d'un objet en mouvement.

Enfin, nous avons choisi de valider notre approche en utilisant les réseaux de Pétri temporels [25] en utilisant l'outil CPN Tools (Université d'Aarhus 2012) ([26],[27], [28], [29]). Cet outil open source permet la modélisation et la validation des systèmes distribués et la visualisation des simulations. CPN Tools est un logiciel qui modélise les réseaux de Pétri ; ils sont scientifiquement reconnu comme une bonne méthodologie stochastique de génie logiciel pour vérifier et évaluer aléatoirement les réactions d'agents dans tous les types de situations (bonnes ou mauvaises). Les réseaux de Pétri sont des outils appropriés pour simuler et valider une conception logicielle telle qu'une communication entre agents. Ils permettent aussi la vérification et la validation de composants logiciels sémantiques tels que des services Web ou des agents de communication sur le

Web. Ils peuvent également être utilisés pour modéliser des comportements autonomes de systèmes multi-agents hétérogènes, ainsi que pour vérifier que ces systèmes peuvent opérer selon des exigences de mission prédéfinies et prendre en compte les contraintes.

Dans le premier chapitre, nous allons présenter notre objectif de travail ainsi que la problématique. Puis, nous allons donner un état de l'art non exhaustif sur les travaux existant sur la fusion et la fission des données, les architectures ambiantes ainsi que des exemples d'application. Nous allons clôturer ce chapitre par la solution proposée pour répondre à notre objectif. Nous présenterons dans le deuxième chapitre les étapes de la méthodologie de travail et les différentes parties de la modélisation de l'architecture. Dans le troisième chapitre, l'architecture du système sera proposée suivi par un exemple d'application, le scénario, la création de l'ontologie et la présentation détaillée de tous ses composants. Nous présenterons dans le quatrième chapitre les étapes qui aboutissent à l'offre de service grâce à la sélection de modalités, la fusion puis la fission des données. Une étude de cas sera présentée pour expliquer le fonctionnement du système. La chapitre cinq fera l'objet d'une validation pas réseaux de Pétri temporels et stochastiques en utilisant l'outil CPNTools. Finalement, une conclusion générale sera présentée qui résume nos principales contributions.

Chapitre 1

Problématique et Etat de l'art

1.1 Introduction

Le travail présenté dans ce mémoire vise à l'amélioration de l'interaction homme-machine-environnement dans le cadre de la robotique mobile d'aide à la personne dépendante. Nous visons dans notre travail à faciliter cette communication en la rendant similaire à la communication interpersonnelle. En effet, l'utilisateur du système proposé pourra dialoguer avec la machine en utilisant ses propres moyens naturels de communication tels que la parole ou le geste. D'abord, nous allons présenter dans ce qui suit les différents problèmes rencontrés lors de la conception de l'architecture du système ainsi que les solutions proposées. Puis, nous allons présenter un état de l'art non exhaustif sur les travaux trouvés dans la littérature et réalisés dans ce domaine.

1.2 Objectif

Notre objectif est de concevoir l'architecture d'un système capable de prendre en compte des informations issues de capteurs ou de modalités dans un environnement dynamique et de les fusionner puis de les fissionner pour fournir des services aux utilisateurs. Les principaux utilisateurs visés par ce type d'architecture de système sont des personnes dépendantes qui nécessitent une aide dans leur quotidien. Pour ce faire, le système ambiant intelligent va comprendre l'environnent où se produit des événements détectés par

des capteurs. Ce système doit pouvoir fusionner ces événements pour comprendre la situation et pouvoir décider et agir pour exécuter un ou plusieurs services logiciels et/ou matériels. Le système ambiant intelligent va sélectionner les services qui peuvent être matériels, comme par exemple : un robot pour se déplacer, un robot pour la manipulation, ou des services logiciels comme un service qui va juste appeler un opérateur, un service pour faire un diagnostic ou un service pour un traitement de données, etc. Enfin, ce système ambiant intelligent aura une bibliothèque de services matériels et logiciels qui lui permettront de faire appel à des services pour réaliser les trois tâches « sensing-understanding-acting » dans un environnement ambiant. Du fait que tout est en réseaux connectés, l'architecture va permettre de fournir des services à l'utilisateur n'importe où et n'importe quand. Cette architecture sera validée par des spécifications et des simulations via des réseaux de Pétri temporels et stochastiques.

1.3 Problématique

Notre travail est basé sur la conception d'un système d'interaction homme-robot-environnement appliqué à un fauteuil roulant muni d'un bras manipulateur, sachant que les systèmes multimodaux ont comme entrées et sorties une combinaison de plusieurs modalités. Le principal défi de notre travail est de construire des moteurs de fusion et de fission capables de gérer les entrées multimodales tout en prenant en compte les informations sémantiques du contexte.

Le premier problème rencontré lors de l'élaboration du cahier des charges est de choisir une représentation commune des modalités et du contexte qui assurera sa compréhension par le système. Notre réponse est de construire une base de connaissances qui inclut tous les éléments de l'environnement désiré. Pour cela nous allons utiliser le concept d'ontologie et définir tous les éléments de l'environnement d'évolution du système et définir la relation entre eux. Ainsi, l'utilisation de l'ontologie assurera l'ouverture du système et sa réutilisabilité.

Le deuxième problème est de savoir quelle modalité est disponible et peut être considérée comme une entrée ou une sortie sachant que l'environnement est dynamique. La

réponse à cette question est de construire un système qui prend en considération l'information contextuelle sémantique ([14],[30], [31]). Ceci peut être effectué en récupérant les informations de l'environnement en continu.

Le troisième problème est comment comprendre la requête de l'utilisateur et comment fusionner toutes les données d'entrées. Pour cela, nous allons construire un moteur de fusion à base de règles qui fusionnera les entrées selon des modèles prédéfinis dans l'ontologie.

Le quatrième problème est de savoir comment envoyer des sous-tâches élémentaires aux modalités de sortie et répondre à la demande de l'utilisateur. Pour cela, nous allons construire un moteur de fission à base de règles qui pourra subdiviser les résultats de la fusion en utilisant des patterns (motifs) prédéfinis dans l'ontologie et envoyer les résultats aux modalités de sorties disponibles.

Le cinquième et dernier problème sera de savoir comment valider l'architecture proposée. Pour ce faire, nous avons choisi d'utiliser les réseaux de Pétri temporels et stochastiques et l'outil CPN Tools pour la modélisation de notre architecture et la visualisation des simulations.

1.4 Etat de l'art

Avant de proposer notre architecture il est nécessaire de faire un état de l'art sur les travaux existants dans les différents domaines pour pouvoir faire des comparaisons et ainsi proposer une architecture novatrice qui permettra de prendre en charge tous les aspects de notre objectif et apporter une solution aux problèmes des systèmes existants. Pour ce faire nous nous sommes concentrés sur les travaux de fusion et de compréhension, les travaux de sélection de services et leur compositions, et enfin le travail sur leur exécution (fission).

1.4.1 Fusion des données

La fusion multimodale ([32], [33], [34]) constitue la composition d'informations issues de plusieurs modalités dans le but d'aboutir à la composition d'un service voulu par un utilisateur. Nous trouvons dans [35] la définition suivante : "La fusion de données décrit les méthodes et les techniques numériques permettant de mélanger des informations provenant de sources différentes (de différentes modalités) afin d'obtenir une décision ou une estimation". Pour pouvoir structurer notre travail et définir les étapes à suivre pour aboutir à l'objectif de recherche, nous devons répondre aux questions suivantes : quelle méthode utiliser pour la fusion ? Quel est le meilleur moment pour fusionner des données ? Doit-on fusionner toutes les données issues des modalités ou doit-on procéder à une sélection ?

Pour réponde à ces questions, nous avons trouvé dans les travaux de Pradeep K. Atrey et al [36] un état de l'art sur les stratégies de fusion multimodale, qui sont utilisées pour combiner de multiples modalités afin d'accomplir diverses tâches d'analyse multimédia. Les recherches de ces dernières années se sont intéressées à la fusion multimodale en raison de l'avantage qu'elle fournit pour diverses tâches d'analyse multimédia. L'intégration de plusieurs médias, leurs fonctionnalités associées, ou la prise de décision intermédiaire dans le but d'effectuer une tâche d'analyse sont présentées comme étant une fusion multimodale [36]. Une analyse de tâches multimédias implique la prise en compte du traitement des données multimodales dans le but d'obtenir des informations précieuses sur les données, une situation ou une activité de niveau supérieur. Des exemples de tâches d'analyse multimédia peuvent être trouvés dans la détection des concepts sémantiques, la détection audio-visuelle d'un interlocuteur, le suivi de personnes, la détection d'événements, etc. Les données multimédias utilisées pour ces tâches pourraient être sensorielles (comme l'audio, la vidéo, le RFID (Radio-frequency identification)) ou non-sensorielles (comme les ressources WWW ou les bases de données). Ces médias et leurs caractéristiques sont fusionnés ensemble pour la réalisation de différentes tâches d'analyse. La fusion de plusieurs modalités peut fournir des informations complémentaires et accroître la précision de la prise de décision du processus

global [36]. La multi modalité donne accès à différentes modalités et leur utilisations sur la base de leurs accessibilité et disponibilité.

Depuis le premier système multimodal, le fameux système de Bolt "Put That There" [37], plusieurs systèmes multimodaux ont été proposés. Par exemple, pour le système de dialogue homme-robot, Prodanov et Drygajlo [38] ont construit un réseau basé sur une structure bayésienne pour l'interprétation des signaux multimodaux. Le système a été utilisé pour un dialogue entre le robot guide RoboX et les visiteurs d'un musée dans des conditions bruyantes. L'utilisation du réseau bayésien dans ce travail a permis la combinaison de la reconnaissance vocale dans un environnement bruyant avec des données à partir d'un scanner laser utilisé pour détecter la présence de personnes à proximité du robot. Dans un autre travail, nous trouvons un compte rendu sur les détections d'algorithmes de fusion pour les robots portables [39]. Dans cet article les auteurs mettent en évidence les différents moyens de fusion pour les capteurs multimodaux. La fusion combine les informations provenant de différents capteurs, soit en utilisant un algorithme de fusion unique, un switching unimodal, un switching multimodal ou un algorithme parallèle de fusion pour plusieurs capteurs (mixing). D'autre part un système cortical distribué pour le traitement de la fusion de capteurs a été mis en œuvre par Axenie et Conradt [40] pour l'estimation du mouvement d'un robot mobile autonome. Les auteurs proposent un modèle "qui est un réseau d'unités de traitement dont la connectivité est obtenue par les relations définies entre les unités. Les relations entre les unités peuvent être codées de manière explicite dans le réseau ou obtenues à la suite d'un processus d'apprentissage " [40]. En outre, le travail présenté dans [41] sur le robot mobile utilise la fusion de données issues de camera et du RFID (Radio-frequency identification) pour la recherche et le suivi d'une personne à l'aide d'un robot mobile. Pour la méthode de recherche les auteurs utilisent un filtre à particule, et pour le suivi de la personne ils ont élaboré une stratégie de contrôle utilisant plusieurs capteurs basés sur la sortie du suiveur et les données RFID.

L'interaction homme-ordinateur utilise la fusion multimodale pour l'interprétation des modalités d'entrée. Pour ce faire, un exemple est présenté par Reddy et Basir [42]

dans le raisonnement probant basé sur les concepts pour la fusion multimodale dans l'interaction homme-ordinateur. Dans ce travail, une approche est proposée pour la fusion sémantique des différentes modalités d'entrée basées sur des modèles. Cette architecture est appliquée sur un système multimodal composé d'un capteur de reconnaissance de gestes et d'une interface de calcul. D'autre part, la fusion de la perception multimodale basée sur l'ART (Theory Adaptive Resonance) pour les robots présentée dans [43] utilise la théorie ART pour la fusion multimodale. L'ART est un réseau de neurone non supervisé qui a une capacité rapide d'apprentissage incrémentielle en ligne.

De plus, le concept d'ontologie est également utilisé pour la fusion multimodale. Par exemple, dans [44] les auteurs présentent leurs travaux sur la détection de violence dans les films. Dans ce travail, l'ontologie est utilisée pour la fusion multimodale. Pour ce faire, les auteurs ont d'abord "réalisé une analyse audio-visuelle pour extraire les signaux audio et visuels. Ensuite deux approches de fusion différentes ont été utilisées : la première est une fusion multimodale qui fournit des décisions binaires sur l'existence de violence, et la seconde est une fusion basée sur l'ontologie et le raisonnement en combinant les listes de signaux audiovisuels avec les violences et les ontologies multimodales" [44]. Dans le même contexte, l'ontologie est utilisée dans la fusion multimodale pour les systèmes d'interaction [45]. Dans ce travail, un moteur de fusion est construit pour un système multimodal. L'environnement est décrit dans l'ontologie puis utilisé dans le moteur de fusion. Il utilise des langages du web sémantique basé sur les standards du W3C ([46], [47]). Ce travail prend en compte uniquement la fusion des données d'entrée et l'applique sur un seul scénario qui est "get that here".

1.4.2 Fission des données

Le processus de fission ([34], [48], [49]), qui consiste à subdiviser les résultats du moteur de fusion en sous-tâches élémentaires, est très important dans une interaction multimodale. En effet le moteur de fission est la partie responsable de la subdivision du résultat de fusion. Cette subdivision va permettre l'envoi de commandes élémentaires aux modalités de sortie selon leurs disponibilités et ainsi répondre aux besoins de l'utilisateur.

Les auteurs des travaux présentés dans [50] et [51] présentent dans leur travail une fusion multimodale, fission et simulation en réalité virtuelle d'un system robotique ambiant intelligent. Dans ce travail, les auteurs ont conçu les agents de fusion et de fission qui communiquent avec le web sémantique en utilisant le langage EKRL. "La programmation des agents se fait en réutilisant des concepts et des modèles par défaut, et en ajoutant des modèles spécifiques de la requête dans la mémoire de l'agent en utilisant l'éditeur de mémoire" [50]. De plus, dans le travail sur l'adaptation de la fission multimodale selon les besoins de l'utilisateur, nous trouvons dans [52] la présentation de l'interface « GUIDE » dédié à l'amélioration de l'interaction entre les personnes et les ordinateurs. Le moteur de fission présenté suit trois tâches lors de la subdivision de l'information. La première est la construction du message, la deuxième est la sélection de la modalité et la troisième est la coordination de l'information de sortie. Le travail présenté dans [52] est basé sur le modèle conceptuel « What-Which-How-Then » (WWHT). Ce modèle conceptuel pour l'interaction multimodale est présenté dans [24], [53] et [54]. Il décrit le cycle de vie d'une représentation multimodale selon les changements du contexte d'interaction évolutif. Dans [49], les auteurs proposent un modèle générique et formel pour la conception et la validation d'interactions multimodales basé sur le modèle WWHT. Leur proposition consiste à définir un système générique pour l'interaction multimodale en particulier pour la fission des modalités de sortie, ce travail se base sur un scénario d'interaction de sortie en utilisant le système SmartKom. D'autre part, pour la gestion de la boîte de dialogue pour une maison intelligente, nous trouvons le travail présenté dans [55] sur la description d'une intégration de fusion et de fission multimodale avec un gestionnaire de dialogue SCXML. Les auteurs ont utilisé un logiciel de composants open source qui satisfait les recommandations architecturales du W3CMMI.

De plus dans [56] les auteurs ont utilisé le concept d'ontologie pour la modélisation des règles de fission. Les auteurs ont présenté dans leur travail un exemple de scénario «apportez-moi une tasse» et comment le robot doit répondre à cette requête à l'aide d'un réseau de Pétri coloré. Le travail se concentre uniquement sur la fission de don-

nées et présume que les données d'entrée ont bien été fusionnées. Le fonctionnement de l'ontologie est modélisé uniquement par réseaux de Pétri en utilisant l'outil CPNTools. Oviatt et Cohen [57] présentent les bénéfices de la combinaison de multiples modalités en entrée et en sortie d'un système multimodal. Les avantages résident dans le fait que le système devient plus robuste et la probabilité d'erreurs est réduite. En outre, la combinaison de la fusion et la fission multimodale pour la gestion du dialogue est présenté dans [58]. Les auteurs présentent leurs travaux sur la capacité d'interaction et de communication d'un robot compagnon. Le système présenté est une approche adaptative au contexte de l'interaction multimodale en utilisant plusieurs couches de contexte. De même, une approche orientée contexte est présentée dans [59] pour la génération de rétroaction multimodale dans un environnement intelligent. Dans ce travail, le NAIF (Natural Ambient Intelligence Framework), qui permet la création d'environnement intelligent, est utilisé pour la génération de sorties multimodales. Enfin, une étude sur l'interaction multimodale dans un environnement ambiant est présentée dans [60]. Ce travail est une étude exploratoire pour déterminer la relation entre les entrées et les sorties multimodales et comment les modalités de sortie peuvent influencer le choix des modalités d'entrées utilisées par une personne lors de son interaction avec le robot.

1.4.3 Architecture ambiante et exemple d'application :

Jean-François Ladry et al [61] présentent dans leurs travaux une description de technique de fusion pour les interfaces multimodales. Le document se concentre sur les exigences relatives à la modélisation et la construction des moteurs de fusion pour les interfaces multimodales. Les auteurs proposent une technique de description formelle dédiée à l'ingénierie des systèmes interactifs multimodaux capables de relever les défis de moteurs de fusion. Ces avantages sont présentés sur une série d'exemples illustrant à la fois les concepts et les processus [61]].

La fusion est un élément particulier d'un système multimodal interactif qui vise généralement à synchroniser plusieurs flux d'informations produits par l'utilisateur lors de l'interaction avec les dispositifs d'entrée. De ce fait, les auteurs nous présentent com-

ment la technique de description formelle ICO (Interactive Cooperatives Objects) peut être utilisée pour la modélisation des moteurs de fusion et, plus précisément, ils nous présentent des constructions de base utiles exprimés en ICO. Ensuite, des exemples de modélisation de fusion sont présentés. Le modèle « Arch » pour la conception de moteur de fusion est présenté dans cet article [61]. L'ICO (Interactive Cooperatives Objects) est une technique de description formelle dédiée à la spécification des systèmes interactifs. Elle utilise des concepts empruntés à l'approche orientée objet (instanciation dynamique, classification, encapsulation, héritage, relation client/serveur) pour décrire les aspects structurels ou statiques des systèmes, et utilise les réseaux de Pétri de haut niveau pour décrire leur dynamique ou leur comportement. On trouve dans cet article [61] un exemple d'application sur un jeu d'échecs virtuel. Dans cet exemple, un utilisateur peut interagir avec des pièces du jeu d'échecs en utilisant un gant de données pour identifier les mouvements du doigt et un FOB (Flock of Bird qui est un détecteur de mouvement) pour localiser la main dans un environnement 3D.

A.A. Mekonnen et al [62] présentent dans leurs travaux sur la coopération entre un robot mobile et des caméras ambiantes pour le suivi multi-personnes, une stratégie de coopération entre des caméras ambiantes et des capteurs embarqués sur un robot mobile [62]. Le robot en question doit pouvoir suivre une personne taguée et en même temps éviter les autres personnes présentes dans son environnement proche. L'approche proposée, dite "tracking-by-detection", utilise le formalisme type filtre particulaire par chaîne de Markov pour fusionner les détections visuelles déportées et les détections issues des divers capteurs embarqués (laser, vision active, RFID) [62]. L'architecture proposée par les auteurs est articulée autour de trois blocs (A, B et C), qui permettent respectivement de :

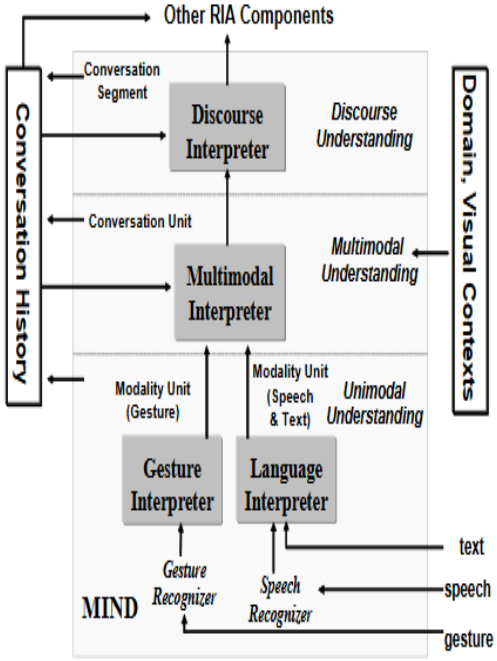
- Identifier et suivre une personne cible identifiable par son badge RFID
- Détecter les personnes non badgées au voisinage immédiat du robot
- Gérer les actionneurs du robot pour contrôler ses déplacements

L'intégration des fonctionnalités précédentes est implémentée sur le robot Rackham en C/C++ et embarquée via des modules GenoM (Generator of Modules) qui sont des

outils d'aide à la conception d'architectures logicielles temps-réel. La communication sans fil entre la CPU dédiée aux caméras déportées et la CPU embarquée sur le robot est à faible débit et doit donc éviter tout transfert d'images brutes. Ainsi, la détection des passants est réalisée sur la CPU du PC dévolue aux caméras ambiantes qui sont connectées au PC via une connexion firewire, tandis que les autres fonctionnalités sont gérées par la CPU embarquée sur Rackham [62].

D'autre part, pour renforcer l'interprétation multimodale basée sur le contexte dans les systèmes d'interaction, Joyce Chai [63] a développé une représentation sémantique pour collecter les informations d'entrée émises par l'utilisateur ainsi que la conversation globale. Cette représentation nommée MIND (Multimodal Interpretation for Natural Dialog), permet d'identifier la signification des entrées multimodales des utilisateurs. Dans un contexte de conversation, les entrées issues par l'utilisateur peuvent être abrégées ou imprécises. La combinaison seule de plusieurs entrées ne peut parvenir à une compréhension complète du dialogue. Par conséquent, MIND utilise un contexte riche (par exemple, contexte de la conversation et contexte du domaine) pour améliorer l'interprétation multimodale [63]. Les auteurs nous présentent dans leur article [63] un exemple d'interaction entre MIND et des personnes cherchant à acheter une maison dans une ville précise, un exemple est présenté sur la Figure 1.1. Cette recherche se fait grâce à l'interaction multimodale entre MIND et l'utilisateur en utilisant deux modalités, à savoir la parole et le geste. L'utilisateur peut dialoguer avec MIND en utilisant un vocabulaire habituellement utilisé entre les personnes. Cet exemple montre que les entrées multimodales des utilisateurs présentent une large gamme de variétés. Ils pourraient être abrégées, ambiguës ou complexes. Donc la fusion directe des entrées ne peut aboutir à une bonne compréhension du dialogue et le traitement de ces entrées nécessite la prise en compte du contexte. Actuellement MIND utilise trois types de contextes : le contexte du domaine, le contexte de la conversation, et le contexte visuel. Le contexte du domaine fournit des informations sur le domaine, le contexte de la conversation reflète l'état d'avancement de l'ensemble de la conversation et le contexte visuel donne les structures sémantiques et syntaxiques détaillées des objets visuels et leurs relations.

Par ailleurs nous trouvons dans [64] une présentation d'approche orientée composant pour la modélisation d'une architecture de système dédié aux personnes à mobilité réduite.



	A collection of houses are shown on the map of Irvington
U1:	*Speech*: How much is this? *Gesture*: Point to the screen (not directly on any object)
R1:	*Speech*: Which house are you interested in? *Graphics*: Highlight two house icons
U2:	*Speech*: The green one.
R2:	*Speech*: The green house costs 250,000 dollars.
U3:	*Speech*: What about this one? *Gesture*: Point to a house icon on the screen
R3:	*Speech*: This house costs 320,000 dollars. *Graphics*: Highlight the house icon and show a picture
U4:	*Speech*: Show me houses with this style around here *Gesture*: Point to a position east of Irvington on the map
R4:	*Speech*: This is a Victorian style house. I find seven Victorian houses in White Plains. *Graphics*: Show seven houses in White Plains
U5:	*Speech*: Compare these two houses with the previous house. *Graphics*: Point to the corner of the screen where two house icons are displayed
R5:	*Speech*: Here is the comparison chart. *Graphics*: Show a chart

FIGURE 1.1: Les composants de MIND suivi d'un segment de conversation

Nous trouvons dans [64] la définition suivante : "l'approche composant telle qu'elle est définie dans le paradigme CBSE (Component Based Software Engineering) définit qu'un composant est une « boîte noire » ne présentant que ses interfaces et ses exigences. Un composant est considéré comme substituable, réutilisable et composable (des composants peuvent être couplés pour obtenir des fonctions plus complexes). À partir de composants élémentaires (Lampes, Volets, Portes), le concepteur peut définir des composants plus complexes de manière récursive jusqu'à l'obtention d'un système global". L'approche proposée est basée sur l'Ingénierie Dirigée par les Modèles (IDM), elle permet d'automatiser la génération du code de commande pour les systèmes domotiques.

Deux acteurs peuvent être identifiés dans cet article, le concepteur et l'utilisateur. Sachant que le concepteur est la personne chargée de construire un environnement domotique, il sera chargé avant tout de construire un environnement de vie adapté à l'utilisateur qui présente des capacités motrices réduites. Pour ce faire, nous trouvons dans cet article une proposition d'architecture basée sur l'IDM. Nous trouvons dans [64] la définition suivante : "L'IDM se distingue des différentes approches issues du génie logiciel. Sa démarche pose comme principe que toute partie d'une application informatique est générée à partir de modèles. L'architecture MDA (Model Driven Architecture) fournit des outils, concepts et langages pour créer et transformer des modèles, développant ainsi une communauté importante autour de ce domaine".

Lyazid Sabri et al., nous présentent dans leurs travaux sur le « Narrative reasoning for cognitive ubiquitous robots » [65], une approche narrative pour la représentation et le raisonnement sur l'espace, les changements et les occurrences dans l'environnement ambiant intelligent. Le domaine d'application principal présenté est l'interaction homme-robot dans les systèmes intelligents ambiants (par exemple, les maisons intelligentes). Ces travaux permettent d'explorer le potentiel de la représentation narrative et le raisonnement pour reconnaître et comprendre les situations. Cette approche utilise les structures hiérarchiques de prédicats sémantiques et les rôles fonctionnels du Narrative Knowledge Representation Language (NKRL) et permet des descriptions sémantiques des entités, des événements et des relations entre les événements. Un scénario dédié à la surveillance dans la maison intelligente des personnes âgées est présenté et analysé pour montrer la faisabilité de l'approche proposée et sa capacité à expliquer les chaînes causales entre les événements observés [65]. Un des principaux avantages de NKRL est la possibilité de mettre en place le modèle sémantique de l'environnement ambiant en utilisant l'Ontologie des concepts (HClass) et l'Ontologie des événements (HTemp).

Dans le cadre du projet PEIS-Ecology (Ecology of Physically Embedded Intelligent Systems) [66], les auteurs combinent les domaines de la robotique ubiquitaire et l'intelligence ambiante. Cette combinaison offre la possibilité de construire des robots intelligents qui sont au service de l'homme. Dans cette approche, les fonctionnalités ro-

botiques avancées ne sont pas fournies par un robot extrêmement intelligent, mais par la coopération de plusieurs composants robotiques simples. En effet, un PEIS est défini comme étant un ensemble de composants logiciels interconnectés résidant dans une seule entité physique (un PEIS peut être aussi simple qu'un toasteur ou aussi complexe qu'un humanoïde). Chaque composant doit avoir des liens le reliant à des capteurs et actionneurs. Ainsi que des ports d'entrées et de sorties les reliant à d'autres composants dans le même PEIS où à d'autres PEIS. De plus, tous les PEIS sont connectés grâce à un modèle uniforme de communication distribuée. Le nombre et la performance de chaque PEIS ne sont pas forcement connus à priori : un nouveau PEIS peut intégrer ou quitter l'écologie à n'importe quel moment, et son existence doit être détecté automatiquement par les autres éléments. L'écologie en question est l'ensemble de dispositifs robotiques qui sont répartis de manière omniprésente sous forme de capteurs, d'actionneurs, d'appareils intelligents, d'objets marqués actifs (muni d'un IC-tag) ou de robots mobiles traditionnels. Ces dispositifs communiquent et collaborent en fournissant des informations ou en exécutant des actions dans l'écologie. L'approche utilisée dans l'écologie des PEIS [66] simplifie de nombreux problèmes rencontrés dans la robotique autonome actuelle. Les fonctionnalités complexes qui sont "on-board" sont remplacées par des fonctionnalités simples et une communication "off-board". Prenons comme exemple un robot qui essaye de saisir une bouteille de lait. Dans un écologie-PEIS, le robot n'aurait pas besoin d'utiliser sa caméra pour acquérir les propriétés de la bouteille (forme, poids, etc.) afin de calculer les paramètres de saisie. Le robot se connecte directement à l'écologie et communique avec la bouteille munie d'un IC-tag qui contient ses informations.

Pour le projet Robot in the kitchen [67], les auteurs nous présentent leurs travaux de recherche sur le robot de service dans la cuisine. Les auteurs visent à développer des robots de services intelligents qui opèrent dans des environnements humains standards, en automatisant les tâches courantes. L'objectif est de permettre à un robot de faire, comme les humains et les animaux, toutes les tâches de détection, de délibération et de sélection d'action par lui-même. Pour ce faire, les auteurs choisissent de combiner la perception intelligente et le contrôle avec l'informatique ubiquitaire pour obtenir un ro-

bot ubiquitaire. Nous trouvons dans l'article la définition suivante : "L'informatique est ubiquitaire lorsque les périphériques informatiques sont distribués et enfouis de manière invisible dans les objets de la vie quotidienne. Ces dispositifs sensibles à leur environnement, se connectent automatiquement les uns aux autres pour former des réseaux de capteurs, échanger des informations, et agir pour modifier leur environnement" [67]. Pour parvenir à leur fin et répondre à leurs exigences de conception, les concepteurs ont utilisé le logiciel open source Player. Le projet Player (anciennement connu sous le nom Player/Stage) produit des outils pour le développement rapide de code de commande des robots. Les trois principaux outils du projet sont : Player, Stage, et Gazebo. Player est une couche d'abstraction matérielle pour les appareils robotiques. Stage et Gazebo sont, respectivement, des simulateurs 2D et 3D de multi-robots. L'approche permet aussi de combiner le concept de perception active avec la détection de capteurs. En d'autres termes, pendant que des capteurs sont placés dans l'environnement pour qu'ils puissent donner la meilleure information possible, le système se configure automatiquement dans une certaine mesure, en identifiant les flux de capteurs en leur assignant des étiquettes qui leur sont propres.

1.5 Synthése et Solution proposée

La revue de littérature présentée dans la section précédente nous a permis de synthétiser les travaux existant dans le domaine de l'interaction multimodale et de la robotique d'aide à la personne dépendante. Cette synthèse nous a conduit à conclure qu'aucun travail n'a permis la combinaison d'un moteur de fusion et d'un moteur de fission pour l'interaction multimodale homme-machine en utilisant une ontologie comme base de connaissance.

La revue de littérature présentée précédemment a été très intéressante pour notre travail. Elle nous a permis de définir notre architecture de fusion et fission multimodale basée sur le concept d'ontologie, et tient compte de son contexte sémantique contrairement à d'autres travaux. Pour la plupart des cas dans la littérature, il est difficile de trouver une validation temporelle et spatiale et une réorganisation en temps réel. Les condi-

tions ne sont pas basées sur l'intelligence ambiante et non sur la base de la mémoire ou du contexte sémantique et du traitement des agents. Les méthodes disponibles pour résoudre la validation temporelle et spatiale des agents basés sur le savoir sont les méthodes formelles (bien que très coûteuses au niveau du calcul) et sont probablement les plus puissantes. Les travaux existants sur la fusion et la fission multimodale sont souvent séparés et sont conçus pour des systèmes spécifiques. Néanmoins, à notre connaissance, aucun travaille ne combine la fusion et la fission multimodale en utilisant l'ontologie comme base de connaissance. Par exemple, le travail présenté dans [55] combine la fusion et la fission pour un gestionnaire de dialogue d'une maison intelligente. Ce travail utilise des moteurs de fusion et fission basés sur OSGi, qui est un framework Java pour le développement et le déploiement de logiciels et de bibliothèques modulaires. OSGi s'est révélé être suffisamment souple pour aborder les différents paramètres dans les déploiements réels des maisons intelligentes. Dans le système que nous proposons, nos agents contrôlent un fauteuil roulant et d'autres robots afin d'aider l'utilisateur handicapé à la maison (ou ailleurs) pour faire des travaux essentiels dans leur vie quotidienne. Par exemple, dans le travail de Mataric [68] la planification ou la réorganisation est uniquement basée sur des obstacles statiques et dynamiques. Les conditions ne sont pas basées sur l'intelligence ambiante et non sur la base de la mémoire ou du contexte sémantique et du traitement des agents. Même s'il applique une bonne approche de planification robotique, son algorithme amélioré n'est pas basé sur l'intelligence d'un agent (capacités de raisonnement multimodal) et évite les comportements d'apprentissage à partir de situations observées dans le temps. Notre travail est intéressant en ce sens car il est adaptable en temps réel.

CPN Tools est une application logicielle qui modélise les réseaux de Petri ; Il est scientifiquement reconnu comme une bonne méthodologie stochastique d'ingénierie logicielle pour vérifier et évaluer de manière aléatoire les réactions des agents dans tous les types de bonnes et de mauvaises situations. Les réseaux de Pétri sont des outils appropriés pour simuler et valider une conception de logiciel comme une communication d'agent, la vérification et la validation de pièces de logiciels sémantiques comme les

services Web, ou les agents de communication basés sur le Web. Il peut également être utilisé pour modéliser les comportements autonomes pour les systèmes multi-agents hétérogènes afin de vérifier leur fonctionnement dans des conditions et contraintes de mission prédéfinies. Nous validons notre approche à l'aide des outils CPN. À notre connaissance, il n'existe pas de formalisme formel basé sur le modèle de réseaux de Pétri qui aide à la validation des processus de fusion et de fission des systèmes d'interaction multimodaux utilisant une ontologie comme base de connaissance.

Notre objectif est de renforcer les interactions homme-machine et l'interaction machine-machine en utilisant plusieurs modalités d'entrée et de sortie. Sur la base de la revue de littérature présentée ci-dessus, nous concluons que la plupart des applications présentées fonctionnent bien, mais manquent de flexibilité. Les hypothèses initiales et les données sont très strictes : cela signifie que le domaine d'application est limité à une application correspondant à un environnement précis. Les aspects de l'interaction environnement-machine ne sont pas toutes mises en œuvre et les interactions homme-environnement ne sont pas présentées de façon sémantique lors du raisonnement et de la prise de décision. La plupart de ces systèmes présentent l'interaction en utilisant uniquement un système de fusion, comme par exemple les travaux présentés dans la section "Fusion des données" dans la section précédente. Tandis que d'autres, présentés dans la section "Fission des données", se concentrent uniquement sur les systèmes de fission. L'utilisation de deux systèmes en une seule boucle d'interaction n'est pas présentée dans les systèmes étudiés. À notre connaissance, nous ne trouvons aucune base théorique pour le problème de l'interaction ambiante dans ces travaux. Il n'y a pas de présentation sémantique formelle de la connaissance, et les processus de fusion et de fission ne sont pas modélisés d'une manière abstraite en utilisant des réseaux de Pétri temporels et stochastiques. La précision et la cohérence de ces modèles ne sont pas vérifiables.

1.6 Conclusion

Nous avons présenté dans ce chapitre un état de l'art non exhaustif sur les travaux existants dans la littérature sur la fusion des données, la fission et la composition de services et leurs exécutions dans le traitement des données multimodales pour différents domaines d'applications. Nous rappelons que notre domaine de recherche se situe dans la robotique d'aide aux personnes dépendantes. Notre objectif est de proposer une architecture ambiante d'interaction homme-robot-environnement. Cette architecture va permettre aux robots de prendre en compte le contexte évolutif pour fournir continuellement du service à l'utilisateur.

Les avantages de notre approche sont les suivants :

1. Notre approche est tout à fait différente car notre objectif est de fournir une architecture capable de résoudre de nombreux problèmes d'interaction dans le cadre de l'assistance ambiante à partir d'exigences techniques générales. La représentation sémantique de l'environnement est formalisée en utilisant une ontologie, et les processus de fusion et de fission sont modélisés en utilisant un ensemble de règles logiques de premier ordre. Ceci fournit une expression riche des contraintes entre les entités et les types de relations entre l'environnement, l'homme et la machine. La différence entre les ontologies existantes utilisées dans le domaine des interactions dans les ouvrages cités et celle présentée dans notre travail est que l'ontologie utilisée dans notre système contient des modèles d'événements associés à des concepts généraux. Les concepts et les événements sont communs dans divers domaines ; afin qu'ils puissent être plus spécialisés au domaine des concepts spécifiques. Cela implique que les modèles correspondent à des concepts génériques qui suffisent à la réalisation de toutes les exigences de l'interaction.

2. Notre approche utilise les réseaux de Pétri temporels et stochastiques pour modéliser l'architecture. Cela nous a permis de modéliser le temps de traitement en utilisant des lois de probabilité. Par conséquent, le contrôle de cohérence permet d'assurer que les modèles de l'ontologie (concepts, événements et instances) sont

sémantiquement cohérents et évitent les conflits. L'utilisation de réseaux de Pétri nous a aussi permis de vérifier l'exhaustivité, l'ordre d'arrivé, et la contrainte temporelle (en raison de conditions préalablement définit des modèles de règles et leurs cohérences dans l'ontologie). Enfin, cette utilisation va nous permettre d'exécuter les applications sur plusieurs scénarios pour différents modèles dans l'ontologie sur des différentes séquences d'événements générés, et vérifier la reconnaissance des événements composés liés aux modèles existants.

Nous concluons, à notre connaissance, qu'aucun travail n'a combiné le processus de fusion et de fission pour l'interaction homme-machine dans une architecture complète.

Chapitre 2

Méthodologie et Modélisation de l'architecture

2.1 Introduction

Ce chapitre présente la méthodologie de travail utilisée lors de la conception de l'architecture proposée. Une méthodologie de travail doit être conçue et suivie pour assurer la bonne succession d'étapes et la rigeur de la modélisation. Ce chapitre présente la modélisation de l'architecture qui se fera en plusieurs étapes. Tout d'abord la modélisation de l'environnement, puis la modélisation de la sélection de modalité, et enfin la modélisation du moteur de fusion ainsi que du moteur de fission.

2.2 Méthodologie générale

Notre objectif pour ce travail de recherche est d'élaborer un système multimodal de communication entre une personne à mobilité réduite et son fauteuil roulant muni d'un bras manipulateur. Pour pouvoir répondre à notre objectif, nous devons définir une méthodologie de travail qui va nous permettre de réaliser notre objectif tout en prenant en compte les travaux existants. Cette méthodologie va nous permettre de concevoir un nouveau système d'interaction qui facilitera l'utilisation du fauteuil roulant.

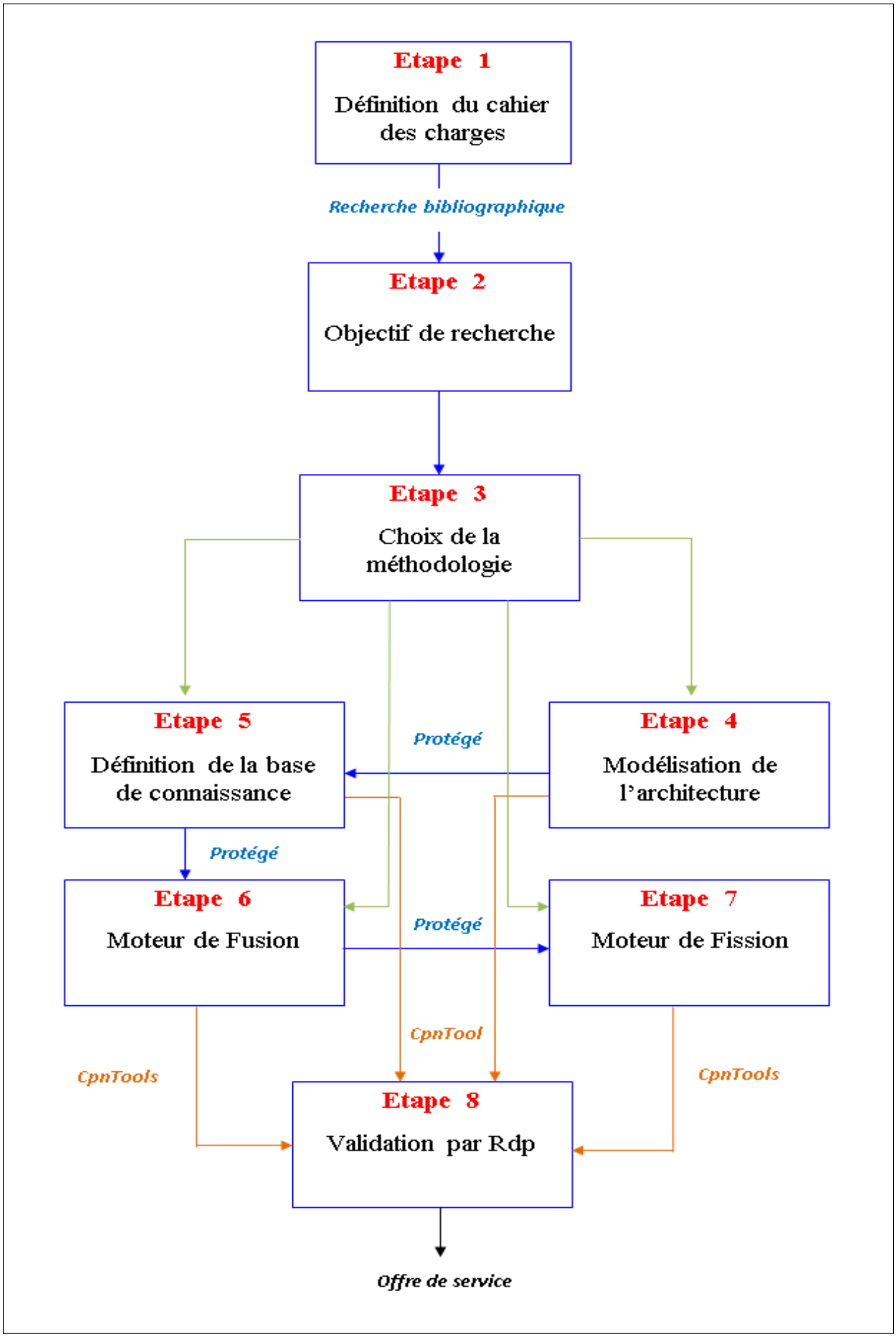


FIGURE 2.1: Méthodologie de travail

Pour ce faire, la Figure 2.1 représente la méthodologie générale de travail qui va nous permettre de concevoir notre architecture étape par étape. En effet, chaque étape sera composée d'un processus qui aura une ou plusieurs entrées et sorties. Les étapes seront reliées entre elles d'une façon logique et qui aboutira à une proposition de service selon la commande de l'utilisateur et l'état du contexte sémantique.

2.2.1 Etapes de la méthodologie de travail

- **Etape 1 : Définition du cahier des charges**

  Cette étape consiste en la définition des caractéristiques du système que nous voulons concevoir et cibler le domaine de travail. Cela va permettre de faciliter la sélection de la bibliographie qui permettra l'analyse de la documentation sur les travaux existant dans le domaine.

- **Etape 2 : Objectif de recherche**

  Apres avoir trié la bibliographie et sélectionné les documents relatifs à notre domaine, nous allons procéder à la synthèse des différentes publications relatives à notre travail et élaborer l'objectif de recherche. Cette synthèse permettra aussi d'élaborer une proposition de méthode de travail en utilisant les travaux trouvés dans la littérature pour répondre à notre objectif de travail.

- **Etape 3 : Choix de la méthodologie**

  La méthodologie choisie pour la conception de l'architecture de commande pour un fauteuil roulant muni d'un bras manipulateur est la commande multimodale en prenant en compte les informations sémantiques du contexte. Nous avons choisi la commande multimodale car elle permet l'utilisation de plusieurs modalités telles que la parole et le geste et rend l'interaction entre l'utilisateur et le fauteuil plus facile. Cette facilité est dûe au fait que l'utilisateur pourra communiquer avec son fauteuil en utilisant des moyens naturels. Cette caractéristique apportera un confort et une confiance lors de l'utilisation du système d'interaction. Sachant que le système visé par notre travail évoluera dans un contexte dynamique et sémantique, il

est impératif de prendre en compte cet aspect lors de la conception de l'architecture.

- **Etape 4 : Modélisation de l'architecture**

  La modélisation de l'architecture va nous permettre de donner un schéma descriptif des composants de l'architecture et les relations entre eux. Cette modélisation permettra aussi de voir l'enchaînement des étapes qui aboutira à l'offre de services selon la commande de l'utilisateur. L'architecture proposée sera composée de :

  ▷ **L'entrée** : Cette partie sera le siège des différentes modalités d'entrée qui pourront être utilisées par l'utilisateur lors de la requête de service ainsi que les différentes informations relatives au contexte dynamique et sémantique de l'environnement global. Les entrées sont prédéfinies, néanmoins, nous pouvons rajouter ou supprimer des entrées à tout moment selon le cas d'utilisation du système.

  ▷ **Le système multimodal** : Cette partie prendra en compte toutes les actions et les transformations relatives à la commande multimodale dans le but de traiter, comprendre et répondre à la requête de l'utilisateur. Tout d'abord, cette partie procèdera à la sélection de modalités selon leurs disponibilitées. Puis, les informations obtenues du contexte ainsi que les modalités seront fusionnées par le moteur de fusion qui par conséquent aura compris la requête de l'utilisateur. Lorsque les entrées sont fusionnées correctement, le moteur de fission procèdera à la subdivision du résultat du moteur de fusion pour pouvoir envoyer des actions élémentaires aux modalités de sortie. Le moteur de fusion et de fission se baseront sur les informations récupérées de la base de connaissance qui est l'ontologie du domaine d'évolution du système.

  ▷ **La sortie** : Cette partie sera composée des différentes modalités de sortie utilisées par le système pour répondre à la requête de l'utilisateur ou pour l'informer de l'évolution du traitement ou de la survenue d'une erreur au cours du traitement.

- **Etape 5 : Définition de la base de connaissance**

  Nous avons choisi d'utiliser le concept d'ontologie comme base de connaissance car elle permet une définition globale et détaillée de l'environnement d'évolution du système de commande multimodal. L'environnement d'évolution du système est l'endroit où se trouvent l'utilisateur et le fauteuil, comme par exemple l'intérieur de la maison ou l'extérieur. L'utilisation de l'ontologie permet aussi de définir les relations sémantiques entre les entités de l'environnement. Nous avons choisi d'utiliser l'outil Protégé [69] pour la conception de l'ontologie. Une présentation détaillée de cet outil sera abordée dans le Chapitre 3.

- **Etape 6 : Moteur de Fusion**

  Le moteur de fusion est la partie responsable de la compréhension et de la fusion des modalités d'entrée tout en prenant en compte l'état évolutif du contexte sémantique. Nous avons choisi d'utiliser la fusion à base de règles car cette méthode est souvent utilisée pour mieux estimer un objet en mouvement et assure un bon alignement temporel entre les différentes modalités. Les modèles de fusion seront stockés dans l'ontologie. Nous allons définir dix-neuf modèles de fusion possibles avec différents cas de figure. L'utilisation de l'ontologie comme base de connaissance permettra de s'adapter à l'état changeant de l'environnement et du profil de l'utilisateur.

- **Etape 7 : Moteur de Fission**

  Le moteur de fission est la partie responsable de l'envoi de tâches élémentaires aux actionneurs de sortie. Ces tâches sont obtenues par la subdivision du résultat de la fusion des données en utilisant les motifs de fission prédéfinis dans l'ontologie. Les motifs de fission sont composés de deux parties, à savoir un problème et une solution. En effet le problème dans notre cas est le résultat de la fusion, tandis que la solution est composée d'actions successives qui aboutissent à l'envoie d'actions élémentaires aux actionneurs de sortie. Lorsque l'ordre des actions d'entrée est respecté le système pourra répondre à la demande de l'utilisateur en

utilisant la fusion en premier puis la fission par l'envoie de tâches élémentaires aux actionneurs de sortie. L'ordre est prédéfini dans les modèles de fusion.

- **Etape 8 : Validation par réseaux de Pétri**

  La validation de l'architecture proposée est une étape très importante de notre travail. La validation sera faite par réseaux de Pétri temporels et stochastiques en utilisant l'outil Cpn Tools ([27], [28]). Nous avons modalisé chaque élément de l'architecture par des sous réseaux de Pétri.

  Les sous réseaux seront reliés entre eux pour former l'architecture globale. De ce fait, nous avons pu valider l'algorithme de fusion et de fission, la sélection de modalités ainsi que l'aspect temporel des modalités. Nous avons aussi créé un générateur aléatoire de commandes qui fournit différentes combinaisons de commandes. Cela nous a permis de valider la vérification et la compréhension du vocabulaire. Les résultats obtenus seront présentés dans le Chapitre 5.

2.3 Modélisation de l'architecture

Comme présenté précédemment la modélisation de l'architecture du système se fera en utilisant une ontologie. Pour cela, la modélisation se fera en plusieurs parties qui sont décrites ci-après :

2.3.1 Modélisation de l'environnement

L'environnement d'évolution du système robotique sera le voisinage du système robotique et de l'utilisateur, qui peut être l'intérieur d'une maison ou l'extérieur. Pour définir et modéliser correctement l'environnement du système il est impératif de définir les localisations potentielles du système robotique ainsi que tous les objets ou personnes qui le forment. Pour ce faire, nous allons définir une ontologie qui sera composée de :

- Le voisinage : Cette partie sera composée de toutes les entités présentes dans l'environnement qui sont au voisinage de l'utilisateur et du système robotique. Les entités seront les objets, les personnes et les localisations possibles du système.

- Contexte : Cette partie sera composée de quatre différents contextes que nous avons jugé importants pour le système et qui sont :
  - ▷ Le contexte environnemental : Cette partie sera responsable des valeurs issues de l'environnement, comme : le niveau de bruit, le niveau de luminosité, les conditions météorologiques ainsi que la température ambiante.
  - ▷ Le contexte de santé : Cette partie prendra en charge la surveillance de l'état de santé de l'utilisateur.
  - ▷ Le contexte du système : Cette partie prendra en charge les valeurs issues des capteurs embarqués sur le fauteuil ainsi que le niveau de batterie du système.
  - ▷ Le contexte de l'utilisateur : Cette partie sera composée d'informations relatives à l'utilisateur comme son nom et son type de handicap.
- Le vocabulaire : On définira dans cette partie le vocabulaire qui pourra être utilisé par la personne dépendante pour demander un service.
- Les relations sémantiques : Les relations sémantiques permettront de définir les relations entre les différents objets de l'environnement, les relations entre l'utilisateur et les différents contextes ainsi que les relations entre les modalités et le contexte.
- Les modèles de fusion et les motifs de fission.
- La vérification temporelle (entre les modalités et le temps maximum d'une commande) qui est nécessaire au bon fonctionnement de l'architecture.

2.3.2 Modélisation de la sélection de modalités

La sélection de modalités est importante pour notre système. Cette caractéristique permettra d'avoir un système optimal car le système sera capable de détecter quelle modalité est utilisable selon l'état du contexte sémantique.

Dans l'ontologie qui sera présentée dans le Chapitre 3 toutes les modalités utilisées seront définies en détail. De plus, en utilisant les propriétés d'objet nous allons pouvoir

définir les relations entre les modalités utilisées et les informations susceptibles de les influencer comme par exemple l'état du contexte. Nous allons aussi définir des instances pour chaque modalité et préciser leurs valeurs maximales ou minimales acceptables en utilisant les propriétés de données.

Enfin, la sélection de modalités selon leurs disponibilités sera faite grâce à des requêtes en utilisant le langage SQWRL ((Semantic Query Enhanced Web Rule Language) ([70], [71]). Ce dernier est un langage de requête basé sur SWRL ([72], [73], [74]) qui fournit des opérateurs SQL pour extraire des informations des ontologies OWL. L'outil Protégé que nous allons utiliser pour définir l'ontologie nous offre la possibilité d'intégrer ces requêtes et de les exécuter. Cette sélection se fera pour les modalités d'entrée avant de fusionner et pour les modalités de sortie après avoir fissionné le résultat du moteur de fusion du système.

2.3.3 Modélisation du moteur de Fusion

Le moteur de fusion sera responsable de la fusion des modalités d'entrée tout en prenant en compte l'état du contexte. Nous avons défini un algorithme de fusion, présenté sur la Figure 2.2. La fusion sera possible lorsque toutes les préconditions prédéfinies dans l'algorithme seront satisfaites. Si l'une des préconditions n'est pas satisfaite, la fusion ne pourra pas avoir lieu et un message d'information sera envoyé à l'utilisateur du fauteuil roulant muni d'un bras manipulateur.

Pour pouvoir assurer une bonne compréhension des évènements par le moteur de fusion nous avons défini dix-neuf modèles de fusion dans l'ontologie. Ces modèles seront présentés en détail dans le chapitre suivant. Les modèles de fusion seront composés de séquences successives d'évènements selon un ordre précis de commandes émises par l'utilisateur. L'ordre est très important car le sens de la commande peut changer selon la position de l'évènement au sein du modèle.

Nous avons défini des instances dans l'ontologie qui forment le vocabulaire utilisé par la personne dépendante. Ces instances seront liées par des propriétés d'objets aussi définis dans l'ontologie et qui permettront de préciser l'ordre d'arrivée des évènements.

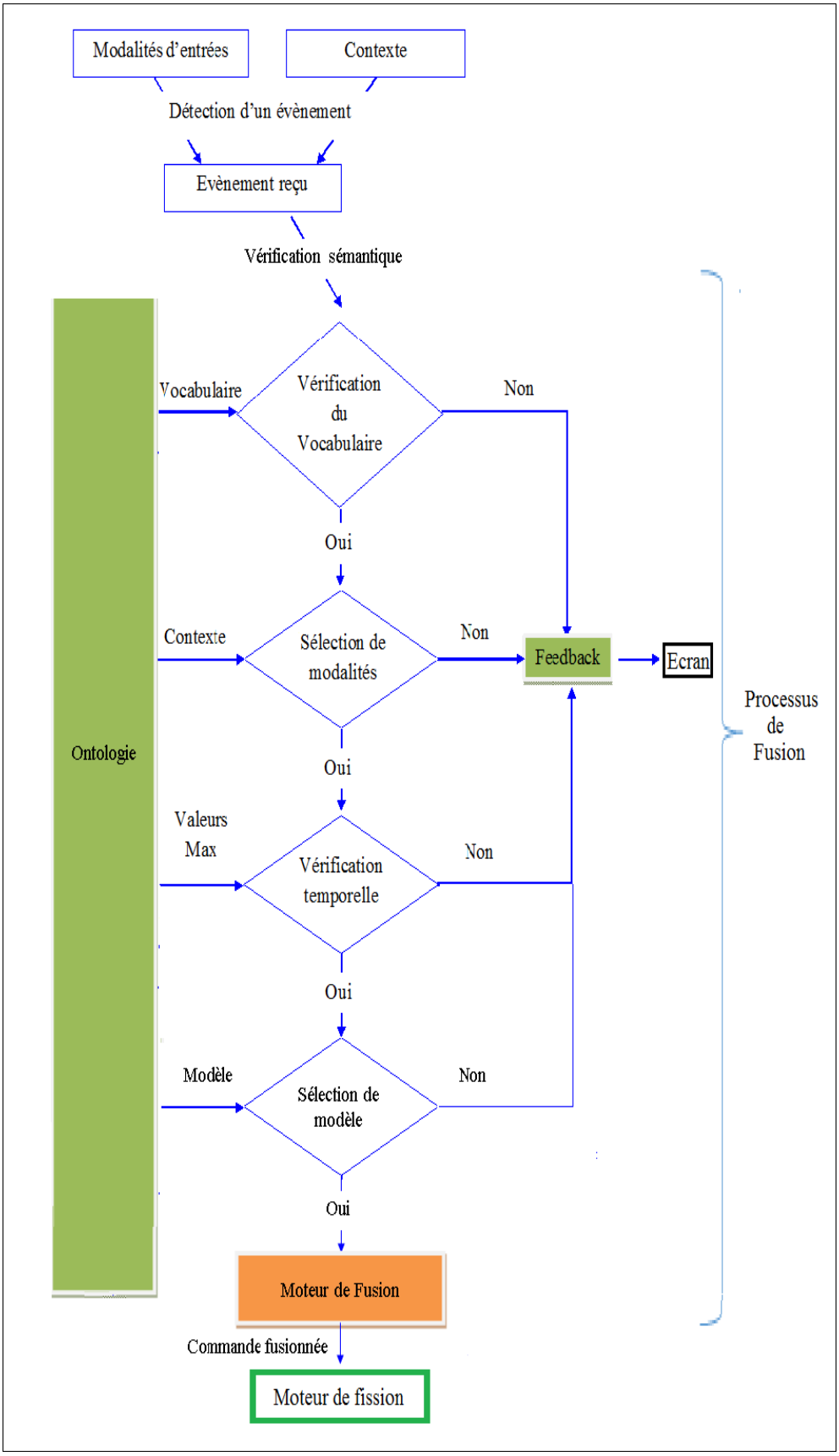


FIGURE 2.2: Algorithme du moteur de fusion

Lorsque le système détecte des évènements, il établira l'ordre d'arrivée de chaque évènement. Puis, le système procèdera à une comparaison des évènements au vocabulaire et aux modèles prédéfinis dans l'ontologie. Cette vérification se fera en utilisant des requêtes SQWRL pour la sélection de modèles et la vérification du vocabulaire. De plus, des règles sémantiques seront utilisées pour assurer le bon sens entre les évènements.

Comme présenté sur la Figure 2.2 lorsqu'un évènement est détecté par le système et que la vérification sémantique est satisfaite, le processus de fusion débute par la vérification du vocabulaire. Cette vérification se fera en comparant le vocabulaire des évènements reçus avec ceux définis dans l'ontologie. Puis, lorsque le vocabulaire est satisfait, le moteur de fusion procèdera à la sélection de modalités selon l'état du contexte et selon les informations définies dans l'ontologie.

