ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

THÈSE PRÉSENTÉE À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DE VERSAILLES SAINT-QUENTIN-EN-YVELINES
(COTUTELLE)

COMME EXIGENCE PARTIELLE
À L'OBTENTION DU
DOCTORAT EN GÉNIE
Ph. D.
COTUTELLE FRANCE-QUÉBEC

PAR
Ahmad WEHBI

FUSION MULTIMODALE POUR LES SYSTÈMES D'INTERACTION

MONTRÉAL, LE 10 JUILLET 2013

PRÉSENTATION DU JURY

CETTE THÈSE A ÉTÉ ÉVALUÉE

PAR UN JURY COMPOSÉ DE :

M. Amar Ramdane Cherif, directeur de thèse
Professeur des Universités à l'université De Versailles Saint Quentin-En-Yvelines

M. Chakib Tadj, directeur de thèse
Professeur au département de génie électrique à l'École de technologie supérieure

M. Chamseddine Talhi, président du jury
Professeur au département de génie logiciel et des TI à l'École de technologie supérieure

Mme Elisabeth Métais, examinateur externe
Professeur au Concervatoire national des arts et métiers

Mme Ghizlane El Boussaidi, membre du jury
Professeur au département de génie logiciel et des TI à l'École de technologie supérieure

M. Roch Glitho, membre du jury
Professeur à l'université Concordia

Mme Nicole Lévy, membre du jury
Professeur au Concervatoire national des arts et métiers

ELLE A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 11 JUIN 2013

À L'UNIVERSITÉ DE VERSAILLES SAINT-QUENTIN-EN-YVELINES

REMERCIEMENTS

Avant tout, j'adresse mes sincères remerciements à M. Amar Ramdane Cherif et M. Chakib Tadj, mes deux directeurs de thèse pour leur soutien tout au long de ces trois années de thèse. Je ne saurais dire combien nos échanges et leurs nombreux conseils m'ont été précieux.

Je remercie vivement les membres de jury – Mme Ghizlane El Boussaidi, Mme Nicole Lévy, Mme Elisabeth Métais, M. Roch Glitho et M. Chamseddine Talhi, – d'avoir participé à mon jury de thèse et les en remercie sincèrement.

Enfin, je ne peux terminer sans remercier mes proches de tout cœur et notamment mes parents qui, au cours de ces trois années de thèse, m'ont toujours soutenu et encouragé, comme d'habitude.

… Spéciale dédicace à une personne qui avec amour et courage s'est infatigablement dévouée pour que je puisse terminer cette thèse.

FUSION MULTIMODALE POUR LES SYSTÈMES D'INTERACTION

Ahmad WEHBI

RÉSUMÉ

Les chercheurs en informatique et en génie informatique consacrent une partie importante de leurs efforts sur la communication et l'interaction entre l'homme et la machine. En effet, avec l'avènement du traitement multimodal et du multimédia en temps réel, l'ordinateur n'est plus considéré seulement comme un outil de calcul, mais comme une machine de traitement, de communication, de collection et de contrôle, une machine qui accompagne, aide et favorise de nombreuses activités dans la vie quotidienne. Une interface multimodale permet une interaction plus flexible et naturelle entre l'homme et la machine, en augmentant la capacité des systèmes multimodaux pour une meilleure correspondance avec les besoin de l'homme. Dans ce type d'interaction, un moteur de fusion est un composant fondamental qui interprète plusieurs sources de communications, comme les commandes vocales, les gestes, le stylet, etc. ce qui rend l'interaction homme-machine plus riche et plus efficace.

Notre projet de recherche permettra une meilleure compréhension de la fusion et de l'interaction multimodale, par la construction d'un moteur de fusion en utilisant des technologies de Web sémantique. L'objectif est de développer un système expert pour l'interaction multimodale personne-machine qui mènera à la conception d'un outil de surveillance pour personnes âgées, afin de leurs assurer une aide et une confiance en soi, à domicile comme à l'extérieur.

Mots clés : Fusion, Interaction multimodale, Ontologie, Modalités, Réseau de Petri coloré.

MULTIMODAL FUSION FOR INTERACTION SYSTEMS

Ahmad WEHBI

ABSTRACT

Researchers in computer science and computer engineering devote now a significant part of their efforts in communication and interaction between human and machine. Indeed, with the advent of real-time multimodal and multimedia processing, computer is no longer seen only as a calculation tool, but as a machine of communication processing, a machine that accompanies, assists or promotes many activities in daily life. A multimodal interface allows a more flexible and natural interaction between a user and a computing system. It extends the capabilities of this system to better match the natural communication means of human beings. In such interactive system, fusion engines are the fundamental components that interpret input events whose meaning can vary according to a given context. Fusion of events from various communication sources, such as speech, pen, text, gesture, etc. allow the richness of human-machine interaction

This research will allow a better understanding of the multimodal fusion and interaction, by the construction of a fusion engine using technologies of semantic web domain. The aim is to develop an expert fusion system for multimodal human-machine interaction that will lead to design a monitoring tool for normal persons, seniors and handicaps to ensure their support, at home or outside.

Keywords: Fusion, Mulimodal interaction, Ontology, Modalities, Colored Petri Net.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

ARID :	Active Reviews for Intermediate Design
ATAM :	Architecture Tradeoff Analysis Method
BRETAM :	Breakthrough, Replication, Empiricism, Theory, Automation and Maturity
CPN :	Coloured Petri Nets
DAWG :	RDF Data Access Working Group
DLP :	Description Logic Programs
DOLCE :	Descriptive Ontology for Linguistic and Cognitive Engineering
EMMA :	Extensible Multimodal Annotation Markup Language
ESPR :	Episodic Skeletal-Plan Refinement Method
GUI :	Graphical User Interface
HMM :	Hidden-Markov Model
HTML :	HyperText Markup Language
ICARE :	Complementarity Assignment Redundancy Equivalence
JADE :	Java Agent Development Framework
KCCU :	Keyboard Cursor Control Units
M3L :	MultiModal Markup Language
MATIS :	Multimodal Air Traffic Information System
MOF :	Meta-Object Facility
MTC :	Member-Team Committee
MuliML :	Multimodal Markup Language
OCL **:**	Object Constraint Language
OIL :	Ontology Inference Layer ou Ontology Interchange Language
OKBC :	Open Knowledge-Base Connectivity
OWL :	Ontology Web Language
RDF :	Resource Description Framework
RuleML :	Rule Markup Language
SAAM :	Software Architecture Analysis Method
SALT :	Speech Application Language Tags
SMUIML :	Synchronized Multimodal User Interfaces Modeling Language

SPARQL : SPARQL Protocol and RDF Query Language
SQWRL : Semantic Query Language Rule-enhanced Web
SUMO : Suggested Upper Merged Ontology
SWRL : Semantic Web Rule Language
TATN : Temporal Augmented Transition Network
UML : Unified Modeling Language
WIMP : Windows, Icons, Menus and Pointing device
XML : Extensible Markup Language
XSLT : Extensible Stylesheet Language Transformation

LISTE DES SYMBOLES ET UNITÉS DE MESURE

C : une classe de l'ontologie

G_m : Modalité Gestuelle

Mi : modalité i

M_m : Modalité à saisie manuelle

R : une propriété d'objet

$t1Mi$: heure d'arrivée de la modalité i

$t2Mi$: heure de fin de la modalité i

$t_{maxActiveModalityTime}$: temps d'attente maximale autorisé pour une autre modalité impliquée dans une même commande

T_0 : temps initial d'une commande

T_{max} : temps maximal d'une commande

T_m : Modalité tactile

VO_m : Modalité Vocale

VI_m : Modalité visuelle

o : Une instance

$\wedge$: AND logique

$\vee$: OR logique

INTRODUCTION

La recherche en informatique consacre aujourd'hui une partie importante de ses efforts sur la communication et l'interaction personne-machine. En effet, avec l'avènement des traitements multimodaux et du multimédia en temps réel, l'ordinateur n'est plus seulement considéré comme un outil de calcul, mais comme une machine de traitement de communication, de perception et de contrôle, une machine qui accompagne, assiste et stimule de nombreuses activités dans la vie quotidienne.

Les systèmes d'interaction multimodale permettent aux utilisateurs d'interagir avec des ordinateurs grâce à des modalités d'entrée différentes (la parole, le geste, la vision, etc.) et des canaux de sortie (texte, graphiques, sons, avatars, discours synthétisés, etc.). Ce type d'interface utilisateur n'est pas seulement bénéfique pour une meilleure accessibilité, mais aussi pour plus de commodité (la reconnaissance naturelle d'un mode d'entrée), ainsi que de flexibilité (l'adaptation aux contextes et à l'environnement).

Un des objectifs de la recherche en interface IHM (interface homme/machine) est d'augmenter les capacités communicatives entre l'homme et l'ordinateur, par exemple en étudiant et en s'inspirant de la multimodalité de la communication humaine pour améliorer la communication entre les utilisateurs et les ordinateurs. Cela consiste à développer des spécifications, des outils, des logiciels et des méthodologies d'évaluation appropriés pour la coordination de plusieurs médias et modalités de communication, comme la parole, les gestes, etc.

L'utilisateur aura le choix de la modalité qu'il peut ou préfère utiliser à un moment donné. On parle alors d'équivalence, dans le sens où les modalités ne sont pas égales, mais lui permettent d'obtenir le même résultat de la part du système. Il est alors important lors de la conception du système de procéder à une étude des besoins et à une analyse de la tâche, afin de proposer une combinaison de médias et des modalités appropriées.

Ce projet de recherche s'inscrit dans ce cadre, et plus spécifiquement, il porte sur la création d'un moteur de fusion pour l'interaction multimodale. La fusion est une étape cruciale et déterminante dans la combinaison et l'interprétation des modalités d'entrées différentes (par

exemple, la parole, le geste, le regard, etc.). Jusqu'à maintenant, le développement de ces moteurs est toujours complexe et encombrant, soit dans la conception, soit dans la spécification ou la validation.

Ce projet propose une nouvelle solution de modélisation, une architecture qui facilite le travail d'un moteur de fusion, en utilisant une ontologie qui décrit l'environnement et prend en considération des contextes et des pré-conditions bien définis.

Le chapitre 1 fera l'objet d'une présentation de l'état de l'art qui aborde les travaux réalisés dans le domaine de l'interaction multimodale. Nous présenterons, dans le chapitre 2, l'objectif de cette recherche, la méthodologie appliquée et l'originalité des travaux. Le chapitre 3 présente les différents composants nécessaires pour la modélisation d'une architecture de fusion. Dans le chapitre 4, nous présenterons la conception de notre architecture proposé, la création d'une ontologie qui décrit l'environnement, la sélection des modalités et des modèles d'ordres, l'algorithme de fusion et un scénario qui montre le fonctionnement de l'algorithme de fusion. Le chapitre 5 fera l'objet d'une validation du fonctionnement de l'architecture à l'aide d'un réseau de Petri coloré en utilisant un outil appelé CPN-Tools. Le Chapitre 6 présente un prototype d'un système de fusion multimodale développé en Java. Enfin, nous conclurons ce document par un résumé de nos principales contributions et par une description des perspectives que nous envisageons pour l'avenir.

CHAPITRE 1

REVUE DE LA LITTÉRATURE

1.1 Introduction

L'étude de la littérature porte sur la fonctionnalité et l'architecture des systèmes et particulièrement celle des systèmes d'interaction multimodale. Comme la problématique à laquelle ce projet de recherche doit répondre est la fusion multimodale, l'étude bibliographique couvrira principalement ce thème. Les différentes architectures proposées dans la littérature seront décrites en détail.

Plusieurs moteurs de fusion ont été développés à ce jour avec des approches différentes pour permettre la fusion des données (Holzapfel ; Latoschik 2002)**,** mais ils ont des buts spécifiques, notamment la résolution des scénarios de fusion. Quand un nouveau scénario est identifié, une pratique courante consiste à mettre la fusion en reconsidération en raison du manque d'outils et de bases servant à des multiples implémentations. Nous croyons que le champ peut être considérablement amélioré si un soutien général à la fusion est prévu, en tenant compte des caractéristiques et des mécanismes existants de la fusion.

Selon (Coutaz 1991), dès qu'il y a interprétation ou génération d'informations de nature différente pour extraire ou pour émettre du sens, le système est dit multimodal. Un nombre important de chercheurs ont étudié l'impact de l'utilisation des techniques de l'interaction multimodale dans les domaines de la sécurité des systèmes critiques :

- des systèmes militaires et l'aviation civile, telles que la planification des missions (Koons 1993), (Bouchet 2004), des cockpits interactifs pour les civils (Bourguet 2002), des aéronefs militaires (Bouchet 2004), de la gestion du trafic aérien et de la gestion des incidents (Navarre 2008), etc. ;
- système de gestion des crises (Pfleger 2004) pour l'envoi des véhicules d'urgence ;
- des environnements interactifs pour la formation par simulation (Martin 1998) dans l'armée, tels que *Simnet* ou *Leathernet.*

Ces études et d'autres telles que présentées dans (Krahnstoever 2002) ont proposé et testé l'utilisation des techniques d'interaction multimodale dans le domaine des systèmes de sécurité critique et ont signalé plusieurs avantages :

- La multimodalité augmente la fiabilité de l'interaction. En effet, elle permet de diminuer considérablement les erreurs critiques au cours de l'interaction. Cet avantage à lui seul peut justifier l'utilisation de la multimodalité lors de l'interaction avec un système de sécurité critique ;
- La multimodalité augmente l'efficacité de l'interaction, en particulier dans le domaine des commandes spatiales nécessaires pour la précision des informations et des coordonnées de localisation) ;
- Les utilisateurs préfèrent interagir principalement de manière multimodale, probablement parce qu'elle permet plus de souplesse dans l'interaction, en tenant compte de la variabilité des utilisateurs ;
- La multimodalité permet l'augmentation de la flexibilité de l'interaction, de sorte que la période d'apprentissage sera plus courte.

Pour toutes ces raisons, la multimodalité a acquis un intérêt accru dans le domaine des systèmes de commande et de contrôle interactifs.

1.2 Terminologie de la fusion

Les mécanismes utilisés pour combiner les données ont reçu différents noms dans le passé. Par exemple, Cubricon (Neal 1989) utilise le terme de "combinaison" (combinant le geste et le langage naturel), tandis que Martin *et al.* (Martin 1998) utilisent le mot "coopération" des modalités. (Cohen 1997), Pfleger (Pfleger 2004) ont utilisé le terme «intégration». De même, Latoschik (Latoschik 2002), Johnston *et al.* (Johnston 2005) et Shiker *et al.* (Shikler 2004) ont travaillé sur l'intégration multimodale. Plus largement, le mot fusion a été utilisé pour décrire un tel mécanisme. Flippo *et al.* (Flippo 2003), Nigay *et al.* (Nigay 1993) et Portillo *et al.* (Portillo 2006) ont parlé de processus de fusion. Milota (Milota 2004) a parlé de stratégies

de fusion. Nigay *et al.* (Nigay 1995) ainsi que Dumas *et al.* (Dumas B. 2009) ont parlé de mécanismes de fusion. Dans certains travaux, l'accent a été mis sur l'élément d'information de la multimodalité, plutôt que sur le procédé de combiner l'information comme dans (Sun 2006) où les auteurs ont utilisé le mot "fusion de données". Une distinction entre la combinaison des modalités d'entrée et de sortie a été rendue explicite par Nigay *et al.* (Nigay 1995) et Melichar *et al.* (Melichar 2006)**,** qui ont défini ce concept comme fusion multimodale des entrées alors que Mansoux *et al.* (Mansoux 2006) se sont concentrés sur un cadre basé sur des composants pour la multimodalité de sortie. De nombreux auteurs ont récemment utilisé le nom de fusion multimodale (Duarte 2006) (Holzapfel 2004).

1.3 Évolution de la fusion multimodale suivant le model BRETAM

Le modèle BRETAM (Melichar 2006) présenté dans la Figure 1.1 est formé de six étapes, à savoir "**B**reakthrough, **R**eplication, **E**mpiricism, **T**heory, **A**utomation and lastly **M**aturity".

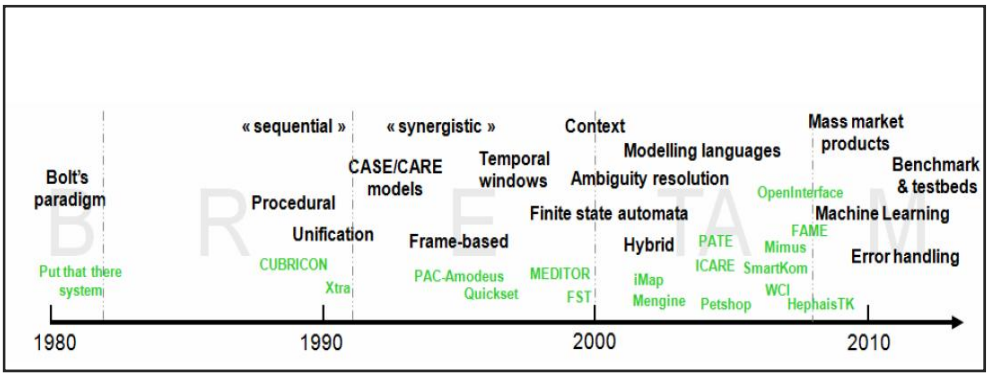


Figure 1.1 Évolution du concept de la fusion multimodale suivant le model BRETAM
Tirée de (Melichar 2006)

1.3.1 Phase de découverte

La technologie des moteurs de fusion a commencé avec BOLT et son système "*put that there*" (Melichar 2006). Toutefois, la notion de "moteur de fusion " n'a été ni présentée ni examinée dans son document, même si la fusion des informations produites par les deux modalités (le geste et la parole) a été abordée lors de l'implémentation. Depuis, les interfaces

multimodales ont commencé à être conçues et implémentées, ce qui a déplacé le champ de recherche dans la phase de réplication.

1.3.2 Phase de réplication

Les travaux de la phase de réplication sont restés à un niveau très élevé d'abstraction, axés sur l'identification des problèmes liés à la fusion plutôt que de proposer des solutions. CUBRICON (Neal 1989) a utilisé la parole avec le geste et l'expression graphique. CUBRICON contient un ensemble de règles permettant d'inférer le référent en cas d'ambiguïté. Cela se fait en sélectionnant l'objet le plus proche qui satisfait certains critères ou par l'émission d'un avis consultatif décrivant les incohérences. Ces règles peuvent être considérées comme la première représentation explicite du comportement du moteur de fusion. Xtra (Wahlster 1991) est un système interactif multimodal fondé sur le clavier pour le langage naturel, et la souris comme modalités d'entrée. L'idée d'Xtra est d'exploiter une interface multimodale afin d'augmenter la bande passante entre les utilisateurs et le système sous-jacent de déclaration d'impôt.

CUBRICON et Xtra peuvent être considérés comme un premier pas vers la conception de nouveaux moteurs de fusion. Toutefois, CUBRICON et XTRA ont mis l'accent sur l'utilisation séquentielle des modalités. Par exemple, dans CUBRICON, l'utilisateur doit arrêter de parler avant de pointer. En conséquence, ces premières descriptions de moteurs de la fusion ne portent que sur une partie de l'espace de conception.

1.3.3 Phase empirique

Quatre importantes contributions ont affecté cette phase : l'intégration de la parole, du regard et des gestes de la main par Koons *et al.* (Koons 1993), l'architecture PAC-Amodeus et son moteur de fusion (Nigay 1995), la plate-forme QuickSe (Cohen 1997) et le moteur de fusion élaboré par Johnston et Bangalore (Johnston 2005). L'étude de Koons *et al.* (Koons 1993) a élaboré les trois modalités : le regard, la parole et les gestes des mains dans un système graphique 3D appelé « blocks world ». Les modalités sont d'abord analysées

individuellement. Les analyseurs produisent par la suite l'information dans un format commun basé sur des trames pour la fusion. Toutes les informations sont reçues en parallèle.
PAC-Amodeus (Nigay 1995) est un modèle d'architecture logicielle pour les systèmes multimodaux interactifs. L'architecture est illustrée par un système appelé MATIS (Multimodal Air Traffic Information System). MATIS est une interface multimodale pour une base de données. Il intègre plusieurs modalités telles que des entrées utilisant le langage naturel (par la parole et le texte via le clavier), des entrées graphiques via une souris et la manipulation directe de la technique d'interaction. Il est possible pour l'utilisateur d'utiliser n'importe quelle modalité pour le déclenchement des commandes du système puisque les modalités sont "équivalentes ", selon les propriétés définies dans (Nigay 1995). Dans MATIS, la fusion est faite à un haut niveau d'abstraction (dans un composant appelé *Dialogue contrôlé* défini dans l'architecture PAC-Amodeus) en utilisant un moteur de fusion générique basé sur une représentation commune, appelée "melting-pot". Dans le melting-pot (qui est une structure en deux dimensions représentant un événement avec les deux parties structurelles et temporelles de l'information), la fusion obéit à trois principes : la complémentarité, le temps et les règles de contexte dans le cas d'un retard important.

QuickSet (Cohen 1997) est un système interactif disposant d'une interface multimodale à modalités : graphique (en utilisant une interaction basée sur un stylo) et de la parole (en utilisant un système de reconnaissance vocale). Ces modalités sont utilisées pour contrôler "Leathernet". Celui-ci est un système de simulation pour la formation des Marines des États-Unis. La fusion dans QuickSet se fait au moyen de l'unification. Celle-ci vérifie la cohérence de deux parties d'une information partielle (à partir des deux modalités). Si les informations reçues sont conformes, Quickset les combine dans un seul résultat.

Johnston & Bangalore (Johnston 2005) ont présenté une interface multimodale pour un répertoire d'entreprise et des systèmes de messagerie interactive. Le système comporte deux modalités : un stylo et la parole. Les deux modules responsables de la reconnaissance (chargés de la réception des événements générés par le périphérique d'entrée) envoient à la partie d'intégration (le moteur de la fusion) un treillis représentant les chaînes possibles reconnues et les possibles gestes reconnus.

Dans ce type de systèmes, l'utilisateur peut utiliser les modalités de façon concurrente, c'est-à-dire en même temps. Ceci rend la tâche des moteurs de fusion plus complexe. Par exemple, le traitement du délai en cours entre les processus liés à l'interprétation des modalités telles que la parole et la manipulation directe des ambiguïtés en utilisant la souris. Dans chacune de ces contributions, les auteurs ont proposé une représentation sur laquelle s'appuie le moteur de fusion

1.3.4 Phase de théorie et d'automatisation

Dans la phase d'automatisation, les théories sont acceptées et utilisées automatiquement pour prédire des expériences et générer des règles de conception. Le premier groupe de travail présenté dans cette phase est un ensemble de cinq contributions sur lesquelles s'appuie la phase empirique présentée dans le paragraphe 1.3.3. Latoschik (Latoschik 2002) a étendu le travail de Johnston et Bangalore dans TATN (temporal augmented transition network) pour représenter des aspects quantitatifs temporels dans le moteur de fusion. La nécessité d'une représentation quantitative du temps a déjà été identifiée dans MATIS. Par contre, Latoschik a introduit une notation formelle pour traiter cette idée. Flippo (Flippo 2003) et Portillo (Portillo 2006) ont combiné des techniques de QuickSet et PAC-Amodeus afin de créer un moteur de fusion hybride qui exploite à la fois des délais et des mécanismes pour la résolution des ambiguïtés de l'unification. Bouchet & Nigay (Bouchet 2004) ont élaboré leurs travaux en se basant sur le modèle PAC-Amodeus et en définissant un ensemble de micromoteurs de fusion autant que des composants logiciels réutilisables et composables.

1.3.5 Phase de maturité

Les interfaces multimodales et les moteurs de fusion ont atteint une telle phase de maturité, quand les théories ont été assimilées et utilisées en routine sans questions. Une façon d'évaluer cette caractéristique pour une technologie, c'est quand elle commence à être déployée dans les grandes applications pratiques, telles que la console de jeux Wii (Schlomer 2008) et l'iPhone (Jobs 2008)**,** qui disposent de plusieurs périphériques d'entrée (les manettes de Wii) ou l'interaction tactile d'iPhone. Cette technologie a été encore déployée dans le

domaine des systèmes de sécurité critique, tels que l'introduction de KCCU (Keyboard Cursor Control Units) dans les cockpits des gros avions du type Airbus 380 ou Boeing 787, qui est nécessaire pour gérer l'utilisation synergique de plusieurs périphériques d'entrée (même si l'un est géré par le pilote et l'autre par le copilote) (Navarre 2008).

1.4 Caractéristiques du moteur de fusion

Un des problèmes majeurs abordés par les moteurs de fusion multimodale réside dans les différents types d'entrées à manipuler. En outre, les moteurs de fusion manipulent une combinaison temporelle des entrées déterministes ainsi que des entrées non déterministes ; dans ce cas, l'interprétation des données provenant des modalités reste incertaine jusqu'à ce que des renseignements supplémentaires tels que le contexte d'un environnement soient fournis.

Entrées probabilistes

Dans un GUI (graphical user interface) standard, la souris et le clavier sont utilisés pour contrôler la machine. Ils correspondent à des événements déterministes. Mais les flux d'entrée peuvent aussi correspondre à la communication naturelle des humains telle que la parole ou le geste. Ils doivent d'abord être interprétés par un outil de reconnaissance probabiliste comme les HMM (Hidden Markov Model). Les résultats peuvent être pondérés par un degré d'incertitude.

Combinaisons multiples et temporelles

La synchronisation entre les modalités d'entrée est une problématique particulière, car l'interprétation peut varier en fonction du temps pendant lequel les modalités sont utilisées.

Adaptation aux contextes, aux tâches et aux utilisateurs

Une commande multimodale peut être interprétée différemment selon le contexte (par exemple, la maison, la voiture, le bureau), la tâche à exécuter, l'état de l'application et les préférences de l'utilisateur.

Gestion des erreurs

Les erreurs seront difficiles à éviter avec des entrées probabilistes qui peuvent être combinées de différentes manières au fil du temps et qui doivent être interprétées pour tenir compte du contexte, des tâches et des préférences des utilisateurs. Les moteurs de fusion devraient prévoir des mécanismes pour permettre aux utilisateurs de corriger les réponses de la machine afin qu'elle puisse apprendre de ses erreurs. Les moteurs de fusion devraient s'attaquer à ces fonctionnalités afin de permettre une interaction multimodale robuste et en temps réel avec les systèmes interactifs.

1.5 Mécanisme de fusion

Un des principaux problèmes de la conception multimodale est la façon de fusionner les entrées multimodales d'une personne pour aboutir à une interprétation consistante. La racine du problème vient de la manière dont les entrées sont gérées dans les interfaces multimodales. Dans la norme des interfaces WIMP (Windows, Icons, Menus and Pointing device), les événements comme les clics de souris ou les commandes du clavier peuvent être traités un à la fois, sans être en face de problèmes de références pour chaque entrée multimodale (donner une référence pour chaque entrée). Dans les systèmes multimodaux, non seulement ces problèmes se posent, mais ils sont en fait une propriété essentielle dans l'interaction multimodale. Un exemple classique est la phrase de Bolt "*Put that there*"(Bellik 1997), dans laquelle une commande vocale *put* doit être reliée à une paire d'événements d'indication *that* et *there*, réalisés dans une modalité de geste. Trois niveaux principaux de fusion sont généralement considérés (Bolt. 1980) : 1) le niveau de fusion des données, 2) le niveau de fusion des caractéristiques et 3) le niveau de fusion décisionnel. Le niveau de fusion des données n'est pas en soi un mode de fusion intermodal, mais il est utilisé quand il y a plusieurs signaux provenant d'un même type de source (par exemple, deux caméras). Le niveau de fusion des caractéristiques est le type de fusion favori, lorsque les modalités étroitement couplées et synchronisées doivent être fusionnées. Le niveau de fusion décisionnel est le plus utilisé dans les applications multimodales. La raison principale est sa capacité à gérer les modalités faiblement couplées comme l'interaction d'un stylo et la

parole. Le niveau de fusion décisionnel a été construit suivant un certain nombre de mécanismes de fusion, dont certains sont apparus récemment. Tous ces mécanismes ont un objectif commun, celui de réaliser la fusion au niveau sémantique et de traiter des interprétations potentiellement erronées provenant de différents canaux de modalité, de synchronisation et des problèmes de référence des entrés multimodales.

1.6 Algorithmes de fusion

Les algorithmes typiques de la fusion au niveau décisionnel sont basés sur la fusion par trames, la fusion par unification et la fusion hybride symbolique/statistique.

La fusion basée sur des trames

La fusion basée sur des trames (Vo 1996) utilise des structures de données appelées trames pour la représentation des données provenant de différentes sources ou modalités. La signification des trames est un concept d'intelligence artificielle qui a été d'abord conçu par Marvin Minsky en 1974 dans son article «A framework for representing knowledge » (Minsky 1974). Dans la proposition initiale de Minsky, les trames sont considérées comme une structure de donnée d'intelligence artificielle qui représente des sous-structures d'une idée représentant des objets comme des paires de valeur-attribut. Les trames sont alors connectées ensemble pour former l'idée.

La signification des trames dans l'interaction multimodale est un peu différente. Dans ce cas, les «idées» représentées par les trames sont étroitement liées aux opérations d'interaction. Vo et Wood donnent dans (Vo 1996) un moyen d'utiliser des trames pour l'interaction multimodale, dont les interprétations sémantiques sont fusionnées de manière incrémentielle. La Figure 1.2 explique comment Vo et Wood ont réalisé la fusion des entrées par une trame de signification. Dans leur système, des trames sont progressivement comblées par les différentes sources, puis fusionnées de manière récursive, jusqu'à ce qu'une hypothèse satisfaisante soit captée par le gestionnaire de dialogue.

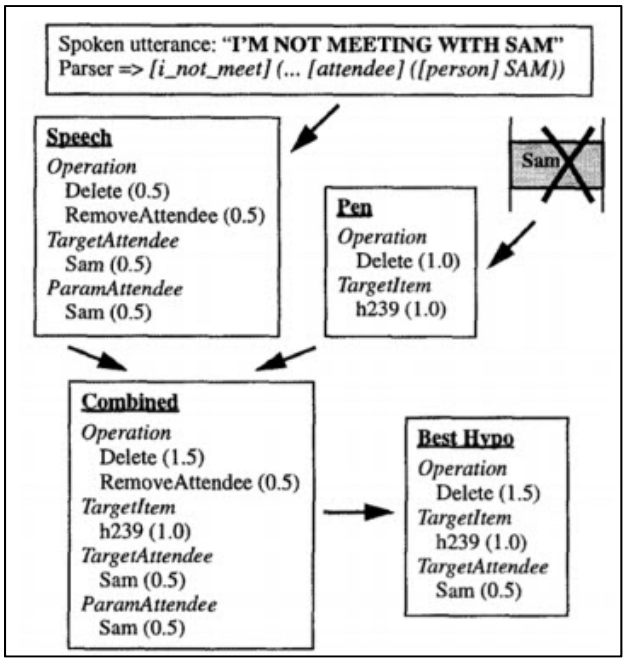


Figure 1.2 Exemple de fusion des trames
Tirée de (Vo 1996)

La fusion par unification

La fusion par unification (Johnston 1997) est basée sur la fusion récursive des structures attribut-valeur pour obtenir une représentation de signification logique. Un exemple est illustré dans la Figure 1.3. Il considère un utilisateur qui interagit avec une application de planification militaire et il désire créer une ligne de fil barbelé : l'utilisateur prononce « fil de barbelés » comme une intention de créer un objet de barbelés dans l'application. Le système multimodal interprète le but, mais n'exécute pas la commande car il reste encore un objet qui est la ligne. L'utilisateur peut alors utiliser la modalité stylo pour dessiner une ligne sur l'écran. Les coordonnés seront alors transmises au module de fusion. La structure étant maintenant terminée, la commande peut être exécutée par le système. Comme avec les trames, les structures attribut-valeur sont utilisées, la principale différence se situe dans la façon dont les données contradictoires sont résolues. Néanmoins, les deux algorithmes de fusion partagent les mêmes bases conceptuelles, comme Johnston *et al.* (Johnston 1997), qui

ont écrit : «*Vo et Wood 1996 présentent une approche pour l'intégration multimodale dans le même esprit de celui présenté ici [...]*. »

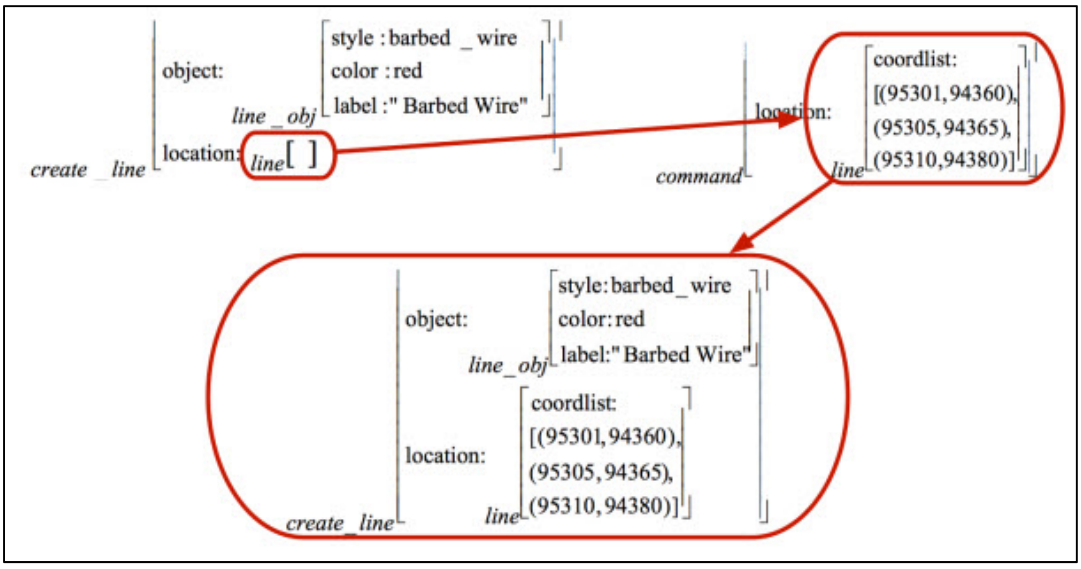
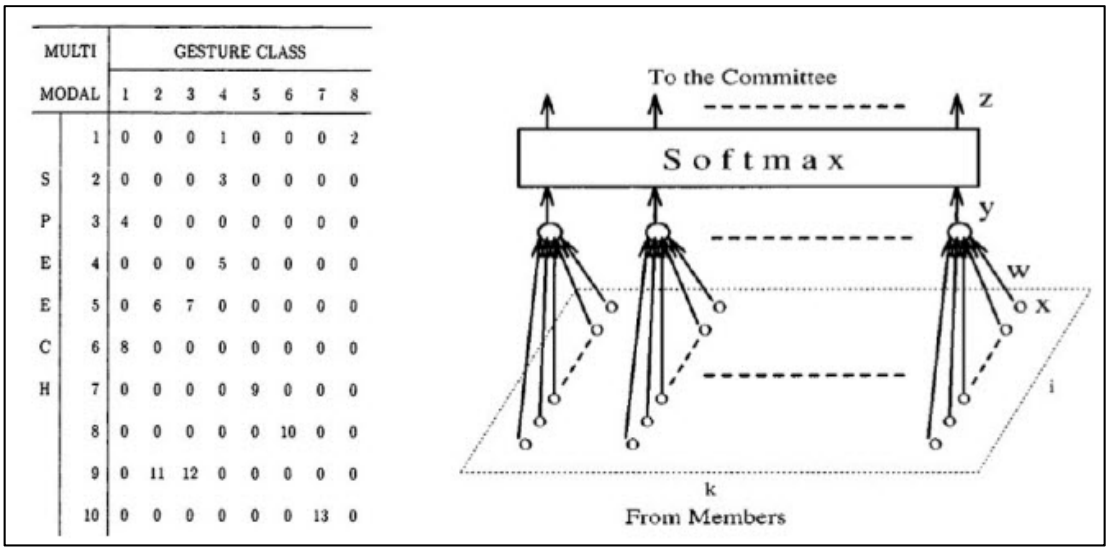


Figure 1.3 Exemple sur la fusion par unification
Tirée de (Johnston 1997)

La fusion hybride symbolique/statistique

La fusion hybride symbolique/statistique (Wu 2002) est une évolution de la fusion par unification qui ajoute des processus statistiques aux techniques de fusion décrites ci-dessus. Ce type de fusion hybride a été démontré pour obtenir des résultats fiables et robustes. Un exemple classique d'une technique de fusion statistique/symbolique est l'architecture *member-Team Committee* (MTC) du système (Wu 1999) . Dans cet algorithme, les entrées de la parole et des gestes sont organisées dans une « carte associative » (voir Figure 1.4) qui décrit les paires parole-gestes. À côté de cette carte associative, il y a l'algorithme basé sur des statistiques, qui est formé par trois couches : 1) la couche inférieure est composée d'éléments de reconnaissance, dont chacune est un estimateur postérieur local affecté par un sous-ensemble de variables d'entrée, un type de modèle spécifié et de complexité, et la validation d'un ensemble des données. 2) La couche supérieure est composée d'« équipes » de membres qui coopèrent afin de déterminer l'intégration des membres multiples. Le but étant de réduire l'incertitude d'intégration. 3) Enfin, les décisions des différentes équipes sont

soumises à un « comité », qui donne une décision finale après avoir comparé les distributions empiriques postérieures des différentes équipes.



MULTI MODAL		GESTURE CLASS							
		1	2	3	4	5	6	7	8
S	1	0	0	0	1	0	0	0	2
P	2	0	0	0	3	0	0	0	0
E	3	4	0	0	0	0	0	0	0
E	4	0	0	0	5	0	0	0	0
C	5	0	6	7	0	0	0	0	0
H	6	8	0	0	0	0	0	0	0
	7	0	0	0	0	9	0	0	0
	8	0	0	0	0	0	10	0	0
	9	0	11	12	0	0	0	0	0
	10	0	0	0	0	0	0	13	0

Figure 1.4 Algorithme de fusion hybride
Tirée de (Wu 1999)

Il est également intéressant de noter que l'apprentissage automatique a déjà été appliqué pour la fusion de données d'entrée dans des systèmes non interactifs. L'apprentissage de la machine a été principalement appliqué au niveau des fonctionnalités, avec moins de travaux réalisés sur la fusion au niveau de la décision avec l'aide de l'apprentissage automatique. Un exemple d'un tel travail est dans (Pan 1999). Celui-ci a proposé une version de « contexte-dépendant » de la méthode d'inférence bayésienne pour la fusion de données multi-sensorielles. Néanmoins, (Jaimes 2007) estime que « d'autres recherches sont encore nécessaires pour enquêter sur les modèles de fusion capables d'utiliser efficacement les signaux complémentaires fournis par de multiples modalités ».

(Jaimes 2007) a également mentionné : « La plupart des chercheurs traitent chaque canal (visuel, audio) de manière indépendante, et la fusion multimodale en est encore à ses débuts. » Ainsi, les chercheurs dans le domaine de l'interaction multimodale ont travaillé afin d'atteindre une fusion multimodale efficace, avec une considération attentive des différentes modalités disponibles et la manière dont les modalités interagissent. Outre la fusion

multimodale, l'apprentissage des machines aidera les systèmes multimodaux à prendre en compte l'aspect « communication-émotions » en fonction de leurs manifestations physiologiques (McNeill 1992), telles que les expressions faciales, les gestes (Clay 2009), les postures, le ton de la voix, la respiration, etc.

1.7 Méthodes de fusion multimodale

Dans cette partie, nous donnons un aperçu des différentes méthodes de fusion qui ont été utilisées par les chercheurs. Les méthodes de fusion sont réparties dans les trois catégories suivantes: les méthodes à base de règles, les méthodes classification et les méthodes d'estimation comme le montre la Figure 1.5.

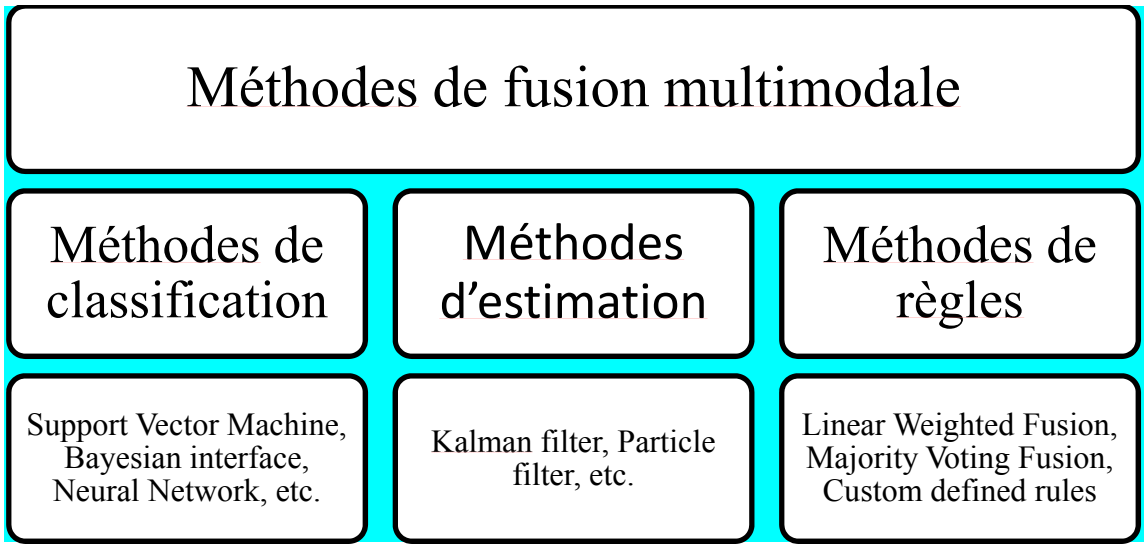


Figure 1.5 Les méthodes de fusion

- La méthode à base de règles comprend une variété de règles de base de la combinaison de l'information multimodale. Il s'agit notamment de méthodes à base de règles statistiques telles que la fusion linéaire pondérée (Wang 2003) (somme et produit), MAX, MIN, AND, OR. Le travail par Kittler (Kittler 1998) a fourni l'introduction théorique de ces règles. En plus de ces règles, il y a des règles personnalisées définies sur mesure (Pfleger 2004) qui sont construites pour la perspective d'une application spécifique. Les systèmes à base de règles fonctionnent généralement bien si la qualité de l'alignement temporel entre les différentes

modalités est bonne. Les méthodes de fusion basées sur des règles, telle que la méthode de fusion pondérée linéaire a été largement utilisée par les chercheurs. Il s'agit d'une approche de calcul moins compliquée et simple. Cette méthode fonctionne bien si le poids des différentes modalités est déterminé de façon appropriée. Dans la littérature, cette méthode a été utilisée pour la détection des visages, le suivi du regard, la détection de monologue, la reconnaissance vocale, l'image et la récupération de la vidéo, et l'identification de la personne. D'autre part, la fusion avec les règles personnalisées permet d'ajouter des règles basées sur les exigences. Cependant, la définition des règles nécessite une bonne connaissance du domaine. Le Tableau 1.1 montre quelques exemples des systèmes de fusion multimodale.

Tableau 1.1 Les systèmes de fusion basés sur les méthodes de règles

	Méthode	**Références**	**Modalités**	**Type du système**
Méthodes de règles	Linear weighted fusion	(McDonald 2005)	Text and video (color, edge and texture)	Video retrieval
		(Yang 2004)	Video (trajectory coordinates)	Human tracking
		(Kankanhalli 2006)	Video (color, motion and texture)	Detection, monologue detection and traffic monitoring
		(Jaffre and Pinquier 2006)	Audio, video index	Person identification from audio-visual sources
	Majority voting rule	(Radova 1997)	Raw speech (set of patterns)	Speaker identification from audio sources
	Custom defined rules	(Corradini 2004)	Speech, 2D gesture	Human computer interaction
		(Holzapfel 2004)	Speech, 3D pointing gesture	Multimodal interaction with robot
		(Pfleger 2005)	Pen gesture,	speech Multimodal dialog system

- La méthode de classification comprend une gamme de techniques de classification qui ont été utilisées pour classer l'observation multimodale dans une catégorie prédéfinie. Les méthodes de cette catégorie sont : *support vector machine* (SVM) (Burges 1998)*, Bayesian inference* (Hall 1997)*, dynamic Bayesian networks* (DBN) (Choudhury 2002)*, neural networks* (Wu 2003) et *maximum entropy mode* (Magalhaes 2007). La méthode de fusion de l'inférence bayésienne, permet une intégration facile de nouvelles observations et l'utilisation d'information a priori. Cependant, le manque d'une information a priori peut conduire à une fusion inexacte. Les réseaux bayésiens dynamiques ont été largement utilisés pour traiter des données de séries chronologiques. La méthode DBN sous ses différentes formes (comme HMM) a été utilisée avec succès pour diverses applications multimédia telles que la reconnaissance vocale, l'identification du locuteur, le suivi du regard, etc. Toutefois, cette méthode détermine difficilement l'état correct d'un DBN. Par rapport à DBN, la fusion par les réseaux de neurones est généralement adaptée pour fonctionner dans les cas de problème d'espace dimensionnel. Toutefois, en raison de la nature complexe d'un réseau, cette méthode souffre de formation lente. La méthode SVM a été préférée en raison de sa meilleure performance de classification, mais DBN a été trouvée plus appropriée pour les modèles des données temporelles. Le Tableau 1.2 montre quelques exemples des systèmes de fusion multimodale basée sur la méthode de classification.

Tableau 1.2 Systèmes de fusion basés sur la méthode de classification

	Méthode	**Références**	**Modalités**	**Type du système**
Méthodes de classification	Support Vector Machine	(Zhu et al. 2006)	(low-level visual features, text color, size, location, edge density, brightness, contrast)	Image classification
		(Ayache et al. 2007)	Visual, text cue	Semantic indexing
	Bayesian Inference	(Pitsikalis et al. 2006)	Audio (MFCC), video (Shape and texture)	Speech recognition
		(Atrey et al.	Audio (ZCR, LPC,	Event

Méthodes de classification	Méthode	Références	Modalités	Type du système
		2006)	LFCC) and video (blob location and area)	detection for surveillance
	Dynamic Bayesian Networks	(Ding and Fan 2007)	Video (spatial color distribution and the angle of yard lines)	Shot classification in a sports Video
		(Town 2007)	Video (face and blob)	ultrasonic sensors Human tracking
	Neural Networks	(Zou and Bhanu 2005)	Audio (spectrogram) and video (blob)	Human tracking
		(Ni et al. 2004)	Image (features details not provided in the paper)	Image recognition
	Maximum Entropy Model	(Magalhaes and Ruger 2007)	Text and Image	Semantic image indexing

- La méthode basée sur l'estimation comprend le filtre de Kalman, le filtre de Kalman étendu et le filtre à particules. Ces méthodes de fusion ont été utilisées pour évaluer l'état d'un objet en mouvement à partir des données multimodales. Par exemple, pour la tâche du suivi d'objets, des modalités multiples telles que l'audio et la vidéo sont fusionnées pour estimer la position de l'objet. Le Tableau 1.3 montre quelques exemples des systèmes de fusion multimodale basée sur la méthode d'estimation.

Tableau 1.3 Systèmes de fusion basés sur la méthode d'estimation

Méthodes d'estimation	Méthode	Références	Modalités	Type du système
	Kalman Filter	(Zhou and Aggarwal 2006)	Video [spatial position, shape, color (PCA),blob]	Person/vehicle tracking
		(Gehrig et al. 2005)	Audio (TDOA), video (position of the speaker)	Single speaker tracking
		(Potamitis et al. 2004)	Audio(position, velocity)	Multiple speaker tracking
		(Vermaak et	Audio(TDOA),	Single speaker

Méthodes d'estimation	Méthode	Références	Modalités	Type du système
	Particle Filter	al. 2007)	visual (gradient)	tracking
		(Nickel et al. 2005)	Audio(TDOA), video(Haar-like features)	Single speaker tracking
		(Perez et al. 2003)	Audi (TDOA), video(coordinates	Single speaker tracking

1.8 Raisonnement et représentation de la connaissance

1.8.1 Web sémantique RDF et OWL

Le Web sémantique met en évidence un ensemble de trois faits : la représentation du sens des informations qui sont publiées sur le Web (appelées ressources), la représentation des relations qui relient les informations ensemble et la normalisation des deux précédents. Nous avons besoin de représenter la sémantique d'une manière à permettre une bonne interaction. Cette sémantique doit être basée sur les langages du web sémantique présentés dans les différentes couches du standard W3C (www.w3.org) dans la Figure 1.6. Nous devons également partager cette représentation, c'est-à-dire la rendre populaire, largement adoptée et standard, de sorte que tout le monde puisse en bénéficier.