Ensuite, le moteur de fusion procèdera à la comparaison du temps entre chaque modalité et le temps total de la commande. Si le temps est inférieur au temps maximal défini dans l'ontologie le moteur pourra passer à la dernière étape qui est la sélection du modèle de fusion. Lorsque toutes les préconditions sont satisfaites le moteur de fusion fusionne les évènements d'entrée et envoie le résultat à l'ontologie pour fissionner et envoyer des commandes élémentaires aux actionneurs de sortie. Dans le cas où une précondition n'est pas satisfaite, le moteur interrompt le processus de fusion et envoie un message d'erreur (feedback) à l'utilisateur.

De plus, nous avons choisi de concevoir le moteur de fusion en utilisant le modèle WWHT [24] pour l'interaction multimodale. En effet, le modèle de WWHT est composé de quatre questions :

- "What" (Quoi) : Quelles sont les informations qui doivent être utilisées.
- "Which" (Quel) : Quel modèle devrions-nous utiliser pour présenter ces informations.
- "How" (Comment) : Comment déployer ces informations en utilisant les modalités.
- "Then" (Puis) : Puis comment gérer la progression de l'information générée "[55].

Dans notre cas, les réponses aux questions sont les suivantes :

- "What" : On se référe à ce que le système détecte. Ça sera les modalités d'entrée disponibles.
- "Which" : le système décidera quel modèle le moteur de fusion doit utiliser pour fusionner les modalités d'entrée.
- "How" : le système va fusionner les informations des modalités d'entrée en utilisant le modèle de fusion choisi parmi les modèles prédéfinis et stockés dans l'ontologie. Lorsque tous les composants d'une commande détectée en entrée correspondent aux composants d'un modèle de fusion dans l'ordre, ce dernier sera sélectionné comme étant le modèle de fusion à utiliser par le moteur de fusion.
- "then" : le système va sélectionner la solution de fission appropriée et envoyer le résultat de la fusion associée à la solution de fission correspondante.

2.3.4 Modélisation du moteur de Fission

Le moteur de fission est la partie responsable de la subdivision du résultat du moteur de fusion en informations élémentaires pour les envoyer aux modalités de sortie disponibles tout en prenant en compte l'état du contexte. Nous avons défini un algorithme de fission qui est présenté sur la Figure 2.3. La fission sera possible si toutes les conditions de l'algorithme de fisson sont satisfaites. Dans le cas contraire, un message d'information sera envoyé à l'utilisateur et le processus de fission sera arrêté.

La subdivision est faite selon des solutions prédéfinies dans l'ontologie. Les commandes de base seront envoyées aux modalités de sortie en fonction de leur disponibilité. Des motifs de fission sont définis dans l'ontologie en deux parties, à savoir un problème et une solution. Le problème est la commande émise par l'utilisateur qui est reconnue par le moteur de fusion alors que la solution se compose de toutes les sous-tâches possibles pour cette commande. Nous avons synthétisé les solutions en définissant neuf solutions de fission possibles qui seront présentées en détail dans le chapitre 3.

Ces solutions seront composées d'actions de sortie qui seront organisées dans un ordre précis. Etant donné que l'ordre est très important (le sens d'une action peut changer si l'ordre n'est pas respecté), nous avons défini des propriétés d'objets qui permettront de définir l'ordre des actions dans la solution proposée. Les solutions de fission ainsi que les propriétés d'objets associées à chaque solution seront présentées en détail dans le chapitre 3.

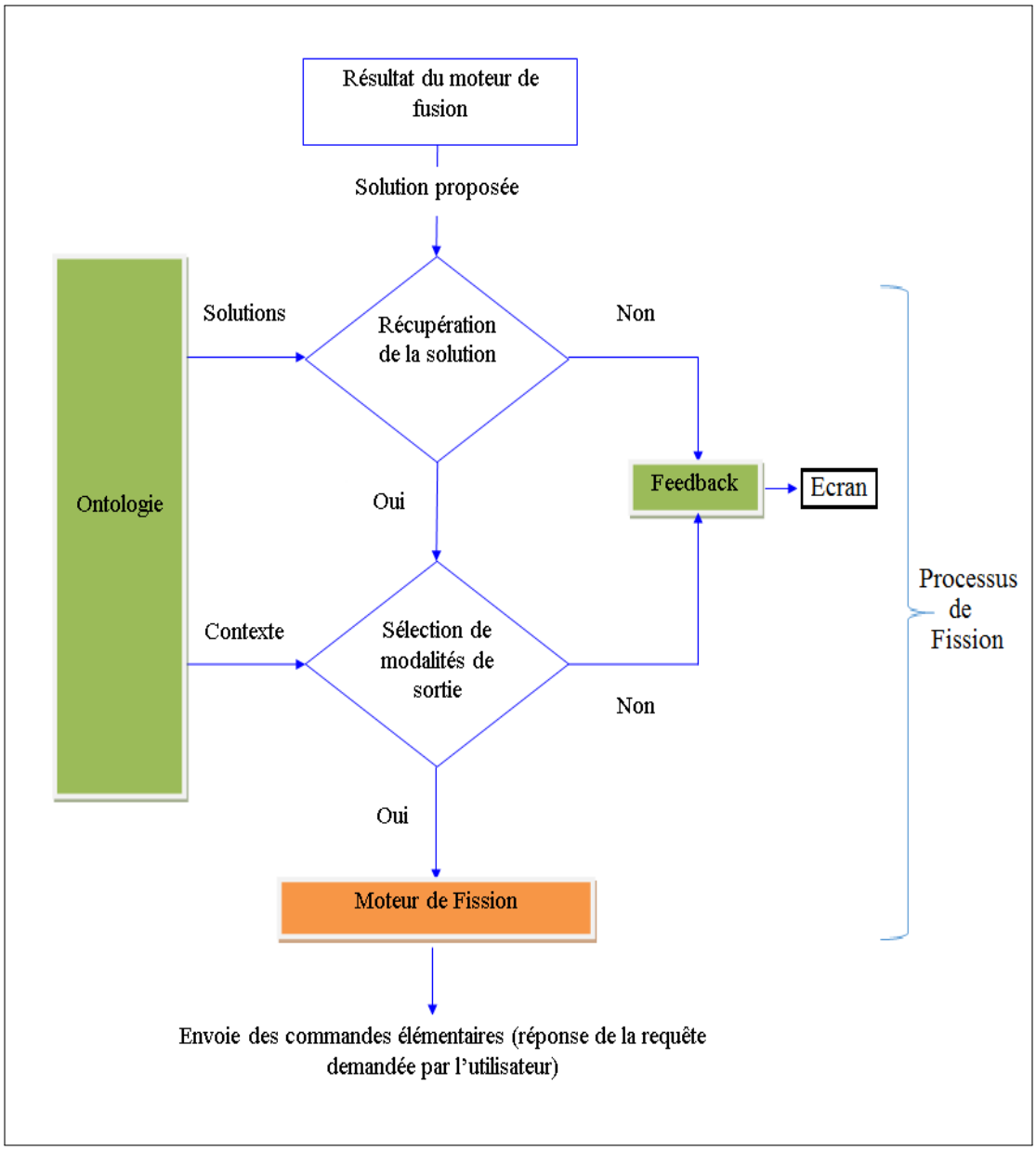


FIGURE 2.3: Algorithme du moteur de Fission

La Figure 2.3 représente l'algorithme de fission utilisé par notre système. Lorsque les

conditions de fusion ont été respectées et la fusion a eu lieu, le moteur de fission reçoit en entrée les évènements détectés qui ont été fusionnés. Le moteur de fusion fournira le numéro de la solution de fission correspondante au résultat de la fusion. Donc le moteur de fission récupéra cette solution de l'ontologie et l'utilisera pour la subdivision de l'information en actions élémentaires. Puis, une vérification des modalités de sortie doit être exécutée et les modalités de sortie disponibles seront sélectionnées. Lorsque ces étapes sont satisfaites le moteur de fission procèdera à la décomposition de l'information en actions élémentaires qu'il enverra par la suite aux modalités de sortie disponibles et ainsi répondra à la requête de l'utilisateur. Dans le cas contraire, si une étape n'est pas satisfaite, le moteur de fission arrête le processus et un message d'information sera envoyé à l'utilisateur.

Comme pour le moteur de fusion le modèle "WWHT" sera utilisé pour le moteur de fission. Cela permettra au moteur de fission de subdiviser le résultat du moteur de fusion en répondant aux questions "What", "Which", "How" et "Then" vue dans la section précédente, et qui sont :

- "What" : se réfère à ce que le système détecte. Ça sera le résultat du moteur de fusion.

- "Quel" : le système décidera quelle solution le moteur de fission devrait utiliser pour subdiviser le résultat de la fusion.

- "How" : le système subdivise le résultat de la fusion en utilisant la solution de fission choisie parmi les solutions prédéfinies et stockées dans l'ontologie.

- "Then" : le système sélectionne les modalités de sortie appropriées en fonction des informations du contexte, par exemple : si l'utilisateur est sourd, alors la modalité sonore sera désactivée et le système utilisera une autre modalité de sortie.

2.4 Conclusion

Nous avons présenté dans ce chapitre la méthodologie de recherche que nous avons utilisée lors de la conception de l'architecture multimodale proposée. Nous avons également présenté les étapes de la modélisation des différents composants de l'architecture. Cette conception étape par étape nous a permis de construire l'architecture en suivant un enchainement logique et précis de chaque étape. Ce chapitre nous conduit à la présentation de l'architecture et du résultat obtenu dans les chapitres suivants.

Chapitre 3

Architecture du système

3.1 Introduction

Nous voulons par le travail présenté dans cette thèse renforcer l'interaction multimodale dans le domaine de la robotique médicale et façiliter l'interaction entre la personne dépendante et son moyen essentiel de vie qui est le fauteuil roulant. Après avoir synthétisé la revue de littérature et établi la méthodologie de travail, nous pouvons maintenant concevoir notre architecture dans le but de fournir un système complet capable de comprendre, fusionner et fissionner des informations récupérées de l'environnement et faciliter l'interaction homme-machine. L'application présentée dans ce mémoire est la commande d'un fauteuil roulant muni d'un bras manipulateur dédié aux personnes dépendantes. L'utilisateur du fauteuil roulant pourra communiquer avec son fauteuil et le bras pour la requête de service comme se déplacer ou apporter un objet. Nous allons présenter dans ce chapitre un exemple d'application choisi pour notre système. L'architecture proposée sera modélisée en utilisant l'outil Protégé en suivant les étapes de la modélisation présentées dans le Chapitre 2.

Le cerveau humain est un organe très complexe capable de comprendre, de décider, de gérer ainsi que de stocker des informations. En effet, dès sa naissance, le cerveau humain commence son apprentissage par la compréhension d'évènements en utilisant l'aspect cause-effet et stocke tous les résultats pour des utilisations futures. Dans le domaine de l'interaction homme–machine, les chercheurs essayent de copier le fonctionnement

du cerveau humain dans le but de créer des machines autonomes capables de comprendre et de décider. Pour ce faire il est impératif que la machine puisse stocker des informations relatives à l'environnement d'évolution du system et assurer la réutilisation de ces données. Pour ce faire, le concept d'ontologie est le plus adéquat. En effet, l'ontologie permet de définir l'environnement d'évolution du système en détail. Cette définition est représentée par des concepts de base et des relations entre eux. Les concepts permettent la définition complète du domaine, et sont représentés par un graphe dont les relations sont de type hiérarchique et sémantique.

3.2 Architecture proposée

3.2.1 Vue générale

Le fauteuil roulant muni d'un bras manipulateur est utilisé par des personnes handicapées ou dépendantes pour interagir avec leur environnement et obtenir des services. L'utilisateur du fauteuil roulant peut utiliser les entrées multimodales, comme la parole, le geste, etc. pour demander des services. L'architecture proposée dans ce travail, présentée dans la Figure 3.1, est un système complet qui détecte les entrées de l'environnement en utilisant des capteurs, comprend et fusionne toutes les données en utilisant le moteur de fusion. Ensuite, le système subdivise les résultats de la fusion en utilisant le moteur de fission et envoie des tâches uni-modales à des actionneurs de sortie ou des gadgets comme le bras manipulateur ou un écran pour répondre à la demande de l'utilisateur.

Les moteurs de fusion et de fission proposés sont basés sur le mécanisme de préconditions. Les conditions préalablement définies dans l'ontologie sont satisfaites en contrôlant la sémantique du vocabulaire, du modèle et de l'aspect temporel en retournant continuellement à l'ontologie qui est la base de connaissance du système multimodal. Notre ontologie sera la base de toutes les déclarations et les informations récupérées de l'environnement. La Figure 3.1 montre une vue générale de notre architecture (le fauteuil roulant représenté sur la Figure 3.1 est le Arlyn ChairBot [75]). L'architecture se compose de quatre parties : l'entrée (Input), l'architecture du système multimodal

(MSA), la base de connaissances (Ontologie) et la sortie (Output).

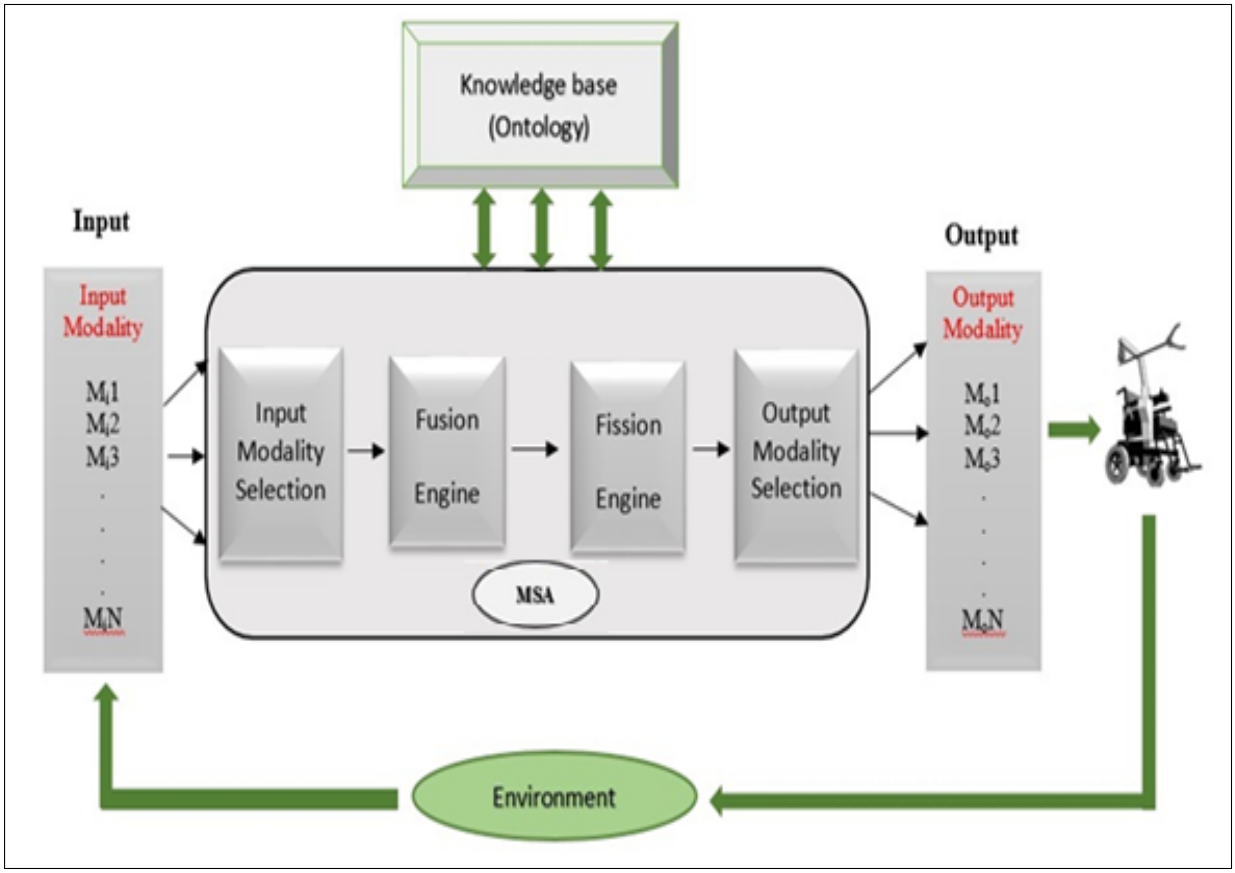


FIGURE 3.1: Architecture du système multimodale

Comme présenté sur la Figure 3.1, la partie entrée (Input) se compose de modalités d'entrée provenant de l'environnement ainsi que les informations contextuelles sémantiques. Cette partie sera expliquée en détail dans le Chapitre 3. Le terme environnement représente l'emplacement physique et toutes les entités présentes dans l'environnement de l'utilisateur. Ces événements seront envoyés à la partie MSA (Multimodal System Architecture). La partie MSA est une partie essentielle car le processus de fusion et de fission a lieu dans cette partie. La MSA est composée de quatre parties : Sélection de modalités d'entrée et de sortie, moteur de fusion et moteur de fission. Les deux parties de sélection de modalités décideront quelle modalité est acceptée selon l'état dynamique du contexte.

Par exemple, si l'environnement est trop bruyant, le système ne sera pas en mesure de détecter une commande vocale ou d'envoyer une réponse vocale à l'utilisateur. Donc

la partie de sélection de modalité désactive toutes les modalités liées au bruit tels que le capteur de voix ou le haut-parleur. Ensuite, en retournant continuellement à la base de connaissances, la MSA fusionne les informations obtenues à partir de l'environnement en utilisant le moteur de fusion et les subdivise en sous-tâches élémentaires en utilisant le moteur de fission puis les envoie aux modalités de sortie disponibles. Rappelons que la base de connaissances est l'ontologie qui décrit en détail tous les éléments de l'environnement et la relation entre eux. Les modèles de fusion et les motifs de fission utilisés pour la fusion et la fission sont stockés dans l'ontologie.

Finalement, la partie sortie (Output) est constituée de modalités de sortie utilisées par le système pour informer l'utilisateur sur l'état d'avancement du processus ou pour répondre à la demande de l'utilisateur.

3.2.2 Exemple d'application et scénario

L'exemple d'application choisi dans cette étude est un fauteuil roulant muni d'un bras manipulateur dédié à aider une personne dépendante. L'architecture proposée sera une architecture ambiante d'interaction homme-robot-environnement. Le mot robot désigne l'ensemble du fauteuil roulant et le bras manipulateur embarqué sur le fauteuil.

Le robot sera en mesure de fournir des services à l'utilisateur comme se déplacer ou apporter un objet. Aussi, l'utilisateur pourra interagir avec le robot en utilisant une interaction multimodale en prenant en compte l'état évolutif du contexte sémantique. Le robot aura sur chaque coin des capteurs (infrarouge et ultrasonique) qui détecteront des obstacles lors du déplacement du fauteuil. Aussi, on aura sur le fauteuil un écran tactile qui servira de moyen de communication entre le robot et l'utilisateur.

Lors de la modélisation de l'architecture, en plus des informations relatives à l'interaction avec le fauteuil (comme la parole, le geste. etc.), il est aussi impératif de prendre en compte les informations relatives au déplacement du fauteuil. Pour ce faire, la détection d'obstacle sera envoyée sous forme d'alarme qui arrêtera le fauteuil. Pour ce qui est de la batterie du système, une vérification continue sera faite et une alarme informant l'utilisateur d'un niveau bas de batterie sera utilisée.

Le scenario choisi pour l'étude de cas pour le système d'interaction multimodale proposé est la requête "Donne lui un verre de jus". On suppose que l'utilisateur a des problèmes auditifs mais n'est pas totalement sourd. La commande choisi pour l'exemple de validation combine la modalité "parole" ainsi que le "geste" en désignant la personne qui recevra le verre de jus.

3.2.3 Conception de l'architecture

La conception de l'architecture de commande d'interaction multimodale sera faite en utilisant le logiciel Protégé. Ce dernier est un logiciel open source qui est composé de plusieurs plugins qui permettent l'intégration des différents composants d'une ontologie. Comme présenté précédemment, nous avons choisi d'utiliser le concept d'ontologie pour la définition des différentes parties de l'architecture. Notre approche est la modélisation d'une architecture qui utilise une ontologie comme base de connaissances pour les moteurs de fusion et de fission. L'ontologie utilisée décrit l'environnement changeant du système multimodal en définissant des scénarios et tous les éléments qui composent l'environnement.

L'utilisation de l'ontologie assure "l'ouverture" du système en prenant en compte un nombre important de modalités d'entrée et de sortie, et aussi la "régularité" lors de la description de la composition de l'environnement en définissant les éléments et les scénarios les plus adéquats, et enfin la "flexibilité" qui permet l'ajout ou le retrait d'entités en fonction du profil de l'utilisateur ou du domaine d'utilisation. Bien que notre système soit appliqué sur un fauteuil roulant muni d'un bras manipulateur, dans le présent document, en ajustant les éléments de l'ontologie, nous pouvons l'appliquer à tout autre système d'interaction homme-machine. Cette caractéristique assure que notre architecture est un système adaptable aux circonstances de l'utilisateur dans une multitude de domaines et d'aspects.

3.3 Pourquoi l'ontologie ?

Nous allons décrire dans ce chapitre l'ontologie du système qui va nous permettre de modéliser l'environnement d'évolution du système robotique. Notre système est un fauteuil roulant muni d'un bras manipulateur dédié aux personnes à mobilité réduite. Notre travail consiste à proposer une architecture ambiante d'interaction homme-robot-environnement, dans le cadre de la robotique d'aide à la personne dépendante. Cette architecture permettra aux robots « Ubiquitous Networked Robots » de prendre en compte le contexte évolutif pour fournir continuellement des services à une personne à mobilité réduite. Le terme "environnement" utilisé dans notre étude représente toutes les entités qui forment le domaine d'évolution du système robotique et seront définies dans notre ontologie qui sera la base de connaissance de l'architecture globale. Nous avons choisi d'utiliser les ontologies car elles permettent :

- Une facilité de communication entre la personne et le système et entre le système lui-même.
- Un accès facile à l'information et sa réutilisation.
- La prise en compte d'un grand nombre d'éléments qui forment l'environnement comme les personnes, les objets, le contexte, etc., ce qui rend l'architecture plus flexible et permet de prendre en considération différents domaine d'application.
- De définir les relations entre les différents composants d'un environnement, en utilisant le concept de propriétés.
- La possibilité d'intégration de règles qui doivent être prises en compte lors du processus de fusion des données. Ces règles seront stockées dans l'ontologie et écrites en langage du web sémantique. Les langages SWRL (pour les règles) et SQWRL (pour les requêtes) sont basés sur les normes du W3C.

Nous utilisons l'ontologie pour définir l'environnement de l'utilisateur du fauteuil roulant muni d'un bras manipulateur. Ce concept permet la définition d'un grand nombre d'éléments lors de la description de l'environnement d'évolution du système robotique.

Aussi, le concept d'ontologie permet la définition de règles utilisées lors de la sélection de modalité et des processus de fusion et de fission. Nous utilisons l'outil open source PROTEGE (www.protege.stanford.edu) [69] qui permet la conception d'ontologies et l'intégration de règles. Dans notre travail, l'environnement du système robotique (le fauteuil roulant muni d'un bras manipulateur) est également l'environnement de l'utilisateur. L'environnement du système est l'intérieur de la maison de l'utilisateur et tous les éléments qui le composent, ainsi que l'extérieur qui peut être un jardin ou le quartier. L'ontologie représentée par un graphe hiérarchique, va nous permettre de définir en détail tous les éléments qui composent l'environnement, d'établir les relations entre eux, et d'obtenir un raisonnement basé sur des informations sémantiques.

3.4 Le concept d'ontologie

Nous avons choisi d'utiliser le concept d'ontologie qui sera conçue en utilisant l'outil open source PROTEGE [69]. Ce dernier permet de construire et de définir tous les concepts qui permettent de former une ontologie.

3.4.1 Composants d'une Ontologie

Une ontologie est une représentation hiérarchique de plusieurs éléments qui forment le cas d'étude. La définition des composants d'un environnement d'étude se fait en utilisant les composants de l'ontologie suivants :

- **Les classes** : représentent un ensemble d'individus d'un domaine spécifique. Les classes peuvent être organisées dans des hiérarchies contenant des superclasses et des sous-classes. Par exemple la classe "Alarme" peut avoir comme sous classe "Alarme de santé" qui aura comme instance le nom de l'alarme (ou son modèle).
- **Individus (instances)** : représentent les entités du domaine qui nous intéresse.
- **Propriétés** : représentent les relations entre les classes et les individus ou entre les classes ou les individus eux-mêmes. Ces relations permettent d'établir les liens entre les différents membres de l'ontologie. Nous avons trois types de propriétés :

1. **Propriété d'objet** : lie les membres de l'ontologie aux autres membres.
2. **Propriété de type de données** : lie une instance à une valeur de type de donnée.
3. **Propriétés d'annotation** : Peuvent être utilisées pour ajouter des informations (métadonnées) à des classes, des individus ou à des propriétés d'objet ou de type de données.

3.4.2 Caractéristiques des propriétés

Le langage OWL permet d'enrichir le sens des propriétés en utilisant des caractéristiques pour chaque propriété.

1. **Propriété fonctionnelle :** Une propriété est dite fonctionnelle si et seulement si cette propriété relie au maximum un individu via cette propriété. Par exemple, si l'individu "A" est relié par une propriété fonctionnelle "has mother" (a pour mère) à l'individu "B" et si ce même individu "A" est relié par la même propriété à l'individu "C". Nous pouvons conclure que l'individu "B" et "C" sont la même personne (avec deux prénoms différents), Car une personne ne peut avoir au maximum qu'une seule mère biologique.
2. **Propriété transitive :** Si une propriété qui relie l'individu "A" à l'individu "B" est dite transitive, et que l'individu "B" est relié à l'individu "C" par la même propriété, on peut conclure que l'individu "A" est relié à l'individu "C" par la même propriété. Par exemple, si l'individu "A" est relié à l'individu "B" par la propriété "has employer" (a pour employeur) l'individu "B", et que "B" est relié par la même propriété à l'individu "C" et sachant que cette propriété est transitive, nous pouvons conclure que "C" est l'employeur de "B" et de "A".
3. **Propriété symétrique :** Si une propriété qui relie l'individu "A" à l'individu "B" est symétrique, on conclut que l'individu "B" est relié à l'individu "A" par la même propriété. Par exemple, si l'individu "A" est relié à l'individu "B" par la propriété symétrique "has colleague" (a comme collègue) l'individu "B", on conclut que "B"

est relié par la même propriété à "A" (si "A" est le collègue de "B", alors "B" est aussi le collègue de "A").

3.5 Description de l'environnement

Nous avons choisi de définir l'environnement de l'utilisateur comme étant l'environnement d'évolution du système robotique. L'environnement sera l'intérieur de la maison de l'utilisateur ainsi que tous les éléments qui la composent, et l'extérieur de la maison qui peut être le jardin ou le quartier.

En premier, nous allons définir les classes et leurs sous-classes, puis nous allons les peupler par des individus, qui peuvent être des personnes, des objets etc. Finalement, nous allons utiliser les propriétés d'objets (pour relier deux objets) et les propriétés de données (pour relier un objet avec une valeur) pour donner un sens à l'ontologie. De plus, nous allons utiliser des relations sémantiques qui vont permettre au robot de comprendre des actions qui sont évidentes pour les personnes. Par exemple, nous allons définir une relation entre les boissons liquides et les objets utilisés pour les liquides. Cette relation permettra au robot de comprendre que chaque liquide a besoin d'un récipient et que lorsque l'utilisateur demande une boisson il est impératif d'utiliser un objet dédié aux liquides (comme un verre ou une bouteille).

3.5.1 Les classes

Les classes qui forment notre ontologie sont les éléments qui composent l'environnement d'évolution du système de commande ainsi que toutes les informations relatives au fonctionnement du système, comme par exemple les modèles de fusion, les motifs de fission, le temps... etc. Nous allons présenter dans ce qui suit les différentes classes, leurs instances et les propriétés qui permettent de donner un sens à l'ontologie. Les instances (ou individus) sont les composants de base d'une ontologie. Ces éléments sont les derniers d'une chaine d'éléments définis sous forme d'une représentation hiérarchique qui commence par les classes et finit par les instances. Ces derniers peuvent être le nom

d'objets, de personnes ou même des mots utilisés sous forme de vocabulaire.

1. **Thing** : Cette classe existe par défaut dans l'outil PROTEGE. Les symboles utilisés par Protégé pour générer les graphes sont représentés dans l'Annexe I.

2. **Event (évènement)** : Nous avons créé cette classe, représentée par la Figure 3.2, pour simuler l'arrivée d'évènements, les reconnaitre et retrouver le modèle de fusion et motif de fission qui correspond à chaque évènement. Plus de détails seront présentés dans le chapitre Processus de Fusion et de Fission et les requêtes de sélection du modèle de fusion pour chaque évènement sont présentées dans l'Annexe I.

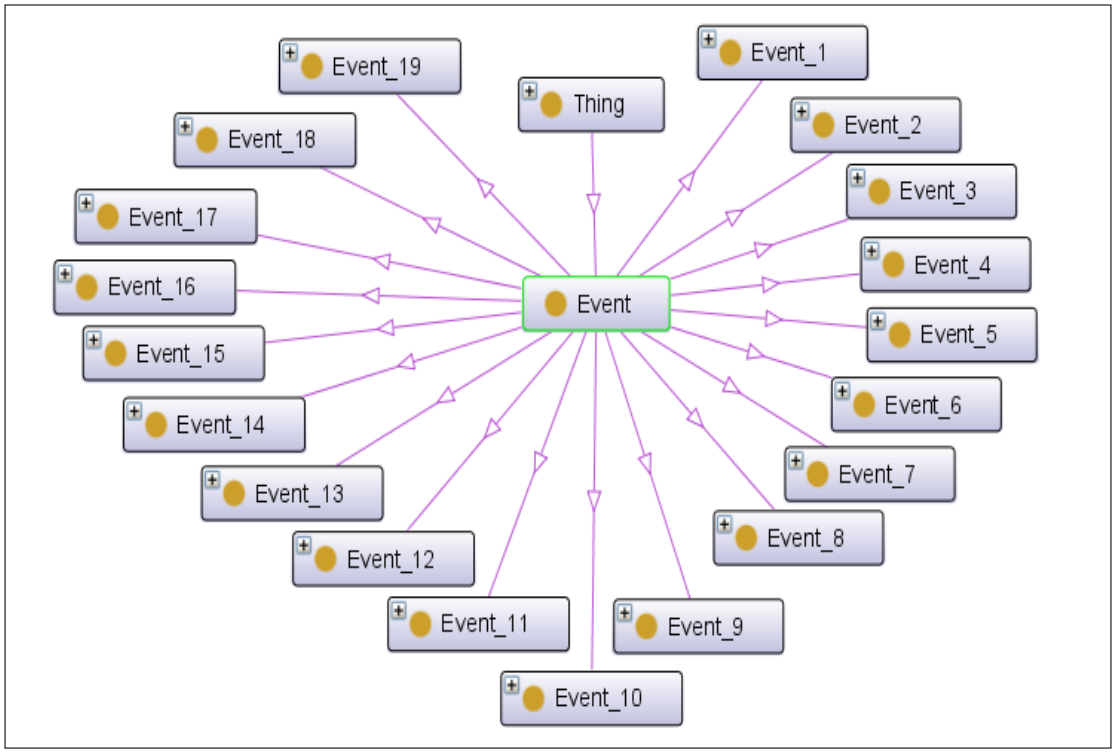


FIGURE 3.2: La classe Event(évènement)

3. **Elements de l'ontologie (Ontology-Elements)** : Cette classe, représentée par la Figure 3.3, est la classe supérieure de toutes les autres classes qui composent l'ontologie.

   3.1 **Le vocabulaire(Vocabulary) :** Désigne le vocabulaire utilisé pour commander le système. La classe "Vocabulary" et ses sous-classes sont représentées

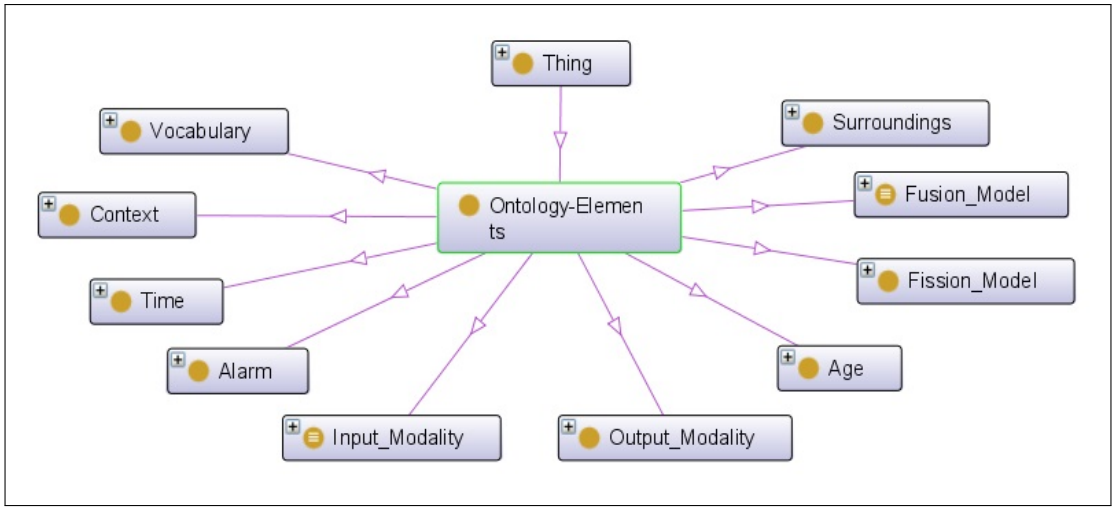


FIGURE 3.3: La classe Eléments de l'ontologie (Ontology Elements) et ses sous classes

dans la Figure 3.4.

i. **Mots pour les objets (Words For Objects)** : a pour sous classes et instances :

- **Mots utilisés pour les objets de décors (Words_Decor_Objects)** : a pour sous classes :

▷ *Mots pour les objets a partie mobile (Words used for movable P Object)* : représentée par la Figure 3.5, a pour instances

∗ Fermer (close/shut)

∗ Verrouiller(lock)

∗ Ouvrir (open)

∗ Déverrouiller (unlock)

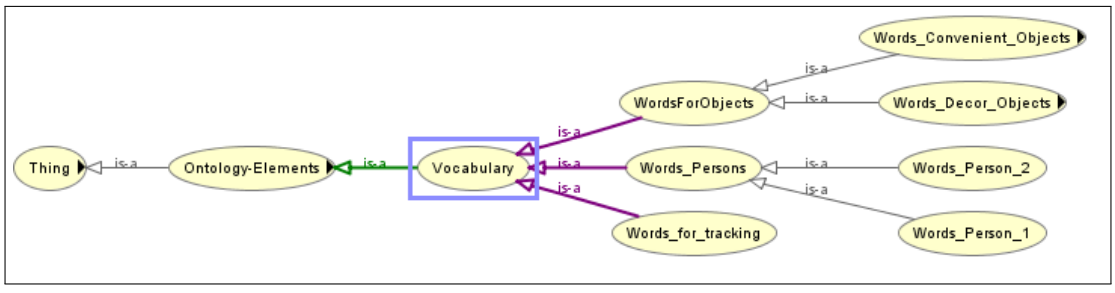


FIGURE 3.4: La classe "Vocabulary" et ses sous classes

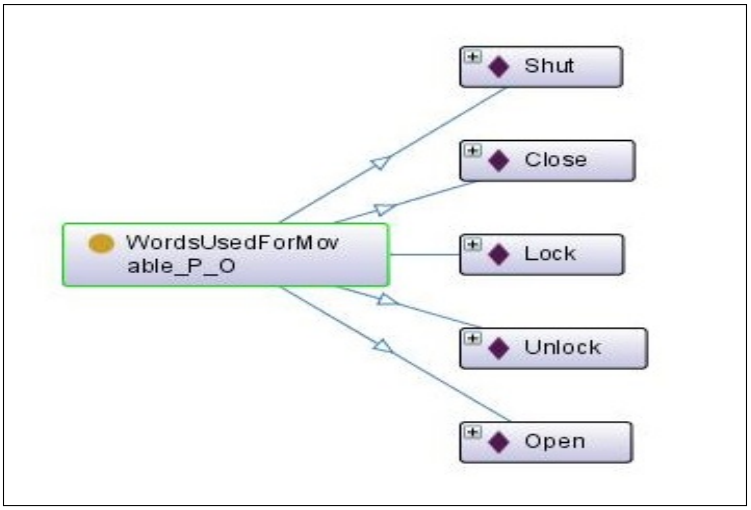


FIGURE 3.5: La classe "Mots pour les objets a partie mobile" (Words used for movable P O) et ses instances.

▷ *La classe "Mots objet électrique grand" (Words Big electrical objects)* : représentée par la Figure 3.6, a pour instances

∗ Allumer (switch on / turn on)

∗ Eteindre (switch off/ turn off)

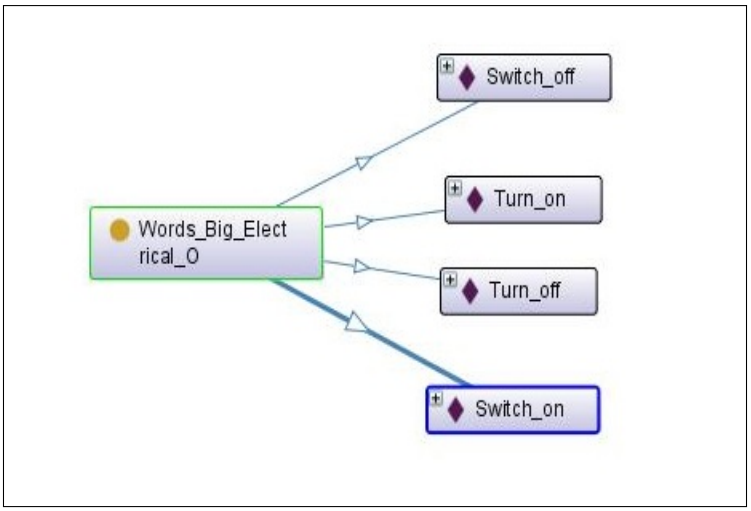


FIGURE 3.6: La classe "Mots objet électrique grand" (Words Big electrical objects) et ses instances

- **Mots utilisés pour les objets utiles (Words_Convenient_Objects)** : a pour sous classes

▷ *La classe "Mots pour meubles légers " (Words lightweight Furniture)* : représentée par la Figure 3.7, a pour instances

- Trouve (Find)
- Déplace (Move)
- Tire (Pull)
- Pousse (Push)

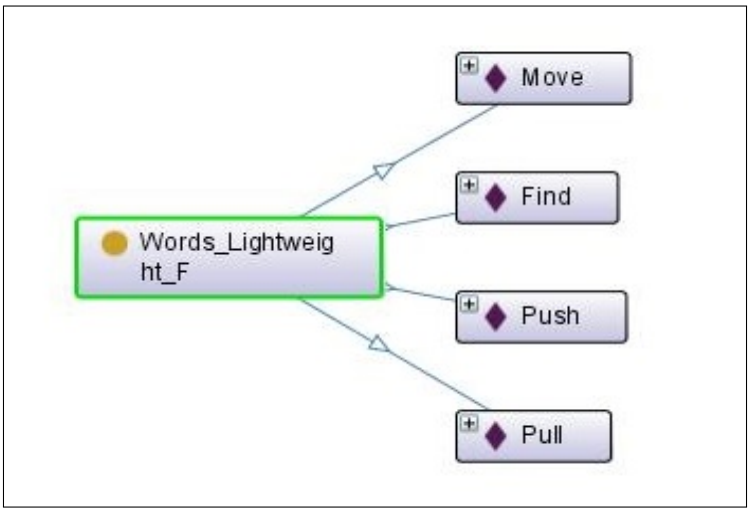


FIGURE 3.7: La classe "Mots pour meubles légers" (Words lightweight Furniture) et ses instances

- *La classe "Mots pour liquide" (Words liquids)* : représentée par la Figure 3.8, a pour instances
  - Trouve (Find)
  - Rempli (Fill)
  - Apporte (Get)
  - Verse (Pour)
  - Prend (Take)
  - Apporte (Bring)

  Sur la Figure 3.8, les propriétés d'objets "compose 15_1" et "has Next 15_1" sont aussi présentées. La propriété "compose 15_1" permet de définir la composition du modèle de fusion 15 et la propriété "has Next 15_1" permet de définir l'ordre du modèle de fusion numéro 15. Sur la Figure 3.8 la classe "Words Liquid" précède la classe "Liquid" dans l'ordre du modèle de fusion 15. Le modèle 15 de fusion est présenté en détail dans le tableau 3.1.

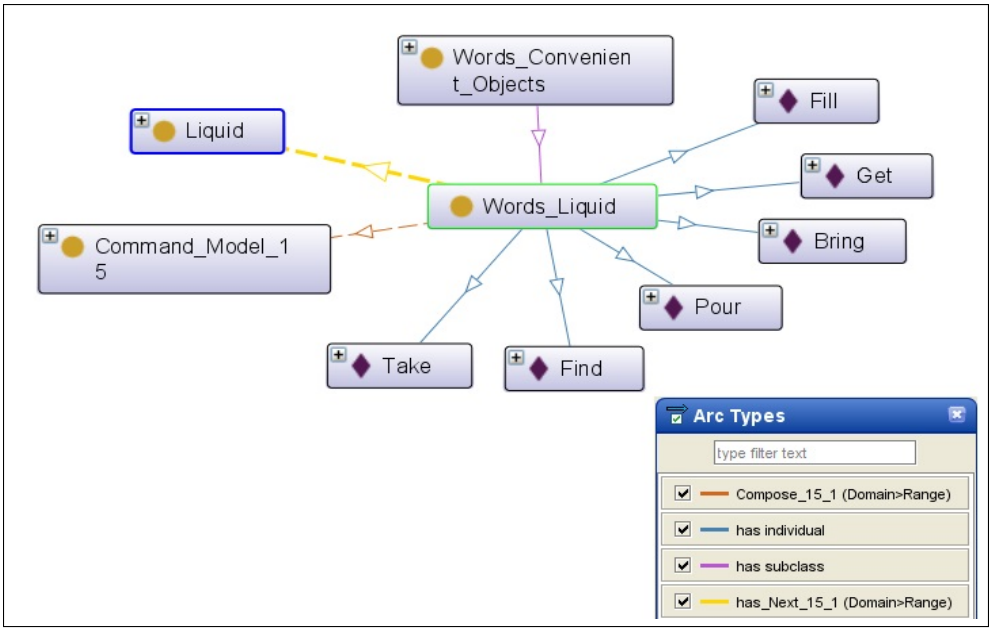


FIGURE 3.8: La classe "Mots pour liquide" (Words liquids) et ses instances

▷ *La classe "Mots objet ouvrable" (Words openable objects)* : représentée par la Figure 3.9, a pour instances

* Ouvre (open)
* Ferme (close)
* Pose (Put)

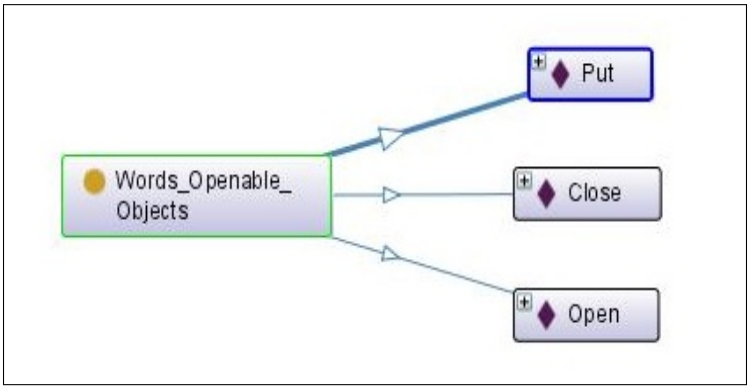


FIGURE 3.9: La classe "Mots objet ouvrable" (Words openable objects) et ses instances

▷ *La classe " Mots objet personnel" (Words personal use objects)* : représentée par la Figure 3.10, a pour instances

* Ramène (bring)
* Trouve (find)

- Apporte (get)
- Pose (put)
- Prend (Take)

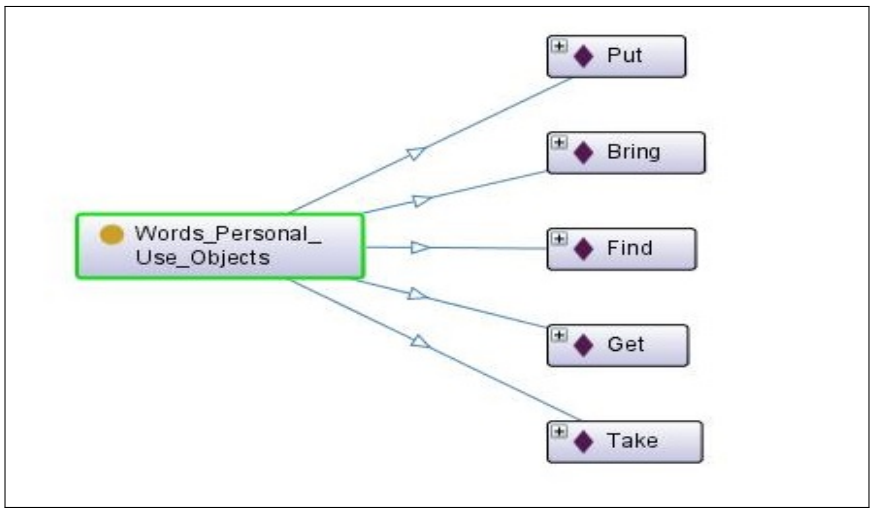


FIGURE 3.10: La classe "Mots objet personnel" (Words personal use objects) et ses instances

- *La classe " Mots petit objet électrique" (Words small electrical objects)* : représentée par la Figure 3.11, a pour instances
  - Ramène (bring)
  - Trouve (Find)
  - Apporte (Get)
  - Pose (Put)
  - Prend (Take)
  - Eteint (Switch off / Turn off)
  - Allume (Switch on / Turn on)
- *La classe " Mots petit objet utiles" (words for small useful objects)* : représentée par la Figure 3.12, a pour instances
  - Ferme (close)
  - Trouve (find)
  - Ouvre (open)

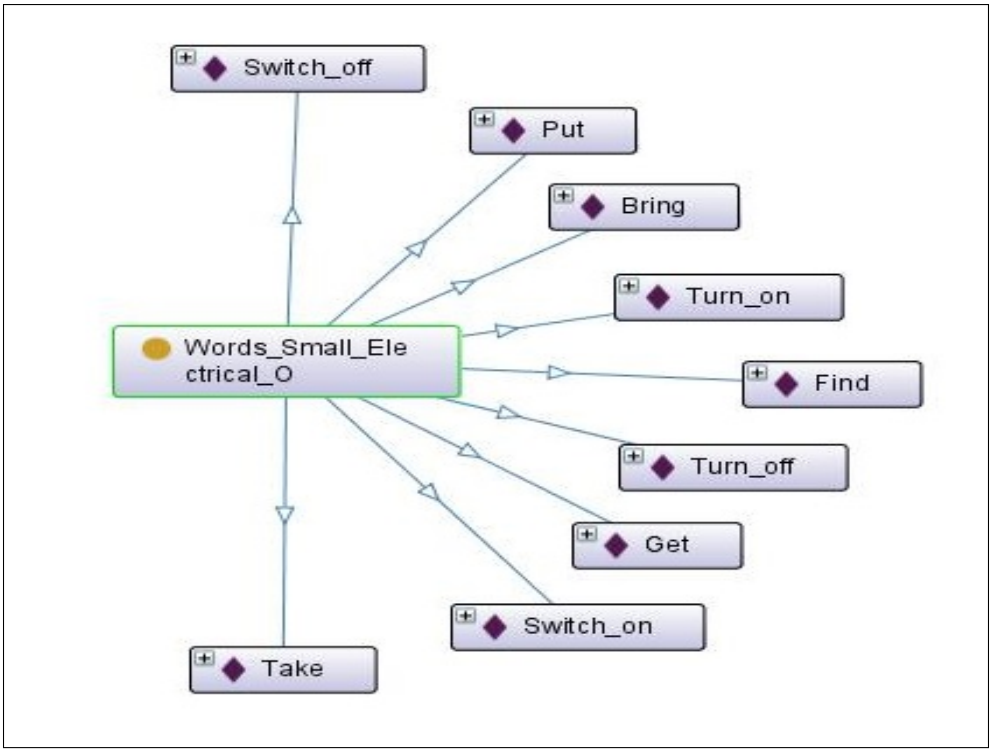


FIGURE 3.11: La classe "Mots petit objet électrique" (Words small electrical objects) et ses instances

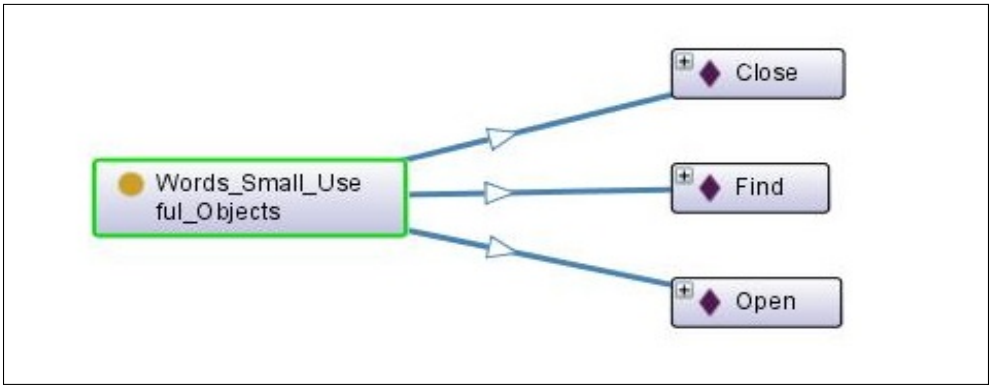


FIGURE 3.12: La classe "Mots pour petit objet utiles" (words for small useful objects) et ses instances.

ii. **Mots pour les personnes (Words For Persons)** : représentée par la Figure 3.13, a pour sous classes et instances :

- *Words_Person_1* : a pour instances :
  - ▷ Répondre (answer)
  - ▷ Appeler (call)
  - ▷ Contacter (contact)

- ▷ Trouver (find)
- ▷ Localiser (Locate)
- ▷ Cherche (search)

- *Words_Person_2* : a pour instances :

- ▷ Donner (give)
- ▷ Apporte (bring)

iii. **Mots pour le suivi(Words For tracking)** : représentée par la Figure 3.14, a pour instances :

- Retourner (return)
- Guide (guide)
- Emmène (take)

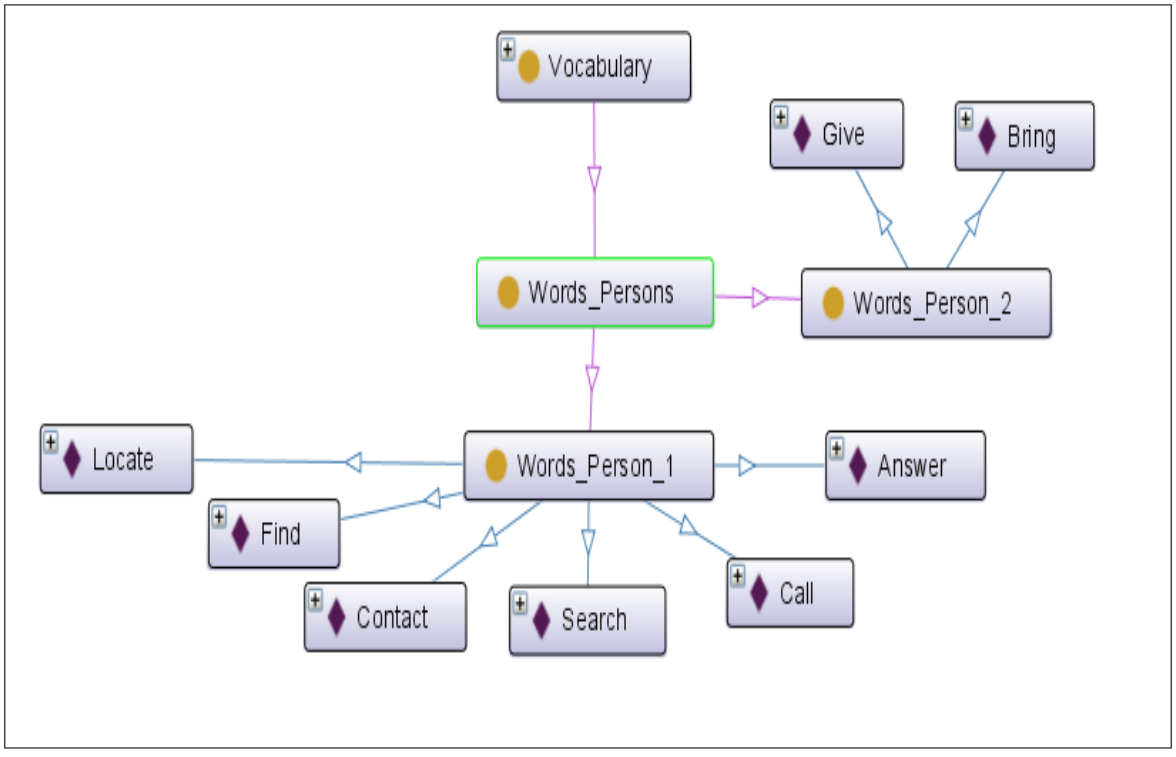


FIGURE 3.13: La classe "Mots pour les personnes" (Words For Persons) et ses instances

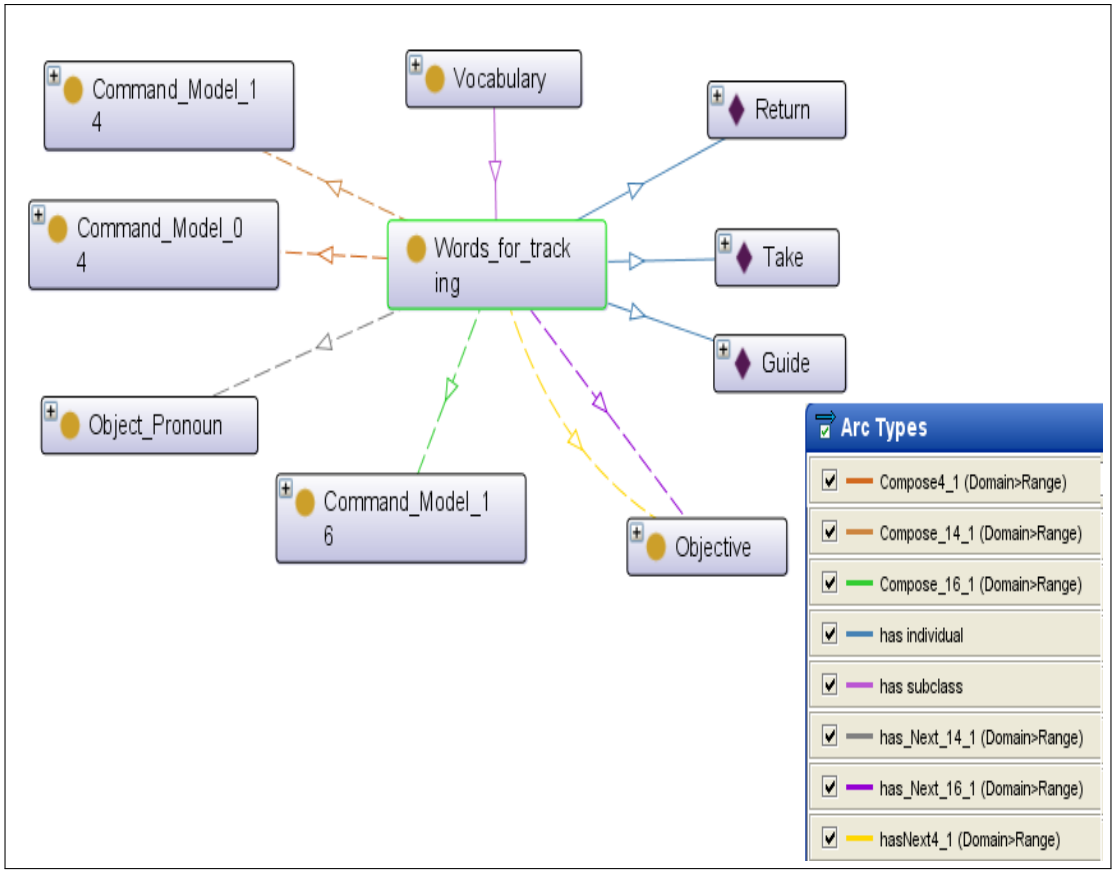


FIGURE 3.14: La classe "Mots pour le suivi" (Words For tracking) et ses instances

Sur la Figure 3.14, nous trouvons les propriétés "compose 04_1", "compose 14_1" et "compose 16_1". Ces propriétés permettent de former les modèles de fusion 04, 14 et 16. Ce qui signifie que la classe "Words for tracking" fait partie des modèles de fusion 04, 14 et 16. Les propriétés "has Next04_1", "has Next 14_1" et "has Next 16_1" permettent de donner l'ordre au seins des modèles de fusion correspondants.

3.2 **Temps (time) :** Le temps maximal pour une commande et le temps maximal entre deux modalités successives sont définies comme individus de la classe "Time" et sont présentés sur la Figure 3.15.

- Temps de commande (command time) : aura comme valeur le temps maximum d'une commande.
- Temps de la modalité (modality time) : aura comme valeur le temps maximum entre deux modalités successives.

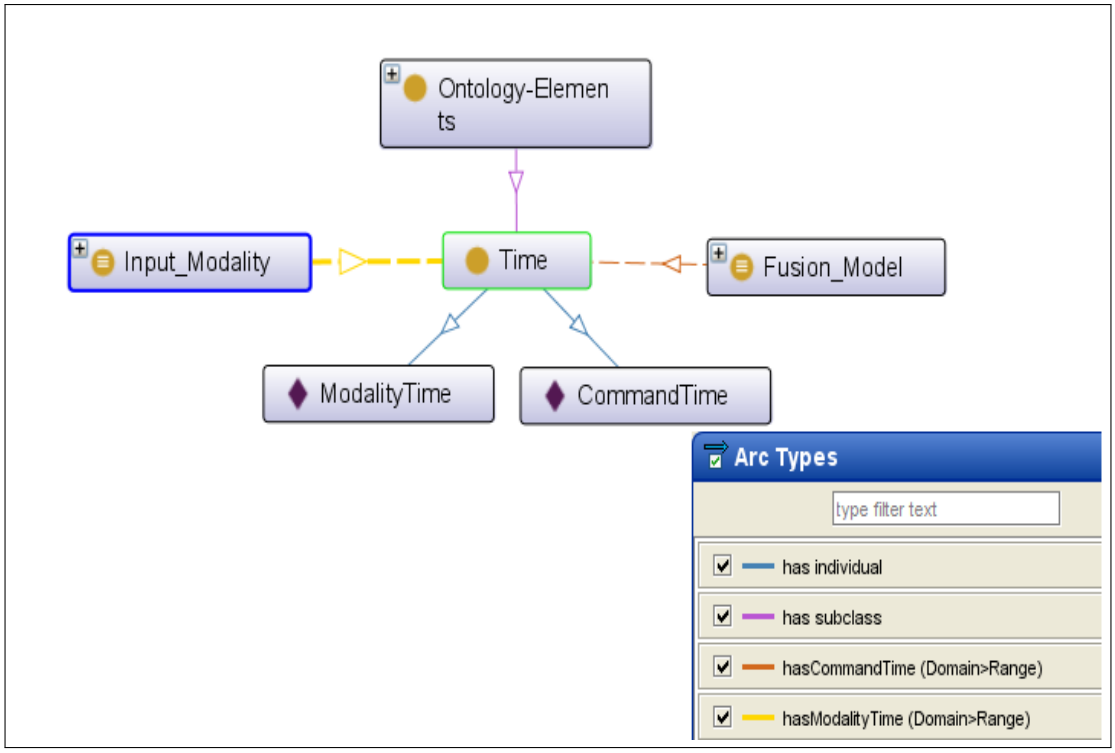


FIGURE 3.15: La classe "Time"

Les propriétés "has Modality time" et "has Commande Time" permettent, respectivement, de relier les classes "Input modality" et "Fusion model" à la classe "Time". Cela va permettre de définir le temps maximum accepté entre l'arrivée de deux modalités successives et le temps maximum d'une commande. Ce qui permet au système de gérer le temps d'exécution et ne pas attendre indéfiniment une modalité ou une commande.

3.3 **Contexte (Context)** : Cette classe, représentée par la Figure 3.16, prend en compte le contexte global du système. Ses sous classes sont :

i. **Contexte de l'environnement (Environment Context)** : Cette sous-classe, représentée par la Figure 3.17, prend en considération les éléments qui forment l'environnement. Ses sous classes sont :

- Niveau de luminosité (Lighting Level) : Information obtenue par le capteur de luminosité. Elle a pour instance une valeur mesurée par un
  - ▷ Capteur de niveau de luminosité (Lighting_level_Sensor)
- Niveau de bruit (Noise Level) : Information obtenue par le capteur de

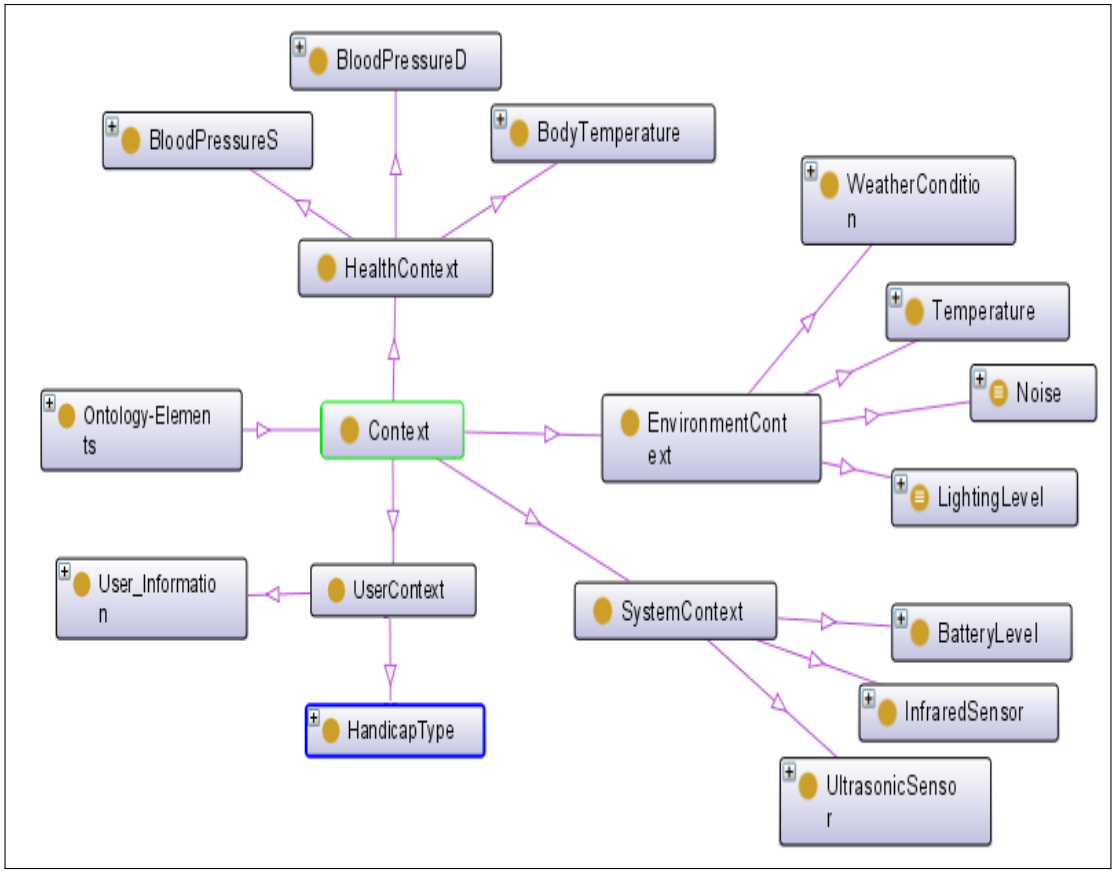


FIGURE 3.16: La classe “Context” et ses sous classes

bruit. Elle a pour instance une valeur mesurée par un

▷ Capteur de niveau de bruit (Noise_Level_Sensor)

- Température ambiante (Temperature) : Information obtenue par le thermomètre. Elle a pour instance une valeur mesurée par un

  ▷ Thermomètre (Thermometer)

- Conditions météorologique (Weather condition) : Information obtenue par internet. Elle a pour instances des valeurs obtenues de capteurs de

  ▷ Pluie (Rain)

  ▷ Neige (Snow)

  ▷ Soleil (Sun)

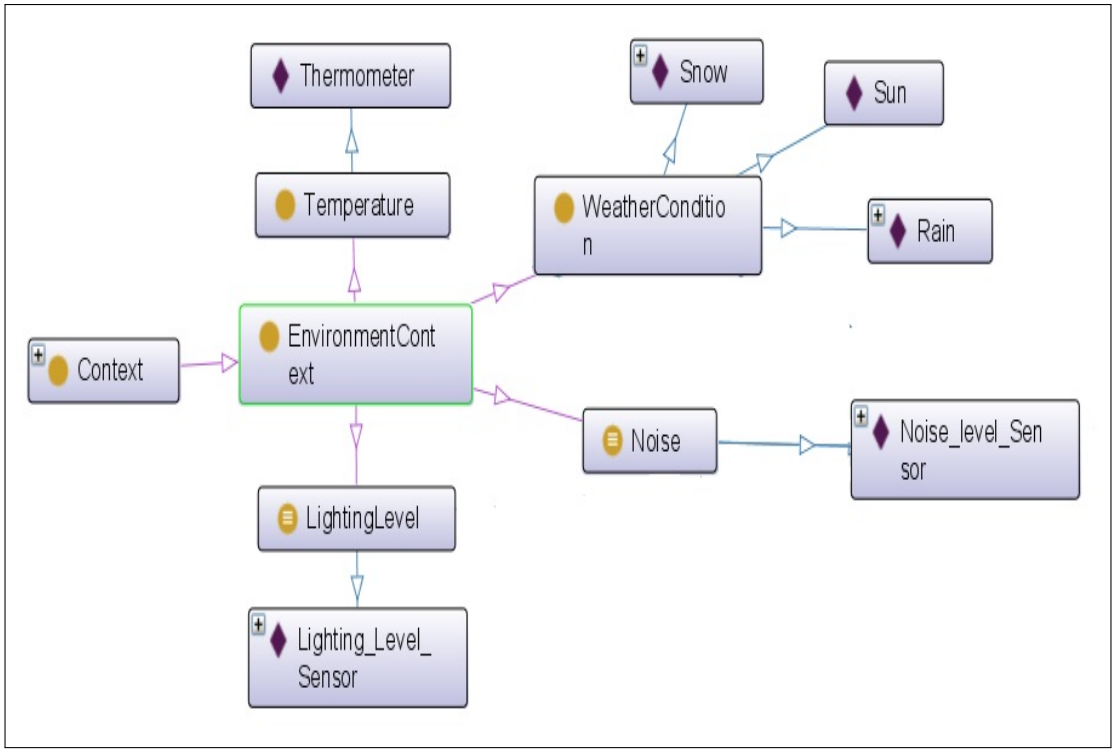


FIGURE 3.17: Les sous classes de la classe "Contexte environnemental" (Environmental Context) et leurs instances

ii. **Contexte de santé (Health Context)** : Cette sous classe,représentée par la Figure 3.18, permet de surveiller l'état de santé de l'utilisateur. Ses sous classes sont :

- Pression artérielle systolique (Blood Pressure S) : a pour instances les valeurs obtenues à partir de capteur de la
  - ▷ Pression systolique (Systolic Pressur)
- Pression artérielle diastolique (Blood Pressure D) : a pour instances les valeurs obtenues à partir de capteur de la
  - ▷ Pression diastolique (Diastolic Pressur)
- Température corporelle (Body Temperature) : a pour instances une valeur obtenue à partir d'un
  - ▷ Thermomètre médical (Medical Thermometer)

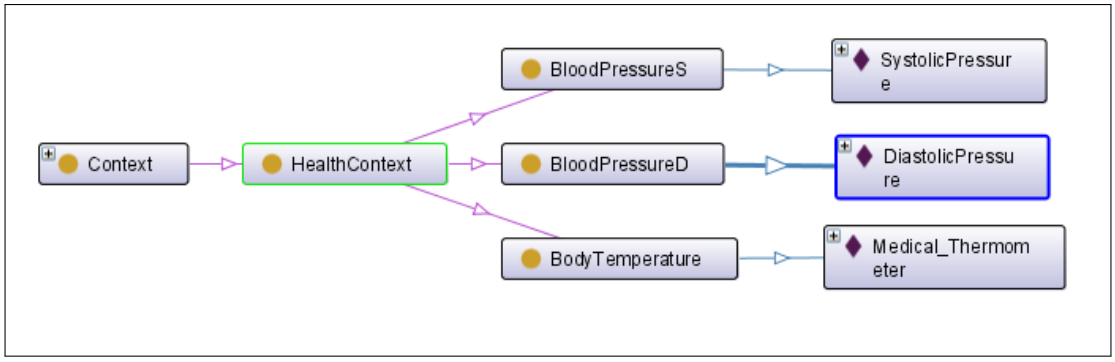


FIGURE 3.18: Les sous classes de la classe "Contexte de santé" (Health context) et leurs instances

iii. **Contexte du système (System Context)** : Cette classe, représentée par la Figure 3.19, permet de prendre en considération les informations issues des différents capteurs embarqués sur le système :

- Niveau de la batterie (Battery Level) obtenu du capteur de batterie. L'instance est la valeur mesurée par un capteur de niveau de
  - ▷ Batterie (battery)
- Information issue du capteur infrarouge (Infrared sensor) obtenue des capteurs embarqués sur le fauteuil. Les instances sont :
  - ▷ Capteur infrarouge 1 (Infrared sensor 1)
  - ▷ Capteur infrarouge 2 (Infrared sensor 2)
  - ▷ Capteur infrarouge 3 (Infrared sensor 3)
  - ▷ Capteur infrarouge 4 (Infrared sensor 4)
- Information issue du capteur ultrason (Ultrasonic Sensor) obtenue des capteurs embarqués sur le fauteuil. Les instances sont :
  - ▷ Capteur ultrasonique 1 (Ultrasonic sensor 1)
  - ▷ Capteur ultrasonique 2 (Ultrasonic sensor 2)
  - ▷ Capteur ultrasonique 3 (Ultrasonic sensor 3)
  - ▷ Capteur ultrasonique 4 (Ultrasonic sensor 4)

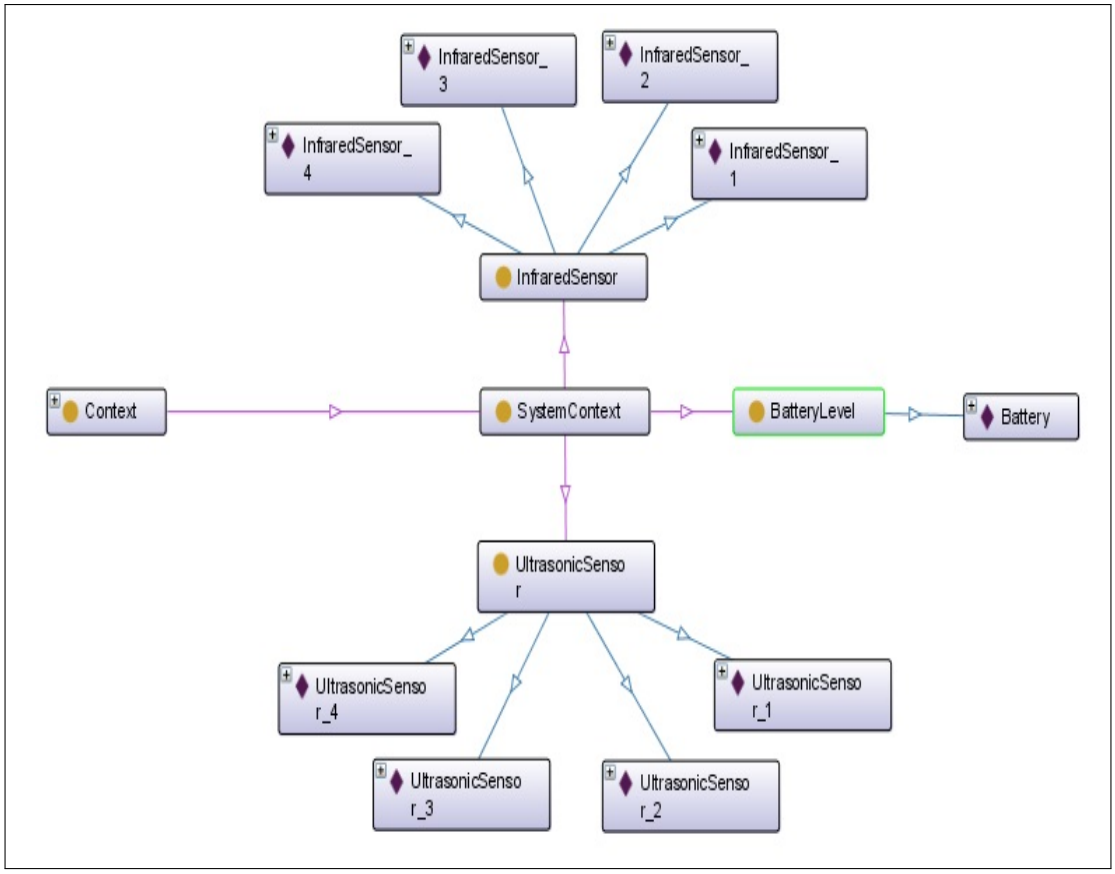


FIGURE 3.19: Les sous classes de la classe "Contexte du système" (System context) et leurs instances.

iv. **Contexte de l'utilisateur (User Context)** : Cette classe, représentée par la Figure 3.20, prend en considération les informations personnelles de l'utilisateur :

- Type d'handicap (Handicap Type) :a pour instances
  - ▷ Muet (mute)
  - ▷ Sourd (deaf)
  - ▷ Aveugle (blind)
  - ▷ Handicap manuel (manual disability)
  - ▷ Problèmes auditifs (Hearing problems)
- Nom et âge de l'utilisateur (User information) : a pour instance :
  - ▷ Utilisateur 1 (User 1)

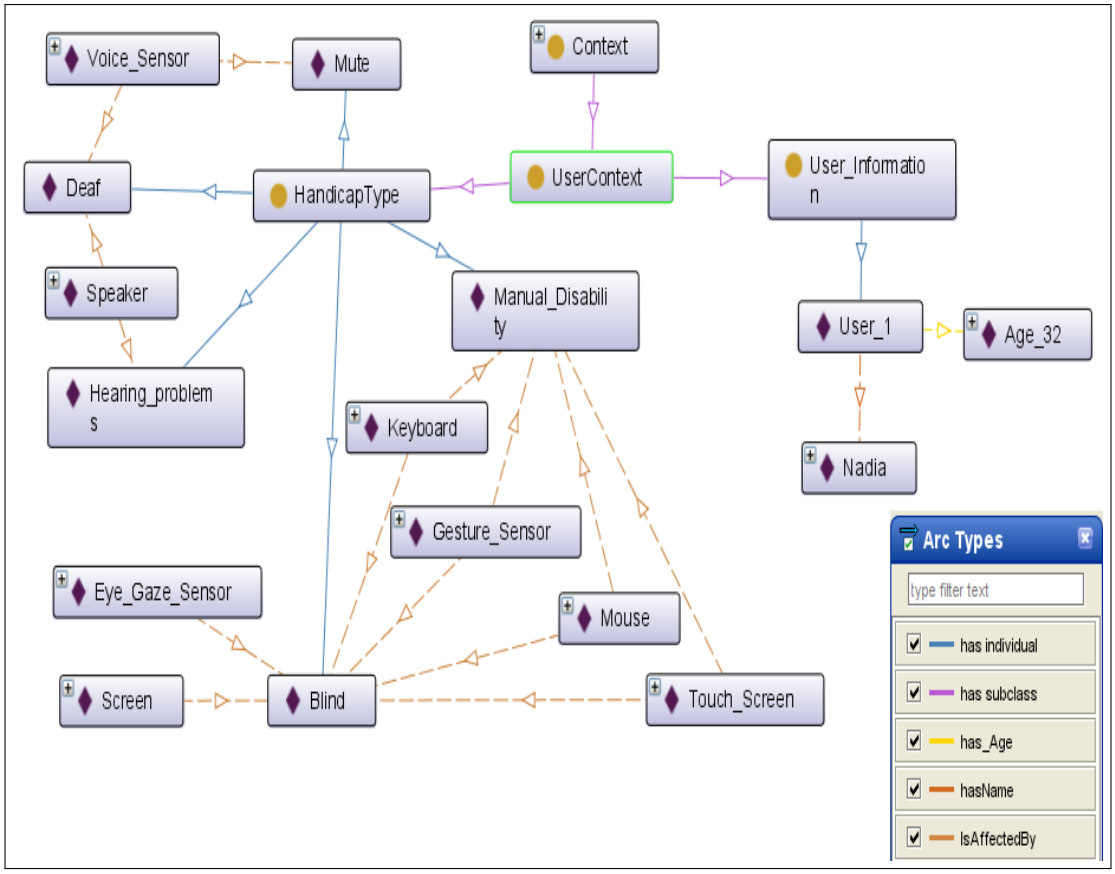


FIGURE 3.20: Les sous classes de la classe "Contexte de l'utilisateur" (User context) et leurs instances.

Sur la Figure 3.20, nous retrouvons aussi les propriétés d'objet "has Age" et "has Name" qui permettent, respectivement, de relier l'instance "user 1" a son âge dans la classe "Age" et son prénom dans la classe "Peoples Names". Aussi, la propriété "Is affected by" permet de relier les instances de la classe "Input Modality" aux instances qui influent sur leur bon fonctionnement. Par exemple, l'instance "Voice sensor" est affectée par les instances "Deaf" (sourd) et "Mute" (muet) (si ces dernières sont présentent comme handicap).

3.4 **Alarme (Alarme)** : Cette classe, représentée par la Figure 3.21, prend en compte les alarmes présentes dans l'environnement. Ses deux sous classes sont :

i. **Alarme de santé (Health Alarm)** : Cette alarme se déclenche lorsque le système détecte un problème de santé, comme une hausse de la tension artérielle ou de la température corporelle, elle a pour instance :

- Alarme 1 (Alarm 1)

ii. **Alarme système (System Alarm)** : Cette alarme se déclenche selon l'état du contexte du système. C'est à dire, si la batterie est faible ou que les capteurs infrarouges ou les capteurs ultrasons (embarqués sur le fauteuil) détectent un obstacle. Elle a pour instance

- Alarme 2 (Alarm 2)

Sur la Figure 3.21, on trouve aussi la propriété d'objet "is Started by" qui permet de relier les instances de la classe "Alarm" avec les instances qui les déclenchent de la classe "Context".

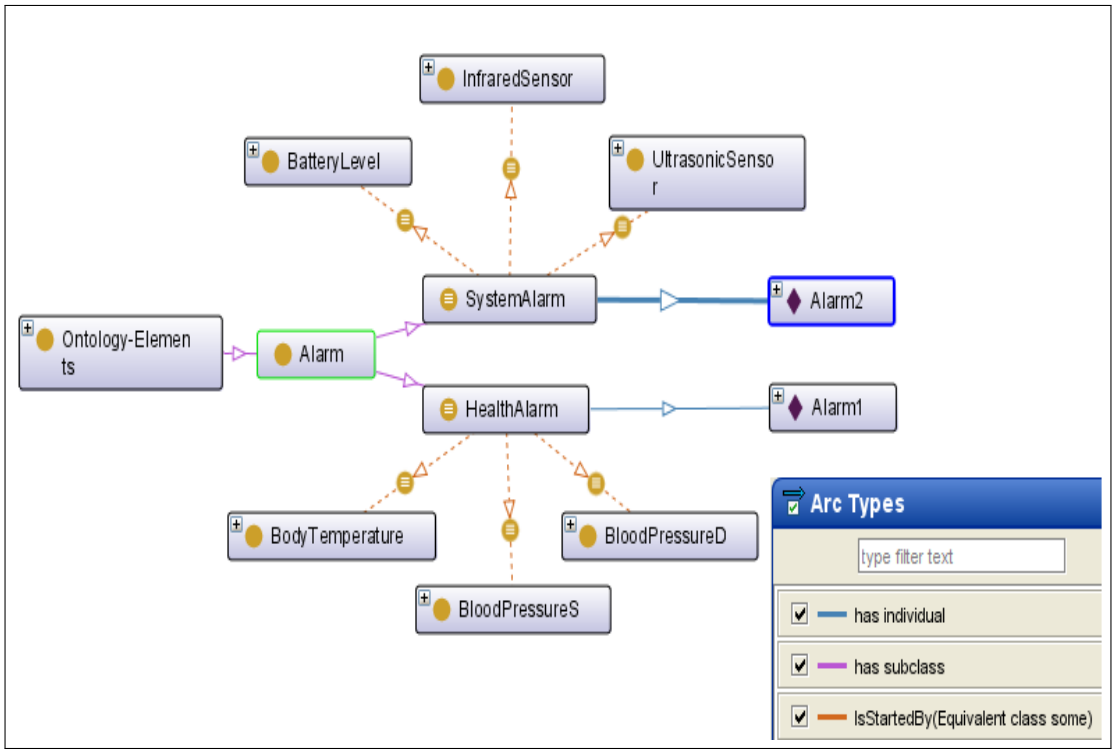


FIGURE 3.21: La classe "Alarm" et ses sous classes

3.5 **Les modalités d'entrée (Input Modality)** : Cette classe, représentée par la Figure 3.22 désigne les modalités d'entrée utilisées par l'utilisateur pour la commande du système. Nous trouvons aussi sur la Figure 3.22 les individus de chaque sous classes ainsi que les propriétés d'objet relatives à chaque élément :

i. **La parole (Speech)** : a pour instance la valeur mesurée par le

- Capteur de voix (Voice Sensor)

ii. **Le geste (Gesture)** : a pour instance la valeur mesurée par le
   - Capteur de geste (Gesture Sensor)

iii. **La commande manuelle (Manual)** : a pour instances
   - Clavier (Keyboard)
   - Souris (Mouse)

iv. **La commande visuelle (Visual)** : a pour instance la valeur captée par le
   - Capteur du regard (Eye gaze sensor)

v. **La commande tactile (Touch)** : a pour instance la valeur reçue par un
   - Ecran tactile (Touch screen)

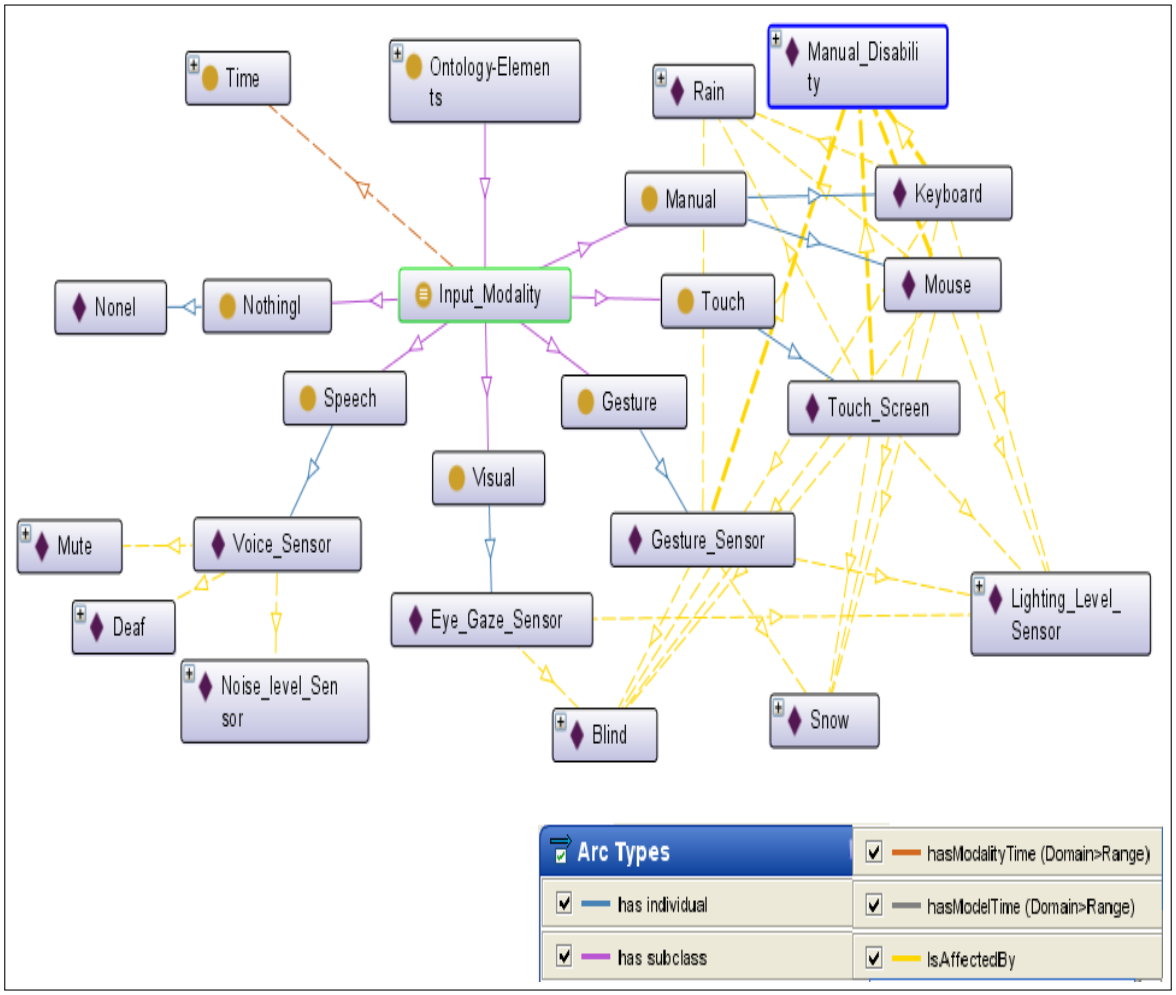


FIGURE 3.22: La classe “Input Modality”, ses sous classes et leurs individus

Aussi, nous trouvons sur la Figure 3.22 une représentation des propriétés

d'objet utilisé pour lier les éléments de ces classes. Par exemple, l'instance de la modalité "Speech" est reliée aux instances "muet", "sourd" et "niveau de bruit" car la modalité "Parole" peut être affectée par le niveau de bruit et le type d'handicap de l'utilisateur.

3.6 **Les modalités de sortie (Output Modality)** : Cette classe, représentée par la Figure 3.23 désigne les modalités de sortie utilisées par le système pour répondre à la requête de l'utilisateur. Si une de ces modalités est désactivée à cause d'une valeur du contexte, une autre sera sélectionnée dans l'ordre d'accessibilité. Si la sortie mobile n'est pas utilisable, l'utilisateur sera avisé par écran sinon par haut-parleur. Si l'écran n'est pas utilisable le haut-parleur sera utilisé et vice versa. Nous trouvons aussi sur la Figure 3.23 les individus de chaque sous classes :

i. **Sortie vocale (Vocal output)** : A pour instance
   - Hautparleur (Speeker)

ii. **Sortie visuelle (Visual output)** : A pour instance
   - Ecran (Screen)

iii. **Sortie mobile (Moving output)** : A pour instances
   - Roues (Wheels)
   - Bras manipulateur (Arm)

La propriété d'objet "Is Affected by", représentée sur la Figure 3.23, permet de relier les instances de la classe "Output Modality" aux instances de la classe "Context" qui affectent leur fonctionnement.

Sur les Figures 3.22 et 3.23 une classe supplémentaire, les classes "Nothing1" et "Nothing", ont été définies pour faciliter la déclaration de règles de sélections de modalité et leur affichage. En effet, le logiciel Protégé ne retourne rien lorsque la règle n'est pas satisfaite, donc pour éviter cela, nous avons déclaré ces classes avec leurs individus "None1" et "None", respectivement, qui vont être affichés lorsqu'il n'y aura aucune modalité à désactiver (c'est-à-dire lorsque toutes les modalités sont disponibles).

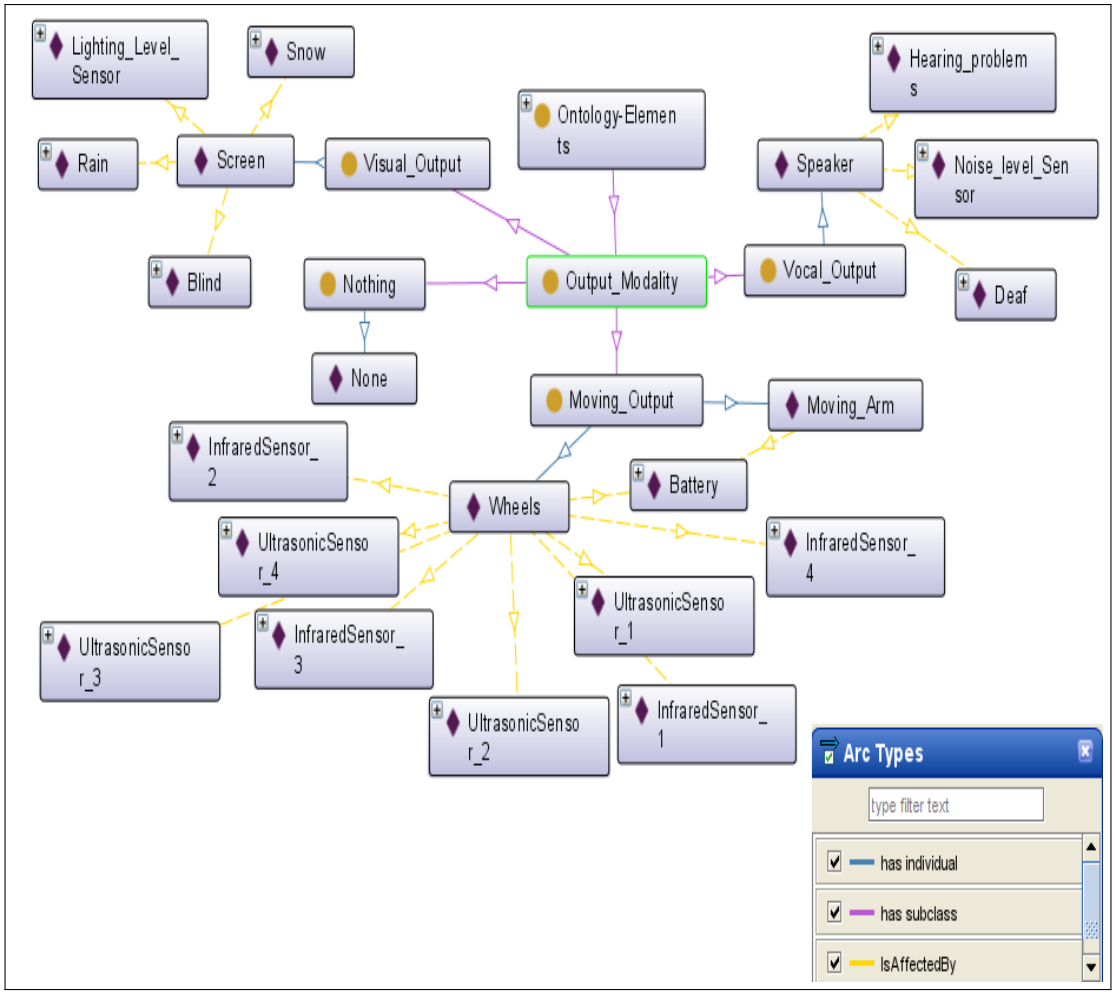


FIGURE 3.23: La classe “Output Modality”, ses sous classes et leurs individus

3.7 **Le voisinage (Surroundings)** : Cette classe, représentée par la Figure 3.24, a trois sous classes :

i. **Objets (Objects)** : Cette classe, représentée par la Figure 3.25, désigne tous les objets trouvés dans l’entourage de l’utilisateur. Ses sous classes sont :

- *Transport means (moyens de transport)* : représentée par la Figure 3.26, cette classe a comme instances les différents moyens de transport possibles de l’utilisateur, qui sont :

  ▷ Bus

  ▷ Voiture (car)

  ▷ Taxi

- ▷ Tram
- ▷ Tube (métro)

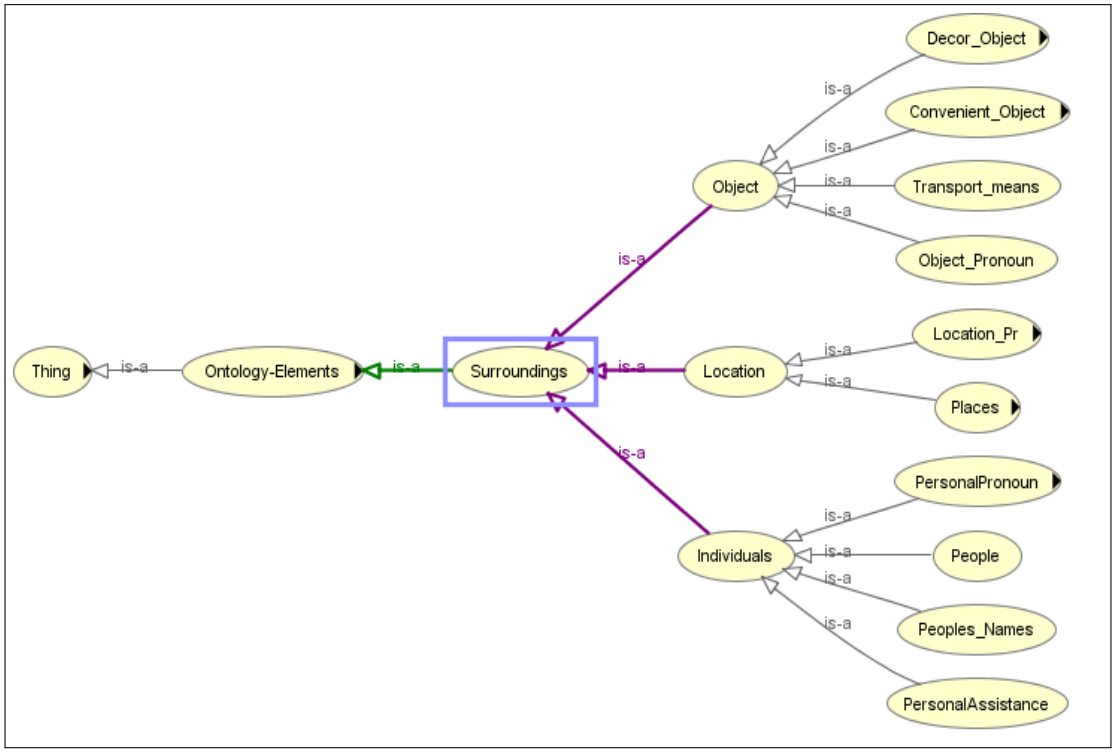


FIGURE 3.24: La classe Surroundings et ses sous classes

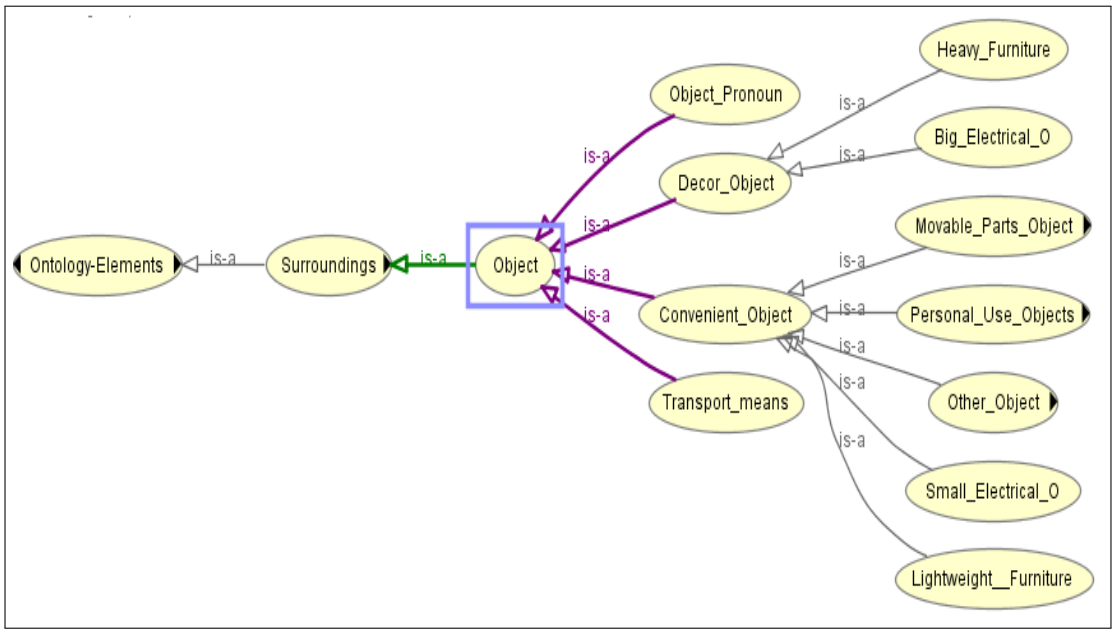


FIGURE 3.25: La classe Objets et ses sous classes

Sur la Figure 3.26, la propriété d'objet "Compose 16_3" précise les classes qui composent le modèle de fusion 16. La propriété "has Next

16_2" permet de définir l'ordre des classes qui composent le modèle de fusion 16.

- *Objets de décor (Decor objects) :*
  - ▷ Meubles lourds (Heavy Furniture) : représentée par la Figure 3.27, a pour instances :
    - ∗ Lit (bed)
    - ∗ Table (table)
    - ∗ Placard (cupboard)
    - ∗ Bureau (desk)
    - ∗ Sofa
    - ∗ Armoire (wardrobe)
    - ∗ Commode (dresser)

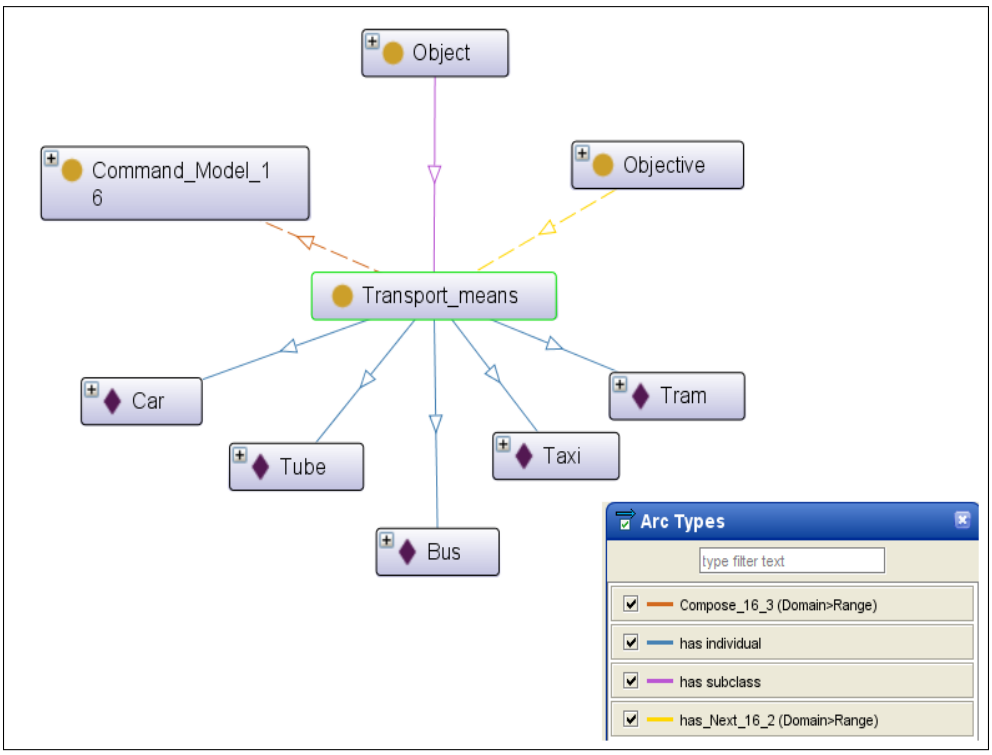


FIGURE 3.26: La classe "Moyen de transport" (Transport) et ses instances

  - ▷ Objets électriques grands (Big electrical objects) : représentée par la Figure 3.28, a pour instances
    - ∗ Lave-vaisselle (dishwasher)

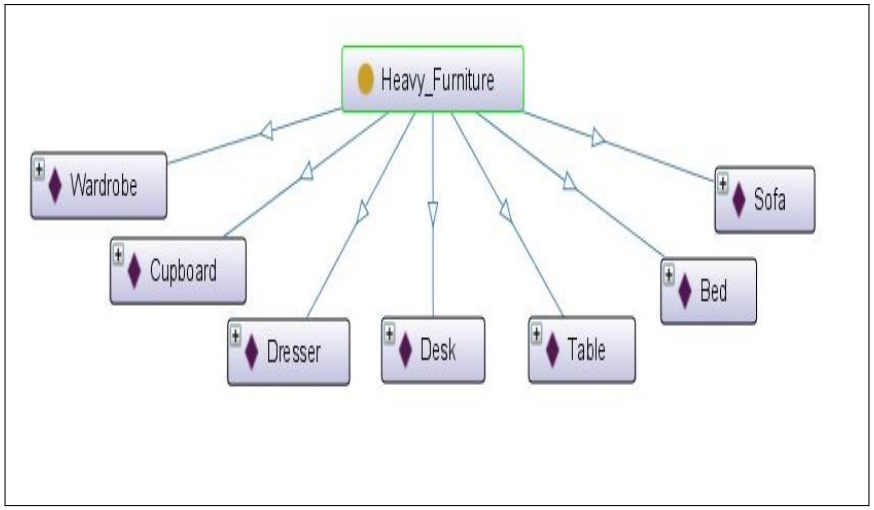


FIGURE 3.27: La classe "Meubles lords" (Heavy furniture) et ses instances

- Micro onde (microwave)
- Four (oven)
- Imprimante (printer)
- Radio (radio)
- Réfrigérateur (Refrigerator)
- Scanner
- Télévision (Television)
- Toasteur (toaster)
- Machine à laver (washing machine)

- *Pronom pour les objets (Object Pronoun)* : aura comme instances les mots utilisés comme pronom d'objet. Cette classe est représentée par la Figure 3.29 et a pour instances :

  - Le (It)
  - Ceci (That)
  - Celles-ci (These)
  - Ce (This)
  - ceux (Those)

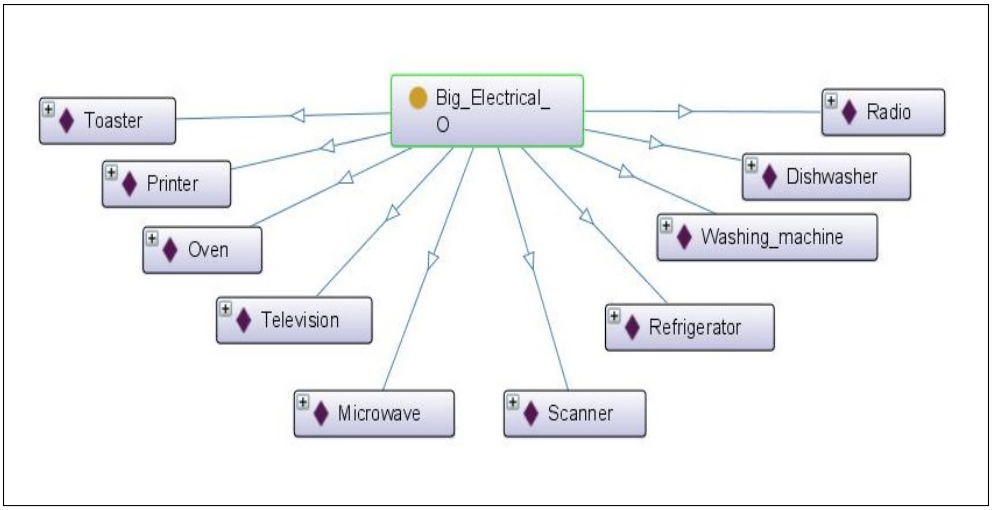


FIGURE 3.28: La classe "Grand objet électrique" (Big Electrical Object) et ses instances

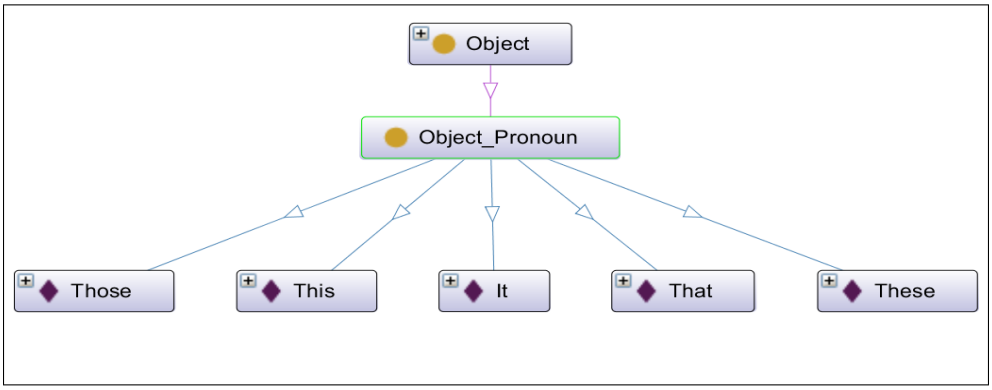


FIGURE 3.29: La classe "Pronom pour les objets" (Object pronoun) et ses instances

- *Objets pratiques (Convenient Objects)* :
  - ▷ Objets à partie mobile (Movable parts objects) : a comme sous classes et instances :
    - ∗ Parties mobiles du batimenet (Movable parts of building)
      - · Fenêtre (window)
      - · Porte (door)
    - ∗ Autres objets à partie mobile (Other Movable parts objects) :
      - · Garde-robe (wardrobe)
      - · Placard (cupboard)
      - · Four (oven)
      - · Commode (dresser)

- Bureau (desk)
- Micro onde (microwave)
- Réfrigérateur (refrigirator)
- Lave-vaisselle (dishwasher)
- Machine à laver (washing machine)

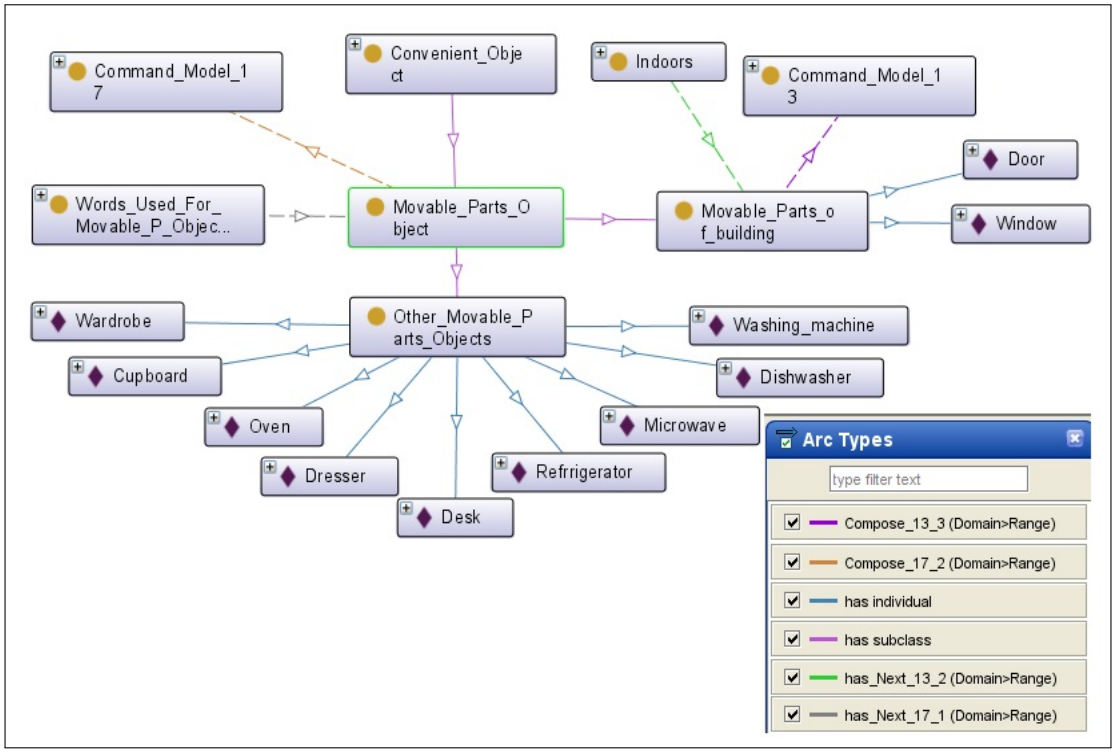


FIGURE 3.30: La classe "Objets à parties mobiles" (Movable parts object) et ses instances

Sur la Figure 3.30, la propriété d'objet "Compose 13_3" et "Compose 17_2" précisent les classes qui composent le modèle de fusion 13 et 17. Les propriétés "has Next 13_2" et "has Next 17_1" permettent de définir l'ordre des classes qui composent les modèles de fusion 13 et 17.