1.8.1.1 RDF Resource Description Framework

La *Resource Description Framework* (RDF) est un langage qui permet de représenter des informations sur les ressources dans le "World Wide Web" (Figure 1.6). Il est particulièrement destiné à la représentation des métadonnées sur les ressources Web, telles que le titre, l'auteur et la date de modification d'une page Web, le copyright et les informations de licence sur un document Web, ou le calendrier de disponibilité pour une ressource partagée. Toutefois, en généralisant le concept d'une ressource Web, RDF peut également être utilisé pour représenter des informations sur des éléments qui peuvent être identifiés sur le Web. Une introduction détaillée peut être trouvée dans (Manola 2004).

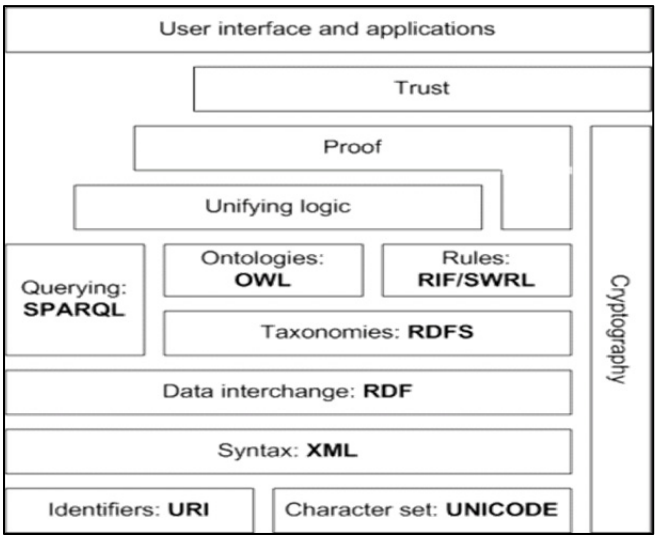


Figure 1.6 Couches du Web sémantique (schéma W3C)

Dans (Stuckenschmidt 2005), il est souligné que, si nous voulions décrire l'information sur le méta-niveau et définir le sens, nous devrions chercher des approches autres que la norme XML. Pour cela, la RDF a été proposée comme un modèle de données pour représenter les données méta.

De nombreuses contributions sur la RDF sont disponibles dans la littérature, telles que (Guha 2004) et (Lassila 2000). Néanmoins, la RDF est représentée sous forme de syntaxe XML. Le langage de schéma correspondant à définir le vocabulaire est appelé Schéma RDF ou RDFS. En RDFS, nous pouvons définir quelles propriétés à appliquer sur un type d'objets et les valeurs à tenir. En outre, nous pouvons décrire les relations entre les objets.

1.8.1.2 OWL Ontology Web Language

Avec OWL (Ontology Web Language), il est plus facile d'exprimer une sémantique qu'avec XML, RDF et RDFS (D L McGuinness 2004), car il y a beaucoup plus de fonctions de restriction sur des relations sémantiques. En outre, selon (Stuckenschmidt 2005), OWL a plus de vocabulaire pour décrire les propriétés et les concepts, les relations entre eux (par exemple, la dis-jointure), la cardinalité (par exemple, exactement une). OWL a une typologie plus riche de propriétés, caractéristiques des propriétés (par exemple la symétrie), et les

classes énumérées. OWL peut notamment être utilisé pour représenter explicitement la signification des termes et les relations entre eux. OWL est une révision de la DAML + OIL langage d'ontologie Web (Heckmann 2005). Dans OWL, le concept de la logique de description devient une classe, le rôle devient une propriété et les instances deviennent des objets ou bien des instances.

1.8.1.3 Comparaison d'RDF, OWL, UML et OCL

Dans cette partie, une comparaison entre quelques langages de modélisation des connaissances est faite, pour montrer les limites de l'RDF, l'UML et l'OCL par rapport à l'OWL. Ces langages ont été choisis car ils ont des restrictions en commun qui permettent la modélisation des connaissances, chacun à sa façon. RDF est un ensemble de concepts ou un langage utilisé pour décrire les connaissances. OWL est un ensemble de concepts ou un langage pour modéliser les connaissances. UML (Unified Modeling Language) est un langage semi-formel utilisé pour une modélisation basée sur les classes. Il comprend quelques similarités avec OWL comme les classes, les associations, les propriétés, les types et les instances (ODM 2008). Les deux approches présentent des restrictions qui peuvent être surmontées par l'intégration. Malgré ces similitudes, UML ne permet qu'une spécification statique de la spécialisation et de la généralisation des classes et des relations, alors qu'OWL fournit des mécanismes pour les définir d'une manière dynamique. Cela signifie qu'OWL permet la reconnaissance de la généralisation et de la spécialisation entre les classes ainsi que l'appartenance des objets en fonction des conditions imposées sur les propriétés des classes. OWL offre une manière plus expressive et évolutive de la modélisation de données et fournit des moyens polyvalents pour décrire les classes, afin de permettre l'inférence de type. En effet, comme l'UML, l'OWL fournit divers moyens pour décrire les classes, qui peuvent également être imbriquées les unes dans les autres d'une manière implicite. Par contre, on peut désigner une classe par un identificateur de classes, une énumération exhaustive des instances, des restrictions de propriétés, une intersection, une union ou un complément d'une description de la classe et d'autre type de restrictions telles que l'équivalence, la symétrie, la dis-jointure qui ne sont pas possibles dans UML. OWL fournit des fonctions importantes

complémentaires à UML et OCL (Object Constraint Language) / MOF (Meta-Object Facility) (OCL 2010) qui permettraient d'améliorer la modélisation du logiciel. Il permet de décrire des classes de plusieurs manières. Il gère ces descriptions comme des entités de première classe. Il fournit des constructions supplémentaires, comme la fermeture transitive des propriétés, et permet la classification dynamique des objets fondée sur la description des classes. Les ontologies OWL peuvent être vérifiées par des raisonneurs qui fournissent des services tels que la vérification de cohérence, la satisfiabilité des classes, la classification des concepts et des instances. Les spécifications d'OWL2 (Cuenca Grau 2008) contiennent plusieurs sous-langues ou profils (Calvanese D. 2009), qui offrent une traçabilité croissante dans l'expressivité (Pareiras F.Silva 2009).

1.8.1.4 Langages du marquage multimodal

Nous avons assisté à un développement remarquable dans le monde des technologies de l'information, notamment à travers les services fournis, qui interagissent d'une manière harmonieuse avec l'utilisateur en raison de l'émergence d'une technologie appelée l'interaction multimodale. Cette technologie fournit des solutions nécessaires aux problèmes de l'hétérogénéité des données.

L'une des technologies récentes est le standard EMMA (Extensible Multimodal Annotation Markup language) (Lalanne 2009). Celui-ci fournit un langage de représentation XML (Extensible Markup Language) basé sur le standard W3C, destiné à être utilisé par les systèmes qui fournissent des interprétations sémantiques pour une variété d'entrées. Les entrées ne sont pas nécessairement limitées à la parole, le texte en langage naturel, l'interface graphique, etc. L'utilisation de XML comme langage de représentation des entrées de l'utilisateur facilite la génération et l'analyse des documents EMMA, en utilisant des outils de traitement et d'analyse XML disponibles dans presque tous les environnements de programmation. Ceci rend inutile la création d'un parseur spécifique pour le décodage de la représentation des entrées. Il facilite également la création, la visualisation et la manipulation des fichiers de log pour les systèmes interactifs, car ils peuvent être manipulés à l'aide d'outils XML tels que XPath et XSLT (Extensible Stylesheet Language Transformation).

EMMA fournit un langage très expressif pour la représentation des entrées, à la fois aux systèmes multimodaux, unimodaux, et simplifie le "plug-and-play" des différents composants d'un système et de ses modalités (M. Johnston 2009).

Une autre technologie accessible est le SALT (Speech Application Language Tags), qui est une norme proposée pour la mise en œuvre des interfaces de langage parlé. Le noyau de SALT est une collection d'objets qui permettent à un logiciel de capter les données et de communiquer avec d'autres composants résidant sur la plate-forme sous-jacente (par exemple, gestionnaire de discours, interface téléphonique, etc.). SALT suit les normes XML, qui permettent aux extensions d'être introduites à la demande sans sacrifier la portabilité des documents. Les fonctions qui ne sont pas déjà définies dans SALT peuvent être introduites au niveau du composant, ou comme une nouvelle fonctionnalité du langage de balisage. En outre, SALT exige que la norme XML soit suivie pour que les extensions puissent être identifiées, et les méthodes qui traitent les extensions peuvent être découvertes et intégrées. En conséquence, les objets de SALT peuvent être incorporés dans une page HTML (HyperText Markup Language) ou un document XML.

L'une des technologies intéressante et prometteuse est le MuliML (Multimodal Markup Language) (Johnston 2009). Les principaux objectifs de MultiML sont la généralité et l'extensibilité. Cela signifie que l'approche doit être suffisamment générale pour qu'elle puisse être applicable dans différents contextes et différents scénarios de l'application. Le deuxième objectif est de développer un format de données qui est extensible, ce qui a plusieurs motivations (*généralement, il est impossible de développer un format de données qui prenne en* compte *la représentation de tous les énoncés de l'utilisateur*). Par conséquent, il est préférable de commencer avec la représentation des modalités choisies et des contextes, en considérant la possibilité d'étendre le format des données par la suite. D'autre part, il faut garder MultiML ouvert aux autres façons de représenter les énoncés de l'utilisateur : par exemple, si quelqu'un développe un nouveau format pour représenter les émotions humaines, il devrait être possible d'importer ce format en MultiML. Une propriété de MultiML est sa structure hiérarchique. Cela signifie que le format affiche uniquement les informations qui sont nécessaires à un moment donné. Une autre propriété importante est basée sur le concept

de sous-spécification, qui se produit lorsque les caractéristiques d'une représentation sont omises parce qu'elles sont indispensables pour le contexte actuel. Cela signifie que la représentation ne montre que les informations nécessaires et devient ainsi plus lisible et plus facile à évaluer. Ainsi, la méthode respectera les aspects de calcul lors de l'élaboration de MultiML. Cela signifie qu'il faut s'assurer que MultiML ne peut pas être traité par un ordinateur, mais aussi que les installations qui sont nécessaires pour la reconnaissance automatique des modalités soient présentes (Johnston 2009).

1.8.2 Les ontologies

Selon (Brisson 2004), le mot « ontologie » vient du grec *ontos* pour *être* et de *logos* pour *univers*. C'est un terme philosophique introduit au XIX^{e} siècle qui caractérise l'étude des êtres dans notre univers. Le mot « ontologie » possède différentes significations et demeure assez ambigu. Il y a une quinzaine d'années, la communauté de la représentation de la connaissance transforme ce concept philosophique en objet : « une ontologie ».
Une ontologie peut donc être définie comme un ensemble d'informations dans lequel sont définis les concepts utilisés dans un langage donné et qui décrit les relations logiques qu'ils entretiennent entre eux.

Le but des ontologies est donc de définir quelles primitives avec leurs sémantiques associées sont nécessaires pour la représentation des connaissances dans un contexte donné. En maintenant une représentation des notions humainement compréhensibles, l'ontologie capture l'isomorphisme entre le système symbolique et les observations du monde réel (Brisson 2004).

1.8.2.1 Ontologie formelle

Une ontologie explicite peut prendre diverses formes, mais elle comprendra impérativement un vocabulaire de termes et de spécification de leur signification. Les ontologies sont distinguées tout au long d'un spectre de formalités et se réfèrent à des degrés différents de formalité pour lesquelles un vocabulaire est créé. Dans la littérature, nous retrouvons les

ontologies dans de nombreux types basés sur différents spectres (Sowa 1984) (Guarino 1998) (Sowa 1999) (Obrst 2003). Par conséquent, le même terme « ontologie » peut être utilisé pour décrire les modèles avec différents degrés de structure.

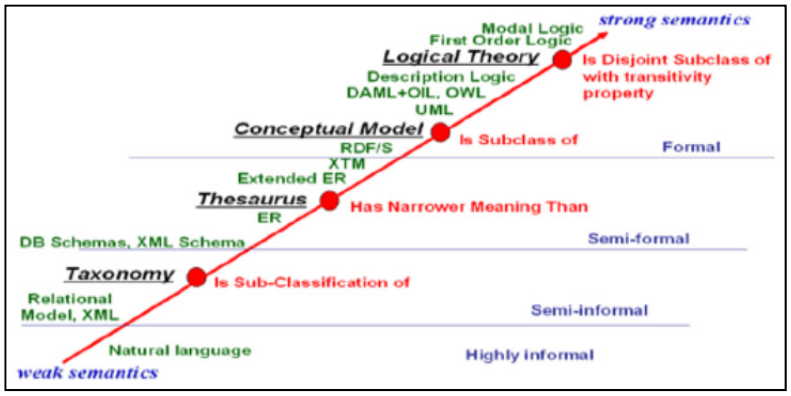


Figure 1.7 Le spectre de l'ontologie mis à jour
Tirée de Obrest (2003)

La Figure 1.7 représente le spectre suivant :

- très informel (highly informal) : vaguement exprimé en langage naturel. Une ontologie informelle contient une liste de types qui sont soit définis ou non définis seulement par le langage naturel ;

- semi-informel (semi-informal) : exprimé sous une forme restreinte et structurée du langage naturel. Dans cette zone, nous identifions une structure faible, comme les taxonomies ;

- semi-formel : exprimé dans un langage formel artificiellement défini. Cette zone comprend les éléments avec plus de structures. Nous distinguons trois parties : 1) en bas, les schémas de bases de données et les schémas de métadonnées (par exemple ICML, ebXML, WSDL), 2) en haut, les modèles conceptuels (modèles OO, UML, XML Topic maps, etc.) et 3) entre eux, les thésaurus (par exemple Wordnet, Verbnet, la PNL, OpenCyc, Lexicon) (Lenat 1990) (Panton 2006) ;

- formel (formal) : généralement exprimé dans un langage logique (par exemple, la logique du premier ordre) avec les théorèmes et les mécanismes des preuves. Les principes d'une ontologie formelle comprennent les théories axiomatiques organisées

en ordre partiel (treillis). Les théories axiomatiques ou simplement les axiomes utilisent la logique du premier ordre à cause d'une riche expressive de contraintes entre l'entité et les types de relations.

1.8.2.2 Types des ontologies

Selon le niveau de granularité dans (Guarino 1998) (Gomez Perez A. 1999), les niveaux d'abstraction des ontologies peuvent être classés en six catégories :

- ontologies de haut niveau : décrivent les concepts très généraux indépendamment du domaine comme le temps, l'espace, la matière, l'objet, l'action, etc. ;
- ontologies générales : définissent un grand nombre de concepts relatifs à la connaissance humaine fondamentale ;
- ontologies de domaines : décrivent le vocabulaire lié à un domaine spécifique comme la médecine, l'automobile, etc. ;
- ontologies de tâches : décrivent les concepts liés à l'exécution d'une tâche ou d'une activité comme le diagnostic, la vente, etc. ;
- ontologies d'application : décrivent les concepts essentiels de la planification d'une application particulière ;
- les méta-ontologies : décrivent les concepts qui sont communs à différents domaines.

1.8.2.3 Critères d'évaluation d'une ontologie

D'après (Gruber 1993) cinq critères permettent de mettre en évidence les aspects importants d'une ontologie :

- la clarté : la définition d'un concept doit faire passer le sens voulu du terme, de manière aussi *objective* que possible (indépendante du contexte). Une définition doit de plus être *complète* (c'est-à-dire définie par des conditions à la fois nécessaires et suffisantes) et documentée en langage naturel ;

- la cohérence : rien qui ne puisse être inféré de l'ontologie ne doit entrer en contradiction avec les définitions des concepts (y compris celles qui sont exprimées en langage naturel) ;
- l'extensibilité : les extensions qui peuvent être ajoutées à l'ontologie doivent être anticipées. Il doit être possible d'ajouter de nouveaux concepts sans avoir à toucher aux fondations de l'ontologie ;
- une déformation d'encodage minimale : une déformation d'encodage a lieu lorsque la spécification influe sur la conceptualisation. Un concept donné peut être plus simple à définir d'une certaine façon pour un langage d'ontologie donné. Bien que cette définition ne corresponde pas exactement au sens initial. Ces déformations doivent être évitées autant que possible ;
- un engagement ontologique minimal : le but d'une ontologie est de définir un vocabulaire pour décrire un domaine, si possible de manière *complète* ; ni plus, ni moins. Contrairement aux bases de connaissances, il n'est pas requis d'une ontologie qu'elle soit en mesure de fournir systématiquement une réponse à une question arbitraire sur le domaine. Toujours selon Gruber, « l'engagement ontologique peut être minimisé en spécifiant la théorie la plus faible (celle permettant le plus de modèles) couvrant un domaine ; elle ne définit que les termes nécessaires pour partager les connaissances consistantes avec cette théorie ».

1.8.2.4 Utilisations des ontologies

Les ontologies sont utilisées pour :

- la communication : le partage des connaissances pour permettre la communication, l'interaction et la coopération ;
- l'interopérabilité : capacité des systèmes à coopérer dans un but commun ;

- la modélisation : spécification et conceptualisation. En utilisant une ontologie pour relier les connaissances déclaratives (connaissance du domaine) et de procédure (résolution de problèmes) ;

- la fiabilité : la fiabilité définit la capacité d'un système ou des composants de s'acquitter de leurs fonctions requises dans des conditions déterminées. Nous pouvons différencier entre les rôles que pourraient jouer l'ontologie pour la fiabilité des logiciels systèmes ;

- la réutilisabilité : la réalisation de la réutilisation dépend en grande partie du partage d'une conceptualisation similaire. En caractérisant les classes de domaines et des tâches de ces domaines, les ontologies fournissent un « Framework » pour déterminer quels aspects d'un système sont réutilisables entre les différents domaines et les tâches. Ainsi, l'ontologie permet d'importer et d'exporter des modules et des composants entre leurs systèmes.

Les solutions à base d'ontologies sont efficaces, en particulier pour :

- assurer l'interopérabilité ou la coopération sémantique et la description des relations entre différents concepts ;

- résoudre le problème d'ambiguïté dans les commandes d'un utilisateur, en utilisant les sens communs des connaissances, afin de comprendre les intentions d'un utilisateur (Barboni E. 2006).

1.8.2.5 Langage des règles pour les ontologies

SWRL (Semantic Web Rule Language) (SWRL 2004) est un langage de règles pour le Web sémantique, combinant le langage OWL-DL et le langage RuleML (Rule Markup Language) (www.ruleml.org).

En comparaison avec DLP (Description Logic Programs), une autre proposition relativement récente de la communauté Web sémantique, permettant d'intégrer des règles et OWL, SWRL prend une approche d'intégration diamétralement opposée. DLP est l'intersection de la

logique de Horn et d'OWL, tandis que SWRL est (approximativement) l'union des deux. Pour DLP, le langage résultant est une logique descriptive d'une forme inhabituelle et peu expressive. Au contraire, SWRL garde la puissance d'OWL-DL, mais au prix de la décidabilité et des implémentations concrètes (Parsia 2005).

SWRL est caractérisé par :

- recommandé par le standard W3C en 2004 (SWRL 2004) ;
- règles enregistrées comme partie de l'ontologie ;
- accroître le soutien des outils : Bossam, R2ML, Hoolet, Pellet, KAON2, RacerPro, SWRLTab ;
- fonctionnement avec les raisonneurs.

Un simple exemple d'une règle SWRL est le suivant :

Personne (Fred) ^ aCommeMembreDeFamille (Fred, ?s) ^ Homme (?s) → aCommeFrère (Fred, ?s)

Cette règle explique qu'il y a une personne (classe dans l'ontologie) qui s'appelle Fred (instance de la classe personne dans l'ontologie), et Fred a comme membre de la famille (aCommeMembreDeFamille est une propriété d'objet dans l'ontologie) une variable S, le S est un homme (classe dans l'ontologie). Donc Fred est le frère de S (aCommeFrère) est une propriété d'objet dans l'ontologie.

1.8.2.6 Langage de requêtes pour les ontologies

- **SPARQL** (SPARQL Protocol and RDF Query Language) (Toby Segaran 2009) est un langage de requête et un protocole qui permet de rechercher, d'ajouter, de modifier ou de supprimer des données RDF disponibles à travers Internet.

  SPARQL accède aux données du Web. Cela signifie qu'en théorie, nous pouvons accéder à toutes les données du Web avec ce standard. L'ambition du W3C est d'offrir une interopérabilité non pas seulement au niveau des services, comme avec les

services Web, mais aussi au niveau des données, structurées ou non, qui sont disponibles à travers Internet.

Ce standard a été créé par le groupe de travail DAWG (Data Access Working Group) du W3C. SPARQL est considéré comme l'une des technologies clés du Web sémantique.

SPARQL est adapté à la structure spécifique des graphes RDF et s'appuie sur les triplets qui les constituent.

SPARQL permet d'exprimer des requêtes interrogatives ou constructives :

- une requête SELECT, de type interrogative, permet d'extraire du graphe RDF un sous-graphe correspondant à un ensemble de ressources vérifiant les conditions définies dans une clause WHERE ;
- une requête CONSTRUCT, de type constructive, engendre un nouveau graphe qui complète le graphe interrogé.

Par exemple, sur un graphe RDF contenant des informations généalogiques, nous pouvons par une requête SELECT trouver les parents ou grands-parents d'une personne donnée. Similairement, avec des requêtes CONSTRUCT, nous pouvons ajouter des relations frère-sœur, cousin-cousine, oncle-neveu, qui ne seraient pas explicitement déclarées dans le graphe initial.

- **SQWRL** (Semantic Query Language rule-enhanced Web) (Martin O'Connor 2009) est basé sur le langage de règles SWRL. SQWRL prend une règle standard SWRL et la traite efficacement comme une spécification du modèle pour une requête. Il remplace la règle conséquente avec une spécification de récupération. Il définit un ensemble d'opérateurs qui peuvent être utilisés pour construire des spécifications de récupération. L'avantage de cette approche est qu'il n'y a aucune extension syntaxique de SWRL. Ainsi, les éditeurs SWRL existants peuvent être utilisés pour générer et modifier des requêtes SQWRL. En outre, le mécanisme standard de sérialisation de SWRL peut être utilisé afin que les requêtes SQWRL puissent être stockées dans des ontologies OWL.
  Un simple exemple d'une requête SQWRL est le suivant :

Personne(?p) ^ aCommeAge(?p, ?a) ^ swrlb:lessThan(?a, 9) → sqwrl:select(?p, ?a)
Cette requête donne les personnes et leur âge à condition qu'ils aient moins de 9 ans.

1.8.2.7 Raisonneurs pour les ontologies

Avec les langages RDF et OWL, un support de raisonnement est fourni en définissant des hiérarchies de classes ainsi que des relations entre celles-ci et entre des propriétés. Il est par exemple possible de définir une propriété comme étant symétrique. De telles définitions consistent en des règles d'inférence pouvant être appliquées par des logiciels appelés raisonneurs à base de connaissances afin d'inférer automatiquement de la connaissance. Les principaux services d'inférence proposés par les raisonneurs sont repris ci-dessous :

Vérification de la consistance

Consiste à s'assurer de la consistance d'une base de connaissances, c.-à-d. de vérifier qu'il n'y a pas de faits contradictoires dans la base de connaissances, tels qu'une assertion dans l'ABox (Calvanese, Giacomo *et al.* 2001) qui soit contraire aux définitions présentes dans la TBox (Calvanese, Giacomo *et al.* 2001). Ainsi, si la TBox définit une propriété *hasChild* comme devant avoir un individu de la classe *Person* pour sujet et objet, une inconsistance apparaît si l'ABox contient une assertion telle que *hasChild* (Fred, rouge) où Fred est bien une personne mais où rouge n'est pas une instance de la classe *Person*.

Satisfaisabilité de concept

Consiste à déterminer si une classe peut posséder une instance. Une classe insatisfaisante entraîne une inconsistance si une instance d'une telle classe est définie. Un exemple de ce type de classe est une classe définie comme l'intersection d'une classe Femme et d'une classe Père, étant donné qu'une personne ne peut être à la fois une femme et un père.

Classification

Consiste à établir les liens hiérarchiques entre toutes les classes d'une ontologie, c'est-à-dire identifier les différentes sous-classes de classes. Ceci permet par exemple de fournir toutes les sous-classes d'une classe donnée.

Réalisation

Consiste à établir la classe la plus spécifique d'un individu, ce qui nécessite d'avoir réalisé une classification auparavant. Par exemple, s'il existe une classe Personne et une classe Femme qui est une sous-classe de la classe Personne, la classe la plus spécifique d'une femme sera la classe Femme et non la classe Personne qui est plus générale.

Parmi les raisonneurs existants, citons RacerPro (KG 2007), FaCt++ (Dmitry Tsarkov 2006) et Pellet (Evren Sirin 2007). Ce dernier, écrit dans le langage Java, est un raisonneur OWL-DL complet, c.-à-d. qu'il supporte toute l'expressivité du langage OWL-DL. Il a de plus été étendu afin de supporter certaines propriétés apportées par OWL2, entre autres les assertions de négation de propriétés, les propriétés disjointes et la capacité qu'un individu et une classe puissent partager la même URI (l'URI dépend de la création de l'ontologie).

1.9 Systèmes et architectures multimodaux existants

1.9.1 Smartkom

L'objectif de SmartKom (Engel 2006) est de développer une interface de dialogue multimodale uniforme pour de nombreuses applications, allant du contrôle électronique en public aux services mobiles. L'hypothèse de ce projet est que l'utilisateur obtient un ensemble de services grâce à un agent d'interaction personnalisé, appelé Smartakus. Les modalités utilisées dans SmartKom sont la parole et les gestes.

Le traitement est basé sur le M3L (MultiModal Markup Language), qui est formulé comme un ensemble de schémas XML (Herzog and P. 2003)**.** Le schéma qui décrit à la fois les intentions de l'utilisateur et du système est défini dans une ontologie, en utilisant l'OIL (Ontology Inference Layer ou Ontology Interchange Language) (http://www.ontoknowledge.org/index.shtml 2003) comme framework pour la représentation des notations. La base de connaissances basée sur OIL est automatiquement transformée en un schéma XML (Gurevych 2003)**.** L'ontologie comprend plus de 700 concepts et 200 relations, qui décrivent les objets abstraits nécessaires pour communiquer

avec l'ensemble des fonctionnalités. Le raisonnement appliqué est le "Closed World", de façon à ce que tout ce qui peut être parlé entre l'utilisateur et le système soit codé dans l'ontologie.

1.9.2 ICARE

La plateforme ICARE (Complementarity Assignment Redundancy Equivalence) (Bouchet 2004) permet aux concepteurs graphiques de manipuler et d'assembler les modules logiciels d'ICARE afin de préciser l'interaction multimodale dédiée à une tâche donnée d'un système interactif en cours de développement. Ainsi, cette plateforme est dirigée vers les concepteurs et non vers les développeurs. Il existe deux types de composants ICARE :

- les composants élémentaires qui permettent au concepteur de définir des « *modalités pures* » (Bernsen 1994) ;
- les composants de composition (ou combinaison) qui permettent au concepteur de spécifier l'usage combiné de modalités. Les composants de composition sont indépendants des modalités à composer.

Le composant élémentaire est encore formé par deux types de composants : *Dispositif* et *Langage d'interaction*. Un composant *Dispositif* représente une couche supplémentaire du pilote d'un dispositif physique. Il s'agit du niveau physique d'une modalité. Tous les composants *Dispositifs* d'ICARE enrichissent aussi les données brutes du pilote en y ajoutant des informations comme l'état de marche du dispositif, une estampille de temps, un facteur de confiance des données produites et une description en termes de manipulation du dispositif par l'utilisateur (par exemple, s'il s'agit d'une modalité active ou passive ainsi que le lieu d'interaction lié au dispositif). Le composant *Langage d'interaction* correspond au niveau logique d'une modalité d'interaction. Par exemple, un composant *Langage d'interaction* peut abstraire les données d'un composant *Dispositif* Souris en une commande correspondant à la sélection dans un menu d'options. Pour la composition, quatre composants ont été définis, permettant de combiner les données de 2 à n composants : un pour la Complémentarité, un pour la Redondance, un pour l'Équivalence et un pour la Redondance/Équivalence. L'assignation n'est pas explicite dans la spécification ICARE, car

elle est représentée par un simple lien entre deux composants. En effet, un composant A lié à un composant B implique que A est assigné à B.

1.9.3 SmartWeb

SmartWeb (Barboni E. 2006) réunit l'interface multimodale et la technologie de réponse aux questions. Pour atteindre l'objectif de répondre correctement aux questions, une combinaison des différents types d'ontologies de domaine dans une base de connaissances a été définie. L'ontologie définie est basée sur le modèle supérieur SUMO (Suggested Upper Merged Ontology) (www.ontologyportal.org) et DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering) (www.loa-cnr.it/DOLCE.html). Le modèle supérieur ontologique de SmartWeb propose essentiellement les concepts de niveau supérieur de DOLCE et l'adoption de plusieurs concepts de SUMO, en ajoutant des taxonomies qui servent aux réponses de type sémantique. SmartWeb utilise une ontologie de discours appelée DISCONTO basée sur la technique d'EMMA pour décrire les interactions multimodales.

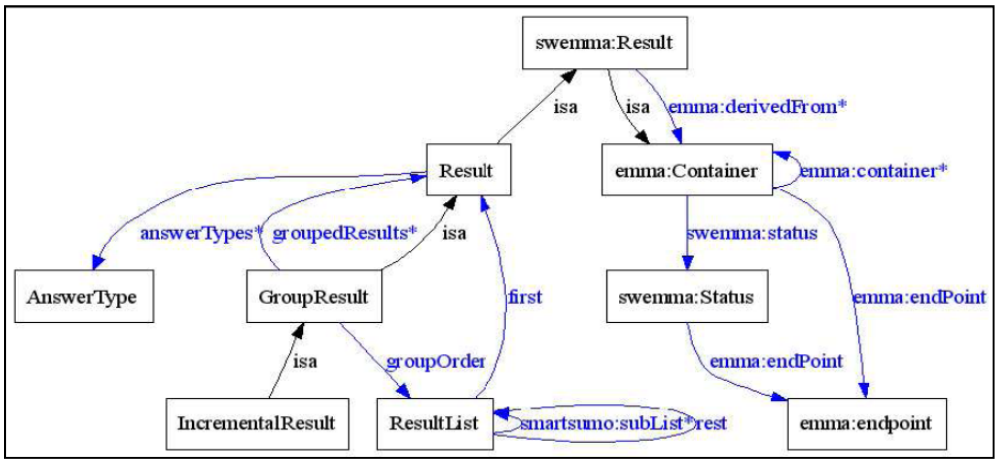


Figure 1.8 Représentation graphique de swemma
(Barboni E. 2006)

La Figure 1.8 montre une représentation ontologique du résultat multimodal. Un nouveau nom d'espace *swemma* a été défini pour l'extension d'*Emma: contanier*. Une de ces extensions est la classe *swemma: Result*, étendue par *GroupResult* et *IncrementalResult*. Pour s'assurer d'une couverture complète de toutes les formes de réponses et être au niveau des

besoins particuliers, une typologie a été développée. Les résultats obtenus sont représentés par une ontologie appelée SMARTMEDIA.

La Figure 1.9 ci-dessous montre comment les trois niveaux ontologiques (SWEMMA, DISCONTO, SMARTMEDIA) utilisés dans SmartWeb traitent la question "Quand l'Allemagne a gagné le Championnat du monde pour le football?". *GroupResult* regroupe plusieurs résultats dans une structure liée.

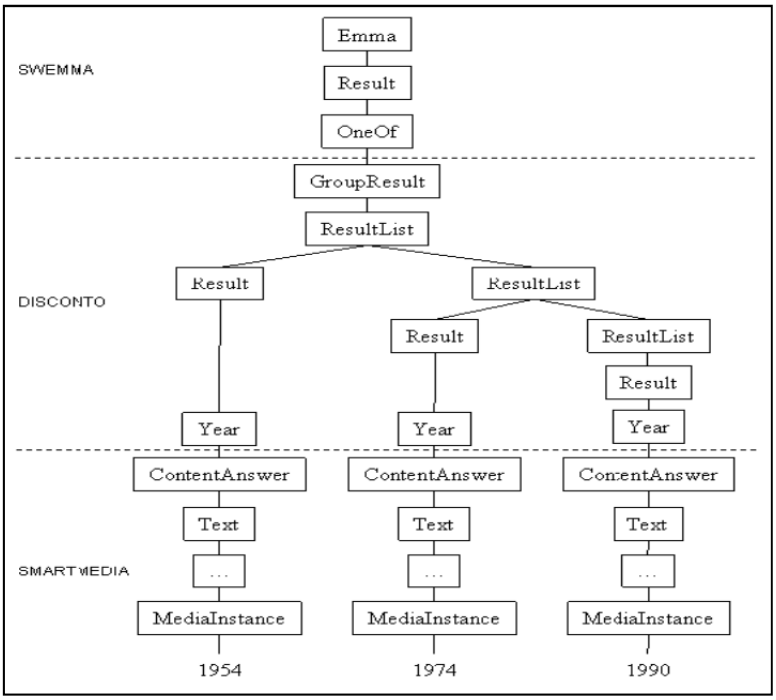


Figure 1.9 Représentation graphique des trois niveaux ontologiques pour la question «Quand l'Allemagne a-t-elle gagné le championnat du monde pour le football ?»
Tirée de (Barboni E. 2006)

1.9.4 HEPHAISTK

HephaisTK (Bruno Dumas 2008) est une boîte à outils écrite entièrement en langage Java et basée sur un système multi-agents appelé JADE (Java Agent Development Framework). Le choix de ce type d'architecture logicielle permet de pouvoir distribuer HephaisTK sur une série de plateformes. Toute source d'entrée est surveillée par un agent défini, qui se charge de

recueillir les données de la source, de les convertir en un type générique si besoin et de les transmettre au reste de la communauté logicielle. Cet agent contient également une description des capacités de la source et peut en informer tout autre agent de manière dynamique. Un agent nommé « postman » se charge de recueillir toutes les données entrantes provenant des différentes sources. Il les stocke ensuite dans une base de données, les laissant à la disposition du reste de la boîte à outils. Ce « postier » permet à d'autres agents de s'inscrire à une certaine catégorie de messages, puis leur renverra automatiquement tout message reçu de cette catégorie. L'atout de cet agent est de servir de générateur unique de « timestamp », évitant de devoir considérer de potentiels problèmes de désynchronisation entre les sources d'entrée.

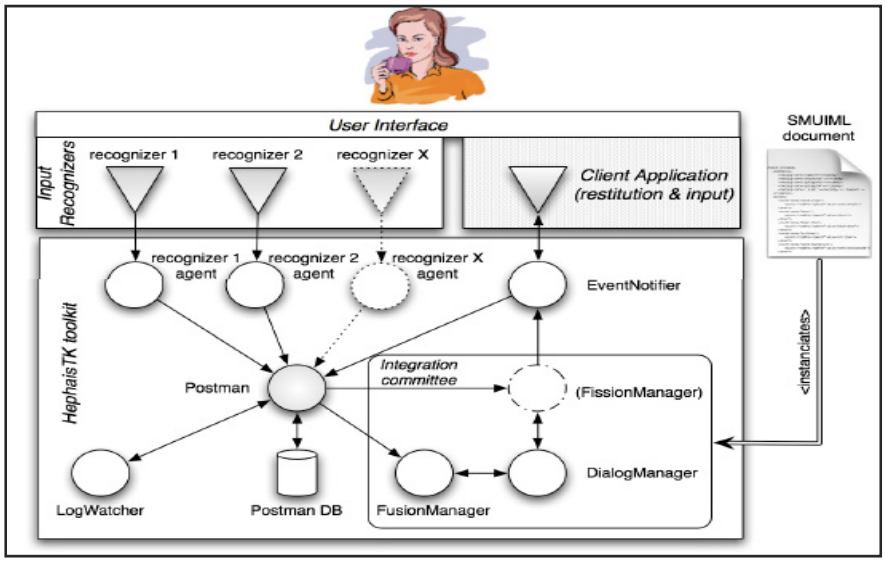


Figure 1.10 Architecture de HEPHAIST
Tiré de Bruno Dumas (2008)

Les agents gérant le dialogue et la fusion sont paramétrés à l'aide d'un script écrit en un langage XML appelé SMUIML (Synchronized Multimodal User Interfaces Modeling Language) (Dumas 2008). Lorsqu'une application cliente instancie HephaisTK, elle doit notamment spécifier l'emplacement du script qui lui est rattaché. Ce script SMUIML permet de définir le dialogue humain/machine multimodal sur trois niveaux différents : 1) une description du dialogue humain/machine proprement dit, de haut niveau, en ayant la possibilité de définir des entités réutilisables, 2) une description de l'ensemble des

événements d'entrée et de sortie qui vont potentiellement intervenir dans le dialogue décrit précédemment, selon leur modalité et 3) une description bas niveau des reconnaisseurs et sources d'entrée utilisés, ainsi que leur éventuelle configuration. Il est à noter que plusieurs reconnaisseurs différents peuvent être disponibles pour une même modalité. Le langage de script permet la spécification précise de la synchronisation des modalités par la prise en compte explicite des propriétés CARE (Complementarity, Assignment, Redundancy, Equivalence). Le choix de se concentrer en premier lieu sur la définition d'un langage type XML permettant de décrire l'interaction multimodale a été fait d'abord afin d'avoir une base formelle. Celle-ci permet l'étude de l'expressivité nécessaire à la description d'une interface multimodale ; ensuite, pour obtenir un formalisme qui soit à la fois humainement compréhensible et facile à traiter pour la machine. Enfin, si pour l'instant ces fichiers de configuration SMUIML sont écrits par le développeur, leur organisation devrait permettre à terme leur génération à l'aide d'une interface graphique.

Le Tableau 1.4 présente d'autres systèmes multimodaux qui ont été développé suivant BRETAM (Melichar 2006), en prenant compte de la technique de fusion utilisée, les types des modalités et la résolution de l'ambigüité. Ce tableau est une synthèse qui montre l'évolution du travail sur la fusion multimodale et en même temps, les limites des applications développés, que ce soit par la méthode de fusion utilisée, le nombre de modalités ou bien le domaine spécialisé d'un système de fusion multimodale.

Tableau 1.4 Systèmes multimodaux et techniques de fusion
Tiré de Lalanne, Nigay et al. (2009)

Phases	**Reference**	**Tool language program**	**Fusion**				**Application Types**
			Notation	**Type**	**Input Devices**	**Ambiguity Resolution**	
B	(Bolt 1980)	Put that here system	None	None	Speech gesture	None	Map manipulation
R	(Neal 1989)	Cubricon	Generalized Augmented Transition Network	Procedural	Speech Mouse Keyboard	Proximity-based	Map manipulation
E	(Koons 1993)	No name	Parse tree	Frame based	Speech, Eye gaze, Gesture	First solution	3D World
	(Nigay 1993)	Pac Amodeus	Melting Pot	Frame based	Speech, Keyboard,	Context-based	Flight Scheduling

P h	Reference	Tool language	Fusion				Application Types
					Mouse	resolution	
	(Cohen 1997)	Quickset	Feature Structure	Unification	Pen Voice	S / G & G / S & N best	Simulation System training
	(Bellik 1997)	MEDITOR	None	Frame-based	Speech Mouse	History Buffer	Text Editor
	(Martin 1998)	TYCOON	Set of processes Guided Propagation Networks	Procedural	Speech Keyboard Mouse	Probability based resolution	Edition of graphical user interfaces
	(Duarte 2005)	FST	Finite State Automata	Procedural	Speech pen	Possible (N best)	Corporate Directory
T & A	(Krahnstoever 2002)	iMap	Stream Stamped	Frame based	Speech gesture	Not given	Crisis Management
	(Dumas, Lalanne et al. 2009)	HephaisTK	XML Typed (SMUIML)	Frame based	Speech Mouse Phidgets	First one	Meeting assistants
	(Holzapfel 2004)	No Name	Typed Feature Structure	Unification	Speech gesture	N Best list	Humanoid Robot
	(Engel 2006)	PATE	XML Typed	Unification	Speech pen	N Best list	Bathroom design Tool
	(Milota 2004)	No Name	Multimodal Parse Tree	Unification	Speech Mouse keyboard Touch screen	S / G & G /S	Graphic Design
	(Melichar 2006)	WCI	Multimodal Generic Dialog Node	Unification	Speech Mouse Keyboard	First One	Multimedia DB
	(Sun 2006)	PUMPP	Matrix	Unification	Speech gesture	S / G	Traffic Control
	(Bourguet 2002)	Mengine	Finite State machine	Procedural	Speech Mouse	Not given	No example
	(Latoschik 2002)	No Name	Temporal Augmented Transiti-on Network	Procedural	Speech gesture	Fuzzy constraint	Virtual reality
	(Bouchet 2004; Bouchet 2004) (Mansoux 2006)	ICARE (Input/ Output)	Melting pot	Melting pot	Speech, Helmet visor HOTAS, Tactile surface, GPS localization, Magnet-ometer, Mouse, Keyboard	Context-based resolution	Aircraft Cockpit, Authentication, Mobile Augmented Reality systems (Game, Post-it), Augmented Surgery
	(Navarre 2005)	Petshop	Petri nets	Petri nets	Speech mouse Keyboard Touch screen	***	Aircraft Cockpit
	(Flippo 2003)	No Name	Semantic tree	Semantic tree	Speech Mouse Gaze gesture	Feedback for missing data	Collabora-tive Map
	(Portillo 2006)	MIMUS	Feature Value Structure (DTAC)	Feature Value Structure (DTAC)	Speech Mouse	Knowledge-able agent	No example

P h	Reference	Tool language	Fusion				Application Types
	(Carlos Duarte 2006)	FAME	Behavioral Matrix	Behavioral Matrix	Speech Mouse Keyboard	Not given	Digital talking Book

1.10 Synthèse et conclusion

Différentes techniques ont été employées pour réaliser la fusion multimodale au fil des années. Au début, ces techniques n'ont pas traité vraiment la notion de fusion ; mais, en passant par les différentes phases et en arrivant à la phase de la maturité, la notion de fusion a été effectivement élaborée. Dans ce chapitre, nous avons présenté l'évolution des moteurs de fusion et des systèmes multimodaux et nous avons interprété les techniques, les architectures et les systèmes qui existent réellement. Comme il a été clairement montré dans ce chapitre, les architectures disponibles dans la littérature sont très spécialisées à cause d'une conception limitée à des systèmes spécifiques, comme dans SmartKom, qui propose une solution multimodale pour certains cas (TV : on/off et sélection des chaînes, fax, téléphone, etc.). De plus, le calcul probabiliste et les équations mathématiques ont été utilisés comme méthodes de fusionnement ; et malgré leur efficacité, parfois, ils restent compliqués et difficiles à faire évoluer.

Des systèmes comme SmartWeb et HephaisTK présentés dans la littérature ont proposé des solutions basées sur les ontologies. Encore une fois, ces solutions restent insuffisantes à cause de leur dépendance à des ontologies de haut niveau (SUMO, DOLCE), qui restent difficiles à interpréter.

Plusieurs méthodes de fusion multimodale ont été proposées ont montré une efficacité dans leurs domaine d'application. Cependant, notre travail ne s'intéresse pas à la fusion de bas niveau (au niveau des capteurs) mais à la fusion de haut niveau, telle que la fusion des événements provenant des modalités suivant des règles de fusion bien définies, d'où notre intérêt à la méthode des règles et plus spécifiquement, la méthode des règles personnalisée. Cette dernière permet l'ajout des règles selon les perspectives d'une application et selon le domaine de connaissance. Notre travail contribue ainsi dans la fusion multimodale en

proposant une modélisation sémantique de l'environnement et la définition d'un mécanisme de sélection automatique des modalités et des modèles de fusion.

Dans ce chapitre, l'étude de ces théories a été très intéressante pour notre travail. Cela nous aidera à proposer notre propre définition d'une architecture de fusion multimodale. L'état de l'art présenté a été nécessaire pour comprendre toutes les bases requises : l'interaction multimodale, la définition du processus de fusion que nous voulons mettre en œuvre, la définition des composants de raisonnement capables de comprendre l'environnement, les langages de la représentation des connaissances, de la description et de la logique afin de modéliser l'architecture et les systèmes présents dans l'environnement d'une manière sémantique et plus naturelle. Nous avons exposé plusieurs architectures d'interaction existantes et nous avons terminé par la présentation d'un survol de plusieurs plateformes de développement utiles pour mettre en œuvre notre architecture. Dans le chapitre suivant, nous préciserons nos objectifs, notre méthodologie, nos contributions et mettrons en évidence l'originalité de nos travaux.

CHAPITRE 2

BUT, OBJECTIFS ET MÉTHODOLOGIE DE RECHERCHE

2.1 Introduction

Ce chapitre présente le but principal de notre projet de recherche et les objectifs choisis pour atteindre ce but. Une section intitulée "Originalité des travaux envisagés" montre que les travaux de ce projet s'inscrivent dans l'innovation pour la modélisation d'une architecture d'interaction multimodale et expose l'intérêt des résultats de ces travaux.

Ce chapitre présente également les différentes phases de la méthodologie de recherche nécessaires pour atteindre le but final, qui consiste en la création d'un système de fusion multimodal.

2.2 Problématique de recherche

Depuis le travail de Bolt (Bolt 1980), le monde académique a fourni des prototypes et des systèmes offrant des techniques d'interaction multimodale. Actuellement, l'ingénierie des systèmes interactifs multimodaux reste toujours complexe et encombrante au niveau de la modélisation, de la spécification, de la validation et de l'implémentation. Cela nous conduit à nous poser de nombreuses questions :

- quels sont les modules nécessaires et comment les faire communiquer (architecture) ?
- comment représenter les données fournies par chaque modalité (langage de représentation) ?
- comment fusionner et interpréter les informations fournies par chaque modalité (algorithme de fusion) ?
- comment prendre en compte les contextes de l'environnement et de l'utilisateur pendant la conception et lors de l'interaction ?

- comment savoir quand une commande multimodale commence et quand elle finit (conditions temporelles) ?
- comment peut-on faciliter le travail de fusionnement et comment atteindre ce but ?

Dans ce cadre-là, et en partant des questions posées précédemment, nous proposons une solution qui consiste à développer une architecture pour l'interaction multimodale. L'originalité de ce travail consiste à faire la description sémantique de l'environnement et la définition d'un mécanisme de sélection automatique des modalités et des modèles de fusion. Nous utiliserons, comme dans la littérature, des technologies telles que les ontologies, les langages du marquage extensible, etc. Notre approche sera faite par la définition d'un moteur responsable de la fusion d'événements dont l'information provient de l'environnement, c.-à-d. les commandes provenant des utilisateurs ou bien du lieu où se trouve le système d'interaction multimodal. Ils seront décrits dans des fichiers XML ou EMMA. La fusion doit suivre une série de pré-conditions, afin d'aboutir à une meilleure compréhension de la requête et à un fusionnement des événements provenant de l'environnement.

2.3 Buts et objectifs de recherche

Le moteur de fusion est un composant fondamental dans un système d'interaction multimodal. Son rôle fondamental est l'interprétation du flux d'entrée. Nous nous concentrons dans ce qui suit sur la conception, la spécification, la construction et l'évaluation d'un moteur de fusion.

Nous développons dans ce projet de recherche un système expert pour l'interaction multimodale personne-machine, en utilisant des technologies comme les ontologies, les langages de règles sémantiques, les langages de marquage, etc. Le but est de spécifier un composant de fusion pour l'interaction multimodale. Ce composant doit vérifier les événements provenant de l'environnement par rapport à des pré-conditions définies et des modèles créés dans l'ontologie afin d'aboutir à une meilleure compréhension de l'entourage et d'avoir les meilleures solutions, en utilisant les standards de W3C, qui fournit un langage de représentation formel comme OWL et XML.

Les paradigmes développés seront mis en œuvre pour concevoir un outil de surveillance pour personnes âgées, afin de leur assurer une aide, à domicile comme à l'extérieur.

Afin d'arriver à nos fins, nous commençons par définir nos objectifs comme suit :

- définition de l'architecture générale. Cela nécessite :
    - la description de l'environnement par la création d'une ontologie ;
    - la précision des différents contextes qui affectent la sélection des modalités ;
    - la sélection des modalités ;
    - la proposition des modèles de fusion ;
    - la création du moteur de fusion ;
- validation de l'architecture ;
- développement d'un outil de surveillance pour personnes âgées.