- ▷ Meubles légers (Lightweight furniture) : représentée par la Figure 3.31, a pour instances
  - ∗ Tabouret (stool)
  - ∗ Chaise (chair)

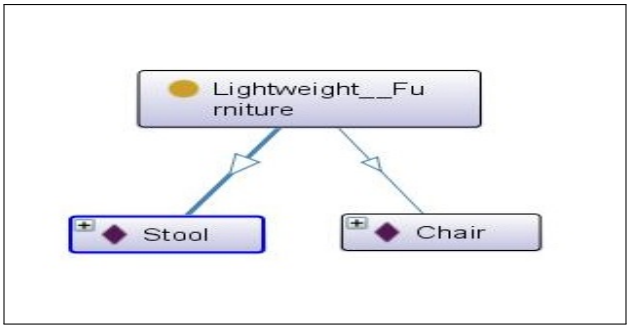


FIGURE 3.31: La classe " Meubles légers" (lightweight furniture) et ses instances

- Autres objets (Other objects) : représentée par la Figure 3.32 a comme sous classes :
  - Objets à partie ouvrable (openable parts objects) : a pour instances
    - Valise (suitcase)
    - Boite (box)
    - Sac (bag)
  - Petits objets utiles (small useful objects) : a pour instances
    - Boîte à lettres (letter box)
    - Poubelle (bin)
- Petits objets électriques (small electrical object) : représentée par la Figure 3.33, à pour instances
  - Appareil photo (camera)
  - Ordinateur (computer)
  - Décodeur (decoder)
  - Lecteur DVD (DVD player)
  - Téléphone fixe (Landline phone)
  - Téléphone portable (mobile phone)
  - Télécommande (remote control)

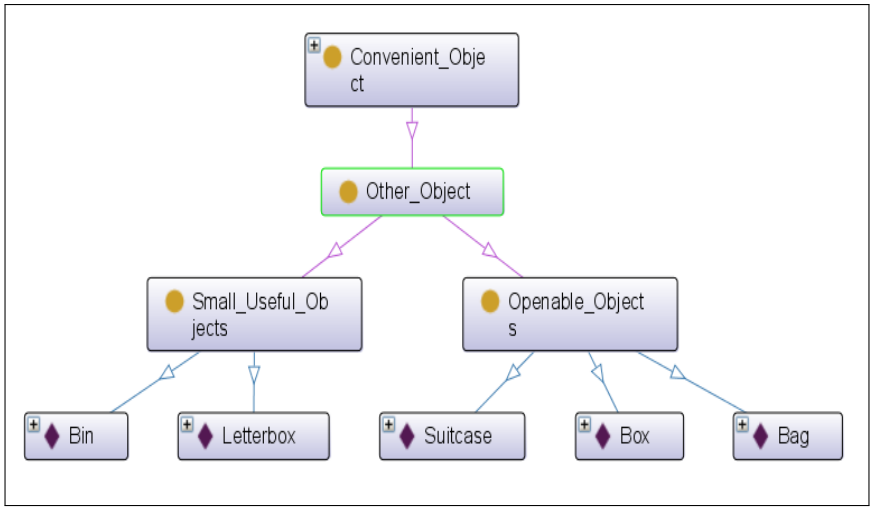


FIGURE 3.32: La classe "Autres objets" (Other objects) et ses instances

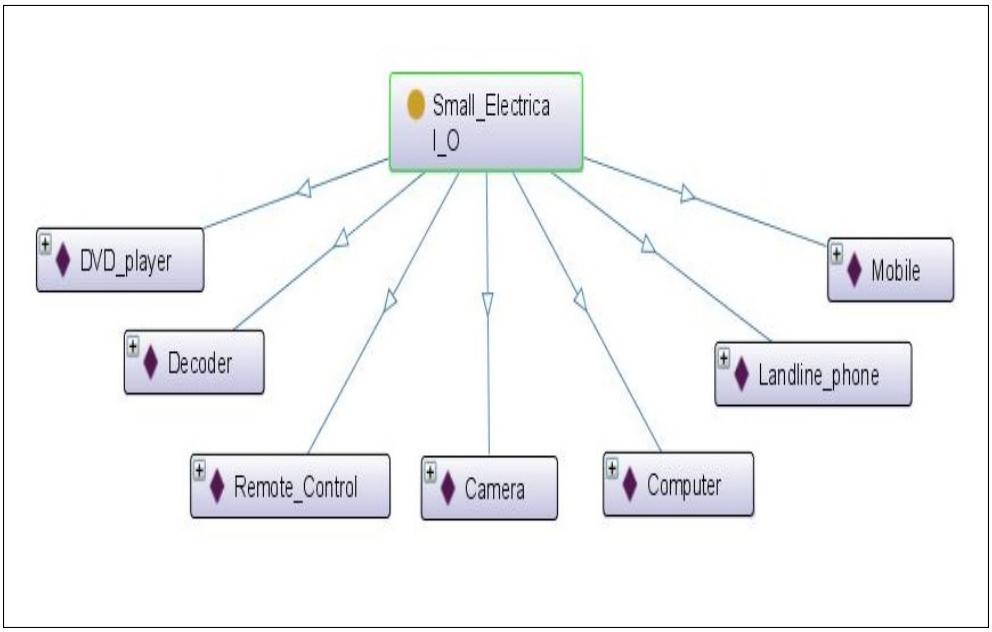


FIGURE 3.33: La classe "Petit objet électrique" (Small Electrical Object) et ses instances

▷ Objets à usage personnel (Personal use objects) : a comme sous classes :

∗ Objets de tous les jours (everyday objects) : représentée par la Figure 3.34 :

· Livre (book)

· Montre (watch)

· Clés (keys)

- Stylo (pen)
- Papier (paper)

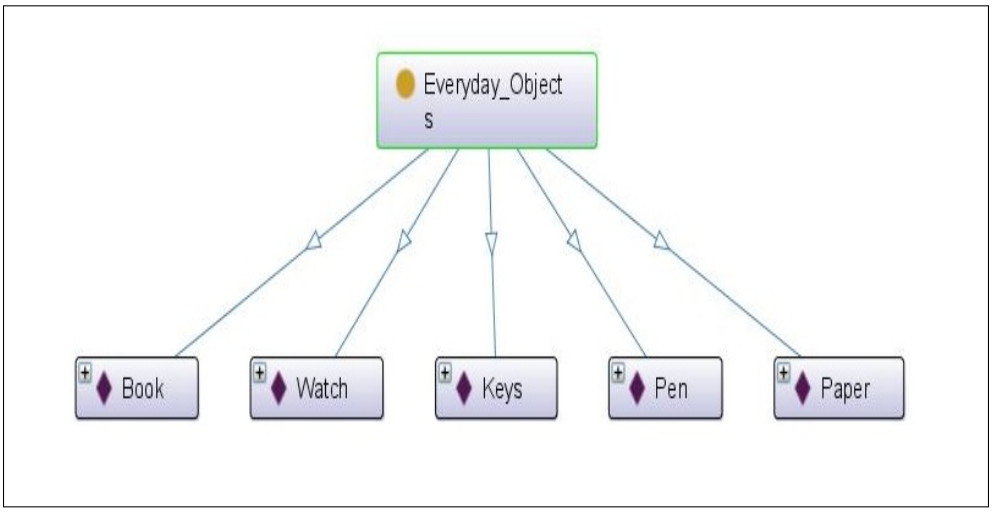


FIGURE 3.34: La classe "Objet de tous les jours" (everyday objects) et ses instances

* Objets de literie (bedding objects) : représentée par la Figure 3.35, a pour instances
  - Couverture (blanket)
  - Oreiller (pillow)
  - Coussin (cushion)
  - Drap (sheet)

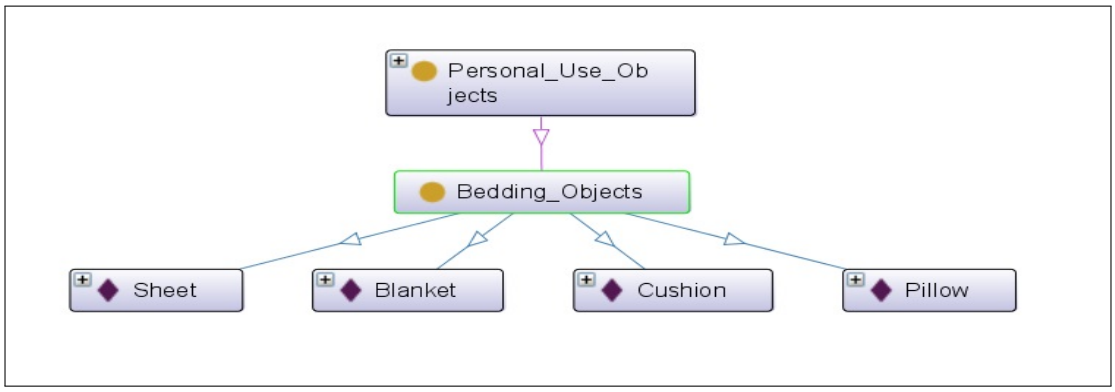


FIGURE 3.35: La classe "Literie" (Bedding objects) et ses instances

* Vêtements (clothes) : représentée par la Figure 3.36, a pour instances
  - Cardigan (cardigan)

- Manteau (coat)
- Robe (dress)
- Gants (gloves)
- Chapeau (hat)
- Veste (jacket)
- Pull (jumper)
- Echarpe (scarf)
- Chemise (shirt)
- Jupe (skirt)
- Chaussettes (socks)
- Chaussures (shoes)
- Pantalons (trousers)

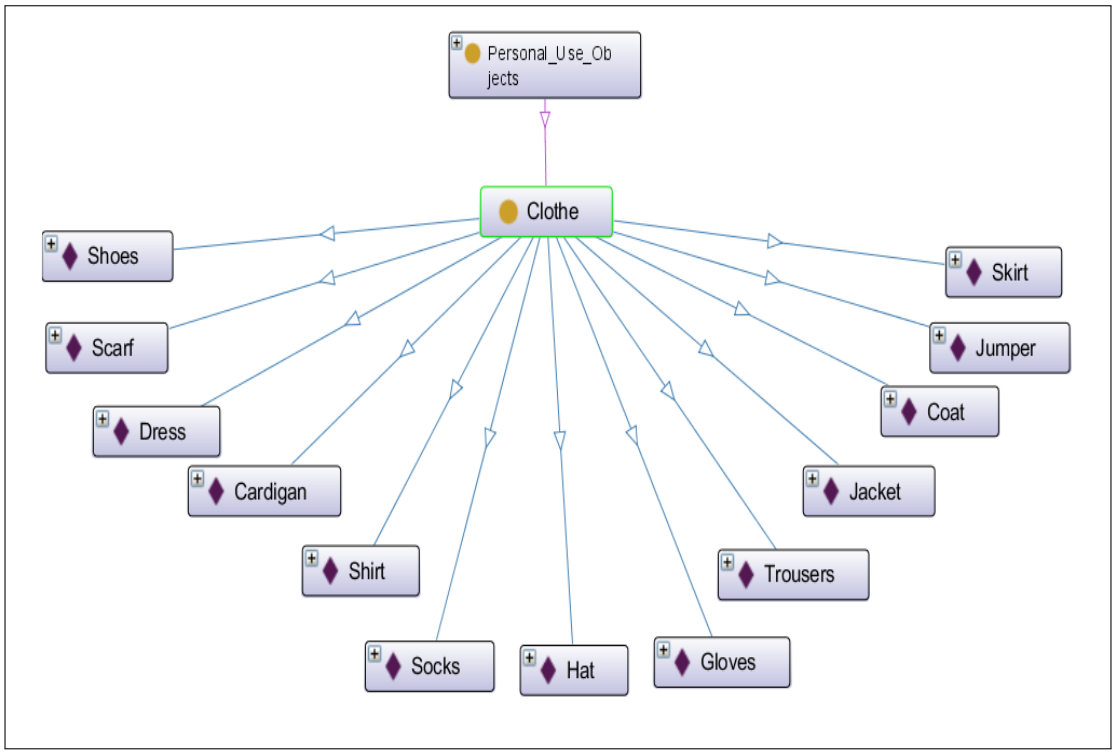


FIGURE 3.36: La classe "Vêtements" (clothe) et ses instances

* Comestible (edible) : a comme sous classes et instances :
  - Nourriture (food) : représentée par la Figure 3.37, a pour instances

- ◇ Pomme (apple)
- ◇ Abricot (apricot)
- ◇ Banane (banana)
- ◇ Pain (bread)
- ◇ Beure (butter)
- ◇ Gâteau (cake)
- ◇ Carotte (carrot)
- ◇ Fromage (cheese)
- ◇ Poulet (chicken)
- ◇ Chocolat (chocolate)
- ◇ Courgette (courgette)
- ◇ Poisson (fish)
- ◇ Raisin (grape)
- ◇ Herbes (herbs)
- ◇ Confiture (jam)
- ◇ Viande (meat)
- ◇ Oignon (onion)
- ◇ Orange (orange)
- ◇ Pâtes (pasta)
- ◇ Pèche (peach)
- ◇ Poire (pear)
- ◇ Pizza
- ◇ Pomme de terre (potato)
- ◇ Riz (rice)
- ◇ Sel (salt)
- ◇ Epices (spices)
- ◇ Fraise (strawberry)
- ◇ Sucre (sugar)
- ◇ Tomate (tomato)

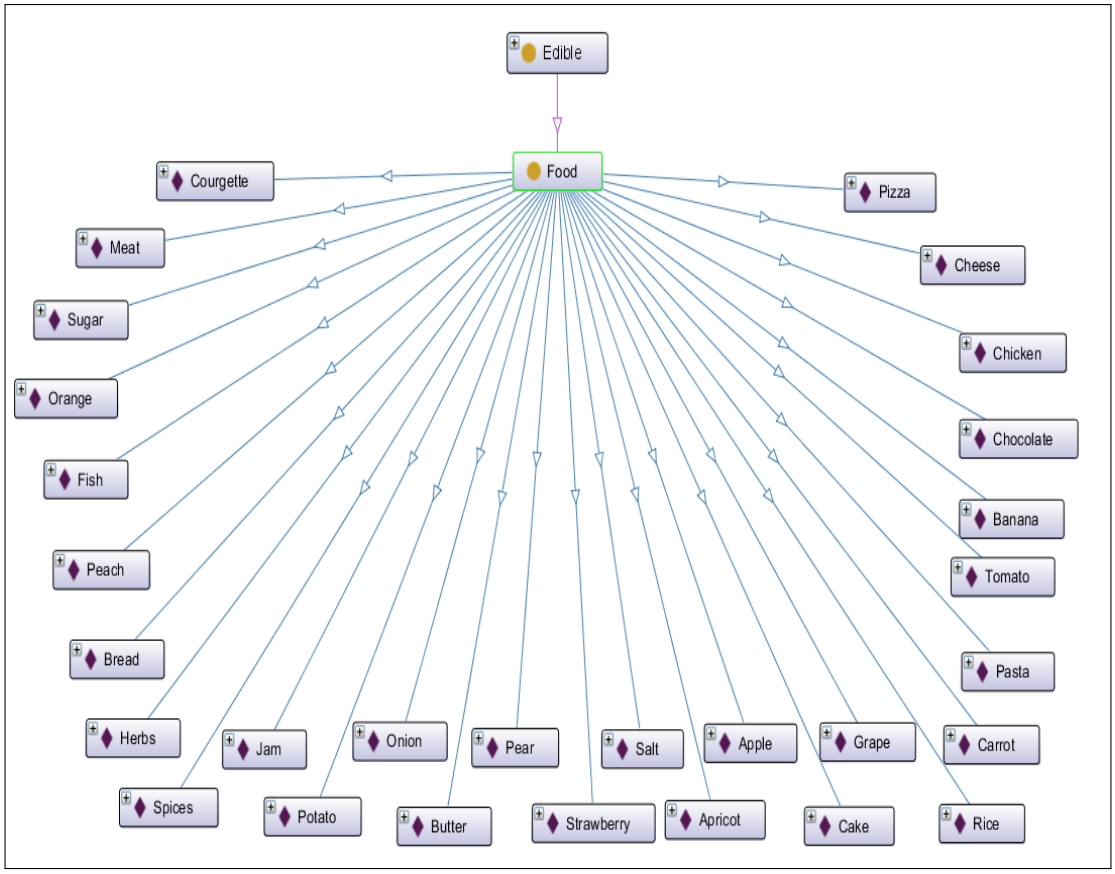


FIGURE 3.37: La classe "Nourriture" (Food) et ses instances

- Liquide (Liquid) : représentée par la Figure 3.38, a pour instances
  - ◇ Café (coffee)
  - ◇ Jus (Juice)
  - ◇ Lait (milk)
  - ◇ Soda
  - ◇ Thé (tea)
  - ◇ l'eau (water)

  Sur la Figure 3.38, la propriété d'objet "Compose 15_2" et "Compose 19_3" précisent les classes qui composent le modèle de fusion 15 et 19. Les propriétés "has Next 15_1" et "has Next 19_2" permettent de définir l'ordre des classes qui composent les modèles de fusion 15 et 19. La propriété d'objet "Needs liquid object" permet de relier les liquides à des objets

utilisés pour les liquides comme un verre ou une tasse.

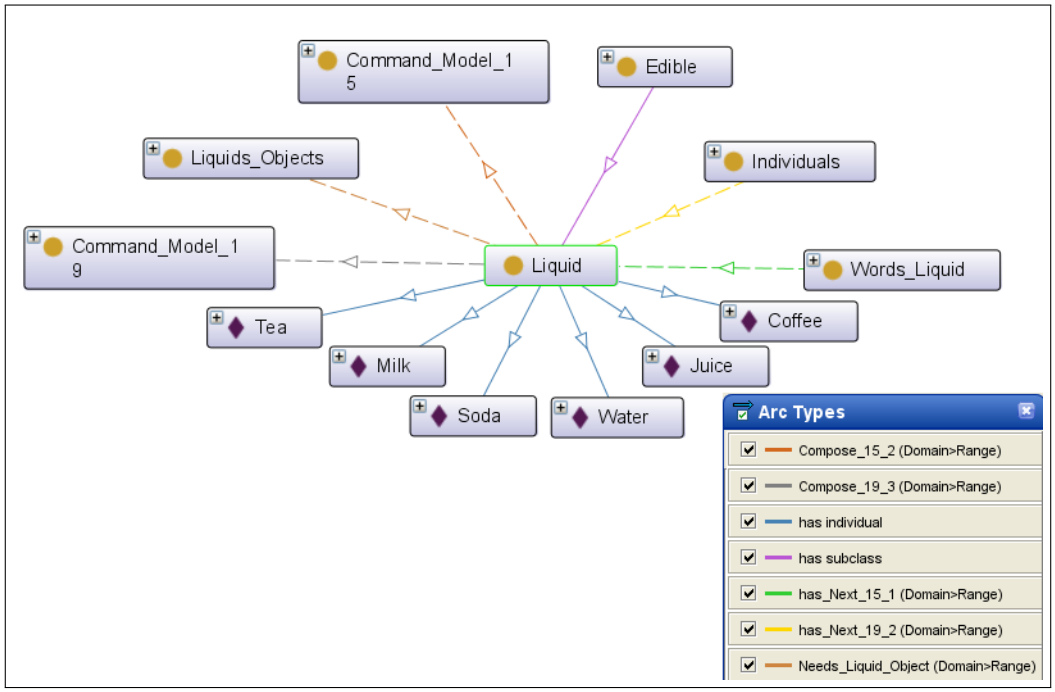


FIGURE 3.38: La classe "Les liquides" (Liquid) et ses instances

· Médicaments (Drug) : représentée par la Figure 3.39, a pour instances

◇ Sirop (sirup)

◇ Comprimés (tablets)

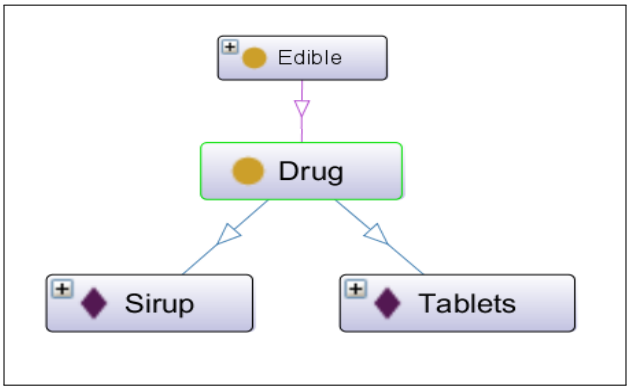


FIGURE 3.39: La classe "Médicaments" (Drug) et ses instances

∗ Objets pour comestible (objects for edible)

· Objets pour la nourriture (food objects) :représentée par la Figure 3.40, a pour instances

- ◇ Bol (bowl)
- ◇ Fourchette (fork)
- ◇ Poêle (frying pan)
- ◇ Bocal (jar)
- ◇ Couteau (knife)
- ◇ Casserole (pan)
- ◇ Assiette (plate)
- ◇ Spatule (spatula)
- ◇ Cuillère (spoon)
- ◇ Plateau (tray)
- ◇ Fouet (whisk)

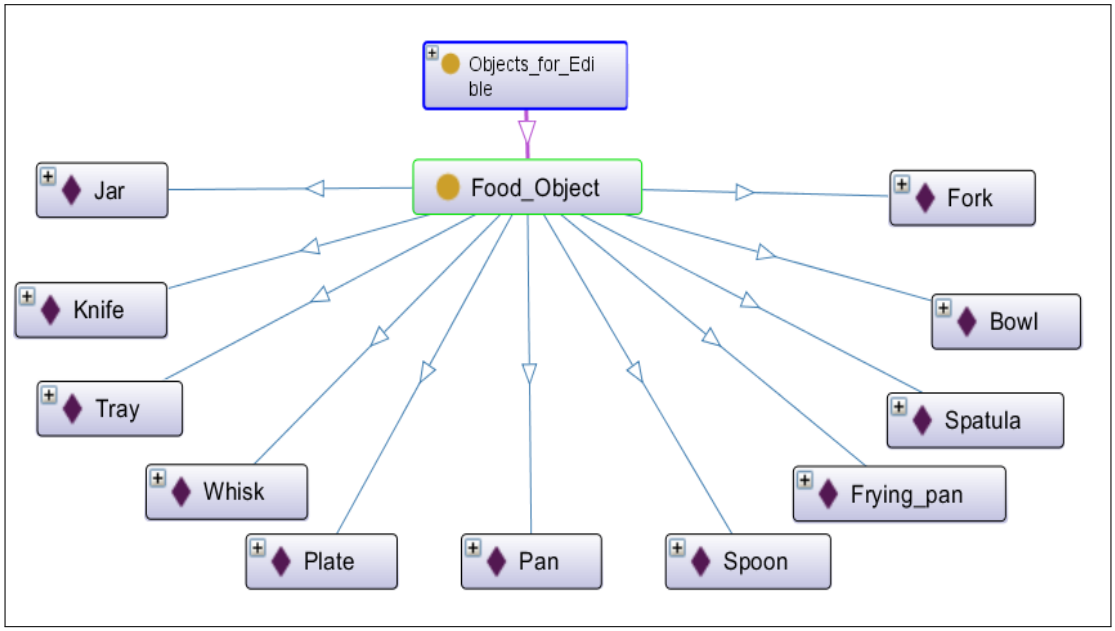


FIGURE 3.40: La classe "Objets pour la nourriture" (Food object) et ses instances

· Objets utilisés pour les liquides (liquids objects) : représentée par la Figure 3.41, a pour instances

- ◇ Tasse (cup, mug)
- ◇ Bouteille (bottle)
- ◇ Verre (glass)
- ◇ Cruche (Jug)

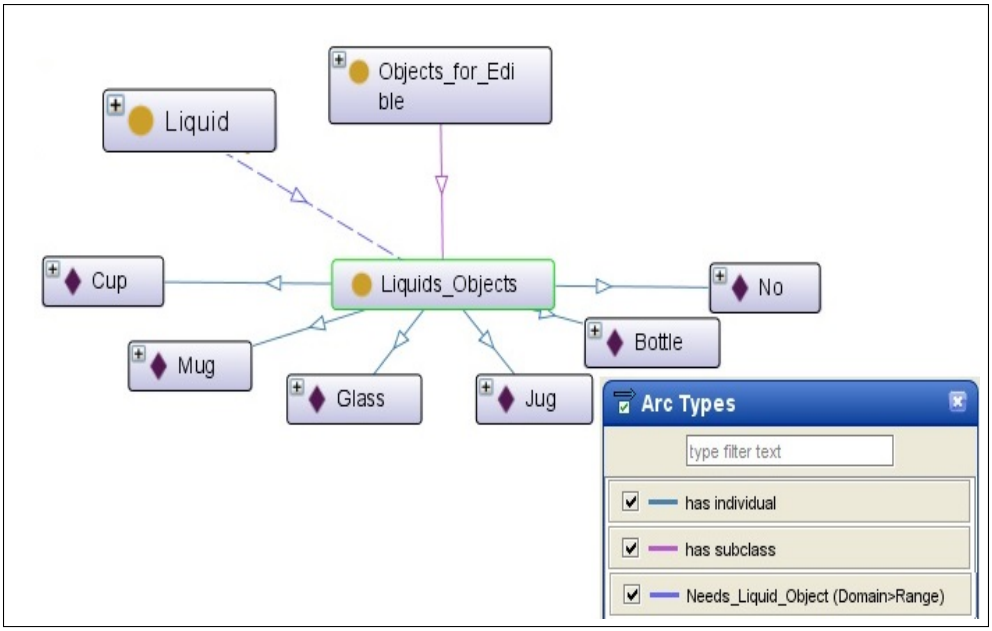


FIGURE 3.41: La classe "Objets pour les liquides" (Liquid objects) et ses instances

Sur la Figure 3.41 la propriété d'objet "needs liquid object" permet de préciser quel instance de la classe "Liquid" a besoin d'un objet pour les liquides.

ii. **Individu (Individual)** : Cette classe, représentée par la Figure 3.42, désigne toutes les personnes potentiellement présentes dans l'entourage du système. Ses sous classes sont :

- *Personnes (People)* : représentée par la Figure 3.43, a pour instances
  - ▷ Tante (aunt)
  - ▷ Frère (brother)
  - ▷ Collègue (colleague)
  - ▷ Cousin (cousin)
  - ▷ Fille (daughter)
  - ▷ Père (father)
  - ▷ Ami (friend)
  - ▷ Mère (mother)
  - ▷ Neveu (nephew)
  - ▷ Nièce (niece)

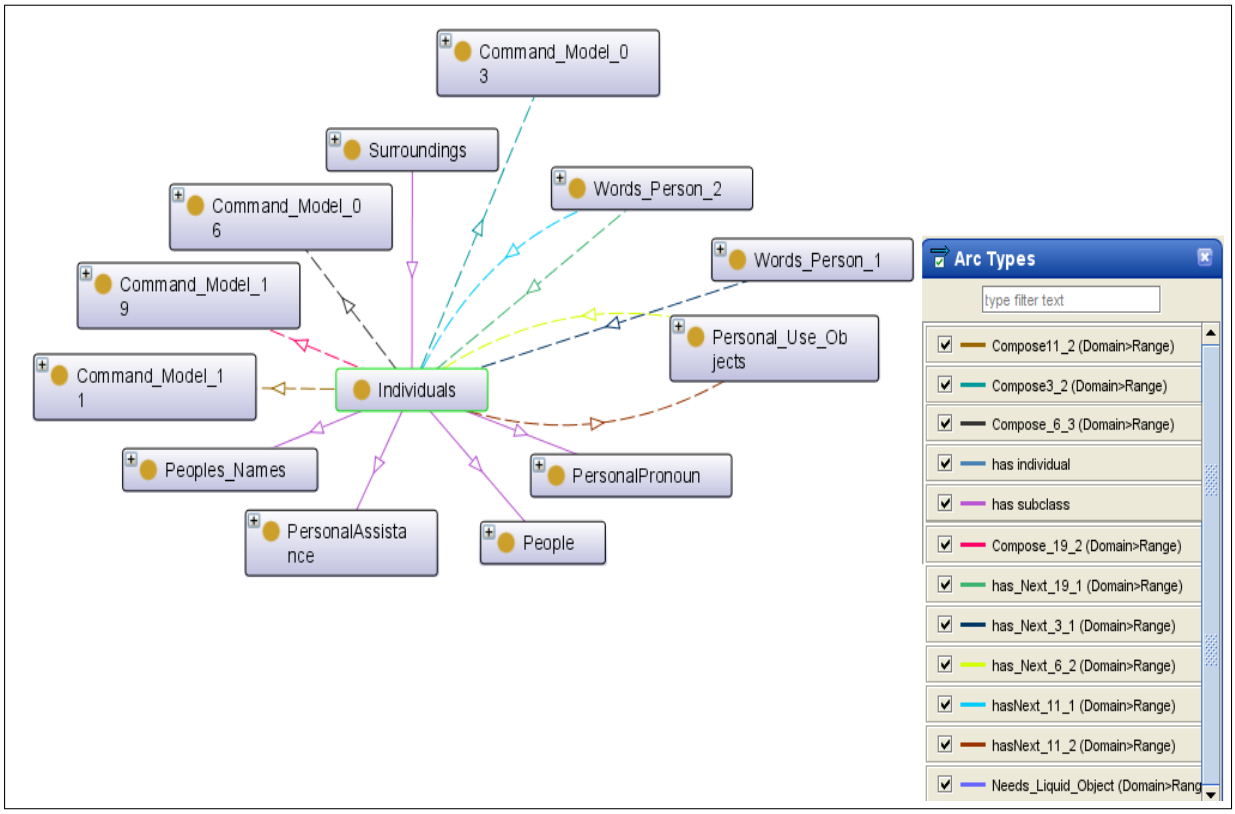


FIGURE 3.42: La classe Individual (individu) et ses sous classes

- ▷ Sœur (sister)
- ▷ Fils (son)
- ▷ Oncle (uncle)

- *Pronom personnel (personal pronoun)* :représentée par la Figure 3.44, a pour sous classes et instances
  - ▷ Objective
    - ∗ Sa/son (her)
    - ∗ Lui (him)
    - ∗ Moi (me)
    - ∗ Leurs (them)
    - ∗ Nous (us)
  - ▷ Possessive
    - ∗ Ma/Mon (my)
    - ∗ Notre (our)

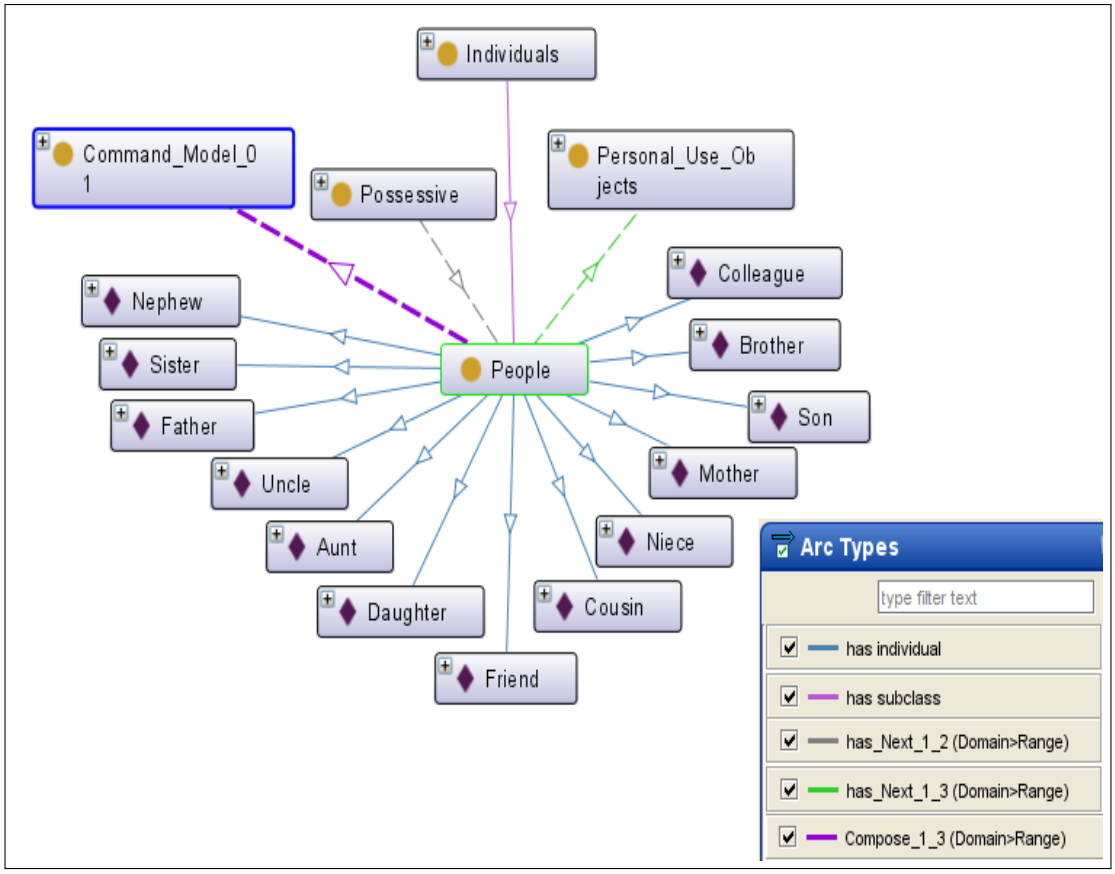


FIGURE 3.43: La classe "Personnes" (People) et ses instances

  - Leur (their)
  - Son (his/her)

- *Assistance personnelle (personal assistance)* : représentée par la Figure 3.45, a pour instances :
  - Infirmière (Nurse)
  - Aide-soignant (Caregiver)
  - Médecin (Doctor)

- *Nom des personnes (peoples names)*représentée par la Figure 3.46, a pour instances :
  - Samia
  - Imene
  - Anna
  - Adam

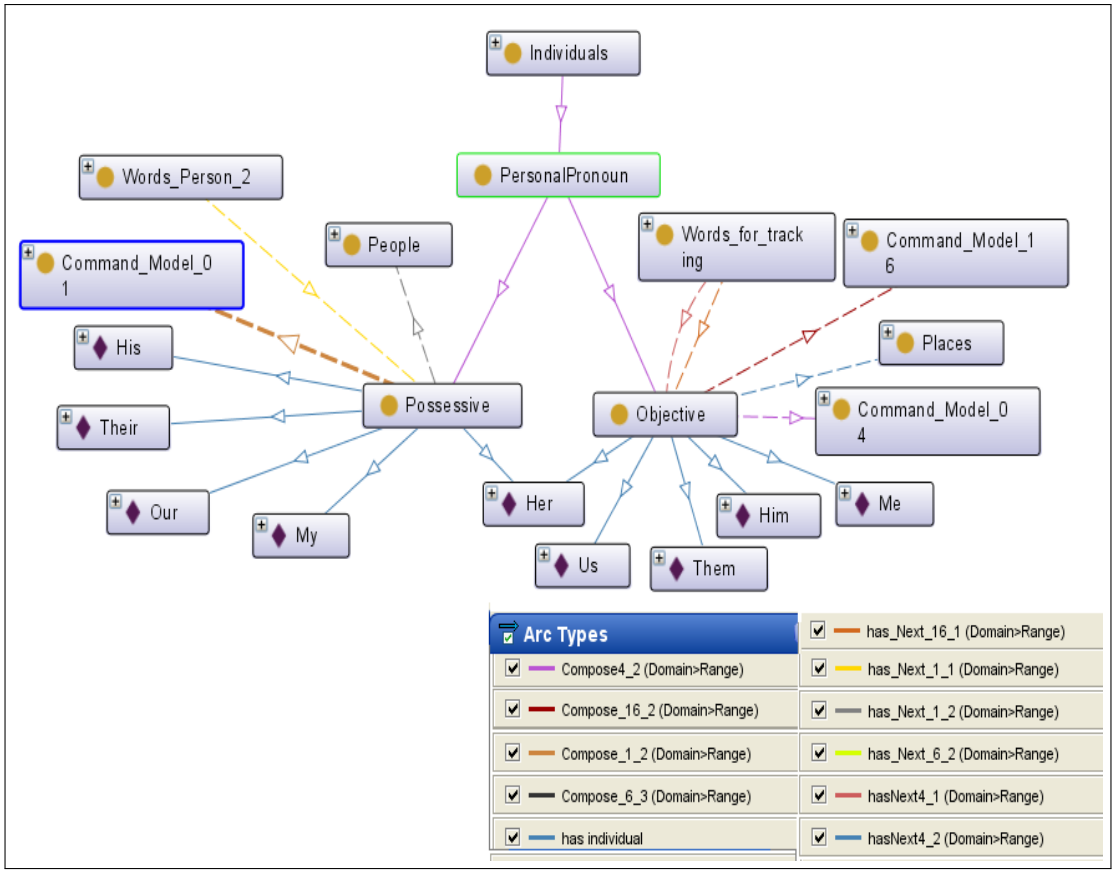


FIGURE 3.44: La classe "Pronom personnel" (Personal pronoun) et ses instances

- David
- Liv
- Ilyes
- Meriam
- Rabah
- Katia
- Yasmine
- Amar
- Jack
- Alto
- Nadia
- Dr_Djaid
- Hans

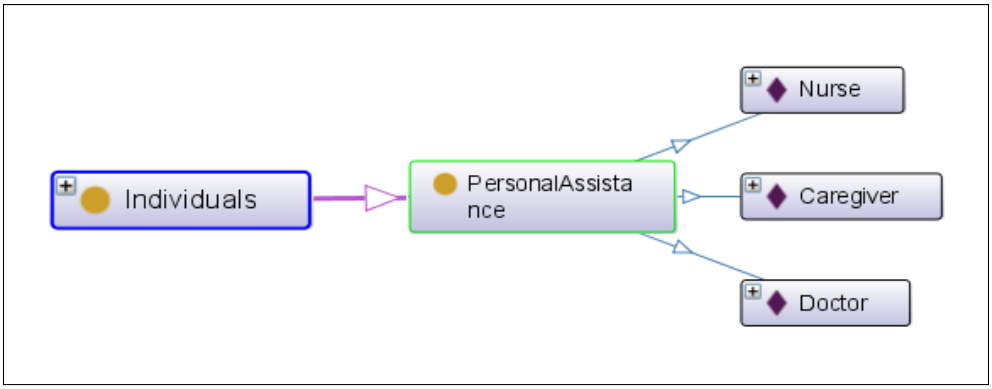


FIGURE 3.45: La classe "Assistant personnel" (Personal assistant) et ses instances

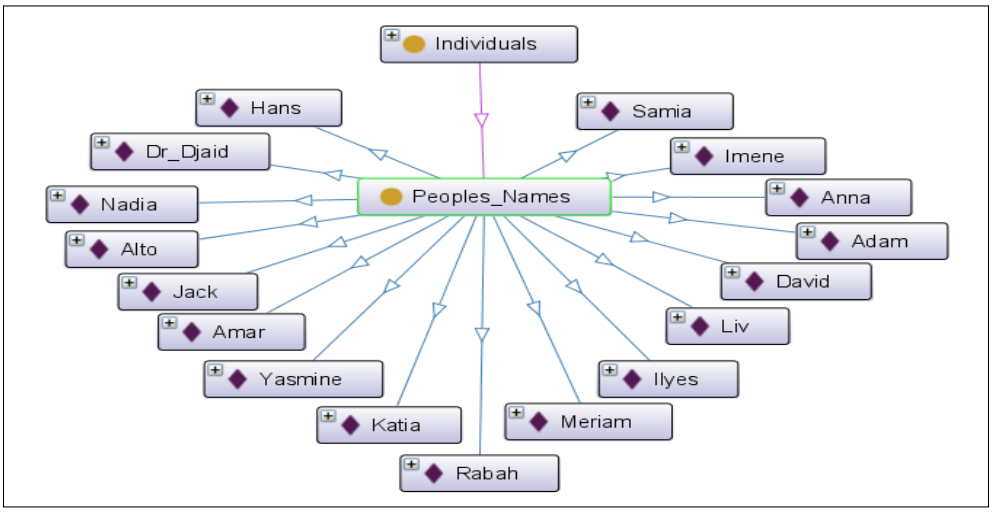


FIGURE 3.46: La classe "Nom des personnes" (peoples names) et ses instances

iii. **Localisation (location)** : Cette classe représentée par la Figure 3.47, désigne les différentes localisations potentielles de l'utilisateur et son fauteuil. La classe "Location" est ses sous classes sont présentées sur la Figure 3.47.

- *Endroits (Places)*
  - ▷ L'intérieur (indoors) : représentée par la Figure 3.48, a pour instances :
    - ∗ Salle de bain (bathroom)
    - ∗ Chambre (bedroom)
    - ∗ Couloir (corridor)
    - ∗ Cuisine (kitchen)

* Salon (living room)
* Séjour (sitting room)
* Toilette (toilet)
* Ici (here)

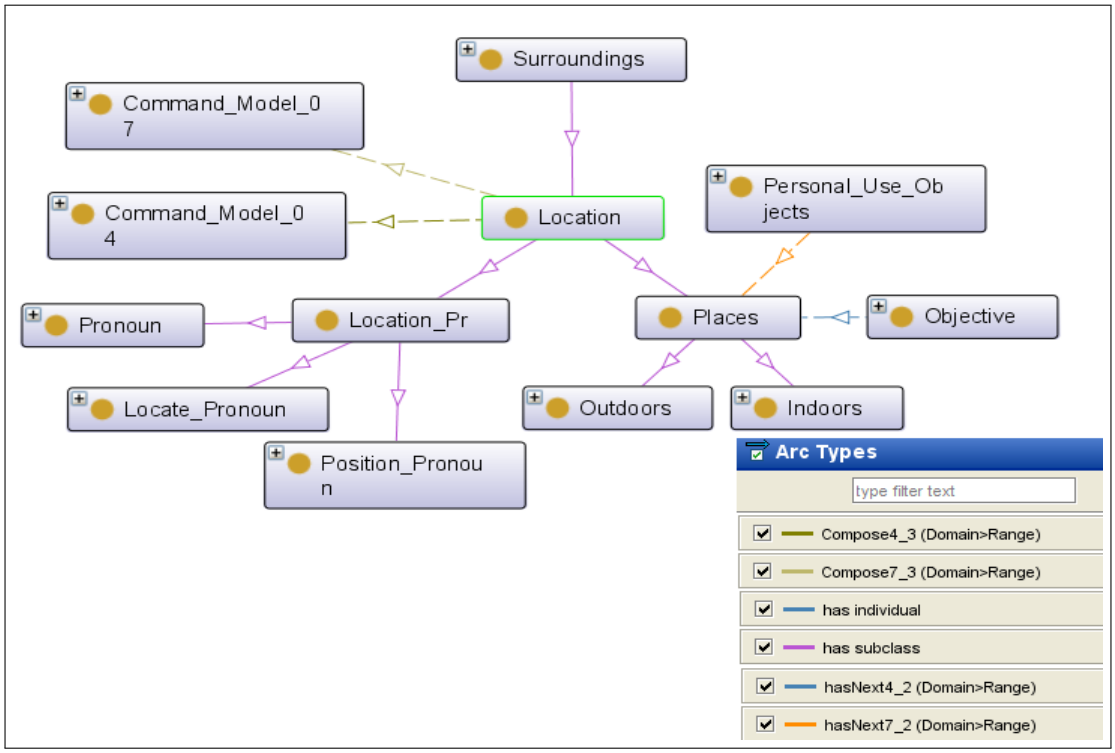


FIGURE 3.47: La classes Location (Localisation) et ses sous classes

▷ L'extérieur (outdoors) : représentée par la Figure 3.49, a pour instances :

* Garage (garage)
* Arrière-cour (backyard)
* Jardin (garden)
* Bureau (office)
* Parc (park)
* Route (road)
* Magasins (shops)
* Patio
* Là-bas (there)

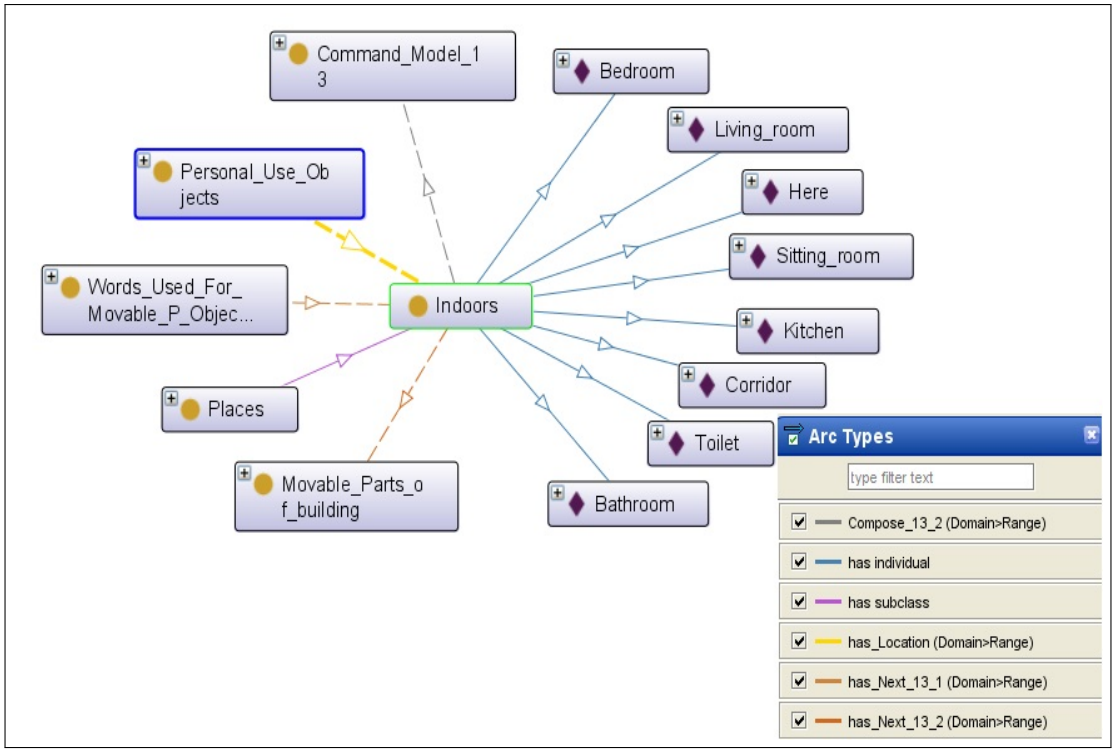


FIGURE 3.48: La classe "Intérieur de la maison" (Indoors) et ses instances

- *Pronom de localisation (Location pronoun)* : Cette classe, représentée sur la Figure 3.50, a comme sous classes et instances :
  - ▷ Pronom de localisation (locate pronoun) : a pour instances
    - ∗ Dans (in)
    - ∗ A l'intérieur (inside)
    - ∗ A l'extérieur (outside)
  - ▷ Pronom (Pronoun) :
    - ∗ Ici (here)
    - ∗ Là-bas (there)
  - ▷ Pronom de position (Position pronoun) :
    - ∗ Derrière (behind)
    - ∗ A côté de (beside)
    - ∗ Devant (in front)
    - ∗ A côté de (next to)
    - ∗ A gauche (to the left)

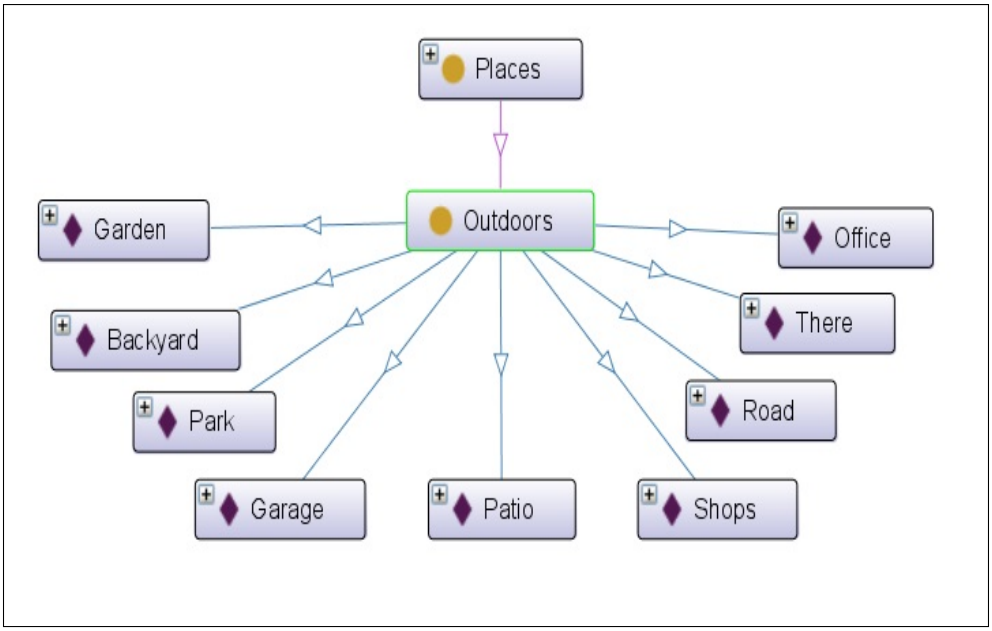


FIGURE 3.49: La classe "Extérieur de la maison (outdoors)" et ses instances

- A droite (to the right)
- Sous (under)
- Dessous (underneath)
- Sur (on)

3.8 **Age** : Cette classe, représentée par la Figure 3.51 a comme instances les âges possibles des personnes présentent dans l'entourage de l'utilisateur, ainsi que l'utilisateur lui même. Ces âges varient de 1 à 100 ans. Une propriété d'objet "hasAge" relie les instances des classes "Peoples Names" et "User information", qui sont le "Domain" de la propriété, aux âges qui leur correspondent dans la classe "Age" ("Range" de la propriété).

3.9 **Modèles de fusion :** Nous avons défini dix-neuf modèles pour la fusion des données, comme présenté sur le Tableau 3.1. Ces modèles décrivent des exemples de commandes faites par l'utilisateur. Chaque modèle a sa propre requête définie dans l'onglet SWRL de Protégé, ces requêtes sont presentées dans l'Annexe I. En effet, nous avons choisi d'utiliser la fusion à base de règles. Pour ce faire, nous devons définir des modèles de scénarios possibles de commandes émises par l'utilisateur et les définir dans la base de connais-

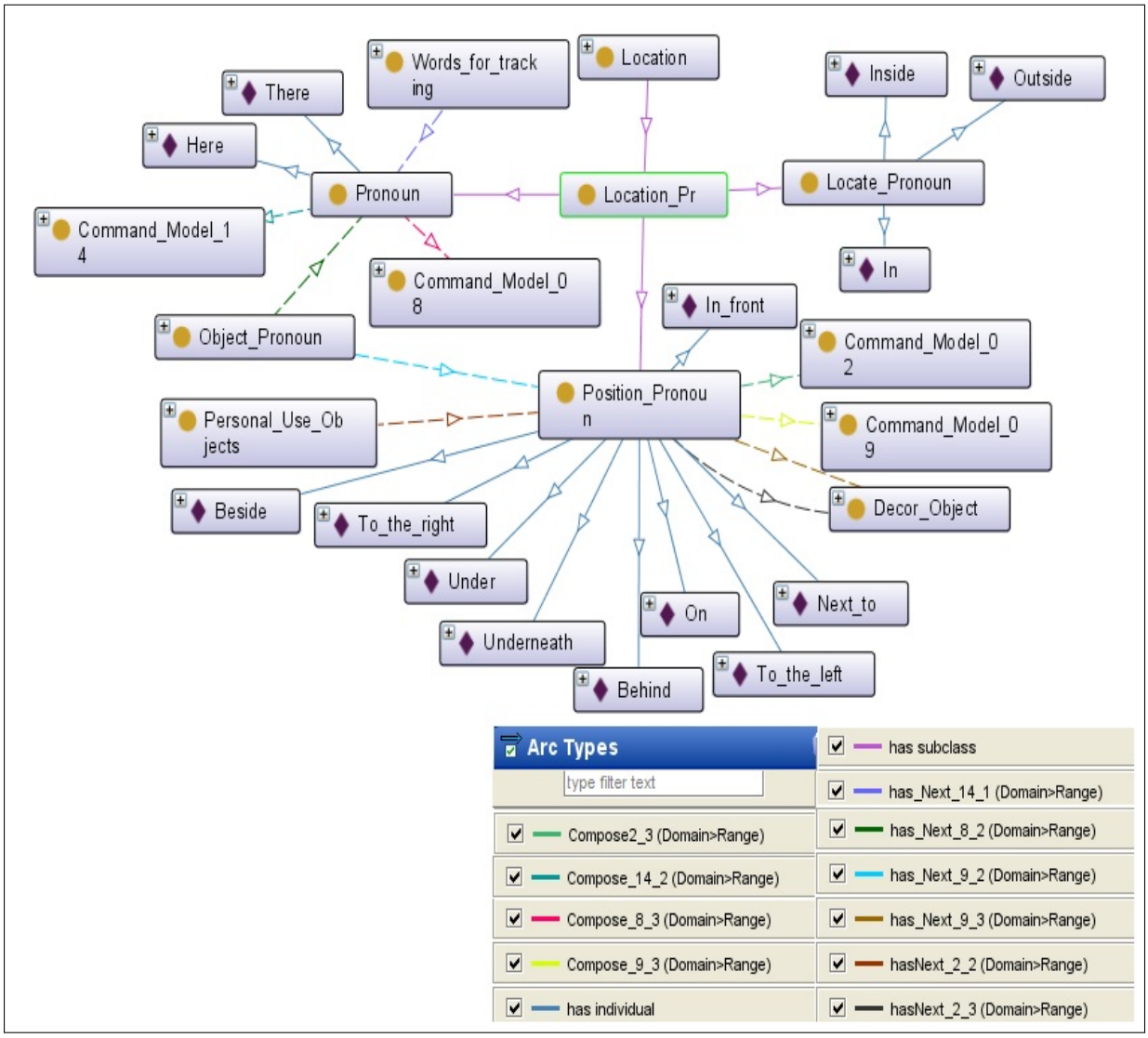


FIGURE 3.50: La classe "Pronoms pour la localisation" (Location pronoun) et ses instances

sance (qui est l'ontologie). Ces scénarios seront accessibles par le moteur de fusion en utilisant des requêtes SQWRL. Les modèles seront composés d'une succession de classe appartenant à l'ontologie. Les classes de chaque modèle seront choisies selon des exemples de commande qui permettent une succession correcte de modalités.

La deuxième colonne du tableau 3.1 correspond à l'ordre des modèles de fusion. Lorsqu'une commande est détectée par le système, elle est sous forme de commandes élémentaires, et chaque commande élémentaire appartient à une classe de l'ontologie. Pour que la fusion ait lieu, il faut que la commande ait un sens. Pour vérifier cela, nous avons défini les modèles de fusion avec un

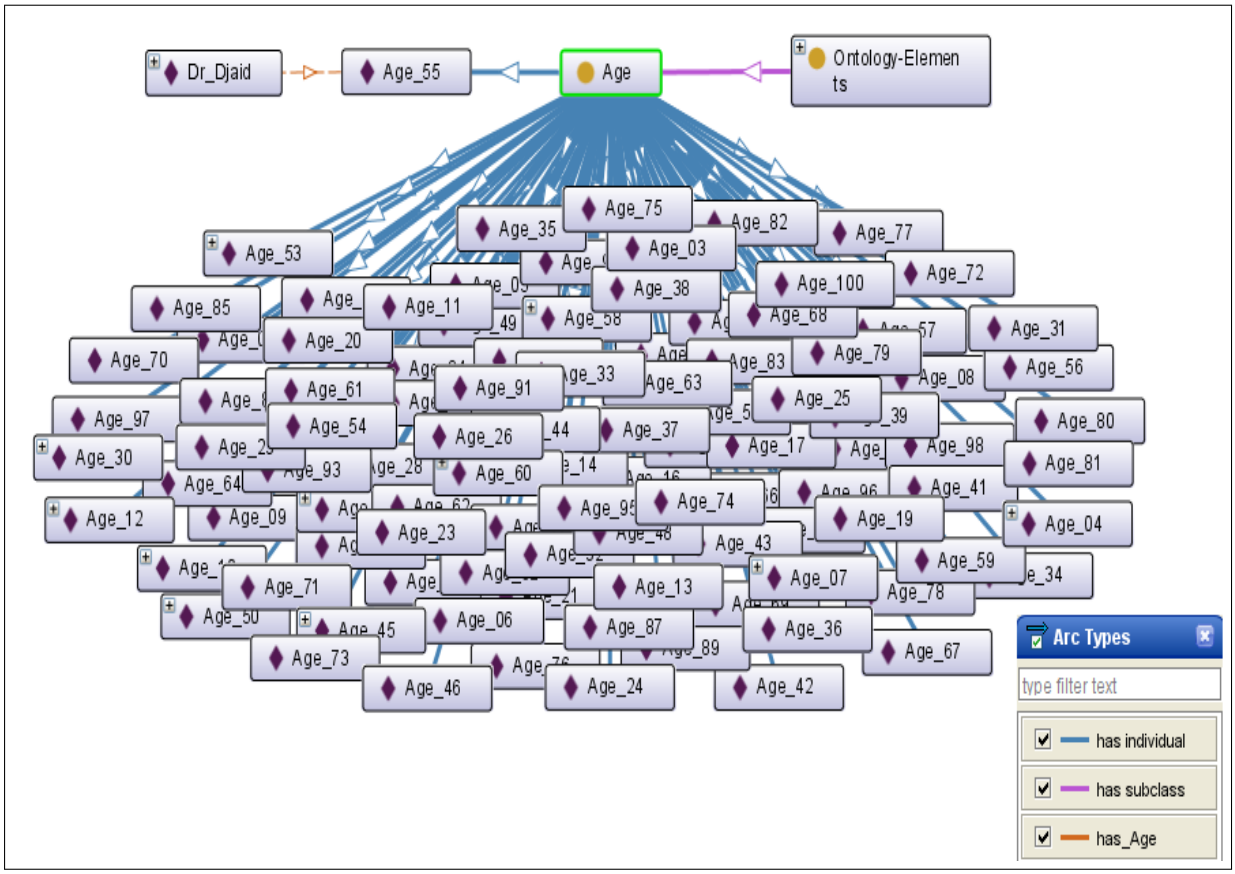


FIGURE 3.51: La classe "Age" et la propriété d'objet "hasAge"

ordre précis pour chaque modèle. Chaque classe appartenant à un modèle sera reliée à la classe qui la suit, dans l'ordre, par une propriété d'objet (hasNext1 .... hasNextn).

Par exemple, la commande Retourne-moi là-bas "Return me there" doit d'abord figurer dans le vocabulaire de l'ontologie (présence des modalités dans les différentes classes). Cette commande doit aussi satisfaire l'ordre d'un modèle de commande parmi les dix-neuf modèles prédéfinis dans l'ontologie. L'exemple "Retourne-moi là-bas" sera reconnu comme étant un exemple d'une commande qui satisfait le quatrième modèle du Tableau 3.1. La classe modèle 4 est composée des classes "words for tracking" (mots pour le suivi), qui est la sous classe de la classe "Vocabulary" (vocabulaire). La classe "Objective" qui est la sous classe de la classe "Individuals" (individu) et, enfin, la classe "Location" (l'emplacement) qui est la sous-classe de la classe "sur-

roundings" (alentours).

De plus, nous avons défini 19 propriétés d'objet qui relient les sous-classes de chaque modèle pour permettre la définition de l'ordre des évènements détectés. En effet, l'ordre d'arrivée des évènements est primordial lors du traitement de la commande, car il peut changer la signification de la commande. Ces propriétés ont la forme "hasNext_a_n" ou "a" est le numéro du modèle et "n" varie de 1 jusqu'au nombre de classes appartenant à chaque modèle. Ces propriétés d'objet définies dans Protégé ont comme **Domain** et **Range** les classes du modèles. Le Domain de chaque propriété est la classe du départ de la flèche et le Range est la classe de destination de la flèche, comme presente dans le tableau 3.1. Par exemple, pour le modèle 11, le Domain de hasNext11_1 est "Words_Person_2" et le Range est la classe "Individuals". Pour ce qui est de la propriété hasNext11_2 le Domain est la classe "Individuals" et le Range est "Personal_use_objects".

Lors de la détection d'un événement, chaque partie doit être une instance des classes du modèle pour pouvoir satisfaire un scénario précis. Pour l'exemple mentionné ci-dessus, le système doit détecter d'abord le mot "Return" (Retourne) comme étant un mot pour le suivi, le mot "me" (moi) comme étant un pronom personnel et enfin le mot "there" (là-bas) comme étant la destination. L'ordre des événements doit être satisfait, dans ce cas il correspond à l'ordre du modèle 4. Sinon l'événement sera rejeté car il ne correspond pas à un modèle prédéfini dans l'ontologie. Un exemple du modèle 4 est présenté sur la Figure 3.52.