2.4 Méthodologie de recherche

Pour atteindre les objectifs de ce projet de recherche, plusieurs étapes de recherche doivent être franchies. La méthodologie de recherche proposée ci-dessous présente les phases et les objectifs nécessaires pour la réalisation de chacune d'elles. Chaque phase comprend au moins un processus, des entrées et des sorties. La Figure 2.1 montre les phases de la méthodologie de recherche.

2.4.1 Phase 1 : Exploration

Le processus de cette phase consiste à analyser les documents de la littérature et de synthétiser le contenu des publications dans le but de trier et d'utiliser les travaux utiles pour notre projet de recherche.

Le principal livrable produit à l'issue de ce processus représente les objectifs de la recherche. Dans ce livrable, nous trouvons la synthèse de toutes les étapes à suivre pour atteindre le but principal de la thèse, la synthèse de tous les travaux réalisés dans le domaine ainsi que les publications et les propositions de démarche qui peuvent éventuellement être utiles pour notre projet de recherche.

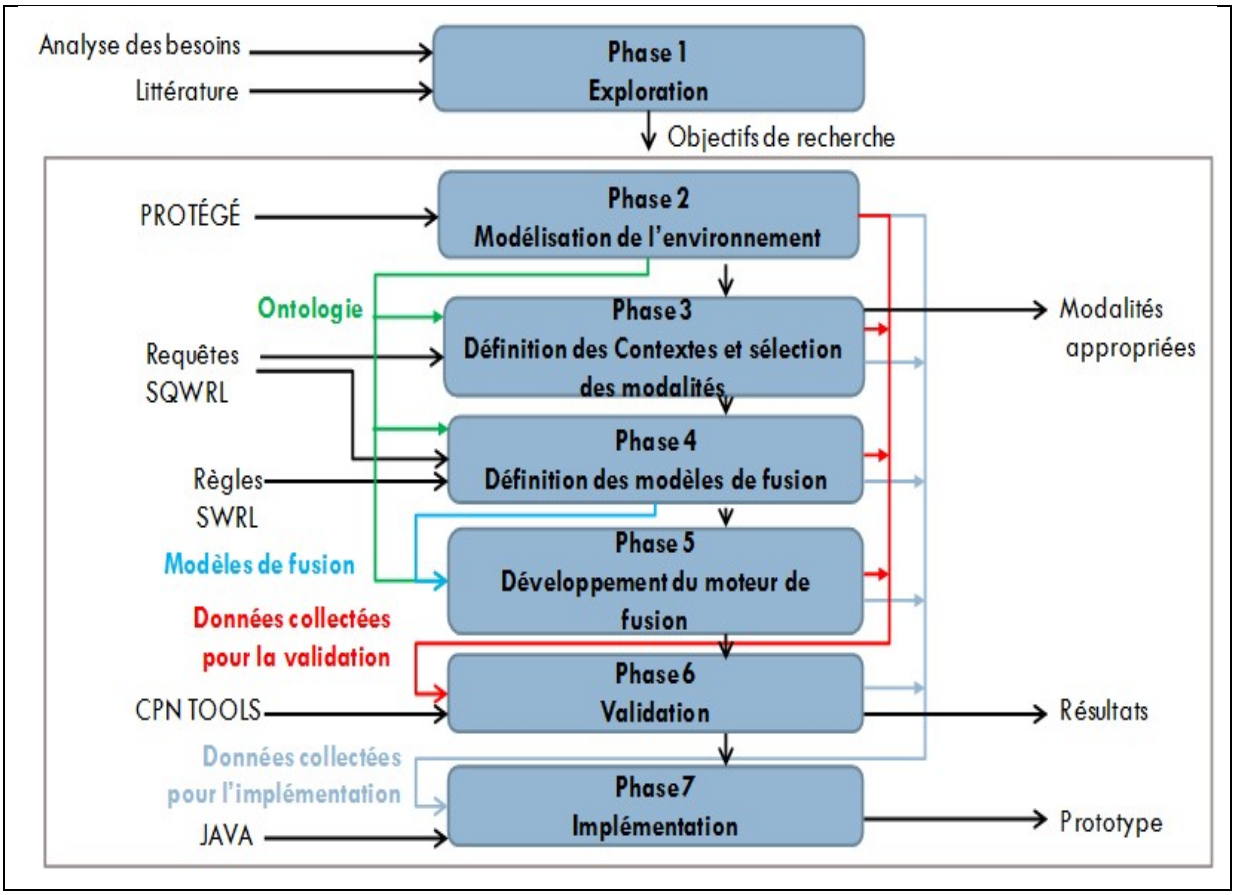


Figure 2.1 Les phases de la méthodologie de recherche

2.4.2 Phase 2 : Modélisation de l'environnement

Dans cette phase, nous avons modélisé notre environnement sous forme d'une ontologie. L''environnement est l'endroit où se trouve le système et l'utilisateur, il peut être à l'intérieur, comme une maison, une chambre, etc., ou à l'extérieur. L'ontologie décrit :

- les objets qui peuvent être trouvés dans l'environnement ;

- les contextes de l'environnement, de l'utilisateur et de localisation;
- les relations sémantiques entre les différents objets ;
- le profil de l'utilisateur ;
- les relations sémantiques entre le profil de l'utilisateur et le contexte de l'utilisateur ;
- les modalités et leurs relations sémantiques avec les contextes;
- un vocabulaire sous forme d'instances ;
- les modèles de fusion.

Cette ontologie a été réalisée à l'aide de l'outil PROTÉGÉ (www.PROTÉGÉ.stanford.edu) (la flèche d'entrée de la phase2 dans la Figure 2.1). Une explication détaillée de tous ces points sera vue dans le chapitre 4.

2.4.3 Phase 3 : Définition des Contextes et sélection des modalités

Dans cette phase, nous avons étudié les effets qui peuvent affecter le fonctionnement des modalités. Trois types de contextes ont été identifiés :

- contexte environnemental ;
- contexte de l'utilisateur ;
- contexte de la localisation.

Dans l'ontologie, nous avons défini ces contextes et leurs relations avec les modalités en créant des instances et des propriétés d'objets et de données.

Pour savoir quelle modalité est affectée par un contexte spécifique, nous avons besoin du langage de requête SQWRL. Avec ces requêtes, nous pouvons savoir si une modalité est activée ou non. Des détails sur cette phase sont prévus dans le chapitre 4.

2.4.4 Phase 4 : Définition des modèles de fusion

Nous avons défini une trentaine de modèles dans l'ontologie. Chaque modèle est une combinaison différente d'évènements qui respectent un ordre spécifique. Cette phase assure une meilleure compréhension d'un ensemble d'évènements, car s'ils sont en ordre, cela peut influencer le sens d'une commande formée par plusieurs évènements.

Pour faciliter le fonctionnement des modèles, un vocabulaire a été créé sous forme d'instances dans l'ontologie. Des propriétés d'objets ont été créées entre ces instances pour savoir à quel modèle appartient une instance donnée et l'ordre correspondant.

Pour savoir quel est le modèle approprié à un ensemble d'évènements, nous avons besoin de requêtes SQWRL et des règles SWRL, qui sont des règles sémantiques. Celles-ci assurent une inférence de la connaissance dans l'ontologie. Les détails sont décrits dans le chapitre 4.

2.4.5 Phase 5 : Développement du moteur de fusion

Dans cette phase, nous avons défini une série des conditions qui doivent être vérifiés. Ces conditions sont utilisées pour fusionner les évènements provenant des modalités. Un algorithme de fusion a été défini qui prend en considération ces conditions à tester avant la fusion. Si l'une des conditions n'est pas respectée, alors il n'y aura pas de fusion. Les détails sont donnés dans le chapitre suivant.

2.4.6 Phase 6 : Validation

Dans cette phase, une validation de la fonctionnalité de l'architecture a été réalisée, en modélisant les différents composants de celle-ci dans un réseau de Petri coloré stochastique. Nous avons utilisé l'outil CPN TOOLS (University of Aarhus 2012). Cet outil permet de faire des simulations et vérifier les deadlock dans chaque composant à part, pour savoir si l'architecture est fonctionnelle. Nous avons défini dans le réseau les différents composants de notre architecture et nous avons validé le fonctionnement de l'algorithme de la fusion et de la sélection des modalités. De plus, nous avons créé un générateur aléatoire d'évènements pour

les simulations. Celui-ci génère des signaux selon les modalités voulues. Les détails de la validation et les résultats obtenus sont présentés dans le chapitre 5.

2.4.7 Phase 7 : Implémentation

Dans cette phase, un prototype est développé en java en utilisant OWL API, nécessaire pour l'implémentation des ontologies. Cette application est dédiée au grand public en général et pour les personnes âgées et/ou handicapées en particulier. L'outil devrait effectivement faciliter certaines manipulations pour les personnes atteintes de certains handicaps moteurs par exemple. Le prototype inclut une interface graphique assez simple et facile à utiliser. Les détails sur le prototype sont prévus dans le chapitre 6.

2.5 Originalité des travaux proposés

D'après la revue de littérature présentée dans le premier chapitre, nous avons constaté la richesse et la variété des études menées dans le domaine de la fusion multimodale. Notre projet est une contribution à travers laquelle nous visons à accentuer l'aspect applicatif des moteurs de fusion, par le développement d'un moteur qui peut être utilisé dans plusieurs types d'applications. Pour ce faire, nous allons réaliser une étude exhaustive dans le but de concrétiser l'objectif de concevoir un système expert pour la fusion d'interaction multimodale, en utilisant des technologies comme les ontologies, les langages du marquage extensible et le Web sémantique. Donc, ce projet permettra de développer des moyens et des techniques et de fournir des résultats pertinents dans le domaine de l'interaction multimodale.

En outre, dans tous les travaux portant sur ce type d'interaction, nous avons remarqué que la plupart des moteurs de fusion ont été spécialisés. En d'autres termes, ils ont été développés de façon à ce que si un nouveau scénario est identifié, il faille mettre en œuvre un nouveau moteur de fusion. Ceci est dû au manque d'outils et des bases servant à de multiples implémentations. L'architecture est basée sur les standards de W3C (XML, OWL, etc.) caractérisés par la lisibilité (aucune connaissance ne doit théoriquement être nécessaire pour

comprendre le contenu de ce type des documents), la structure arborescente (permettant de modéliser la majorité des problèmes informatiques), l'intégrabilité, l'extensibilité, etc.

De même, l'architecture est basée sur les ontologies qui permettent de répondre aux besoins de la communication entre personnes, personnes-systèmes et systèmes- systèmes. Elles offrent l'interopérabilité, la fiabilité, la réutilisabilité (fixation de la structure des connaissances), un accès commun à l'information et une compréhension partagée des concepts.

L'architecture proposée a trois caractéristiques principales :

- transparence : l'intégration d'un grand nombre de modalités qui empêchent la restriction de l'application à un domaine particulier, par l'utilisation d'une ontologie évolutive. Celle-ci permettra l'ajout des nouvelles connaissances telles que les modalités et les contextes ;
- flexibilité: l'utilisation de l'ontologie permet la description de l'environnement et les différents objets existants ;
- cohérence : par la description de plus grand nombre d'objets et évènements dans l'environnement.

En concevant une architecture qui répond aux faiblesses des modèles déjà étudiés, nous faisons face à de nombreux défis. Nous citons ci-dessous ceux que nous considérons et nous indiquons les solutions que nous souhaitons mettre en œuvre pour les résoudre :

Défi n^{o} 1 : quelle serait la représentation optimale de l'environnement dans l'architecture ?
Solution proposée : nous adoptons une solution basée sur les ontologies. Les modalités, les événements, les objets et leurs caractéristiques seront stockés sous forme de concepts dans une ontologie qui décrit les relations entre eux.

Défi n^{o} 2 : quelles seraient les paramètres qui devraient être pris en compte ?
Solution proposée : outre les modalités, les événements et les objets, l'ontologie doit inclure les paramètres qui permettent de décrire un contexte d'utilisateur (par exemple, les données personnelles et les préférences, le type d'handicap, etc.) Le contexte environnemental (par

exemple, niveau de luminosité du lieu de travail, niveau de bruit, etc.) et le contexte de la localisation (si l'utilisateur se trouve à l'intérieur ou à l'extérieur).

Défi n^{o} 3 : comment représenter des règles qui doivent être prises en compte pour la fusion ?
Solution proposée : notre ontologie doit contenir toutes les conditions possibles nécessaires aux règles de fusion. Ces règles sont utilisées par le moteur de fusion. La description de ces règles par SWRL et SQWRL est basée sur les standards du W3C et sont stockées dans l'ontologie.

Challenge n^{o} 4 : comment représenter des règles nécessaires à la sélection d'une modalité ?
Solution proposée : la sélection est faite selon l'effet d'un contexte spécifique sur une modalité. Cette sélection sera faite par des requêtes SQWRL.

Défi n^{o} 5 : dans quel sens cette architecture se révélera être applicable à de nombreux types de systèmes informatiques?
Solution proposée : l'ontologie comme solution s'appuie sur les langages du web sémantique basés sur les normes du W3C. Cette ontologie décrira tout l'environnement. Cela rend la solution proposée applicable à de nombreux systèmes et l'empêche de se limiter à un domaine d'application spécifique.

L'architecture proposée fournit des fonctionnalités qui permettent de gérer un grand nombre de modalités, de soutenir différents types de systèmes multimodaux et de faciliter le travail d'un moteur de fusion en lui fournissant la plus probable combinaison des différents événements de l'environnement.

CHAPITRE 3

ARCHITECTURE PROPOSÉE

3.1 Introduction

Dans cette thèse, nous voulons définir de nouveaux composants actifs pour représenter l'environnement et puis renforcer l'interaction multimodale. Suite à l'étude générale des domaines et des composants existants, nous avons maintenant toutes les informations qui nous permettent de modéliser une architecture conforme aux formalismes existants et à la représentation des connaissances dans nos composants. Dans ce chapitre, nous allons définir et démontrer la conception de l'architecture.

3.2 Exigences pour modéliser une architecture de l'interaction multimodale

Avant de présenter l'architecture proposée pour notre système d'interaction multimodale, il faut savoir qu'il y a quelques exigences qui doivent être prise en compte pour qu'une architecture multimodale soit faisable.

- E1- Niveaux d'abstraction : des niveaux d'abstraction différents sont conseillés, car une description d'interaction multimodale peut être répartie: par exemple, l'architecture doit séparer la description des événements de dialogue homme-machine. En outre, les différentes parties ou structures peuvent grandement aider lors du développement.

- E2- Modélisation du dialogue homme-machine : il devrait y avoir un moyen de modéliser le dialogue homme-machine, que ce soit avec une machine à états, avec une approche impérative, avec approche de contrôle des structures, une approche déclarative, ou une autre approche.

- E3- Adaptabilité aux contextes et à l'utilisateur : comme les interfaces d'interactions multimodales offrent souvent une équivalence entre les modalités, l'adaptabilité aux

contextes et à l'utilisateur (également appelé plasticité) devrait être prise en compte par une architecture dédiée à l'interaction multimodale. Du côté des entrées, l'adaptabilité mettrait l'accent sur l'utilisation des informations de l'utilisateur et des contextes pour aider à la compréhension des commandes à fusionner.

- E4- Contrôle du mécanisme de fusion : l'algorithme utilisé pour fusionner les données d'entrée multimodales peut être très complexe et pourrait produire des résultats différents en fonction de ses propriétés. Ainsi, les langages de description devraient prendre en considération les paramètres de fusion et les moyens de les contrôler, par exemple, la gestion des paramètres de fusion.

- E5- La synchronisation temporelle : une architecture d'interaction multimodale doit prendre en compte la synchronisation temporelle. Cela surgit lorsque plusieurs événements peuvent conduire à une action donnée. Comment les données doivent-elles être fusionnées si ces événements sont activés en même temps ? Ainsi, le processus de fusion bénéficierait grandement de contrôle de synchronisation temporelle, par exemple en prenant en compte les propriétés CARE présentées dans le chapitre précédent.

- E6- Traitement des erreurs : le traitement des erreurs devrait être pris en compte dès le début du mécanisme de fusion. Les systèmes multimodaux disposent d'un grand nombre de sources possibles d'erreurs, que ce soit dans la reconnaissance, l'intégration ou bien la sélection de la réponse. Par conséquent, une architecture de l'interaction multimodale doit fournir un moyen de gérer les erreurs de reconnaissance, par exemple en permettant des choix par défaut pour préciser, ou d'encourager la conception de dialogues guidés.

- E7- Gestion d'événements : un mécanisme pour la description des événements et de gestion devrait être pris en considération, puisque les événements semblent le moyen naturel le mieux adapté dans une application multimodale qui devrait fonctionner.

- E8- Représentation des sources d'entrée et de sortie : un mécanisme pour représenter les sources effectives d'entrée et de sortie peut également être intéressant. Cela permettra aux créateurs de l'interface multimodale d'avoir le contrôle sur certains paramètres.

Tableau 3.1 Huits exigences pour quatre buts

	E1	E2	E 3	E4	E5	E6	E7	E8
Communication	X	X	X	X	X			
Apprentissage	X	X	X	X	X			
Modélisation		X	X	X	X	X	X	
Configuration		X	X	X	X	X	X	X

Le Tableau 3.1 présente les huit exigences décrites ci-dessus et les met en correspondance avec les quatre buts pour définir une architecture multimodale d'un langage. Les exigences ont été classés du niveau le plus axé utilisateur (niveau d'abstraction) au plus axé système (entrée / sortie représentation des données), d'où la forme diagonale adoptée par les croix dans le tableau.

3.3 Approche générale

L'architecture proposée est basée sur les critères et les exigences mentionnés dans la section précédente. Dans cette partie, une présentation de l'approche générale est identifiée. La Figure 3.1 montre le cycle suivi par un tel système afin d'assurer une interaction correcte. Le cycle commence par l'environnement, qui est le fournisseur des différents événements au système d'interaction multimodale ; après, c'est la détection de ces événements par l'utilisation de différentes modalités reliées au système, ensuite, c'est la compréhension des événements détectés par les modalités, une fois la compréhension achevée, le système doit décider si ces événements peuvent former une commande qui a un sens ou non, c'est plutôt le moteur de fusion qui est responsable de la décision, à la fin, c'est l'action, qui est l'exécution de la commande comprise et c'est la responsabilité du moteur de fission.

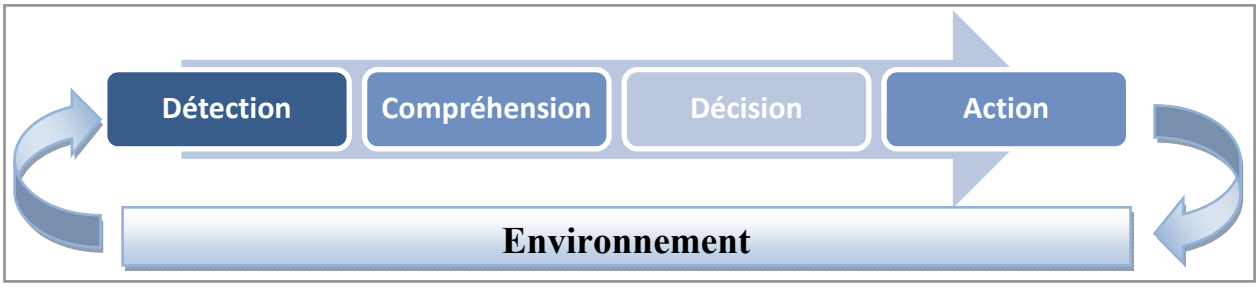


Figure 3.1 Cycle d'un système d'interaction multimodale

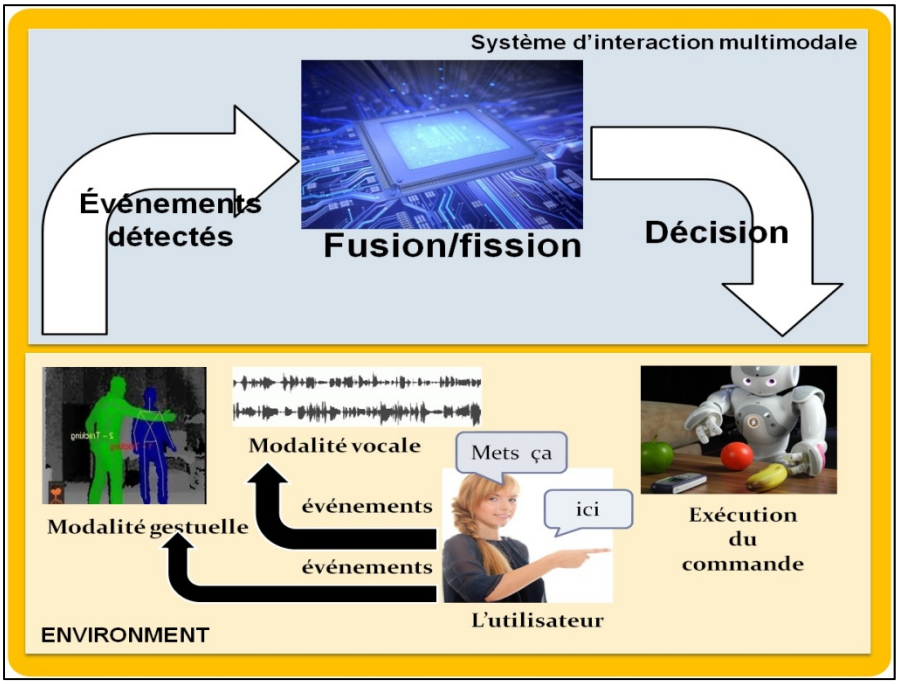


Figure 3.2 Approche appliquée au scénario « mets ça ici »

La Figure 3.2 représente l'approche en appliquant le scénario « mets ça ici ». Dans l'architecture, il y aura un utilisateur qui envoie des événements et qui seront captés par différentes modalités. Dans ce cas, l'utilisateur envoie deux types d'événements : des paroles en disant *mets ça ici* et des gestes en pointant vers l'objet désigné et la place où l'objet doit être déplacé. Ces événements sont détectés par les modalités disponibles dans l'environnement. Dans le cas qui nous intéresse, une modalité vocale et une autre gestuelle. Une fois détectés, les événements seront fusionnés et fissionnés, et à la fin, un robot ou un

système exécute la commande envoyée par l'utilisateur et elle sera exécutée dans l'environnement.

L'architecture générale d'un système d'interaction multimodale sera formée par les composants suivants : 1) un environnement qui fournit la connaissance ambiante sous forme d'événements, 2) des modalités qui détectent les événements et 3) un moteur de fusion nécessaire à la compréhension d'une commande.

Un aperçu général de la conception architecturale est illustré dans la Figure 3.3. Elle montre les différentes composantes nécessaires pour un système d'interaction multimodale et comment un utilisateur ou un objet fournissent des événements en utilisant différentes modalités conduisant à la fusion par le moteur de fusion à l'intérieur du système multimodal. Ces composants sont les suivants :

- environnement : c'est l'endroit où le système multimodal, l'utilisateur, les objets et les modalités cohabitent. Il peut être à l'intérieur comme à l'extérieur ;
- ontologie : c'est la base de connaissances qui décrit tous les détails dans l'environnement (objets, événements, contextes, etc.) ;
- sélection de modalités : c'est la partie responsable de la sélection d'une modalité selon un contexte particulier. Les modalités sont affectées par trois types de contextes :
    - contexte environnemental : considéré comme une information sur l'environnement d'un système informatique. Citons comme exemple le niveau de la lumière, de l'obscurité, le niveau du bruit et le niveau de la température dans l'endroit où le système multimodal se situe ;
    - contexte de l'utilisateur : il décrit le profil de l'utilisateur, surtout s'il a des handicaps ;
    - contexte de localisation : c'est l'identification de la place où se trouve le système multimodal, (bureau, rue, chambre, etc.) ;

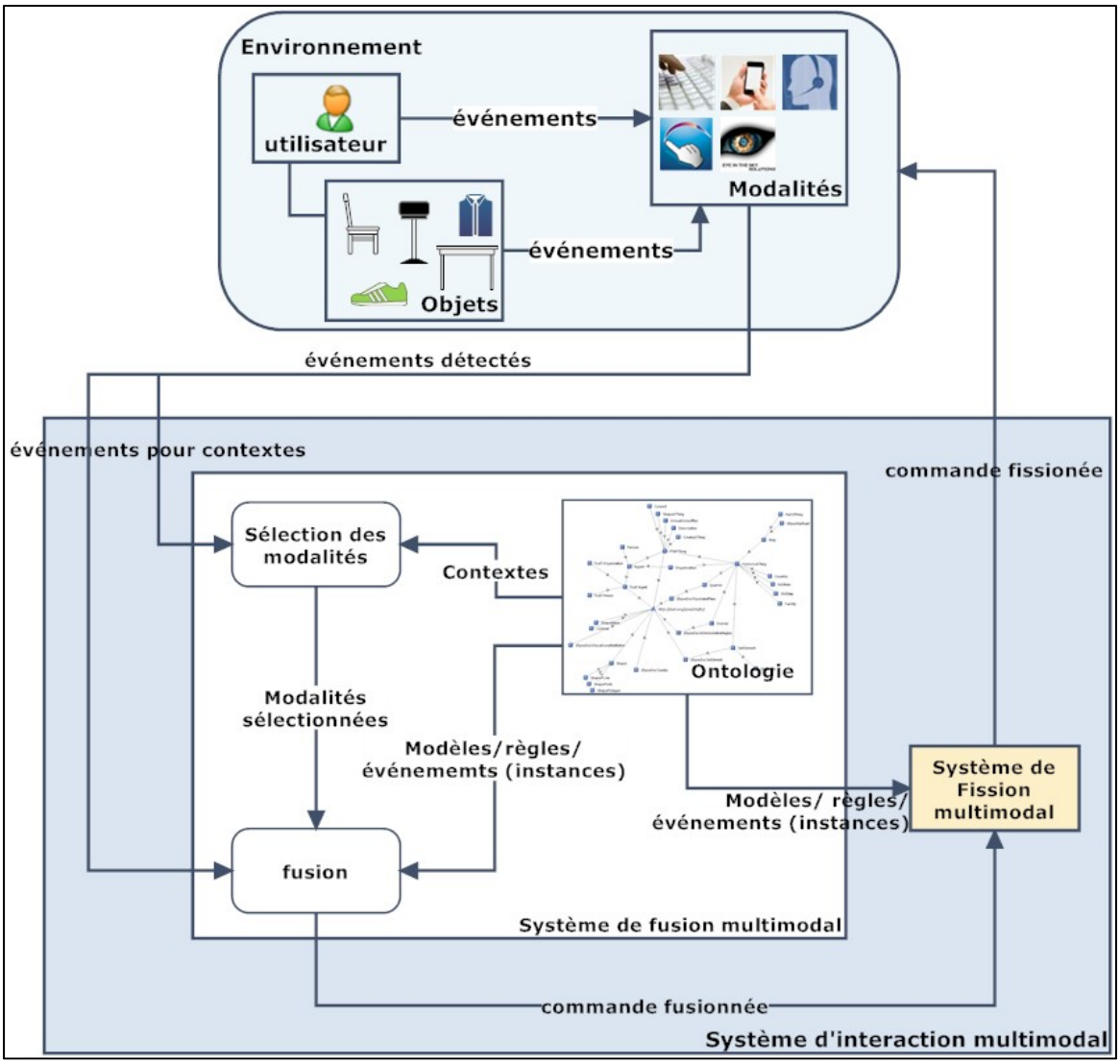


Figure 3.3 La représentation générale de l'architecture

- fusion : la partie responsable de la fusion des événements qui viennent de l'environnent, selon des modèles et des pré-conditions prédéfinis dans l'ontologie ;
- système de fusion multimodale : c'est le système qui contient le moteur de fusion, l'ontologie et le mécanisme de sélection ;
- système de fission multimodale : c'est un système responsable de l'exécution d'une commande après sa compréhension dans le système de fusion multimodale ;
- système d'interaction multimodale : c'est le système complet qui contient à la fois la fusion et la fission.

Le système de fusion multimodale interprète l'environnement, fusionne les événements provenant des modalités d'entrée et produit ensuite une commande complexe. Ensuite, le système de fission multimodale interprète cette commande et la divise en plusieurs tâches élémentaires afin de les présenter à des modalités de sortie disponibles dans l'environnement. L'exécution de ces tâches nécessite la découverte et la sélection de services appropriés pour répondre à ces diverses tâches. Dans notre thèse, nous nous intéressons seulement au système de fusion (le système de fission fait l'objet d'une autre étude dans l'équipe).

3.4 Architecture du système de fusion multimodale

La Figure 3.4 représente l'architecture du système de fusion multimodale. Les événements sont détectés par les modalités dans l'environnement et seront envoyés au système à partir de fichiers XML. Pour sélectionner les modalités, le composant *sélection des modalités* reçoit de l'environnement les événements concernant les contextes ainsi que les instances de contextes décrites dans l'ontologie. Une comparaison est faite au niveau de tous les contextes et les modalités appropriées seront sélectionnées. Une fois sélectionnées, les modalités peuvent envoyer les événements de l'utilisateur au moteur de fusion. Dans ce dernier cas, plusieurs pré-conditions seront testées. Ces pré-conditions, présentées dans la Figure 3.5 seront détaillées dans le chapitre 4. Il est à noter que les modèles de fusion sont sélectionnés à l'aide des requêtes SQWRL stockées dans l'ontologie.

Une fois les pré-conditions testées, le moteur de fusion décide s'il y aura un fusionnement des événements afin d'obtenir une commande fusionnée ou non. Dans le cas négatif, un feedback est généré pour que le système fasse une correspondance avec un autre modèle de fusion dans l'ontologie.

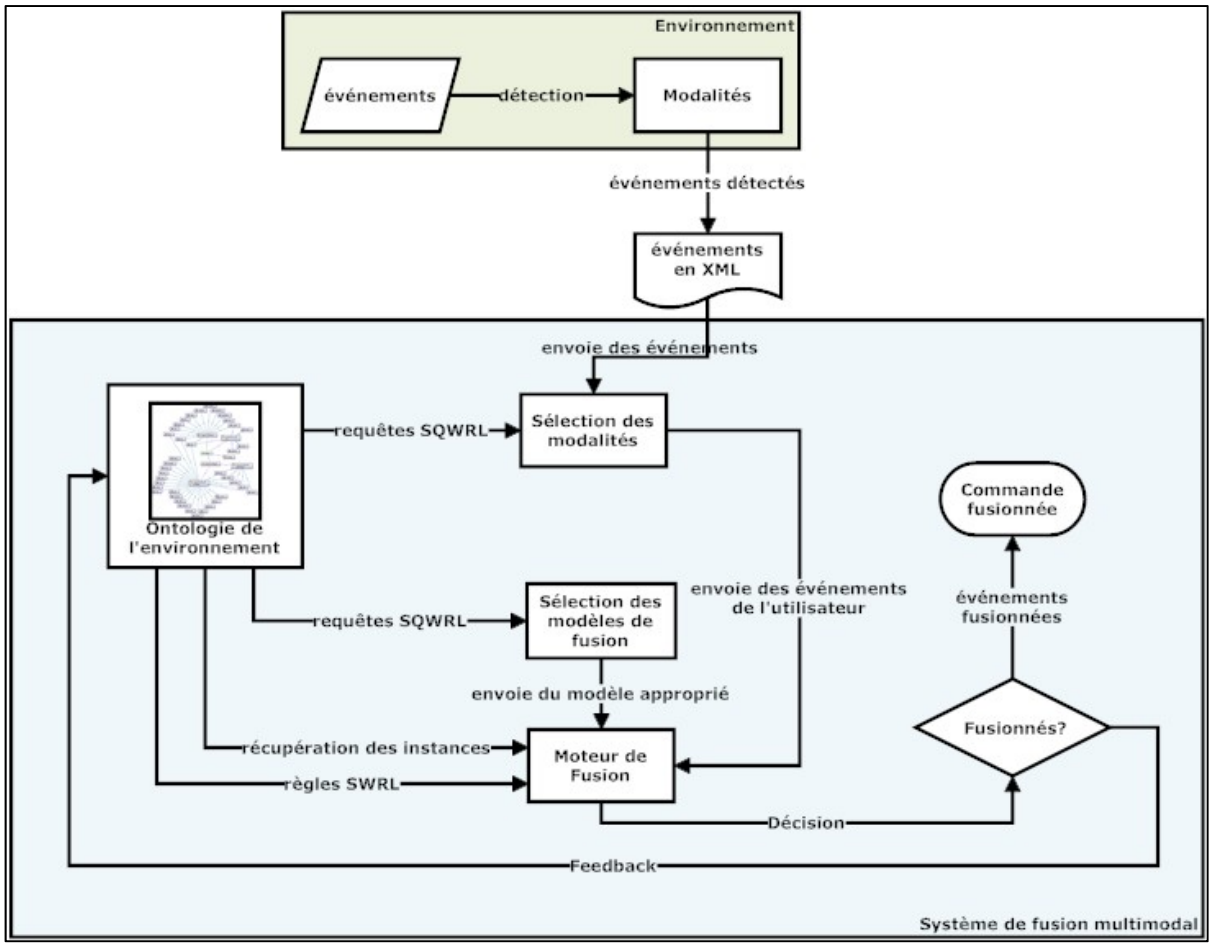


Figure 3.4 Architecture du système de fusion multimodale

3.4.1 L'architecture de sélection des modalités

La Figure 3.5 représente le composant responsable de la sélection des modalités (partie jaune). Dans cette partie, la vérification des contextes déjà présentés dans la section 3.3 est faite. Les événements concernant les contextes sont récupérés à partir des fichiers XML. Le contexte environnemental est testé en prenant les instances concernant ce contexte dans l'ontologie à l'aide des requêtes SQWRL. Si les événements provenant de l'environnement coïncident avec ceux qui existent dans l'ontologie, la modalité reste active. Sinon, elle sera désactivée. Un raisonnement similaire est fait pour le contexte de l'utilisateur et celui de la localisation.

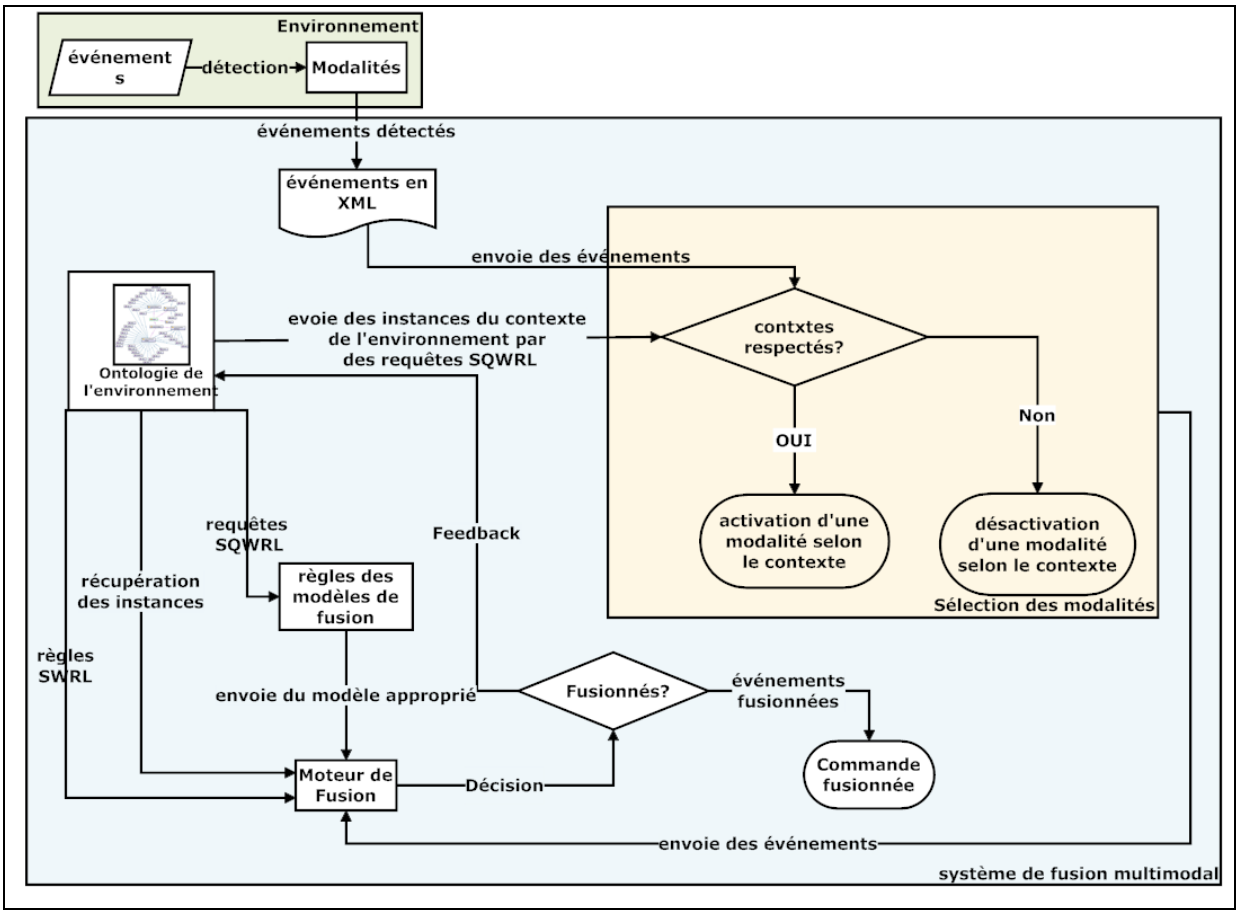


Figure 3.5 La sélection des modalités

Notons qu'une modalité peut être affectée par un ou plusieurs contextes en même temps. Par exemple, la modalité gestuelle a besoin de vérifier le contexte environnemental et celui de l'utilisateur. Il faut que le niveau de lumière dans l'endroit où se trouve la modalité soit suffisant pour qu'elle détecte correctement les gestes. Similairement, pour le contexte de l'utilisateur, il ne faut pas que celui-ci soit aveugle, par exemple, car une personne aveugle ne peut pointer vers des objets dans l'environnement. Dans ce cas, la modalité gestuelle sera désactivée car elle ne vérifie pas ces deux contextes.

3.4.2 L'architecture du moteur de fusion

La Figure 3.6 représente le fonctionnement du moteur de fusion. Elle montre comment il teste les différentes pré-conditions afin de fusionner les événements provenant de l'environnement. Le moteur de fusion vérifie quatre pré-conditions :

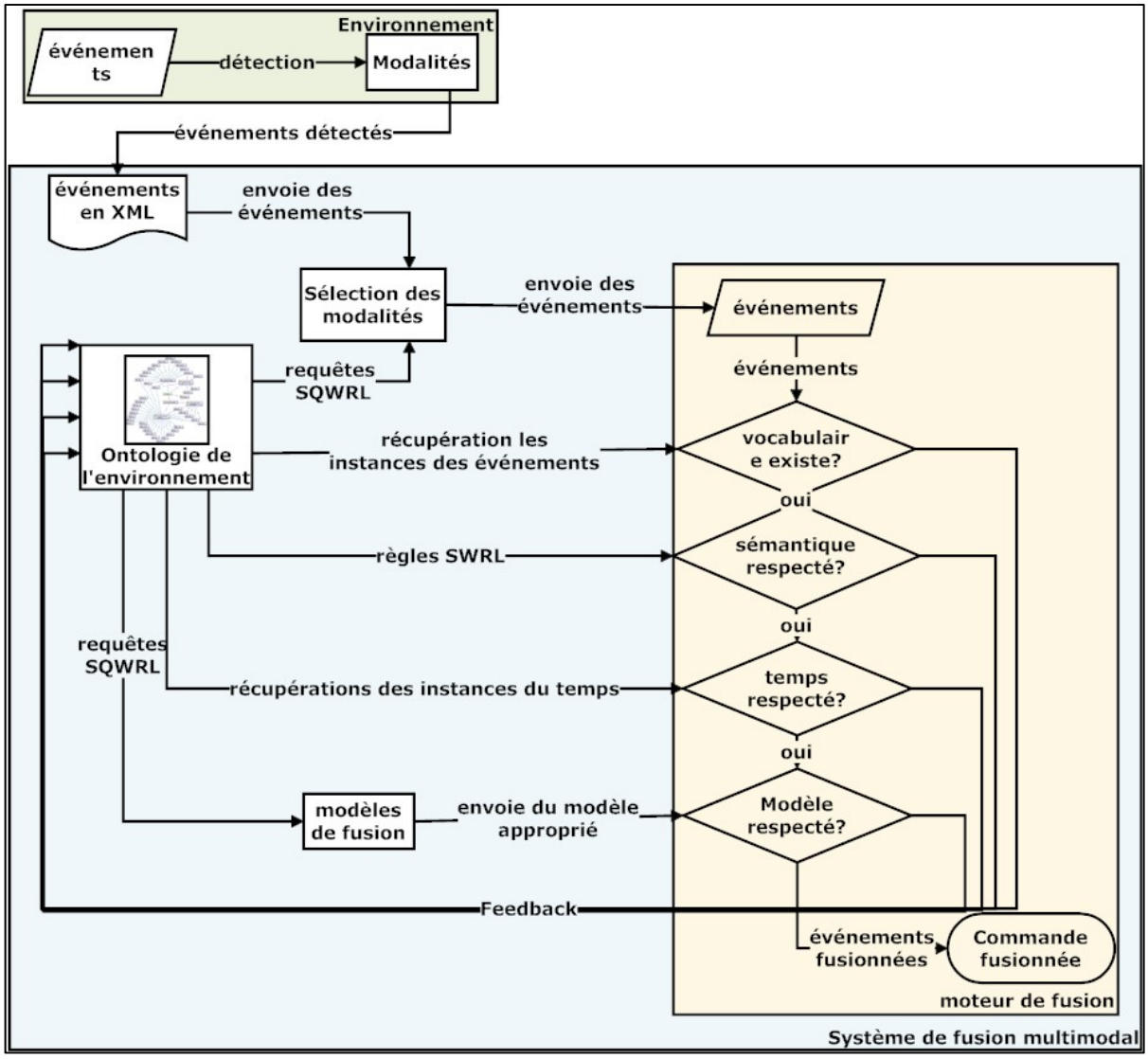


Figure 3.6 Architecture du moteur de fusion

- la vérification du vocabulaire : pour savoir si les événements de l'environnement sont décrits dans l'ontologie ou non, en récupérant les instances de l'ontologie afin de savoir

le type d'un tel événement (ex : un objet, une action, etc.). Si le vocabulaire est vérifié, une autre pré-condition sera vérifiée. Sinon, l'utilisateur est informé et la fusion n'aura pas lieu ;

- la vérification de la sémantique : en vérifiant si les relations entre les événements identifiés sont sémantiquement corrects. Par exemple, si nous avons un événement de type objet et comme instance « mur » et un autre événement de type action et une instance « bouge », il faut savoir qu'il n'y a pas de relation entre ces deux instances car on ne peut pas bouger un mur. Ces relations sont testées à partir des règles SWRL. Une fois vérifiée, une autre pré-condition sera testée. Sinon, l'utilisateur est informé et la fusion n'aura pas lieu ;
- la vérification de l'aspect temporel : en comparant le temps des événements avec des intervalles de temps définis dans l'ontologie, car un système ne peut pas attendre indéfiniment les événements de l'environnement. Par exemple, si un utilisateur envoie une requête incomplète au système, après un certain temps, ce dernier doit réagir, soit en annulant la requête soit en demandant à l'utilisateur de la compléter. Une fois vérifiée, la dernière pré-condition sera testée. Sinon, l'utilisateur est informé et la fusion n'aura pas lieu ;
- la vérification d'un modèle : les modèles sont des combinaisons d'événements dans un ordre précis et qui peuvent avoir un sens. Ils sont récupérés de l'ontologie à partir des requêtes SQWRL et comparés avec l'ordre des événements provenant de l'environnement. La sélection des modèles est présentée dans la Figure 3.7.

Une fois toutes les pré-conditions testées, le moteur de fusion fusionne les événements et comprend la situation qui se passe dans l'environnement.

Notons que les pré-conditions sont indépendantes et qu'elles peuvent être vérifiées dans n'importe quel ordre. Par contre, si l'une des pré-conditions n'est pas vérifiée, il n'y aura pas de fusion.

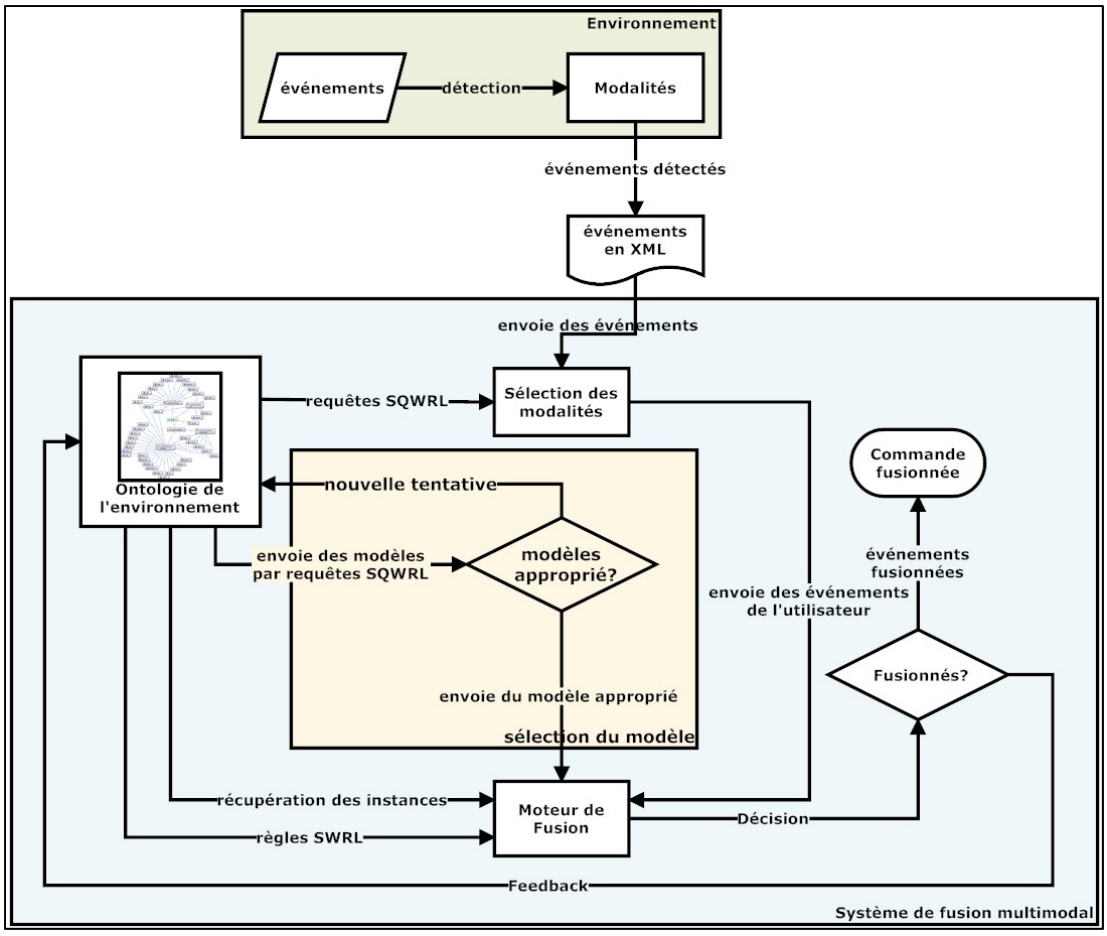


Figure 3.7 Sélection des modèles

3.5 Conclusion

Dans ce chapitre, nous avons présenté les différents composants nécessaires pour notre architecture. Nous avons également présenté la relation entre ces composants et les différentes pré-conditions nécessaire pour la fusion multimodale. L'architecture proposée a pris en considération les exigences décrites au début de ce chapitre, que ce soit par l'interprétation des erreurs en proposant un vocabulaire spécifique, ou par la définition des différents types d'événements dans une ontologie ou bien par l'aspect temporel à respecter. Dans le chapitre suivant, nous présenterons la conception de l'architecture avec tous ses composants en détail.

CHAPITRE 4

CONCEPTION DE L'ARCHITECTURE

4.1 Introduction

Dans ce chapitre, chaque composant de l'architecture présentée précédemment sera modélisé. Il existe quatre composants principaux à modéliser qui sont les suivants :

- la modélisation de l'environnement : à l'aide d'une ontologie qui le décrit en détail ;
- la sélection des modalités : en se basant sur des contextes définis dans l'ontologie et à partir des requêtes SQWRL ;
- la sélection des modèles de fusion : qui sont décrits dans l'ontologie en utilisant des requêtes SQWRL ;
- le moteur de fusion : qui utilise les trois composants précédents afin de fusionner les événements de l'environnement.

4.2 Modélisation de l'environnement

Pour modéliser l'environnement, nous utiliserons le formalisme d'ontologie. Une ontologie est une description explicite et formelle des concepts (classes) du domaine et des relations (propriétés, instances, restrictions, etc.) entre ces concepts. L'ontologie permet de définir un vocabulaire commun pour partager une même connaissance. Elle est une base de connaissances car elle contient les règles applicables aux concepts et aux relations et l'un des meilleurs formalismes pour l'ingénierie des connaissances De cette façon, il sera possible de partager une compréhension commune de la structure de l'information, de permettre l'analyse et la réutilisation de la connaissance du domaine.

4.2.1 Outil de modélisation de l'ontologie

Nous utilisons un outil qui s'appelle PROTÉGÉ, c'est un logiciel qui permet de construire des ontologies. Nous avons retenu ce choix parce que PROTÉGÉ est un atelier logiciel qui

intègre plusieurs plug-ins capables de mettre en pratique de façon concrète les différents aspects de l'ingénierie des connaissances. Pour cette raison, nous nous limiterons à décrire l'outil PROTÉGÉ, ce qui nous apparaît plus complet, au lieu de présenter différents outils.

Pour répondre à un besoin de plus en plus grandissant de pouvoir créer des outils d'acquisition de connaissances, une équipe de l'université de Stanford a décidé d'élaborer un logiciel qui accomplirait cette tâche : PROTÉGÉ. D'abord construit pour générer des outils personnalisés pour l'acquisition de connaissances, cet atelier de développement logiciel permet maintenant de construire des systèmes de connaissances (Gennari 2002). Les auteurs ont été motivés par la nécessité de réduire le goulot d'étranglement créé dans l'acquisition de connaissances. Leur objectif était de minimiser la tâche lors de la construction de bases de connaissances (Gennari 2002).

À l'origine, PROTÉGÉ était une petite application destinée à construire des outils d'acquisition de connaissances spécialisés dans la planification médicale. Le système a ensuite connu différentes évolutions vers un environnement flexible et robuste de développement.

L'évolution de PROTÉGÉ s'est faite en quatre grandes phases durant une quinzaine d'années de développement. La première version, PROTÉGÉ-I, était une application limitée au domaine de la planification médicale et comprenait une seule méthode de résolution de problèmes, ESPR ou *episodic skeletal-plan refinement method*, (Tu 1989). Cette méthode est basée sur l'instanciation et l'amélioration progressive des plans squelettiques pour des problèmes spécifiques. C*et al*gorithme de résolution des problèmes était intégré directement dans le système. Les utilisateurs fournissaient les données à l'exécution. La deuxième version, PROTÉGÉ-II, incorporait plusieurs méthodes de résolution de problèmes, séparées du système. Ainsi, on pouvait appliquer différentes méthodes de résolution de problèmes à différentes bases de connaissances. La troisième version, PROTÉGÉ/Win, intégrait de façon dynamique un ensemble d'outils indépendants pour construire un système de connaissance. Dans cette version, l'accent avait été mis sur la facilité d'intégration des outils dans le système global et sur les notions de modularité et de réutilisation. Cette version était conçue

pour une communauté limitée d'usagers. Enfin, la dernière version, PROTÉGÉ-2000, inclut le modèle de connaissances OKBC (Open Knowledge-Base Connectivity) et une interface de programmation qui ont rendu l'outil plus extensible et flexible.

4.2.2 Modélisation de l'ontologie

4.2.2.1 Définition des classes

Comme mentionné précédemment, le rôle de l'ontologie est de décrire sémantiquement l'environnement qui entoure l'utilisateur et le système multimodal. La Figure 4.1 représente les principaux concepts nécessaires à la description de l'environnement dans n'importe quel scénario.

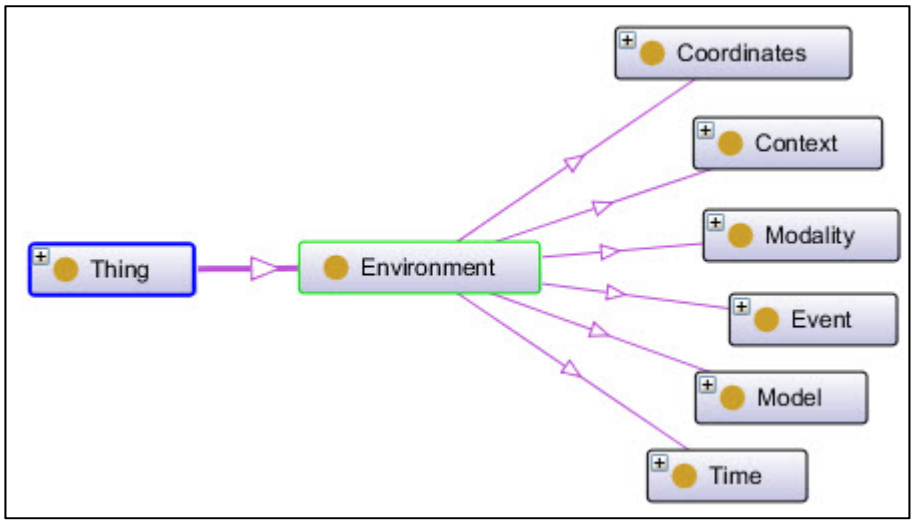


Figure 4.1 Les concepts essentiels de l'environnement

Elle est formée par ces 6 concepts généraux:

- Model : une classe générale définie dans l'ontologie qui contient un ensemble de modèles qui représentent différentes combinaisons d'événements correctes pour différentes commandes d'un utilisateur ;

- Context : une classe définie dans l'ontologie qui décrit les différents contextes qui affectent la sélection des modalités ;
- Coordinates : une classe définie dans l'ontologie utilisée pour localiser des objets, des places et des personnes ;
- Modality : une classe définie dans l'ontologie qui contient les différentes modalités utilisées dans l'environnement ;
- Time: une classe définie dans l'ontologie qui représente le temps maximal pour une commande et le temps entre les différents événements qui forment cette commande ;
- Events: sont captés par les modalités et envoyés vers le système de fusion pour être fusionné.

Le concept *Environmen*t est relié avec les autres concepts par des relations de type *hasSubCalss* pour désigner qu'ils sont des sous-classes de la classe mère *Environment.* Cette relation est une propriété du langage OWL.

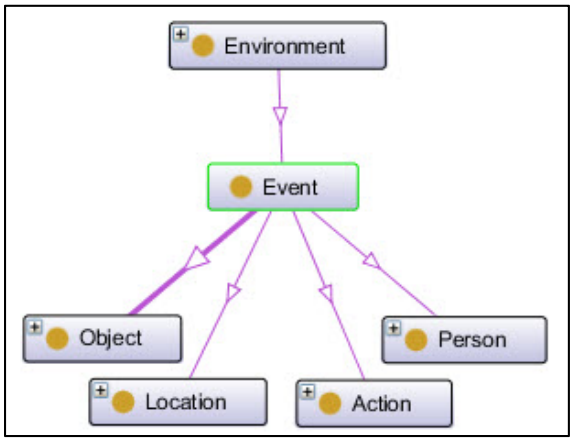


Figure 4.2 Les sous-classes de la classe Event

La Figure 4.2 représente les quatre sous-classes de la classe *Event* suivantes :

- Object : décrit les différents objets de l'environnement (ex. : table, livre, pomme, etc.) ;
- Location : décrit les différents endroits de l'environnement (ex. : cuisine, salle de séjour, ici, là-bas, etc.) ;

- Action : décrit les ordres envoyés par l'utilisateur au système multimodal (ex. : donner, chercher, fermer, etc.) ;
- Person : décrit les différentes personnes présentes dans l'environnement (ex. : tante, père, mère, lui, moi, etc.).

Chacun de ces quatre concepts est formé par des sous-classes qui décrivent en détail les différents types d'objets, d'endroits, d'actions et de personnes. La Figure 4.3 montre les différentes sous-classes de la classe *Object* et la relation *hasCoordinates* avec la classe *Coordinates* qui est une relation de type « propriété d'objet ». Cette relation a été créée dans l'ontologie pour dire que chaque objet a des coordonnées.

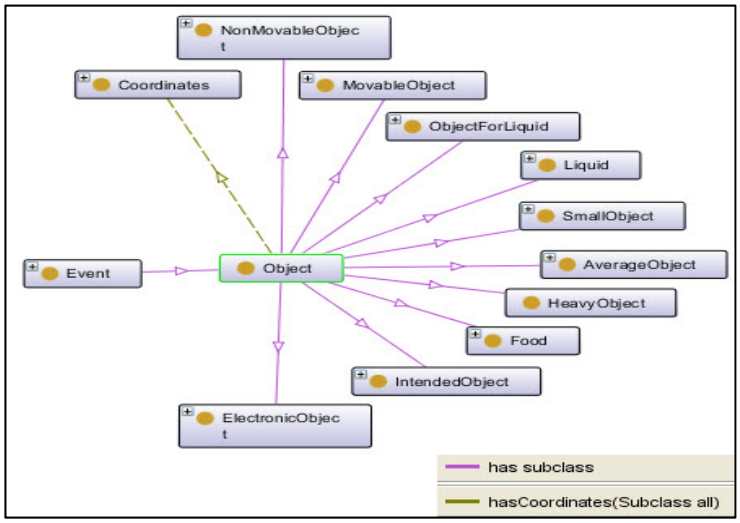


Figure 4.3 La classe object et ses sous-classes

Figure 4.4 La restriction DisjointWith

À noter que toutes les sous-classes d'*Object* ont hérité cette propriété, puisque *Coordinates* est reliée avec la classe mère *Object.* Cette classe contient :

- NonMovableObject : sont les objets qui ne peuvent pas être déplacés (ex : murs, porte, etc.) ;
- MovableObject : sont les objets qui ne font pas partie de NonMovableObject. Pour cela, nous avons créé une restriction dans l'ontologie de type *DisjointWith* (méthode du langage OWL) pour dire que la classe MovableObject et ses sous-classes n'ont pas de relation avec la classe NonMovableObject et ses sous-classes (voir Figure 4.4) ;
- Liquid : sont les objets de type liquide (eau, jus, soda, etc.) ;
- SmallObject : sont les objets de petite taille (clé, livre, montre, etc.) ;
- AverageObject : sont les objets de taille moyenne (micro-onde, chaise, etc.) ;
- HeavyObject : sont les objets lourds (lit, armoire, réfrigérateur, etc.) ;
- Food : sont les objets de type aliment (pomme, banane, viande, etc.) ;
- IntendedObject : sont des objets désignés par un geste (ça, ces, etc.) ;
- ElectronicObject : sont les objets électroniques (TV, radio, lave-linge, etc.).

Cette répartition des objets nous aidera dans la construction des modèles de fusion. Par exemple l'action "allumer" ou bien "éteindre" est utilisée pour les objets électroniques.
La Figure 4.5 représente la « propriété d'objet » *hasObject* pour montrer que les concepts *IntendedObject, Liquid, AverageObject, SmallObject, ElectronicObject* et *Food* sont des objets qui peuvent être déplacés. Les objets lourds ne font pas partie du *MovableObject*, car ils ne peuvent pas être déplacés d'une manière simple comme les autres objets.

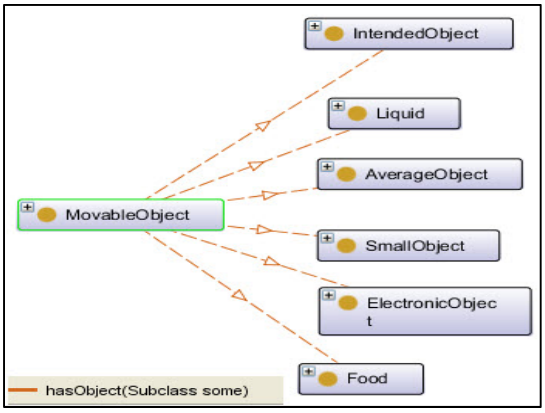


Figure 4.5 Les objets qui peuvent être déplacés

La Figure 4.6 représente la classe *Action* et ses sous-classes. Cette classe représente les différents ordres envoyés par l'utilisateur au système. Les sous-classes de la classe *Action* sont les suivantes*:*

- ActionForPerson : sont les actions utilisées pour les personnes (ex. : appeler, demander, parler, etc.) ;
- ActionForLocation : sont les actions pour les endroits (ex. : trouver, chercher, etc.) ;
- ActionForObject : contient deux sous-classes, *ActionForMovableObjext* (ex. : bouger, prendre, ramener, etc.) et *ActionForNonMovableObject* (ex. : fermer, nettoyer, etc.).

Les actions peuvent avoir des instances identiques. Par exemple, l'instance *chercher*, peut être utilisée pour trouver un endroit ou bien une personne.

La Figure 4.7 représente les deux classes *Location* et *Person* et leurs sous-classes. La classe *Location* décrit les endroits détectés par le système multimodal. Elle contient les trois sous-classes suivantes :

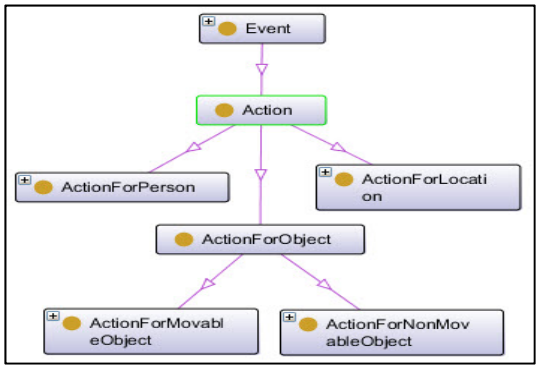


Figure 4.6 La classe *Action* et ses sous-classes

- Outdoor : les endroits qui sont en dehors de la maison (ex. : bureau, jardin, rue, etc.) ;
- Indoor : les différentes pièces de la maison ou bien n'importe quel environnement fermé (ex. : séjour, bureau, établissement, etc.) ;
- IntendedLocation : ce sont les endroits désignés par un geste (ex. : ici, là, là-bas, etc.).

La classe *Person* définit les personnes (ex : père, mère, frère, etc.) et elle a une sous-classe *IntendedPerson*, qui correspond à la personne désignée (ex : lui, nous, moi, etc.).

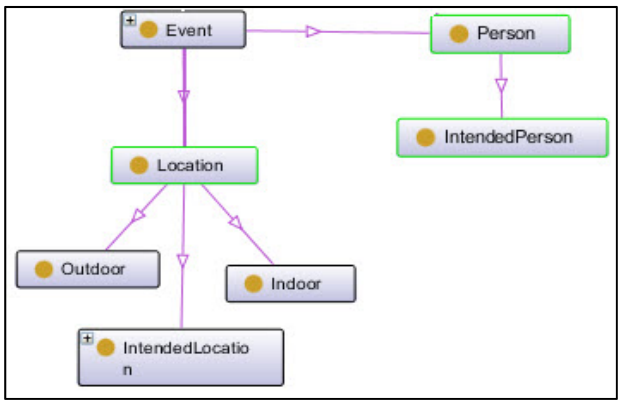


Figure 4.7 Les classes *Location* et *Person*

Les classes *Location, Person et Object* ont toutes une relation *hasCoordinate* avec la classe *Coordinates,* car chacune d'elles a des coordonnées. Cela est représenté dans la Figure 4.8.

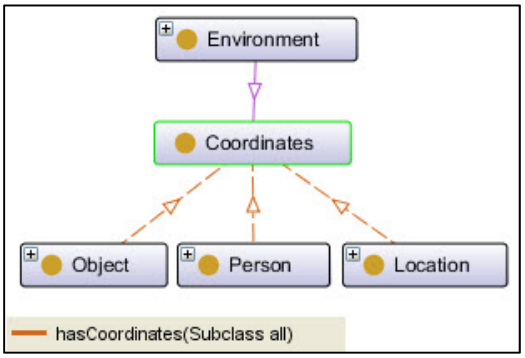


Figure 4.8 La relation avec la classe *Coordinates*

La Figure 4.9 représente la classe *Context* et ses sous-classes. Cette classe décrit les différents contextes qui peuvent affecter la sélection d'une modalité. Elle est formée par trois sous-classes :

- UserContext : décrit le profil d'un utilisateur et s'il présente un quelconque handicap. Elle contient les sous-classes suivantes : *LName, FName, Sex, Age, et Handicap* ;

- EnvironmentalContext : décrit le niveau de bruit, de lumière et de température dans l'endroit où se trouve le système multimodal. Elle contient les sous-classes suivantes :
  - Noise : décrit le niveau du bruit dans l'endroit où se trouve le système multimodal ;
  - Light : décrit le niveau de lumière dans l'endroit où se trouve le système multimodal ;
  - Temperature : décrit le degré de la température dans l'endroit où se trouve le système multimodal ;
- LocationContext : décrit si une modalité peut être utilisée dans un endroit spécifique, par exemple, il n'est pas recommandé d'utiliser les gestes en conduisant une voiture.

La Figure 4.10 représente la classe *Modality* et les types de modalités sous forme de sous-classes. Nous avons identifié cinq types de modalités d'entrée :

- GestuelModality : représente les modalités gestuelles destinées à la reconnaissance des gestes d'un utilisateur (ex : gants électroniques, etc.) ;

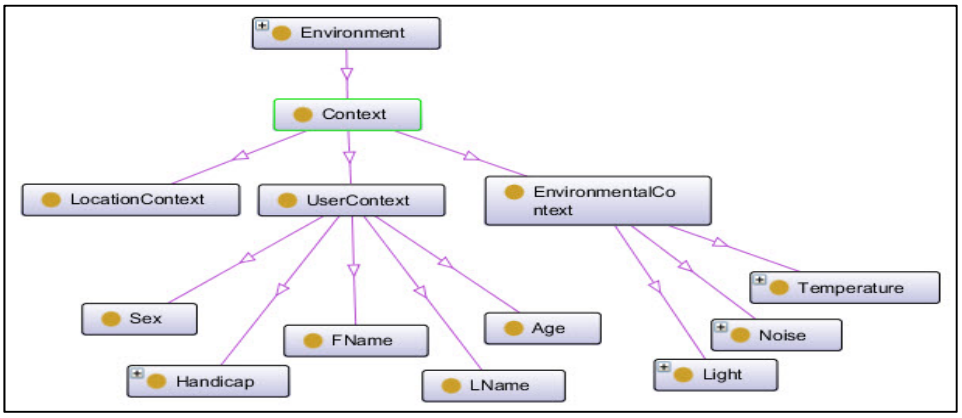


Figure 4.9 La classe *Context* et ses sous-classes

- VocalModlity : représente les modalités vocales destinées à la reconnaissance des commandes vocales d'un utilisateur (ex. : microphone, système de reconnaissance vocale, etc.) ;
- TactileModality : représente les modalités tactiles destinées à la reconnaissance d'une commande d'un utilisateur par la touche (ex. : écran tactile, etc.) ;

- VisualModality : représente les modalités visuelles destinées à la reconnaissance du mouvement des yeux (ex. : capteur de mouvement des yeux, etc.) ;
- ManualModality : représente les modalités destinées à saisir manuellement les commandes comme une souris ou un clavier.

La Figure 4.11 représente la classe *Time* qui décrit l'aspect temporel d'une commande. Elle est formée par les deux sous-classes suivantes :

- MaxCommandTime : représente le temps maximal d'une commande ;
- MaxActiveModalityTime : représente le temps maximal entre deux modalités actives (l'aspect temporel sera expliqué en détail dans le chapitre suivant).

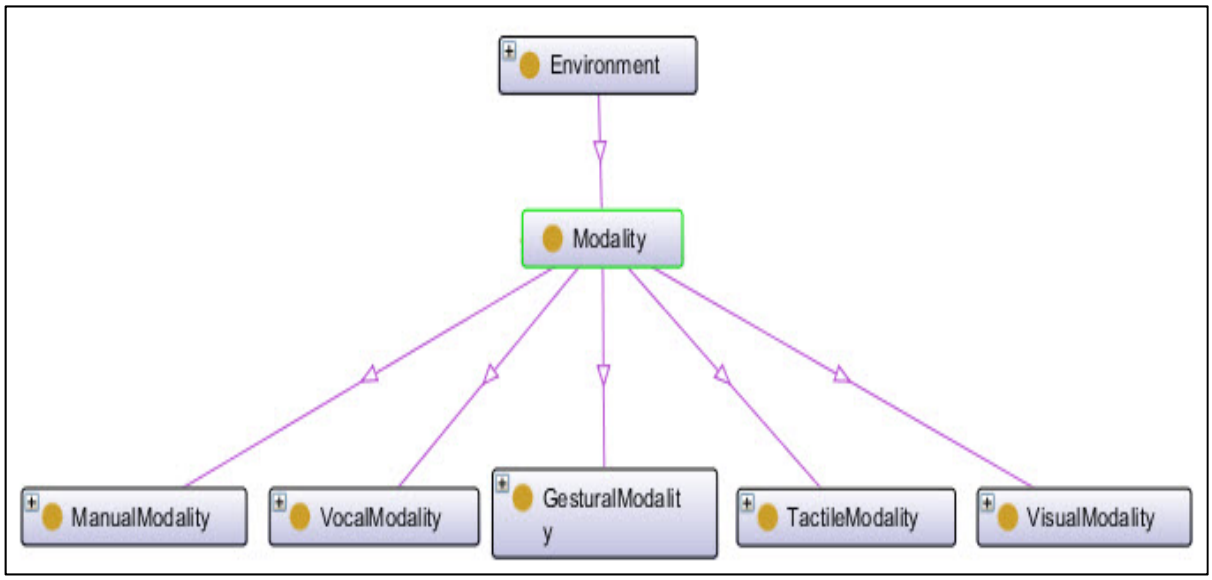


Figure 4.10 La classes *Modality* et ses sous-classes

Les modèles de fusion sont représentés dans la Figure 4.12. Chaque modèle décrit un ordre spécifique des événements qui forme une commande complète (les modèles seront détaillées dans la partie sélection des modèles).

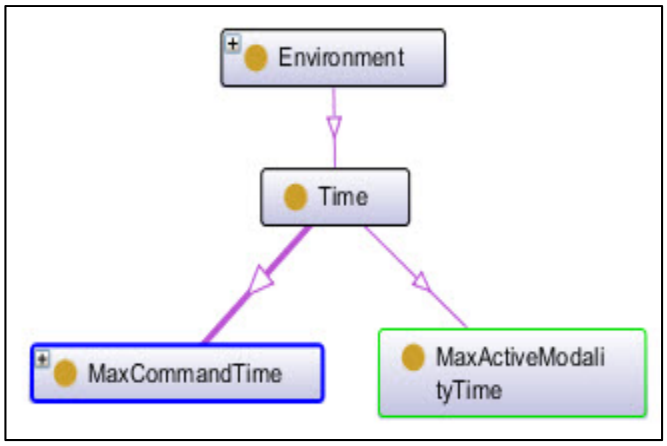


Figure 4.11 La classe *Time* et ses sous-classes

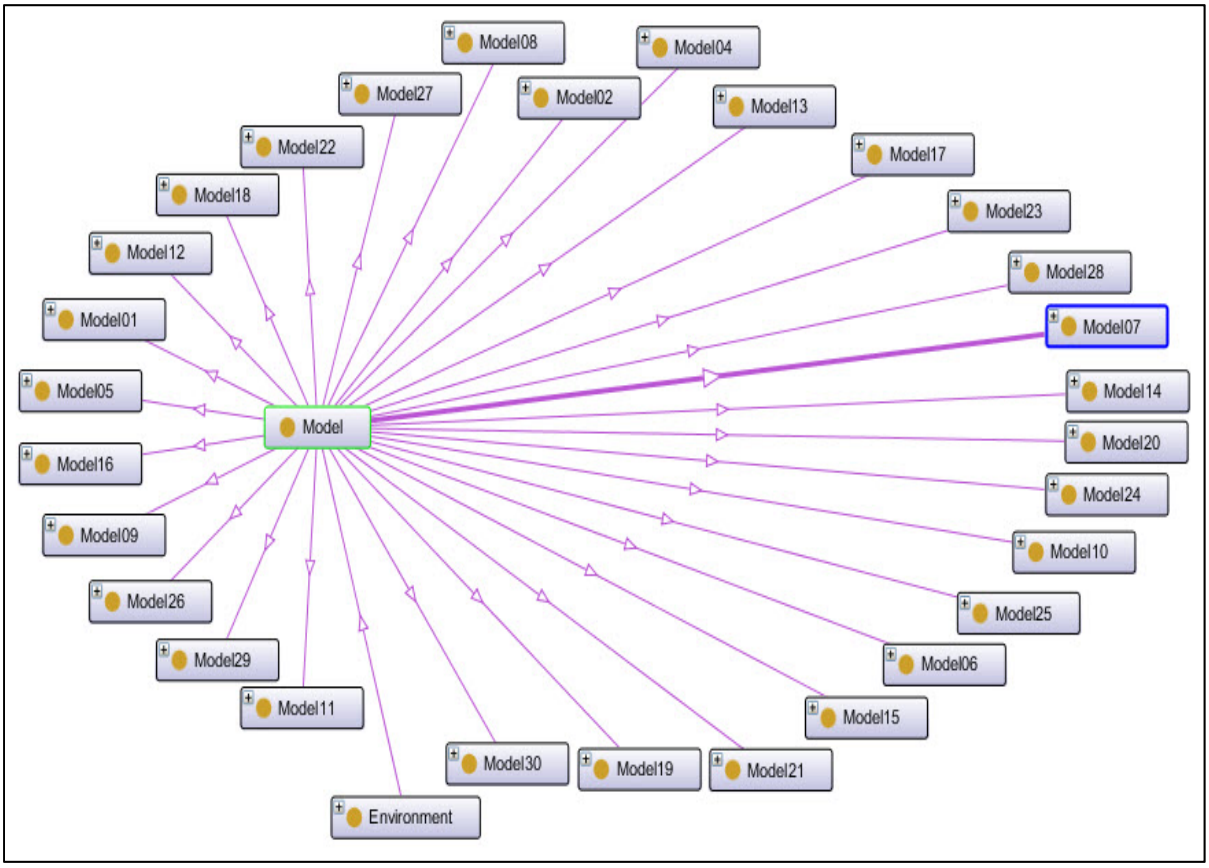


Figure 4.12 Les différents modèles

4.2.2.2 Définition des instances de l'ontologie

Une fois les classes qui décrivent l'environnement sont définies, nous avons besoin de définir notre vocabulaire. Le vocabulaire dans notre modélisation est la représentation explicite des connaissances qui désignent un objet, une personne, un endroit, une action, une modalité, un temps et des coordonnées de l'environnement. D'une manière plus simple le vocabulaire est utilisé pour dire qu'une clé, un livre, une montre, des lunettes, etc. font partie de la classe *SmallObject* par exemple. Le vocabulaire est défini dans l'ontologie sous forme d'instances. Le Tableau 4.1 montre les différentes instances définies et leurs classes appropriées. Les instances de la partie grise du tableau sont normalement définies lors de la configuration de la première utilisation du système. Dans ce document, les instances sont données à titre d'exemples. En réalité, chaque environnement a ses propres modalités, sources de lumière et de bruit et chaque utilisateur a son propre rythme de parole, donc des intervalles de temps différents.

Tableau 4.1 Les instances définis dans l'ontologie

Classe	**Instance**
ActionForLocation	*Check, find, locate, search*
ActionForMovableObject	*Change, clean, close, cut, drag, draw, drop, fill, find, get, give, hold, lock, move, open, place, repare, put, read, take, turnoff, turnon, unlock, write*
ActionForNonMovableObject	*Clean, close, lock, ope, unlock*
ActionForPerson	*Answer, ask, call, contact, demand, find, give, help, locate, reply, show, take*
Indoor	*Bathroom, bedroom, corner, dinnerroom, kitchen, livingroom*
Outdoor	*Backyard, road*
IntendedLocation	*After, before, behind, from, here, in, inside, left, on outdide, right, there, to under*
AverageObject	*Box, chair, computer, desk, dvdplayer, luggage, microwave, radio, sterio, table, television*
ElectronicObject	*Computer, dishwasher, dvdplayer, microwave, mobile, oven, phone, radio, refrigarator, remotecotrol, sterio, television,*

Classe	Instance
	washingmachine
Food	*Apple, banana, bread, carrot, cheese, chicken, fish, meat, orange, pasta, pineapple, pizza, salt, spices, strawberry, suggar, tomato*
HeavyObject	*Bed, closet, dishwasher, door, oven, refrigerator, sofa, wall, washingmachine, window*
IntendedObject	*It, that, these, this*
Liquid	*Beer, coffe, jus, milk, soda, tea, water, wine*
MovableObject	*Sont les instances d'AverageObject, ElectronicObject, Food, IntendedObject, Liquid, ObjectForLiquid, SmallObject et heavyObject sauf wall, door, window*
NonMovableObject	*wall, door, window*
ObjectForLiquid	*Bottle, cup*
SmallObject	*Bag, book, boot, bottle, cable, cigaret, cup, drug, flower, fork, glass, glasses, glove, key, knife, mobile, notebook, pants, paper, pen, phone, plate, remotecontrol, ruler, spoon, tshirt, vest, watch*
Person	*Aunt, brother, cousin, father, freind, mother, nephew, niece, sister, son, uncle*
IntendedPerson	*Her, him, me, them, us*
GesturalModality	*Gestures_Sensor01_Living_Room*
ManualModality	*Keaybord, mouse*
TactileModality	*tactileScreen*
VisualModality	*Eye_Gaze_Sensor*
VocalModality	*Voice_Sensor_living_Room*
MaxActiveModalityTime	*5*
MaxCommandTime	*10*
Light	*LightBedRoom, LightDesk, LightLivingRoom*
Noise	*NoiseFromOutside*
LocationContext	*Home, work, road*
Handicap	*Blind, deaf, manualHandicap, mute*
Coordinates	*Les coordonnées des différents objets, endroits et personnes*

La Figure 4.13 montre comment les instances sont définies dans l'ontologie. Le schéma représente les instances des classes *ActionForPerson, ActionForMovableObject,*

ActionForNonMovableObject et *ActionForLocation* et montre comment elles peuvent partager des instances identiques parfois comme dans le cas de *find, take* et *give* par exemple.

4.2.2.3 Relations sémantiques de l'ontologie

Après la création des classes et des instances, dans cette partie nous présenterons les relations sémantiques entre les classes elles-mêmes, les classes et les instances, les instances elles-mêmes. Deux types de relations ont été utilisées dans l'ontologie, des propriétés d'objets (entre classes et/ou instances) et des propriétés de données (entre instances et leurs valeurs). Les relations sémantiques sont très importantees car elles tiennent le rôle d'une « cartographie » de la connaissance, elles permettent de situer les connaissances les unes par rapport aux autres.

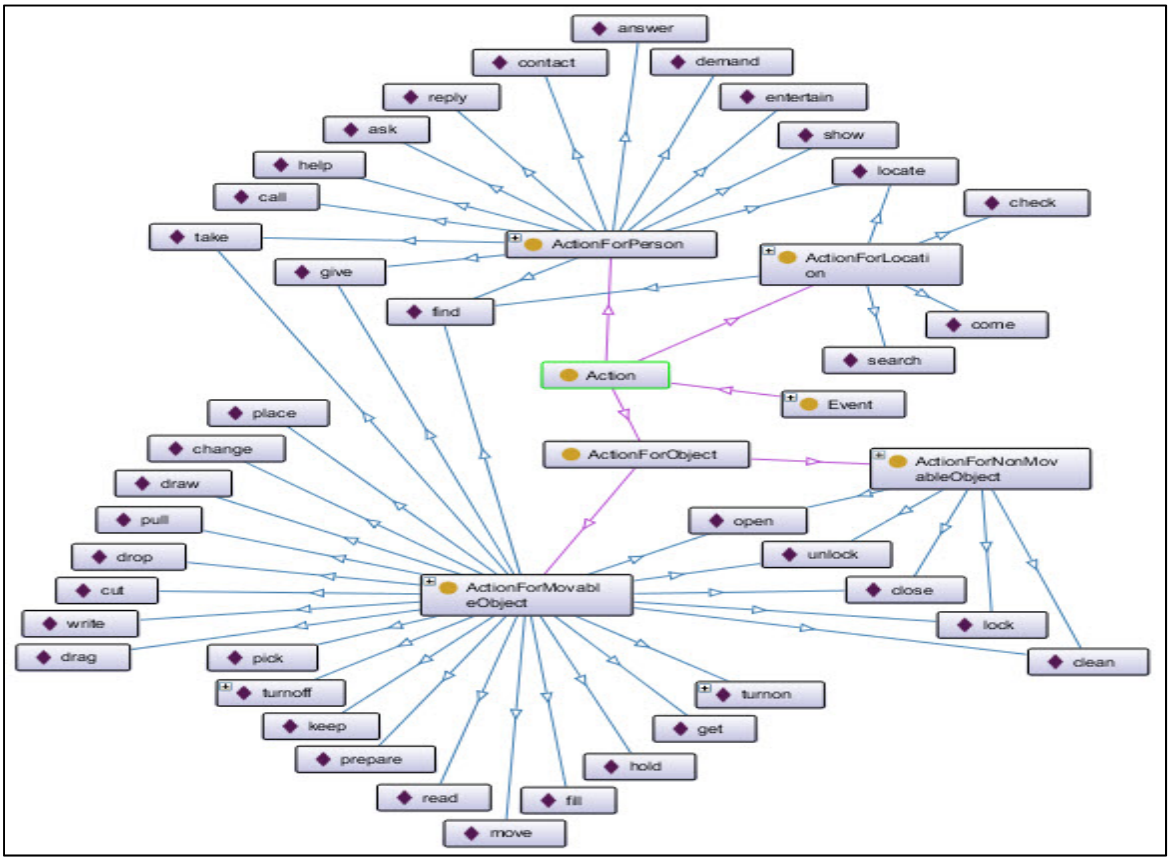


Figure 4.13 Les instances des sous classe de la classe Action

La Figure 4.14 représente la relation entre la classe *Modality* et les sous-classes de la classe *Context.* Une telle relation existe, car les modalités sont affectées par les différents contextes et spécialement par la lumière, le bruit, la température, le type d'handicap et l'endroit où se trouve une modalité, d'où la création des relations *hasLight, hasLocationContext*, *hasNoise*, *hasTempérature, hasUserContext* entre *Modality* et *Light, LocationContext, Noise*, *Temperature* et *Handicap* respectivement.

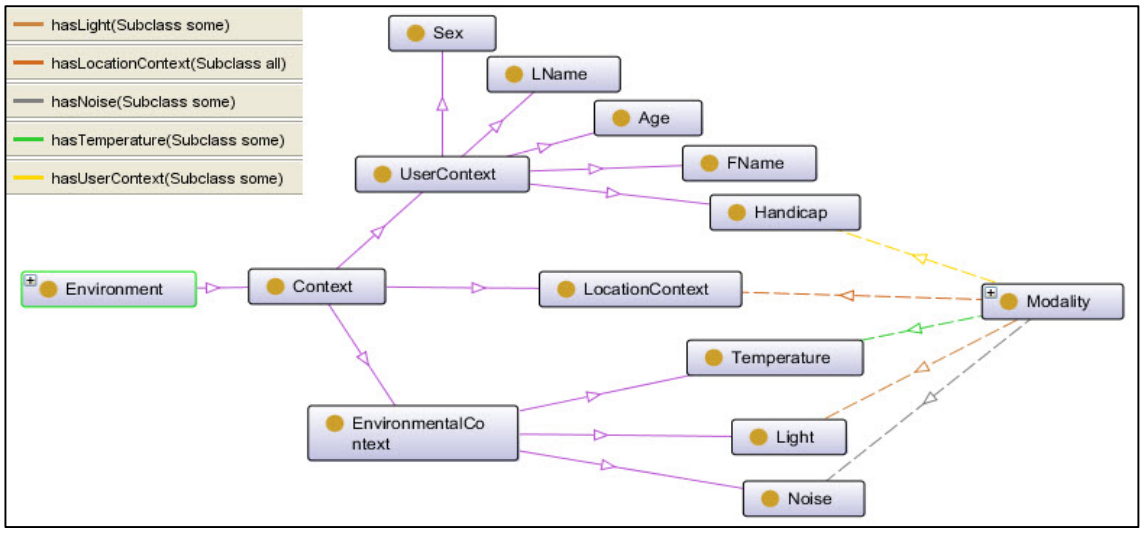


Figure 4.14 La relation entre la classe *Modality* et *Context*

La Figure 4.15 représente la relation entre les deux classes *Model* et *Modality* avec la classe *Time.* Ces relations existent pour vérifier l'aspect temporel d'une commande complète représentée par un modèle de fusion, et le temps entre les modalités, d'où la relation *hasTime* entre *Model* et *MaxCommandTime* et entre *Modality* et *MaxActiveModalityTime* (l'aspect temporel sera détaillé dans la section 4.5).

La Figure 4.16 représente un exemple de relation de propriétés d'objets et de données entre les instances. La classe *VocalModality* a une instance *Voice_Sensor_Living_Room* qui désigne un capteur du son qui se trouve dans une salle de séjour. Cette instance a une relation de type propriété d'objet avec l'instance *NoiseFromOutside* de la classe Noise. De même

Voice_Sensor_Living_Room a une autre propriété d'objet *hasUserContext* avec les deux instances *deaf* et *mute* de la classe *Handicap*. *NoiseFromOutside* a une propriété de donnée *hasNoiseLevel* qui a une valeur égale à 5. *Deaf* et *mute* ont une autre propriété de donnée *hasHandicap* qui a comme valeur false dans les deux cas.

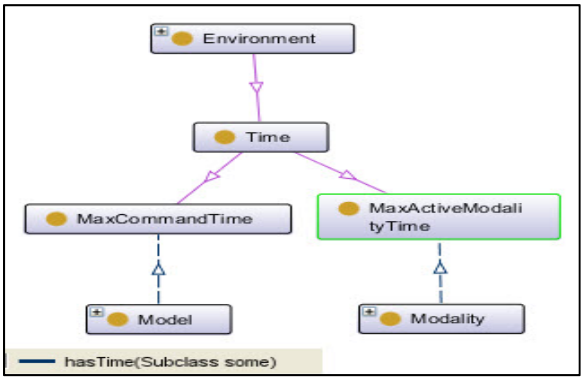


Figure 4.15 Les relations entre les classes *Modality*, *Model* avec la classe *Time*

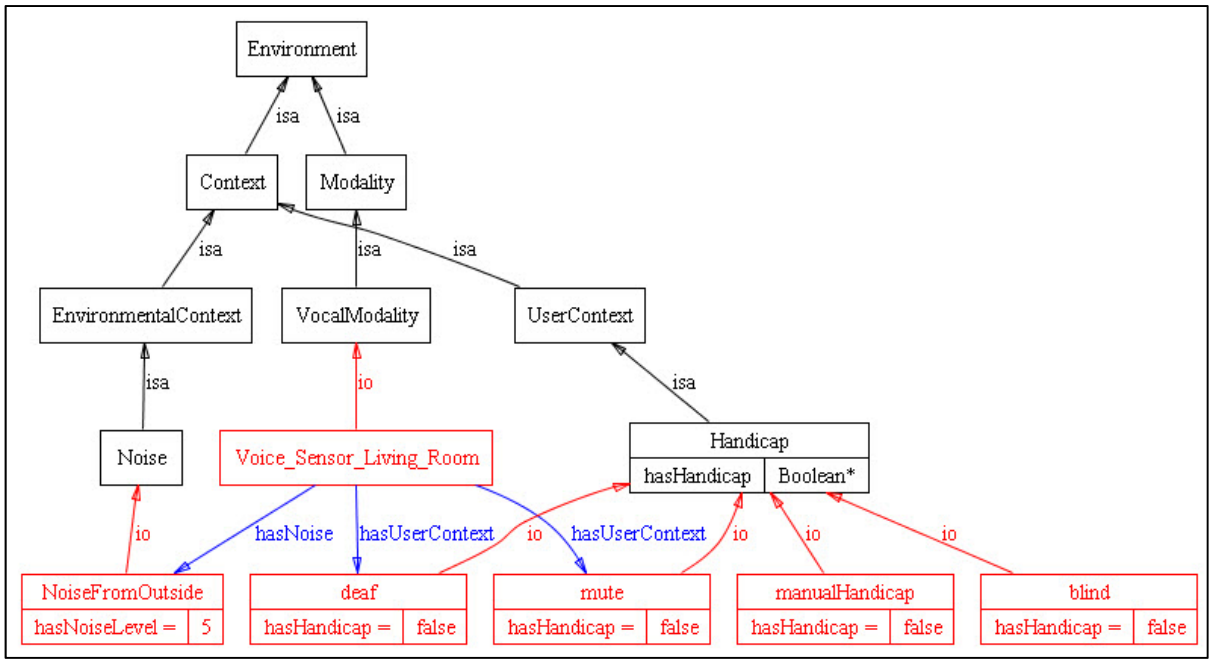


Figure 4.16 Les relations de propriétés d'objets et de données entre les instances

Ces relations sémantiques sont utilisées par les raisonneurs dans le but de comprendre l'environnement, le but est de savoir le type et le niveau de bruit détecté (type :

NoiseFromOutside, niveau : *5*) et le type d'handicap pour un utilisateur. La modalité vocale est affectée par deux types d'handicaps sourd (*deaf*) et muet (*mute*), car une personne sourde ou muette ne peut pas envoyer des commandes vocales au système. Si les deux ont une valeur fausse, et si le niveau du bruit détecté inférieur à 5 par exemple, la modalité vocale reste active, sinon elle sera désactivée. À noter que ces valeurs peuvent être modifiées selon les données captées par des capteurs spécifiques et selon le profil de l'utilisateur.

Comme conclusion, nous pouvons déduire que les relations sémantiques sont très importantes dans la description de l'environnement puisqu'elles montrent la relation entre les différents concepts. Sans ces relations sémantiques, ces concepts qui forment cet environnement restent sans aucune importance et sans aucun sens. Le Tableau 4.2 et le Tableau 4.3 montrent toutes les relations sémantiques créées dans l'ontologie.

Tableau 4.2 Les relations de propriétés d'objets déclarées dans l'ontologie

Relation Sémantique	**Description**
hasCoordinates	Relation entre les classes *Person, Place* et *Object* avec la classe *Coordinates* qui désigne qu'une personne, une place et un objet ont des coordonnées.
hasLiquid	Relation entre les deux instances *cup* et *bottle* de la classe *SmallObject* et les instances de la classe *liquid* qui désigne qu'un liquide a besoin d'un verre ou d'une bouteille
hasLocationContext	Relation entre les instances de la classe *Modality* et les instances de la classe *LocationContext* pour savoir quel type d'endroit affecte une modalité
hasModality	Relation entre les instances des sous-classes de la classe *Event* et la classe *Modality* qui désigne que chaque événement a besoin d'une modalité
hasNextM01, …, hasNextM30	Les relations entre les instances de chaque modèle défini dans l'ontologie pour savoir l'ordre que les instances doivent suivre dans chaque modèle
hasLight	Relation entre la classe *Modality* et ses sous-classes avec la classe *Light* pour savoir par quel type de lumière est affectée une modalité
hasNoise	Relation entre la classe *Modality* et ses sous-classes avec la classe

Relation Sémantique	Description
	Noise pour savoir par quel type du bruit est affectée une modalité
hasObject	Relation entre la sous-classe *MovableObject* et les différentes sous-classes de la classe *Object,* utilisée pour savoir que ce type d'objets peuvent être des liquides, des petits objets, etc.
hasTemperature	Relation entre la classe *Modality* et ses sous-classes avec la classe *Temperature* pour savoir si la température affecte une modalité
hasTime	Relation entre la classe *Model* et la classe *Time,* car il faut connaître le temps maximal d'un commande et le temps maximal accepté entre deux événements successifs
hasUserContext	Relation entre la classe *Modality* et ses sous-classes avec la classe *UserContext* pour savoir par quel type d'handicap est affectée une modalité

Tableau 4.3 Les relations de propriétés de données déclarées dans l'ontologie

Propriété de donnée	Classe	Type de données	Description
hasHandicap	Handicap	Booléen	Relation pour les instances de la classe *Handicap* de type booléen pour savoir si une personne a un handicap ou non
hasLightnessLevel	Light	Integer	Relation pour les instances de la *classe Light* de type integer pour savoir le niveau de la lumière
hasNoiseLevel	Noise	Integer	Relation pour les instances de la *classe Noise* de type integer pour connaître le niveau du bruit
hasTemperatureLevel	Temperature	Integer	Relation pour les instances de la classe *Temperature* de type integer pour connaître le niveau de la température
isAtHome	LocationContext	Booléen	Relation pour les instances de la classe *LocationContext* de type booléen pour savoir si une personne se trouve dans une maison
isAtRoad	LocationContext	Booléen	Relation pour les instances de la classe *LocationContext* de type boolien pour connaître si une personne se trouve

Propriété de donnée	Classe	Type de données	Description
			dans la rue
isAtWork	LocationContext	Booléen	Relation pour les instances de la classe *LocationContext* de type boolien pour connaîtr si une personne se trouve à son travail

Tableau 4.4 La classe *Modality* avec ses relations sémantiques décrites en syntaxe OWL

```
<!-- http://www.owl-ontologies.com/EnvironmentOnto.owl#Modality -->
  <owl:Class rdf:about="http://www.owl-
ontologies.com/EnvironmentOnto.owl#Modality">
    <rdfs:subClassOf rdf:resource="http://www.owl-
ontologies.com/EnvironmentOnto.owl#Environment"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="http://www.owl-
ontologies.com/EnvironmentOnto.owl#hasLight"/>
        <owl:someValuesFrom rdf:resource="http://www.owl-
ontologies.com/EnvironmentOnto.owl#Light"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="http://www.owl-
ontologies.com/EnvironmentOnto.owl#hasTime"/>
        <owl:someValuesFrom rdf:resource="http://www.owl-
ontologies.com/EnvironmentOnto.owl#MaxActiveModalityTime"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="http://www.owl-
ontologies.com/EnvironmentOnto.owl#hasNoise"/>
        <owl:someValuesFrom rdf:resource="http://www.owl-
ontologies.com/EnvironmentOnto.owl#Noise"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
```

```
            <owl:onProperty rdf:resource="http://www.owl-
ontologies.com/EnvironmentOnto.owl#hasLocationContext"/>
            <owl:onClass rdf:resource="http://www.owl-
ontologies.com/EnvironmentOnto.owl#LocationContext"/>
            <owl:minQualifiedCardinality
rdf:datatype="&xsd;nonNegativeInteger">1</owl:minQualifiedCardinality>
         </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
         <owl:Restriction>
            <owl:onProperty rdf:resource="http://www.owl-
ontologies.com/EnvironmentOnto.owl#hasTemperature"/>
            <owl:someValuesFrom rdf:resource="http://www.owl-
ontologies.com/EnvironmentOnto.owl#Temperature"/>
         </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
         <owl:Restriction>
            <owl:onProperty rdf:resource="http://www.owl-
ontologies.com/EnvironmentOnto.owl#hasUserContext"/>
            <owl:someValuesFrom rdf:resource="http://www.owl-
ontologies.com/EnvironmentOnto.owl#Handicap"/>
         </owl:Restriction>
      </rdfs:subClassOf>
   </owl:Class>
```

4.3 Sélection des modalités

Après avoir défini l'environnement sous forme d'une ontologie, nous focalisons sur la sélection des modalités qui sont affectées par des contextes spécifiques. Dans cette partie, les contextes déjà vus dans l'ontologie seront détaillés et les requêtes de sélection seront définies.

Le contexte est considéré comme une information sur l'environnement d'un système informatique, ou bien comme des conditions qui déterminent un événement qui nous semble sans limite apparente. En effet, qu'obtiendrons-nous si nous envisageons de décrire dans le moindre détail les composants d'un système informatique ? De plus, il peut sembler ambitieux de vouloir décrire l'ensemble des conditions qui régissent le déclenchement d'un événement donné. C'est pour ces raisons et bien d'autres que des chercheurs, non satisfaits

des définitions générales, ont essayé de définir ce terme pour permettre son utilisation dans leurs recherches. Avant de définir nos propres contextes, nous allons présenter dans l'ordre chronologique d'apparition une liste non exhaustive des définitions du contexte dans le domaine de l'informatique.

(Schilit 1994) a considéré que le contexte possède trois aspects importants qui consistent à répondre aux questions suivantes : Où es-tu ? Avec qui es-tu ? De quelles ressources disposes-tu à proximité ?

(Theimer. 1994) a défini le contexte comme la localisation, la description de personnes et d'objets dans l'entourage et les changements à ces objets.

(Brown 1995) a défini le contexte comme : « les éléments de l'environnement d'un utilisateur dont l'ordinateur a connaissance ».

(Brown 1997) a proposé un ensemble d'éléments extensibles pour caractériser le contexte dont les éléments de base sont : la localisation, l'ensemble des objets dont l'utilisateur a besoin, le temps et l'orientation spatiale (direction).