Tableau 3.1: Modèles de fusion définies dans l'ontologie

Classe	Composition, Ordre et Propriétés d'objet	Exemples
Modèle 1	Mots pour les personnes_2 $\xrightarrow{hasNext1\_1}$ Possessive $\xrightarrow{hasNext1\_2}$ personnes $\xrightarrow{hasNext1\_3}$ objets à usage personnel (Words person_2 $\xrightarrow{hasNext1\_1}$ Possessive $\xrightarrow{hasNext1\_2}$ People $\xrightarrow{hasNext1\_3}$ Liquids Objects)	Donne à sa mère un verre (Give her mother a cup)
Modèle 2	Mots objets à usage personnel $\xrightarrow{hasNext2\_1}$ objets à usage personnel $\xrightarrow{hasNext2\_2}$ pronom pour position $\xrightarrow{hasNext2\_3}$ objets de décore (Words personal use objects $\xrightarrow{hasNext2\_1}$ personal use objects $\xrightarrow{hasNext2\_2}$ Position pronoun $\xrightarrow{hasNext2\_3}$ Décor object)	Pose la bouteille sur la table (Put bottle on the table)
Modèle 3	Mots pour les personnes_1 $\xrightarrow{hasNext3\_1}$ individus (Words for persons_1 $\xrightarrow{hasNext3\_1}$ Individuals)	Appelle mon père (Call father)
Modèle 4	Mots pour le suivi $\xrightarrow{hasNext4\_1}$ Objective $\xrightarrow{hasNext4\_2}$ Endroits (Words for tracking $\xrightarrow{hasNext4\_1}$ Objective $\xrightarrow{hasNext4\_2}$ Places)	Emmène moi à la chambre (Take me to the room)
Modèle 5	Mots objets à usage personnel $\xrightarrow{hasNext5\_1}$ objets à usage personnel(Words personal use objects $\xrightarrow{hasNext5\_1}$ Personal use objects)	Ramène le stylo(Bring the pen)
Modèle 6	Mots personne_2 $\xrightarrow{hasNext6\_1}$ objets à usage personnel $\xrightarrow{hasNext6\_2}$ Individus(Words person_2 $\xrightarrow{hasNext6\_1}$ Personal use objects $\xrightarrow{hasNext6\_2}$ Individuals)	Donne les clés à mon père (Give the keys to father)

Modèle 7	Mots objets à usage personnel $\xrightarrow{hasNext7\_1}$ Objets à usage personnel $\xrightarrow{hasNext7\_2}$ Endroits (Words personal use objects $\xrightarrow{hasNext7\_1}$ Personal use objects $\xrightarrow{hasNext7\_2}$ Places)	Emmène l'assiette à la cuisine (Take the plate to the kitchen)
Modèle 8	Mots objets à usage personnel $\xrightarrow{hasNext8\_1}$ Pronom d'objets $\xrightarrow{hasNext8\_2}$ Pronom de localisation (Words personal use objects $\xrightarrow{hasNext8\_1}$ Object pronoun $\xrightarrow{hasNext8\_2}$ Pronoun)	Pose ca ici (Put that here)
Modèle 9	Mots objets à usage personnel $\xrightarrow{hasNext9\_1}$ Pronom d'objets $\xrightarrow{hasNext9\_2}$ Pronom pour la position $\xrightarrow{hasNext9\_3}$ Objets de décore (Words personal use objects $\xrightarrow{hasNext9\_1}$ Object pronoun $\xrightarrow{hasNext9\_2}$ Position pronoun $\xrightarrow{hasNext9\_3}$ Décor object)	Pose cela sous la table (Put that under the table)
Modèle 10	Mots objets utiles $\xrightarrow{hasNext10\_1}$ Pronom d'objets (Words convenient object $\xrightarrow{hasNext10\_1}$ Object pronoun)	Prend ceci (Take this)
Modèle 11	Mots pour personne_2 $\xrightarrow{hasNext11\_1}$ Individus $\xrightarrow{hasNext11\_2}$ Objets à usage personnel (sans la classe "Liquide") (Words person_2 $\xrightarrow{hasNext11\_1}$ Individuals $\xrightarrow{hasNext11\_2}$ Personal use objects(without the class"Liquid"))	Donne à mon frère un stylo (Give my brother a pen)
Modèle 12	Mots meubles légers $\xrightarrow{hasNext12\_1}$ Meubles légers (Words lightweight furniture $\xrightarrow{hasNext12\_1}$ Lightweight furniture)	Trouve un tabouret (Find stool)

Modèle 13	Mots utilisés pour les parties mobiles du bâtiment $\xrightarrow{hasNext13\_1}$ L'intérieur $\xrightarrow{hasNext13\_2}$ Objets a partie mobile du bâtiment (Words used for movable P objects $\xrightarrow{hasNext13\_1}$ Indoors $\xrightarrow{hasNext13\_2}$ Movable parts of buildings)	Ferme la porte de la chambre (Close room's door)
Modèle 14	Mots pour le suivi $\xrightarrow{hasNext14\_1}$ Pronom d'objet $\xrightarrow{hasNext14\_2}$ Places (Words for tracking $\xrightarrow{hasNext14\_1}$ Object pronoun $\xrightarrow{hasNext14\_2}$ Places)	Retourne ça là-bas (Return this there)
Modèle 15	Mots pour liquide $\xrightarrow{hasNext15\_1}$ Liquide(Words liquid $\xrightarrow{hasNext15\_1}$ Liquid)	Apporte de l'eau (Bring water)
Modèle 16	Mots pour suivi $\xrightarrow{hasNext16\_1}$ Objective $\xrightarrow{hasNext16\_2}$ Moyens de transport (Words for tracking $\xrightarrow{hasNext16\_1}$ Objective $\xrightarrow{hasNext16\_2}$ Transport means)	Emmène moi à la voiture (Take me to the car)
Modèle 17	Mots objets à parties mobiles $\xrightarrow{hasNext17\_1}$ Objets à parties mobiles (Words used movables P objects $\xrightarrow{hasNext17\_1}$ Movable parts objects)	Ferme la fenêtre (Close window)
Modèle 18	Mots grands objets électriques $\xrightarrow{hasNext18\_1}$ Grands objets électriques (Words big electrical objects $\xrightarrow{hasNext18\_1}$ Big electrical object)	Allume la télé (Turn on TV)
Modèle 19	Mots personne_2 $\xrightarrow{hasNext19\_1}$ Individus $\xrightarrow{hasNext19\_2}$ Liquide (Words Person_2 $\xrightarrow{hasNext19\_1}$ Individuals $\xrightarrow{hasNext19\_2}$ Liquid)	Donne à ma mère du jus (Give mother some juice)

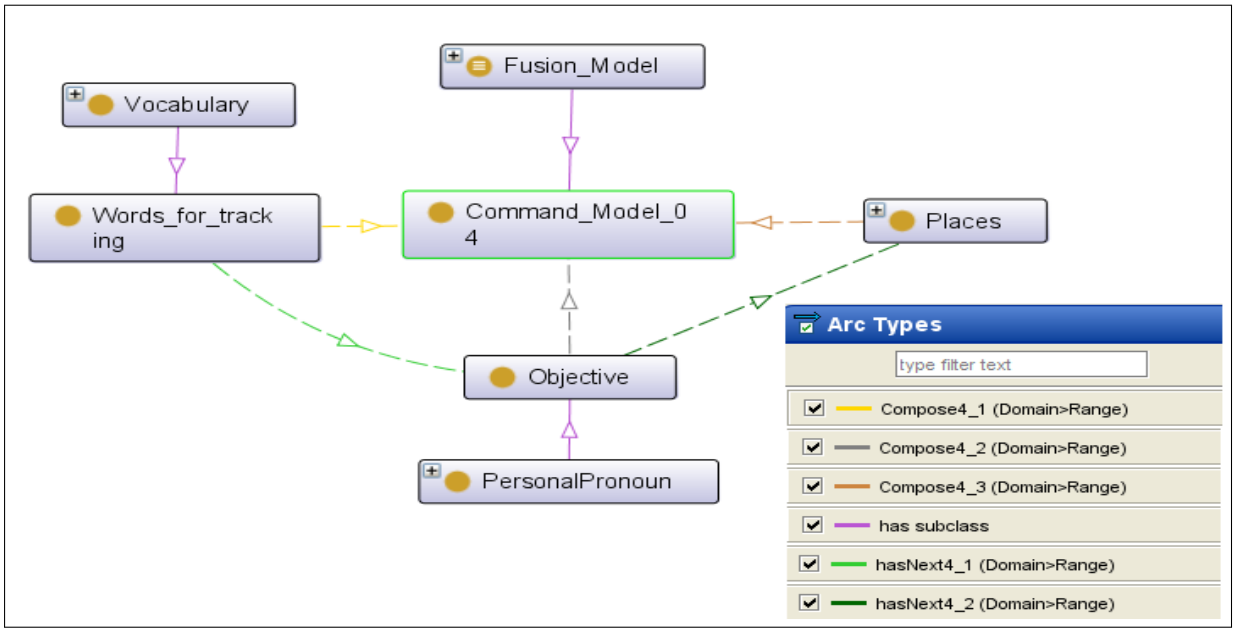


FIGURE 3.52: Exemple du modèle 4

3.10 **Motifs de fission :** Le moteur de fission est aussi important que le moteur de fusion. En effet, le moteur de fission subdivise les résultats du moteur de fusion en sous-tâches élémentaires et les envoie aux modalités de sortie disponibles tout en prenant en compte l'état du contexte. La subdivision est faite selon des motifs prédéfinis dans l'ontologie. Les commandes de base seront envoyées aux modalités de sortie en fonction de leurs disponibilités. Les motifs de fission sont définis dans l'ontologie en deux parties, à savoir un problème et une solution. Le problème est la commande émise par l'utilisateur et obtenue par le moteur de fusion tandis que la solution se compose de toutes les sous-tâches possibles pour cette commande.

Pour le processus de fission, nous avons défini neuf solutions possibles dans l'ontologie et sont présentées dans le Tableau 3.2. Les requêtes de sélection des solutions de fission sont présentées dans l'Annexe I. Nous avons défini la classe "Fission solution", présentée sur la Figure 3.53, et les sous classes "Fisson solution 1" jusqu'à "fission solution 9". Par exemple, la classe "Fission solution 1" est la solution pour les modèles 1, 6 et 11 du moteur de fusion. La demande "donne-moi un verre " sera exécutée par les taches suivantes : Se

déplacer vers la position → prendre l'objet → se déplacer vers la position de la personne → déposer l'objet.

Aussi, nous avons défini la classe "fission tasks" qui a comme instances les différentes taches à exécuter par les actionneurs de sortie. Chaque classe de solution est désignée par des taches à exécuter. L'ordre d'exécution est établie par la création d'une propriété d'objet "has_Next_F" qui permet de relier les instances de chaque solution et l'instance qui la suit dans chaque solution et qui est dans la classe "Fission tasks", un exemple est présente sur la Figure 3.54. Le "Domain" de la propriété "has_Next_F" est la classe "Fission tasks" et le "Range" est la classe "Fission Model".

Tableau 3.2: Les solutions de fission définies dans l'ontologie

Classe	**Sous classes**	**Propriétés d'objet**	**Modèle de fusion correspondant**
Solution 1	Se déplacer à la position de l'objet $\xrightarrow{has\_Next\_F}$ Prendre l'objet $\xrightarrow{has\_Next\_F}$ se déplacer à la position de la personne $\xrightarrow{has\_Next\_F}$ déposer l'objet (Move to object position $\xrightarrow{has\_Next\_F}$ Grab object $\xrightarrow{has\_Next\_F}$ Move to people position $\xrightarrow{has\_Next\_F}$ Put object)	has_Next_F	1, 6 and 11
Solution 2	Se déplacer à la position de l'objet $\xrightarrow{has\_Next\_F}$ Prendre l'objet $\xrightarrow{has\_Next\_F}$ se déplacer à la localisation $\xrightarrow{has\_Next\_F}$ déposer l'objet (Move to object position $\xrightarrow{has\_Next\_F}$ Grab object $\xrightarrow{has\_Next\_F}$ Move to Location $\xrightarrow{has\_Next\_F}$ Put object)	has_Next_F	2 and 7

Solution 3	Trouver la personne $\xrightarrow{has\_Next\_F}$ répondre à la requête (Find person $\xrightarrow{has\_Next\_F}$ answer request)	has_Next_F	3
Solution 4	Détecter la position $\xrightarrow{has\_Next\_F}$ si la porte est ouverte $\xrightarrow{has\_Next\_F}$ se deplacer vers la position (Detect position $\xrightarrow{has\_Next\_F}$ If door open $\xrightarrow{has\_Next\_F}$ Move to position) Détecter la position $\xrightarrow{has\_Next\_F}$ si la porte est fermee $\xrightarrow{has\_Next\_F}$ ouvrir la porte $\xrightarrow{has\_Next\_F}$ se deplacer vers la position (Detect position $\xrightarrow{has\_Next\_F}$ If door shut $\xrightarrow{has\_Next\_F}$ Open door $\xrightarrow{has\_Next\_F}$ Move to position)	has_Next_F	4,14 and 16
Solution 5	Se déplacer vers la position de l'objet $\xrightarrow{has\_Next\_F}$ prendre l'objet (Move to object position $\xrightarrow{has\_Next\_F}$ Grab object)	has_Next_F	5 and 10
Solution 6	Se déplacer vers la position de l'objet $\xrightarrow{has\_Next\_F}$ répondre à la requête (Move to position $\xrightarrow{has\_Next\_F}$ Answer request)	has_Next_F	12, 13 and 17

Solution 7	Se déplacer vers la position $\xrightarrow{has\_Next\_F}$ prendre l'objet pour liquide $\xrightarrow{has\_Next\_F}$ verser liquide $\xrightarrow{has\_Next\_F}$ Se déplacer vers la position de la personne $\xrightarrow{has\_Next\_F}$ Déposer l'objet pour liquide (Move to position $\xrightarrow{has\_Next\_F}$ Grab liquid object $\xrightarrow{has\_Next\_F}$ Pour liquid $\xrightarrow{has\_Next\_F}$ Move to people position $\xrightarrow{has\_Next\_F}$ Drop liquid object)	has_Next_F	15 and 19
Solution 8	Trouve objet $\xrightarrow{has\_Next\_F}$ répondre à la requête (Find object $\xrightarrow{has\_Next\_F}$ Answer request)	has_Next_F	18
Solution 9	Se déplacer vers la position de l'objet $\xrightarrow{has\_Next\_F}$ prendre l'objet $\xrightarrow{has\_Next\_F}$ détecter la position $\xrightarrow{has\_Next\_F}$ se déplacer vers la position $\xrightarrow{has\_Next\_F}$ Déposer l'objet (Move to object position $\xrightarrow{has\_Next\_F}$ Grab object $\xrightarrow{has\_Next\_F}$ Detect position $\xrightarrow{has\_Next\_F}$ Move to position $\xrightarrow{has\_Next\_F}$ Drop object)	has_Next_F	8 and 9

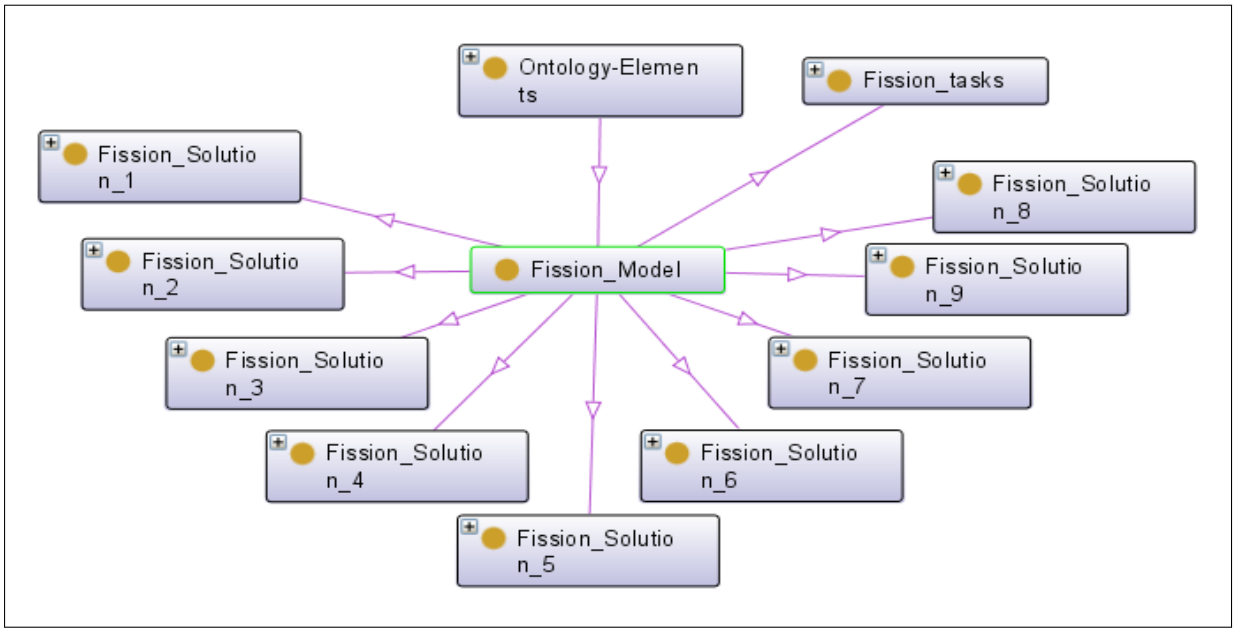


FIGURE 3.53: Solution de Fission.

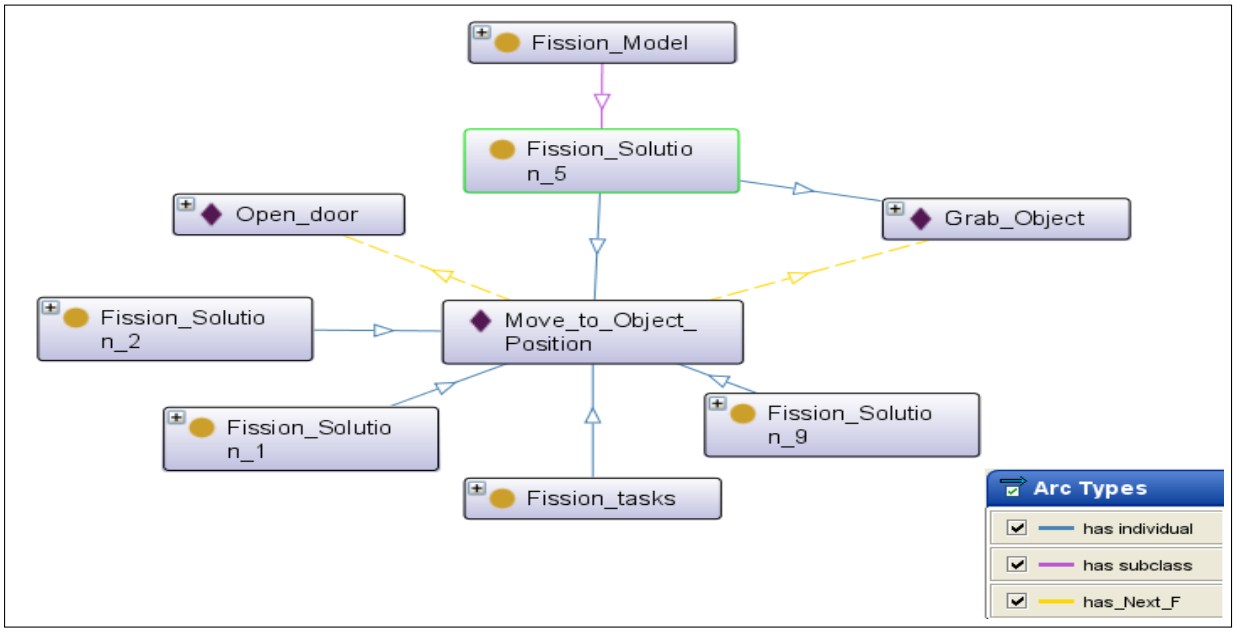


FIGURE 3.54: Exemple de la solution de fission 5.

3.5.2 Les propriétés

Les propriétés permettent de définir les relations qui existent entre les entités de l'ontologie. Ces relations sont très importantes car elles permettent de lier les différentes parties de l'ontologie. Nous avons utilisé dans notre travail deux types de propriétés ; les

propriétés d'objets qui permettent de lier les classes ou les individus entre eux, et les propriétés de données qui permettent de lier les individus à des valeurs. La Figure 3.55 montre un exemple de propriétés d'objets et de données de la classe "Time" (temps).

1. **Propriété d'objets**

   - **Has Modality Time** : Relie la classe "Input Modality" (modalité d'entrée) et ses sous classes avec l'instance "Modality Time" (temps de modalité). Chaque modalité ne doit pas dépasser le temps maximum défini pour sa détection. Le "Domain" de cette propriété est la classe "Input Modality" et le "Range" est la classe "Time".
   - **Has Model Time** : Relie la classe "fusion Model" (modèle de fusion) avec l'instance "Command Time" (temps de la commande). Le temps d'exécution du modèle de fusion ne doit pas dépasser le temps maximum de la commande. Le "Domain" de cette propriété est la classe "fusion Model" et le "Range" est la classe "Time".
   - **Needs liquid object** : Relie les instances de la classe "Liquid" (Liquide) avec les instances de la classe "Liquids Objects" (objets pour liquide). Lorsque l'utilisateur demande une boisson liquide, il est indispensable d'associer au liquide un verre ou une tasse etc. Le "Domain" de cette propriété est la classe "Liquid" et le "Range" est la classe "Liquids Objects".
   - **Is Affected By** : Relie les instances des sous classes de la classe "Input Modality" (modalité d'entrée) et les instances de la classe "Output Modality" (modalité de sortie) avec certaines instances des sous classes de la classe "Context". Par exemple, elle relie l'instance "Voice Sensor" (capteur de voix) avec les instances, qui peuvent influencer son utilisation, comme "Mute" (Muet) et "Deaf" (Sourd) et "Noise Level" (niveau de bruit) pour savoir si la modalité sera désactivée ou pas. Le "Domain" de cette propriété est les classes "Input Modality" et "Output Modality", et le "Range" est la classe "Context".

- **Is Activated By** : Relie les instances de la classe "Alarme" avec les instances de la classe "Health context" (contexte de santé) et "System context" (contexte du système). Le "Domain" de cette propriété est la classe "Alarme" et le "Range" est les classes "Health context" et "System context".
- **Is detected By** : Relie les instances de la classe "Surroundings" (entourage) et ses sous classes aux instances de la classe "Modality" (modalité) et ses sous classes. Le "Domain" de cette propriété est la classe "Surroundings" et le "Range" est le classe "Input Modality".
- **Is same as** : Relie les instances de la classe "Personal pronoun" (pronom personnel) aux instances des classes "Personal assistance" (assistance personnelle), "Peoples names" (nom des personnes) et "People" (personnes). Le "Domain" de cette propriété est la classe "Personal pronoun" et le "Range" est les classes "Personal assistance", "Peoples names" et "People".
- **has Name** : Relie les instances des classes "People" (personnes), "Personal assistance" (assistance personnelle) et "User information" (information sur l'utilisateur), qui sont le "Domain", aux instances de la classe "Peoples names" (nom des personnes), qui est le "Range".
- **has Age** : Relie les instances des classes "User information" (information sur l'utilisateur) et "Peoples names" (nom des personnes) (qui sont le "Domain"), aux instances de la classe "Age" (Range).
- **has Location** : Relie les instances de la classe "Personal use objects" (Objets à usage personnel) aux instances de la classe "Indoors" (l'intérieur). Le "Domain" est la classe "Personal use objects" et le "Range" est la classe "Indoors".
- **has Next N_X** : Est utilisée pour définir l'ordre au sein des modèles de fusion (ou N représente le numéro du modèle est X varie selon le nombre de classes qui composent le modèle). Le "Domain" et "Range" de chaque propriété sont les classes qui composent le modèle dans l'ordre.
- **Compose N_X** : Est utilisée pour définir les classes qui composent chaque

modèle de fusion (ou N représente le numéro du modèle est X varie selon le nombre de classes qui composent le modèle). Le "Domain" est composé de classes de l'ontologie et "Range" est le modèle N.

2. **Propriété de type de données**

- **Has State** : Propriété de type de données pour la classe "movable parts objects" (objet à partie mobile), qui est le "Domain". Elle permet de donner l'état de de la partie mobile de l'objet (ouvert ou fermé). Elle a comme type "string".
- **Has Time Start / has Time End** : Propriété de type de données qui permet de définir le temps de début et d'arrêt de la modalité, qui a comme "Domain" la classe "Input Modality". Elle a comme type "int".
- **Has Answer** : Propriété de type de données pour la classe "weather condition" (Condition météorologique), qui est le "Domain". Elle a comme type de données "int".
- **has Battery Level** : Propriété de type de données pour l'instance de la classe "Battery Level" (niveau de batterie), qui est le "Domain". Elle permet de fournir le niveau de la batterie du système et a comme type de données "int".
- **has Blood Pressure** : Propriété de type de données pour la classe "Blood Pressure"(tension artérielle) de type de données " float ". Elle a deux sous propriétés :
  - ▷ **has Diastolic Pressure** : Propriété de type de données pour l'instance "Diastolic Pressure" (tension diastolique) de la classe "Blood Pressure D" (tension artérielle), qui est le "Domain". Elle est de type de données " float ".
  - ▷ **has Systolic Pressure** : Propriété de type de données pour l'instance " Systolic Pressure" (tension systolique) de la classe "Blood Pressure S" (tension artérielle), qui est le "Domain". Elle a comme type de données " float ".

- **has Degree** : Propriété de type de données des instances des classes "Temperature", qui est le "Domain", de type "float".
- **has Detected Inf / has Detected Ult** : Propriété de type de données des instances des classes "Infrared Sensor" (capteur infrarouge) et "Ultrasonic Sensor" (capteur ultrasonique) (qui sont le "Domain", respectivement) de type "int". Cette propriété permet d'obtenir des informations sur les capteurs du fauteuil (détection ou non détection d'obstacle).
- **has Disability** : Propriété de type de données pour les instances de la classe "Handicap Type" (Type de handicap), qui est le "Domain", de type "boolean". Cette propriété permet de savoir le type de handicap de l'utilisateur et ainsi désactiver certaines modalités d'entrée.
- **has Eye Mouvement** : Propriété de type de données pour le capteur du regard de la classe "Visual"(visuelle), qui est le "Domain", de type réel "float". Cette propriété permet de recevoir les informations du capteur du regard.
- **has Gesture Position** : Propriété de type de données de type "boolean" pour la classe "Gesture" (geste), qui est le "Domain", et son instance.
- **has Light** : Propriété de type de données de la classe "Lighting level" (niveau de luminosité), qui est le "Domain". Elle permet de récupérer le niveau de la lumière ambiante. Cette propriété a le type de données "float".
- **has Noise** : Propriété de type de données de la classe "Noise" (bruit), qui est le "Domain". Elle permet de connaitre le niveau de bruit ambiant. Cette propriété a le type de données " float".
- **has Max Time** : Propriété de type de données des instances de la classe "Time" (temps), qui est le "Domain". Elle permet de connaitre le temps maximum d'une commande et le temps maximum entre deux modalités. Cette propriété a le type de données "int".
- **has Temperature** : Propriété de type de données de la classe "Body Temperature" (température corporelle), qui est le "Domain", et son instance. Cette propriété est de type "float" et donne la température corporelle de l'utilisateur.

- **has Touch** : Propriété de type de données de la classe "Touch" (tactile), qui est le "Domain", et son instance. Elle fournit le positon détecté par l'écran tactile. Cette propriété est de type "float".
- **Is Available** : Propriété de type de données de la classe "Output Modality" (modalité de sortie), qui est le "Domain". Cette propriété est de type "int".

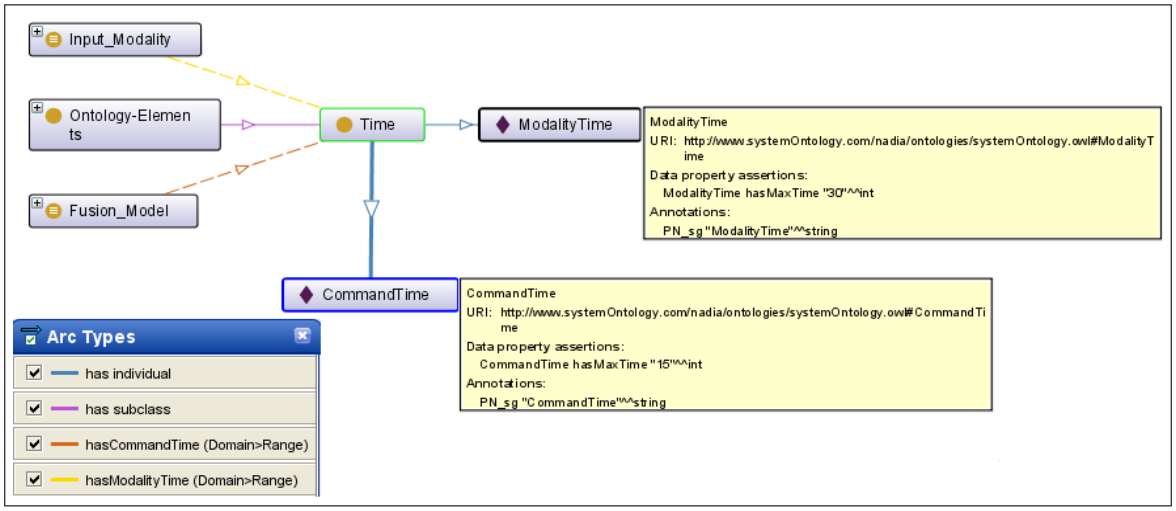


FIGURE 3.55: Exemple des propriétés d'objets et de type de données de la classe "Time" (temps).

3.6 Conclusion

En effet, le concept d'ontologie nous a permis de définir toutes les entités qui forment l'environnement du système ainsi que les relations entre elles en utilisant les propriétés. Cette ontologie ainsi définie sera la base de connaissance du système final. Nous avons intégré toutes les entités qui, à notre avis, pourront être présentes dans l'environnement de l'utilisateur du système multimodal (fauteuil roulant muni d'un bras manipulateur). Néanmoins, l'intégration ou la suppression d'un élément est possible à n'importe quel moment selon l'état de l'environnement ou l'utilisation par un autre utilisateur. Cette caractéristique est un des avantages de l'utilisation du concept d'ontologie comme base de connaissance car elle assure l'adaptabilité de l'architecture à n'importe quel environnement et n'importe quelle application.

Chapitre 4

Processus de Fusion et de Fission

4.1 Introduction

Nous avons présenté dans les chapitres précédents les différents composants de l'architecture que nous proposons pour faciliter l'interaction multimodale. Nous allons présenter dans le présent chapitre le fonctionnement des différentes étapes qui aboutissent à l'offre de service suite à une requête de l'utilisateur du système. Ces étapes sont la sélection de modalités, le processus de fusion et enfin le processus de fission.

Tout d'abord, nous allons nous concentrer sur la sélection de modalités qui se fera en deux parties. La première étant la sélection de modalités d'entrée selon l'état du contexte. Cette opération va permettre au système de procéder directement à la réception d'évènements selon la disponibilité des modalités d'entrée. Puis, le système procédera à la fusion des données.

Le moteur de fusion est une partie très importante dans un système multimodal. Il permet la fusion des modalités d'entrée récupérées de l'environnement pour répondre à la requête de l'utilisateur. Lorsque l'utilisateur du système fait une demande de service, le système détecte les évènements et les fusionne tout en prenant en compte l'état du contexte sémantique pour répondre à la requête de l'utilisateur. Le moteur de fusion présenté dans ce travail est basé sur le mécanisme de règles. Ce dernier est souvent utilisé pour mieux estimer un objet en mouvement et permet d'avoir un bon alignement temporel entre les modalités.

Nous allons présenter aussi dans ce chapitre le moteur de fission qui est aussi très important dans une architecture multimodale. Le moteur de fission est la partie responsable de la subdivision du résultat du moteur de fusion en actions élémentaires compréhensibles par les actionneurs de sortie. Le moteur présenté dans ce chapitre est aussi basé sur le mécanisme de règles.

Puis, lorsque toutes ces étapes seront exécutées, le système procédera à la sélection des modalités de sortie selon l'état du contexte et décidera des modalités qui seront utilisées pour répondre à la requête de l'utilisateur.

4.2 Sélection de modalités :

La multimodalité consiste en la combinaison de plusieurs modalités dans le but d'interagir avec une entité ou pour appréhender un phénomène. Les applications d'interaction multimodales ont été démontrées pour offrir une meilleure flexibilité et fiabilité par rapport aux autres systèmes d'interaction homme/machine [76]. Par conséquent, ces applications sont préférées par les utilisateurs aux applications d'interaction uni-modales [77] ; Ainsi, les systèmes multimodaux représentent une nouvelle classe de systèmes d'interaction capables d'interpréter les informations provenant de divers canaux sensoriels. Ils utilisent une manière plus riche et plus naturelle de communication, tels que la parole ou le geste, et plus généralement tous les cinq sens. Ces systèmes peuvent également intégrer les compétences de calcul des ordinateurs dans le monde réel, en offrant des moyens plus naturels d'interaction pour les humains. Généralement dans le domaine de la robotique mobile, la fusion se fait au niveau du signal et permet d'estimer les positions et les angles du robot mobile pour une meilleure localisation.

De nombreux chercheurs ont travaillé sur la fusion des capteurs ([39],[40] et [41]) en utilisant une variété de capteurs, car la fusion effective des données provenant de capteurs est très importante dans l'augmentation de la capacité du système robotique à accomplir des tâches complexes. Des processus de fusion-fission combinés à des systèmes interactifs appropriés sont nécessaires pour être mis en œuvre et formellement modélisés pour la validation. De plus, une machine qui imite les compétences de com-

munication humaine doit être en mesure de faire face aux différentes modalités d'entrée et de sortie. La combinaison et l'interprétation de multiples modalités sont possibles en utilisant des moteurs de fusion et de fission.

L'utilisation de multiples modalités dans des environnements dynamiques par des systèmes robotiques nécessite une sélection selon l'état évolutif de l'environnement. En effet, l'être humain s'adapte à son environnement naturellement et sélectionne le moyen le plus approprié selon les conditions environnementales. Par exemple, si l'environnement est sombre, l'être humain n'utilisera automatiquement pas le geste pour désigner un objet ou un emplacement. Il décidera par lui-même quelle est la meilleure modalité à utiliser pour designer l'objet ou l'emplacement. Cette faculté naturelle possédée par l'être humain doit être intégrée dans le robot d'interaction homme/machine. Pour ce faire, nous avons choisi d'utiliser des règles de sélection de modalités selon l'état de l'environnement sémantique. Cette intégration est introduite dans la base de connaissance du système (Ontologie). Sachant que nous avons construit notre ontologie en utilisant l'outil Protégé, nous avons utilisé l'onglet SWRL de Protégé pour définir nos conditions de sélection de modalités d'entrée et de sortie en utilisant le langage SWRL pour les règles et SPARQL pour les requêtes.

4.2.1 Sélection de modalités d'entrée :

Les modalités d'entrée utilisées par notre système sont : le geste, la parole, le visuel, le tactile et le manuel. Cependant, ces modalités peuvent être affectées par de nombreux facteurs issus de l'environnement ; tels que la luminosité, le bruit, les conditions météorologiques ainsi que le type de handicap de l'utilisateur. En effet, chaque modalité peut être affectée par un seul ou plusieurs facteurs et doit impérativement être désactivé car le système ne pourra pas prendre en compte la commande correctement et le facteur d'erreur sera très important. Les requêtes de sélection de services utilisées et définies dans Protégé sont présentées dans le Tableau 4.1 ci-dessous.

Nous avons utilisé la notion de "namespace" ou "systemeOntology" est le nom de notre ontologie. Par exemple, pour les conditions météorologiques de la première ligne

du tableau 4.1, les parties "Input_Modality" et "Weather_Condition" sont des classes de l'ontologie. La propriété d'objet "IsAffectedBy" relie les deux classes. Cette propriété a comme Domain la classe "Input_Modality" et la classe "Context" comme Range. Si une variable "m" (instance) appartient à la classe "Input_Modality" et elle est reliée à la variable "w" (instance) de "Weather_condition" par la propriété d'objet "IsAffectedBy", on procède à la vérification de la valeur de "w". Si cette dernière est égale à 1 (en utilisant la commande "swrl :equal"), présence de pluie ou de neige, cela implique que la modalité ne peut être utilisée car les conditions météorologiques ne le permettent pas (Désactivation de la modalité).

La commande "sqwrl :selectDistinct" permet de prendre un ou plusieurs arguments, qui sont généralement des variables utilisées dans la spécification de modèle de la requête dans l'ontologie, et construit une table en utilisant les arguments en tant que colonnes du tableau. Le mot clé "DISTINCT" élimine les lignes dupliquées des résultats d'une instruction "SELECT". Si "DISTINCT" n'est pas spécifié, toutes les lignes sont renvoyées, y compris des duplications.

En effet, les informations obtenues à partir de la surveillance continue de l'environnement seront comparées aux informations définies comme limites acceptables dans l'ontologie. Ces limites sont de 60 dB pour le bruit maximum accepté et de 40 lux pour le niveau de luminosité minimum accepté. Les valeurs présentées sont des exemples du scenario choisi pour la validation de notre approche et sont modifiables à tout moment. Aussi, les conditions météorologiques affectent l'utilisation du système. Par exemple, sachant que nous avons opté pour l'utilisation de l'option tactile lors de la commande du système, cette possibilité ne pourra pas être utilisée s'il y a de la pluie ou de la neige. Donc si cette valeur est détectée positive par le système la modalité tactile sera désactivée. Enfin, le type de handicap est très important lors de la sélection de modalité. Selon le type de handicap de l'utilisateur certaines modalités seront désactivées, par exemple, si l'utilisateur est muet, la modalité parole sera désactivée.

Cette particularité apporte à notre système une optimisation lors de l'utilisation des modalités. En effet, le système sera en mesure d'utiliser les modalités ou attendre des

informations spécifiques selon la disponibilité des modalités, ce qui rend le système optimal et efficace.

Tableau 4.1: Sélection de modalités d'entrée

Modalité affectée par	**Requête de sélection**
Conditions météorologiques	systemOntology :Input_Modality(?m) ∧ systemOntology :WeatherCondition(?w) ∧ systemOntology :IsAffectedBy(?m, ?w) ∧ systemOntology :hasAnswer(?w, ?Ans) ∧ swrlb :equal(?Ans, 1) → sqwrl :selectDistinct(?m, ?w)
Niveau de luminosité	systemOntology :LightingLevel(?l) ∧ systemOntology :Input_Modality(?m) ∧ systemOntology :IsAffectedBy(?m, ?l) ∧ systemOntology :hasLight(?l, ?Lux) ∧ swrlb :lessThan(?Lux, 40) → sqwrl :selectDistinct(?m, ?l, ?Lux)
Niveau de bruit	systemOntology :Input_Modality(?NotAvailable4In) ∧ systemOntology :Noise(?n) ∧ systemOntology :IsAffectedBy(?NotAvailable4In, ?n) ∧ systemOntology :hasNoise(?n, ?dB) ∧ swrlb :greaterThan(?dB, 60)∧ tbox :isDirectSubClassOf(?Class4In, systemOntology :Input_Modality) ∧ abox :hasIndividual(?Class4In, ?AvailableIn) ∧ tbox :notEqualTo(?NotAvailable4In, ?AvailableIn)→ sqwrl :selectDistinct(?NotAvailable4In, ?n, ?dB, ?AvailableIn)

Type de Handicap	systemOntology :HandicapType(?h) ∧ systemOntology :Input_Modality(?NotActive) ∧ systemOntology :IsAffectedBy(?NotActive, ?h) ∧ systemOntology :hasDisability(?h, ?han) ∧ swrlb :equal(?han, 1) → sqwrl :selectDistinct(?NotActive, ?h, ?han)

Les requêtes de sélection permettent de sélectionner les modalités qui restent sélectionnées et celles à désactiver. Prenons pour exemple les modalités d'entrée qui peuvent être affectées par le niveau de luminosité. En utilisant la deuxième requêtes présentée dans le tableau 4.1, nous pouvons vérifier si le niveau de lumière détecté est inférieur au niveau accepté en utilisant « swrlb : lessThan (?Lux,40) ». Si la condition est satisfaite, toutes les modalités qui dépendent du niveau de luminosité ne seront pas prises en compte et désactivé.

D'autre part, si le lieu est bruyant, le système ne pourra pas utiliser le haut-parleur comme modalité de sortie. Pour vérifier ces informations, nous allons exécuter la requête relative au niveau de bruit (3ème ligne du tableau 4.1) dans l'onglet "swrl" de Protégé, le résultat est présenté sur la Figure 4.1. On suppose que le niveau de bruit (?n) est supérieur à la limite (que nous avons définie à 60 dB), de ce fait la modalité parole détectée par le capteur de voix (Voice sensor) (?m) sera la modalité d'entrée à ne pas prendre en compte et la modalité haut-parleur sera la modalité de sortie non disponible. Le résultat de l'exécution de la requête est présenté sur la Figure 4.1. L'information " ?dB" représente la valeur du bruit détecté par le système.

4.2.2 Sélection de modalités de sortie :

Pour les modalités de sortie nous avons choisi d'utiliser : une sortie visuelle, une sortie vocale et une sortie mobile (le fauteuil ou le bras). Comme pour les modalités d'entrée, les modalités de sortie sont affectées par l'état changeant du contexte sémantique. En effet, les modalités de sortie sont aussi affectées par le niveau de bruit et le niveau de

?NotAvailable4In	?n	?dB	?AvailableIn
systemOntology:Voice_Sensor	systemOntology:Noise_level_Sensor	75	systemOntology:Eye_Gaze_Sensor
systemOntology:Voice_Sensor	systemOntology:Noise_level_Sensor	75	systemOntology:Gesture_Sensor
systemOntology:Voice_Sensor	systemOntology:Noise_level_Sensor	75	systemOntology:Keyboard
systemOntology:Voice_Sensor	systemOntology:Noise_level_Sensor	75	systemOntology:Mouse
systemOntology:Voice_Sensor	systemOntology:Noise_level_Sensor	75	systemOntology:Nonel
systemOntology:Voice_Sensor	systemOntology:Noise_level_Sensor	75	systemOntology:Touch_Screen
?NotAvailable4	?n	?dB	?Available
systemOntology:Speaker	systemOntology:Noise_level_Sensor	75	systemOntology:Moving_Arm
systemOntology:Speaker	systemOntology:Noise_level_Sensor	75	systemOntology:None
systemOntology:Speaker	systemOntology:Noise_level_Sensor	75	systemOntology:Screen
systemOntology:Speaker	systemOntology:Noise_level_Sensor	75	systemOntology:Wheels

FIGURE 4.1: Exemple de sélection de modalités d'entrée et de sortie sur Protégé affectées par le bruit.

lumière, le type de handicap et les conditions météorologiques. Les requêtes de sélection de modalités de sortie sont définies dans le Tableau 4.2 ci-dessous.

Ces requêtes de sélection, sont comme pour les requêtes d'entrée, des requêtes qui permettent de vérifier les relations des modalités de sortie avec les informations du contexte. Si une information du contexte ne correspond pas à l'une des valeurs acceptées (du contexte), ces requêtes vont permettre la désactivation des modalités affectées par cette valeur et la présentation des autres modalités disponibles, comme présenté sur la Figure 4.1.

La troisième ligne du Tableau 4.2 permet de vérifier la disponibilité et la non disponibilité des modalités de sortie affectées par le bruit. Nous avons exécuté cette requête dans l'onglet "swrl" de Protégé, le résultat est présenté sur la Figure 4.1. Nous utilisons le "name space" "sysytem Ontology" qui est le nom de notre ontologie. La partie "systemOntology :Output_Modality(?NotAvailable4)" permet de retourner la variable " ?NotAvailable4" qui est une instances de "Output_Modality" (fait partie de l'ontologie). Nous allons aussi récupérer le niveau de bruit et voir quelle modalité est affectée par le niveau de bruit en vérifiant la connexion par propriété d'objet "IsAffectedBy" qui

relie le niveau de bruit et la modalité de sortie. Nous récupérons le niveau de bruit par la commande "systemOntology :hasNoise(?n, ?dB)", sachant que "has Noise" est une propriété de donnée qui permet de définir le niveau de bruit. Puis, nous allons procéder a la vérification du niveau de bruit grâce à la commande "swrlb :greaterThan(?dB, 60)". Si le niveau de bruit est supérieur au niveau accepté, les modalités affectées par le bruit seront désactivées et affichées dans la colonne " ?NotAvailable4". Les autres (disponibles) seront affichées dans la colonne " ?Available". On conclue que sur la Figure 4.1, la colonne " ?NotAvailable4" retourne les modalités non disponible, la colonne " ?n" retourne la cause de la non disponibilité (dans ce cas le capteur de bruit qui fournit la cause), la colonne " ?db" retourne la valeur du bruit et enfin la colonne " ?Available" retourne les autres modalités disponibles.

Tableau 4.2: Sélection de modalités de sortie

Modalité affectée par	**Requête de sélection**
Conditions météorologiques	systemOntology :Output_Modality(?NotAvailable) ∧ systemOntology :WeatherCondition(?w) ∧ systemOntology :IsAffectedBy(?NotAvailable, ?w) ∧ systemOntology :hasValue(?NotAvailable, ?Ans) ∧ systemOntology :hasAnswer(?w, ?val) ∧ swrlb :equal(?Ans, 1) ∧ swrlb :equal(?val, 1) ∧ tbox :isDirectSubClassOf(?Class, systemOntology :Output_Modality) ∧ abox :hasIndividual(?Class, ?Available) ∧ tbox :notEqualTo(?NotAvailable, ?Available) → sqwrl :selectDistinct(?NotAvailable, ?w, ?val, ?Class, ?Available)

Niveau de luminosité	systemOntology :LightingLevel(?l) ∧ systemOntology :Output_Modality(?NotAvailable) ∧ systemOntology :IsAffectedBy(?NotAvailable, ?l) ∧ systemOntology :hasLight(?l, ?Lux) ∧ swrlb :lessThan(?Lux, 40) ∧ tbox :isDirectSubClassOf(?Class, systemOntology :Output_Modality) ∧ abox :hasIndividual(?Class, ?Available) ∧ tbox :notEqualTo(?NotAvailable, ?Available) → sqwrl :selectDistinct(?NotAvailable, ?l, ?Lux, ?Available)
Niveau de bruit	systemOntology :Output_Modality(?NotAvailable4) ∧ systemOntology :Noise(?n) ∧ systemOntology :IsAffectedBy(?NotAvailable4, ?n) ∧ systemOntology :hasNoise(?n, ?dB) ∧ swrlb :greaterThan(?dB, 60) ∧ tbox :isDirectSubClassOf(?Class4, systemOntology :Output_Modality) ∧ abox :hasIndividual(?Class4, ?Available) ∧ tbox :notEqualTo(?NotAvailable4, ?Available) → sqwrl :selectDistinct(?NotAvailable4, ?n, ?dB, ?Available)

Type de Handicap	systemOntology :Output_Modality(?NotAvailable) ∧ systemOntology :HandicapType(?h) ∧ systemOntology :IsAffectedBy(?NotAvailable, ?h) ∧ systemOntology :hasDisability(?h, ?Ans) ∧ swrlb :equal(?Ans, 1) ∧ tbox :isDirectSubClassOf(?Class, systemOntology :Output_Modality) ∧ abox :hasIndividual(?Class, ?Available) ∧ tbox :notEqualTo(?NotAvailable, ?Available) → sqwrl :selectDistinct(?NotAvailable, ?h, ?Available)

4.3 Alarme

Nous avons utilisé dans notre système deux types d'alarmes qui sont l'alarme de santé et l'alarme système. Ces alarmes sont activées lorsque le système détecte un problème de santé chez l'utilisateur ou un problème relié au système comme la détection d'un obstacle.

L'alarme de santé est actionnée si l'une des valeurs de l'état de santé de l'utilisateur est alarmante. Ces dernières sont : la tension systolique, la tension diastolique, et la température corporelle de l'utilisateur. Si le système détecte un problème dans une de ces valeurs l'alarme sera activée et un message sera envoyé à l'utilisateur (ou personnel médical) pour prendre en charge ce problème.

Aussi, étant donné que le système utilise une batterie pour se déplacer, une alarme sera actionnée si la batterie se décharge. De plus, nous supposons que le fauteuil est muni de capteurs d'obstacle sur ses quatre coins. Lorsque le système détecte un obstacle l'alarme se déclenche et informe l'utilisateur de la détection.

Ces Alarmes sont déclenchées en utilisant des requêtes SQWRL définies dans Protégé. Les requêtes sont présentées dans le Tableau 4.3 ci-dessous. Prenons comme

exemple la dernière ligne du tableau 4.3 qui permet d'activer l'alarme correspondant au niveau de la batterie. Si la valeur " ?B", qui est l'instance de la classe "Battery Level", est reliée par la propriété d'objet "Is Activated By" à la valeur de la variable " ?A", qui est l'instance de la classe "System Alarm", le système procède à la vérification du niveau de batterie en utilisant la commande "systemOntology :hasBatteryLevel(?B, ?l)" et vérifie si le niveau " ?l" est inferieur à la valeur maximale autorisée, qui est 20%, en utilisant la commande swrlb :lessThan(?l, 20). La commande "sqwrl :selectDistinct(?A, ?B, ?l)" va permettre de retourner les valeurs qui correspondent à la requête et sans dupliquas.

Tableau 4.3: Requêtes de sélection d'alarme

Alarme déclenchée par	**Requête de sélection**
Tension systolique	ssystemOntology :BloodPressureS(?S) ∧ systemOntology :HealthAlarm(?A) ∧ systemOntology :IsActivatedBy(?A, ?S) ∧ systemOntology :hasSystolicPressure(?S, ?p) ∧ swrlb :greaterThan(?p, 12) → sqwrl :selectDistinct(?A, ?S, ?p)
Tension diastolique	systemOntology :BloodPressureD(?D) ∧ systemOntology :HealthAlarm(?A) ∧ systemOntology :IsActivatedBy(?A, ?D) ∧ systemOntology :hasDiastolicPressure(?D, ?p) ∧ swrlb :lessThan(?p, 7) → sqwrl :selectDistinct(?A, ?D, ?p)
Température	systemOntology :BodyTemperature(?T) ∧ systemOntology :HealthAlarm(?A) ∧ systemOntology :IsActivatedBy(?A, ?T) ∧ systemOntology :hasTemperature(?T, ?p) ∧ swrlb :greaterThan(?p, 37) → sqwrl :selectDistinct(?A, ?T, ?p)

Capteur infrarouge	systemOntology :InfraredSensor(?IS) ∧ systemOntology :SystemAlarm(?A) ∧ systemOntology :IsActivatedBy(?A, ?IS) ∧ systemOntology :hasdetectedInf(?IS, ?s) ∧ swrlb :equal(?s, 1) → sqwrl :selectDistinct(?A, ?IS, ?s)
Capteur ultrasonique	systemOntology :SystemAlarm(?A) ∧ systemOntology :UltrasonicSensor(?US) ∧ systemOntology :IsActivatedBy(?A, ?Us) ∧ systemOntology :hasDetectedUlt(?US, ?s) ∧ swrlb :equal(?s, 1) → sqwrl :selectDistinct(?A, ?US, ?s)
Batterie	systemOntology :BatteryLevel(?B) ∧ systemOntology :SystemAlarm(?A) ∧ systemOntology :IsActivatedBy(?A, ?B) ∧ systemOntology :hasBatteryLevel(?B, ?l) ∧ swrlb :lessThan(?l, 20) → sqwrl :selectDistinct(?A, ?B, ?l)

4.4 Moteur de Fusion

Le moteur de fusion permet la fusion des informations obtenues à partir de l'environnement, tels que les modalités et les informations contextuelles. Lorsque l'utilisateur émet une requête, le système détecte les événements et les fusionne pour offrir un service demandé par l'utilisateur. Pour ce faire, notre moteur de fusion utilise le modèle WWHT [24] pour l'interaction multimodale. Dans notre cas, les réponses aux questions sont les suivantes :

- “Quoi” (What) : Quelle information a été détectée par le système, par exemple : une modalité vocale ou un geste ?
- “Quel” (who) : le système décidera quel modèle le moteur de fusion doit-il utiliser pour fusionner les modalités d’entrées (parmi les modèles définis dans l’ontologie).
- “Comment” (how) : le système va fusionner les informations d’entrée en utilisant le modèle de fusion choisi.
- “Ensuite” (then) : Sélectionner la solution de fission appropriée et envoyer le résultat de la fusion ainsi que la solution de fission correspondante.

Avant de procéder à la fusion, le système doit passer par différentes étapes. Ces étapes représentent les règles à suivre pour aboutir à la fusion des données. Si chaque étape est satisfaite, on passe à l’étape suivante, sinon le processus de fusion est interrompu et un message sera envoyé à l’utilisateur. Les étapes sont :

1. **Vérification sémantique** : le système devra faire une vérification sémantique. En effet, en utilisant le concept d’ontologie, nous devons vérifier la cohérence de l’ontologie en vérifiant la relation entre ses composants. Cela peut être fait en utilisant le moteur d’inférence Pellet et le plug-in Jess de Protégé. Le moteur Pellet vérifiera les inférences, la taxonomie et la cohérence entre les classes de l’ontologie et le moteur Jess vérifiera les requêtes SQWRL définies dans notre ontologie.
2. **Vérification du vocabulaire** : le système vérifie la présence des événements détectés dans le vocabulaire prédéfini dans l’ontologie. Cette vérification permet au système de rejeter toute information indéfinie qui ne sera pas utilisée par le moteur de fusion tel que le bruit d’une porte ou d’une télévision. Cette vérification ce fait en utilisant une requête SQWRL dans Protégé qui permettra de comparer les évènements reçus de l’environnement avec le vocabulaire définit dans l’ontologie. Cette vérification est très importante car elle permet de rejeter toute information qui pourrait altérer le bon fonctionnement des moteurs de fusion et de fission et produire des résultats erronés. Comme par exemple lorsque le système détecte

une commande en même temps que la sonnerie d'un téléphone. La vérification permettra de filtrer ces informations, rejeter la sonnerie du téléphone et garder la commande émise par l'utilisateur. La requête SQWRL est la suivante :

systemOntology : Event(?x) $\wedge$ systemOntology :Ontology-Elements(?y) $\wedge$ tbox :equalTo(?x, ?y) $\wedge$ tbox :isDirectSubClassOf(?m, systemOntology :Ontology-Elements) $\wedge$ abox :hasIndividual(?m, ?y) $\rightarrow$ sqwrl :selectDistinct(?m, ?y)

Nous avons créé une classe "Event", indépendante de l'ontologie, qui aura comme instances les parties d'une commande émise par un utilisateur. Si un évènement "x" est détecté (défini dans "Event"), il sera comparé à l'élément "y" qui fait partie des éléments qui composent l'ontologie (Ontologie-Elements). Si "x" correspond à un élément de l'ontologie il sera accepté est affiché avec sa classe d'appartenance représentée par " ?m".

3. **vérification de l'ordre** : Pour ce faire, les événements obtenus des différentes modalités seront vérifiées à l'aide des requêtes SQWRL. Nous avons défini dix-neuf modèles, présentés dans le Tableau 3.1 du Chapitre 3, qui décrivent des exemples de commandes faites par l'utilisateur dans un contexte de commande donné. Lorsqu'un évènement arrive, l'ordre doit être vérifié pour voir s'il correspond à un modèle prédéfini dans l'ontologie. Chaque modèle a sa propre requête définie dans l'onglet SWRL de Protégé. Lorsqu'une série d'évènements est détectée par le système, la recherche du modèle est primordiale. L'ordre d'arrivée des évènements doit être équivalent à l'ordre prédéfini dans les modèles de l'ontologie. Si l'ordre n'est pas respecté cela signifie que la commande n'a pas de sens comme par exemple "Jus moi donne". Nous avons défini dix-neuf requêtes de sélection des dix-neuf modèles de l'ontologie.

   Les requêtes de verification de l'ordre (qui sont les requêtes de selection de modèles) suivent la syntaxe suivante :

systemOntology :Event(?x) ∧ .... ∧ systemOntology :Event(?z) ∧ systemOntology :hasNext_N_NN(?x, ?z)∧ systemOntology :hasNext_Event_N(?x, ?z) ∧ tbox :isInRangeOf(?m, systemOntology :Compose_N) ∧ tbox :isInDomainOf(?s, systemOntology :hasNext_FN) → sqwrl :selectDistinct(?m, ..., ?s)

Ou "x" et "z" sont des instances de l'ontologie. HasNext_N_NN ou "N" varie de 1 à 19 (19 modèles) et NN varie de 1 jusqu'au nombre de classes qui compose chaque modèle. HasNext_Event_N est une propriété d'objet qui permet de définir l'ordre d'arrivée d'un évènement ou "N" varie de 1 à 19 car nous avons donné 19 exemples pour retrouver les 19 modèles. HasNext_FN est une propriété d'objet qui permet de conclure le motif de fission correspondant au modèle de fusion sélectionné, ou "N" varie de 1 à 19.

Par exemple, la commande "Emmène moi là-bas" (Take me there) sera classée comme étant un exemple d'une commande qui satisfait le modèle 4 du Tableau 3.1. Les modèles ont été définis comme des classes dans l'ontologie et chaque classe est composée d'autres classe de l'ontologie reliées par des propriétés d'objet. Les classes des modèles sont composées de classes déjà définies dans l'ontologie. Par exemple, les classes du modèle 4 sont : "mots pour le suivi" qui est la sous-classe de la classes "Vocabulaire". La classe "objective" et "places" qui sont les sous-classes de la classe "Alentours".

Pour l'exemple mentionné ci-dessus, le système doit détecter d'abord le mot "Emmène" comme une instance de la classe "mots pour le suivi", le mot "moi" comme une instance de la classe "objective" et, enfin, le mot "là-bas" comme une instance de la classe "places". L'ordre des événements doit être satisfait et correspondre à l'ordre du modèle 4 (propriété d'objet "hasNext_4). Sinon, l'événement sera rejeté car il ne correspond pas à un modèle prédéfini dans l'ontologie. Les requêtes de sélection du modèle de fusion qui correspond aux exemples d'évènements sont présentées dans l'Annexe I.

4. **vérification du temps** : le contrôle du temps permettra la vérification du temps

entre deux modalités successives et le temps total de la commande. En effet, le temps entre deux modalités doit être inférieur au temps maximal prédéfini autorisé (de 5 secondes) et le temps total d'une commande doit être inférieur au maximum autorisé prédéfini dans l'ontologie (15 secondes). Nous avons choisi d'introduire cette vérification afin que le système n'attende pas les commandes de l'utilisateur indéfiniment. Cette vérification se fera en utilisant une requête de vérification de temps défini dans l'onglet SWRL de protégé.

La règle présentée ci-dessous permet de vérifier le temps entre les modalités et le temps de la commande. La propriété has_Time_start permet de connaitre le temps d'arriver de chaque modalité. Pour avoir le temps entre les modalités on procède à la soustraction du temps de deux modalités successives. Pour ce qui est du temps de la commande totale on procède a la soustraction du temps de la première modalité du temps de la dernière modalité.

En satisfaisant ces quatre étapes (règles), le système sera en mesure de reconnaître les événements obtenus de l'environnement et de compléter la fusion selon le modèle choisi (prédéfini dans l'ontologie) tout en prenant en compte l'état dynamique du contexte.

La requête utilisée est définie comme suit :

systemOntology : Words_Convenient_Objects(?x) ∧ systemOntology : Individuals(?y) ∧ systemOntology : Liquid(?z) ∧ systemOntology :Gesture(?g) ∧ systemOntology :has_Time_start(?g, ?Tsg) ∧ systemOntology :has_Time_start(?x, ?Tsx) ∧ systemOntology :has_Time_start(?y, ?Tsy) ∧ systemOntology :has_Time_start(?z, ?Tsz) ∧ swrlb :subtract(?Txg, ?Tsg, ?Tsx) ∧ swrlb :subtract(?Tgy, ?Tsy, ?Tsg) ∧ swrlb :subtract(?Tyz, ?Tsz, ?Tsy) ∧ swrlb :subtract(?FullTimeC, ?Tsz, ?Tsx) → sqwrl :selectDistinct(?x, ?g, ?y, ?z, ?Txg, ?Tgy, ?Tyz, ?FullTimeC)

4.5 Moteur de Fission

Le moteur de fission est aussi important que le moteur de fusion. En effet, le moteur de fission subdivise les résultats du moteur de fusion en sous-tâches élémentaires et les envoie aux modalités de sortie disponibles tout en prenant en compte l'état du contexte. La subdivision est faite selon des motifs prédéfinis dans l'ontologie. Ces commandes de base seront envoyées aux sorties des modalités en fonction de leurs disponibilités. Des motifs de fission sont définis dans l'ontologie en deux parties, à savoir un problème et une solution. Le problème est la commande émise par l'utilisateur qui est le résultat du moteur de fusion, tandis que la solution se compose de toutes les sous-tâches possibles pour cette commande. En outre, les modalités de sortie choisies sont : l'écran, le haut-parleur, les roues et le bras. Ces derniers peuvent être désactivés en fonction de l'état du contexte en utilisant les requêtes présentées dans la section 4.2.2.