(Ryan 1997) : « les éléments du contexte sont : la localisation de l'utilisateur, l'environnement, l'identité et le temps ».

(Ward 1997): « les états des environnements possibles de l'application ».

(Pascoe 1997): « ensemble d'états physiques et conceptuels ayant un intérêt pour une entité particulière ».

(Schmidt 1999): « connaissances à propos de l'utilisateur et les états des équipements, l'entourage, la situation et la localisation ».

(Brézillon 1999): « tout ce qui n'intervient pas explicitement dans la résolution d'un problème mais le contraint ».

(Chen 2000): « ensemble des états environnementaux et paramètres qui déterminent le comportement d'une application ou dans lequel un événement de l'application se déroule et ayant un intérêt pour l'utilisateur ».

(Dey 2001): « toute information qui peut être utilisée pour caractériser la situation d'une entité. Toute entité est une personne, ou un objet qui est considéré comme significatif à l'interaction entre l'utilisateur et l'application, incluant l'utilisateur et l'application eux-mêmes ».

(Henricksen 2002) :« circonstance ou situation dans laquelle une tâche informatique se déroule».

Malgré le grand nombre de définitions existantes et les similarités (la plupart font références à la localisation et l'environnement), le mot contexte reste toujours général. Deux techniques sont utilisées par les chercheurs pour la définition du contexte : l'une est basée sur l'énumération des exemples du contexte et l'autre fait plutôt des tentatives en vue de formaliser le terme. L'importance du contexte dans le domaine de l'interaction personne-machine et les systèmes mobiles a généré des définitions centrées sur l'utilisateur et d'autres sur l'application

4.3.1.1 Définition des contextes

Les contextes dans notre recherche permettent d'optimiser le travail du système de fusion multimodale, que ce soit par l'activation ou la désactivation des modalités spécifiques afin de garantir que les événements captés soient identifiables d'une manière correcte.
Puisque le système se trouve dans l'environnement et reçoit ces commandes d'un utilisateur, trois contextes différents ont été identifiés, qui sont les suivants :

- **le contexte environnemental** : il est responsable de l'ambiance générale de l'environnement, c'est-à-dire le bruit, le niveau de la lumière et la température de l'endroit où se trouvent le système et l'utilisateur.
    - **niveau de lumière** : qui affecte l'utilisation des modalités gestuelles, manuelles et visuelles, car si le niveau de lumière est trop faible, ces modalités seront incapables

d'identifier les événements d'une manière correcte ; par exemple, une modalité gestuelle ne détectera pas un objet pointé par l'utilisateur ;

- **niveau du bruit** : qui affecte les modalités vocales, car s'il y a beaucoup de bruit dans l'environnement, cette modalité ne sera pas capable d'identifier les commandes vocales d'un utilisateur ;
- **la température** : malgré son effet indirect sur les modalités, elle reste un paramètre qui doit être pris en considération dans des cas spécifiques, comme par exemple s'il fait froid et l'utilisateur ne veut pas envoyer des commandes gestuelles ou bien tactiles ;

- **le contexte de l'utilisateur** : qui est le profil de l'utilisateur, surtout à savoir s'il a un handicap ou non, car un utilisateur aveugle ne peut ni envoyer des commandes gestuelles, ni visuelles, donc ces modalités seront désactivées ;
- **le contexte de la localisation** : pour savoir si l'utilisateur est dans une maison, dans la rue ou bien à son travail, car par exemple s'il est dans la rue et entrain de conduire une voiture, il est préférable dans ce cas-là d'envoyer des commandes vocales ou bien tactiles plutôt que des commandes visuelles qui vont disperser sa concentration sur la conduite d'une voiture.

Les Tableau 4.5, Tableau 4.6, Tableau 4.7 et Tableau 4.8 et les équations ci-dessous résument ces paramètres afin de mieux comprendre l'effet des contextes sur les modalités.

(les symboles – et + sont utilisés pour indiquer la pertinence et la non-pertinence des modalités respectivement).

Tableau 4.5 Pertinence des modalités par rapport au niveau du bruit

Modalité	Niveau bas du bruit	Niveau moyen du bruit	Haut niveau du bruit
Tactile	+	+	+
Vocale	+	+	-
Gestuelle	+	+	+
Manuelle	+	+	+
Visuelle	+	+	+

Tableau 4.6 Pertinence des modalités par rapport au niveau de la lumière

Modalité	Niveau bas de lumière	Niveau moyen de lumière	Haut niveau de lumière
Tactile	-	+	+
Vocale	+	+	+
Gestuelle	-	-	+
Manuelle	-	+	+
Visuelle	-	-	+

Tableau 4.7 Pertinence des modalités par rapport au contexte de l'utilisateur

Modalité	Handicap manuelle	Aveugle	Sourd	Muet
Tactile	-	-	+	+
Vocale	+	+	-	-
Gestuelle	-	-	+	+
Manuelle	-	-	+	+
Visuelle	+	-	+	+

Tableau 4.8 Pertinence des modalités par rapport au contexte de localisation

Modalité	Maison	Dehors	Travail
Tactile	+	+	-
Vocale	+	+	-
Gestuelle	+	+	+
Manuelle	+	+	-
Visuelle	+	+	-

$VO_m = (utilisateur \neq muet \vee utilisateur \neq \text{sourd}) \wedge (localisation \neq dehors) \wedge (niveau\ du\ bruit \neq haut\ niveau\ du\ bruit)$

$M_m = (utilisateur \neq handicap\ manuel \vee utilisateur \neq \text{aveugle}) \wedge (niveau\ de\ lumière \neq niveau\ bas\ de\ lumière)$

$VI_m = (utilisateur \neq \text{aveugle}) \wedge (localisation \neq \text{dehors}) \wedge (niveau\ de\ lumière \neq niveau\ bas\ de\ lumière)$

$G_m = (utilisateur \neq handicap\ manuel \vee utilisateur \neq \text{aveugle}) \wedge (niceau\ de\ lumière \neq niveau\ bas\ de\ lumière)$

Où VO_m : $Modalité\ vocale$, M_m : $Modalité\ Manuelle$, VI_m : Modalité visuelle et G_m : modalité géstuelle.

4.3.2 Définition des contextes dans l'ontologie

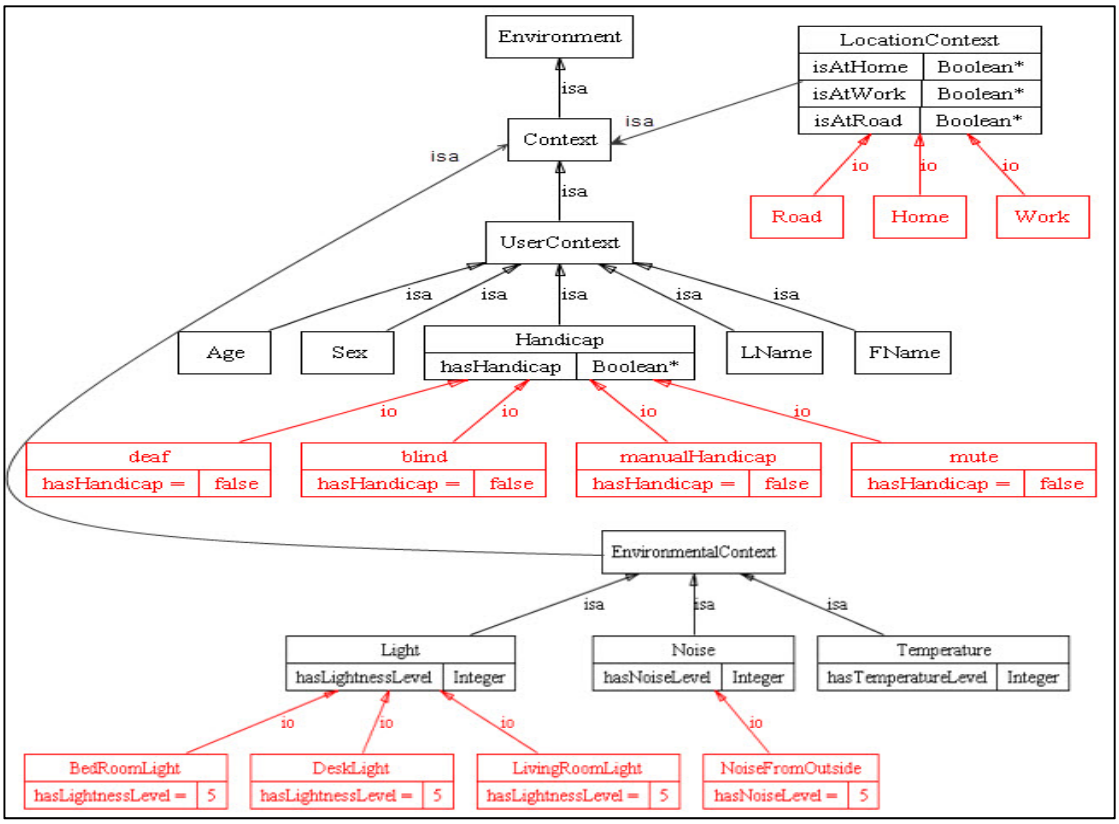


Figure 4.17 Les contextes dans l'ontologie, leurs instances et leurs propriétés d'objets et de données

La Figure 4.17 représente tous les contextes définis dans l'ontologie avec leurs instances et leurs propriétés d'objets et de données. La classe *Handicap* a les instances *deaf, blind, manualHandicap* et *mute* qui sont reliées avec des valeurs booléennes vraies ou fausses à l'aide de la propriété de données *hasHandicap.* La classe *LocationContexte* a les instances *home, work* et *road* qui sont reliées à des valeurs booléennes vraies ou fausses à l'aide des propriétés de données *isAtHome, isAtWork*, *isAtRoad*. Les deux sous-classes *Light* et *Noise* de la classe *EnvironmentalContex*t ont les instances *BedRoomLight*, *DeskLight*,

LivingRoomLight et *NoiseFromOutsides*, les instances de la classe *Light* sont reliées à leurs valeurs par une propriété de données *hasLightnessLevel* et les instances de *Noise avec hasNoiseLevel.*

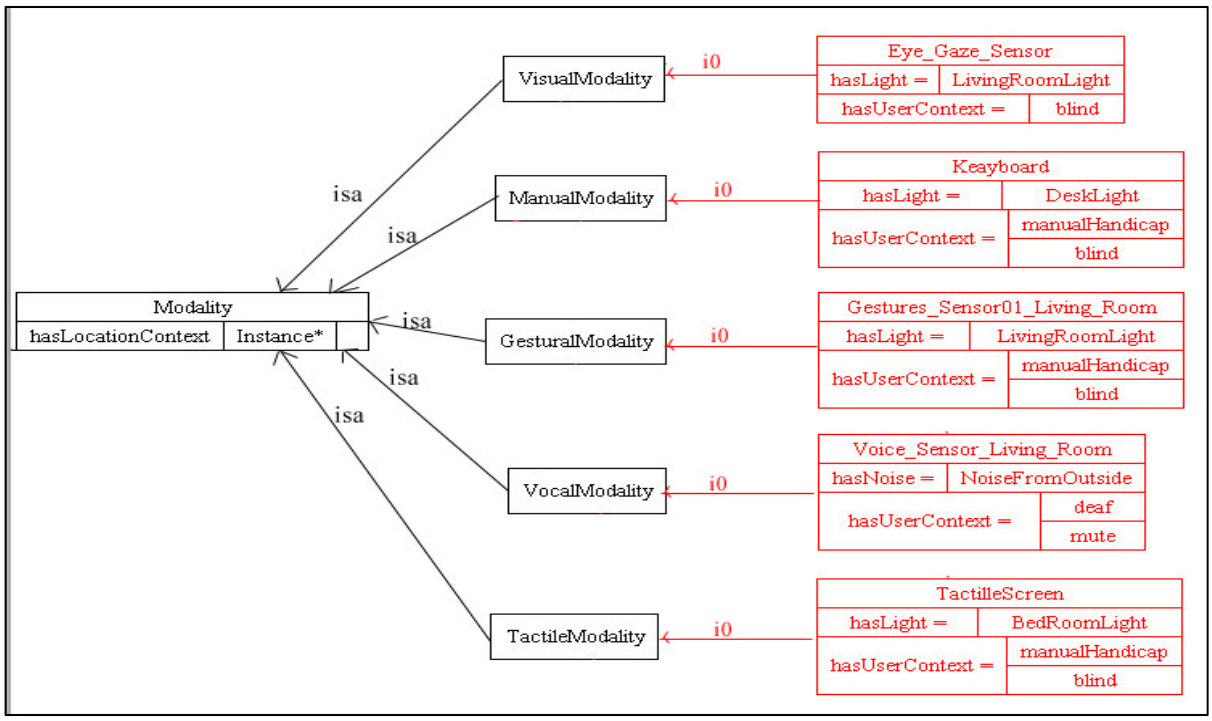


Figure 4.18 Relations entre contextes et modalités

La Figure 4.18 montre comment les modalités sont affectées par les contextes. Chaque modalité est reliée aux contextes qui affectent sa fonctionnalité. La modalité visuelle a une instance *Eye_Gaze_Sensor* qui est affectée par le niveau de la lumière du contexte environnemental et le type d'handicap du contexte de l'utilisateur. *Eye_Gaze_Sensor* est relié à l'instance *LivingRoomLight* de la classe *Light* par une propriété d'objet *hasLight* et à l'instance *blind* de la classe *Handicap* par une autre propriété d'objet *hasContextUser*. La modalité visuelle est affectée par ces deux contextes car si le niveau de la lumière est faible et/ou si l'utilisateur est aveugle, il sera difficile de capter les événements provenant de l'environnement, la modalité sera alors désactivée.

Les modalités à saisie manuelle, gestuelle et tactile sont affectées par le niveau de la lumière et le type d'handicap de l'utilisateur. S'il fait sombre et/ou si l'utilisateur est aveugle ou a un handicap manuel ces modalités seront désactivées.
La modalité vocale s'intéresse au niveau du bruit et au type d'handicap de l'utilisateur ; s'il y a beaucoup de bruit et/ou si l'utilisateur est sourd ou muet, cette modalité sera inutile et désactivée.

4.3.3 Les requêtes de sélection des modalités

Après la description des contextes et de leurs relations avec les modalités dans l'ontologie, la sélection des modalités peut être réalisée par de simples requêtes afin de sélectionner la modalité la plus appropriée par rapport aux contextes définis. Ces requêtes sont définies en utilisant le langage des requêtes SQWRL. Elles comparent les événements provenant des fichiers XML à ceux qui sont définis dans l'ontologie. À noter que SQWRL existe sous forme d'un plugin dans PROTÉGÉ. La représentation générale de la requête responsable de la sélection des modalités selon la description logique d'OWL est la suivante :

$$C(o) \sqcap C'(o') \sqcap R(o,o') \sqcap R'(o',o'') \sqcap swrlb{:}\,lessThan(o'',n) \rightarrow sqwrl{:}\,selectDistinct(o')$$

Avec *C, C'* des classes de l'ontologie, *R, R'* des propriétés d'objets et $o, o'\ et\ o''$ des instances.
La requête (1) est un exemple qui sélectionne dans l'ontologie toutes les modalités qui doivent être désactivées si le niveau de lumière est plus petit que 4 lux (lux : unité de lumière).

Modality(?m) ∧ Light(?l) ∧ hasLight(?m, ?l) ∧ hasLightnessLevel(?l, ?level) ∧ swrlb:lessThan(?level, 4) → sqwrl:selectDistinct(?m) *(1)*

Tableau 4.9 Description textuelle de la requête (1)

IF *"l" IS A Light*

```
        AND "l" HAS LightnessLevel "level" WHERE "level" IS LESS THAN 4
AND IF
        "m" IS A Modality
        AND "m" HAS Light "l"
THEN
        "m" sqwrl:selectDistinct
```

$$Modality(?m) \wedge Handicap(?h) \wedge hasUserContext(?m, ?h) \wedge hasHandicap(?h, ?han) \wedge swrlb:equal(?han, 1) \rightarrow sqwrl:selectDistinct(?m) \quad (2)$$

La requête (2) est un exemple qui sélectionne dans l'ontologie les modalités qui doivent être désactivées selon le type d'handicap de l'utilisateur.

Tableau 4.10 Description textuelle de la requête (2)

```
IF
        "m" IS A Modality
        AND "m" HAS UserContext "h"
                WHERE "h" IS A Handicap
                AND "h" HAS Handicap "han" WHERE "han" IS EQUAL TO 1
THEN
        "m" sqwrl:selectDistinct
```

$$Modality(?m) \wedge LocationContext(?loc) \wedge hasLocationContext(?m, ?loc) \wedge isAtRoad(?loc, ?loca) \wedge swrlb:equal(?loca, 1) \rightarrow sqwrl:selectDistinct(?m) \quad (3)$$

La requête (3) est un exemple qui sélectionne dans l'ontologie les modalités qui doivent être désactivées si l'utilisateur se trouve dans la rue

Tableau 4.11 Description textuelle de la requête (3).

```
IF
        "loc" IS A LocationContext
        AND "loc" HAS VALUE "loca" FOR isAtRoad WHERE "loca" IS EQUAL TO 1
AND IF
        "m" IS A Modality
        AND "m" HAS LocationContext "loc"
THEN
        "m" sqwrl:selectDistinct
```

Modality(?m) ∧ Noise(?l) ∧ hasNoise(?m, ?l) ∧ hasNoiseLevel(?l, ?level) ∧ swrlb:greaterThan(?level, 6) → sqwrl:selectDistinct(?m) (4)

La requête (4) est un exemple qui sélectionne dans l'ontologie les modalités qui doivent être désactivées si le niveau de bruit est plus grand que 6 dB. La Figure 4.19 représente l'exécution de la requête (4) dans PROTÉGÉ et le résultat obtenu.

Tableau 4.12 Description textuelle de la requête (4)

IF *"l" IS A Noise* *AND "l" HAS NoiseLevel "level" WHERE "level" IS GREATER THAN 6* *AND IF* *"m" IS A Modality* *AND "m" HAS Noise "l"* *THEN* *"m" sqwrl:selectDistinct*

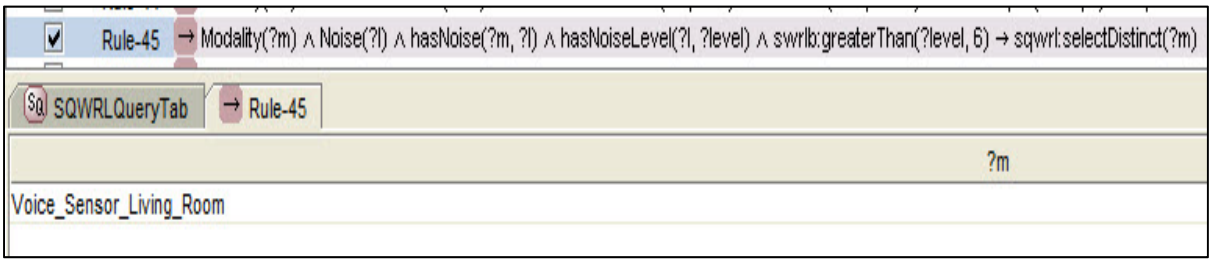


Figure 4.19 Exécution de la requête (4) sur PROTÉGÉ

Tableau 4.13 Exemple de fichiers XML pour le contexte de l'environnement et de l'utilisateur

<!—contexte de l'environnement--> <events> <Command> <GModality> <type>Gestures_Modality_Living_Room</type> <LightLevel>6</LightLevel> </GModality> <VModality> <type>Gestures_Modality_Living_Rom</type> <NoiseLevel>3</NoiseLevel> </VModality></Command></events>	<!—contexte de l'utilisateur--> <events> <contexts> <UserContext <blind>0</blind> <manualHandicap>false</manualHandicap> <deaf>0</deaf> <mute>0</mute> </UserContext> </contexts> </events>

4.4 Sélection des modèles

Comme mentionné auparavant, nous avons défini une trentaine de modèles de fusion dans l'ontologie. Chaque modèle représente une combinaison correcte des événements pour différentes commandes d'un utilisateur. Un exemple de modèle est présenté dans la Figure 4.20. Il montre les différents événements qui forment le Model01 et l'ordre suivi par l'utilisation de la propriété d'objet *hasNextM01*. Ce modèle est constitué par :

ActionForMovableObject →SmallObject→IntendedLocation→AverageObject

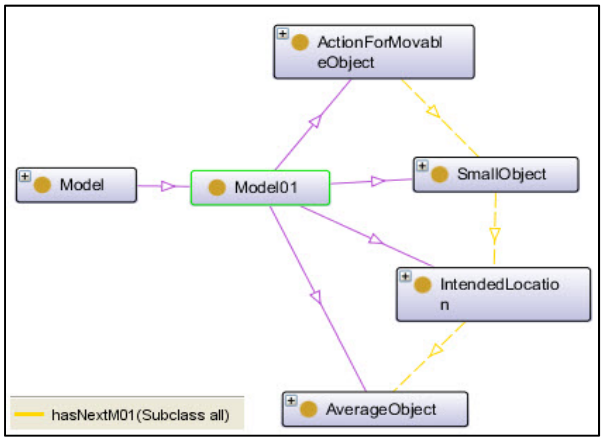


Figure 4.20 Le modèle Model01

Le Tableau 4.14 montre les modèles décrits dans l'ontologie ; chacun est illustré par un exemple.

Tableau 4.14 Les différents modèles de l'ontologie

Classe	***Modèle***	***Exemple***
Model01	*ActionForMovableObject→SmallObject →IntendedLocarion→AverageObject*	*put the cup on that table*
Model02	*ActionForMovableObject →AverageObject →IntendedLocation*	*Put the table there*
Model04	*ActionForMovableObject → MovableObject*	*Drop the key*
Model05	*ActionForMovableObject→MovableObject*	*Give the pen to father*

Classe	***Modèle***	***Exemple***
	→Person	
Model06	*ActionForMovableObject→ MovableObject → IntendedLocation →Location*	*Take the dish to the kitchen*
Model07	*ActionForMovableObject→ MovableObject → IntendedLocation*	*Get the mobile here*
Model08	*ActionForMovableObject→ IntendedObject → IntendedLocation*	*Put that here*
Model09	*ActionForMovableObject→ IntendedObject → IntendedLocation→MovableObject*	*Put that under the table*
Model10	*ActionForMovableObject→ IntendedObject*	*Take this*
Model11	*ActionForMovableObject→IntendedLocation → MovableObject*	*Clean under the cup*
Model12	*ActionForPerson→Person*	*Call mother, Call her*
Model13	*ActionForPerson→Person→ MovableObject*	*Give me the pen, Give my brother the pen*
Model14	*ActionForPerson→Person→ IntendedLocation →Location*	*Take my nephew to the bedroom, Take him to the bedroom*
Model15	*ActionForPerson→Person→ IntendedLocation*	*take me here*
Model16	*ActionForPerson→Person→ IntendedLocation→Person*	*Take me to my son*
Model17	*ActionForPerson →Person→IntendedObject*	*Give me that*
Model18	*ActionForNonMovableObject→ IntendedObject*	*Clean that*
Model19	*ActionForNonMovableObject→ Location*	*Clean the kitchen*
Model120	*ActionForNonMovableObject*	*Clean the wall*

Classe	***Modèle***	***Exemple***
	→NonMovableObject	
Model21	*ActionForNonMovableObject →IntendedLocation→NonMovableObject*	*Clean under the window*
Model22	*ActionForNonMovableObject →IntendedObject→NonMovableObject*	*Open that door*
Model23	*ActionForNonMovableObject →NonMovableObject→Location*	*Close the door of the bathroom*
Model24	*ActionForLocation→MovableObject*	*Find the ball*
Model25	*ActionForLocation→MovableObject →IntendedLocation→Location*	*Find a cup in the kitchen*
Model26	*ActionForLocation→Person*	*Find him, find father*
Model27	*ActionForLocation→IntendedLocation*	*Search here*
Model28	*ActionForLocation→IntendedLocation →MovableObject*	*Search under the table*
Model29	*ActionForLocation→IntendedLocation →NonMovableObject*	*Search behind the wall*
Model30	*ActionForMovableObject→Person →ObjectForLiquid→Liquid*	*Give me a cup of water*

La Figure 4.21 représente un exemple de relations entre les instances de la classe *Model02*. Le modèle02 est formé par trois sous-classes *ActionForMovableObject*, *AverageObject* et *IntendedLocation* qui ont comme instances *get*, *box* et *here* respectivement. Une relation *hasNextM02* de type propriété d'objet a été créée pour définir l'ordre des instances. Cela veut dire qu'après *get* il y a *box* suivie par *here.* Ce type de relations aide à mieux comprendre l'ordre des événements dans une commande spécifique.

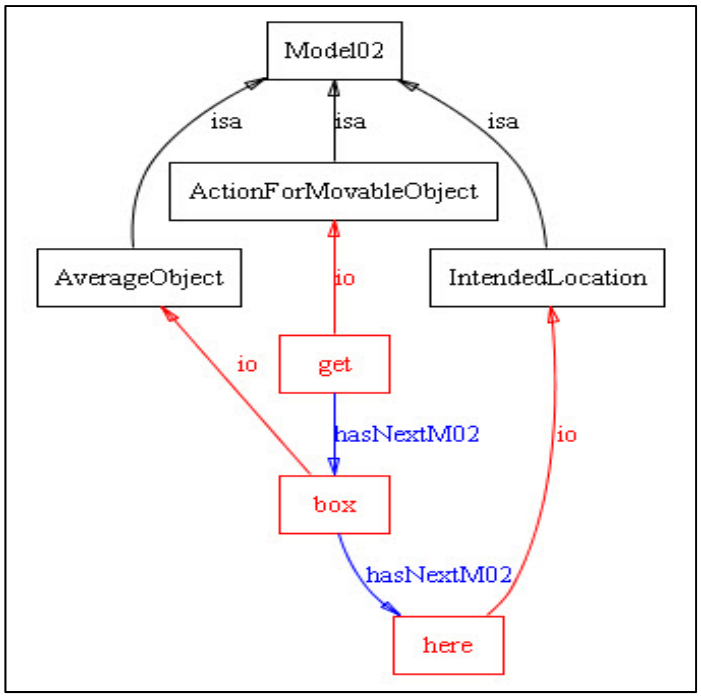


Figure 4.21 Relations entre les instances de la classe Model02

4.4.1 Requêtes de sélection des modèles

Une fois les modèles définis dans l'ontologie, il suffit de déclarer les requêtes SQWRL pour sélectionner un modèle approprié à une combinaison d'événements détectés par les modalités. Chaque modèle a sa propre requête. Une trentaine de requêtes ont été créées pour sélectionner chaque modèle. La représentation générale d'une requête de sélection du modèle selon la description logique d'OWL est comme suit :

$$C_1(o_1) \sqcap \ldots \sqcap C_n(o_n) \sqcap R_1(o_1, o_2) \sqcap \ldots \sqcap R_n(o_n, o_{n+1}) \\ \sqcap tbox{:}\, isDirectSuperClassOf\big((o', C_1) \sqcap \ldots \sqcap (o', C_n)\big) \\ \rightarrow sqwrl{:}\, selectDistinct(o')$$

Avec *C* une classe de l'ontologie, *R* une propriété d'objet et o, o' des instances.

La requête (5) est un exemple qui sélectionne le modèle 05. Dans cet exemple, *x, y* et *z* sont des événements détectés par les modalités. La requête demande à l'ontologie de vérifier si *x,*

y et *z* sont des instances des classes *ActionForMovableObject, MovableObject* et *Person* respectivement, de vérifier si 1) *y* vient après *x*, 2) *z* vient après *y* et 3) si ces trois classes sont reliées directement à une super classe *m* qui désigne un modèle. Si les conditions sont vérifiées, la requête sélectionne le modèle approprié qui est dans ce cas le modèle 05. La Figure 4.22 représente l'exécution de la requête (5) dans PROTÉGÉ et le résultat obtenu.

ActionForMovableObject(?x) ∧ MovableObject(?y) ∧ Person(?z) ∧ hasNextM05(?x, ?y) ∧ hasNextM05(?y, ?z) ∧ tbox:isDirectSuperClassOf(?m, ActionForMovableObject) ∧ tbox:isDirectSuperClassOf(?m, MovableObject) ∧ tbox:isDirectSuperClassOf(?m, Person) → sqwrl:selectDistinct(?m) *(5)*

Tableau 4.15 Description textuelle de la requête (5)

IF
"y" IS A MovableObject
AND "y" HAS NextM05 "z"
WHERE "z" IS A Person
AND IF
"x" IS AN ActionForMovableObject
AND "x" HAS NextM05 "y"
AND IF
"m" tbox:isDirectSuperClassOf "ActionForMovableObject"
AND "m" tbox:isDirectSuperClassOf "MovableObject"
AND "m" tbox:isDirectSuperClassOf "Person"
THEN
"m" sqwrl:selectDistinct

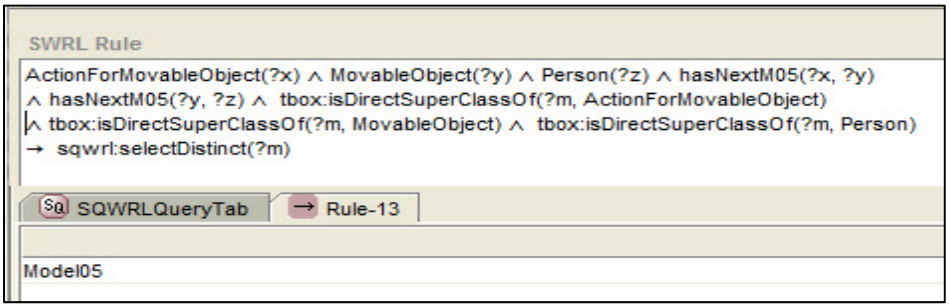


Figure 4.22 Exécution de la requête (5) dans PROTÉGÉ

4.5 La fusion multimodale

Après la description de l'environnement, la définition des requêtes de sélection des modalités et des modèles, le moteur de fusion peut intervenir pour fusionner les événements détectés. Dans cette partie, nous proposons notre algorithme de fusion. Celui-ci est basé sur des préconditions qui doivent être testées afin de pouvoir fusionner les événements. Le mode de fonctionnement est le suivant : si les préconditions des événements des modalités A et B sont présentes, C contiendra les événements fusionnés à partir de A et B (voir Figure 4.23).

Figure 4.23 Principe de la fusion

4.5.1 Les pré-conditions

Les pré-conditions sont des paramètres et des contraintes qui doivent être testés par le moteur de fusion afin de décider si un ensemble d'événements forme une commande compréhensible. Il existe quatre pré-conditions à tester :

La vérification du vocabulaire

Elle consiste à vérifier si les événements détectés par les modalités sont définis sous forme d'instances dans l'ontologie. Cela peut être fait par une simple requête SQWRL. Celle-ci reçoit l'événement et le compare avec les instances existantes. Si elle le trouve, la vérification est donc faite. Sinon, l'évènement est rejeté. Cette vérification est importante, non seulement car elle reconnait les événements provenant de l'environnement, mais également car elle peut éliminer les événements indésirables qui n'ont pas de sens par rapport au système. Un exemple d'événement éliminé est le son provenant de la fermeture d'une porte.

$$C(o) \sqcap tbox{:}\, equalTo(o, x) \rightarrow sqwrl{:}\, selectDistinct\ (o)$$

Avec *C* une classe de l'ontologie, *x* un événement de l'environnement et *o* une instance. La requête (6) demande l'ontologie de vérifier si l'événement "pen" est une instance dans l'ontologie.

$$Event\ (?m) \wedge tbox{:}equalTo(?m, pen) \rightarrow\ sqwrl{:}selectDistinct(?m) \qquad (6)$$

Tableau 4.16 Description textuelle de la requête (6)

IF "m" IS AN Event AND "m" tbox:equalTo "pen" THEN "m" sqwrl:selectDistinct

La vérification de l'ordre

Elle consiste à vérifier si un ensemble d'événements peut être identifié comme un modèle déclaré dans l'ontologie. Pour cela il suffit de tester les événements dans les requêtes de sélection des modèles (voir section 4.4) pour savoir à quel modèle ils appartiennent. Les modèles sont divisés en quatre catégories :

- des modèles pour les objets mobiles ;
- des modèles pour les objets non mobiles ;
- des modèles pour les personnes ;
- des modèles pour les emplacements.

La vérification sémantique

Elle consiste à vérifier les composants de l'ontologie tels que les relations entre les concepts et les instances, les relations entre les instances et leurs valeurs et les règles SWRL. Cette vérification peut être faite en utilisant le moteur d'inférence PELLET et JESS du PROTÉGÉ afin de vérifier la cohérence, la taxonomie et les inférences dans l'ontologie. La vérification sémantique est très importante, sinon le système considérera n'importe quelle commande qui obéit à un modèle spécifique comme étant une commande correcte. Par exemple, si

l'utilisateur demande au système d'apporter de l'eau «*bring me water* », le système doit savoir que, pour répondre à cette requête, il lui faut une bouteille ou bien un verre d'eau. Donc le système doit savoir que la commande est en réalité « *bring me a cup of water* » ou « *bring me a bottle of water* » et elle obéit à un modèle *ActionForMovableObject→Person→ObjectForLiquid→Liquid* et non au modèle *ActionForMovableObject→Person→MovableObject*.

$$Liquid(?y) \wedge SmallObject(?x) \wedge hasLiquid(?x, ?y) \rightarrow hasNextM30(?x, ?y) \quad (7)$$

La règle (7) montre que si l'utilisateur demande un liquide, celui-ci doit être ramené soit dans une bouteille soit dans un verre. Ceux-ci sont de petits objets. Le Tableau 4.17 et la Figure 4.24 montrent la description textuelle et l'exécution de la requête (7) sur PROTÉGÉ respectivement.

Tableau 4.17 Description textuelle de la requête (7)

IF "x" IS A SmallObject AND "x" HAS Liquid "y" WHERE "y" IS A Liquid THEN "x" HAS NextM30 "y"

La vérification temporelle

Elle permet de décider si la fusion des événements détectés est possible. L'aspect temporel est très important, car le système ne peut pas attendre indéfiniment les commandes d'un utilisateur. Par exemple, si l'utilisateur envoie un événement et fait une interruption, le système attend pendant un certain intervalle de temps et réagit par la suite, soit en annulant l'événement, soit en demandant à l'utilisateur de renvoyer un ensemble d'événements qui peut être une commande complète. Similairement, le temps entre les événements et très important car il permet de prédire si les événements correspondent à une même commande. La Figure 4.25 montre quelques-uns des cas possibles.

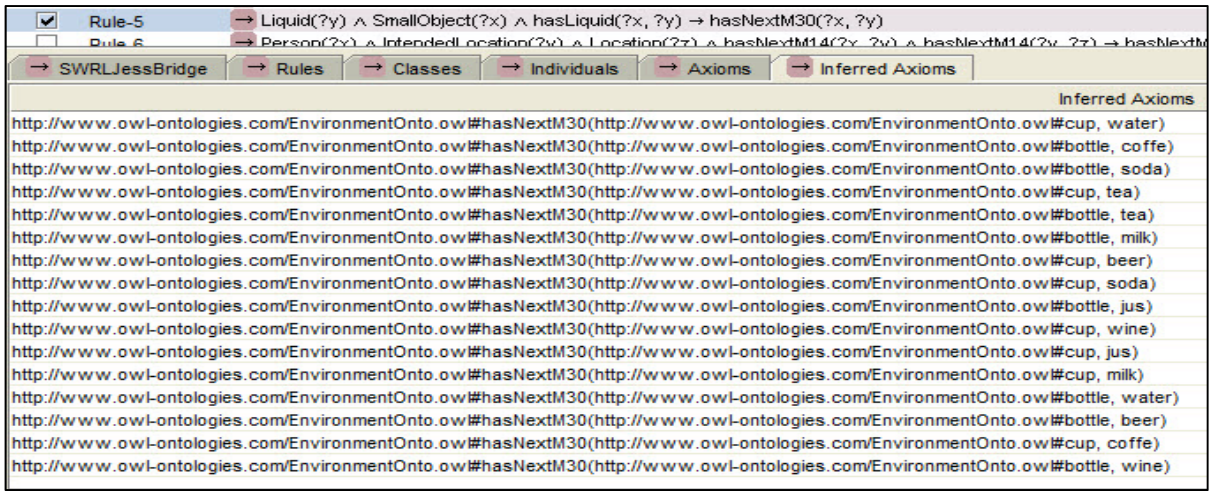


Figure 4.24 Exécution de la requête (7) sur PROTÉGÉ

Cas A :

Une modalité M_1 arrive à T_1M_1 et se termine à T_2M_1. Une autre modalité M_2 arrive à T_1M_2 et se termine à T_2M_2. Cependant, ΔT qui est la différence du temps entre le T_1M_2 et le T_2M_1est supérieur à $T_{maxActiveModalityTime}$. Donc, nous sommes dans un cas monomodal. Nous sommes en présence de deux fonctions : f_1: $Com_1 = M_1$ et f_2: $Com_2 = M_2$ avec Com_1 et Com_2 deux commandes différents.

Cas B :

Une modalité M_1 arrive à T_1M_1 et se termine à T_2M_1. Une autre modalité M_2 arrive à T_1M_2 et se termine à T_2M_2, mais ΔT est inférieure à $t_{maxActiveModalityTime}$. Dans ce cas, une vérification des pré-conditions est nécessaire pour déterminer si la fusion est possible. Dans le cas affirmatif, la fonction résultante sera: f: $C = M_1 + M_2$.

Cas C :

Deux modalités M_1 et M_2 arrivent dans le même intervalle de temps. Ici, la fusion est évidente. Les autres pré-conditions seront nécessaires pour savoir à quel type de modèle appartient cet ensemble d'événements, et la fonction sera f: $C = M_1 + M_2$.

Cas D :

Une modalité M_1 arrive à T_1M_1 et se termine à T_2M_1 et après rien ne se passe. Deux cas sont possibles : 1) soit l'utilisateur n'a pas envoyé un autre événement au système avec une autre modalité, donc nous sommes dans un cas monomodal, 2) soit il a dépassé le temps maximal d'une commande. Dans ce cas, le système rejette la commande et il la considère comme une commande incomplète qui n'a pas de sens, et la fusion n'aura pas lieu.

4.5.2 Algorithme de fusion

Maintenant que tous les modules nécessaires à la fusion sont définis, que ce soit la définition de l'environnement, la sélection des modalités qui détectent les événements et les pré-conditions, un algorithme doit être défini pour organiser ce travail. L'algorithme de fusion a pour rôle de tester les événements par rapport aux conditions précisées dans la section précédente. Donc le moteur de fusion doit vérifier si les événements détectés par les modalités et qui sont envoyés à partir des fichiers XML, existent dans l'ontologie, respectent la sémantique de l'environnement, obéissent à un modèle déclaré dans l'ontologie et respectent l'aspect temporel. Le Tableau 4.18 représente l'algorithme de fusion. Le principe est que le moteur de fusion reçoit les événements par des fichiers XML et fait les tests suivants :

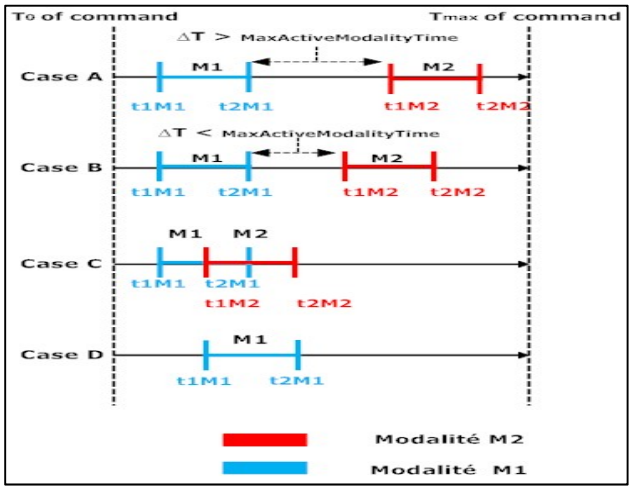


Figure 4.25 Aspect temporel entre les événements

- la comparaison de chaque événement avec les instances déclarées dans l'ontologie en appliquant la requête définie dans la vérification du vocabulaire. S'il trouve au moins deux événements provenant de deux modalités différentes, il passe à la seconde précondition, et s'il ne trouve aucun d'eux, dans ce cas il n'y aura pas de fusion. Les événements non trouvés sont éliminés par le système ;
- la vérification sémantique, en exécutant le raisonneur PELLET pour tester la sémantique de l'ontologie. Si les événements respectent la sémantique, le moteur passe à la troisième précondition, sinon, il n'y aura pas de fusion car les événements sont sémantiquement faux;
- la vérification des modèles de fusion des événements, en appliquant les requêtes de sélection des modèles, pour déterminer à quel modèle appartient une série d'événements. Si le moteur n'arrive pas à trouver le modèle, il n'y aura pas de fusion ;
- la vérification du temps sera testée, pour savoir si une série d'événements appartiennent à une seule commande. Ici, deux conditions seront testées, la première est de vérifier que le temps ne dépasse pas le temps maximal d'une commande, le deuxième est de vérifier que le temps entre la fin d'un événement et le début d'un autre est respecté.

Si toutes les préconditions sont vérifiées, le moteur de fusion fusionne les événements selon le modèle sélectionné afin d'obtenir une commande complète. Si l'une des préconditions n'est pas respectée, la fusion n'aura pas lieu.

Tableau 4.18 Algorithme de fusion

```
1   Recupération des événements à partir du fichier XML
2   //vérification si chaque événement est définie comme instance dans l'ontologie
3   If (la requête ne retourne pas vide) {
4   //Afficher la classe de chaque événement identifié
5   // vérification de la sémantique
6   Exécution du raisonneur PELLET
7   If (sémantique= true) {
8   // vérification du modèle
9   If (événement du type actionForMovableObject) {
10  Exécution des requêtes de la partie actionForMovableObject
11  If (modèle trouvé) {
1   Afficher le modèle}
```

13 If (événement du type actionForNonMovableObject) {
14 Exécution des requêtes de la partie actionForNonMovableObject
15 If (modèle trouvé) {
16 Afficher le modèle}
17 If (événement du type actionForPerson) {
18 Exécution des requêtes de la partie actionForPerson
19 If (modèle trouvé) {
20 Afficher le modèle}
21 If (événement du type actionForPlace) {
22 Exécution des requêtes de la partie actionForPlace
23 If (modèle trouvé) {
24 Afficher le modèle}
25 //vérification du temps;
26 T_0 est le temps de début du premier événement
27 T_f est le temps d'arrêt du dernier événement
28 If ($T_f - T_0$) <=MaxCommandTime) {//si la différence du temps es plus petit ou égal au temps maximal d'une commande défini dans l'ontologie
29 // les événements ont respecté le temps maximal d'une commande
30 //vérification du temps entre les événements
31 $\Delta T = T_{0Mi+1} - T_{fMi}$
32 If (ΔT <= MaxActiveModalityTime) {//si la différence du temps entre deux événements est plus petit ou égal au temps maximal entre deux événements défini dans l'ontologie
33 // les événements ont respecté le temps maximal entre eux.
34 } // fin de if de ligne 32
35 }// fin de if de ligne 28
36 Else pas de fusion // la condition temporelle n'a pas été respectée
37 }// fin de if de ligne 20
38 }// fin de if de ligne 16
39 }// fin de if de ligne 12
40 }// fin de if de ligne 8
41 Else pas de fusion // le modèle n'a pas été identifié
42 }// fin de if de ligne 7
43 Else pas de fusion // la sémantique n'a pas été respectée
44 }// fin de if de ligne 3
45 Else pas de fusion // le vocabulaire n'a pas été identifié
46 Affichage de la commande fusionnée selon le modèle détecté;

4.5.3 Scénario "get that here"

Dans cette partie, nous présentons le scénario « get that here » pour montrer comment appliquer l'algorithme de la fusion. Pour cela, il faut définir les fichiers XML responsables des événements et des contextes. Le Tableau 4.19 et le
Tableau 4.21 représentent les fichiers XML pour le contexte de l'environnement, pour le contexte de l'utilisateur et pour les événements respectivement.

Le premier fichier montre que le type de lumière présent dans l'endroit où se trouve le système est une lumière dans la chambre de séjour et son niveau est égal à 6 avec un bruit provenant de la télévision de la chambre de séjour avec un niveau 3. Le deuxième fichier montre que l'utilisateur n'a aucun type d'handicap. Donc nous pouvons déduire que le système se trouve dans une chambre lumineuse et calme et que l'utilisateur ne présente aucun handicap.

Tableau 4.19 Fichier XML pour le contexte environnemental

```
<events>
   <Command>
          <Light>
                    <type>Light_Living_Room</type>
                     <LightLevel>6</LightLevel>
           </Light>
          <Noise>
                     <type>TVNoise_Living_Room</type>
                     <NoiseLevel>3</NoiseLevel>
          </Noise>
  </Command>
</events>
```

Tableau 4.20 Fichier XML pour le contexte de l'utilisateur

```
<events>
<contexts>
          <UserContext>
                    <blind>0</blind>
                    <manualHandicap>false</manualHandicap>
                    <deaf>0</deaf>
```

```
                <mute>0</mute>
        </UserContext>
</contexts>
</events>
```

En appliquant les requêtes suivantes :

*Modality (?m) ∧ Light(*Light_Living_Room*) ∧ hasLight(?m,* Light_Living_Room*) ∧ hasLightnessLevel(*Light_Living_Room*, 6) ∧ swrlb:lessThan(6, 5) → sqwrl:selectDistinct(?m)*

Aucune modalité affectée par la lumière ne doit être désactivée car le niveau de la lumière est plus grand que 5 (le 5 est défini lors de la configuration du système durant la première utilisation).

*Modality(?m) ∧ Noise(*TVNoise_Living_Room*) ∧ hasNoise(?m,* TVNoise_Living_Room*) ∧ hasNoiseLevel(*TVNoise_Living_Room*, 3) ∧ swrlb:greaterThan(3, 6) → sqwrl:selectDistinct(?m)*

Aucune modalité affectée par le bruit ne doit être désactivée car le niveau de bruit est plus petit que 6 (le 6 est défini lors de la configuration du système durant la première utilisation).

Modality(?m) ∧ Handicap(?h) ∧ hasUserContext(?m, ?h) ∧ hasHandicap(?h, ?han) ∧ swrlb:equal(?han, 1) → sqwrl:selectDistinct(?m)

Aucune modalité n'est affectée par un type d'handicap puisqu'ils ont tous une valeur de 0. Nous pouvons donc déduire que toutes les modalités peuvent être utilisées.

Le troisième fichier montre les différents événements détectés par une modalité gestuelle et une modalité vocale (Tableau 4.22).

Tableau 4.21 Fichier XML pour les événements

```
<events>
  <Comand>
  <Modality>
        <type>VocalModality</type>
        <event>get</event>
        <startTime>0.1</startTime>
        <endTime>0.3</endTime>
    </Modality>
 <Modality>
        <type>VocalModality</type>
        <event>that</event>
        <startTime>0.4</startTime>
        <endTime>0.6</endTime>
    </Modality>
    <Modality>
        <type>GestureModality</type>
        <event>(9,15,8)</event>
        <startTime>0.5</startTime>
        <endTime>0.6</endTime>
    </Modality>
<Modality>
        <type>VocalModality</type>
        <event>hello</event>
        <startTime>0.7</startTime>
        <endTime>0.8</endTime>
    </Modality>
 <Modality>
        <type>VocalModality</type>
        <event>here</event>
        <startTime>0.9</startTime>
        <endTime>0.11</endTime>
    </Modality>
```



```
        <Modality>
        <type>GestureModality</type>
        <event>(11,20,10)</event>
        <startTime>0.15</startTime>
        <endTime>0.22</endTime>
    </Modality>
    <Modality>
        <type>VocalModality</type>
        <event>hi</event>
        <startTime>0.75</startTime>
        <endTime>0.78</endTime>
    </Modality>
    <Modality>
        <type>VocalModality</type>
        <event>zzz</event>
        <startTime>0.95</startTime>
        <endTime>0.97</endTime>
    </Modality>
    <Modality>
        <type>VocalModality</type>
        <event>no</event>
        <startTime>1.5</startTime>
        <endTime>2.7</endTime>
    </Modality>
  </Comand>
</events>
```

Tableau 4.22 Événements détectés

Modalité	**Événement**	**Temps du début**	**Temps d'arrêt**
VocalModality	Get	0.1	0.3
VocalModality	That	0.4	0.6
GesturalModality	(9,15,8)	0.65	0.69

Modalité	**Événement**	**Temps du début**	**Temps d'arrêt**
VocalModality	Hello	0.7	0.8
VocalModality	Here	0.9	1.1
GesturalModality	(11,20,10)	1.5	2
VocalModality	Hi	0.75	0.78
VocalModality	Zzzz (bruit d'une porte)	0.95	0.97
VocalModality	No	1.5	2.7

Le vocabulaire est vérifié par l'application de la requête suivante pour chaque événement et le résultat est représenté dans le Tableau 4.23 :

Event (?m) ∧ tbox:equalTo(?m, événement) → sqwrl:selectDistinct(?m)

Tableau 4.23 Résultat de la vérification du vocabulaire

événement	classe
Get	ActionForMovableObject
that	IntendedObject
(9,15,8)	Coordinates
Here	IntendedLocation
(11,20,10)	Coordinates

Les événements hello, hi, zzzz et no, sont éliminés car ils ne sont pas définis sous forme d'instances dans l'ontologie et le système n'a pas pu récupérer leurs classes.

Le raisonneur est exécuté et il n'a pas détecté de contradiction au niveau de la sémantique entre les événements.

La sélection du modèle est limitée aux requêtes concernant les *ActionForMovableObject*. Le moteur de fusion essaye de trouver la requête qui définit un tel ordre *ActionForMovableObject→ IntendedObject→ IntendedLocation* qui est la suivante :

ActionForMovableObject(?x) ∧ IntendedObject(?y) ∧ IntendedLocation(?z) ∧ hasNextM08(?x, ?y) ∧ hasNextM08(?y, ?z) ∧ tbox:isDirectSuperClassOf(?m, ActionForMovableObject) ∧ tbox:isDirectSuperClassOf(?m, IntendedObject) ∧ tbox:isDirectSuperClassOf(?m, IntendedLocation) → sqwrl:selectDistinct(?m)

L'exécution de cette requête nous donne le modèle 8 (Model08) de l'ontologie. Donc cette série d'événements a bien été identifiée.

Finalement, il reste à vérifier l'aspect temporel des événements. Deux opérations sont nécessaires : a) la soustraction du temps d'arrêt du dernier événement et le temps du début du premier événement pour savoir si le temps maximal d'une commande est respecté et b) la soustraction du temps entre deux événements consécutifs afin de savoir si le temps maximal entre les événements est respecté. Le temps maximal d'une commande et le temps maximal entre les événements sont définis par rapport à l'utilisateur lors de la première configuration du système. Supposons que le temps maximal d'une commande est défini à 10 s et le temps maximal entre deux événements est de 3 s.

2 - 0.1 = 1.9 s < 10 s : le temps maximal d'une commande est respecté.
0.4 - 0.3 = 0.1s < 3 s : temps respecté entre that et get
0.65 – 0.6 = 0.05s < 3 s : temps respecté entre (9, 15, 8) et that
0.9 – 0.69 = 0.21 s < 3 s : temps respecté entre here et (9, 15, 8)
1.5 – 1.1 = 0.5 s < 3 s : temps respecté entre (11, 20, 10) et here

En conclusion, l'aspect temporel est bien respecté et toutes les autres conditions sont respectées. Donc la fusion a eu lieu et le moteur de fusion a obtenu la commande « get that here » et les coordonnées de *that* sont (9, 15, 8) et de *here* sont (11, 20, 10).

Maintenant, reprenons les mêmes contextes de scénario que le précédent et supposons que le système détecte deux événements *get* et *me* qui sont définis dans l'ontologie et qui appartiennent aux deux classes *ActionForMovableObjet* et *IntendedPerson* respectivement. La sémantique ici est respectée. Il n'y a pas de contradiction et les relations sont consistantes. Par contre, lorsque le système sélectionne le modèle, il ne trouvera pas le modèle associé à un tel ordre, car en réalité, cette série d'événements n'est pas complète. La commande *get me* n'a pas de sens. Donc, le moteur de fusion les éliminera et ne continuera pas la vérification des autres conditions. Ainsi, la fusion n'aura pas lieu.

4.6 Conclusion

Ce chapitre a présenté la conception des différents composants de l'architecture proposée pour la fusion multimodale, que ce soit la modélisation de l'environnement avec toutes ses contraintes dans une ontologie, la définition d'un vocabulaire, les relations sémantiques, les contextes et les modèles, la sélection des modalités et des modèles de fusion. Nous avons finalement exposé un algorithme de la fusion multimodale.
Des requêtes ont été définies par rapport aux contextes de l'environnement, de l'utilisateur et de la localisation afin de faciliter la sélection des modalités. Des requêtes de la sélection des modèles ont été développées afin de choisir un modèle associé à une série d'événements qui forme une commande complète. Des préconditions ont été définies pour vérifier si la fusion peut avoir lieu. Des règles SWRL et les relations de propriétés d'objets et de données ont été définies pour simplifier la description de l'environnement. Pour conclure, un scénario a été proposé pour mieux comprendre le mode de fonctionnement du moteur de fusion.

Dans ce chapitre, un nouveau mécanisme de fusion a été proposé, basé sur la description des connaissances et les relations entre elles, pour faciliter la communication entre l'homme est la machine. La présentation sémantique de l'environnement et des différents contextes permet d'exprimer explicitement les connaissances de l'environnement afin de pouvoir les manipuler et de faire ressortir des raisonnements intéressants pour le moteur de fusion.

CHAPITRE 5

VALIDATION DE L'ARCHITECTURE

5.1 Introduction

La validation est une étape importante dans le processus de développement d'une architecture informatique. Elle permet de prouver que l'architecture mise en place répond à des exigences initiales de manière positive, surtout dans un système en temps réel. Ce type de système se distingue des autres systèmes informatiques par le fait qu'il est soumis à des contraintes temporelles fortes, dues à la criticité de certaines actions qui lui permettent d'interagir avec l'environnement d'un procédé à contrôler.

Plusieurs méthodes d'évaluation des architectures logicielles ont été proposées selon des méthodologies telle que ATAM (Architecture Tradeoff Analysis Method) (Ipek Ozkaya Mai 2007), SAAM (Software Architecture Analysis Method) (JEONG Gu-Beom mai 2006) voir ARID (Active Reviews for Intermediate Design) (R. Kazman 2005) et des qualités désirées (Babar 2004), (Mattsson 2006). Parmi ces méthodes, nous pouvons citer celles qui sont basées sur les expériences, les simulations, les scénarios et la modélisation mathématique. Ces méthodes peuvent être utilisées séparément ou combinées pour obtenir de meilleurs résultats. Il est fortement recommandé de faire l'évaluation tôt au niveau de la conception architecturale, en développant et en évaluant un modèle d'architecture logicielle qui aide à comprendre la fonctionnalité du système, ses possibilités et sa complexité. L'utilisation des méthodes formelles pour modéliser et décrire le comportement du système à un niveau élevé d'abstraction peut contribuer à la consistance et à la fiabilité de la conception architecturale, facilitant ainsi le procédé de validation. La simulation permet d'étudier le comportement de l'architecture logicielle pour vérifier ses propriétés au moyen de « model-checking », pour analyser le comportement dynamique du système dans certains cas d'utilisation.

5.2 Les réseaux de Petri colorés et le CPN-Tools

Les réseaux de Petri ont été inventés en 1962 par Carl Adam Petri. Ils sont utilisés comme méthode formelle pour la modélisation d'une classe importante de systèmes (Reisig 1985).

Ils permettent d'exprimer les propriétés comportementales et structurales des systèmes. Un aspect important des réseaux de Petri est qu'ils peuvent être représentés graphiquement ou mathématiquement, ce qui améliore leur clarté et leur lecture. Cependant, le problème d'explosion d'états reste l'obstacle majeur pour la modélisation et l'analyse des systèmes complexes. Les RdPC (Réseau de Petri Colorées) ont été proposés comme extension des réseaux de Petri ordinaires pour surmonter le problème d'explosion d'états en offrant un modèle plus compact. Les RdPC utilisent une modélisation à événements discrets, qui combine les réseaux de Petri ordinaires et le langage de programmation fonctionnel ML fondé sur le standard ML. Un modèle de RdPC d'un système décrit ses états et les événements (transitions) qui offrent la possibilité au système de changer d'état comme les réseaux de Petri ordinaires mais en utilisant des jetons de couleurs différentes. Une couleur est une information additionnelle sur le jeton (Jensen 1998). Notre utilisation des RdPC est motivée aussi par l'existence d'outils logiciels qui appuient la construction graphique et la visualisation des modèles. Parmi ces outils, nous avons choisi le CPN-Tools (Kristensen 2009) (University of Aarhus 2012) (CPN-Tools: Computer Tool for Coloured Petri Nets) pour explorer le comportement du modèle de l'architecture.

5.3 Simulation de l'architecture

La Figure 5.1 représente le réseau de Petri coloré général de l'architecture. Il est formé par des transitions et des places représentées sous forme de rectangles et de nœuds respectivement. Cinq générateurs principaux sont employés comme des transitions en double rectangle, qui génèrent des événements temporels d'une manière aléatoire. (*Vocal Event Generator, Manual Event Generator, Gesture Event Generator, Visual Event Generator et Tactil Event Generator*). Ces générateurs simulent le fonctionnement des modalités qui détectent des événements à partir de l'environnement. Chaque générateur est lié à sa modalité correspondante. Chaque modalité est liée aux quatre types d'événements définis dans l'ontologie (*Action, Person, Location et Object*). Ces événements sont présentés comme des transitions en double rectangle (*Action, events, object events, person events et location events*). Enfin, chaque événement est associé au moteur de fusion (*fusion*). La transition en

double rectangle signifie que c'est une transition composite expliquée dans un sous-réseau. Le nom du sous-réseau est donné dans l'encadré à côté de chaque transition.

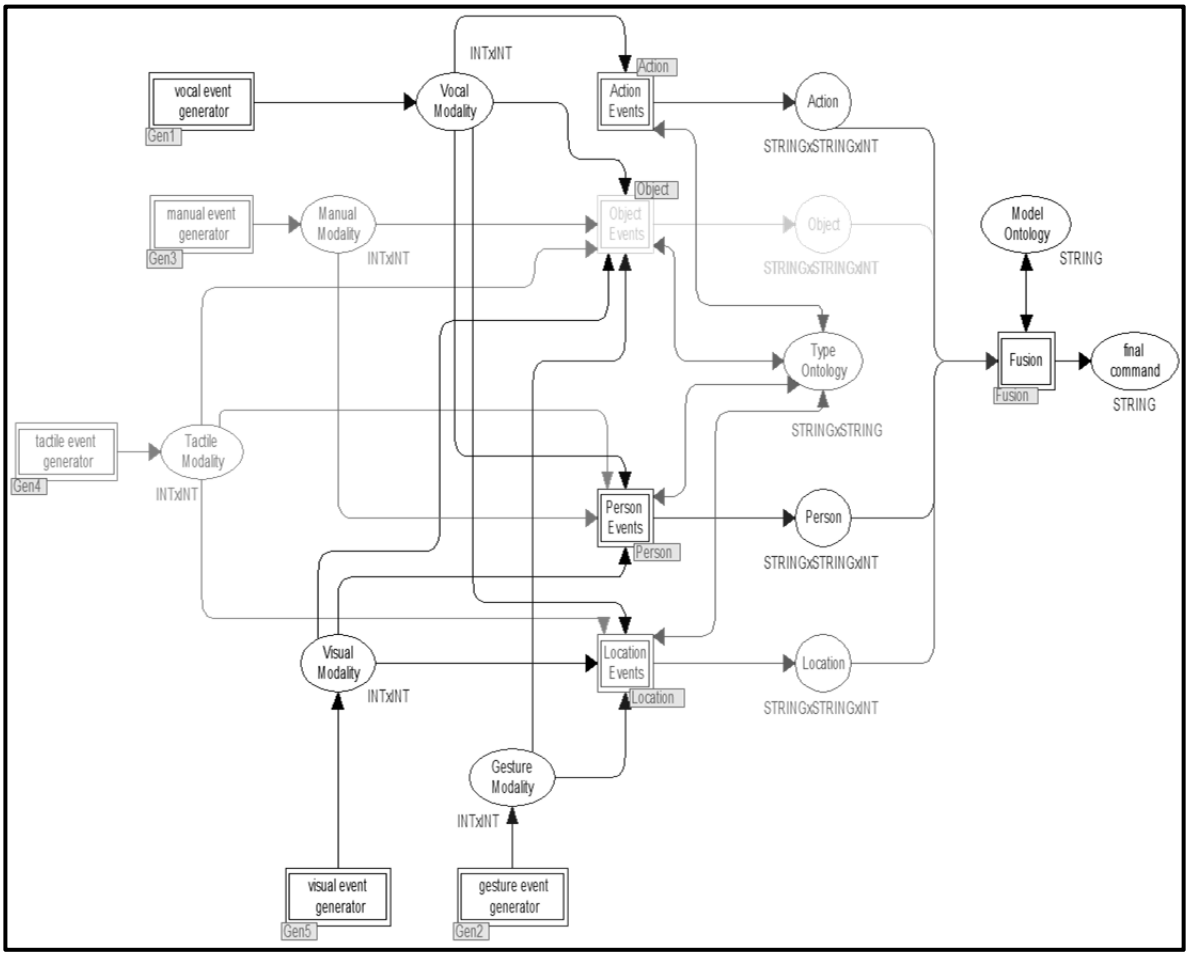


Figure 5.1 Représentation générale de l'architecture par le réseau de Petri coloré

5.3.1 Définition des paramètres

Pour cette validation, deux générateurs sont pris en compte : un générateur aléatoire pour la modalité gestuelle et un autre pour la modalité vocale. Un échantillon des instances de l'ontologie est défini comme jetons (tokens) à l'intérieur de chacun des quatre types d'événements représentés dans le réseau.

- événements de type *Action*: (1,"get"), (2, "give") et (3, "put") ;
- événements de type *Object*: (1, "that") et (2, "this") ;
- événements de type *Person*: (1,"father") et (2,"me") ;
- événements de type *Location*: (1,"here") et (2,"there").

La durée maximale autorisée pour une commande *MaxCommadTime* est déclarée à 10 s. (Par exemple 10 secondes est le temps maximum donné pour formuler une commande comme "give me this" qui contient trois événements: «give», comme action, «me» comme personne et "this" comme objet). Le temps maximal autorisé entre deux événements successifs *MaxActiveModalityTime* est déclaré à 5 s.

Parmi les 30 modèles déclarés dans l'ontologie, deux sont pris en compte dans cette validation :

- AFMO→IO→IL (*ActionForMovableObject → IntendedObject → IntendedLocation, Model08*) ;
- AFP→P→IO *(ActionForPerson → Person → IntendedObject, Model17*).

Pour le contexte de l'environnement, le niveau de luminosité est considéré pour la modalité gestuelle et le niveau du bruit pour la modalité vocale. Les niveaux sont définis de 1 à 10 en tant que jetons de chaque générateur (sur les places *Noise level detected et Light level detected*). La comparaison est effectuée selon le niveau 5 :

- si le niveau de bruit est supérieur à 5, la modalité vocale est désactivée et
- si le niveau de luminosité est inférieur à 5, la modalité gestuelle est désactivée.

Cinq catégories ont été considérées pour le contexte de l'utilisateur :

- surdité et mutisme pour la modalité vocale ;
- aveugle, handicap manuel pour la modalité gestuelle ;
- sans handicap pour les deux modalités.

En réalité, le temps entre les événements et le temps d'une commande, le niveau de lumière et du bruit sont définis selon le profil de l'utilisateur, car chaque personne parle d'une manière différente et chaque environnement a son propre contexte environnemental.

Pour simplifier la simulation, nous avons supposé que les événements détectés sont sémantiquement corrects afin d'éviter la création d'un réseau du petri coloré propre au raisonneur.

5.3.2 Implémentation

Le Tableau 5.1 représente les déclarations des variables du réseau coloré dans CPN-Tools. Après avoir déclaré tous les paramètres nécessaires pour que le moteur de fusion fonctionne correctement, la validation fonctionnelle de l'architecture peut commencer.

Tableau 5.1 Déclarations des variables dans CPN-Tools

```
colset UNIT = unit;
colset INT = int;
colset STRING = string;
colset BOOL = bool;
colset Event = with event timed;
colset Memory = product INT*STRING;
colset Object = with object timed;
colset ObjectAttribut = product Object * INT * INT;
colset INTxINTxINT= product INT*INT*INT;
colset STRINGxSTRINGxINT = product STRING*STRING*INT;
colset INTxINT= product INT*INT;
colset STRINGxINT = product STRING*INT;
colset STRINGxSTRING = product STRING*STRING;
fun round x = floor(x+0.5);
fun inTime() = IntInf.toInt(time());
fun discExp x = round(exponential(x));
fun NormalLaw(x,y) = round(normal(x,y));
var Current,Active,light,Noise: BOOL;
val InterToArrive = ref 0.5;
```

```
var person,location:STRINGxSTRINGxINT;
var handicap,decision,model1,model2,model3,Act,Loc,Obj,Per,Mod,c,c1,c2,c3,c4,d:
STRING;
var NextTime, TimeRecon, TimeExecution, n, p, nTreat,t,Index:INT;
var voice,movement: INTxINT;
var a,lim,E,E2,T,T2,W,W2,W3,Tb, Ta,Tc,Td: INT;
```

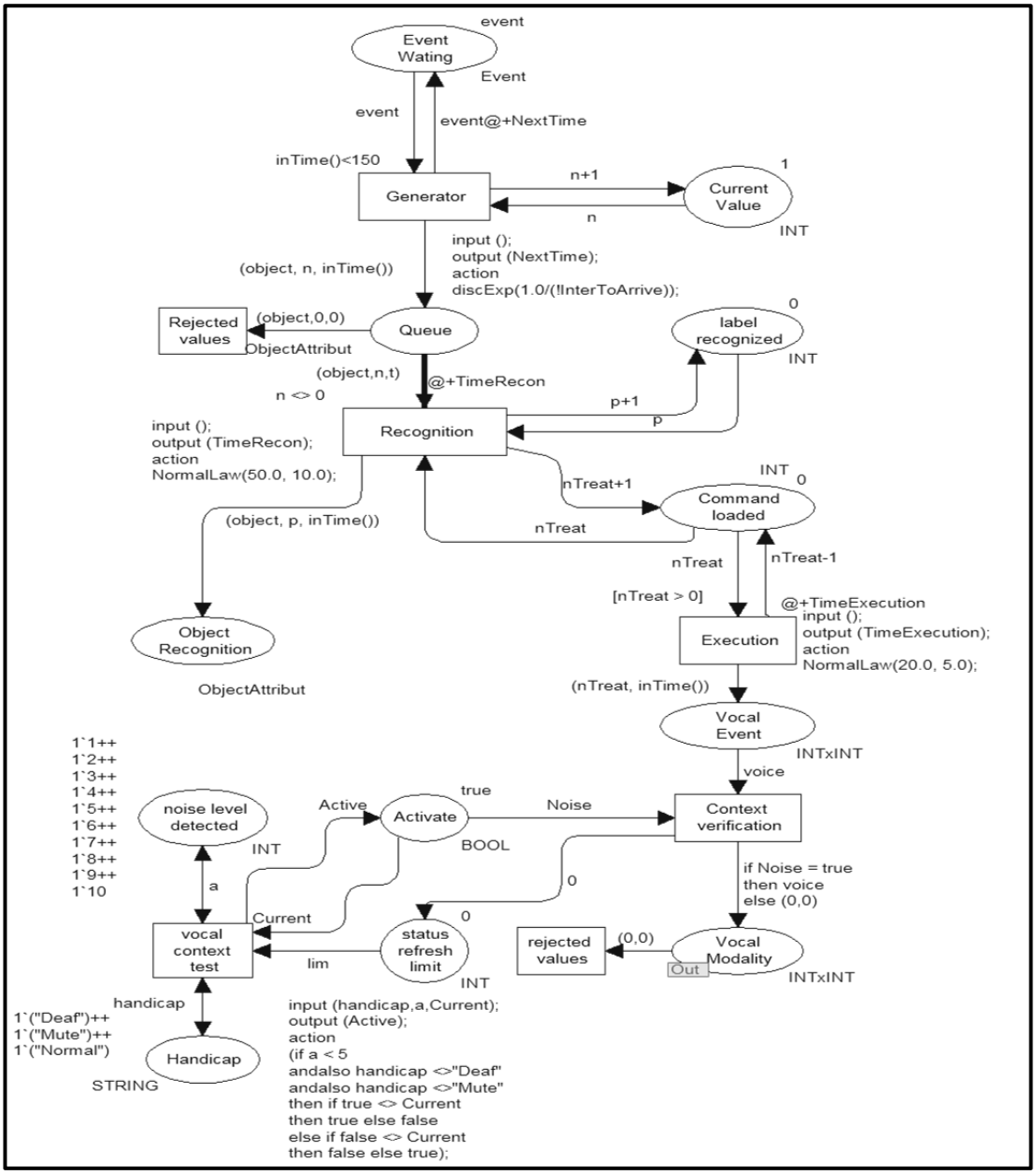


Figure 5.2 Générateur aléatoire pour la modalité vocale

Le générateur aléatoire d'événements est représenté par des jetons qui arrivent dans une file d'attente (***Queue***) Les caractéristiques du générateur sont les suivantes :

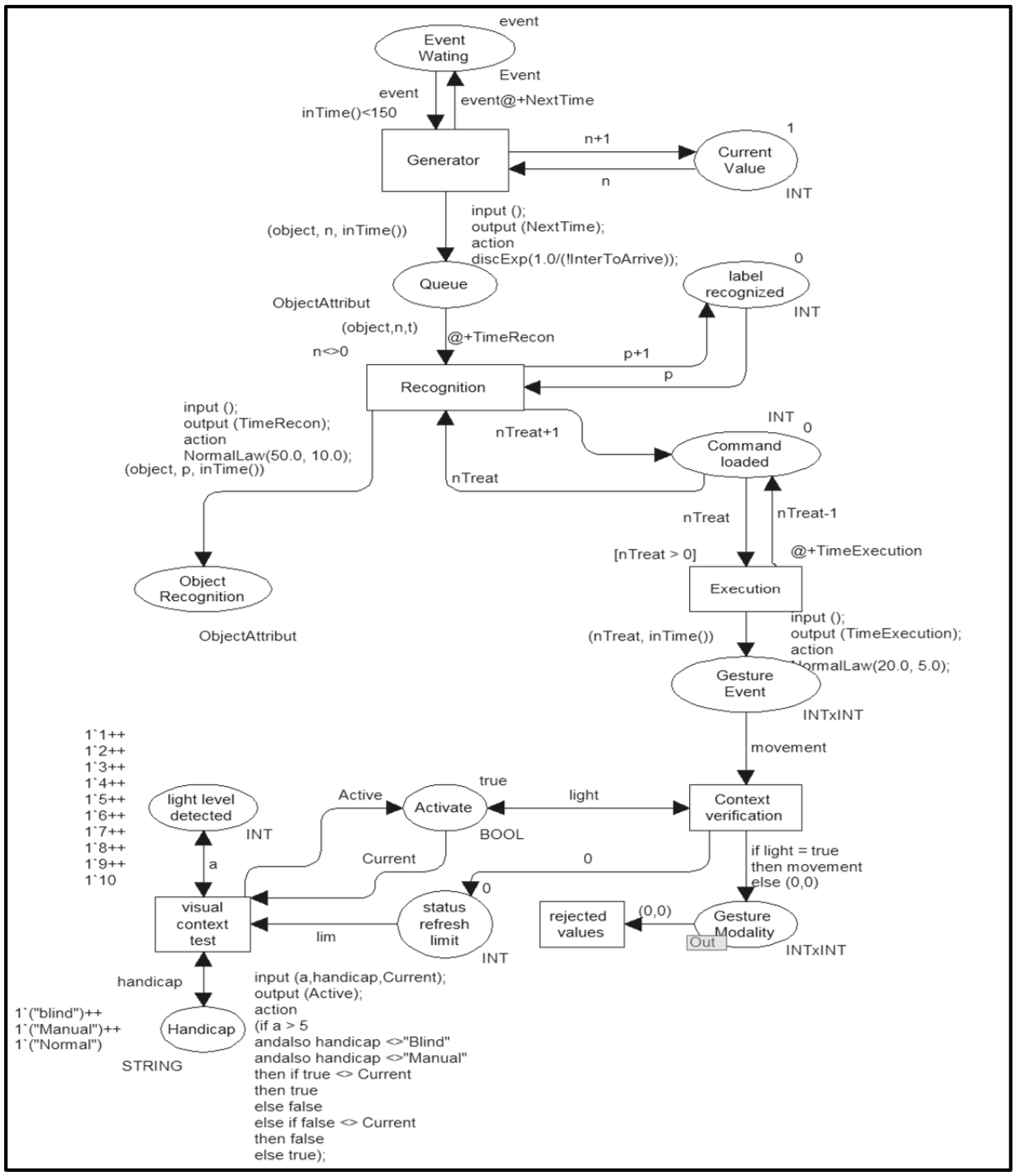


Figure 5.3 Générateur aléatoire pour la modalité gestuelle

- chaque jeton est représenté par un multiset de colorsets (type) ***ObjectAttribut*** qui est le produit de trois colorsets :
  - un symbole est déclaré par :

    Color Object = with object timed;
  - un entier ***n*** représentant le numéro d'arrivée du jeton (le premier jeton à la position1, le second 2, etc.) ;
  - un entier représentant le temps d'arrivée du jeton dans la file d'attente : il est donné par l'appel d'une fonction : ***intTime()*** qui renvoie le temps courant ;

- le temps d'arrivée des jetons dans le ***Queue*** suit une loi exponentielle :

  fun discExp x = round(exponential(x));

  fun round x = floor(x+0.5)

- les jetons dans la file d'attente sont reconnus par un système de reconnaissance trivial modélisé par une transition appelée ***Recognition***. Ses caractéristiques se limitent à :
  - retirer, un à un, un certain nombre de jetons de ***Queue.*** Ce nombre sera inférieur à une valeur fixée par l'utilisateur (appelée ***ncapacity***) ;
  - déposer dans une nouvelle file d'attente (***ObjectRecongition***) les objets reconnus qui auront le même colorset que les objets à reconnaître ;
  - la reconnaissance de chaque symbole durera un temps aléatoire suivant une loi normale :

    fun LoiNormal (x,y) = round(normal(x,y)) ;
  - une place (***LabelRecognized***) initialisée avec un jeton (un entier ***p*** qui vaudra initialement0) est rattachée à ***Recognition*** et jouera le rôle de compteur du nombre de symboles reconnus ;
  - les symboles générés par ***Recognition*** dans la place ***ObjectRecongnition*** auront pour multiset **:** ***(object, p, t)*** avec t un entier ;

- une modélisation triviale de l'exécution des jetons reconnus, par ***Recognition*** avec un temps d'exécution aléatoire suivant une loi normale**,** sera composée des éléments suivants :
  - une place compteur (***Command Loaded***) initialisée par un jeton (un entier ***nTreat*** déclaré initialement à 0) est rattachée à ***Recognition*** et jouera également le rôle de compteur du nombre de symboles reconnus ;
  - une transition reliée à ***Command Loaded*** appelée ***Execution*** ;
  - une place reliée à ***Execution*** appelée ***CommandeExecutde***.

Avant d'envoyer des événements au moteur de fusion pour les tester, une vérification des contextes doit être faite. Le niveau du bruit et celui de la lumière sont définis sur les places *Noise Level Detected* et *Light Level Detected* respectivement de 1 à 10 pour le contexte environnemental. Les types des handicaps sont définis sur les places *Handicap* des deux générateurs. Si le niveau du bruit est supérieur à 5 et/ou l'utilisateur est sourd ou muet (voir code de la transition *Vocal Context Test*), la modalité vocale sera désactivée et les événements ne pourront pas être envoyés au système multimodal et seront rejetés dans la place *Rejected Values*. Sinon, un jeton sera envoyé dans la place *Activate* du type booléen, qui, à son tour, active la transition *Context Verification* et par la suite l'événement peut être envoyé au système. De même dans le générateur responsable des événements gestuels, si le niveau de la lumière est inférieur à 5 et/ou l'utilisateur est aveugle ou paralysé, la modalité gestuelle sera désactivée et les évènements correspondants seront rejetés dans la place *Rejected Values*. Sinon, un jeton sera envoyé dans la place *Activate* du type booléen, qui, à son tour, active la transition *Context Verification,* et par la suite l'événement peut être envoyé au système.

Les trois autres générateurs sont définis de la même manière. Chaque générateur contient sa propre partie de la sélection des modalités en fonction des contextes appropriés.

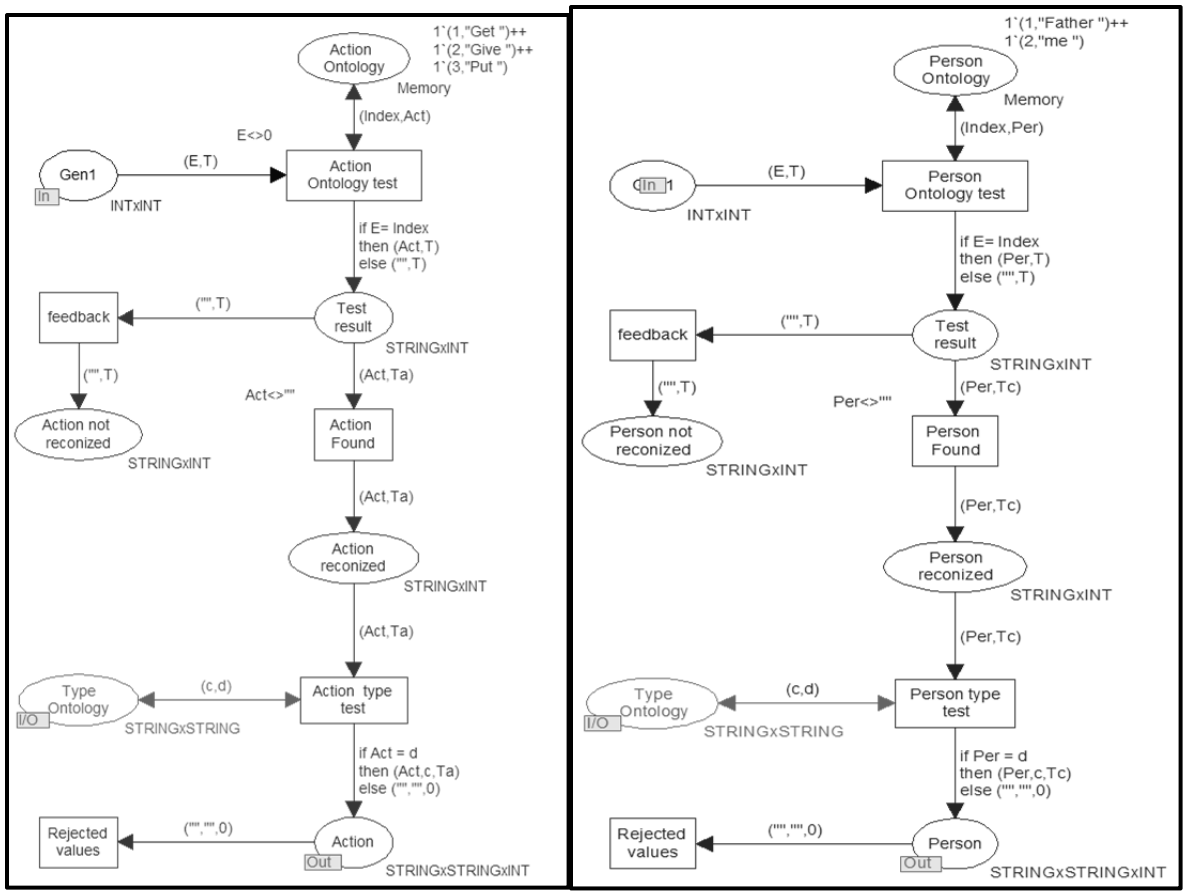


Figure 5.4 Vérification des événements du type *Action* et *Person*

La Figure 5.4 et la Figure 5.5 représentent la vérification du vocabulaire des différents événements aléatoires provenant des générateurs. Les quatre réseaux partagent la même structure de base et reçoivent un événement E (comme index) à un temps spécifique T. Dans chacun d'eux, une comparaison est réalisée pour vérifier si les événements provenant des générateurs existent comme instance dans l'ontologie. Les instances sont présentées comme des jetons dans les places *Action Ontology, Person Ontology, Object Ontology* et *Location Ontology*.

D'abord une comparaison est effectuée en vérifiant si le vocabulaire existe dans l'ontologie, en comparant l'index de l'événement provenant du générateur avec ceux des instances (voir le code sur les transitions *Action Ontology test, Person Ontology test, Object Ontology Test* et *Location Ontology Test*). Si les indexes sont égaux, cela signifie que l'événement existe

dans l'ontologie, sinon, il sera rejeté dans la place *feedback.* Une fois l'événement reconnu, une autre vérification du type d'événement est faite. Par exemple, si l'événement est une action, il faut savoir si c'est une ActionForMovableObject ou ActionForPerson, etc. (voir le code dans les transitions *Action type test, Person type test, Object type test* et *Location type test*), (exemple: Get, a le ActionForMovableObject comme type d'action dans l'ontologie). Si le type n'est pas identifié, l'événement sera encore rejeté dans la place *Rejected values.* À la fin, nous obtiendrons un événement reconnu avec son type et son temps dans les places *Action, Person, Object* et *Location*, qui ont STRING*STRING*INT comme type de variable.

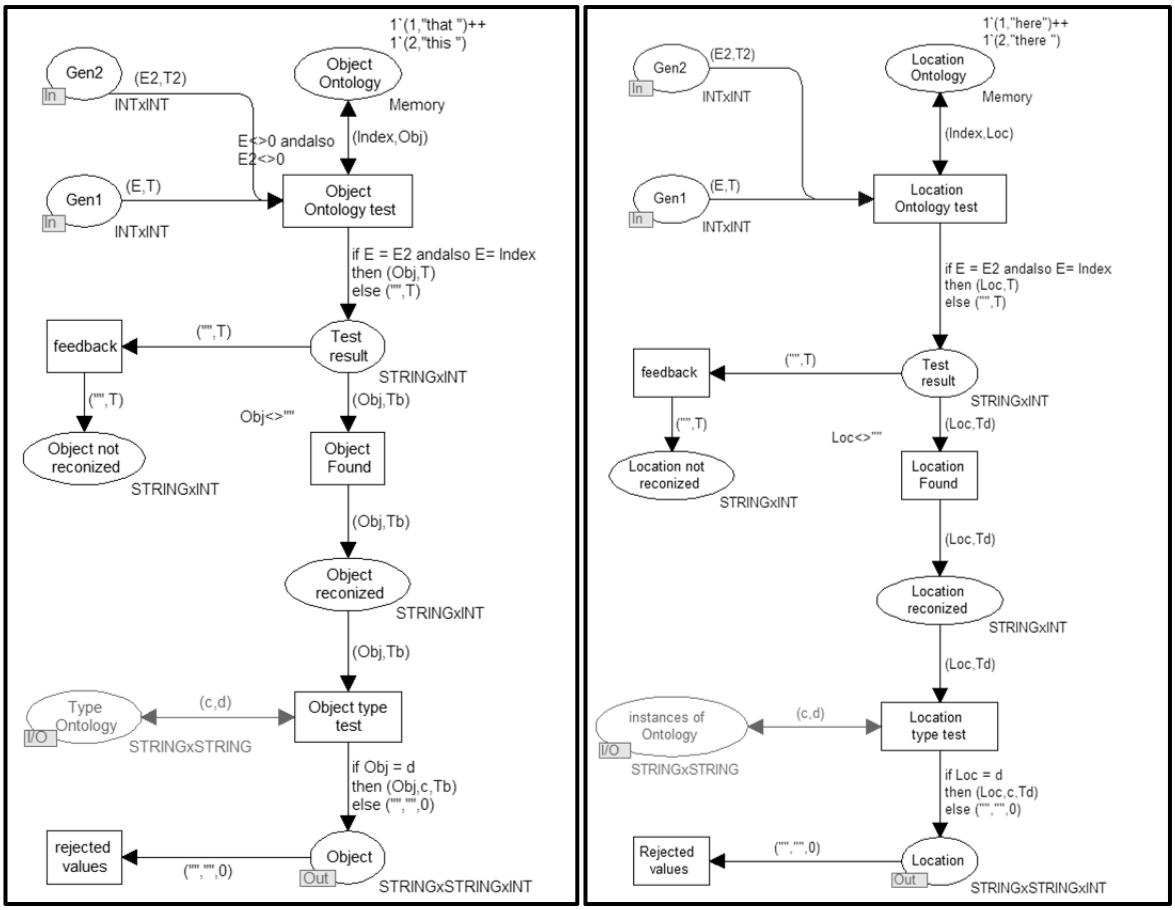


Figure 5.5 Vérification des événements du type *Object* et *Location*

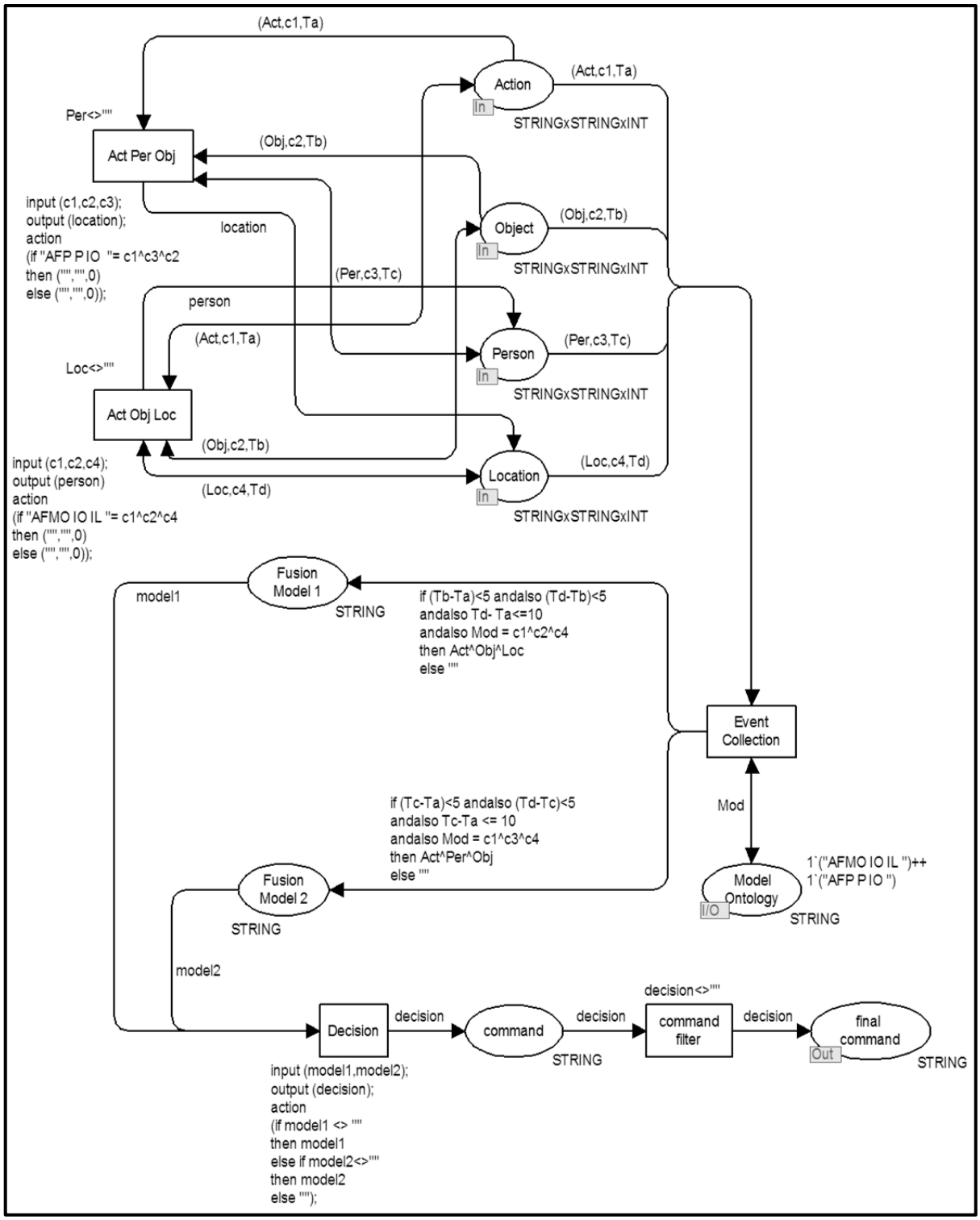


Figure 5.6 Vérification temporelle et des modèles de fusion

La Figure 5.6 représente la vérification temporelle et des modèles dans le moteur de fusion. Celui-ci a pour entrée les événements sortant des quatre places *Action, Object, Person* et *Location*.

La transition *Act Per Obj* est reliée aux places *Action, Person* et *Object* et la transition *Act Obj Loc* est reliée aux places *Action, Object* et *Location.* Ces deux transitions et leurs arcs ont été créés pour permettre au réseau de procéder dans le cas où moins que quatre événements sont utilisés. Par exemple, la commande « *give me that* » utilise trois événements de types *Action, Perso* et *Object*, mais pas le type *Location.* Pour éviter le « *deadLock* » dans le réseau, un événement vide est généré par la place *Location* (voir le code sur les transitions *Act Per Obj* et *Act Obj Loc*) et qui sera éliminé par la suite. Une fois les événements collectés avec leurs types dans la transition *Event Collection*, une vérification du modèle et de l'aspect temporel sera faite. Pour la vérification temporelle, la différence de temps entre deux événements successifs et le temps maximal autorisé pour une commande ne doit pas dépasser les 5 et 10 secondes respectivement. Pour la vérification des modèles de fusion, deux modèles qui existent dans l'ontologie ont été définis sous forme de jetons dans la place *Model Ontology*, *AFMO IO IL* et *AFP P IO.* Si la concaténation des types d'événements est correspondante à l'un des modèles de l'ontologie, donc les événements reconnus appartiennent à ce modèle. Les conditions qui vérifient le temps et les modèles se trouvent sur les deux arcs provenant de la transition *Event Collection*. Si le vocabulaire, le temps et les modèles ont été respectés, alors la fusion se produit et une commande complète est obtenue dans la place *Final Command*, sinon il n'y a pas de fusion, parce que l'une des conditions n'a pas été respectée. Nous avons également inclus un filtre pour éliminer les événements vides générés en cas de valeurs rejetées sur la transition *comand filter.*

5.3.3 Résultat de la simulation

Quatre simulations ont été effectuées pour vérifier la fonctionnalité de l'architecture, 22 événements ont été générés dans la première simulation, 25 dans la deuxième, 32 au troisième et 22 événements dans la dernière. Dans chaque simulation, un nombre différent d'événements est obtenu grâce aux générateurs qui génèrent des événements aléatoires. Pour

chaque simulation, nous avons obtenu 3 graphes. Le premier représente le nombre d'événements reconnus et non reconnus par le système. Le second représente le temps de la reconnaissance d'un événement. Le troisième représente la fusion et la non-fusion qui ont eu lieu selon un modèle. Les résultats de la première et de la deuxième simulation sont représentés dans la Figure 5.7 et ceux de la troisième et la quatrième dans la Figure 5.8.

Dans la première simulation, 22 événements ont été générés. Le premier graphe montre que les événements *get*, *give* et *here* sont reconnus une seule fois, alors que l'événement *that* est reconnu deux fois. Il y a 17 autres événements qui ne sont pas reconnus (*others*). La reconnaissance de ces événements signifie que notre système a été capable de les identifier comme des instances de l'ontologie. La première condition qui est le vocabulaire a donc bien été vérifiée. Le deuxième graphe montre à quel instant ces événements ont été reconnus :

- le premier *that* a été reconnu à la 2^{ième} sec, le second *that* à la 8^{ième} s ;
- le *here* a été reconnu à la 5^{ieme} s ;
- le *get* a été reconnu à la 1^{ére} s ;
- le *give* a été reconnue à la 50^{ième} s.

D'après ces temps-là, nous pouvons déduire que le temps entre deux événements successifs a bien été respecté avec les événements *get*, le premier *that* (2-1=1 s < 5 s) et *here* (5-2= 3 s < 5 s). En plus, le temps maximal autorisé pour une commande est encore respecté (1+2+5=8 s < 10 s). Donc la condition temporelle est encore respectée. Le dernier graphe montre que le système a reconnu une commande qui obéit au modèle du type AFMO IO IL, puisque *get* est un *ActionForMovableObjec*t, *that* est un *IntendedObject* et *here* est un *IntendedLocation*. À la lumière de ces résultats, nous pouvons affirmer que toutes les conditions ont été vérifiées. Le moteur de fusion a pu reconnaître une commande complète « *get that here* ». D'autre part, 6 autres combinaisons d'événements ont été éliminées car ils ne respectent pas les conditions temporelles et les modèles de fusion.