Rappelons que nous avons défini neuf solutions de fission possibles dans l'ontologie qui correspondent aux 19 modèles de fusion cités précédemment (présentées dans le Chapitre 3 précèdent). Ces solutions seront sélectionnées en utilisant des requêtes SQWRL définies dans l'ontologie. Lorsque le moteur de fusion fournit le résultat de la fusion, il accompagne ce résultat par le numéro de la solution de fission correspondante au modèle de fusion. Cela facilite le travail du moteur de fission car il n'aura qu'à sélectionner cette solution et l'exécuter pour obtenir des actions élémentaires à envoyer aux actionneurs de sortie disponibles. Les requêtes de sélection des solutions de fission suivent la syntaxe suivante :

systemOntology : Fission_Solution_x(?a) $\wedge$ systemOntology : Fission_Solution_x(?b) ..... $\wedge$ systemOntology : Fission_Solution_x(?n) $\wedge$ systemOntology : hasNext_Fx(?a, ?b) $\wedge$....$\wedge$ systemOntology : hasNext_Fx(?.., ?n) $\wedge$ tbox : isInDomainOf(?s, systemOntology : hasNext_Fx) $\rightarrow$ sqwrl : selectDistinct(?s, ?a,...., ?n)

Cette requête permet de sélectionner les solutions de fissions. Nous rappelons que les

actions générées par le moteur de fission sont déclarées sous formes d'instances. Cette requête permet de récupérer les instances reliées par la propriété d'objet « hasNext_F ».

Comme le moteur de fusion, le modèle WWHT sera utilisé pour le moteur de fission. Cela permettra au moteur de fission de subdiviser les résultats du moteur de fusion en répondant aux questions "Quoi", "Quel", "Comment" et "Ensuite". En répondant à ces questions, le processus de fission se termine :

- «Quoi» : se réfère à ce que le système a détecté, qui sera le résultat du moteur de fusion.

- «Quel» : le système décidera quelle solution le moteur de fission devrait utiliser pour subdiviser le résultat de la fusion.

- «Comment» : le système subdivise le résultat de la fusion en utilisant la solution de fission choisie parmi les solutions prédéfinies et stockées dans l'ontologie.

- «Ensuite» : le système sélectionne les modalités de sortie appropriées en fonction des informations du contexte, par exemple : si l'utilisateur est sourd, alors la modalité hautparleur sera désactivée. Pour le processus de fission, nous avons défini 9 solutions possibles dans l'ontologie. Ces solutions sont présentées dans le Tableau 3.2 du Chapitre 3. Chaque solution est une classe de l'ontologie, les instances sont connectées en utilisant les propriétés d'objets pour définir leur ordre d'exécution.

4.6 Etude de cas

Nous allons présenter un exemple de commandes effectuées par l'utilisateur et la façon dont le système traite et fusionne les informations en utilisant le moteur de fusion et subdivise le résultat de la fusion en utilisant le moteur de fission. Le scénario choisi pour tester notre moteur de fusion est "Donne-lui du jus" (Give her juice) en pointant vers une personne. Cette requête utilise deux types de modalités que sont la parole et le geste. Dans ce cas, le système doit comprendre le sens de la phrase, trouver un modèle qui lui

corresponde et fusionner les informations. Aussi, le système doit prendre en compte la durée maximale autorisée entre les modalités et le temps total de la commande.

Puis, le système doit subdiviser le résultat de la fusion en sous-tâches élémentaires et envoyer la commande à des modalités de sortie disponibles en utilisant le moteur de fission. Nous soulignons que lorsque l'utilisateur demande une boisson liquide (comme du jus), le système doit comprendre que cette boisson doit être apportée dans un objet approprié, tel qu'un verre ou une bouteille.

Après avoir lancé le raisonneur Pellet sur Protégé et que la syntaxe et la taxonomie ont été vérifiées et qu'aucun problème n'a été détecté, nous commençons notre test du moteur de fusion. Nous supposons que le système a détecté les entrées décrites dans le Tableau 4.4. Cette détection permet de répondre à la question "What" du modèle WWHT utilisé par le moteur de fusion.

Tableau 4.4: Exemple de commande émise par l'utilisateur

Modalité	**Evènement**	**Temps d'arrivé (seconde**
Parole	Donne (Give)	0
Geste	Capteur de geste	3
Parole	Lui (Her)	5
Son de la télévision	ding	5
Parole	Jus (Juice)	7

Le moteur de fusion vérifie les évènements détectés en entrée en les comparant au vocabulaire défini dans l'ontologie et fournit les classes où ces évènements sont définis comme étant leurs individus. Cette procédure est réalisée en exécutant la requête de "vérification du vocabulaire", présentée ci-dessus, dans l'onglet "swrl" de Protégé, le résultat est présenté sur la Figure 4.2.Nous notons que le moteur de fusion a reconnu les mots : "Give", "Her", "Juice". Aussi, le moteur de fusion fournit les informations du capteur de geste qui a détecté la position de "Her" et donne l'emplacement des instances

dans les différentes classes de l'ontologie. Le mot "ding" qui est un son émis par la télévision et détecté par le système est rejeté car il ne fait pas parti du vocabulaire défini dans l'ontologie et ne correspond à aucune combinaison de commande prédéfinie dans les modèles de fusion.

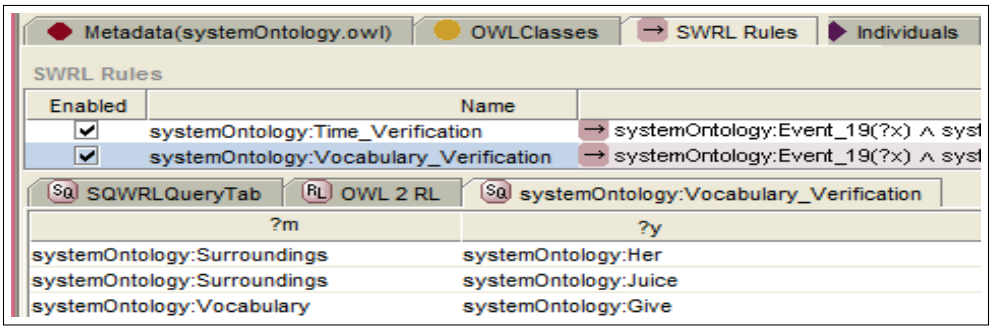


FIGURE 4.2: Résultat de la vérification du vocabulaire

En outre, supposons que l'utilisateur du fauteuil roulant a comme problème de santé des problèmes auditifs. Supposons aussi que le niveau de luminosité est supérieure à la valeur minimale acceptable et que le niveau de bruit est inférieur à la valeur maximale acceptable. Aussi, supposons que la batterie est complètement chargée et que l'utilisateur se trouve dans une situation de bonne santé (les tensions artérielles et la température sont bonnes). Dans ce cas, les modalités qui seront désactivés ne sont que ceux qui sont touchés par le problème d'audition, qui est la modalité de sortie "Speaker". Toutes les autres modalités d'entrée et de sortie sont disponibles. Le résultat de la requête de sélection de modalités, après execution dans Protégé, est présenté sur la Figure 4.3.

?NotAvailable	?h	?Available
systemOntology:Speaker	systemOntology:Hearing_problems	systemOntology:Moving_Arm
systemOntology:Speaker	systemOntology:Hearing_problems	systemOntology:Screen
systemOntology:Speaker	systemOntology:Hearing_problems	systemOntology:Wheels

FIGURE 4.3: Résultat de la sélection de modalités

Pour la question "Which", le système va lancer une requête pour trouver le modèle correspondant dans l'ontologie. Le moteur de fusion doit trouver le modèle prédéfini de la commande "Give Her some Juice" qui est représenté par l'ordre suivant : Mots pour

objets pratiques → personnes → liquide. Le modèle correspondant à cet ordre est le modèle 19 présenté dans le Tableau 3.1 du Chapitre 3

Pour la question «How», le moteur de fusion va fusionner les modalités d'entrée en utilisant le modèle de fusion. Nous avons exécuté la requête pour l'exemple de l'évènement 19 (la requête est présentée dans l'Annexe I),Comme presenté sur la Figure 4.4, le moteur de fusion a compris que le liquide demandé doit être apporté dans un objet utilisé pour les liquides et l'a rajouté dans la commande finale. Cette compréhension est possible car nous avons rajouté une propriété d'objet "needs liquid object" qui relie les liquides aux objets utilisés pour les liquides. Donc lorsque la requête du modèle est activée, le moteur de fusion recherche cette propriété et retourne l'objet qui correspond au liquide, dans notre cas c'est "cup" (verre).

SQWRLQueryTab | OWL 2 RL | systemOntology:Event_Check_19

?m	?x	?y	?DetectedBy
systemOntology:Command_Model_19	systemOntology:Give	systemOntology:Her	systemOntology:Gesture_Sensor

?a	?z	?s
systemOntology:Cup	systemOntology:Juice	systemOntology:Fission_Solution_7

FIGURE 4.4: Résultat de la Fusion

Le moteur de fusion a reconnu le modèle comme étant le modèle 19 de l'ontologie et que la réponse à une telle demande sera la réponse du modèle : Mots pour objets pratiques → personnes → Object utilisés pour le liquide → Liquide. En fait, le moteur de fusion a reconnu le modèle et ajouté l'objet "cup" qui peut être utilisé pour apporter le jus.

Enfin, à la question "Then", le moteur de fusion envoie le résultat de la fusion et de la solution de fission correspondante après avoir fait une vérification du temps entre les modalités et le temps total de la commande. Le résultat est présenté sur la Figure 4.5, où Txy et Tyz sont les temps entre les modalités successives "Give", "Her" et "Juice" respectivement, et FullTime-C est le temps de la commande totale. Nous remarquons que les temps entre les modalités sont moins de "5 secondes" (qui est le maximum autorisé) et que le temps de commande complète est inférieure à "15 secondes" (le temps

de commande maximum autorisé). On conclut que la condition de temps a été satisfaite pour les deux temps.

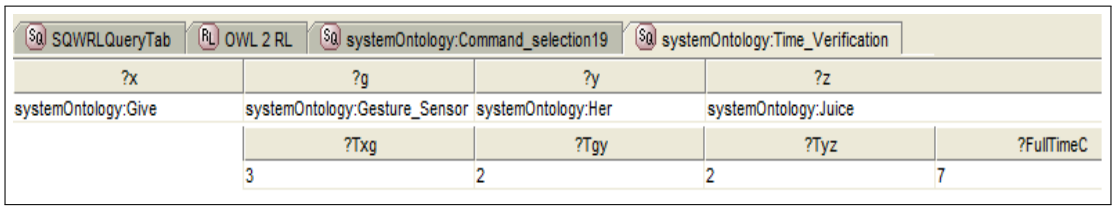


FIGURE 4.5: Résultat de la vérification du temps

Maintenant que le moteur de fusion a fusionné les informations recueillies à partir de l'environnement et a compris la demande de l'utilisateur, le système a besoin d'envoyer ces informations aux modalités de sortie disponibles en utilisant le moteur de fission. Revenons à la Figure 4.4, lors de la fusion des informations, le moteur de fusion conclut que le motif de fission correspondant à notre exemple et qui doit être utilisé est la solution de fission 7. Ainsi, la question "Which" du modèle WWHT appliqué au moteur de fission est résolue.

Lors de l'exécution de cette requête, le système subdivise les résultats du moteur de fission en sous-tâches élémentaires. Le résultat est présenté dans la Figure 4.6. Le résultat de la fission est de se déplacer vers la position, saisir l'objet liquide, verser le liquide, se déplacer vers la position de la personne, puis déposer l'objet. Lors de l'utilisation du modèle WWHT pour le moteur de fission, la réponse à la question "What" est l'information obtenue à partir du moteur de fusion qui précise la solution de fission à utiliser .

Maintenant que la solution de fission a été identifiée, pour répondre à la question "How" le système doit trouver la solution de fission correspondante et l'exécuter. Nous avions défini neuf solutions de fission dans le Tableau 3.2 du Chapitre 3 qui décrivent des exemples de fission selon les modèles de fusion. Selon le motif de fission, le moteur de fission va diviser le résultat de la fusion à l'aide d'une solution prédéfinie dans l'ontologie. Chaque solution a sa propre requête définie dans l'onglet SWRL de Protégé. Lors de l'exécution de cette requête, le système subdivise les résultats du moteur de fusion en sous-tâches élémentaires. Le résultat est présenté dans la Figure 4.6.

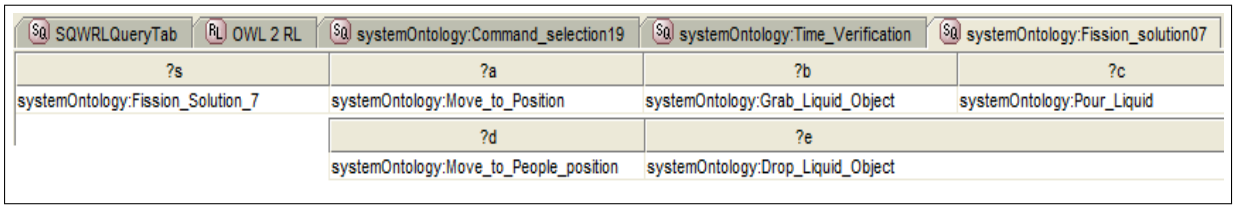


FIGURE 4.6: Résultat de la Fission

Par exemple,la solution de l'exemple "Donne lui du jus" (Give her some juice) présenté précédemment sera la solution de fission numéro 7, qui est :

Fission solution 7 : Move to position → Grab liquid object → Pour liquid → Move to people position → Drop liquid object

Pour la question "Then" le moteur de fission procède à la vérification du contexte en utilisant la requête SQWRL qui sélectionnera les modalités de sortie disponibles et décidera quelles modalités de sortie peuvent être utilisées en fonction de l'état du contexte. Rappelons que nous supposons que l'utilisateur du fauteuil roulant a des problèmes auditifs et que toutes les autres valeurs sont dans les intervalles acceptables. Dans ce cas, seule la modalité de sortie vocale (haut-parleur) sera désactivée.

Le résultat est présenté sur la Figure 4.3. Le choix des modalités de sortie est effectué en vérifiant l'état du contexte. En effet, le système récupère en continu des informations du contexte provenant de l'environnement et les compare à des conditions de sélection de modalités prédéfinies dans l'ontologie. Si une condition de désactivation de modalité est satisfaite, le système désactive la modalité de sortie correspondante.

Enfin, en répondant à ces quatre questions, le système sera en mesure d'envoyer les tâches subdivisées aux actionneurs de sortie correspondants. Cela signifie que le processus de fusion et de fission est finalisé et que la demande de l'utilisateur "Donne lui du jus" est répondu.

4.7 Conclusion

Nous avons présenté dans ce chapitre les différentes étapes qui doivent être suivies pour aboutir à la fusion et la fission des données pour répondre à la requête de l'utilisateur, et réalisé ainsi la boucle d'interaction complète (perception, compréhension, décision et action). En effet, la perception et la compréhension sont réalisées par le processus de fusion de modalités d'entrée. Tandis que la décision et l'action sont réalisées par le processus de fission. Ces processus sont à base de règles logiques qui utilisent une ontologie pour décrire le contexte d'interaction dans le système de commande.

D'abord la sémantique était vérifiée à l'aide du moteur Pellet de Protégé pour la vérification de la cohérence de l'ontologie. Puis, en répondant aux questions du modèle WWHT, la vérification du vocabulaire, la sélection de modalité et l'ordre des événements ont été satisfait et ont permis de trouver le modèle de fusion correspondant aux entrées multimodales détectées. Aussi, la vérification du temps a été satisfaite lors de la vérification du temps entre l'arrivée des différentes modalités et le temps total de la commande. Enfin, le moteur de fission a subdivisé le résultat de fusion en sous-tâches élémentaires à l'aide d'une solution sélectionnée parmi les solutions prédéfinies et stockées dans l'ontologie.

En satisfaisant toutes les préconditions, le moteur de fusion a fusionné tous les évènements d'entrée et le moteur de fission a subdivisé le résultat de la fusion. Par conséquent le système a répondu à la demande de l'utilisateur qui était "Donnez-lui du jus" en utilisant la commande "Donnez-lui un verre de jus".

Donc non seulement le système a pu fusionner et fissionner les informations récupérées de l'environnement, mais aussi a compris que le jus qui est une boisson doit être apporté dans un verre sans que l'utilisateur ne le précise dans sa requête. Cette caractéristique montre que le système a bien compris la demande de l'utilisateur et fournit le service requis en utilisant la réponse la plus adéquate. Nous concluons que l'architecture proposée est un système autonome capable de comprendre la requête de l'utilisateur et fournir la réponse la plus optimale.

Chapitre 5

Validation de l'approche

5.1 Introduction

La validation de notre système est une étape très importante de notre travail. Pour cela, nous avons choisi de modéliser le système en utilisant les réseaux de Pétri temporels et stochastiques en utilisant l'outil CPN Tools [26], qui permet la modélisation et la visualisation des réseaux de Pétri.

Pour comprendre le mécanisme de nos moteurs de fusion et de fission, nous allons utiliser trois scénarios de commandes potentielles de l'utilisateur et explorer le comportement de l'architecture proposée. Ces scénarios sont : Emmène-moi là-bas (take me there), apporte-moi une cuillère (bring me a spoon) et donne lui du jus (give her juice) Qui sont le modèle de fusion 4, 11 et 19 et ont les solutions 4, 1 et 7 de fission respectives.

Nous avons choisi de simuler le fonctionnement de l'ontologie pour ces trois exemples sur CPNTools et non pas toute l'ontologie (qui a été présenté dans le chapitre Architecture du système) car nous avons voulu apporter une validation fonctionnelle de nos algorithmes de fusion et de fission. Ces derniers se composent de plusieurs processus qui sont la sélection de modalités, la vérification du temps et de l'ordre, la sélection du modèle de fusion et du motif de fission. Ces processus aboutissent à l'offre de service via une boucle complète d'interaction et de perception, de compréhension, de décision et d'action. L'ontologie qui sera modélisée par CPNTools sera uniquement pour les trois

exemples pris en charge par la validation.

5.2 But de la validation

Le but de notre étude est de développer une base théorique pour l'interaction homme-machine-environnement afin de valider formellement l'architecture proposée. Cette validation formelle prouve que la méthode utilisée dans notre travail est efficace dans l'environnement de simulation. Dans ce travail, la représentation sémantique de l'environnement est formalisée en utilisant l'ontologie, et les processus de fusion et de fission sont modélisés en utilisant un ensemble de règles logiques de premier ordre. L'architecture est modélisée en utilisant les réseaux de Pétri temporels et stochastiques "RDPTS".

Un générateur aléatoire d'événements d'entrée pour simuler le fonctionnement de l'architecture a été également conçu et développé. L'ontologie définit les concepts qui sont communs dans divers domaines, nous définissons dans ce travail l'ontologie en utilisant deux types de classes : les événements et les concepts. Nous utilisons le langage OWL pour décrire les relations sémantiques entre les concepts sous forme de prédicats.

Dans notre architecture deux processus importants que sont la Fusion et la Fission, partagent l'ontologie et l'utilisent comme base de connaissance. Les modèles d'événements de la fusion et de la fission sont modélisés par un ensemble de règles logiques de premier ordre et sont stockés dans l'ontologie. Ensuite, les agents de fusion et de fission utilisent un moteur d'inférence pour le raisonnement et la prise de décision.

Dans un environnement réel robotisé, il est facile de programmer les services du serveur pour prendre en charge les capteurs et les actionneurs utilisés dans l'application robotique souhaitée. Cependant, toute architecture interactive, comme celle proposée dans le présent document, peut également recevoir avec précision les entrées nécessaires et fournir les sorties nécessaires en utilisant les serveurs comme cela est montré dans les travaux présentés dans [64]. Dans ce travail, le modèle obtenu après avoir mis en relation les services et les codes de commande est implanté physiquement dans un serveur permettant ainsi de publier les services domotiques et les modes préalablement définis. Il s'agit d'une architecture décentralisée où les serveurs communiquent entre eux pour

délivrer des services [64].

Ce qui reste à valider, dans notre travail, sont les fonctionnalités importantes du processus de cette architecture que sont la fusion et la fission en utilisant l'ontologie définie. Cette validation va prouver la faisabilité et la pertinence de notre architecture pour les applications robotiques réelles, grâce à :

1. L'utilisation du formalisme de l'ontologie OWL qui nous permet de vérifier la cohérence de la représentation des connaissances de l'environnement robotique. La vérification des instances est réalisée en utilisant PROTEGE pour des applications robotiques réelles. Le mécanisme de raisonnement est défini à l'aide de règles de fusion et de fission.

2. L'utilisation du réseau de Pétri Temporel et Stochastique nous permet de modéliser le temps de traitement en utilisant quelques lois de probabilité. Nous exécutons les applications sur plusieurs scénarios pour différents modèles de l'ontologie. Cela est possible en générant différentes séquences d'événements et en vérifiant les événements reconnus en les comparant aux modèles prédéfinis dans l'ontologie.

La vérification de la cohérence est appliquée en utilisant l'outil CPN Tools. Ce dernier vérifie la relation sémantique définie dans l'ontologie, la compatibilité des relations et des opérations de composition ainsi que la cohérence sémantique des modèles de règles (fusion et fission) pour éviter les conflits. La vérification de cohérence est donc intrinsèque au moteur d'inférence. La validation temporelle du moteur d'inférence est réalisée en utilisant CPN Tools pour la validation d'un scénario choisi ainsi que tous les scénarios possibles représentés par des modèles de règles dans l'ontologie (le cas général).

Nous avons vérifié l'exhaustivité de l'architecture, l'ordre d'arrivé, et la contrainte temporelle liée à la condition préalable d'un modèle de règle de fusion et sa consistance. Toutes les contraintes sont bien respectées en raison de la conception de la base de connaissances et des règles / algorithme utilisé par le moteur d'inférence.

5.3 Réseau de Pétri

5.3.1 L'environnement de simulation

Nous allons présenter dans cette section l'environnement de simulation que nous avons utilisé pour valider l'architecture du sytème proposé. Nous avons construit un réseau de Pétri qui simule le fonctionnement global de l'architecture, la déclaration de variables dans CPN Tools est presentée dans l'Annexe II. Le réseau de Pétri de notre architecture, présenté sur la Figure 5.1, représente l'aspect global de l'ensemble des éléments de l'architecture. Cette figure présente le réseau de Pétri qui est composé de plusieurs sous réseaux (dans chaque rectangle doublé) qui seront expliqué par la suite dans des figures séparées. Le réseau de Pétri est très grand et ne peut être représenté sur une seule image, de ce fait, nous avons choisi de le construire sous forme de sous réseaux relié.

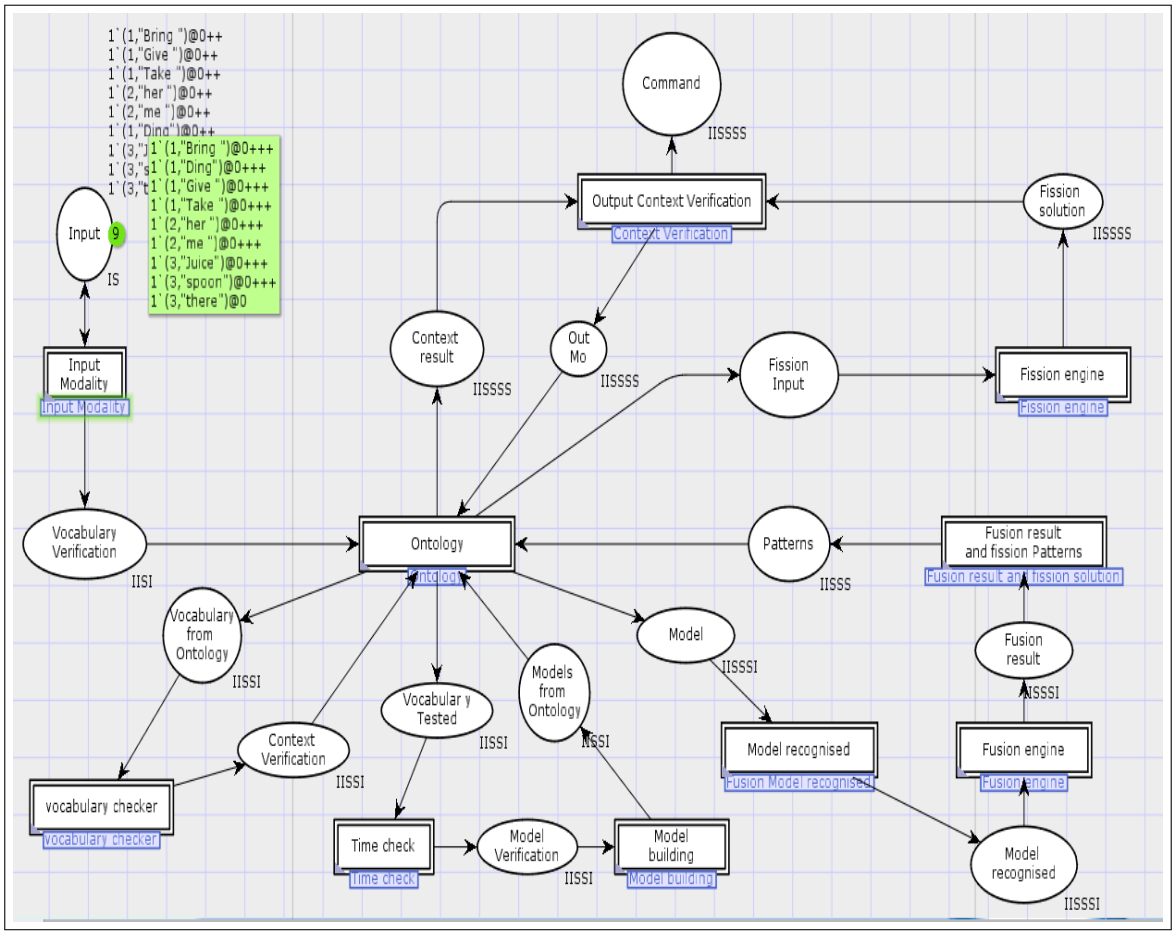


FIGURE 5.1: Vue générale de l'architecture du système multimodal sur CPN Tools.

La Figure 5.1 montre le fonctionnement de l'architecture proposée. Tout d'abord, il y a une génération aléatoire de combinaison d'entrée qui sera perçu comme une commande et réalisée par le sous réseau "Input Modality". Cette génération aléatoire va prouver que le système est capable de comprendre une commande et de différencier entre une commande qui a un sens et une autre sans sens. Pour cela nous avons rajouté le mot "Ding" dans la place "Input" qui est le lieu de la déclaration du vocabulaire utilisé dans le cadre de cette validation, la place "Input" est l'entrée du réseau "Input Modality" qui va permettre de générer aléatoirement des commandes composées de trois mots (par commande) parmi le vocabulaire et de fournir un temps aléatoire a chaque commande, comme presenté sur la Figure 5.2.

Sur la Figure 5.1 on voit bien que l'ontologie est un élément clé de l'architecture car la majorité des sous réseaux reviennent constamment à l'ontologie lors de la prise en charge de la commande. Lorsqu'une combinaison aléatoire est générée dans le sous réseau représenté par le rectangle "Input modality", nous pouvons avoir différentes combinaisons, ces combinaisons seront envoyées au sous réseau "Vocabulary checker" qui va vérifier la présence du vocabulaire dans l'ontologie. Cette partie va permettre de rejeter toute commande émise avec le son "Ding" car il ne sera pas reconnu comme un vocabulaire de l'ontologie.

Puis, si la commande générée aléatoirement, est formée par un vocabulaire reconnu et défini dans l'ontologie, on passe au sous réseau "Time check" pour vérifier le temps entre les modalités et le temps global de la commande. Cela se fait en retournant vers l'ontologie ou les temps maximums autorisés sont définis. Si le temps est accepté la commande sera envoyée au prochain sous réseau, sinon la commande sera rejetée et le système passe à une autre commande (en envoyant un feedback pour la commande rejetée).

Maintenant que le vocabulaire et le temps sont vérifiés, nous allons passer à la construction du modèle pour vérifier s'il correspond à un modèle de fusion défini dans l'ontologie. Pour cela, le sous réseau "Model building" va reconnaitre chaque mot d'une commande et retrouver sa classe défini dans l'ontologie. Lorsque les classes des mots

qui composent une commande sont trouvées, elles seront regroupées pour composer un modèle qui sera comparé au modèle de fusion (présenté dans le tableau 3.1 du chapitre Architecture du système) par le sous réseau "Model recognised". Rappelons que chaque modèle de fusion est composé d'une succession de classe de l'ontologie dans un ordre bien précis. Si le modèle qui a été composé des classes de la commande (générée aléatoirement) correspond à un modèle de fusion défini dans l'ontologie, le système l'envoie pour plus de traitement, sinon cette commande sera rejetée comme étant incorrecte, comme par exemple la commande "Me her there", qui n'a pas de sens malgré que le vocabulaire est présent dans l'ontologie.

Puis, le système procède à la fusion des données d'entrée en combinant les mots et retrouve le motif de fission qui correspond au modèle de fusion utilisé. Le système vérifie le contexte de sortie, pour voir si les modalités de sortie sont disponibles, et envoie la commande qui a respecté toutes les contraintes et suivi les algorithmes de fusion et de fission étape par étape jusqu'à obtenir la commande fusionnée et fissionnée finale (qui sera celle qui a satisfait tous les tests de l'architecture).

1. **Modalité d'entrée (Input Modality) :** Le module responsable de la génération de modalités d'entrée est représenté sur la Figure 5.2. ce module est responsable de l'envoi d'une combinaison aléatoire de mots pour le système. Cette combinaison sera une commande émise par un utilisateur. Comme on le voit sur la Figure 5.2, neuf mots on été définis : bring, give, take, her, me, ding, spoon, juice et there (apporter, donner, emmène, elle, moi, ding, cuillère, jus et là). Chaque exemple sera formé par trois mots et la formation d'exemples est limitée par le nombre Mmax. Aussi, lorsqu'un mot est généré aléatoirement un temps lui est associé aléatoirement aussi. Le temps entre le premier et le troisième mot sera le temps total de la commande et sera vérifié plus tard, ainsi que le temps entre l'arrivée des modalités. Le mot, le numéro de l'exemple et le temps seront envoyés au module "Vérification du vocabulaire" (vocabulary verification). Ce module va envoyer les informations à comparer avec la base de connaissances stockée dans l'ontologie.

   En envoyant les mots aléatoirement, nous pouvons avoir deux configurations : une

bonne combinaison de mots qui aboutit à une phrase correcte ou une combinaison de mots qui n'a pas de sens. Cela permettra au système de reconnaître les modèles corrects et de rejeter les combinaisons incorrectes tout en envoyant un feedback pour informer l'utilisateur.

Les mots dans la liste sont accompagnés par une valeur "n", par exemple 1'(n=1, "Bring"). Lors du premier envoi, un des mots qui a la valeur n=1 sera envoyé aléatoirement et une génération aléatoire du temps sera affectée à ce mot dans la place "Time". Lorsque ce mot arrive à la transition "Received" un compteur va permettre d'incrémenter la valeur de "n" et la renvoyer à la transition "Event" qui va envoyer un mot aléatoire parmi les mots qui ont n=2 et ainsi de suite. Lorsque "n" sera égale à 3, le compteur remet la valeur de "n" a 1 et l'envoie à la transition "Event". Ceci va permettre de générer une autre commande composée de trois mots aléatoires parmi le vocabulaire défini.

2. **Ontologie (Ontology)** : est la base de connaissances spécifique aux trois exemples de validation du système. Nous avons choisi de modéliser le fonctionnement de l'ontologie uniquement pour ces trois exemples, nous n'avons pas voulu dupliquer le travail du chapitre Architecture du système.

   Comme le montre la Figure 5.3, huit mots accompagnés de leurs classes respectives sont déclarés comme vocabulaire dans l'ontologie. Nous rappelons que cette validation se concentre sur trois scénarios, donc la déclaration du vocabulaire est focalisé sur ces scénarios. Par exemple, les places "wco1" et "wco2" représentent la classe "mots objet pratique" qui contienent les mots «bring» et «give» respectivement. La place "WFT" représente la classe "mots pour le suivi" et contient le mot "Take". La place "L" représente la classe "Location" et contient le mot "There". "PP" et "PS" représentent les classes "pronom personnel" et "Personnes" et contienent les mots «Me» et «her» respectivement. Enfin, la place "co" et "lq" représentent les classes "objet pratique" et "liquide" et contienent les mots «Spoon» et «Juice», respectivement.

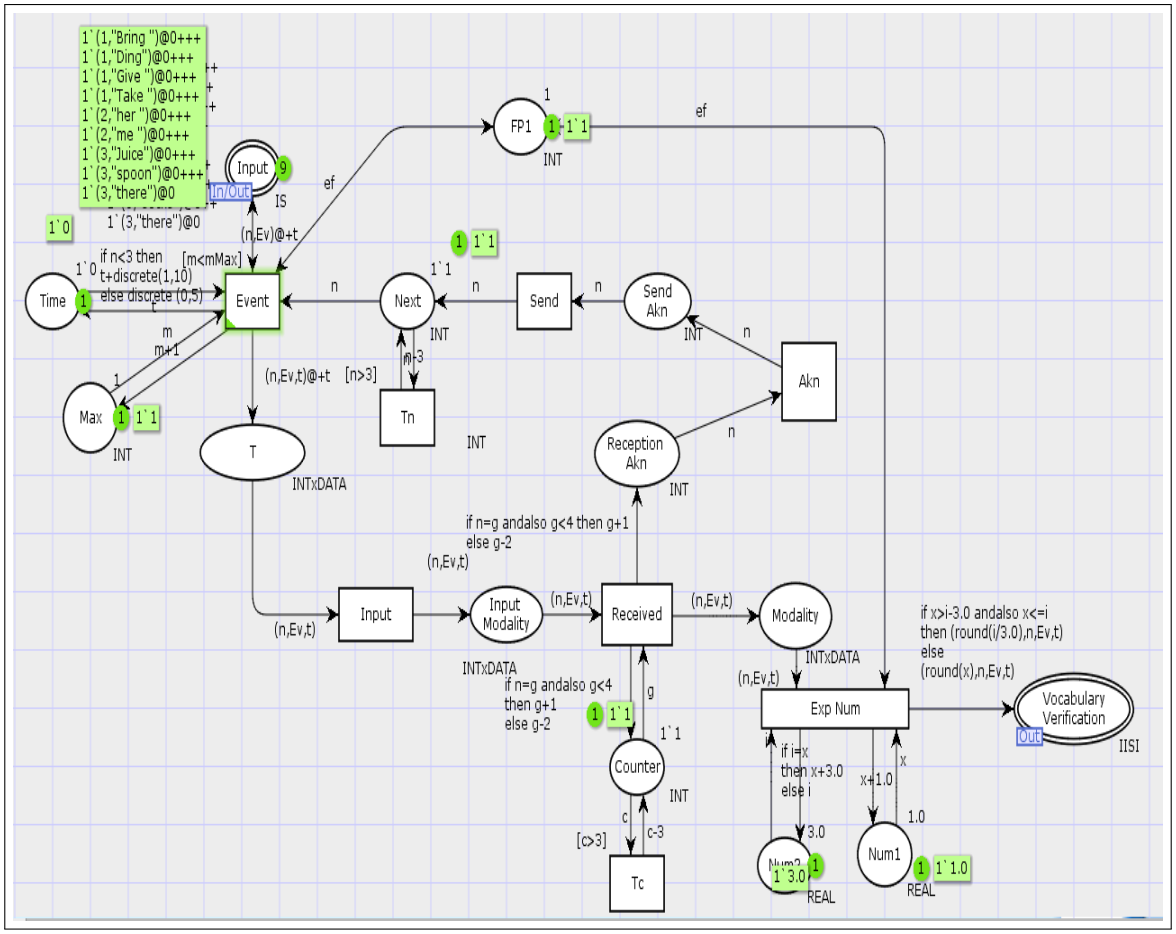


FIGURE 5.2: Transition "Input Modality"

Donc, quand un mot est détecté comme modalité d'entrée, le système va rechercher dans ce module et récupérer la classe correspondante. Nous soulignons que le mot "Ding" déclaré dans l'entrée n'a pas de sens (il peut être un son émis par un téléviseur et détecté par le système). C'est pour cette raison qu'il n'est pas déclaré comme vocabulaire dans l'ontologie. Donc, si, par exemple une commande contient le mot "Ding" elle sera rejetée. Rappelons que nous avons choisi de déclarer ces mots selon nos scénarios, et l'un des avantages de l'utilisation du concept d'ontologie est l'ouverture du système. Cela signifie qu'un autre scénario avec d'autres mots peuvent être ajoutés à n'importe quel moment et rendre notre système adaptable à toute situation.

La partie de l'ontologie représentée sur la Figure 5.3 permet de comparer les modalités d'entrée au vocabulaire définie dans l'ontologie, dans les places "wco1",

"L", "Wft", "pp", "ps", "co", "lq" et "wco2". Lorsqu'une commande arrive dans la place "Vocabulary verification", la modalité sera comparée au vocabulaire défini.

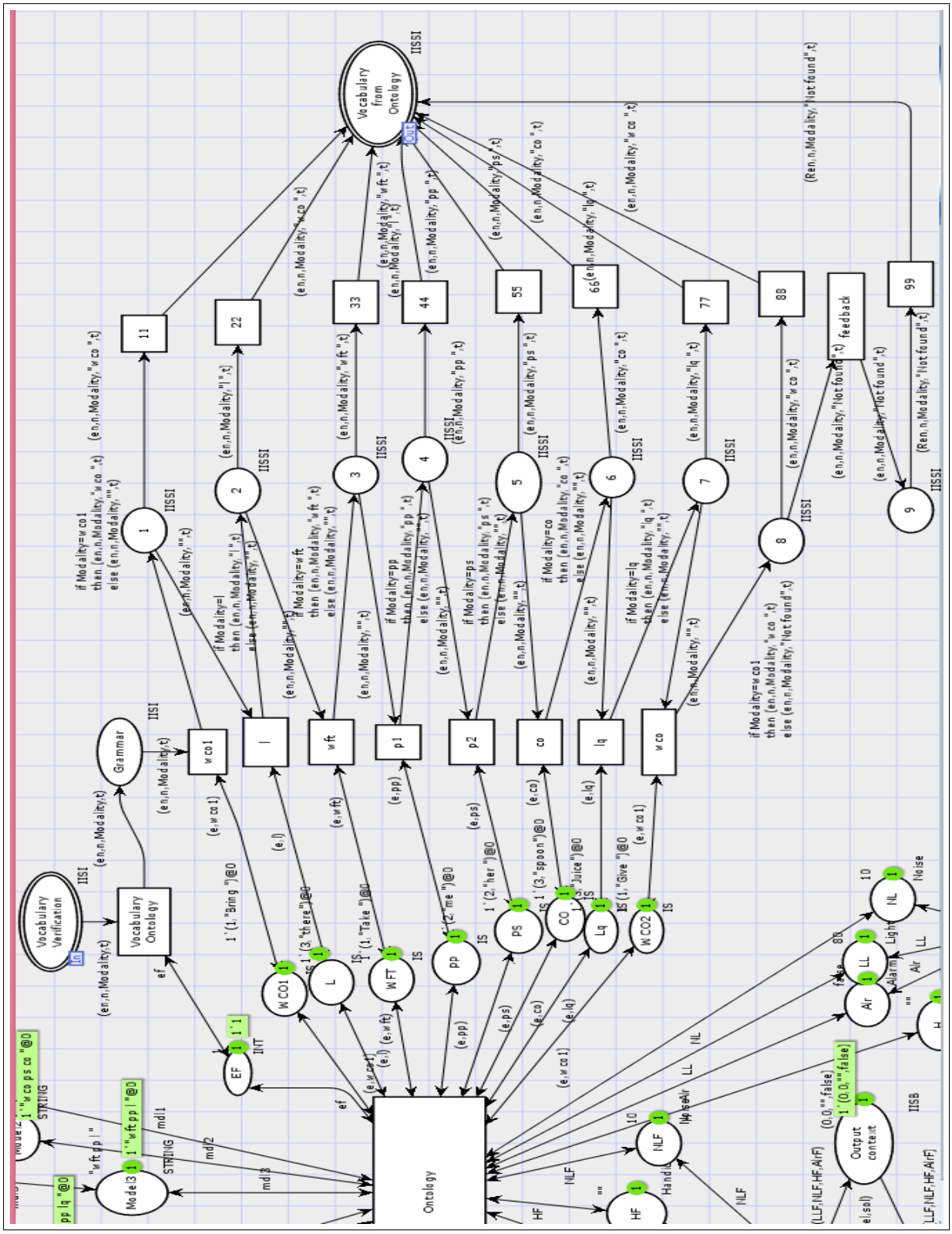


FIGURE 5.3: Partie de l'ontologie responsable de la vérification du vocabulaire

Si la modalité est retrouvée, la classe qui lui correspond sera ajoutée dans la liste de la commande comme par exemple (en, n, Modality, "wco",t), et envoyée par la suite à la place "Vocabulary from ontology", et sera traitée par les autres sous réseaux. Sinon elle sera comparée aux autres mots dans le vocabulaire jusqu'à ce que le bon et retrouvé. Si la modalité de l'exemple de commande ne correspond à aucun mot du vocabulaire, cet exemple sera envoyé à la transition "feedback" avec la mention "not found".

3. **Vérificateur du vocabulaire (Vocabulary checker) :** Ce sous réseau, illustré sur la Figure 5.4, va permettre de filtrer les commandes qui ont un sens (vocabulaire retrouvé dans l'ontologie) et les commandes sans sens. Le réseau "Ontologie" présenté précédemment envoie toutes les combinaisons retrouvées (vocabulaire trouvé ou pas) à la place "Vocabulary from ontology" qui est la place d'entrée du réseau "Vocabulary checker". Nous allons dans ce réseau rejeter les commandes qui ont comme partie la notion "not found" ainsi que les autres mots de l'exemple. Rappelons qu'un exemple sera composé de trois commandes, c'est à dire si une commande comprend le mot "Ding", par exemple "Ding her there", les mots (commandes) qui sont associés au mot rejeté seront rejetés aussi, c'est à dire, l'exemple complet composé des commandes "Ding", "her" et "there" sera rejeté. Pour pouvoir aboutir à ce résultat, lorsqu'une commande arrive à la place "Vocabular Akn", la partie "exp" sera vérifiée. Si elle est égale a "Not found" elle sera envoyée à la place "Rjected", sinon, elle sera envoyée à la place "Test2 Akn". Dans cette place, on va revérifier la valeur de "exp", si elle est égale à "incomplete" elle sera envoyée à la place "rejected exp" (c'est le cas des mots "her" et "there " de l'exemple "Ding her there"). Sinon l'exemple qui sera composé de trois mots (commandes) retrouvés dans l'ontologie, sera envoyé à la place "Context verification". Cette dernière est liée à son tour à l'ontologie et permet la vérification de l'état du contexte et la sélection des modalités d'entrée disponibles.

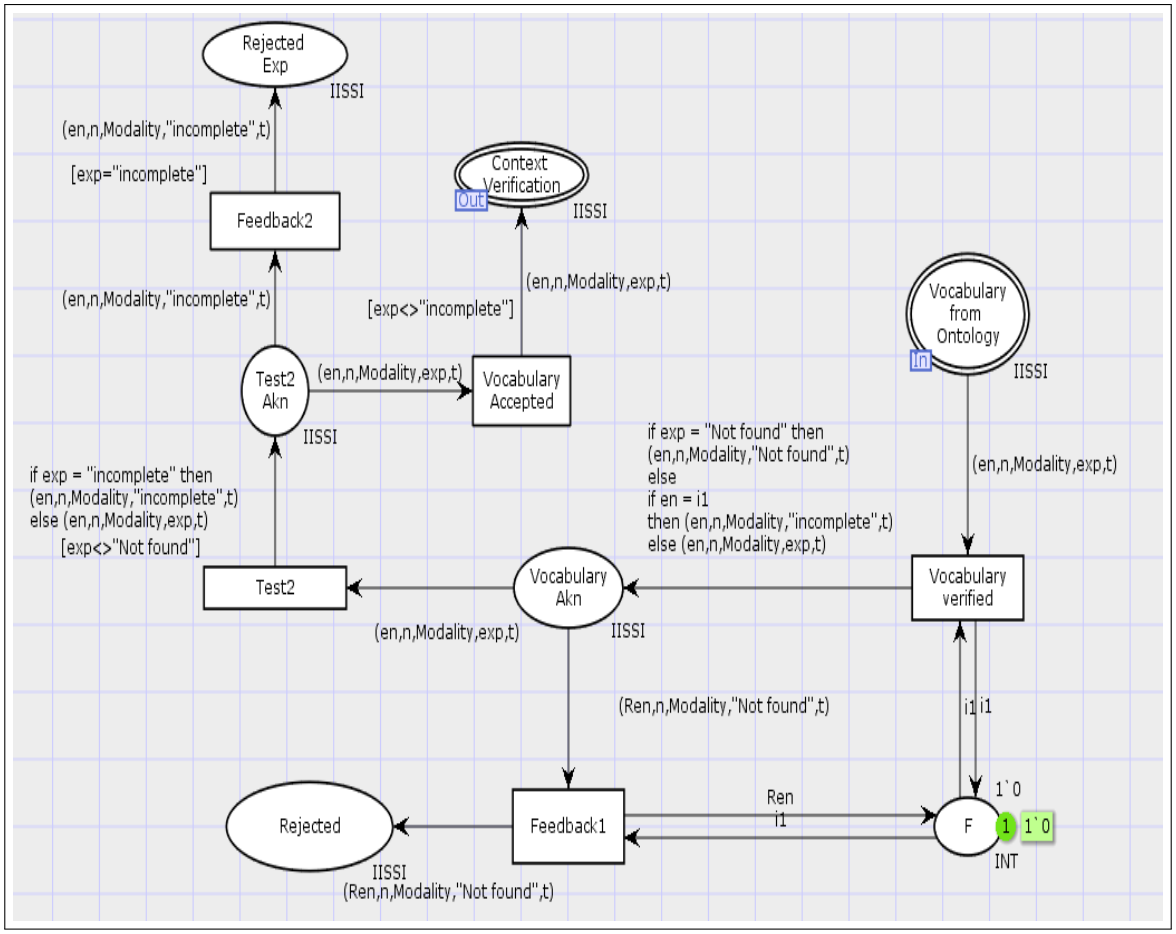


FIGURE 5.4: La transition “Vocabulary checker”

4. **Vérification du contexte d’entrée (Context verification)** : Lorsque les commandes ont été reconnues dans le vocabulaire de l’ontologie, nous allons procéder à la vérification du contexte d’entrée. La partie du module de l’ontologie responsable de la sélection de modalités est présentée sur la Figure 5.5. Cette figure présente la définition du contexte et la sélection de modalités. En effet, nous avons défini la place "H" pour le type de handicap, la place "Alr" pour l’alarme, la place "LL" pour le niveau de lumière et la place «NL» pour le niveau de bruit. Les endroits correspondants pour le contexte de sortie sont respectivement "HF", "ALRF", "LLF" et "NLF". Selon les valeurs de ces informations récupérées du contexte, les modalités d’entrée et les modalités de sortie seront sélectionnées.

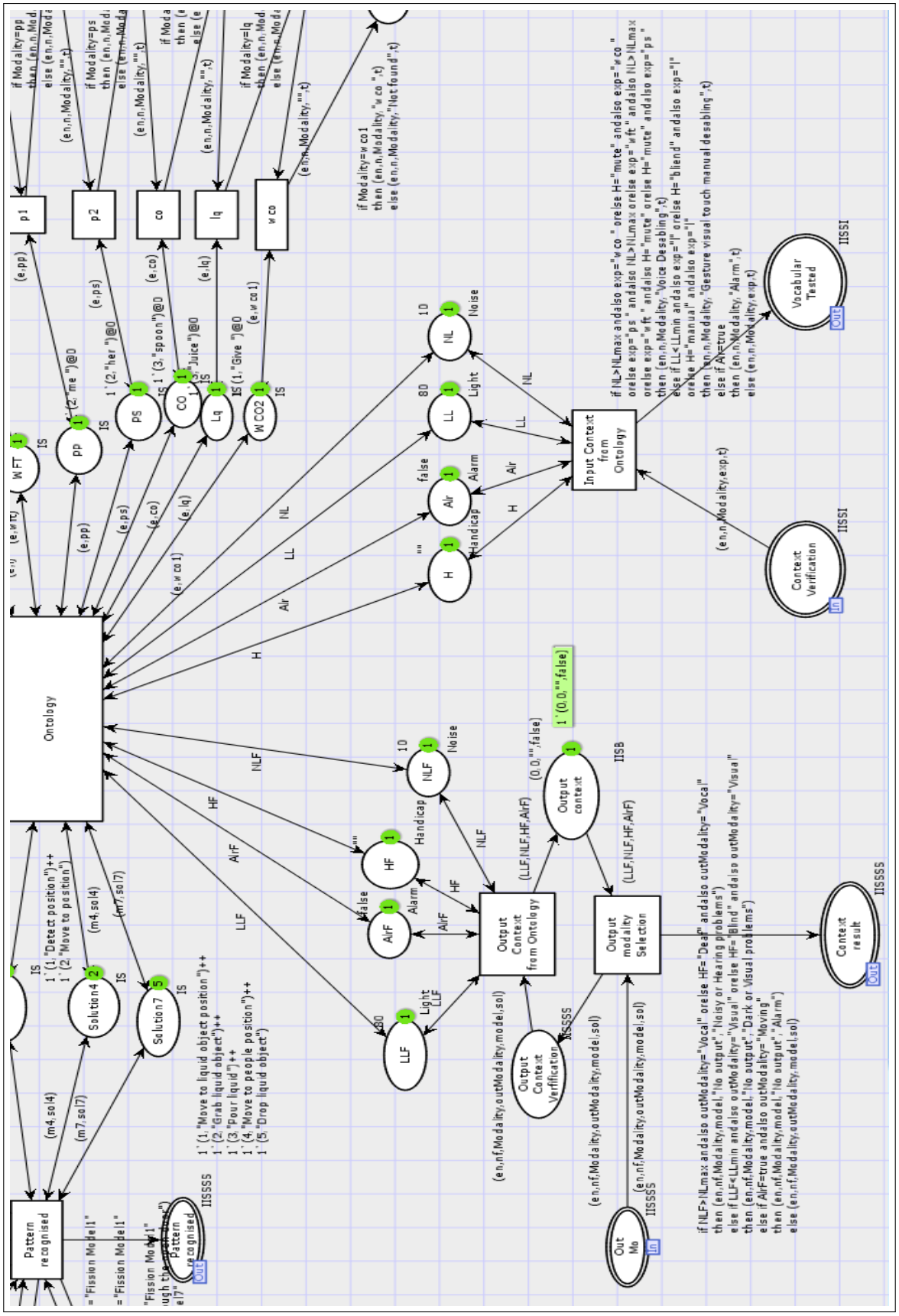


FIGURE 5.5: Partie de l'ontologie responsable de la sélection de modalités d'entrée et de sortie en fonction de l'état du contexte

Dans le cadre de cette validation, nous avons choisi de définir toutes les informations comme étant dans les intervalles de valeurs acceptés. Nous avons opté pour ce choix pour éviter l'arrêt du processus pour non disponibilité de modalités et pouvoir visualiser le comportement du moteur de fusion et du moteur de fission.

5. **Vérification temporelle (Time check) :** ce module vérifie le temps entre les modalités et le temps total de la commande. Si le temps est accepté, le processus se poursuivra, sinon le processus s'arrête et un message d'information sera envoyé a l'utilisateur. Après la sélection des modalités et la vérification du vocabulaire, le système doit vérifier le temps entre l'arrivée des modalités et le temps total de la commande. Cette vérification est effectuée de sorte que le système n'attendra pas indéfiniment l'arrivé d'une entrée. Si le temps entre deux modalités est supérieure à 5 secondes ou le temps total de la commande est plus de 15 secondes, l'exemple sera rejeté et envoyé à la place "MTime Rejeté" et "CTime Rejeté" respectivement. Le module responsable de cette vérification est présenté sur la Figure 5.6.

   Lorsqu'un exemple arrive, le temps de la modalité sera récupéré et stocké. Puis, pour ce qui est du temps entre les modalités le système procèdera à la soustraction des deux temps correspondant aux deux mots successifs. Et pour ce qui est du temps total de la commande le système procèdera à la soustraction du temps du troisième et du premier mot de la commande. Si le temps est accepté l'exemple sera envoyé à la place de sortie de ce module qui est "Ontology Model" pour un traitement ultérieur.

6. **Construction du modèle (Model building) :** La Figure 5.7 montre la partie "Model building". Rappelons que dans le Chapitre 3 nous avions décrit l'ontologie du système et les modèles utilisés pour le processus de fusion. Les modèles sont une représentation successive de classes de l'ontologie. Donc, dans cette validation, pour que le système soit en mesure de reconnaître un modèle, il faut comparer le modèle de l'exemple (la succession des classes de l'exemple) avec les modèles stockés dans l'ontologie.

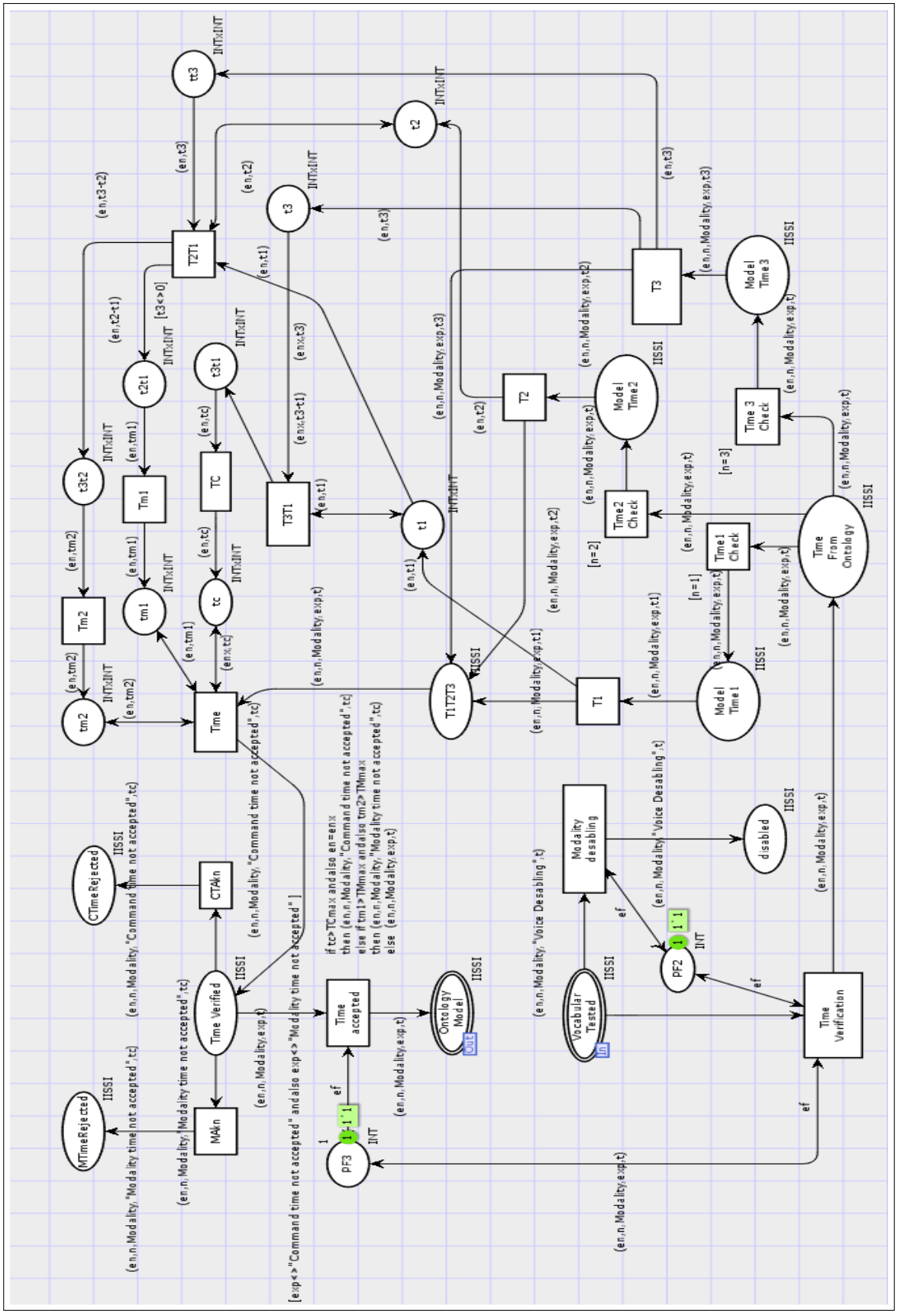


FIGURE 5.6: Vérification du temps

lorsque le mot a été reconnu par le vérificateur du vocabulaire, le nom de la classe correspondante a été ajouté. Ce module sera chargé de relier les noms des classes pour chaque exemple pour obtenir le modèle de la commande générée aléatoirement. C'est à dire lorsqu'une commande d'un exemple arrive à la transition "Event Model", le nom de la classe qui est dans "exp" sera enregistré et rajouté au nom du deuxième puis troisième mot de l'exemple, dans l'ordre, au sein de la place "Cmp1". Par exemple la combinaison "wco p co" (mots objets pratiques, personne, objet pratique) sera comparée avec les modèles de fusion prédéfinis dans l'ontologie comme représenté sur la Figure 5.8. Lorsque le modèle est reconnu, l'exemple sera envoyé au moteur de fusion de la Figure 5.9 ce qui permettra de fusionner les informations et conclure la solution de fission correspondante.

7. **Modèle reconnu (Model recognized) :** Ce module va rechercher dans l'ontologie le modèle correspondant au modèle de commande. Lorsque le modèle construit dans la partie précédente arrive à la place "Models from ontology", une comparaison sera faite avec les modèles définis dans les places "model 1" jusqu'à "Model 5" de l'ontologie. Si le modèle de l'exemple est reconnu, l'exemple de commande sera envoyée à la place "Model" avec le numéro du model. Dans le cas contraire, où la combinaison des classes de l'exemple ne correspond à aucun model, la notion "model not recognised" sera envoyée a la place du nom du modèle.

8. **Moteur de fusion (Fusion engine) :** ce module, représenté dans la partie gauche de la Figure 5.9, sera responsable de la fusion des modalités d'entrée en utilisant les modèles de fusion prédéfinis dans l'ontologie.

   Lorsque le modèle a été reconnu et le numéro du modèle a été attribué à chaque mot d'un exemple donné, la fusion peut être effectuée. Lorsqu'une commande d'un exemple arrive à la transition "Modality fusion", la modalité sera enregistrée et rajoutée à la modalité suivante pour former une commande unique composée de la fusion des trois mots (commandes), puis envoyée à la place "Fusion".

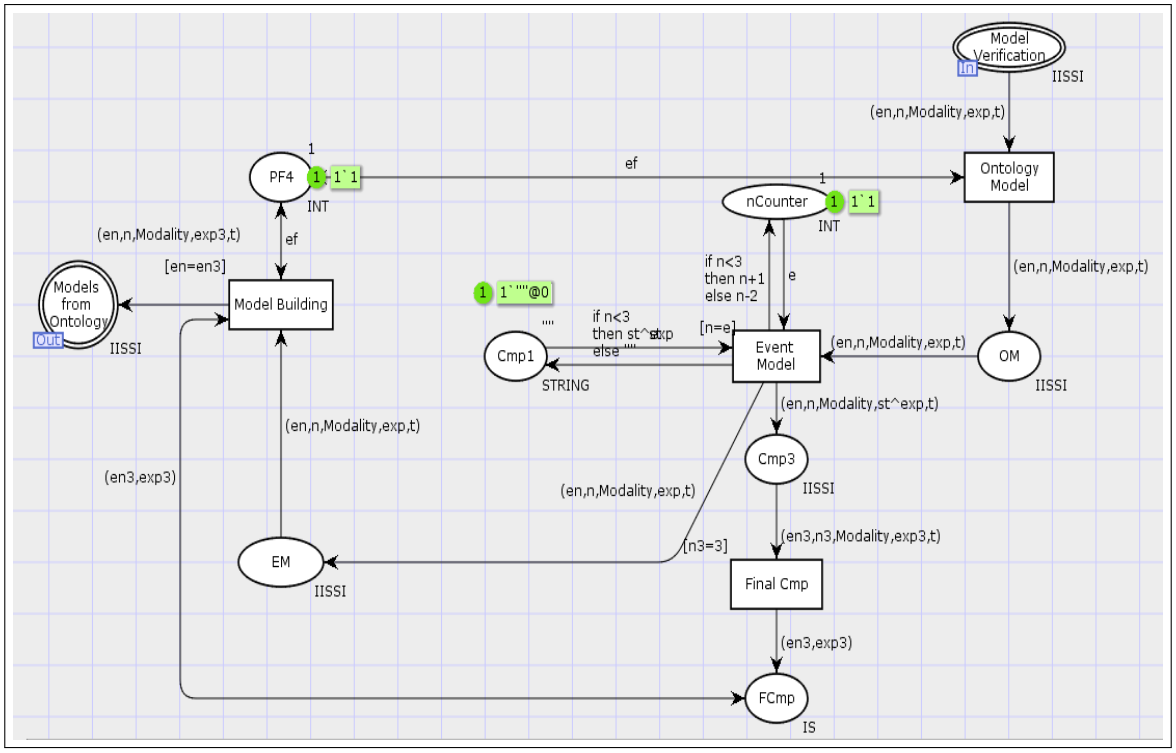


FIGURE 5.7: La transition "Model building"

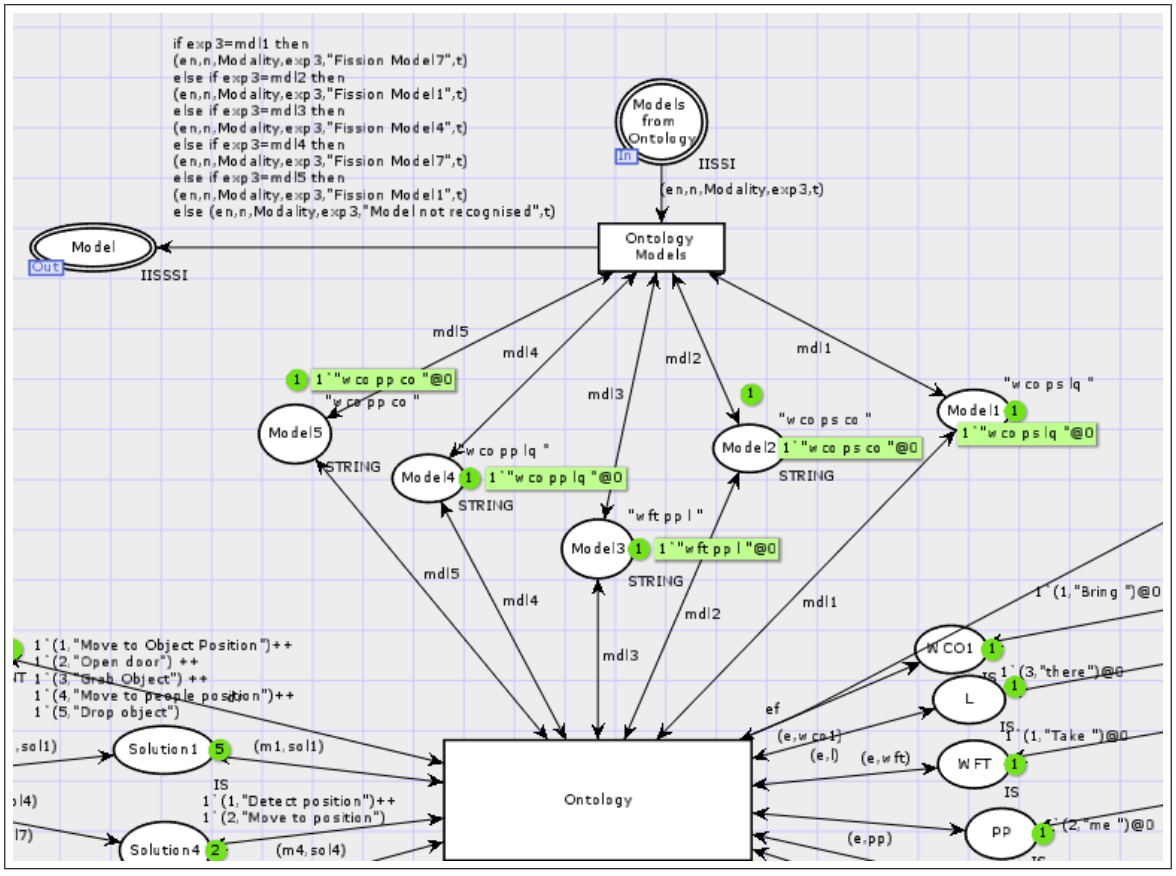


FIGURE 5.8: Partie de l'ontologie pour la définition des modèles de fusion

Si le motif de fission associé à un exemple est "fission model 7", cela implique qu'il faut introduire "cup" dans la commande finale. Cela veut tout simplement dire que la commande contient un liquide et il est impératif de l'apporter dans un verre.

9. **Résultat de fusion et motif de fission (Fusion result and fission patterns) :** Ce module va envoyer le résultat du moteur de fusion accompagné par la solution de fission correspondante.

   Dans la deuxième partie, représenté dans la partie droite de la Figure 5.9, nous allons filtrer les commandes et laisser passer que la dernière commande qui sera composée de la fusion des donnes d'entrée et envoyer le résultat de la fusion dans la partie "Modality" de l'arc (en=numero de l'exemple, c=compteur, Modality=la fusion, exp= composition des classe de l'exemple, model= model de fission).

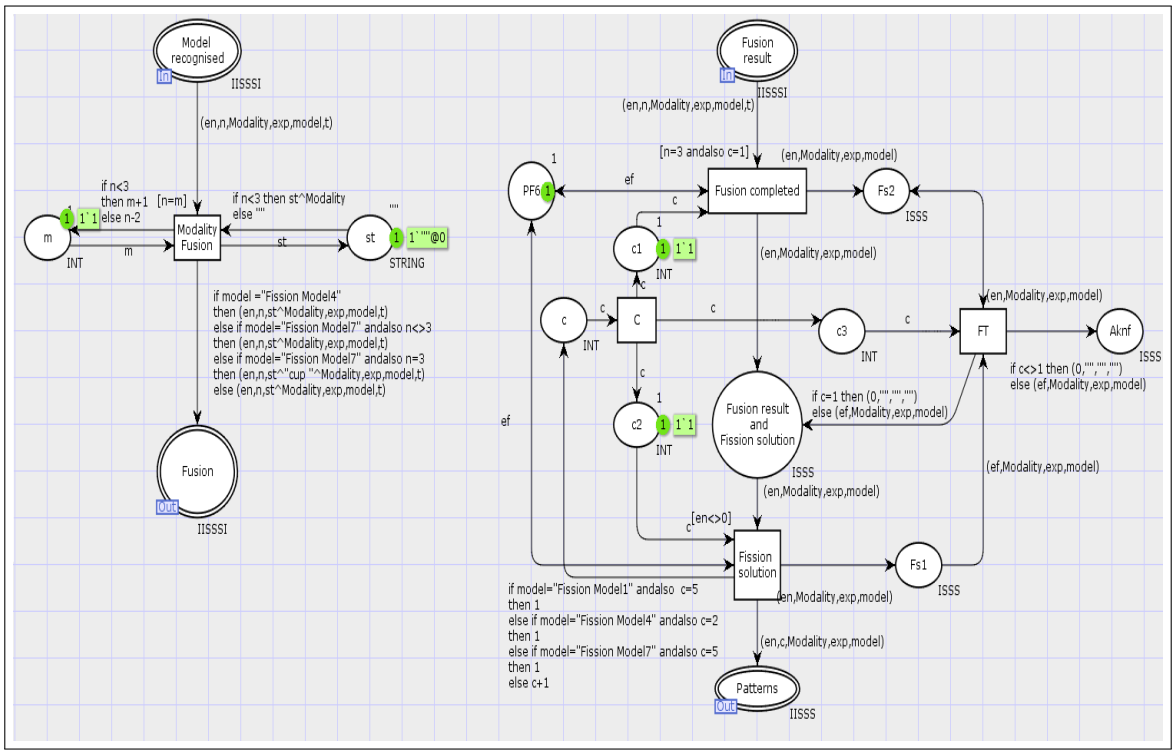


FIGURE 5.9: Moteur de fusion

10. **Moteur de fission (Fission engine) :** ce module va subdiviser le résultat de la

fusion en actions élémentaires en utilisant les solutions de fission prédéfinies dans l'ontologie.

La Figure 5.11 montre les solutions de fission possible pour nos trois scénarios stockés dans l'ontologie. Comme présenté auparavant dans le moteur de fusion, si l'utilisateur demande un liquide, le système comprend qu'un objet utilisé pour les liquides est primordial et doit être ajouté lors de la fusion des entrées. Par exemple lors de la fusion des entrées "donne", "moi" et "jus" le résultat de la fusion sera "donne-moi un verre de jus" et la solution de fission correspondante est la solution de Fission 7.

En outre, pour le processus de fission, nous avons aussi défini l'état de la porte dans l'ontologie. En effet, si la réponse d'une demande est de passer d'une pièce à l'autre en passant par une porte, nous devons informer le fauteuil roulant qu'il doit d'abord ouvrir la porte ou tout simplement passer à travers si elle est déjà ouverte. Si nous n'incluons pas cette contrainte, le robot va s'arrêter car la porte fermée sera détectée comme une barrière.

Par exemple, si la demande est «Apporte-moi une cuillère» sachant que l'utilisateur du fauteuil est dans le salon et que la cuillère est dans la cuisine et la porte est fermée, la solution de fission sera la solution 4 qui est : "se déplacer vers la position de l'objet" $\rightarrow$ "porte ouverte"$\rightarrow$ "saisir l'objet"$\rightarrow$ "se déplacer vers la personne" $\rightarrow$ "déposer l'objet".

Sur la Figure 5.11, on trouve le moteur de fission qui fournira la solution de fission finale et le type de solution donnée. Les types de solution sont :

- **Mouvement** : si la demande comprend le déplacement d'un endroit à un autre
- **Visuel** : si la demande comprend une réponse sur écran comme la demande "montre-moi le numéro de téléphone de mon père"
- **Vocal** : si la réponse doit être rendue à l'utilisation en utilisant le haut-parleur

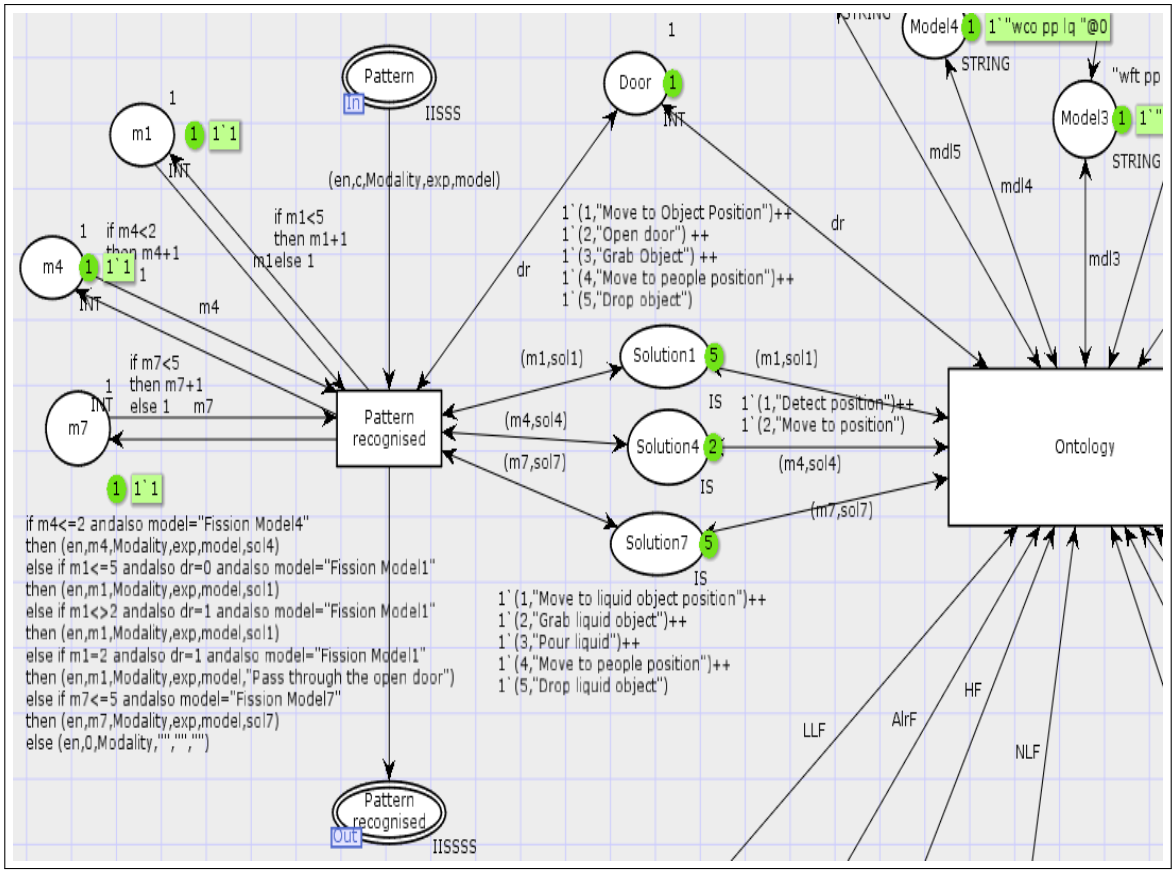


FIGURE 5.10: Solutions de fission dans l'ontologie

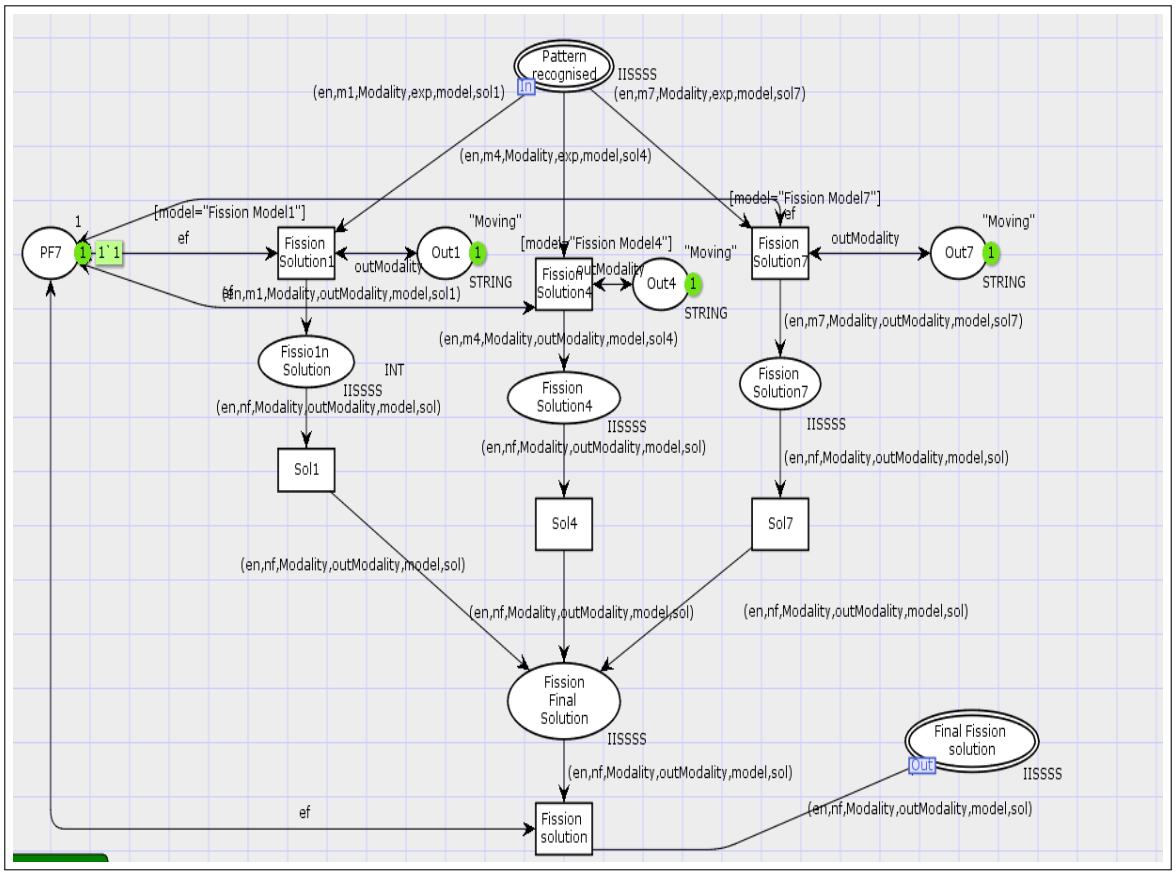


FIGURE 5.11: Solution de fission pour le moteur de fission

11. **Vérification du contexte de sortie (Output context verification) :** ce module vérifie la disponibilité des modalités de sortie selon l'état du contexte et envoie la réponse finale obtenue après fusion et fission des données aux actionneurs de sortie disponibles, comme présenté sur la Figure 5.5, et envoie le résultat final à la place "Command" (commande) comme indiqué sur la Figure 5.12.

Nous avons lancé la simulation pour des exemples aléatoires, la réponse se présente comme suit (Numéro de l'exemple, la position du mot dans l'exemple, le résultat de la fusion, type du résultat de sortie, numéro de la solution de Fission, sous-taches de la fission).

Pour le premier exemple, la simulation a généré l'exemple "Ding lui cuillère" qui n'est pas une commande valide. Le système a rejeté cet exemple comme le montre la Figure 5.12. Pour le deuxième exemple la simulation a généré la demande "Donne moi jus" et pour la troisième "apportez-lui cuillère".

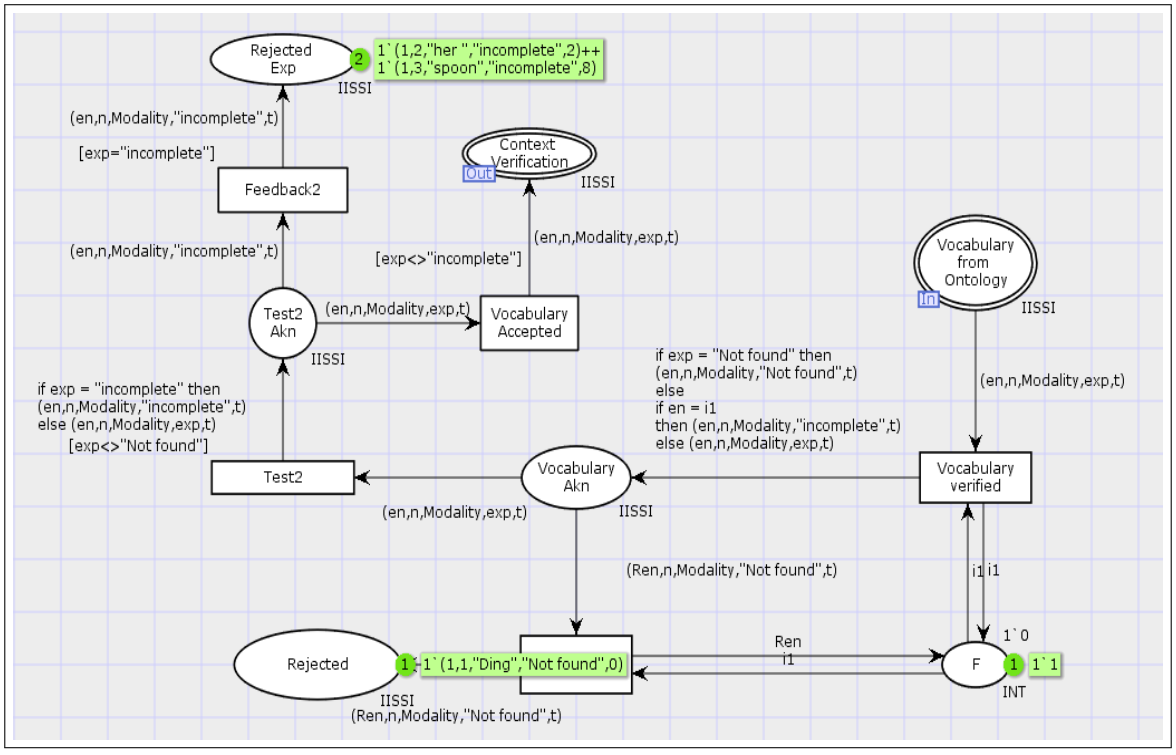


FIGURE 5.12: Exemple rejeté

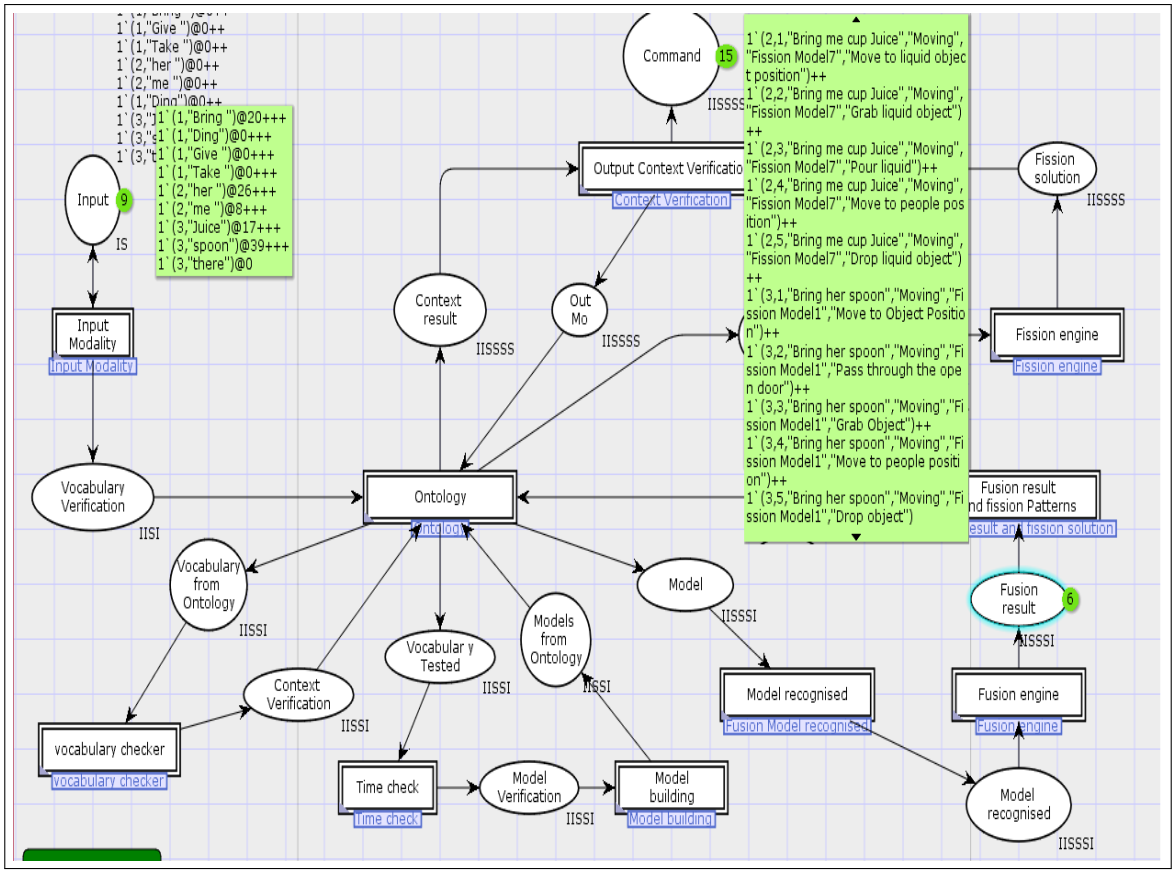


FIGURE 5.13: Résultat final

Comme le montre la Figure 5.13 notre architecture a fusionné avec succès les modalités d'entrée et subdivisé le résultat en sous-tâches élémentaires après avoir vérifié toutes les conditions. En outre, pour visualiser le comportement de notre architecture avec plusieurs entrées, nous avons choisi de générer au hasard 33 exemples. Comme mentionné précédemment, chaque exemple est composé de trois modalités d'entrées.

Le résultat de la première et la deuxième simulation est présenté sur la Figure 5.14 et la Figure 5.15, respectivement et la troisième simulation est présentée sur la Figure 5.16.

Comme on le voit sur la Figure 5.14, dans la première simulation cinq exemples ont été acceptés et traités par le système. Dix exemples ont été rejetés parce que le modèle ne correspondait pas à un modèle défini dans l'ontologie, par exemple l'exemple «Donnez-moi là». Aussi, sept et quatre exemples ont été rejetés car le temps total de la commande et le temps entre les modalités sont supérieurs aux temps maximum acceptés. Enfin sept

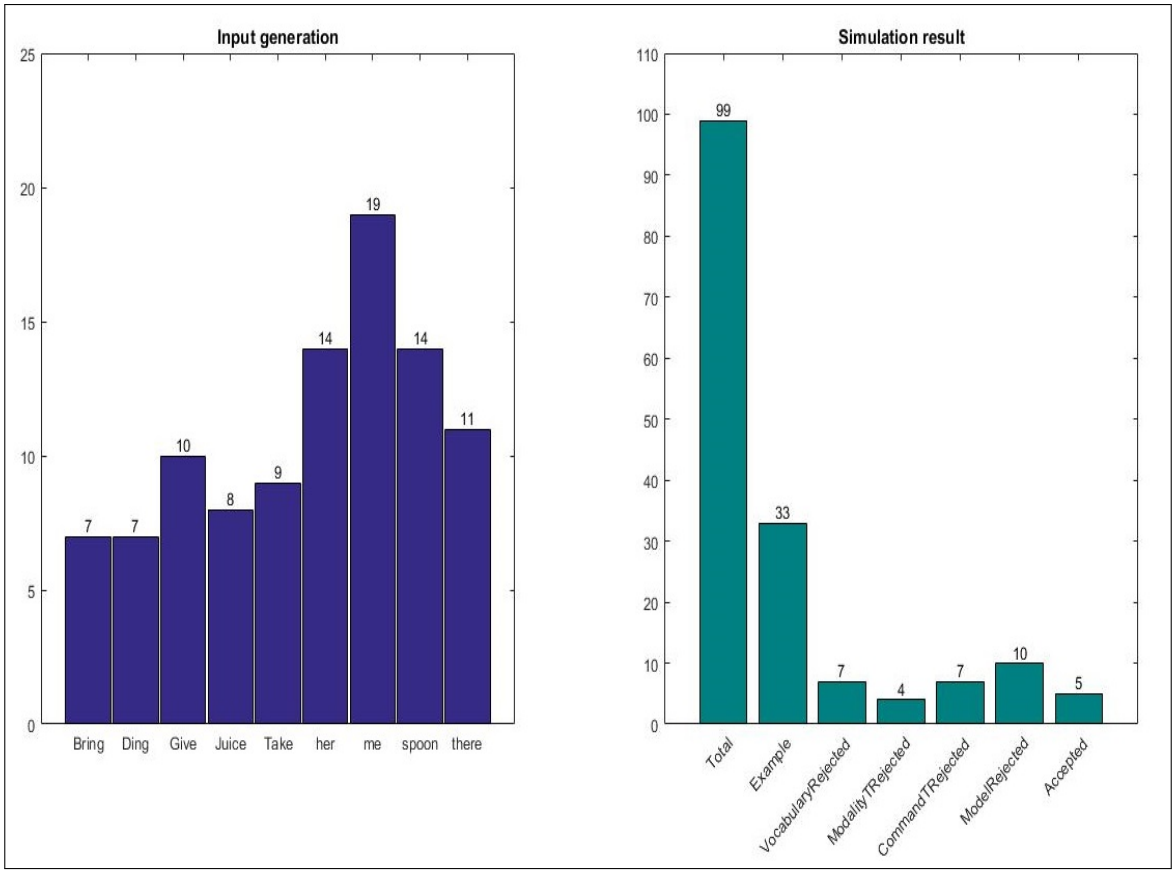


FIGURE 5.14: Résultat de la première simulation

exemples ont été rejetés car le vocabulaire utilisé n'a pas été défini dans l'ontologie, à savoir «Ding».

En outre, pour la deuxième simulation, la Figure 5.15 montre que onze exemples ont été acceptés. Six exemples ont été rejetés car le modèle utilisé pour la commande ne correspond pas aux modèles prédéfinis dans l'ontologie. Aussi, trois et six exemples ont été rejetés à cause du temps de commande et du temps entre les modalités respectivement. Et enfin, sept exemples ont été rejetés car le vocabulaire ne correspondait pas au vocabulaire prédéfini dans l'ontologie.

Enfin, pour la troisième et dernière simulation qui est présentée sur la Figure 5.16 nous avons onze modèles acceptés et fusionnés puis fissionnés correctement. Pour ce qui est des modèles rejetés, nous avons cinq modèles rejetés à cause du modèle qui n'est pas conforme aux modèles définis dans l'ontologie. Quatre ont été rejetés en raison du temps total de commande et quatre à cause du temps entre chaque commande. Enfin

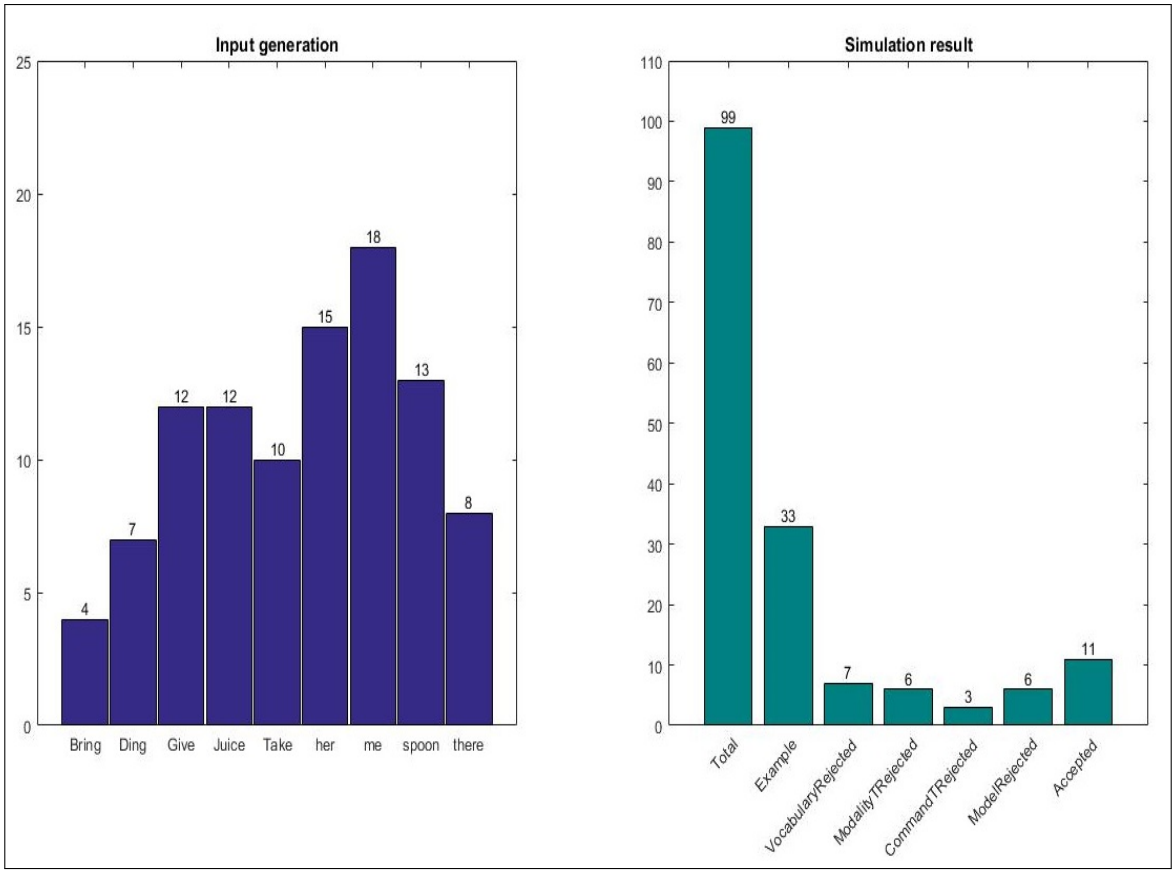


FIGURE 5.15: Résultat de la deuxième simulation

neuf exemples ont été rejetés à cause du vocabulaire utilisé.

Selon ces simulations, nous concluons que notre système est en mesure de comprendre les modalités d'entrée, tandis que les commandes qui ne correspondent pas aux conditions préalablement définies sont rejetées. Aussi, notre architecture a pu fusionner et fissionner correctement les commandes qui ont un sens et envoyer une réponse à l'utilisateur. Nous avons démontré que l'architecture proposée est un système autonome qui est en mesure de comprendre l'environnement et de décider en répondant à la demande de l'utilisateur. L'utilisation des moteurs de fusion et de fission tout en prenant en compte l'état du contexte évolutif, ainsi que l'utilisation du concept d'ontologie comme base de connaissances a permis d'offrir le service souhaité par l'utilisateur.

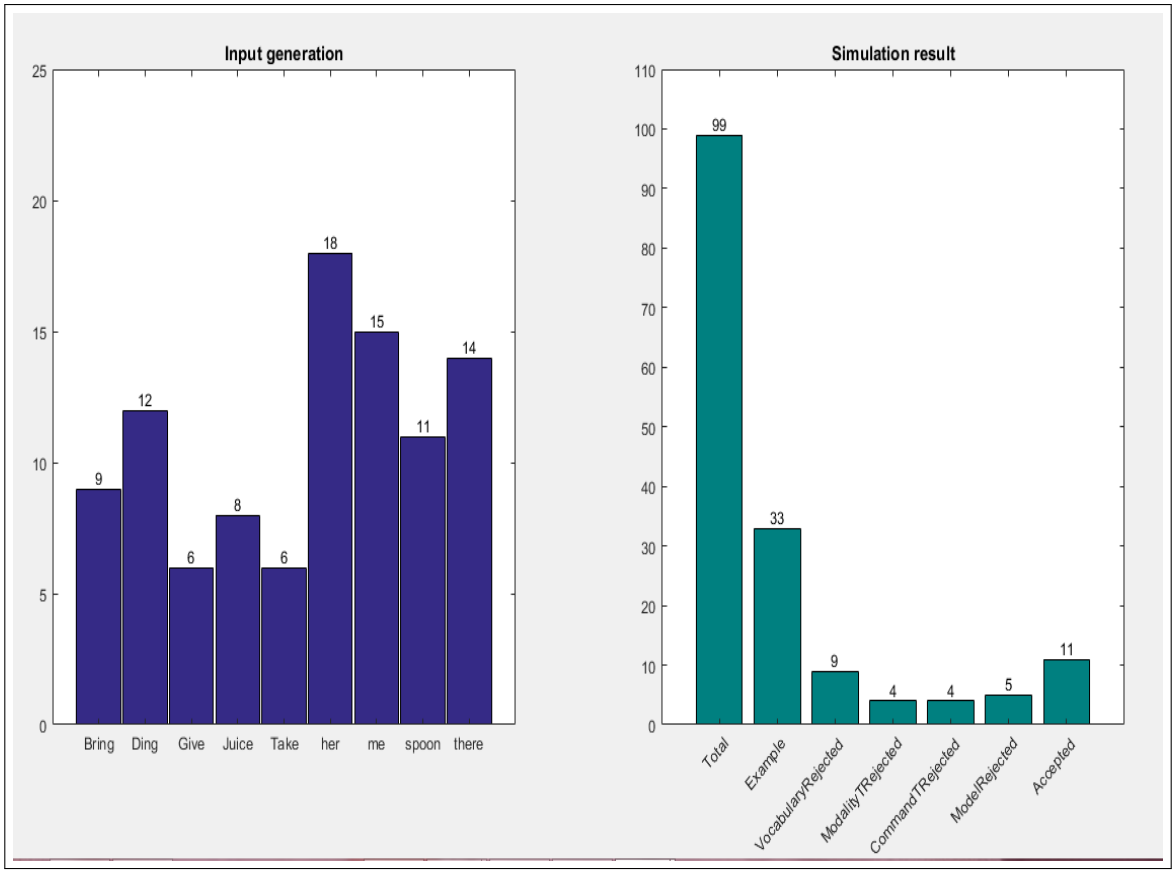


FIGURE 5.16: Résultat de la troisième simulation

5.4 Implementation en temps réel

Dans ce travail, nous avons présenté une architecture multimodale qui permet de répondre à la commande d'un utilisateur d'une chaise roulante muni d'un bras manipulateur. Cette architecture permet l'interaction entre le robot (fauteuil roulant) et l'utilisateur. Pour valider notre approche sur un robot réel, nous avons choisi d'appliquer un scénario simple à l'aide du robot Nao ([79] [80] [81]). Certes, Nao est un humanoïde et non un fauteuil roulant, nous avons décidé de valider notre approche en utilisant ce robot car il est déjà fonctionnel dans notre laboratoire. Le fauteuil roulant robotisé est en cour de conception par une autre équipe du laboratoire et n'est pas encore fonctionnel.

La validation à l'aide de Nao prouve également que notre architecture, initialement conçue pour un robot de type fauteuil roulant, peut être adaptée à tout type de robot dans le but d'assistance par l'interaction homme-machine.

5.4.1 Déscription technique

Pour exécuter certains scénarios, nous utilisons des wrappers et un driver qui pilote la machine. Le driver lit et intègre les événements issus du réseau pour lire ou agir sur le matériel. Les wrappers prennent plusieurs données et les transforment en un ou plusieurs événements. Le réseau gère des évènements sous forme de prédicas avec arguments (move (object, x, y)).

Un wrapper a une liste de modèles. Pour un exemple de saisie, le wrapper ajoute des données de capteurs dans le modèle d'événements puis il envoie cet évènement concret aux agents de fusion. Pour l'exemple de sortie, le wrapper reçoit l'événement d'un agent (agents de fission), récupère les données de l'événement pour piloter les actionneurs, en envoyant un message texte ou vocal aux actionneurs impliqués directement ou à travers une communication réseau. Voir la figure 5.17.

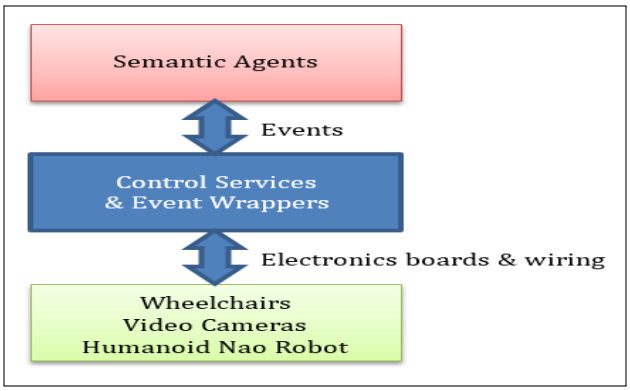


FIGURE 5.17: Relation entre les éléments de l'architecture

5.4.2 Scénario

Ici, le scénario qui va être présenté est celui du robot qui rappelle à une personne de prendre son médicament. La tâche du robot Nao est de s'occuper d'une personne âgée nommée Amar. Le robot reçoit les coordonnées de la position de la personne. Il interprète le discours et les gestes de la personne à travers Kinect, ce qui permet de reconnaître l'état de l'utilisateur (c'est-à-dire actif, dormant, entrain de regarder la télévision, déplace ses bras, marche, se tenir debout ou parler). Voir la figure 5.18.

FIGURE 5.18: Kinect : capteur visuel et audio. Nao : actionneur gestuel et vocal

Nao doit rappeler l'utilisateur de prendre son médicament car il est 10h. Si l'utilisateur répond, Nao attend le prochain évènement. Dans le cas contraire, Nao doit utiliser le geste pour réveiller l'utilisateur. Si l'utilisateur ne réagit toujours pas, Nao doit se déplacer vers l'utilisateur et le toucher. S'il se réveille il le rappelle de prendre son médicament sinon il envoie une alarme au personnel de santé.

Nous avons adapté notre architecture présentée dans les chapitres précédents pour pouvoir l'appliquer sur le robot Nao. Le contexte reste le même, nous avons rajouté un évènement propre à ce scenario dans la classe "Event" qui a pour objectif de déclencher la vérification du temps. Nous avons aussi rajouté une sous classe dans la classe "Time" (temps) qui est "Current Time" (temps actuel) pour que le robot puisse vérifier le temps de prise de médicaments. Nous avons rajouté la classe "User state" (état de l'utilisateur) et "User position" comme sous classe de la classe "User context". Pour la classe "User state" nous avons déclaré deux instances qui sont "answering" et "not answering" (répondre et pas répondre) qui vont permettre de vérifier l'état de l'utilisateur (s'il répond aux sollicitations du robot ou non). Tandis que la classe "User position" a comme instance "Current position" (Position actuel) qui aura comme valeur les coordonnés (x, y) de la position de l'utilisateur. Enfin la classe "Fusion_Fisson_Nao" (qui a

trois sous classe) est une sous classe de la classe "Fission models" où nous avons déclaré les modèles de fusion et de fission relatifs au scenario appliqué sur Nao.

De plus, nous avons rajouté quelques propriétés. La propriété d'objet "has_Nao_Next" a été définie pour préciser l'ordre des actions à exécuter par le robot Nao. La propriété "has_Action" est une propriété de donné de type "string" qui permet de définir l'action à réaliser par le robot. La propriété "has_detected_state" est une propriété de donné de type "Boolen" qui permet de donner l'information sur la réaction de l'utilisateur (répond ou pas). De plus, la propriété d'objet "has_position_coordinates_x(y)" permet de définir les coordonnées de l'instance de la classe "user position".

Nous avons déclaré trois requêtes qui vont permettre au robot de fusionner les informations et de fournir une action, la requête du modèle 1 est :

systemOntology : Event_Nao(?Event) ∧ systemOntology : Current_Time(?Check) ∧ systemOntology : has_Time(?Check, ?Time) ∧ systemOntology : Fission_Nao_1(?y) ∧ systemOntology : has_Action(?y, ?Action1) ∧ systemOntology : User_state(?position) ∧systemOntology : has_detected_state(?position, ?Result) ∧ swrlb : equal(?Result, true) ∧ systemOntology : Fission_Nao_2(?WhatNext) ∧ tbox : isDirectSubClassOf(?Class, systemOntology : Fusion_Fission_Nao) ∧ abox : hasIndividual(?Class, ?state) ∧ systemOntology : has_Nao_Next(?position, ?Next) ∧ systemOntology : has_Nao_Next(?y, ?w) ∧ tbox : equalTo(?y, ?Next) ∧ systemOntology : has_Nao_Next(?y, ?whatdoNext) →sqwrl : selectDistinct(?Check, ?Time, ?Action1, ?position, ?whatdoNext)

En exécutant cette requête, nous allons d'abord retrouver l'évènement relatif à Nao qui est "Event_Nao", puis nous allons vérifier le temps avec le temps de prise de médicament qui est 10h. Lorsque Nao rappelle l'utilisateur de prendre son médicament et il répond, Nao attend un autre évènement, comme montre sur la Figure 5.20. Sinon Nao passe à l'action 2 qui est de faire un geste comme présenté sur la Figure 5.19. . Le résultat de l'exécution de cette requête est présenté sur la Figure 5.19 pour le cas où l'utilisateur ne répond pas et sur la Figure 5.20 pour le cas où l'utilisateur répond.

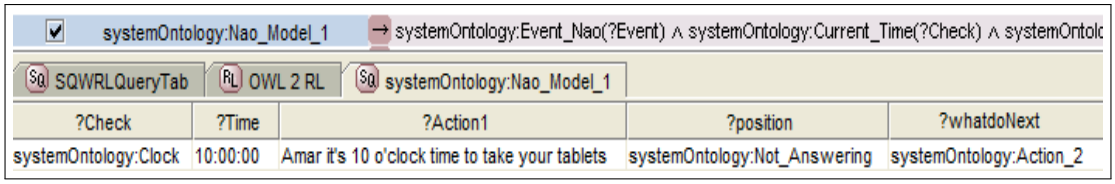


FIGURE 5.19: Résultat de l'exécution de la requête du modèle 1 lorsque l'utilisateur ne répond pas

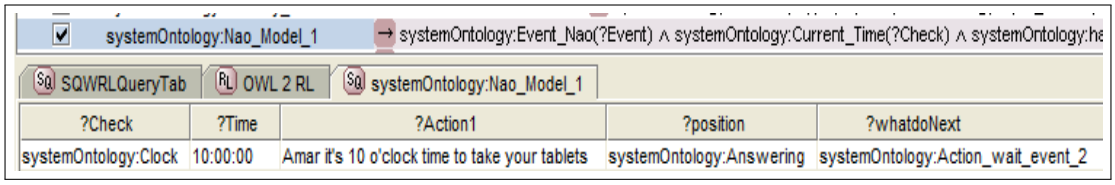


FIGURE 5.20: Résultat de l'exécution de la requête du modèle 1 lorsque l'utilisateur répond

Pour ce qui est du modèle 2, la requête est :

systemOntology :Fission_Nao_2(?What) ∧ systemOntology :has_Action(?What, ?Action) ∧ systemOntology :User_state(?position)∧ systemOntology :has_detected_state(?position, ?Result) ∧ swrlb :equal(?Result, true) ∧ systemOntology :Fission_Nao_3(?WhatNext) ∧ tbox :isDirectSubClassOf(?Class, systemOntology :Fusion_Fission_Nao) ∧ abox :hasIndividual(?Class, ?state) ∧ systemOntology :has_Nao_Next(?position, ?Next) ∧ systemOntology :has_Nao_Next(?What, ?whatdoNext) ∧ tbox :equalTo(?What, ?Next) ∧ systemOntology :has_Action_2(?What, ?Say) → sqwrl :selectDistinct(?Action, ?Say, ?position, ?whatdoNext)

Le résultat de l'exécution de cette requête est présenté sur la Figure 5.21 pour le cas où l'utilisateur ne répond pas et sur la Figure 5.22 pour le cas où l'utilisateur répond.

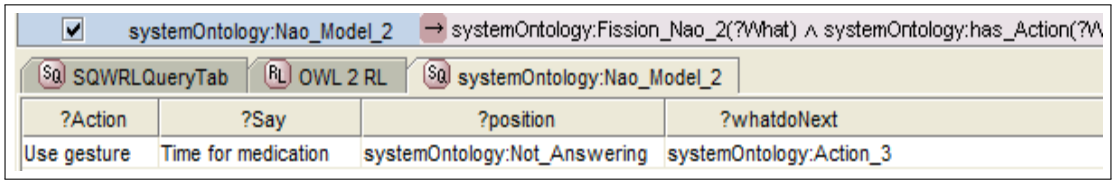


FIGURE 5.21: Résultat de l'exécution de la requête du modèle 2 lorsque l'utilisateur ne répond pas

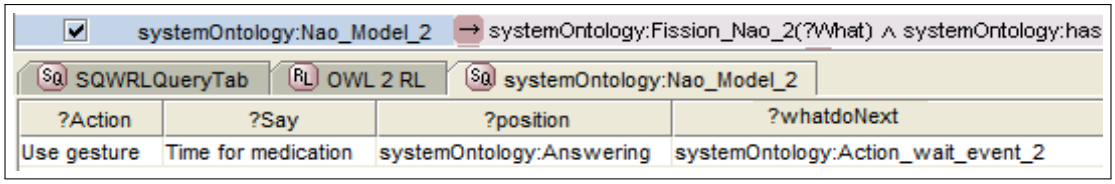


FIGURE 5.22: Résultat de l'exécution de la requête du modèle 2 lorsque l'utilisateur répond

Pour ce qui est du modèle 3, la requête est :

systemOntology :Fission_Nao_3(?What) ∧ systemOntology :has_Action(?What, ?Action) ∧ systemOntology :User_state(?State) ∧ systemOntology :User_Position(?p) ∧ systemOntology :has_position_coordinates_x(?p, ?x) ∧ systemOntology :has_position_coordinates_y(?p, ?y) ∧ systemOntology :has_detected_state(?State, ?Result) ∧ swrlb :equal(?Result, true) ∧ tbox :isDirectSubClassOf(?Class, systemOntology :Fusion_Fission_Nao) ∧ abox :hasIndividual(?Class, ?s) ∧ systemOntology :has_Nao_Next(?State, ?Next) ∧ systemOntology :Fission_Nao_4(?WhatNext) ∧ systemOntology : has_Nao_Next(?What, ?w) ∧ systemOntology :has_Nao_Next(?WhatNext, ?whatDoNext) ∧ tbox :equalTo(?What, ?Next) ∧ tbox :equalTo(?w, ?WhatNext) ∧ systemOntology :has_Action_3(?What, ?do) → sqwrl :selectDistinct(?Action, ?x, ?y, ?do, ?State, ?whatDoNext)

Le résultat de l'exécution de cette requête est présenté sur la Figure 5.23 pour le cas où l'utilisateur ne répond pas et sur la Figure 5.24 pour le cas où l'utilisateur répond.

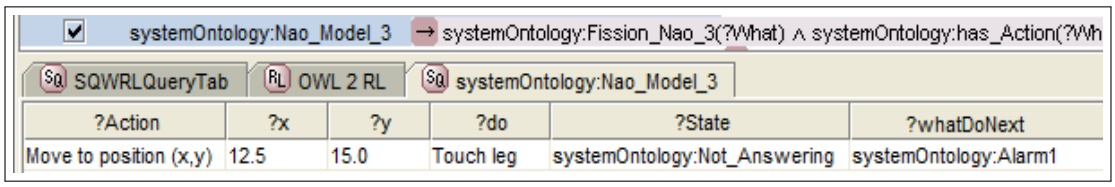


FIGURE 5.23: Résultat de l'exécution de la requête du modèle 3 lorsque l'utilisateur ne répond pas

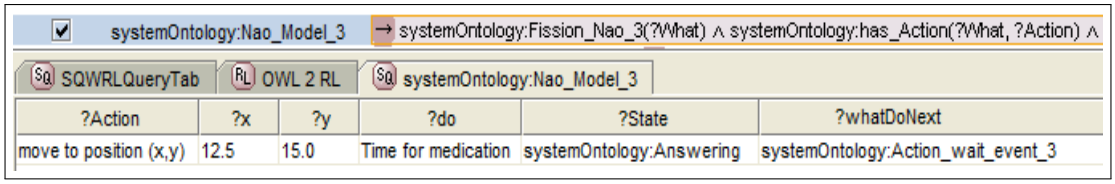


FIGURE 5.24: Résultat de l'exécution de la requête du modèle 3 lorsque l'utilisateur répond

5.4.3 Application sur Nao

Pour ce qui est de l'exécution sur Nao, le Kinect utilisé est muni d'événements de la reconnaissance vocale, gestes et de la reconnaissance d'objets. Le wrapper est dans l'ordinateur qui est connecté au Kinect via un USB ; Il utilise le réseau Ethernet pour communiquer les événements aux agents responsables des processus de fusion et de fission.

Le robot Nao reçoit des événements via Wi-Fi (réseau Ethernet). Un wrapper situé à l'intérieur du robot utilise le broker NaoQi développé par Aldebaran [82] pour donner des ordres au robot (synthèse vocale, gestes, marche, arrêt, reconnaissance d'objets et reconnaissance faciale). Le robot accède également aux enregistrements médicaux (par exemple, des maladies telles que le diabète) stockées dans l'ordinateur du patient via une connexion Wi-Fi. Voir la figure 5.25.

Nao est également capable d'obtenir d'autres informations pertinentes sur l'utilisateur en utilisant d'autres formes de modalités. Par exemple, le battement de cœur de la personne peut être obtenu via la montre LG Urban attachée au poignet de l'utilisateur. Grâce au calendrier Google Agenda de la personne, on pourrait avoir l'heure de prise de médicaments par le malade. Ceci dit, le journal médical doit être pré-rempli par un personnel médical compétent.

Supposons que le robot sait que Amar doit prendre des médicaments à 10h00 et remarque que Amar est allongé sur le fauteuil dans un état de sommeil. Par conséquent, le robot va fusionner les informations reçu des entrées qui sont l'heure de prise de médicaments et Amar ne bouge pas. Nous avons adapté notre architecture pour pouvoir l'adapter au robot Nao. Dans ce cas de figure, le robot a vérifié le temps de prise des médicaments puis a vérifié l'état de Amar. Compte tenu de toutes ces informations d'en-

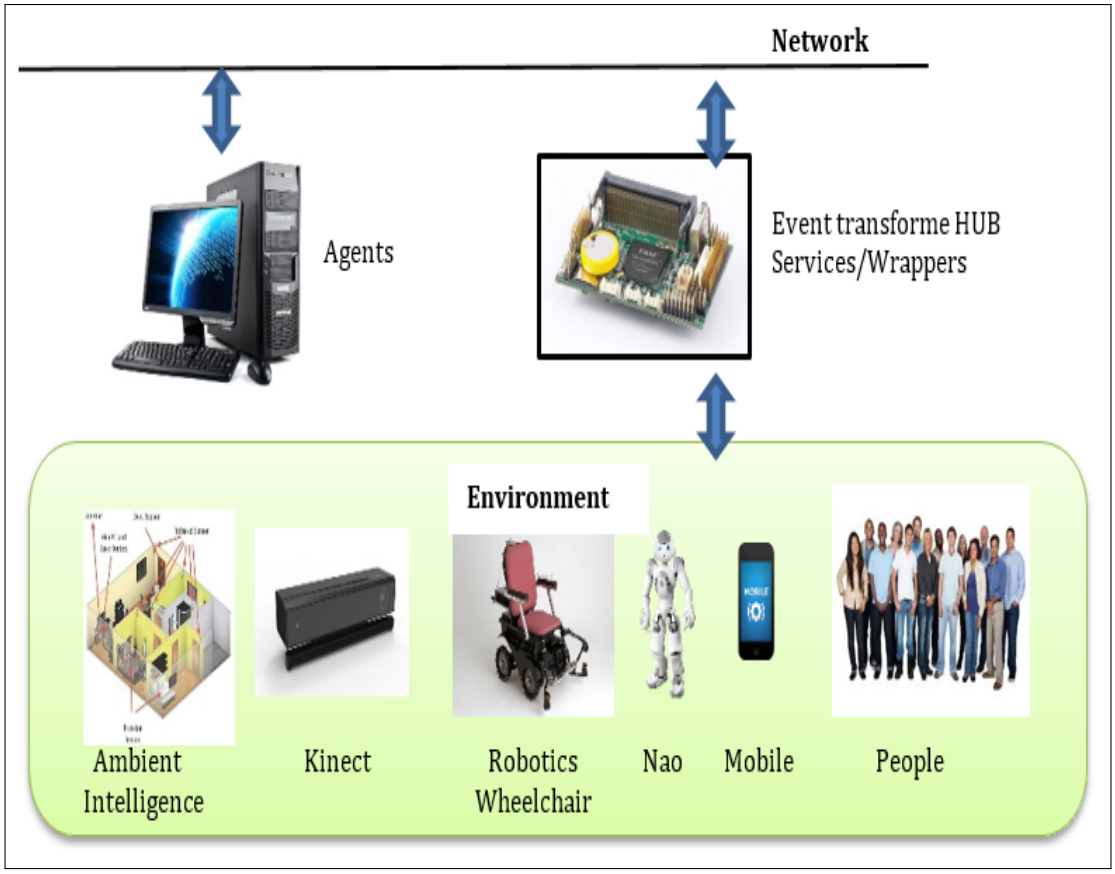


FIGURE 5.25: Communication d'événements

trée, le moteur de fusion combine toutes les informations et fournie l'action souhaitée, qui est : rappelez Amar de prendre ses médicaments. Le moteur de fission envoie le résultat à Nao qui enverra alors un rappel vocal à Amar. Le robot reste dans sa position et prononce "Amar, il est temps de prendre vos médicaments".