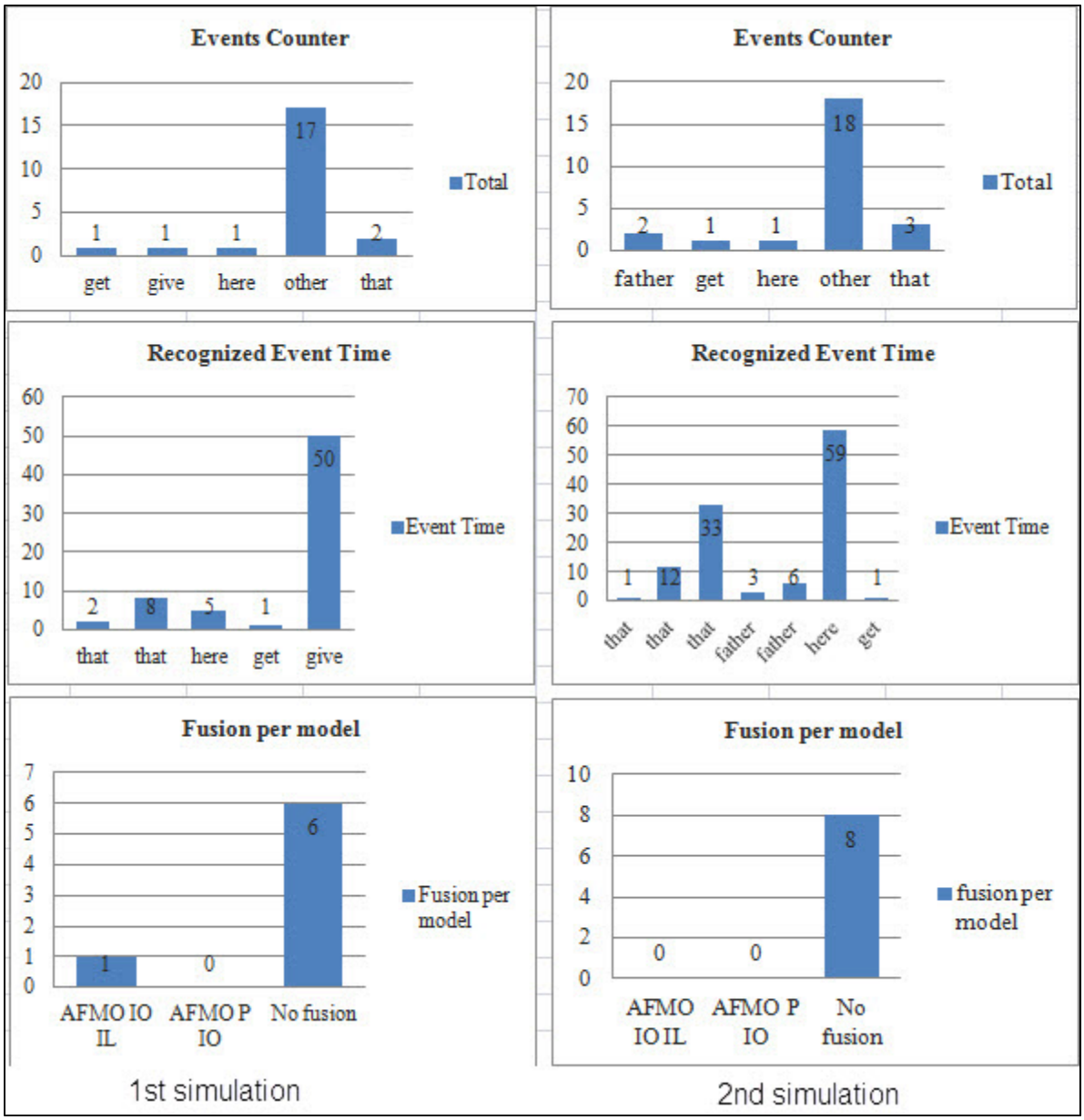


Figure 5.7 Première et deuxième simulation

Dans la seconde simulation, 25 événements ont été générés. Le premier graphe montre que les événements *get* et *here* ont été reconnus une seul fois, l'événement *father* deux fois et *that* trois fois. 18 autres événements n'ont pas été identifiés. La reconnaissance de ces 7 événements signifie que l'architecture a été capable de les identifier comme des instances de

l'ontologie. La vérification du vocabulaire est bien faite. Le deuxième graphe montre à quel instant ces événements ont été reconnus :

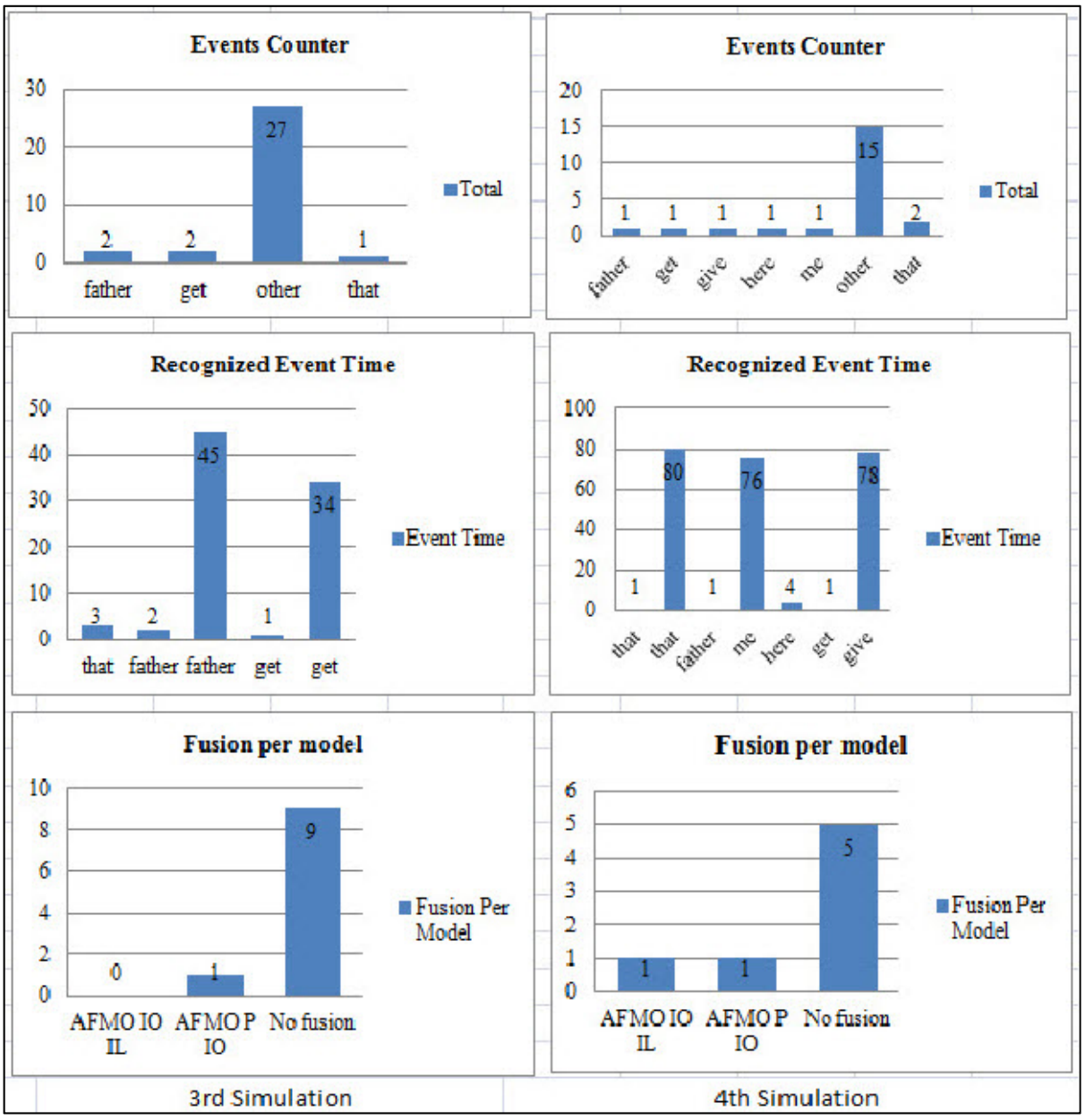


Figure 5.8 Troisième et quatrième simulation

- le *that* a été reconnu à la 1ère s, à la 12ième et à la 33ième s pour la deuxième et la troisième fois respectivement ;
- le *father* a été reconnu à la 3ième s et à la 6ième s pour la deuxième fois ;

- le *here* a été reconnu à la 59^{ième} s ;
- le *get* a été reconnu au temps 1 s.

Dans cette simulation, trois commandes sont possibles ; *get father here*, *get that here* et *get here that*. Mais le moteur de fusion élimine les deux commandes qui contiennent l'évenement *here,* car il ne respecte pas les deux conditions temporelles et il dépasse les 5 secs et les 10 s, puisqu'il était reconnu à la 59^{ième} s. Pour la commande restante « *get father that* ». Le premier événement *that* (à la 1^{ére} s) est encore éliminé, car selon le temps d'arrivé, il est venu avant les deux événements *father* (3^{ième} et 6^{ième} s) et il ne respecte pas le modèle de l'AFP P IO. Pour les deux autres *that*, ils sont encore éliminés car bien qu'ils viennent après l'événement *father* et respectent le modèle, ils dépassent largement le temps autorisé entre deux événements successifs et le temps maximal pour une commande. (12-1 =10 s > 10 et 5 s) (33-1 =32 s >10 et 5 s). Donc dans ce cas, nous remarquons que la condition temporelle et de matching du modèle n'ont pas été respectées, donc la fusion n'a pas eu lieu.

Dans la troisième simulation, 32 événements ont été générés. Le premier graphe montre que les événements *father* et *get* ont été reconnus deux fois respectivement, le *that* une seule fois et 27 autres événements n'ont pas été reconnus. La reconnaissance de ces 6 événements signifie que le système a été capable de les identifier comme des instances de l'ontologie. La vérification vocabulaire est faite. Le deuxième graphe montre à quels instants ces événements ont été reconnus :

- le *that* a été reconnu à la 3^{ième} s ;
- le *Father* a été reconnu un fois à la 2^{ième} s et une autre fois à la 45^{ième} fois ;
- le *get* a été reconnu une fois à la 1^{ière} s et une autre fois à la 34^{ième} s.

Dans cette simulation, une seule commande est possible : *get father that*. L'événement *get* est rejeté à la 34^{ième} sec parce qu'il est arrivé après *father* et *that* (à la 2^{ième} et 3^{ième} s) et il ne respecte pas le temps ni le matching avec un modèle. Le *father* à la 45^{ième} s est encore éliminé car il ne respecte pas le temps. Donc il reste *get* à 1 s, *father* à 2 s et *that* à 3 s. Le

dernier graphique montre que le système a reconnu une commande complète qui obéit au modèle du type AFMO P IO puisque le *get* est un *ActionForMovableObject*, le *father* est un *Person* et le *that* est un *IntendedObject*. Dans cette simulation, la fusion a eu lieu une seul fois et 9 autres combinaisons ont été éliminées pas le moteur de fusion

Dans la quatrième simulation, 22 événements ont été générés. Le premier graphe montre que les événements *father, get, give, here* et *me* ont été reconnus une seul fois, le *that* deux fois et 15 autres événements non identifiés. La reconnaissance de ces 7 événements signifie que le système a été capable de les identifier comme des instances de l'ontologie. Le deuxième graphe montre à quels instants ces événements ont été reconnus :

- le *that* a été reconnu à la $2^{ième}$ s pour la première fois et à la $81^{ième}$ s pour la deuxième fois ;
- le *father* a été reconnu à la $1^{ère}$ s ;
- le *me* a été reconnu à la $77^{ième}$ s ;
- le *here* a été reconnu à la $4^{ième}$ s ;
- le *get* a été reconnu à la $1^{ère}$ s ;
- le *give* a été reconnu à la $76^{ième}$ s.

Dans cette simulation, une fusion a été réalisée pour chaque modèle et deux commandes ont été obtenues : *get that here* (*get* à 1s, *that* à 2s et *here* à 4 ms, le modèle est AFMO IO IL) et *give me that* (give à 76s, me à 77s et *that* 81s, le modèle est AFP P IO) car toutes les conditions ont été respectées. 5 combinaisons ont été éliminées par le moteur de fusion.

5.4 Conclusion

Dans ce chapitre, on a pu valider le bon fonctionnement de l'architecture à l'aide d'un réseau de Petri coloré stochastique. Les quatre simulations ont montré comment le moteur de fusion vérifie toutes les conditions afin de décider si la fusion peut avoir lieu. Le réseau a pris en considération les différents contextes définis dans le chapitre précédent pour mieux choisir les modalités, le vocabulaire déclaré sous forme des instances dans l'ontologie, la vérification temporelle et la sélection des modèles.

La première simulation a vérifié le scénario « *get that here* » qui a été retenu par le moteur de fusion car il respecte toutes les conditions. La deuxième a montré comment le moteur de fusion élimine les événements dans le cas où au moins l'une des conditions n'est pas respectée. La troisième a vérifié le scénario « *get father that* » et la dernière simulation a montré que l'architecture est capable de reconnaître plusieurs scénarios « *get that here »* et « *give me that »* et de faire la fusion pour plusieurs commandes si elles respectent les conditions.

La validation du fonctionnement de l'architecture a été une étape cruciale dans le travail, afin de s'assurer que le système réagit bien dans son environnement, étant donné qu'il doit fonctionner dans un environnement dynamique et en temps réel. Nous avons donc vérifié à l'aide de scénarios types que le système de fusion multimodale pouvait assurer cette communication entre l'homme et la machine dans un environnement, tout en considérant les contraintes et les défis posés par un tel d'environnement.

CHAPITRE 6

IMPLÉMENTAION DU PROTOTYPE

6.1 Introduction

Dans ce chapitre, nous présentons l'élaboration du prototype du système de fusion multimodale. Celui-ci disposera de quelques fonctionnalités et servira de démonstrateur. L'application proposée est dédiée à toutes les personnes incluant les personnes âgées et les handicapées. En finalité, le système permettra à ses utilisateurs d'être autonomes. Ce prototype inclut une interface graphique assez simple et facile à utiliser.

Le système gérera sémantiquement la connaissance autour d'une ontologie et pourra interagir avec l'environnement en s'appuyant sur un transfert de connaissances de l'utilisateur vers l'outil, puis d'une restitution des connaissances stockées dans l'outil vers l'utilisateur.

Le prototype se focalise sur l'idée essentielle de notre projet, soit la fusion multimodale. Nous nous intéressons à la sélection des modalités et au moteur de fusion, de façon que le système comprennd l'environnement et les commandes envoyées par l'utilisateur.

6.2 Réalisation

Le langage de programmation utilisé pour la réalisation du prototype est le JAVA (java.com), sous la plateforme NetBeans 6.9.1 (netbeans.org/). Celle-ci est un environnement de développement intégré (EDI), placé en source libre par Sun. L'OWL API (OWL Application Programming Interface) (http://owlapi.sourceforge.net/) a été utilisé pour établir la connexion entre le code et l'ontologie. L'OWL API est une API Java, pour créer, manipuler et sérialiser des ontologies OWL. Elle contient les composants suivants :

- une API pour OWL 2 ;
- éditeur et analyseur des fichiers RDF / XML ;
- éditeur et analyseur des fichiers OWL / XML ;

- analyseur et éditeur fonctionnel de syntaxe OWL ;
- analyseur et éditeur du turtle
- analyseur KRSS
- OBO analyseur format de fichier plat

} langage de représentation pour les ontologies ;

- interfaces de raisonnement pour travailler avec des raisonneurs comme FACT + +, ermite, Pellet et Racer.

Pour tester l'application, le scénario « *get that here* » défini dans le 4ième chapitre est appliqué dans ce système. Donc les mêmes fichiers XML et les mêmes paramètres sont utilisés ici.

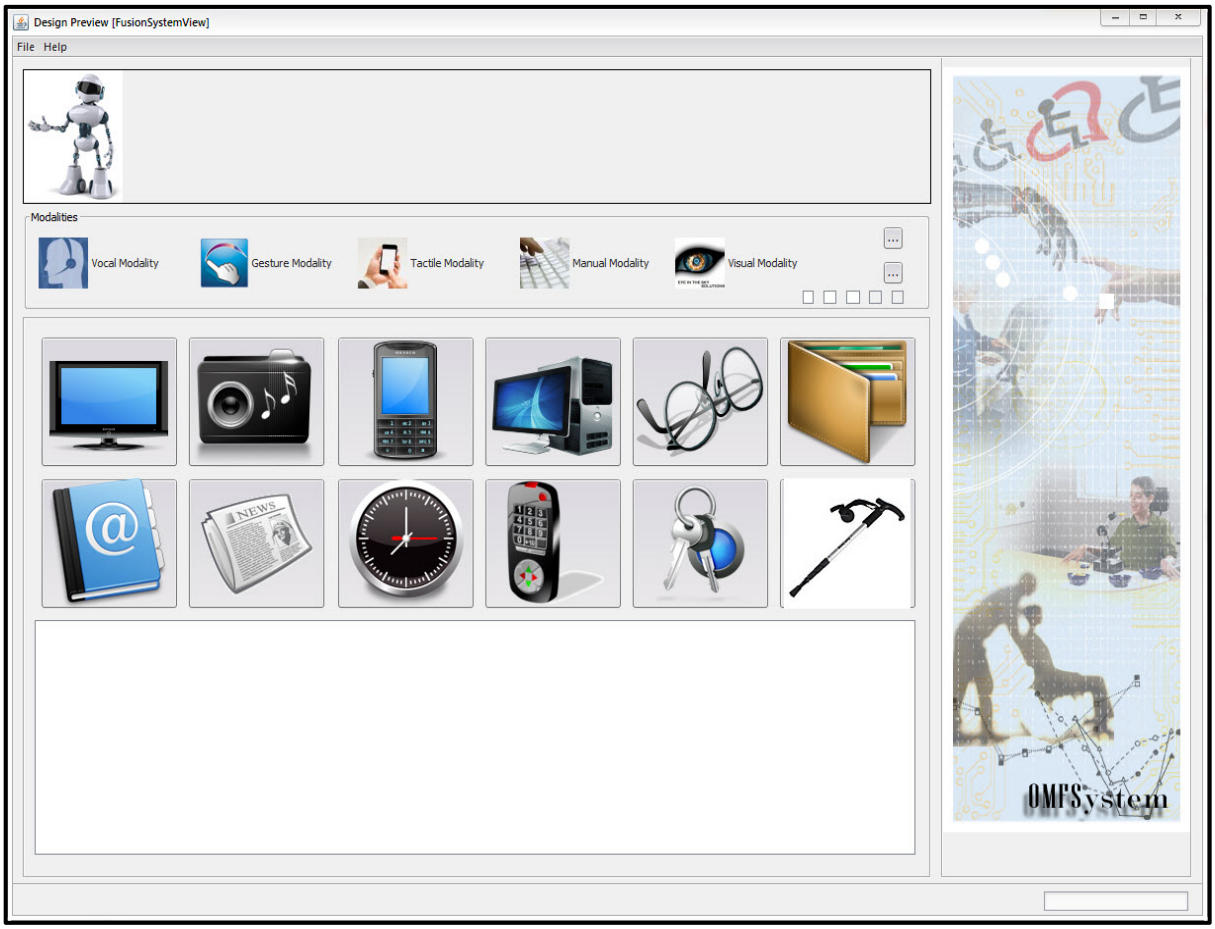


Figure 6.1 Interface du système

Deux interfaces graphiques ont été créées pour l'implémentation du prototype : une interface technique pour montrer le fonctionnement du moteur de fusion dans chaque étape et une

autre interface destinée aux utilisateurs du système. La Figure 6.1 représente l'interface générale du système de fusion multimodale destinée aux utilisateurs. Elle est formée par trois parties essentielles : les modalités, les boutons tactiles pour choisir les objets représentés par les photos et la partie résultat. La Figure 6.2 représente l'interface technique du système. Elle contient également trois parties : une pour la sélection des modalités, une deuxième pour visualiser les événements et une troisième pour montrer le fonctionnement du moteur de fusion.

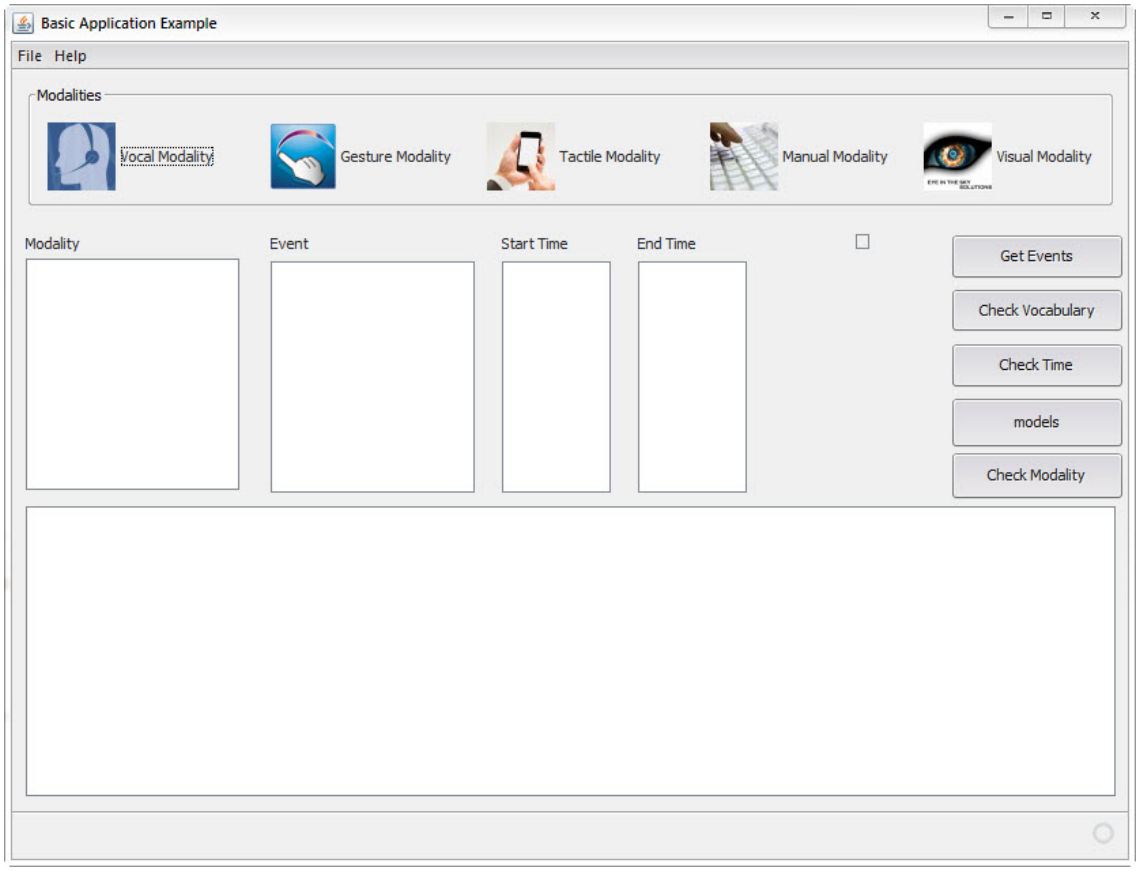


Figure 6.2 Interface technique du système

Nous avons représenté cinq types de modalités qui peuvent être activées ou désactivées selon les contextes reçus de l'environnement. La Figure 6.3 représente une partie du code responsable de la sélection des modalités. Les requêtes sont définies et vérifiées afin de

déterminer quelle modalité doit être désactivée ou non. Ce code se connecte à l'ontologie et applique les requêtes pour générer les informations dont le système a besoin.

```
OWLOntology ont;
OWLOntologyManager man = OWLManager.createOWLOntologyManager();
String path = "file:/Users/Ahmad/Documents/NetBeansProjects/FusionSystem/EnvironmentOnto.rdf-xml.owl";
ont = man.loadOntologyFromOntologyDocument(IRI.create(path));
System.out.println("Loaded: " + ont.getOntologyID());

OWLModel owlModel = ProtegeOWL.createJenaOWLModelFromURI(path);
String rule1 ="Modality(?m) ∧ Noise(?l) ∧ hasNoise(?m, ?l) ∧ hasNoiseLevel(?l, ?level) "
        + "∧ swrlb:greaterThan(?level, 6) →  sqwrl:selectDistinct(?m)";
String rule2 ="Modality(?m) ∧ Light(?l) ∧ hasLight(?m, ?l) ∧ hasLightnessLevel(?l, ?level) "
        + "∧ swrlb:lessThan(?level, 4) →  sqwrl:selectDistinct(?m)";
String rule3 ="Modality(?m) ∧ Handicap(?h) ∧ hasUserContext(?m, ?h) ∧ hasHandicap(?h, ?han) "
        + "∧ swrlb:equal(?han, 1) →  sqwrl:selectDistinct(?m)";
String rule4 ="Modality(?m) ∧ LocationContext(?loc) ∧ hasLocationContext(?m, ?loc) "
        + "∧ isAtRoad(?loc, ?loca) ∧ swrlb:equal(?loca, 1) →  sqwrl:selectDistinct(?m)";
SQWRLQueryEngine queryEngine = SQWRLQueryEngineFactory.create(owlModel);

String qname1 = "Rule1";
queryEngine.createSQWRLQuery(qname1, rule1);
SQWRLResult result1 = queryEngine.runSQWRLQuery(qname1);

String qname2 = "Rule2";
queryEngine.createSQWRLQuery(qname2, rule2);
SQWRLResult result2 = queryEngine.runSQWRLQuery(qname2);

String qname3 = "Rule3";
queryEngine.createSQWRLQuery(qname3, rule3);
SQWRLResult result3 = queryEngine.runSQWRLQuery(qname3);

String qname4 = "Rule4";
queryEngine.createSQWRLQuery(qname4, rule4);
SQWRLResult result4 = queryEngine.runSQWRLQuery(qname4);
```

Figure 6.3 Code pour sélectionner les modalités

La Figure 6.4 représente l'exécution du code précédent avec les modalités qui doivent être désactivées selon les contextes reçus par l'environnement. En réalité, ces modalités sont des instances dans l'ontologie. Chaque instance appartient à une classe de modalités. En connaissant ces instances, on peut déduire quelle modalité doit être arrêtée ou non.

Figure 6.4 Résultat du code de la Figure 6.3

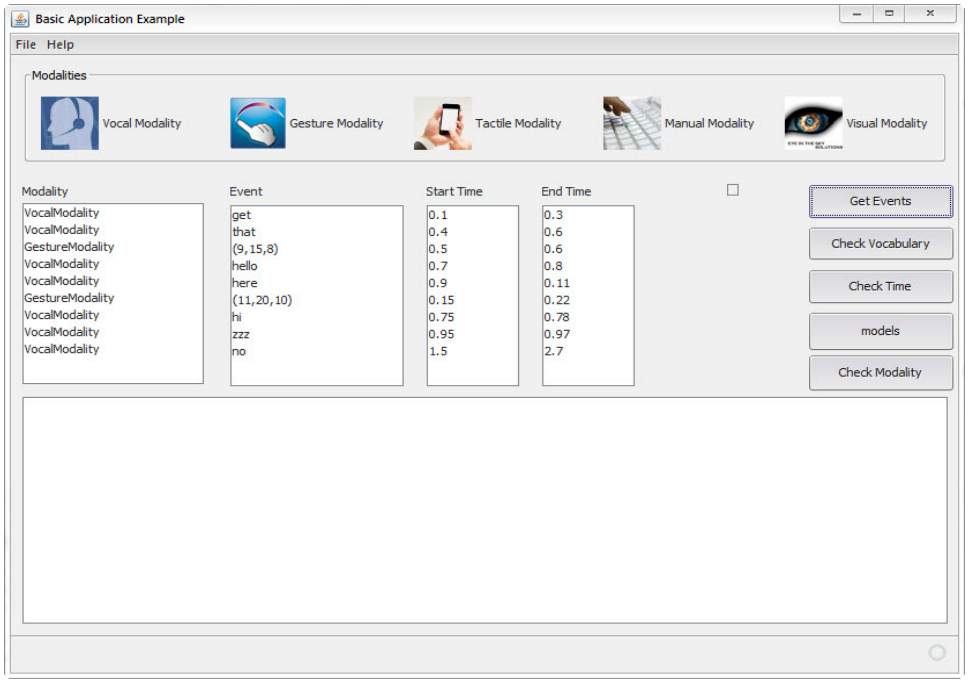


Figure 6.5 Réception des événements à partir des fichiers XML

La Figure 6.5 représente la réception des événements captés par les modalités à partir des fichiers XML. Ces fichiers contiennent le type de modalité active, l'événement et son temps de début et d'arrêt. En appuyant sur le bouton *Get Events*, les événements bruts sont affichés dans des listes mais sans savoir s'ils sont définis dans l'ontologie ou non. La Figure 6.6 ci-dessous est un extrait du code responsable d'afficher les événements dans le système à partir des fichiers XML.

```
try {

    jList1.setModel(listModel1);
    jList2.setModel(listModel2);
    jList3.setModel(listModel3);
    jList4.setModel(listModel4);
    jList10.setModel(listModel10);
    listModel1.clear();
    listModel2.clear();
    listModel3.clear();
    listModel4.clear();
    listModel10.clear();
    File fXmlFile = new File("events.xml");
    DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
    DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
    Document doc = dBuilder.parse(fXmlFile);
    doc.getDocumentElement().normalize();
    NodeList nList = doc.getElementsByTagName("Modality");
    for (int temp = 0; temp < nList.getLength(); temp++) {
       Node nNode = nList.item(temp);
       if (nNode.getNodeType() == Node.ELEMENT_NODE) {
          Element eElement = (Element) nNode;
          listModel1.addElement(getTagValue("type", eElement));
          listModel2.addElement( getTagValue("event", eElement));
          listModel10.addElement( getTagValue("event", eElement));
          listModel3.addElement( getTagValue("startTime", eElement));
          listModel4.addElement(getTagValue("endTime", eElement));
        }
    }
} catch (Exception e) {
}
```

Figure 6.6 Code qui récupère les événements des fichiers XML

Une fois que les événements soient reçus par le système, le moteur de fusion commence par la vérification du vocabulaire afin de déterminer si ces événements sont définis dans l'ontologie sous forme d'instances ou non. Les événements trouvés sont conservés et les autres sont éliminés. La Figure 6.7 montre les événements reconnus avec leurs classes et ceux qui ne sont pas identifiés.

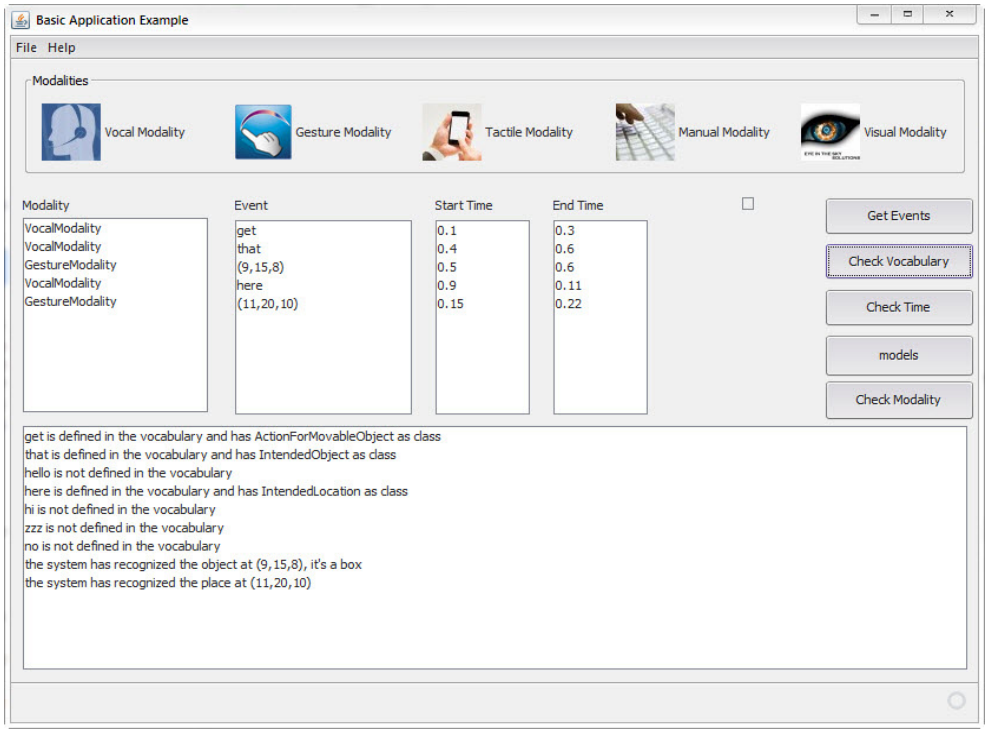


Figure 6.7 Reconnaissance des événements et élimination des autres

```
jList5.setModel(listModel5);
PrefixOWLOntologyFormat pm = man.getOntologyFormat(ont).asPrefixOWLOntologyFormat();
OWLDataFactory factory = man.getOWLDataFactory();
OWLReasoner reasoner = new StructuralReasoner(ont, new SimpleConfiguration(), BufferingMode.NON_BUFFERING);
for (int i = 0; i <listModel2.getSize(); i++) {
    OWLNamedIndividual martin = factory.getOWLNamedIndividual((String) (listModel2.getElementAt(i)), pm);
    Set<OWLClassExpression> assertedClasses = martin.getTypes(ont);
  if (assertedClasses.isEmpty() && !(listModel2.getElementAt(i).toString().contains("("))) {
        listModel5.addElement(listModel2.getElementAt(i) + " is not defined in the vocabulary");
        listModel2.removeElementAt(i);
        listModel1.removeElementAt(i);
        listModel3.removeElementAt(i);
        listModel4.removeElementAt(i);
        i = i - 1;
    }
    for (OWLClass c : reasoner.getTypes(martin, true).getFlattened()) {
        boolean asserted = assertedClasses.contains(c);
        if (asserted) {
            listModel5.addElement(listModel2.getElementAt(i) + " is defined in the vocabulary and has "
                    + renderer.render(c) + " as class");
        }
    }

}
```

Figure 6.8 Code de la vérification du vocabulaire

À la suite de la vérification du vocabulaire, le système vérifie la sémantique des événements reconnus en exécutant le raisonneur. Celui-ci vérifie la consistance et l'inférence. La Figure

6.9 représente le résultat de la vérification sémantique dans le système et La Figure 6.10 montre le code qui exécute le raisonneur afin d'inférer les événements.

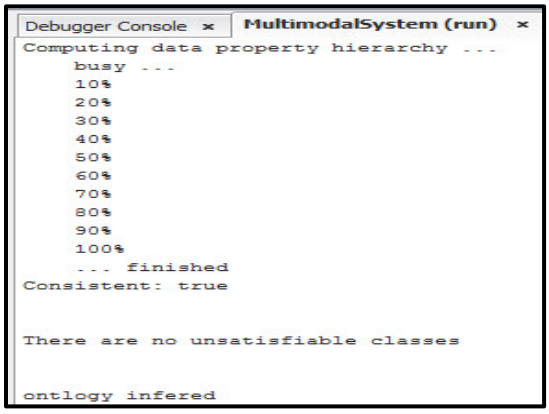


Figure 6.9 Résultat du raisonnement

```
public void shouldCreateInferredAxioms() throws OWLOntologyCreationException,OWLOntologyStorageException {
        OWLReasonerFactory reasonerFactory = new StructuralReasonerFactory();
       // OWLOntologyManager man = OWLManager.createOWLOntologyManager();
           // OWLOntology ont = man.loadOntologyFromOntologyDocument(IRI.create(PIZZA_IRI));
            OWLReasoner reasoner = reasonerFactory.createNonBufferingReasoner(ont);
            reasoner.precomputeInferences(InferenceType.CLASS_HIERARCHY);
            List<InferredAxiomGenerator<? extends OWLAxiom>> gens =
                    new ArrayList<InferredAxiomGenerator<? extends OWLAxiom>>();
            gens.add(new InferredSubClassAxiomGenerator());
            OWLOntology infOnt = man.createOntology();
            InferredOntologyGenerator iog = new InferredOntologyGenerator(reasoner, gens);
            iog.fillOntology(man, infOnt);
            man.saveOntology(infOnt, new StringDocumentTarget());
            System.out.println("ontlogy infered");
        }
public void shouldUseReasoner() throws OWLOntologyCreationException {
            System.out.println("Loaded " + ont.getOntologyID());
            OWLReasonerFactory reasonerFactory = new StructuralReasonerFactory();
            ConsoleProgressMonitor progressMonitor = new ConsoleProgressMonitor();
            OWLReasonerConfiguration config = new SimpleConfiguration(progressMonitor);
            OWLReasoner reasoner = reasonerFactory.createReasoner(ont, config);
             reasoner.precomputeInferences();
            boolean consistent = reasoner.isConsistent();
            System.out.println("Consistent: " + consistent);
            System.out.println("\n");
            org.semanticweb.owlapi.reasoner.Node<OWLClass> bottomNode = reasoner.getUnsatisfiableClasses();
            Set<OWLClass> unsatisfiable = bottomNode.getEntitiesMinusBottom();
            if (!unsatisfiable.isEmpty()) {
                System.out.println("The following classes are unsatisfiable: ");
                for (OWLClass cls : unsatisfiable) {
                  System.out.println("    " + cls);
                }
            } else {
                System.out.println("There are no unsatisfiable classes");
            }
            System.out.println("\n");

             }
```

Figure 6.10 Code du raisonnement

Une fois la sémantique vérifiée, le système vérifie l'aspect temporel en comparant le temps entre les événements et le temps total des événements pour déterminer s'ils respectent les paramètres temporels définis. La Figure 6.11 montre le résultat de la vérification temporelle et la Figure 6.12 montre le code correspondant. Nous avons considéré dans cette comparaison que le temps maximal d'une commande est de 10 secondes et que le temps maximal entre deux événements successifs est de 5 secondes.

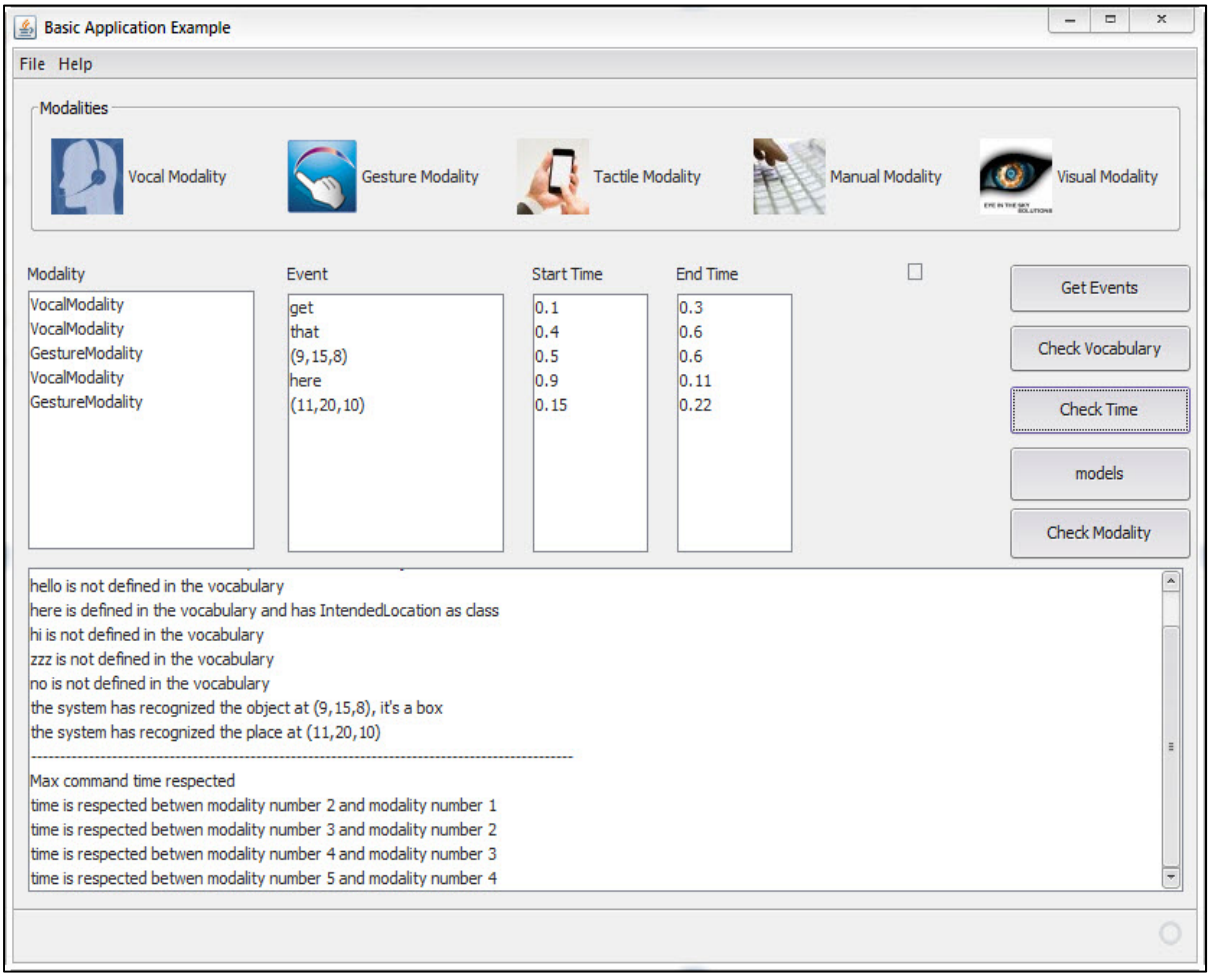


Figure 6.11 Résultat de la vérification temporelle

Après la vérification du vocabulaire, de la sémantique et du temps, le moteur de fusion vérifie la dernière condition qui est le matching entre les événements détectés et les modèles définies dans l'ontologie. Cette vérification a pour but de déduire une combinaison qui peut

former une commande complète et qui a un sens. Comme nous l'avons mentionné dans la section 4.4, cela est fait par l'exécution d'une requête qui est formée par les types des événements reconnus et donne comme résultat le modèle associé. La Figure 6.13 montre que le modèle 08 a été reconnu et par la suite le résultat de la fusion est «*get that (pen) here (11, 20, 10)* ». La Figure 6.14 représente le code pour la vérification du modèle.

```
float maxcomtime = (float) 0.0;
float modalitytime = (float) 0.0;
jList5.setModel(listModel5);
String mt = null;
String ct = null;
listModel5.addElement("------------------------------------------------------------"
        + "-----------------------------------");
OWLDataFactory factory = man.getOWLDataFactory();
OWLReasoner reasoner = new StructuralReasoner(ont, new SimpleConfiguration(), BufferingMode.NON_BUFFERING);
PrefixOWLOntologyFormat pm = man.getOntologyFormat(ont).asPrefixOWLOntologyFormat();
OWLClass ModalityTime = factory.getOWLClass(":MaxActiveModalityTime", pm);
OWLClass CommandTime = factory.getOWLClass(":MaxCommandTime", pm);
for (OWLNamedIndividual MaxActiveModalityTime : reasoner.getInstances(ModalityTime, false).getFlattened()) {
    mt = (renderer.render(MaxActiveModalityTime));
}
    for (OWLNamedIndividual MaxCommandTime : reasoner.getInstances(CommandTime, false).getFlattened()) {
    ct = (renderer.render(MaxCommandTime));
    }
    String first = (listModel3.getElementAt(0)).toString();
    String last = (listModel4.lastElement()).toString();
    maxcomtime = Float.parseFloat(first) + Float.parseFloat(last);
    if (maxcomtime <= Integer.parseInt(ct)) {
         listModel5.addElement("Max command time respected");
    } else {
         listModel5.addElement("Max command time not respected");
    }
    for (int i = 0; i <listModel3.size(); i++) {
         String firstValue = listModel3.getElementAt(i + 1).toString();
         String secondValue = listModel3.getElementAt(i).toString();
         modalitytime = Float.parseFloat(firstValue) - Float.parseFloat(secondValue);
         System.out.println(+modalitytime);
         if (modalitytime <= Integer.parseInt(mt)) {
        listModel5.addElement("time is respected betwen modality number " + (i + 2) +
                " and modality number " + (i + 1));
         }
    }
listModel5.addElement("------------------------------------------------"
        + "------------------------------------------------");
```

Figure 6.12 Code de la vérification temporelle

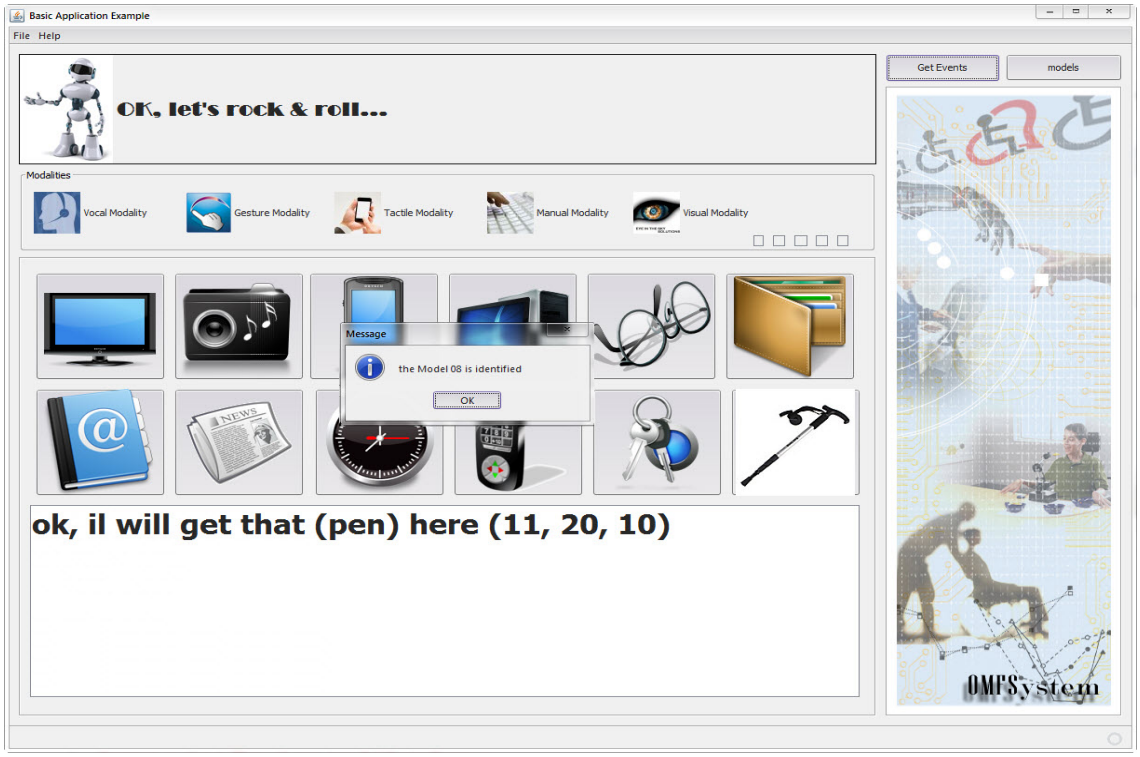


Figure 6.13 Résultat final de la fusion après reconnaissance du modèle

```
PrefixOWLOntologyFormat pm = man.getOntologyFormat(ont).asPrefixOWLOntologyFormat();
            OWLDataFactory factory = man.getOWLDataFactory();
            OWLReasoner reasoner = new StructuralReasoner(ont, new SimpleConfiguration(), BufferingMode.NON_BUFFERING);
            for (int i = 0; i <listModel2.getSize(); i++) {
                OWLNamedIndividual martin = factory.getOWLNamedIndividual((String) (listModel2.getElementAt(i)), pm);
                Set<OWLClassExpression> assertedClasses = martin.getTypes(ont);
                 for (OWLClass c : reasoner.getTypes(martin, true).getFlattened()) {
                    boolean asserted = assertedClasses.contains(c);
                    if (asserted) {
                         + renderer.render(c) + " as class");
                    }
            OWLOntology ont;
            OWLOntologyManager man = OWLManager.createOWLOntologyManager();
            String path = "file:/Users/Ahmad/Documents/NetBeansProjects/FusionSystem/EnvironmentOnto.rdf-xml.owl";
            ont = man.loadOntologyFromOntologyDocument(IRI.create(path));
            System.out.println("Loaded: " + ont.getOntologyID());

            OWLModel owlModel = ProtegeOWL.createJenaOWLModelFromURI(path);
            String rule5 ="tbox:isDirectSuperClassOf(?m, + "+renderer.render(c)+") ∧
                    + tbox:isDirectSuperClassOf(?m, "+ renderer.render(c+1)+") "
                    + "∧  tbox:isDirectSuperClassOf(?m,"+ + renderer.render(c)"+) →  sqwrl:selectDistinct(?m)";

              String qname5 = "Rule5";
            queryEngine.createSQWRLQuery(qname5, rule5);
            SQWRLResult result5 = queryEngine.runSQWRLQuery(qname5);
            JOptionPane.showMessageDialog(null,"the Model "+f+" is identified");
             System.out.println("ok i will "+ listModel2.getElementAt(i));
                }
```

Figure 6.14 Code pour la vérification du modèle

6.3 Conclusion

Dans ce chapitre, un prototype a été implémenté pour appliquer le scénario « *get that here* ». Le but était de démontrer que l'architecture proposé est applicable dans un système réel. Ce prototype s'est focalisé sur la technique du moteur de fusion, mais il peut être développé d'une manière complète pour assurer les différents services selon les spécifications requises.

Le système développé est dédié à la fusion multimodale. L'objectif initial était de comprendre l'environnement et ses événements. Pour qu'il soit complet, il serait nécessaire d'intégrer un module de fission responsable de l'exécution de la commande reconnue.

Avec l'OWL API, la connexion entre le langage de développement JAVA et l'ontologie est établie. Le raisonnement et l'inférence des différentes classes et propriétés ont été vérifiés et les requêtes responsables de la sélection des modalités et des modèles ont pu être exécutées.

CONCLUSION

L’avancement technologique actuel a créé la nécessité de produire des systèmes de plus en plus performants, faciles à utiliser et permettant de répondre aux besoins des utilisateurs. Pour atteindre cet objectif, les systèmes multimodaux ont été développés pour combiner plusieurs modalités en fonction de la tâche, des préférences et des intentions communicationnelles.

Au cours de notre étude, nous avons modélisé et implémenté une architecture de fusion multimodale qui permet l'interaction entre l’homme et la machine. Cette architecture est fondée sur la compréhension de l’environnement et l’établissement de la relation entre ses différents objets. Dans ce projet, plusieurs modalités d'entrée ont été utilisées (la parole, le toucher, le geste, le mouvement des yeux, etc.). Le choix des modalités appropriées repose sur un mécanisme de sélection selon le contexte. Un moteur de fusion a été développé en se basant sur un processus de vérification des pré-conditions. Cette architecture est conçue en définissant un ensemble de concepts, de propriétés de l'objet et de données et des instances dans une ontologie. Lors de cette conception, nous avons pris en considération les contextes de l’environnement, de l’utilisateur et de localisation, associés à chaque modalité. Un algorithme de fusion a été défini. Celui-ci tient compte du vocabulaire, de l'ordre des événements, de la sémantique et de l'aspect temporel. Pour assurer la cohérence, de nombreuses règles SWRL ont été créées dans l’ontologie, en utilisant le langage OWL, un standard du W3C. Ce choix rend la solution proposée formelle. Des requêtes SQWRL ont été créées afin de sélectionner la meilleure modalité adaptée à un contexte et de choisir le modèle approprié à une combinaison d’événements. Une validation à l’aide de réseaux de Petri colorés a été réalisée pour s’assurer du bon fonctionnement du système. Un prototype a également été développé.

Cette thèse est un subtil mélange de génie logiciel, de conception de l'architecture, de la représentation des connaissances et du raisonnement. L'ontologie a été utilisée pour conceptualiser la connaissance de l'environnement et la rendre explicite et compréhensible. Le Web sémantique exploite les ontologies pour fournir des ressources sémantiquement

significatives. Les points principaux de cette thèse résident dans la compréhension de l'environnement, l'extraction, la création et l'exploitation de la signification des événements qui se déroulent dans l'environnement. La fusion multimodale fournit une interaction homme- machine plus efficace à l'aide de modalités complémentaires ou redondantes.