Le scénario est si Amar ne répond pas et le Kinect indique que l'état du sujet est "dormir". Nao va refaire la fusion des données avec ces nouvelles informations (heure de prise de médicaments et Amar ne répond par il dort). Le robot tente à nouveau avec une autre modalité qui est le geste en déplaçant son bras pour faire un signe à Amar (voir Figure 5.26). Si Amar ne répond toujours pas, le système obtient un nouveau contexte (c'est-à-dire une nouvelle situation) et envoie les nouvelles informations d'entrée au moteur de fusion. Le nouveau contexte qui nécessite une nouvelle action (qui est se déplacer vers Amar) est alors géré par le moteur de fission. En tant que tel, le robot s'approche de la position d'Amar. Ici, Nao essaiera de le réveiller en touchant sa jambe

(Figure 5.27).

Le prochain scénario dépendra des réponses de la personne à l'action du robot. Si le sujet se réveille, Nao lui rappellera de prendre ses médicaments par un message vocal. Si le sujet n'a pas de réaction, le robot envoie un message d'urgence au personnel médical via le réseau lui demandant de venir voir ce qui s'est passé.

FIGURE 5.26: Nao utilise un geste pour informer Amar qu'il doit prendre ses médicaments

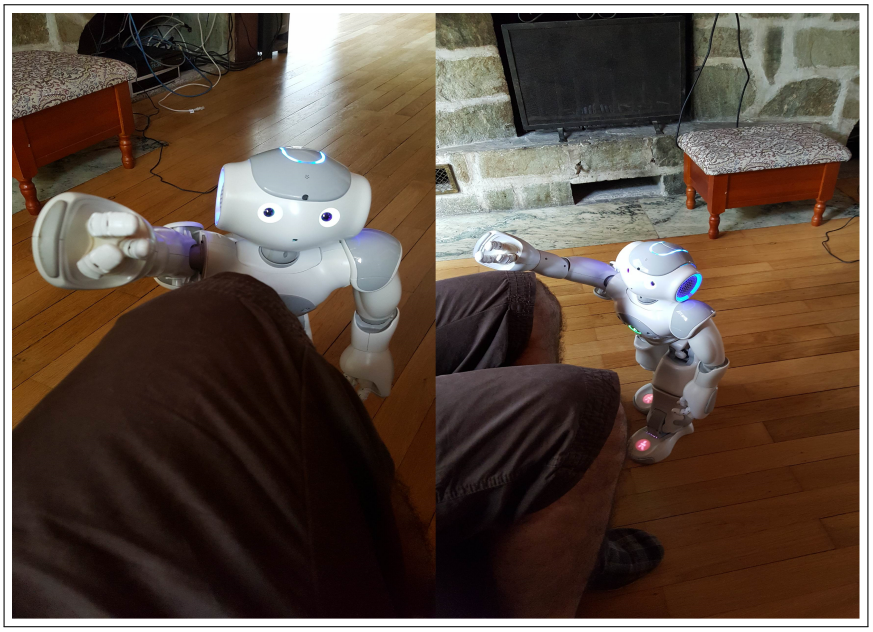


FIGURE 5.27: Nao touchant le sujet pour le réveiller

5.5 Conclusion

Nous avons présenté dans ce chapitre la validation de l'architecture proposée pour faciliter l'interaction homme-machine en prenant en compte l'état du contexte sémantique. La validation par réseau de Pétri en utilisant l'outil CPN Tools nous a permis de modéliser le fonctionnement de l'architecture proposée en prenant en compte tous les aspects définis dans les chapitres précédents et la visualisation du résultat. En effet, nous avons pu construire une architecture en utilisant les réseaux de Pétri sous forme d'un graphe hiérarchique composé de plusieurs éléments de l'architecture. La collaboration de ces modules nous a permis d'aboutir à la fusion des évènements d'entrées et à la fission du résultat du moteur de fusion et de répondre à la requête de l'utilisateur.

La simulation présentée dans ce chapitre nous a permis de voir le fonctionnement du moteur de fusion qui vérifie toutes les préconditions avant de décider si la fusion doit avoir lieu. Aussi, elle nous a permis de voir comment le moteur de fission subdivise le résultat de fusion en suivant aussi les préconditions.

L'utilisation d'un générateur aléatoire de mots de commandes à l'entrée du réseau de Pétri nous a permis de générer trente trois commandes différentes. Étant donné que les commandes variaient entre des commandes sans aucun sens et des commandes correctes, cela nous a permis de voir comment les moteurs de fusion et de fission peuvent comprendre une commande et rejeter des commandes erronées tout en prenant en compte le temps. Cet aspect est très important dans une interaction multimodale car elle est souvent utilisée dans des environnements dynamiques.

Conclusion Générale et Perspectives

Nous avons présenté dans ce mémoire un système d'interaction homme-machine-environnement dédié à un utilisateur de fauteuil roulant muni d'un bras manipulateur. L'utilisation de systèmes multimodaux rend l'interaction entre l'humain et la machine plus facile du fait de l'utilisation de moyens naturels de communication tels que la parole ou le geste.

Nous avons développé un moteur de fusion et de fission multimodale en utilisant un système intelligent qui prend en compte l'information contextuelle. Dans l'architecture proposée, les moteurs de fusion et de fission sont basés sur le mécanisme de préconditions stockées dans l'ontologie.

L'intégration du concept d'ontologie dans notre architecture a démontré son impact sur les systèmes d'interaction homme-machine. En effet, l'utilisation de l'ontologie a permis la description complète de l'environnement d'évolution du système étudié (l'utilisateur et le fauteuil) et a assuré une compréhension commune de ses entités. Aussi, l'utilisation du concept d'ontologie comme base de connaissance a assuré la réutilisabilité des informations stockées à tout moment. Ainsi que l'ouverture du système par la possibilité d'ajout ou de suppression d'informations selon le domaine d'utilisation et selon le profil de l'utilisateur.

L'ontologie a été définie à l'aide de classes, de propriétés et d'instances qui ont permis la description de l'environnement de notre étude de cas. Elle est aussi ouverte pour toute adaptation à un autre cas d'étude en ajoutant ou en supprimant des informations au sein de l'ontologie. En outre, nous avons réussi à adapter le modèle WWHT qui a été initialement conçu pour gérer les modalités d'interaction Homme-Machine, à un système fondé sur des modalités d'action telles que le déplacement d'un fauteuil roulant et

l'actionnement d'un bras robotisé. Nous avons validé notre approche en construisant un modèle en utilisant l'outil CPN Tools pour la conception de réseau de Pétri temporel et stochastique.

La simulation a prouvé que le système est capable de reconnaître les événements d'entrée et répondre à la demande d'un utilisateur en utilisant différents modules. Aussi le système est en mesure d'envoyer des tâches uni-modales à des modalités de sortie après fusion et fission des événements d'entrée. La simulation a démontré que notre système est capable de détecter une fausse entrée et de rejeter cette commande, et de rejeter un exemple si les temps ne sont pas acceptés et informer l'utilisateur de ces contraintes. Ainsi, en utilisant CPN Tools nous avons pu modéliser et valider les exigences temporelles de notre système en plus des exigences fonctionnelles.

Nous estimons que l'architecture d'interaction multimodale proposée dans ce mémoire contribue à l'avancée des recherches dans le domaine des systèmes d'interaction homme-machine-environnement. En effet, un système qui est capable de comprendre un environnement est un système autonome qui est capable de décider et effectuer la tâche conformément à une requête de l'utilisateur. En effet, l'architecture proposée est capable de comprendre la demande de l'utilisateur, trouver le modèle correspondant dans l'ontologie et, selon ce modèle, notre système est capable de fusionner les informations de l'entrée et envoyer le résultat au moteur de fission. Le moteur de fission est en mesure de subdiviser les informations fusionnées en tâche uni-modale et décider quelle modalité de sortie devrait être utilisée pour répondre correctement à la demande.

Nous concluons que l'architecture proposée dans ce mémoire est un système complet qui combine les processus de fusion et de fission pour mieux comprendre une commande émise par un utilisateur de fauteuil roulant muni d'un bras manipulateur, et par conséquent, faciliter le processus d'interaction homme-machine-environnement en prenant en compte les informations contextuelles sémantiques.

Comme perspective, il serait très intéressant d'enrichir la base de connaissance en ajoutant plus de concepts et d'évènements dans l'ontologie et d'introduire d'autres scenarios de commande et d'autres solutions de fission qui vont rendre le système plus ouvert. Il serait aussi très intéressant d'enrichir la base de connaissances par l'ajout d'une diversité de vocabulaire en utilisant d'autres langues comme le Français ou l'Arabe. Cet apport va rendre le système plus riche et cibler un plus grand nombre d'utilisateurs.

On peut aussi considérer comme perspective la prise en compte d'un plus grand nombre de modalités d'entrée, comme par exemple l'utilisation de retour des signaux EMG de l'utilisateur comme modalité d'entrée. Cette prise en compte va permettre aux personnes handicapées, comme les tétraplégiques, d'interagir avec la machine grâce à l'utilisation d'un casque qui va permettre la transmission des signaux EMG pour améliorer la compréhension du comportement de la personne dans la boucle d'interaction.

De plus, il serait intéressant d'introduire l'apprentissage lorsque le système ne reconnait pas un évènement. Cela va permettre au système d'apprendre les nouvelles situations et de prédire de nouvelles actions. Dans ce cadre de type d'interaction où l'homme est le centre de cette interaction l'utilisateur va devenir un acteur principal dans la décision et non pas juste faire partie de l'environnement. La machine pourra ainsi utiliser les retours de l'utilisateur pour affiner ces modèles d'interactions pour s'adapter aux préférences de l'utilisateur. La machine pourra également apprendre des nouvelles situations et de nouveaux comportements.

L'introduction d'une ontologie d'événements en utilisant le formalisme EKRL (Environment Knowledge Representation Language) est aussi à considérer. Ce dernier étant plus riche et plus expressif et pourait représenter des situations très complexes de l'environnement. Une autre perspective qui constitue la continuité de l'expérimentation présentée dans le cadre de cette thèse est d'appliquer notre architecture sur un fauteuil roulant muni d'un bras manipulateur une fois qu'il sera réalisé et finalisé dans notre laboratoire.

Annexe I

Requêtes de sélection de modèle de fusion

Modèle 1 : systemOntology :Words_Person_2(?a) ∧ systemOntology :Possessive(?x) ∧ systemOntology :People(?y) ∧ systemOntology :Personal_Use_Objects(?z) ∧ systemOntology :has_Next_1_1(?a, ?x) ∧ systemOntology :has_Next_1_2(?x, ?y) ∧ systemOntology :has_Next_1_3(?y, ?z) ∧ tbox :isInRangeOf(?m, systemOntology :Compose_1) ∧ tbox :isInDomainOf(?s, systemOntology :hasNext_F1) → sqwrl :selectDistinct(?m, ?a, ?x, ?y, ?z, ?s)

Modèle 2 : systemOntology :Decor_Object(?a) ∧ systemOntology :Position_Pronoun(?z) ∧ systemOntology :Personal_Use_Objects(?y) ∧ systemOntology :Words_Personal_Use_Objects(?x) ∧ systemOntology :hasNext_2_1(?x, ?y) ∧ systemOntology :hasNext_2_2(?y, ?z) ∧ systemOntology :hasNext_2_3(?z, ?a) ∧ tbox :isInRangeOf(?m, systemOntology :Compose_2) ∧ tbox :isInDomainOf(?s, systemOntology :hasNext_F2) → sqwrl :selectDistinct(?m, ?x, ?y, ?z, ?a, ?s)

Modèle 3 : systemOntology :Individuals(?y) ∧ systemOntology :Words_Person_1(?x) ∧ systemOntology :has_Next_3_1(?x, ?y) ∧ tbox :isInRangeOf(?m, systemOntology :Compose_3) ∧ tbox :isInDomainOf(?s, systemOntology :hasNext_F3) → sqwrl :selectDistinct(?m, ?x, ?y, ?s)

Modèle 4 : systemOntology :Places(?z) ∧ systemOntology :Objective(?y) ∧ systemOntology :Words_for_tracking(?x) ∧ systemOntology :hasNext4_1(?x, ?y) ∧ systemOntology :hasNext4_2(?y, ?z) ∧ tbox :isInRangeOf(?m, systemOntology :Com-

pose_4) ∧ tbox :isInDomainOf(?s, systemOntology :hasNext_F4) → sqwrl :selectDistinct(?m, ?x, ?y, ?z, ?s)

Modèle 5 : systemOntology :Words_Personal_Use_Objects(?x) ∧ systemOntology :Personal_Use_Objects(?y) ∧ systemOntology :has_Next_5_1(?x, ?y) ∧ tbox :isInRangeOf(?m, systemOntology :Compose_5) ∧ tbox :isInDomainOf(?s, systemOntology :hasNext_F5) → sqwrl :selectDistinct(?m, ?x, ?y, ?s)

Modèle 6 : systemOntology :Words_Person_2(?x) ∧ systemOntology :Personal_Use_Objects(?y) ∧ systemOntology :Individuals(?z) ∧ systemOntology :has_Next_6_1(?x, ?y) ∧ systemOntology :has_Next_6_2(?y, ?z) ∧ tbox :isInRangeOf(?m, systemOntology :Compose_6) ∧ tbox :isInDomainOf(?s, systemOntology :hasNext_F1) → sqwrl :selectDistinct(?m, ?x, ?y, ?z, ?s)

Modèle 7 : systemOntology :Words_Personal_Use_Objects(?x) ∧ systemOntology :Personal_Use_Objects(?y) ∧ systemOntology :Places(?z) ∧ systemOntology :hasNext7_1(?x, ?y) ∧ systemOntology :hasNext7_2(?y, ?z) ∧ tbox :isInRangeOf(?m, systemOntology :Compose_7) ∧ tbox :isInDomainOf(?s, systemOntology :hasNext_F2) → sqwrl :selectDistinct(?m, ?x, ?y, ?z, ?s)

Modèle 8 : systemOntology :Words_Personal_Use_Objects(?x) ∧ systemOntology :Object_Pronoun(?y) ∧ systemOntology :Pronoun(?z) ∧ systemOntology :has_Next_8_1(?x, ?y) ∧ systemOntology :has_Next_8_2(?y, ?z) ∧ tbox :isInRangeOf(?m, systemOntology :Compose_8) ∧ tbox :isInDomainOf(?s, systemOntology :hasNext_F9) → sqwrl :selectDistinct(?m, ?x, ?y, ?z, ?s)

Modèle 9 : systemOntology :Words_Personal_Use_Objects(?x) ∧ systemOntology :Object_Pronoun(?y) ∧ systemOntology :Position_Pronoun(?z) ∧ systemOntology :Decor_Object(?a) ∧ systemOntology :has_Next_9_1(?x, ?y) ∧ systemOntology :has_Next_9_2(?y, ?z) ∧ systemOntology :has_Next_9_3(?z, ?a) ∧ tbox :isInRangeOf(?m, systemOntology :Compose_9) ∧ tbox :isInDomainOf(?s, systemOntology :hasNext_F9) → sqwrl :selectDistinct(?m, ?x, ?y, ?z, ?a, ?s)

Modèle 10 :systemOntology :Words_Convenient_Objects(?x) ∧ systemOntology :Object_Pronoun(?y) ∧ systemOntology :has_Next_10_1(?x, ?y) ∧ tbox :isInRangeOf(?m, systemOntology :Compose_10) ∧ tbox :isInDomainOf(?s, systemOntology :hasNext_F5) → sqwrl :selectDistinct(?m, ?x, ?y, ?s)

Modèle 11 : systemOntology :Personal_Use_Objects(?z) ∧ systemOntology :Individuals(?y) ∧ systemOntology :Words_Person_2(?x) ∧ systemOntology :hasNext_11_1(?x, ?y) ∧ systemOntology :hasNext_11_2(?y, ?z) ∧ tbox :isInRangeOf(?m, systemOntology :Compose_11) ∧ systemOntology :has_Location(?z, ?Object_Location) ∧ tbox :isInDomainOf(?s, systemOntology :hasNext_F1) → sqwrl :selectDistinct(?m, ?x, ?y, ?z, ?Object_Location, ?s)

Modèle 12 : systemOntology :Words_Lightweight_F(?x) ∧ systemOntology :Lightweight_Furniture(?y) ∧ systemOntology :has_Next_12_1(?x, ?y) ∧ tbox :isInRangeOf(?m, systemOntology :Compose_12) ∧ tbox :isInDomainOf(?s, systemOntology :hasNext_F6) → sqwrl :selectDistinct(?m, ?x, ?y, ?s)

Modèle 13 : systemOntology :Words_Used_For_Movable_P_Object(?x) ∧ systemOntology :Indoors(?y) ∧ systemOntology :Movable_Parts_of_building(?z) ∧ systemOntology :has_Next_13_1(?x, ?y) ∧ systemOntology :has_Next_13_2(?y, ?z) ∧ tbox :isInRangeOf(?m, systemOntology :Compose_13) ∧ tbox :isInDomainOf(?s, systemOntology :hasNext_F6) → sqwrl :selectDistinct(?m, ?x, ?y, ?z, ?s)

Modèle 14 : systemOntology :Words_for_tracking(?x) ∧ systemOntology :Object_Pronoun(?y) ∧ systemOntology :Places(?z) ∧ systemOntology :has_Next_14_1(?x, ?y) ∧ systemOntology :has_Next_14_2(?y, ?z) ∧ tbox :isInRangeOf(?m, systemOntology :Compose_14) ∧ tbox :isInDomainOf(?s, systemOntology :hasNext_F4) → sqwrl :selectDistinct(?m, ?x, ?y, ?z, ?s)

Modèle 15 : systemOntology :Words_Liquid(?x) ∧ systemOntology :Liquid(?y) ∧ systemOntology :has_Next_15_1(?x, ?y) ∧ tbox :isInRangeOf(?m, systemOntology :Compose_15) ∧ systemOntology :Needs_Liquid_Object(?y, ?z) ∧ tbox :isInDomainOf(?s, systemOntology :hasNext_F7) → sqwrl :selectDistinct(?m, ?x, ?z, ?y, ?s)

Modèle 16 : systemOntology :Words_for_tracking(?x) ∧ systemOntology :Objective(?y) ∧ systemOntology :Transport_means(?z) ∧ systemOntology :has_Next_16_1(?x, ?y) ∧ systemOntology :has_Next_16_2(?y, ?z) ∧ tbox :isInRangeOf(?m, systemOntology :Compose_16) ∧ tbox :isInDomainOf(?s, systemOntology :hasNext_F4) → sqwrl :selectDistinct(?m, ?x, ?y, ?z, ?s)

Modèle 17 : systemOntology :Words_Used_For_Movable_P_Object(?x) ∧ systemOntology :Movable_Parts_Object(?y) ∧ systemOntology :has_Next_17_1(?x, ?y) ∧ tbox :isInRangeOf(?m, systemOntology :Compose_17) ∧ tbox :isInDomainOf(?s, systemOntology :hasNext_F6) → sqwrl :selectDistinct(?m, ?x, ?y, ?s)

Modèle 18 : systemOntology :Words_Big_Electrical_O(?x) ∧ systemOntology :Object(?y) ∧ systemOntology :has_Next_18_1(?x, ?y) ∧ tbox :isInRangeOf(?m, systemOntology :Compose_18) ∧ tbox :isInDomainOf(?s, systemOntology :hasNext_F8) → sqwrl :selectDistinct(?m, ?x, ?y, ?s)

Modèle 19 : systemOntology :Words_Person_2(?x) ∧ systemOntology :Individuals(?y) ∧ systemOntology :Liquid(?z) ∧ systemOntology :has_Next_19_1(?x, ?y) ∧ systemOntology :has_Next_19_2(?y, ?z) ∧ tbox :isInRangeOf(?m, systemOntology :Compose_19) ∧ tbox :isInDomainOf(?s, systemOntology :hasNext_F7) → sqwrl :selectDistinct(?m, ?x, ?y, ?z, ?s)

Requêtes de sélection de solution de fission

Solution 1 : systemOntology :Fission_Solution_1(?a) ∧ systemOntology :Fission_Solution_1(?b) ∧ systemOntology :Fission_Solution_1(?c) ∧ systemOntology :Fission_Solution_1(?d) ∧ systemOntology :Fission_Solution_1(?e) ∧ systemOntology :Fission_Solution_1(?f) ∧ systemOntology :has_Next_F(?a, ?b) ∧ systemOntology :has_Next_F(?b, ?c) ∧ systemOntology :has_Next_F(?c, ?d) ∧ systemOntology :has_Next_F(?d, ?e) ∧ systemOntology :has_Next_F(?e, ?f) ∧ tbox :isInDomainOf(?FissionSolution1, systemOntology :has_Next_F) → sqwrl :selectDistinct(?FissionSolution1, ?a, ?b, ?c, ?d, ?e, ?f)

Solution 2 : systemOntology :Fission_Solution_2(?a) ∧ systemOntology :Fission_Solution_2(?b) ∧ systemOntology :Fission_Solution_2(?c) ∧ systemOntology :Fission_Solution_2(?d) ∧ systemOntology :has_Next_F(?a, ?b) ∧ systemOntology :has_Next_F(?b, ?c) ∧ systemOntology :has_Next_F(?c, ?d) ∧ tbox :isInDomainOf(?FissionSolution2, systemOntology :has_Next_F) → sqwrl :selectDistinct(?FissionSolution2, ?a, ?b, ?c, ?d)

Solution 3 : systemOntology :Fission_Solution_3(?a) ∧ systemOntology :Fission_Solution_3(?b) ∧ systemOntology :has_Next_F(?a, ?b) ∧ tbox :isInDomainOf(?FissionSolution3, systemOntology :has_Next_F) → sqwrl :selectDistinct(?FissionSolution3, ?a, ?b)

Solution 4 : systemOntology :Fission_Solution_4(?a) ∧ systemOntology :Fission_Solution_4(?b) ∧ systemOntology :has_Next_F(?a, ?b) ∧ tbox :isInDomainOf(?FissionSolution4, systemOntology :has_Next_F) → sqwrl :selectDistinct(?FissionSolution4, ?a, ?b)

Solution 5 : systemOntology :Fission_Solution_5(?a) ∧ systemOntology :Fission_Solution_5(?b) ∧ systemOntology :has_Next_F(?a, ?b) ∧ tbox :isInDomainOf(?FissionSolution5, systemOntology :has_Next_F) → sqwrl :selectDistinct(?FissionSolution5, ?a, ?b)

Solution 6 : systemOntology :Fission_Solution_6(?a) ∧ systemOntology :Fission_Solution_6(?b) ∧ systemOntology :has_Next_F(?a, ?b) ∧ tbox :isInDomainOf(?FissionSolution6, systemOntology :has_Next_F) → sqwrl :selectDistinct(?FissionSolution6, ?a, ?b)

Solution 7 : systemOntology :Fission_Solution_7(?a) ∧ systemOntology :Fission_Solution_7(?b) ∧ systemOntology :Fission_Solution_7(?c) ∧ systemOntology :Fission_Solution_7(?d) ∧ systemOntology :Fission_Solution_7(?e) ∧ systemOntology :has_Next_F(?a, ?b) ∧ systemOntology :has_Next_F(?b, ?c) ∧ systemOntology :has_Next_F(?c, ?d) ∧ systemOntology :has_Next_F(?d, ?e) ∧ tbox :isInDomainOf(?FissionSolution7, systemOntology :has_Next_F) → sqwrl :selectDistinct(?FissionSolution7, ?a, ?b, ?c, ?d, ?e)

Solution 8 : systemOntology :Fission_Solution_8(?a) ∧ systemOntology :Fission_Solution_8(?b) ∧ systemOntology :has_Next_F(?a, ?b) ∧ tbox :isInDomainOf(?FissionSolution8, systemOntology :has_Next_F) → sqwrl :selectDistinct(?FissionSolution8, ?a, ?b)

Solution 9 : systemOntology :Fission_Solution_9(?a) ∧ systemOntology :Fission_Solution_9(?b) ∧ systemOntology :Fission_Solution_9(?c) ∧ systemOntology :Fission_Solution_9(?d) ∧ systemOntology :has_Next_F(?a, ?b) ∧ systemOntology :has_Next_F(?b, ?c) ∧ systemOntology :has_Next_F(?c, ?d) ∧ tbox :isInDomainOf(?FissionSolution9, systemOntology :has_Next_F) → sqwrl :selectDistinct(?FissionSolution9, ?a, ?b, ?c, ?d)

Requêtes de la recherche de modèle de fusion pour les évènements

Evènement 1 : systemOntology :Event(?z) ∧ systemOntology :Event(?y) ∧ systemOntology :Event(?x) ∧ systemOntology :Event(?a) ∧ systemOntology :has_Next_1_1(?a, ?x) ∧ systemOntology :has_Next_1_2(?x, ?y) ∧ systemOntology :has_Next_1_3(?y, ?z) ∧ systemOntology :has_Next_Event_1(?a, ?x) ∧ systemOntology :has_Next_Event_1(?x, ?y) ∧ systemOntology :has_Next_Event_1(?y, ?z) ∧ tbox :isInDomainOf(?s, systemOntology :hasNext_F1) ∧ tbox :isInRangeOf(?m, systemOntology :Compose_1) → sqwrl :selectDistinct(?m, ?a, ?x, ?y, ?z, ?s)

Evènement 2 : systemOntology :Event(?a) ∧ systemOntology :Event(?z) ∧ systemOntology :Event(?y) ∧ systemOntology :Event(?x) ∧ systemOntology :has-Next_2_1(?x, ?y) ∧ systemOntology :hasNext_2_2(?y, ?z) ∧ systemOntology :has-Next_2_3(?z, ?a) ∧ systemOntology :has_Next_Event_2(?x, ?y) ∧ systemOntology :has_Next_Event_2(?y, ?z) ∧ systemOntology :has_Next_Event_2(?z, ?a) ∧ tbox :isInRangeOf(?m, systemOntology :Compose_2) ∧ tbox :isInDomainOf(?s, systemOntology :hasNext_F2) → sqwrl :selectDistinct(?m, ?x, ?y, ?z, ?a, ?s)

Evènement 3 : systemOntology :Event(?y) ∧ systemOntology :Event(?x) ∧ systemOntology :has_Next_3_1(?x, ?y) ∧ systemOntology :has_Next_Event_3(?x, ?y) ∧ tbox :isInRangeOf(?m, systemOntology :Compose_3) ∧ tbox :isInDomainOf(?s, systemOntology :hasNext_F3) → sqwrl :selectDistinct(?m, ?x, ?y, ?s)

Evènement 4 : systemOntology :Event(?z) ∧ systemOntology :Event(?y) ∧ systemOntology :Event(?x) ∧ systemOntology :hasNext4_1(?x, ?y) ∧ systemOntology :hasNext4_2(?y, ?z) ∧ systemOntology :has_Next_Event_4(?x, ?y) ∧ systemOntology :has_Next_Event_4(?y, ?z) ∧ systemOntology :hasGesturePosition(?z, ?Gesture-Detected) ∧ tbox :isInRangeOf(?m, systemOntology :Compose_4) ∧ tbox :isInDomainOf(?s, systemOntology :hasNext_F4) → sqwrl :selectDistinct(?m, ?x, ?y, ?z, ?Gesture-Detected, ?s)

Evènement 5 : systemOntology :Event(?x) ∧ systemOntology :Event(?y) ∧ systemOntology :has_Next_5_1(?x, ?y) ∧ systemOntology :has_Next_Event_5(?x, ?y) ∧ tbox :isInRangeOf(?m, systemOntology :Compose_5) ∧ tbox :isInDomainOf(?s, systemOntology :hasNext_F5) ∧ systemOntology :has_Location(?y, ?Location) → sqwrl :selectDistinct(?m, ?x, ?y, ?Location, ?s)

Evènement 6 : systemOntology :Event(?x) ∧ systemOntology :Event(?y) ∧ systemOntology :Event(?z) ∧ systemOntology :has_Next_6_1(?x, ?y) ∧ systemOntology :has_Next_6_2(?y, ?z) ∧ systemOntology :has_Next_Event_6(?x, ?y) ∧ systemOntology :has_Next_Event_6(?y, ?z) ∧ tbox :isInRangeOf(?m, systemOntology :Compose_6) ∧ tbox :isInDomainOf(?s, systemOntology :hasNext_F1) → sqwrl :selectDistinct(?m, ?x, ?y, ?z, ?s)

Evènement 7 : systemOntology :Event(?x) ∧ systemOntology :Event(?y) ∧ systemOntology :Event(?z) ∧ systemOntology :hasNext7_1(?x, ?y) ∧ systemOntology :hasNext7_2(?y, ?z) ∧ systemOntology :has_Next_Event_7(?x, ?y) ∧ systemOntology :has_Next_Event_7(?y, ?z) ∧ tbox :isInRangeOf(?m, systemOntology :Compose_7) ∧ tbox :isInDomainOf(?s, systemOntology :hasNext_F2) → sqwrl :selectDistinct(?m, ?x, ?y, ?z, ?s)

Evènement 8 : systemOntology :Event(?x) ∧ systemOntology :Event(?y) ∧ systemOntology :Event(?z) ∧ systemOntology :has_Next_8_1(?x, ?y) ∧ systemOntology :has_Next_8_2(?y, ?z) ∧ systemOntology :has_Next_Event_8(?x, ?y) ∧ systemOntology :has_Next_Event_8(?y, ?z) ∧ tbox :isInRangeOf(?m, systemOntology :Compose_8) ∧ tbox :isInDomainOf(?s, systemOntology :hasNext_F9) → sqwrl :selectDistinct(?m, ?x, ?y, ?z, ?s)

Evènement 9 : systemOntology :Event(?x) ∧ systemOntology :Event(?y) ∧ systemOntology :Event(?z) ∧ systemOntology :Event(?a) ∧ systemOntology :has_Next_9_1(?x, ?y) ∧ systemOntology :has_Next_9_2(?y, ?z) ∧ systemOntology :has_Next_9_3(?z, ?a) ∧ systemOntology :has_Next_Event_9(?x, ?y) ∧ systemOntology :has_Next_Event_9(?y, ?z) ∧ systemOntology :has_Next_Event_9(?z, ?a) ∧

tbox :isInRangeOf(?m, systemOntology :Compose_9) ∧ tbox :isInDomainOf(?s, systemOntology :hasNext_F9) → sqwrl :selectDistinct(?m, ?x, ?y, ?z, ?a, ?s)

Evènement 10 : systemOntology :Event(?x) ∧ systemOntology :Event(?y) ∧ systemOntology :has_Next_10_1(?x, ?y) ∧ systemOntology :has_Next_Event_10(?x, ?y) ∧ tbox :isInRangeOf(?m, systemOntology :Compose_10) ∧ tbox :isInDomainOf(?s, systemOntology :hasNext_F5) → sqwrl :selectDistinct(?m, ?x, ?y, ?s)

Evènement 11 : systemOntology :Event(?z) ∧ systemOntology :Event(?y) ∧ systemOntology :Event(?x) ∧ systemOntology :hasNext_11_1(?x, ?y) ∧ systemOntology :hasNext_11_2(?y, ?z) ∧ systemOntology :has_Location(?z, ?Object_Location) ∧ systemOntology :has_Next_Event_11(?x, ?y) ∧ systemOntology :has_Next_Event_11(?y, ?z) ∧ tbox :isInDomainOf(?s, systemOntology :hasNext_F1) ∧ tbox :isInRangeOf(?m, systemOntology :Compose_11) → sqwrl :selectDistinct(?m, ?x, ?y, ?z, ?Object_Location, ?s)

Evènement 12 : systemOntology :Event(?x) ∧ systemOntology :Event(?y) ∧ systemOntology :has_Next_12_1(?x, ?y) ∧ systemOntology :has_Next_Event_12(?x, ?y) ∧ tbox :isInRangeOf(?m, systemOntology :Compose_12) ∧ tbox :isInDomainOf(?s, systemOntology :hasNext_F6) → sqwrl :selectDistinct(?m, ?x, ?y, ?s)

Evènement 13 : systemOntology :Event(?x) ∧ systemOntology :Event(?y) ∧ systemOntology :Event(?z) ∧ systemOntology :has_Next_13_1(?x, ?y) ∧ systemOntology :has_Next_13_2(?y, ?z) ∧ systemOntology :has_Next_Event_13(?x, ?y) ∧ systemOntology :has_Next_Event_13(?y, ?z) ∧ tbox :isInRangeOf(?m, systemOntology :Compose_13) ∧ tbox :isInDomainOf(?s, systemOntology :hasNext_F6) → sqwrl :selectDistinct(?m, ?x, ?y, ?z, ?s)

Evènement 14 : systemOntology :Event(?x) ∧ systemOntology :Event(?y) ∧ systemOntology :has_Next_14(?x, ?y) ∧ systemOntology :has_Next_Event_14(?x, ?y) ∧ tbox :isInRangeOf(?m, systemOntology :Compose_14) ∧ tbox :isInDomainOf(?s, systemOntology :hasNext_F4) → sqwrl :selectDistinct(?m, ?x, ?y, ?s)

Evènement 15 : systemOntology :Event(?x) ∧ systemOntology :Event(?y) ∧ systemOntology :has_Next_15_1(?x, ?y) ∧ systemOntology :has_Next_Event_15(?x, ?y) ∧ tbox :isInRangeOf(?m, systemOntology :Compose_15) ∧ systemOntology :Needs_Liquid_Object(?y, ?BroughtIn) ∧ systemOntology :has_Location(?y, ?Location) ∧ tbox :isInDomainOf(?s, systemOntology :hasNext_F7) → sqwrl :selectDistinct(?m, ?x, ?BroughtIn, ?y, ?Location, ?s

Evènement 16 : systemOntology :Event(?x) ∧ systemOntology :Event(?y) ∧ systemOntology :Event(?z) ∧ systemOntology :has_Next_16_1(?x, ?y) ∧ systemOntology :has_Next_16_2(?y, ?z) ∧ systemOntology :has_Next_Event_16(?x, ?y) ∧ systemOntology :has_Next_Event_16(?y, ?z) ∧ tbox :isInRangeOf(?m, systemOntology :Compose_16) ∧ tbox :isInDomainOf(?s, systemOntology :hasNext_F4) → sqwrl :selectDistinct(?m, ?x, ?y, ?z, ?s)

Evènement 17 : systemOntology :Event(?x) ∧ systemOntology :Event(?y) ∧ systemOntology :has_Next_17_1(?x, ?y) ∧ systemOntology :has_Next_Event_17(?x, ?y) ∧ tbox :isInRangeOf(?m, systemOntology :Compose_17) ∧ tbox :isInDomainOf(?s, systemOntology :hasNext_F6) → sqwrl :selectDistinct(?m, ?x, ?y, ?s)

Evènement 18 : systemOntology :Event(?x) ∧ systemOntology :Event(?y) ∧ systemOntology :has_Next_18_1(?x, ?y) ∧ systemOntology :has_Next_Event_18(?x, ?y) ∧ tbox :isInRangeOf(?m, systemOntology :Compose_18) ∧ tbox :isInDomainOf(?s, systemOntology :hasNext_F8) → sqwrl :selectDistinct(?m, ?x, ?y, ?s)

Evènement 19 : systemOntology :Event(?x) ∧ systemOntology :Event(?y) ∧ systemOntology :Event(?z) ∧ systemOntology :has_Next_19_1(?x, ?y) ∧ systemOntology :has_Next_19_2(?y, ?z) ∧ systemOntology :has_Next_Event_19(?x, ?y) ∧ systemOntology :has_Next_Event_19(?y, ?z) ∧ systemOntology :Needs_Liquid_Object(?z, ?a) ∧ systemOntology :IsDetectedBy(?y, ?DetectedBy) ∧ tbox :isInRangeOf(?m, systemOntology :Compose_19) ∧ tbox :isInDomainOf(?s, systemOntology :hasNext_F7) → sqwrl :selectDistinct(?m, ?x, ?y, ?DetectedBy, ?a, ?z, ?s)

Tableau 1: Symboles utilisés sur Protégé

Symboles utilisés	**Significations**
	Classes
	Instances
	Propriétés d'objets (différentes couleurs pour différentes propriétés)
	Relie les classes aux sous classes et les classes aux instances

Annexe II

Déclaration de variables dans CPN Tools du Chapitre 5

colset UNIT = unit;

colset BOOL = bool;

colset INT = int;

colset INTINF = intinf;

colset TIME = time;

colset REAL = real;

colset STRING = string timed;

colset IISSS=product INT*INT*STRING*STRING*STRING;

colset IISI = product INT*INT*STRING*INT;

colset IISB= product INT*INT*STRING*BOOL;

colset IISSI = product INT*INT*STRING*STRING*INT;

colset IISSSI = product INT*INT*STRING*STRING*STRING*INT;

colset INTxINT= product INT*INT;

colset ISSS=product INT*STRING*STRING*STRING;

colset IISSSS=product INT*INT*STRING*STRING*STRING*STRING;

colset Temps=with temps timed;

colset INTxSTRINGxSTRING=product INT*STRING*STRING*INT;

colset INTxDATA = product INT*STRING*INT timed;

colset IS=product INT*STRING timed;

var Modality, model, Event, Input, H, HF, E, Ev, wco1, wft, pl, mdl1, mdl2, mdl3, mdl4, mdl5, co, lq, pp, l, ps, wco2, exp, st, exp1, exp2, exp3, rej, cup, sol1, sol4, sol7,

```
sol, outModality, why, out : STRING;
    var n, m, NL, NLF, LL, LLF, g, t, p, e, k, et, nt, ent, nn, c, en, Ren, i1, t1, t2, t3, tT,
tc, enx, tm1, tm2, n1, n2, n3, en1, en2, en3, enf, enn, m1, m4, m7, mn, c, ef, nf : INT;
    var ,AlrF :BOOL;
    var i,x : REAL;
    val NLmax = 50;
    val LLmin=40;
    val nMax=20;
    val TCmax =15;
    val TMmax=10;
    colset Noise = INT;
    colset Handicap = STRING;
    colset Light= INT;
    colset Alarm = BOOL;
```

Journal

- ***"Fusion and Fission Engine for an Assistant Robot Using an Ontology Knowledge Base"****, Nadia Touileb Djaid, Sébastien Dourlens, Nadia Saadia and Amar Ramdane-Cherif. Journal of Ambient Intelligence and Smart Environments. (Online 05 Novembre 2017).*

Conférence

- ***"Fission Engine for an Ambient Assistance Robot Based on the Ontology Concept "****, Nadia Touileb Djaid, Nadia Saadia and Amar Ramdane-Cherif, The 6th International Conference on Ambient Systems, Networks and Technologies. (ANT 2016), Madrid. Spain.*
- ***"Multimodal Fusion engine for an intelligent assistance robot using Ontology "****, Nadia Touileb Djaid, Nadia Saadia and Amar Ramdane-Cherif, The 6th International Conference on Ambient Systems, Networks and Technologies. (ANT 2015) London. UK*
- ***"Ontology Based Fusion Engine for Interaction with an Intelligent Assistance Robot "****, Nadia Touileb Djaid, Nadia Saadia and Amar Ramdane-Cherif, International Conference on Swarm Intelligence (ICSI 2015), Beijing. China.*
- ***"Multimodal Structure for the management of Energies in a residential Home",*** *Hadia Bouguessa, Nadia Saadia, Nadia Touileb and Amina Makhlouf, The Sixth International Energy, Energy and Environment Symposium (IEEES6), (2013), RTEU Rize Turkey.*
- ***"Multimodal architecture to strengthen the interaction of the robot in ambient intelligence environments"****, Nadia Touileb Djaid, Nadia Saadia and Amar Ramdane-Cherif, 14th ACM International Conference on Ubiquitous computing, UbiRobs (2012), Pittsburgh, Pennsylvania. USA.*

Bibliographie

[1] **Cook, D. J., Augusto, J. C., Jakkula, V. R.** :*Ambient intelligence : Technologies, applications, and opportunities*. Pervasive and Mobile Computing, Volume 5, Issue 4, Pages 277–298. (2009).

[2] **Bick M. and Kummer T. F.** : *Ambient Intelligence and Ubiquitous Computing*. Handbook on Information Technologies for Education and Training, Springer Berlin Heidelberg. Pages 79-100. (2008). Doi : 10.1007/978-3-540-74155-8_5

[3] **J. C. Augusto and P. McCullagh** : *Ambient Intelligence : Concepts and Applications*. Computer Science and Information Systems, Volume 4, Issue 1. Pages 1-28. (2007)

[4] **A. A. H. Mousa** : *Ubiquitous Pervasive Computing*. International Journal of Innovative Research and Development, Vol 2, Issue 10, Pages 276-282.(2013)

[5] **S. Sathyakeerthy, M. Di Rocco, F. Pecora, A. Saffiotti** : *Scaling up ubiquitous robotic systems from home to town (and beyond)*. Proceeding UbiComp 13 Adjunct, Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication, Pages 107-110. (2013). Doi : 10.1145/2494091.2494121

[6] **N. Dipsis, K. Stathis** :*Ubiquitous Agents for Ambient Ecologies*. Pervasive and Mobile Computing Volume 8, Issue 4, Pages 562–574. (2012)

[7] **T. Choi, Y. Chon, H. Cha** : *Energy-efficient WiFi scanning for localization*. Pervasive and Mobile Computing Volume 37, Pages 124–138. (2017)

[8] **D. Yuan, S. S. Kanher, M. Hollick** : *Instrumenting Wireless Sensor Networks — A survey on the metrics that matter*. Pervasive and Mobile Computing. Volume 37, Pages 45–62. (2017)

[9] **Chibani, A. et al** : *Ubiquitous robotics : Recent challenges and future trends*. Journal Robotics and Autonomous Systems. Volume 61, Issue 11, Pages 1162-1172. (2013)

[10] **L. Merino, A. Gilbert, J. Capitán, R. Bowden, J. Illingworth, A. Ollero** : *Data fusion in ubiquitous networked robot systems for urban services*. Annals of telecommunications, Volume 67, Issue 7, Pages 355–375. (2012). Doi :10.1007/s12243-012-0311-1

[11] **J. H. Kim** : *Ubiquitous Robot*. Advances in Soft Computing, Pages 451–459. (2005)

[12] **G. Amato et al.** :*Robotic UBIquitous COgnitive Network*. Ambient Intelligence - Software and Applications. Advances in Intelligent and Soft Computing, Volume 153. Springer, Berlin, Pages 191-195. (2012). Doi : 10.1007/978-3-642-28783-1_23

[13] **M. Weiser, P. Alto** : *The computer for the 21st century*. ACM SIGMOBILE Mobile Computing and Communications Review - Special issue dedicated to Mark Weiser, Volume 3, Issue 3, Pages 3-11. (1999). Doi : 10.1145/329124.329126

[14] **Hong, J-y. ; , Suh, E-h. ; Kim, S-j.** : *Context-aware systems : A literature review and classification*. Expert Systems with Applications, Volume 36, Issue 4. Pages 8509–8522. (2009)

[15] **N. F. Noy and D. L. McGuinness** : *Ontology Development 101 : A Guide to Creating Your First Ontology*. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880. (2001).

[16] **J. Hallam, H. Bruyninckx** : *An Ontology of Robotics Science*. European Robotics Symposium 2006. Springer Tracts in Advanced Robotics, Volume 22. Springer, Pages 1-14. (2006). Doi : 10.1007/11681120_1

[17] **E.G. Tsardoulias et al.** : *Merging Robotics and AAL Ontologies : The RAPP Methodology*. Progress in Automation, Robotics and Measuring Techniques. Advances in Intelligent Systems and Computing, Volume 351. Springer, Pages 285-297. (2015). Doi : 10.1007/978-3-319-15847-1_28

[18] **M. Bezold** : *An Ontology-Based Adaptation Framework for Multimodal Interactive Systems*. Engineering Interactive Systems. Lecture Notes in Computer Science, Volume 5247. Springer, Pages 205-212. (2008). Doi : 10.1007/978-3-540-85992-5_18

[19] **André, E.; Martin, J. C.** : *Multimodal systems*. Mitkov, R., ed. : The Oxford Handbook of Computational Linguistics 2nd edition, Oxford University Press, Pages 1–15. (2014)

[20] **Gnjatović, M. et al.** : *Adaptive multimodal interaction with industrial robot*. IEEE 10th Jubilee International Symposium on Intelligent Systems and Informatics (SISY). Pages 329-333. (2012). Doi : 10.1109/SISY.2012.6339538

[21] **Uday, V. and Vijaychand, K.** : *Multimodal interaction for service robot control*. INTERNATIONAL JOURNAL OF REVIEWS ON RECENT ELECTRONICS AND COMPUTER SCIENCE, Volume 1, Issue 6. Pages 1664-1669 . (2013).

[22] **Miyoshi, K.; Konomura, R.; Hori, K.** : *Entertainment Multi-rotor Robot that realises Direct and Multimodal Interaction*. Proceedings of the 28th International BCS Human Computer Interaction Conference. Pages 218-221. (2014)

[23] **Salah, A. A.** :*Natural Multimodal Interaction with a Social Robot : What are the Premises ?*. Proceedings of the Workshop on Roadmapping the Future of Multimodal Interaction Research including Business Opportunities and Challenges. Pages 43-45. (2014). Doi : 10.1145/2666253.2666261

[24] **Rousseau C., Bellik Y., Vernier F.** : *WWHT : Un modèle conceptuel pour la presentation multimodale d'information*. In Proceedings of the 17th international conference on Francophone sur l'Interaction Homme-Machine. Doi : 10.1145/1148550.1148558.(2005).

[25] **T. Murata** : *Petri Nets : Properties, Analisys and Applications*. Proceedings of the IEEE Volume 77, Issue 4. Pages 541-580. (1989).

[26] **K. Jensen, L. M. Kristensen, L. Wells** : *Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems*. International Journal on Software Tools for Technology Transfer, 9, Issue 3, Pages 213-254.(2007).

[27] http ://cpntools.org/ (Accédé le 20-09-2017)

[28] **M. Westergaard** : *CPN Tools 4 : Multi-formalism and Extensibility*. International Conference on Applications and Theory of Petri Nets and Concurrency. Pages 400-409. (2013). Doi : 10.1007/978-3-642-38697-8_22

[29] **L. Wells** : *Performance Analysis using CPN Tools*. 06 Proceedings of the 1st international conference on Performance evaluation methodolgies and tools Article No. 59. Italy. (2006).

[30] **Geihs, K. ; Wagner, M.** : *Context-Awareness for Self-adaptive Applications in Ubiquitous Computing Environments*. International Conference on Context-Aware Systems and Applications. Pages 108-120. (2012). Doi : 10.1007/978-3-642-36642-0_11

[31] **Sonntag, D. et al** :*SmartWeb Handheld — Multimodal Interaction with Ontological Knowledge Bases and Semantic Web Services*. Artifical Intelligence for Human Computing, Lecture Notes in Computer Science, Volume 4451. Springer, Berlin, Heidelberg. Pages 272-295. (2007). Doi : 10.1007/978-3-540-72348-6_14

[32] **Lahat, D. ; Adalı, T. ; Jutten, C.** : *Multimodal Data Fusion : An Overview of Methods, Challenges and Prospects*. Proceedings of the IEEE, Institute of Electrical

and Electronics Engineers, Multimodal Data Fusion, Volume 103, Issue 9. Pages 1449-1477. (2015)

[33] **Atrey, P.K. ; et al.** : *Multimodal fusion for multimedia analysis : a survey*. Multimedia Systems, Volume 16, Issue 6, Pages 345–379. (2010). Doi : 10.1007/s00530-010-0182-0

[34] **Schnelle-Walka, D. ; Duarte, C. ; Radomski, S.** :*Multimodal Fusion and Fission within the W3C MMI Architectural Pattern*. Multimodal Interaction with W3C Standards : Toward Natural User Interfaces to Everything. Pages : 393-415. (2017). Doi : 10.1007/978-3-319-42816-1_19

[35] **R. Reynaud** : *La fusion de données du capteur au raisonnement*. Traitement du Signal, Volume 11. Issue 6. (1994).

[36] **P. K. Atrey et al.** : *Multimodal fusion for multimedia analysis : a survey* . Multimedia Systems (Springer 2010).

[37] **R.A. Bolt** : *"Put-that-there". Voice and gesture at the graphics interface* . ACM SIGGRAPH Computer Graphics, Volume 14, Pages 262-270. (1980).

[38] **P. Prodanov, A. Drygajlo** : *Bayesian networks based multi-modality fusion for error handling in human-robot dialogues under noisy conditions* . Speech communication, Volume 45, Pages 231-248. (2005).

[39] **D. Novak, R. Riener** : *A survey of sensor fusion methods in wearable robotics* . Robotics and autonomous systems, Volume 73, Pages 155-170. (2015).

[40] **C. Axenie, J. Conradt** : *Cortically inspired sensor fusion network for mobile robot ergomotion estimation*. Robotics and Autonomous Systems, Volume 71, Pages 69-82. (2015).

[41] **T. Germa, F. Lerasle, N. Ouadah, V. Cadenat** : *Vision and RFID data fusion for tracking people in crowds by a mobile robot* . Computer Vision and Image understanding, Volume 114, Pages 641-651. (2010).

[42] **B. Srinath Reddy, O.A. Basir** : *Concept-based evidential reasoning for multimodal fusion in human-computer interaction* . Applied Soft Computing, Volume 10, Pages 567–577. (2010).

[43] **E. Berghöfer, D. Schulze, C. Rauch, M. Tscherepanow, T. Köhler, S. Wachsmuth, S.** : *ART-based fusion of multi-modal perception for robots*. Neuro computing, Volume 107, Pages 11-22. (2013).

[44] **T. Perperis, T. Giannakopoulos, A. Makris, D. I. Kosmopoulos, S. Tsekeridou, S. J. Perantonis, S. Theodoridis** : *Multimodal and ontology-based fusion approaches of audio and visual processing for violence detection in movies* . Expert systems with applications, Volume 38, Pages 14102–14116. (2011).

[45] **A. Wehbi, A. Zaguia, A. Ramdane-Cherif, C. Tadj** : *Multimodal Fusion for Interaction Systems*. Journal of Emerging Trends in Computing and Information Sciences, Volume 4, Pages 445-458. (2013).

[46] **Hassanzadeh O.** :*Introduction to Semantic Web Technologies and Linked Data*. University of Toronto. (2011)

[47] **Szeredi, P. ; Lukcsy, G. ; Benk, T.** : *The Semantic Web Explained : The Technology and Mathematics behind Web 3.0*. Cambridge University Press New York. (2014)

[48] **Grifoni, P.** :*Multimodal Fission*. Multimodal Human Computer Interaction and Pervasive Services. Pages 103-120. (2009)

[49] **Mohand Oussaïd, L. ; Aït Ameur, Y. ; Ahmed Nacer, M.** : *A generic formal model for fission of modalities in output multi-modal interactive systems*. VECoS'09 Proceedings of the Third international conference on Verification and Evaluation of Computer and Communication Systems. Pages 125-136. (2009).

[50] **O. Adjali, M. Dulva Hina, S. Dourlens, A. Ramdane-Cherif** : *Multimodal Fusion, Fission and Virtual Reality Simulation for an Ambient Robotic Intelligence*. Procedia Computer Science, Volume 52, Pages 218-225. (2015).

[51] **O. Adjali, M. D. Hina, S. Dourlens, A. Ramdane-Cherif** : *Techniques for Multimodal Fusion and Fission for an Intelligent Robotic Application*. Adv Robot Autom Volume 4, Issue 140. Doi : 10.4172/2168-9695.1000140. (2015).

[52] **D. Costa, C. Duarte** : *Adapting Multimodal Fission to User's Abilities*. Universal Access in Human-Computer Interaction, Volume 6765, Pages 347-356.(2011).

[53] **Rousseau, C.; Bellik, Y.; Vernier, F.; Bazalgette, D.** : *A framework for the intelligent multimodal presentation of information*. Signal Processing - Special section : Multimodal human-computer interfaces, Volume 86, Issue 12. Pages 3696-3713. (2006). Doi : 10.1016/j.sigpro.2006.02.041

[54] **Rousseau, C.; Bellik, Y.; Vernier, F.** : *Un modèle conceptuel pour une présentation multimodale et contextuelle de l'information*. Revue d'Interaction Homme-Machine. Volume 7, Issue 2. Pages 1-28. (2006)

[55] **D. Schnelle-Walk, S. Radeck-Arneth, J. Striebinger** : *Multimodal Dialog management in a Smart Home Context with SCXML*. In Proceedings of 2nd Workshop on Engineering Interactive Systems with SCXML, Pages 23–25. Doi : 10.13140/RG.2.1.3790.5120.(2015).

[56] **A. Zaguia, A. Wahbi, M. Miraoui, C. Tadj, A. Ramdane Cherif** : *Modeling Rules Fission and Modality Selection Using Ontology*. Journal of Software Engineering and Applications, Volume 6, Issue 7, Pages 354-371. Doi :10.4236/jsea.2013.67045.(2013).

[57] **S. L. Oviatt, P. R. Cohen** : *Multimodal Interfaces That Process What Comes Naturally*. In Communications of the ACM, Volume 43, Issue 3, Pages 45-53. Doi : 10.1145/330534.330538.(2000).

[58] **F. Honold, F. Schüssel, M. Weber, F. Nothdurft, G. Bertrand, W. Minker** : *Context Models for Adaptive Dialogs and Multimodal Interaction*. The 9th International Conference on Intelligent Environments (IE), Pages 57-64. (2013).

[59] **D. Perroud, L. Angelini, O. Abou Khaled, E. Mugellini** : *Context-Based Generation of Multimodal Feedbacks for Natural Interaction in Smart Environments*. In AMBIENT, The Second International Conference on Ambient Computing, Applications, Services and Technologies, Pages 19-25. (2012).

[60] **G. Pruvost, T. Heinroth, Y. Bellik, W. Minker** : *User Interaction Adaptation within Ambient Environments*. Next Generation Intelligent Environments, Pages 153-194. (2011).

[61] **J. F. Ladry et al.** : *Formal Description Techniques to Support the Design,Construction and Evaluation of Fusion Engines for SURE (Safe, Usable, Reliable and Evolvable) Multimodal Interfaces*. Proceedings of the 2009 international conference on Multimodal interfaces, Pages 185-192. (2009). Doi :10.1145/1647314.1647347

[62] **A.A. Mekonnen et al.** : *Coopération entre un robot mobile et des caméras d'ambiance pour le suivi multi-personnes*. RFIA 2012 (Reconnaissance des Formes et Intelligence Artificielle). (2012).

[63] **J. Chai** : *Semantics-based Representation for Multimodal Interpretation in Conversational Systems*. COLING 02 Proceedings of the 19th international conference on Computational linguistics, Volume 1. (2002).

[64] **W. Allègre et al.** : *Conception d'un système domotique pour l'assistance aux personnes dépendantes*. Journal des Sciences et Technologies pour le Handicap, Volume 4, Issue 2, Pages 161-187. (2010). doi :10.3166/sth.161-187

[65] **L. Sabri et al.** : *Narrative reasoning for cognitive ubiquitous robots*. Knowledge Representation for Autonomous Robots, Workshop at IEEE/RSJ International Conference on Intelligent Robots and Systems. (2010).

[66] **A. Saffiotti et al.** : *The PEIS-Ecology Project : a progress report*. ICRA-07 Workshop on Network Robot Systems IEEE Int Conf on Robotics and Automation. Rome, Italy. (2007).

[67] **R. B. Rusua et al.** : *Robots in the kitchen : Exploiting ubiquitous sensing and actuation*. Robotics and Autonomous Systems Journal (Special Issue on Network Robot Systems). (2008).

[68] M.N Nicolescu, M.J Matari'c. Matari' c. A Hierarchical Architecture for Behaviour-Based Robots. The First International Joint Conference on Autonomous Agents and MultiAgent Systems Bologna. Pages 227-233. (2002)

[69] http ://protege.stanford.edu/

[70] https ://github.com/protegeproject/swrlapi/wiki/SQWRL

[71] **M. O'Connor, A. Das** : *SQWRL : a query language for OWL.* OWLED'09 Proceedings of the 6th International Conference on OWL : Experiences and Directions, Volume 529, Pages 208-215. (2009).

[72] **S**WRL : http ://www.w3.org/Submission/SWRL/ (2004).

[73] **P. Hitzler, M. Krötzsch, S. Rudolph** : *Knowledge Representation and Reasoning for the Semantic Web-OWL 2 Rules.* A tutorial at KI 2009, Paderborn, Germany. (2009).

[74] **M. O'Connor, R. Shankar, S. Tu, C. Nyulas, A. Das, M. Musen** : *Efficiently Querying Relational Databases Using OWL and SWRL.* International Conference on Web Reasoning and Rule Systems, Pages 361-363. (2007). Doi : 10.1007/978-3-540-72982-2_31

[75] **The Arlyn ChairBot**. Retrieved February 07, 2016, from http ://www.osbornej.com/ChairBotHome.html

[76] **S.L. Oviatt** : *Multimodal interfaces. The Human-Computer Interaction Handbook.*Fundamentals, Evolving Technologies and Emerging Applications, 2nd edition, CRC Press, chap. 14, Pages 286—304. (2008).

[77] **S.L Oviatt** : *Advances in Robust Multimodal Interface Design*. IEEE Computer Graphics and Applications, Volume 23, Issue 5, Pages 62-68. (2003).

[78] **D. Gouaillier,V. Hugel,P. Blazevic,C. Kilner,J.O. Monceaux,P. Lafourcade,. . . ,B. Maisonnier** : *Mechatronic design of NAO humanoid*. ICRA'09. IEEE international conference on robotics and automation, Pages 769-774. (2009). Doi : 10.1109/ROBOT.2009.5152516.

[79] **D. Gouaillier,V. Hugel,P. Blazevic,C. Kilner,J.O. Monceaux,P. Lafourcade,. . . ,B. Maisonnier** : *Mechatronic design of NAO humanoid*. ICRA'09. IEEE international conference on robotics and automation, Pages 769-774. (2009). Doi : 10.1109/ROBOT.2009.5152516.

[80] **M. Fakoor, A. Kosari, M. Jafarzadeh** : *Humanoid robot path planning with fuzzy Markov decision processes*. Journal of Applied Research and Technology. Volume 14, Issue 5, Pages 300-310. (2016). Doi : 10.1016/j.jart.2016.06.006

[81] http ://doc.aldebaran.com/2-1/home_nao.html

[82] http ://doc.aldebaran.com/2-1/index_dev_guide.html