Comme perspectives, il serait très intéressant de rendre complet le cycle qui forme un système d'interaction multimodale, par l'ajout d'une autre partie très importante qui est la fission. En effet, un vrai système multimodal est formé à la fois par la fusion et la fission. L'ontologie peut être enrichie par de nouvelles connaissances afin de prendre en considération le plus grand nombre possible de scénarios qui peuvent être appliqués dans l'environnement. L'une des techniques d'enrichissement de l'ontologie serait le traitement des événements éliminés. Il serait également possible d'introduire des paramètres et une logique floue pour le choix des décisions prises, surtout au niveau de la sélection des modalités. Par exemple, au lieu de les désactiver, il serait judicieux de pouvoir prédire les événements qui seraient détectés dans des cas où un contexte n'a pas été respecté. De plus, si nos travaux ont prouvé leur validité à travers différentes expérimentations, pour faire sortir ce travail du monde de la recherche et permettre son intégration dans un contexte plus grand public, il faudra réaliser des expérimentations bien plus larges dans des domaines applicatifs plus variés et en incluant notamment des personnes réellement naïves (c'est-à-dire ne sachant absolument pas comment fonctionne un système multimodal et quelles en sont les limites). Ce type d'expérimentation est primordial afin d'évaluer l'acceptabilité de notre interface par les différents utilisateurs. Le prototype peut être amélioré pour contrôler totalement l'ontologie, surtout si de nouvelles connaissances interviennent. Des options plus diverses peuvent être proposées aux utilisateurs afin de leur permettre un contrôle plus efficace au niveau du système multimodal.

ANNEXE I

Déclarations du réseau du Petri coloré dans le chapitre 5

Ces pages représentent les déclarations des variables et des fonctions de monitor dans CPN-Tools

```
colset UNIT = unit;
colset INT = int;
colset STRING = string;
colset BOOL = bool;
colset Event = with event timed;
colset Memory = product INT*STRING;
colset Object = with object timed;
colset ObjectAttribut = product Object * INT * INT;
colset INTxINTxINT= product INT*INT*INT;
colset STRINGxSTRINGxINT = product STRING*STRING*INT;
colset INTxINT= product INT*INT;
colset STRINGxINT = product STRING*INT;
colset STRINGxSTRING = product STRING*STRING;
fun round x = floor(x+0.5);
fun inTime() = IntInf.toInt(time());
fun discExp x = round(exponential(x));
fun NormalLaw(x,y) = round(normal(x,y));
var Current,Active,light,Noise: BOOL;
val InterToArrive = ref 0.5;
var person,location:STRINGxSTRINGxINT;
var handicap,decision,model1,model2,model3,Act,Loc,Obj,Per,Mod,c,c1,c2,c3,c4,d:
STRING;
var NextTime, TimeRecon, TimeExecution, n, p, nTreat,t,Index:INT;
var voice,movement: INTxINT;
fun pred (bindelem) =
```

```
let
  fun predBindElem (Fusion'Event_Collection (1,
                        {Act,Loc,Mod,Obj,Per,Ta,Tb,Tc,Td,
                        c1,c2,c3,c4})) = true
    | predBindElem _ = false
in
  predBindElem bindelem
end

fun obs (bindelem) =
let
  fun obsBindElem (Fusion'Event_Collection (1,
                        {Act,Loc,Mod="AFMO IO IL ",Obj,Per,Ta,Tb,Tc,Td,
                        c1,c2,c3,c4})) = 1
    | obsBindElem (Fusion'Event_Collection (1,
                        {Act,Loc,Mod="AFMO P IO ",Obj,Per,Ta,Tb,Tc,Td,
                        c1,c2,c3,c4})) = 2
    | obsBindElem _ = 0
in
  obsBindElem bindelem
end

fun pred (bindelem) =
let
  fun predBindElem (Action'Action_type_test (1,
                        {Act,Ta,c,d})) = true
    | predBindElem _ = false
in
  predBindElem bindelem
end
```

```
fun obs (bindelem) =
let
  fun obsBindElem (Action'Action_type_test (1,
                        {Act="Get ",Ta,c,d="Get "})) = 1
    | obsBindElem (Action'Action_type_test (1,
                        {Act="Give ",Ta,c,d="Give "})) = 2
    | obsBindElem (Action'Action_type_test (1,
                        {Act="Put ",Ta,c,d="Put "})) = 3
    | obsBindElem _ = 0
in
  obsBindElem bindelem
end

fun pred (bindelem) =
let
  fun predBindElem (Location'Location_type_test (1,
                        {Loc,Td,c,d})) = true
    | predBindElem _ = false
in
  predBindElem bindelem
end

fun obs (bindelem) =
let
  fun obsBindElem (Location'Location_type_test (1,
                        {Loc="there ",Td,c,d="there "})) = 2
    | obsBindElem (Location'Location_type_test (1,
                        {Loc="that ",Td,c,d="that "})) = 1
```

```
    | obsBindElem _ = 0
in
  obsBindElem bindelem
end

fun pred (bindelem) =
let
  fun predBindElem (Person'Person_type_test (1,
                        {Per,Tc,c,d})) = true
    | predBindElem _ = false
in
  predBindElem bindelem
end

fun obs (bindelem) =
let
  fun obsBindElem (Person'Person_type_test (1,
                        {Per="Father ",Tc,c,d="Father "})) = 1
    | obsBindElem (Person'Person_type_test (1,
                        {Per="me ",Tc,c,d="me "})) = 2
    | obsBindElem _ = 0
in
  obsBindElem bindelem
end
```

ANNEXE II

Rapports de simulations du réseau de Petri dans le chapitre 5

```
CPN Tools simulation report for:
/cygdrive/C/Users/Ahmad/Desktop/fusion engine last update 2.cpn
Report generated: Mon Jun 25 10:13:19 2012

1     0     Generator @ (1:Gen1)
 - n = 1
 - NextTime = 0
2     0     Recognition @ (1:Gen1)
 - n = 1
 - t = 0
 - nTreat = 0
 - p = 0
 - TimeRecon = 49
3     0     Execution @ (1:Gen1)
 - nTreat = 1
 - TimeExecution = 21
4     0     Context_verification @ (1:Gen1)
 - voice = (1,0)
 - Noise = true
5     0     vocal_context_test @ (1:Gen1)
 - Current = true
 - lim = 0
 - handicap = "Deaf"
 - a = 8
 - Active = false
6     0     vocal_context_test @ (1:Gen1)
 - Current = false
 - lim = 0
 - handicap = "Deaf"
 - a = 7
 - Active = true
7     0     Generator @ (1:Gen1)
 - n = 2
 - NextTime = 0
8     0     Recognition @ (1:Gen1)
 - n = 2
 - t = 0
 - nTreat = 0
 - p = 1
 - TimeRecon = 58
9     0     Execution @ (1:Gen1)
 - nTreat = 1
 - TimeExecution = 15
10    0     Context_verification @ (1:Gen1)
 - voice = (1,0)
```

```
 - Noise = true
11    0     Generator @ (1:Gen1)
 - n = 3
 - NextTime = 0
12    0     Recognition @ (1:Gen1)
 - n = 3
 - t = 0
 - nTreat = 0
 - p = 2
 - TimeRecon = 49
13    0     Execution @ (1:Gen1)
 - nTreat = 1
 - TimeExecution = 20
14    0     Context_verification @ (1:Gen1)
 - voice = (1,0)
 - Noise = true
15    0     Generator @ (1:Gen1)
 - n = 4
 - NextTime = 0
16    0     Recognition @ (1:Gen1)
 - n = 4
 - t = 0
 - nTreat = 0
 - p = 3
 - TimeRecon = 51
17    0     Execution @ (1:Gen1)
 - nTreat = 1
 - TimeExecution = 16
18    0     Context_verification @ (1:Gen1)
 - voice = (1,0)
 - Noise = true
19    0     vocal_context_test @ (1:Gen1)
 - Current = true
 - lim = 0
 - handicap = "Deaf"
 - a = 8
 - Active = false
20    0     Generator @ (1:Gen1)
 - n = 5
 - NextTime = 1
21    0     Recognition @ (1:Gen1)
 - n = 5
 - t = 0
 - nTreat = 0
 - p = 4
 - TimeRecon = 48
22    0     Execution @ (1:Gen1)
 - nTreat = 1
 - TimeExecution = 16
23    0     Context_verification @ (1:Gen1)
```

```
 - voice = (1,0)
 - Noise = false
24    0     rejected_values @ (1:Gen1)
25    0     visual_context_test @ (1:Gen2)
 - Current = true
 - lim = 0
 - a = 8
 - handicap = "Normal"
 - Active = false
26    0     vocal_context_test @ (1:Gen1)
 - Current = false
 - lim = 0
 - handicap = "Deaf"
 - a = 2
 - Active = true
27    0     Generator @ (1:Gen2)
 - n = 1
 - NextTime = 1
28    0     Recognition @ (1:Gen2)
 - n = 1
 - t = 0
 - nTreat = 0
 - p = 0
 - TimeRecon = 51
29    0     Execution @ (1:Gen2)
 - nTreat = 1
 - TimeExecution = 19
30    0     Context_verification @ (1:Gen2)
 - movement = (1,0)
 - light = false
31    0     rejected_values @ (1:Gen2)
32    0     visual_context_test @ (1:Gen2)
 - Current = false
 - lim = 0
 - a = 3
 - handicap = "blind"
 - Active = true
33    0     vocal_context_test @ (1:Gen1)
 - Current = true
 - lim = 0
 - handicap = "Deaf"
 - a = 3
 - Active = false
34    0     vocal_context_test @ (1:Gen1)
 - Current = false
 - lim = 0
 - handicap = "Normal"
 - a = 5
```

```
 - Active = true
35    0     Action_Ontology_test @ (1:Action)
 - Tmax = 50
 - Tmin = 0
 - Index = 1
 - Act = "Get "
 - E = 1
 - T = 0
36    0     Action_Found @ (1:Action)
 - Ta = 0
 - Act = "Get "
37    0     Action_type_test @ (1:Action)
 - Ta = 0
 - Act = "Get "
 - d = "Get "
 - c = "AFMO "
38    0     Person_Ontology_test @ (1:Person)
 - E = 1
 - T = 0
 - Tmax = 150
 - Tmin = 75
 - Per = "me "
 - Index = 2
39    0     Person_Ontology_test @ (1:Person)
 - E = 1
 - T = 0
 - Tmax = 150
 - Tmin = 75
 - Per = "me "
 - Index = 2
40    0     feedback @ (1:Person)
 - T = 0
41    0     feedback @ (1:Person)
 - T = 0
42    0     Person_Ontology_test @ (1:Person)
 - E = 1
 - T = 0
 - Tmax = 150
 - Tmin = 75
 - Per = "me "
 - Index = 2
43    0     feedback @ (1:Person)
 - T = 0
44    1     Generator @ (1:Gen1)
 - n = 6
 - NextTime = 0
45    1     Recognition @ (1:Gen1)
 - n = 6
 - t = 1
 - nTreat = 0
```

```
 - p = 5
 - TimeRecon = 47
46    1     Execution @ (1:Gen1)
 - nTreat = 1
 - TimeExecution = 22
47    1     Context_verification @ (1:Gen1)
 - voice = (1,1)
 - Noise = true
48    1     Generator @ (1:Gen1)
 - n = 7
 - NextTime = 1
49    1     Recognition @ (1:Gen1)
 - n = 7
 - t = 1
 - nTreat = 0
 - p = 6
 - TimeRecon = 48
50    1     Execution @ (1:Gen1)
 - nTreat = 1
 - TimeExecution = 21
51    1     vocal_context_test @ (1:Gen1)
 - Current = true
 - lim = 0
 - handicap = "Normal"
 - a = 8
 - Active = false
52    1     Context_verification @ (1:Gen1)
 - voice = (1,1)
 - Noise = false
53    1     rejected_values @ (1:Gen1)
54    1     Generator @ (1:Gen2)
 - n = 2
 - NextTime = 1
55    1     Recognition @ (1:Gen2)
 - n = 2
 - t = 1
 - nTreat = 0
 - p = 1
 - TimeRecon = 49
56    1     Execution @ (1:Gen2)
 - nTreat = 1
 - TimeExecution = 18
57    1     Context_verification @ (1:Gen2)
 - movement = (1,1)
 - light = true
58    1     visual_context_test @ (1:Gen2)
 - Current = true
 - lim = 0
```

```
 - a = 1
 - handicap = "Normal"
 - Active = false
59    1     Object_Ontology_test @ (1:Object)
 - Tmax = 75
 - Tmin = 0
 - Obj = "that "
 - Index = 1
 - E = 1
 - T = 1
 - E2 = 1
 - T2 = 1
60    1     Object_Found @ (1:Object)
 - Tb = 1
 - Obj = "that "
61    1     Object_type_test @ (1:Object)
 - Tb = 1
 - Obj = "that "
 - d = "that "
 - c = "IO "
62    1     vocal_context_test @ (1:Gen1)
 - Current = false
 - lim = 0
 - handicap = "Normal"
 - a = 2
 - Active = true
63    2     Generator @ (1:Gen1)
 - n = 8
 - NextTime = 0
64    2     Generator @ (1:Gen1)
 - n = 9
 - NextTime = 1
65    2     Recognition @ (1:Gen1)
 - n = 8
 - t = 2
 - nTreat = 0
 - p = 7
 - TimeRecon = 48
66    2     Execution @ (1:Gen1)
 - nTreat = 1
 - TimeExecution = 24
67    2     Recognition @ (1:Gen1)
 - n = 9
 - t = 2
 - nTreat = 0
 - p = 8
 - TimeRecon = 52
68    2     Context_verification @ (1:Gen1)
 - voice = (1,2)
 - Noise = true
```

```
69    2     vocal_context_test @ (1:Gen1)
 - Current = true
 - lim = 0
 - handicap = "Deaf"
 - a = 1
 - Active = false
70    2     Execution @ (1:Gen1)
 - nTreat = 1
 - TimeExecution = 21
71    2     Context_verification @ (1:Gen1)
 - voice = (1,2)
 - Noise = false
72    2     rejected_values @ (1:Gen1)
73    2     vocal_context_test @ (1:Gen1)
 - Current = false
 - lim = 0
 - handicap = "Deaf"
 - a = 5
 - Active = true
74    2     Generator @ (1:Gen2)
 - n = 3
 - NextTime = 0
75    2     Recognition @ (1:Gen2)
 - n = 3
 - t = 2
 - nTreat = 0
 - p = 2
 - TimeRecon = 48
76    2     Execution @ (1:Gen2)
 - nTreat = 1
 - TimeExecution = 22
77    2     Generator @ (1:Gen2)
 - n = 4
 - NextTime = 0
78    2     Context_verification @ (1:Gen2)
 - movement = (1,2)
 - light = false
79    2     rejected_values @ (1:Gen2)
80    2     Recognition @ (1:Gen2)
 - n = 4
 - t = 2
 - nTreat = 0
 - p = 3
 - TimeRecon = 49
81    2     visual_context_test @ (1:Gen2)
 - Current = false
 - lim = 0
 - a = 5
```

```
 - handicap = "Normal"
 - Active = true
82    2     Execution @ (1:Gen2)
 - nTreat = 1
 - TimeExecution = 20
83    2     Context_verification @ (1:Gen2)
 - movement = (1,2)
 - light = true
84    2     visual_context_test @ (1:Gen2)
 - Current = true
 - lim = 0
 - a = 5
 - handicap = "Normal"
 - Active = false
85    2     Generator @ (1:Gen2)
 - n = 5
 - NextTime = 0
86    2     Location_Ontology_test @ (1:Location)
 - E2 = 1
 - T2 = 2
 - E = 1
 - T = 2
 - Tmax = 75
 - Tmin = 0
 - Loc = "here "
 - Index = 1
87    2     Location_Found @ (1:Location)
 - Td = 2
 - Loc = "here "
88    2     Location_type_test @ (1:Location)
 - Td = 2
 - Loc = "here "
 - d = "here "
 - c = "IL "
89    2     AFMO_IO_IL @ (1:Fusion)
 - Tb = 1
 - c2 = "IO "
 - Obj = "that "
 - Ta = 0
 - c1 = "AFMO "
 - Act = "Get "
 - Td = 2
 - c4 = "IL "
 - Loc = "here "
 - person = ("","",0)
90    2     Event_Collection @ (1:Fusion)
 - Td = 2
 - c4 = "IL "
 - Loc = "here "
 - Per = ""
```

```
 - Tc = 0
 - c3 = ""
 - Tb = 1
 - c2 = "IO "
 - Obj = "that "
 - Ta = 0
 - c1 = "AFMO "
 - Act = "Get "
 - Mod = "AFMO IO IL "
91    2     Decision @ (1:Fusion)
 - model3 = "Get that here "
 - model2 = ""
 - model1 = ""
 - decision = "Get that here "
92    2     command_filter @ (1:Fusion)
 - decision = "Get that here "


CPN Tools simulation report for:
/cygdrive/C/Users/Ahmad/Desktop/general net.cpn
Report generated: Tue Jun 26 11:18:29 2012

1     0     visual_context_test @ (1:Gen4)
 - Current = true
 - lim = 0
 - a = 8
 - handicap = "Manual"
 - Active = false
2     0     Generator @ (1:Gen2)
 - n = 1
 - NextTime = 0
3     0     Generator @ (1:Gen4)
 - n = 1
 - NextTime = 0
4     0     Generator @ (1:Gen1)
 - n = 1
 - NextTime = 1
5     0     Generator @ (1:Gen4)
 - n = 2
 - NextTime = 0
6     0     vocal_context_test @ (1:Gen1)
 - Current = true
 - lim = 0
 - handicap = "Mute"
 - a = 8
 - Active = false
7     0     Recognition @ (1:Gen4)
 - n = 1
 - t = 0
```

```
 - nTreat = 0
 - p = 0
 - TimeRecon = 50
8     0     visual_context_test @ (1:Gen3)
 - Current = true
 - lim = 0
 - a = 2
 - handicap = "blind"
 - Active = false
9     0     Generator @ (1:Gen4)
 - n = 3
 - NextTime = 1
10    0     Recognition @ (1:Gen4)
 - n = 2
 - t = 0
 - nTreat = 1
 - p = 1
 - TimeRecon = 50
11    0     visual_context_test @ (1:Gen2)
 - Current = true
 - lim = 0
 - a = 9
 - handicap = "Manual"
 - Active = false
12    0     Recognition @ (1:Gen4)
 - n = 3
 - t = 0
 - nTreat = 2
 - p = 2
 - TimeRecon = 47
13    0     Generator @ (1:Gen3)
 - n = 1
 - NextTime = 1
14    0     Recognition @ (1:Gen1)
 - n = 1
 - t = 0
 - nTreat = 0
 - p = 0
 - TimeRecon = 48
15    0     Generator @ (1:Gen5)
 - n = 1
 - NextTime = 2
16    0     Execution @ (1:Gen4)
 - nTreat = 3
 - TimeExecution = 24
17    0     Generator @ (1:Gen2)
 - n = 2
 - NextTime = 0
18    0     Recognition @ (1:Gen2)
 - n = 2
```

```
 - t = 0
 - nTreat = 0
 - p = 0
 - TimeRecon = 54
19    0     Recognition @ (1:Gen2)
 - n = 1
 - t = 0
 - nTreat = 1
 - p = 1
 - TimeRecon = 54
20    0     Generator @ (1:Gen2)
 - n = 3
 - NextTime = 1
21    0     Execution @ (1:Gen2)
 - nTreat = 2
 - TimeExecution = 21
22    0     Recognition @ (1:Gen2)
 - n = 3
 - t = 0
 - nTreat = 1
 - p = 2
 - TimeRecon = 52
23    0     Execution @ (1:Gen1)
 - nTreat = 1
 - TimeExecution = 19
24    0     Recognition @ (1:Gen3)
 - n = 1
 - t = 0
 - nTreat = 0
 - p = 0
 - TimeRecon = 51
25    0     Execution @ (1:Gen3)
 - nTreat = 1
 - TimeExecution = 20
26    0     Execution @ (1:Gen4)
 - nTreat = 2
 - TimeExecution = 22
27    0     Execution @ (1:Gen2)
 - nTreat = 2
 - TimeExecution = 19
28    0     Context_verification @ (1:Gen1)
 - Noise = false
 - voice = (1,0)
29    0     Context_verification @ (1:Gen3)
 - movement = (1,0)
 - light = false
30    0     Recognition @ (1:Gen5)
 - n = 1
```

```
 - t = 0
 - nTreat = 0
 - p = 0
 - TimeRecon = 47
31    0     Context_verification @ (1:Gen4)
 - movement = (2,0)
 - light = false
32    0     Context_verification @ (1:Gen4)
 - movement = (3,0)
 - light = false
33    0     Context_verification @ (1:Gen2)
 - movement = (2,0)
 - light = false
34    0     Person_Ontology_test @ (1:Person)
 - E = 0
 - T = 0
 - Per = "Father "
 - Index = 1
35    0     visual_context_test @ (1:Gen4)
 - Current = false
 - lim = 0
 - a = 6
 - handicap = "blind"
 - Active = true
36    0     rejected_values @ (1:Gen3)
37    0     feedback @ (1:Person)
 - T = 0
38    0     visual_context_test @ (1:Gen3)
 - Current = false
 - lim = 0
 - a = 7
 - handicap = "blind"
 - Active = true
39    0     Context_verification @ (1:Gen2)
 - movement = (2,0)
 - light = false
40    0     rejected_values @ (1:Gen2)
41    0     Execution @ (1:Gen4)
 - nTreat = 1
 - TimeExecution = 16
42    0     rejected_values @ (1:Gen2)
43    0     Context_verification @ (1:Gen4)
 - movement = (1,0)
 - light = true
44    0     Execution @ (1:Gen2)
 - nTreat = 1
 - TimeExecution = 22
45    0     Context_verification @ (1:Gen2)
 - movement = (1,0)
 - light = false
```

```
46    0     visual_context_test @ (1:Gen5)
 - Current = true
 - lim = 0
 - a = 6
 - handicap = "blind"
 - Active = false
47    0     visual_context_test @ (1:Gen2)
 - Current = false
 - lim = 0
 - a = 7
 - handicap = "Manual"
 - Active = true
48    0     rejected_values @ (1:Gen2)
49    0     visual_context_test @ (1:Gen4)
 - Current = true
 - lim = 0
 - a = 2
 - handicap = "Normal"
 - Active = false
50    0     visual_context_test @ (1:Gen2)
 - Current = true
 - lim = 0
 - a = 8
 - handicap = "Manual"
 - Active = false
51    0     visual_context_test @ (1:Gen4)
 - Current = false
 - lim = 0
 - a = 6
 - handicap = "Manual"
 - Active = true
52    0     rejected_values @ (1:Gen4)
53    0     Execution @ (1:Gen5)
 - nTreat = 1
 - TimeExecution = 13
54    0     Context_verification @ (1:Gen5)
 - movement = (1,0)
 - light = false
55    0     visual_context_test @ (1:Gen5)
 - Current = false
 - lim = 0
 - a = 9
 - handicap = "blind"
 - Active = true
56    0     rejected_values @ (1:Gen4)
57    0     visual_context_test @ (1:Gen2)
 - Current = false
 - lim = 0
```

```
 - a = 3
 - handicap = "Normal"
 - Active = true
58    0     rejected_values @ (1:Gen5)
59    1     Generator @ (1:Gen4)
 - n = 4
 - NextTime = 1
60    1     Generator @ (1:Gen3)
 - n = 2
 - NextTime = 1
61    1     Recognition @ (1:Gen4)
 - n = 4
 - t = 1
 - nTreat = 0
 - p = 3
 - TimeRecon = 47
62    1     Generator @ (1:Gen1)
 - n = 2
 - NextTime = 0
63    1     Generator @ (1:Gen2)
 - n = 4
 - NextTime = 1
64    1     Execution @ (1:Gen4)
 - nTreat = 1
 - TimeExecution = 19
65    1     Recognition @ (1:Gen3)
 - n = 2
 - t = 1
 - nTreat = 0
 - p = 1
 - TimeRecon = 48
66    1     Generator @ (1:Gen1)
 - n = 3
 - NextTime = 0
CPN Tools simulation report for:
/cygdrive/C/Users/Ahmad/Desktop/fusion engine last update 2.cpn
Report generated: Mon Jun 25 16:00:37 2012

1     0     Generator @ (1:Gen1)
 - n = 1
 - NextTime = 0
2     0     Generator @ (1:Gen1)
 - n = 2
 - NextTime = 1
3     0     Recognition @ (1:Gen1)
 - n = 1
 - t = 0
 - nTreat = 0
 - p = 0
 - TimeRecon = 50
```

```
4     0     Execution @ (1:Gen1)
 - nTreat = 1
 - TimeExecution = 18
5     0     Context_verification @ (1:Gen1)
 - voice = (1,0)
 - Noise = true
6     0     Recognition @ (1:Gen1)
 - n = 2
 - t = 0
 - nTreat = 0
 - p = 1
 - TimeRecon = 46
7     0     Execution @ (1:Gen1)
 - nTreat = 1
 - TimeExecution = 18
8     0     Context_verification @ (1:Gen1)
 - voice = (1,0)
 - Noise = true
9     0     vocal_context_test @ (1:Gen1)
 - Current = true
 - lim = 0
 - handicap = "Normal"
 - a = 10
 - Active = false
10    0     vocal_context_test @ (1:Gen1)
 - Current = false
 - lim = 0
 - handicap = "Mute"
 - a = 3
 - Active = true
11    0     vocal_context_test @ (1:Gen1)
 - Current = true
 - lim = 0
 - handicap = "Mute"
 - a = 5
 - Active = false
12    0     visual_context_test @ (1:Gen2)
 - Current = true
 - lim = 0
 - a = 3
 - handicap = "blind"
 - Active = false
13    0     Generator @ (1:Gen2)
 - n = 1
 - NextTime = 0
14    0     Generator @ (1:Gen2)
 - n = 2
 - NextTime = 0
```

```
15    0     Generator @ (1:Gen2)
 - n = 3
 - NextTime = 0
16    0     Recognition @ (1:Gen2)
 - n = 2
 - t = 0
 - nTreat = 0
 - p = 0
 - TimeRecon = 51
17    0     Recognition @ (1:Gen2)
 - n = 1
 - t = 0
 - nTreat = 1
 - p = 1
 - TimeRecon = 53
18    0     Generator @ (1:Gen2)
 - n = 4
 - NextTime = 0
19    0     Recognition @ (1:Gen2)
 - n = 3
 - t = 0
 - nTreat = 2
 - p = 2
 - TimeRecon = 44
20    0     Recognition @ (1:Gen2)
 - n = 4
 - t = 0
 - nTreat = 3
 - p = 3
 - TimeRecon = 47
21    0     Execution @ (1:Gen2)
 - nTreat = 4
 - TimeExecution = 18
22    0     Context_verification @ (1:Gen2)
 - movement = (4,0)
 - light = false
23    0     Execution @ (1:Gen2)
 - nTreat = 3
 - TimeExecution = 19
24    0     Execution @ (1:Gen2)
 - nTreat = 2
 - TimeExecution = 17
25    0     Generator @ (1:Gen2)
 - n = 5
 - NextTime = 0
26    0     Generator @ (1:Gen2)
 - n = 6
 - NextTime = 1
27    0     rejected_values @ (1:Gen2)
28    0     Context_verification @ (1:Gen2)
```

```
 - movement = (2,0)
 - light = false
29    0     rejected_values @ (1:Gen2)
30    0     visual_context_test @ (1:Gen2)
 - Current = false
 - lim = 0
 - a = 10
 - handicap = "Normal"
 - Active = true
31    0     Execution @ (1:Gen2)
 - nTreat = 1
 - TimeExecution = 17
32    0     Context_verification @ (1:Gen2)
 - movement = (3,0)
 - light = true
33    0     visual_context_test @ (1:Gen2)
 - Current = true
 - lim = 0
 - a = 2
 - handicap = "Normal"
 - Active = false
34    0     Context_verification @ (1:Gen2)
 - movement = (1,0)
 - light = false
35    0     rejected_values @ (1:Gen2)
36    0     Recognition @ (1:Gen2)
 - n = 5
 - t = 0
 - nTreat = 0
 - p = 4
 - TimeRecon = 54
37    0     Execution @ (1:Gen2)
 - nTreat = 1
 - TimeExecution = 22
38    0     Recognition @ (1:Gen2)
 - n = 6
 - t = 0
 - nTreat = 0
 - p = 5
 - TimeRecon = 51
39    0     visual_context_test @ (1:Gen2)
 - Current = false
 - lim = 0
 - a = 7
 - handicap = "Normal"
 - Active = true
40    0     Context_verification @ (1:Gen2)
 - movement = (1,0)
```

```
 - light = true
41    0     visual_context_test @ (1:Gen2)
 - Current = true
 - lim = 0
 - a = 6
 - handicap = "blind"
 - Active = false
42    0     Action_Ontology_test @ (1:Action)
 - Tmax = 50
 - Tmin = 0
 - Index = 1
 - Act = "Get "
 - E = 1
 - T = 0
43    0     Action_Found @ (1:Action)
 - Ta = 0
 - Act = "Get "
44    0     Action_type_test @ (1:Action)
 - Ta = 0
 - Act = "Get "
 - d = "Get "
 - c = "AFMO "
45    0     Person_Ontology_test @ (1:Person)
 - E = 1
 - T = 0
 - Tmax = 75
 - Tmin = 0
 - Per = "Father "
 - Index = 1
46    0     Person_Found @ (1:Person)
 - Per = "Father "
 - Tc = 0
47    0     Person_type_test @ (1:Person)
 - Per = "Father "
 - Tc = 0
 - d = "Get "
 - c = "AFMO "
48    0     Rejected_values @ (1:Person)
49    0     visual_context_test @ (1:Gen2)
 - Current = false
 - lim = 0
 - a = 9
 - handicap = "Manual"
 - Active = true
50    0     Execution @ (1:Gen2)
 - nTreat = 1
 - TimeExecution = 19
51    0     Context_verification @ (1:Gen2)
 - movement = (1,0)
 - light = true
```

```
52    0     visual_context_test @ (1:Gen2)
 - Current = true
 - lim = 0
 - a = 4
 - handicap = "blind"
 - Active = false
53    1     Generator @ (1:Gen2)
 - n = 7
 - NextTime = 1
54    1     Recognition @ (1:Gen2)
 - n = 7
 - t = 1
 - nTreat = 0
 - p = 6
 - TimeRecon = 48
55    1     Execution @ (1:Gen2)
 - nTreat = 1
 - TimeExecution = 21
56    1     Context_verification @ (1:Gen2)
 - movement = (1,1)
 - light = false
57    1     visual_context_test @ (1:Gen2)
 - Current = false
 - lim = 0
 - a = 9
 - handicap = "Manual"
 - Active = true
58    1     rejected_values @ (1:Gen2)
59    1     Generator @ (1:Gen1)
 - n = 3
 - NextTime = 0
60    1     Generator @ (1:Gen1)
 - n = 4
 - NextTime = 2
61    1     Recognition @ (1:Gen1)
 - n = 4
 - t = 1
 - nTreat = 0
 - p = 2
 - TimeRecon = 51
62    1     Recognition @ (1:Gen1)
 - n = 3
 - t = 1
 - nTreat = 1
 - p = 3
 - TimeRecon = 48
63    1     Execution @ (1:Gen1)
 - nTreat = 2
```

```
 - TimeExecution = 21
64    1     Context_verification @ (1:Gen1)
 - voice = (2,1)
 - Noise = false
65    1     Execution @ (1:Gen1)
 - nTreat = 1
 - TimeExecution = 18
66    1     vocal_context_test @ (1:Gen1)
 - Current = false
 - lim = 0
 - handicap = "Deaf"
 - a = 3
 - Active = true
67    1     rejected_values @ (1:Gen1)
68    1     Context_verification @ (1:Gen1)
 - voice = (1,1)
 - Noise = true
69    1     vocal_context_test @ (1:Gen1)
 - Current = true
 - lim = 0
 - handicap = "Normal"
 - a = 5
 - Active = false
70    1     Object_Ontology_test @ (1:Object)
 - Tmax = 75
 - Tmin = 0
 - Obj = "that "
 - Index = 1
 - E = 1
 - T = 1
 - E2 = 1
 - T2 = 0
71    1     Object_Found @ (1:Object)
 - Tb = 1
 - Obj = "that "
72    1     Object_type_test @ (1:Object)
 - Tb = 1
 - Obj = "that "
 - d = "that "
 - c = "IO "
73    2     Generator @ (1:Gen2)
 - n = 8
 - NextTime = 1
74    2     Recognition @ (1:Gen2)
 - n = 8
 - t = 2
 - nTreat = 0
 - p = 7
 - TimeRecon = 54
75    2     Execution @ (1:Gen2)
```

```
 - nTreat = 1
 - TimeExecution = 20
76    2     Context_verification @ (1:Gen2)
 - movement = (1,2)
 - light = true
77    2     visual_context_test @ (1:Gen2)
 - Current = true
 - lim = 0
 - a = 5
 - handicap = "Manual"
 - Active = false
78    3     Generator @ (1:Gen2)
 - n = 9
 - NextTime = 0
79    3     Generator @ (1:Gen1)
 - n = 5
 - NextTime = 1
80    3     Recognition @ (1:Gen1)
 - n = 5
 - t = 3
 - nTreat = 0
 - p = 4
 - TimeRecon = 47
81    3     Execution @ (1:Gen1)
 - nTreat = 1
 - TimeExecution = 19
82    3     Context_verification @ (1:Gen1)
 - voice = (1,3)
 - Noise = false
83    3     rejected_values @ (1:Gen1)
84    3     vocal_context_test @ (1:Gen1)
 - Current = false
 - lim = 0
 - handicap = "Normal"
 - a = 6
 - Active = true
85    3     Recognition @ (1:Gen2)
 - n = 9
 - t = 3
 - nTreat = 0
 - p = 8
 - TimeRecon = 48
86    3     Generator @ (1:Gen2)
 - n = 10
 - NextTime = 1
87    3     Execution @ (1:Gen2)
 - nTreat = 1
 - TimeExecution = 20
```

```
88    3     Context_verification @ (1:Gen2)
 - movement = (1,3)
 - light = false
89    3     Recognition @ (1:Gen2)
 - n = 10
 - t = 3
 - nTreat = 0
 - p = 9
 - TimeRecon = 46
90    3     visual_context_test @ (1:Gen2)
 - Current = false
 - lim = 0
 - a = 10
 - handicap = "Normal"
 - Active = true
91    3     Execution @ (1:Gen2)
 - nTreat = 1
 - TimeExecution = 20
92    3     rejected_values @ (1:Gen2)
93    3     Context_verification @ (1:Gen2)
 - movement = (1,3)
 - light = true
94    3     visual_context_test @ (1:Gen2)
 - Current = true
 - lim = 0
 - a = 6
 - handicap = "blind"
 - Active = false
95    4     Generator @ (1:Gen2)
 - n = 11
 - NextTime = 1
96    4     Recognition @ (1:Gen2)
 - n = 11
 - t = 4
 - nTreat = 0
 - p = 10
 - TimeRecon = 49
97    4     Execution @ (1:Gen2)
 - nTreat = 1
 - TimeExecution = 18
98    4     Context_verification @ (1:Gen2)
 - movement = (1,4)
 - light = false
99    4     rejected_values @ (1:Gen2)
100   4     Generator @ (1:Gen1)
 - n = 6
 - NextTime = 0
101   4     Recognition @ (1:Gen1)
 - n = 6
 - t = 4
```

```
 - nTreat = 0
 - p = 5
 - TimeRecon = 49
102   4     Execution @ (1:Gen1)
 - nTreat = 1
 - TimeExecution = 20
103   4     Context_verification @ (1:Gen1)
 - voice = (1,4)
 - Noise = true
104   4     Generator @ (1:Gen1)
 - n = 7
 - NextTime = 1
105   4     vocal_context_test @ (1:Gen1)
 - Current = true
 - lim = 0
 - handicap = "Mute"
 - a = 8
 - Active = false
106   4     Recognition @ (1:Gen1)
 - n = 7
 - t = 4
 - nTreat = 0
 - p = 6
 - TimeRecon = 51
107   4     Execution @ (1:Gen1)
 - nTreat = 1
 - TimeExecution = 22
108   4     Context_verification @ (1:Gen1)
 - voice = (1,4)
 - Noise = false
109   4     vocal_context_test @ (1:Gen1)
 - Current = false
 - lim = 0
 - handicap = "Mute"
 - a = 2
 - Active = true
110   4     Location_Ontology_test @ (1:Location)
 - E2 = 1
 - T2 = 3
 - E = 1
 - T = 4
 - Tmax = 75
 - Tmin = 0
 - Loc = "here "
 - Index = 1
111   4     Location_Found @ (1:Location)
 - Td = 4
 - Loc = "here "
```

```
112   4     Location_type_test @ (1:Location)
 - Td = 4
 - Loc = "here "
 - d = "here "
 - c = "IL "
113   4     AFMO_IO_IL @ (1:Fusion)
 - Tb = 1
 - c2 = "IO "
 - Obj = "that "
 - Ta = 0
 - c1 = "AFMO "
 - Act = "Get "
 - Td = 4
 - c4 = "IL "
 - Loc = "here "
 - person = ("","",0)
114   4     Event_Collection @ (1:Fusion)
 - Td = 4
 - c4 = "IL "
 - Loc = "here "
 - Per = ""
 - Tc = 0
 - c3 = ""
 - Tb = 1
 - c2 = "IO "
 - Obj = "that "
 - Ta = 0
 - c1 = "AFMO "
 - Act = "Get "
 - Mod = "AFMO IO IL "
115   4     Decision @ (1:Fusion)
 - model3 = "Get that here "
 - model2 = ""
 - model1 = ""
 - decision = "Get that here "
116   4     command_filter @ (1:Fusion)
 - decision = "Get that here "
117   4     rejected_values @ (1:Gen1)
118   4     visual_context_test @ (1:Gen2)
 - Current = false
 - lim = 0
 - a = 5
 - handicap = "Normal"
 - Active = true
119   5     Generator @ (1:Gen2)
 - n = 12
 - NextTime = 1
120   5     Recognition @ (1:Gen2)
 - n = 12
 - t = 5
```

```
 - nTreat = 0
 - p = 11
 - TimeRecon = 49
121   5     Execution @ (1:Gen2)
 - nTreat = 1
 - TimeExecution = 21
122   5     Context_verification @ (1:Gen2)
 - movement = (1,5)
 - light = true
123   5     visual_context_test @ (1:Gen2)
 - Current = true
 - lim = 0
 - a = 3
 - handicap = "Manual"
 - Active = false
124   5     Generator @ (1:Gen1)
 - n = 8
 - NextTime = 1
125   5     Recognition @ (1:Gen1)
 - n = 8
 - t = 5
 - nTreat = 0
 - p = 7
 - TimeRecon = 51
126   5     Execution @ (1:Gen1)
 - nTreat = 1
 - TimeExecution = 20
127   5     Context_verification @ (1:Gen1)
 - voice = (1,5)
 - Noise = true
128   5     vocal_context_test @ (1:Gen1)
 - Current = true
 - lim = 0
 - handicap = "Deaf"
 - a = 2
 - Active = false
129   5     Action_Ontology_test @ (1:Action)
 - Tmax = 50
 - Tmin = 0
 - Index = 1
 - Act = "Get "
 - E = 1
 - T = 5
130   5     Action_Found @ (1:Action)
 - Ta = 5
 - Act = "Get "
131   5     Action_type_test @ (1:Action)
 - Ta = 5
```

```
 - Act = "Get "
 - d = "Get "
 - c = "AFMO "
132   6     Generator @ (1:Gen1)
 - n = 9
 - NextTime = 1
133   6     Recognition @ (1:Gen1)
 - n = 9
 - t = 6
 - nTreat = 0
 - p = 8
 - TimeRecon = 48
134   6     Execution @ (1:Gen1)
 - nTreat = 1
 - TimeExecution = 20
135   6     Context_verification @ (1:Gen1)
 - voice = (1,6)
 - Noise = false
136   6     rejected_values @ (1:Gen1)
137   6     vocal_context_test @ (1:Gen1)
 - Current = false
 - lim = 0
 - handicap = "Deaf"
 - a = 4
 - Active = true
138   6     Generator @ (1:Gen2)
 - n = 13
 - NextTime = 1
139   6     Recognition @ (1:Gen2)
 - n = 13
 - t = 6
 - nTreat = 0
 - p = 12
 - TimeRecon = 52
140   6     Execution @ (1:Gen2)
 - nTreat = 1
 - TimeExecution = 24
141   6     Context_verification @ (1:Gen2)
 - movement = (1,6)
 - light = false
142   6     visual_context_test @ (1:Gen2)
 - Current = false
 - lim = 0
 - a = 9
 - handicap = "blind"
 - Active = true
143   6     rejected_values @ (1:Gen2)
144   7     Generator @ (1:Gen2)
 - n = 14
 - NextTime = 1
```

```
145   7     Recognition @ (1:Gen2)
 - n = 14
 - t = 7
 - nTreat = 0
 - p = 13
 - TimeRecon = 51
146   7     Execution @ (1:Gen2)
 - nTreat = 1
 - TimeExecution = 18
147   7     Context_verification @ (1:Gen2)
 - movement = (1,7)
 - light = true
148   7     Generator @ (1:Gen1)
 - n = 10
 - NextTime = 0
149   7     Generator @ (1:Gen1)
 - n = 11
 - NextTime = 0
150   7     Recognition @ (1:Gen1)
 - n = 11
 - t = 7
 - nTreat = 0
 - p = 9
 - TimeRecon = 56
151   7     Recognition @ (1:Gen1)
 - n = 10
 - t = 7
 - nTreat = 1
 - p = 10
 - TimeRecon = 52
152   7     Execution @ (1:Gen1)
 - nTreat = 2
 - TimeExecution = 21
153   7     Execution @ (1:Gen1)
 - nTreat = 1
 - TimeExecution = 19
154   7     Context_verification @ (1:Gen1)
 - voice = (1,7)
 - Noise = true
155   7     Context_verification @ (1:Gen1)
 - voice = (2,7)
 - Noise = true
156   7     Generator @ (1:Gen1)
 - n = 12
 - NextTime = 0
157   7     vocal_context_test @ (1:Gen1)
 - Current = true
 - lim = 0
```

```
 - handicap = "Normal"
 - a = 4
 - Active = false
158   7     Generator @ (1:Gen1)
 - n = 13
 - NextTime = 1
159   7     Recognition @ (1:Gen1)
 - n = 12
 - t = 7
 - nTreat = 0
 - p = 11
 - TimeRecon = 49
160   7     Recognition @ (1:Gen1)
 - n = 13
 - t = 7
 - nTreat = 1
 - p = 12
 - TimeRecon = 51
161   7     vocal_context_test @ (1:Gen1)
 - Current = false
 - lim = 0
 - handicap = "Deaf"
 - a = 5
 - Active = true
162   7     Person_Ontology_test @ (1:Person)
 - E = 2
 - T = 7
 - Tmax = 75
 - Tmin = 0
 - Per = "Father "
 - Index = 1
163   7     feedback @ (1:Person)
 - T = 7
164   7     Person_Ontology_test @ (1:Person)
 - E = 1
 - T = 7
 - Tmax = 75
 - Tmin = 0
 - Per = "Father "
 - Index = 1
165   7     Person_Found @ (1:Person)
 - Per = "Father "
 - Tc = 7
166   7     Person_type_test @ (1:Person)
 - Per = "Father "
 - Tc = 7
 - d = "Father "
 - c = "P "
167   7     Execution @ (1:Gen1)
 - nTreat = 2
```

```
 - TimeExecution = 22
168   7     Execution @ (1:Gen1)
 - nTreat = 1
 - TimeExecution = 21
169   7     Context_verification @ (1:Gen1)
 - voice = (1,7)
 - Noise = true
170   7     vocal_context_test @ (1:Gen1)
 - Current = true
 - lim = 0
 - handicap = "Deaf"
 - a = 2
 - Active = false
171   7     Context_verification @ (1:Gen1)
 - voice = (2,7)
 - Noise = false
172   7     vocal_context_test @ (1:Gen1)
 - Current = false
 - lim = 0
 - handicap = "Deaf"
 - a = 9
 - Active = true
173   7     rejected_values @ (1:Gen1)
174   7     Object_Ontology_test @ (1:Object)
 - Tmax = 75
 - Tmin = 0
 - Obj = "that "
 - Index = 1
 - E = 1
 - T = 7
 - E2 = 1
 - T2 = 7
175   7     Object_Found @ (1:Object)
 - Tb = 7
 - Obj = "that "
176   7     Object_type_test @ (1:Object)
 - Tb = 7
 - Obj = "that "
 - d = "that "
 - c = "IO "
177   7     AFMO_P_IO @ (1:Fusion)
 - Tb = 7
 - c2 = "IO "
 - Obj = "that "
 - Ta = 5
 - c1 = "AFMO "
 - Act = "Get "
 - Per = "Father "
```

```
 - Tc = 7
 - c3 = "P "
 - location = ("","",0)
178   7    Event_Collection @ (1:Fusion)
 - Td = 0
 - c4 = ""
 - Loc = ""
 - Per = "Father "
 - Tc = 7
 - c3 = "P "
 - Tb = 7
 - c2 = "IO "
 - Obj = "that "
 - Ta = 5
 - c1 = "AFMO "
 - Act = "Get "
 - Mod = "AFMO P IO "
179   7    AFMO_P_IO @ (1:Fusion)
 - Tb = 6
 - c2 = "IO "
 - Obj = "that "
 - Ta = 3
 - c1 = "AFMO "
 - Act = "Get "
 - Per = "Father "
 - Tc = 7
 - c3 = "P "
 - location = ("","",0)
180   7    Event_Collection @ (1:Fusion)
 - Td = 0
 - c4 = ""
 - Loc = ""
 - Per = "Father "
 - Tc = 7
 - c3 = "P "
 - Tb = 6
 - c2 = "IO "
 - Obj = "that "
 - Ta = 3
 - c1 = "AFMO "
 - Act = "Get "
 - Mod = "AFMO P IO "
181   7    Decision @ (1:Fusion)
 - model3 = ""
 - model2 = ""
 - model1 = ""
 - decision = ""
182   7    Decision @ (1:Fusion)
 - model3 = ""
 - model2 = ""
```

```
- model1 = ""
- decision = ""
```

ANNEXE III

Requêtes SQWRL et règles SWRL

ActionForMovableObject(?x) ∧ AverageObject(?y) ∧ IntendedLocation(?z) ∧ hasNextM02(?x, ?y) ∧ hasNextM02(?y, ?z) ∧ tbox:isDirectSuperClassOf(?m, ActionForMovableObject) ∧ tbox:isDirectSuperClassOf(?m, AverageObject) ∧ tbox:isDirectSuperClassOf(?m, IntendedLocation) → sqwrl:selectDistinct(?m)

ActionForMovableObject(?x) ∧ SmallObject(?y) ∧ IntendedLocation(?z) ∧ AverageObject(?l) ∧ hasNextM01(?x, ?y) ∧ hasNextM01(?y, ?z) ∧ hasNextM01(?z, ?l) ∧ tbox:isDirectSuperClassOf(?m, ActionForMovableObject) ∧ tbox:isDirectSuperClassOf(?m, SmallObject) ∧ tbox:isDirectSuperClassOf(?m, IntendedLocation) ∧ tbox:isDirectSuperClassOf(?m, AverageObject) → sqwrl:selectDistinct(?m)

ActionForMovableObject(?x) ∧ MovableObject(?y) ∧ hasNextM04(?x, ?y) ∧ tbox:isDirectSuperClassOf(?m, ActionForMovableObject) ∧ tbox:isDirectSuperClassOf(?m, MovableObject) → sqwrl:selectDistinct(?m)

ActionForMovableObject(?x) ∧ MovableObject(?y) ∧ Person(?z) ∧ hasNextM05(?x, ?y) ∧ hasNextM05(?y, ?z) ∧ tbox:isDirectSuperClassOf(?m, ActionForMovableObject) ∧ tbox:isDirectSuperClassOf(?m, MovableObject) ∧ tbox:isDirectSuperClassOf(?m, Person) → sqwrl:selectDistinct(?m)

ActionForMovableObject(?x) ∧ MovableObject(?y) ∧ IntendedLocation(?z) ∧ Location(?a) ∧ hasNextM06(?x, ?y) ∧ hasNextM06(?y, ?z) ∧ hasNextM06(?z, ?a) ∧ tbox:isDirectSuperClassOf(?m, ActionForMovableObject) ∧ tbox:isDirectSuperClassOf(?m, MovableObject) ∧ tbox:isDirectSuperClassOf(?m, IntendedLocation) ∧ tbox:isDirectSuperClassOf(?m, Location) → sqwrl:selectDistinct(?m)

ActionForMovableObject(?x) ∧ MovableObject(?y) ∧ IntendedLocation(?z) ∧ hasNextM07(?x, ?y) ∧ hasNextM07(?y, ?z) ∧ tbox:isDirectSuperClassOf(?m, ActionForMovableObject) ∧ tbox:isDirectSuperClassOf(?m, MovableObject) ∧ tbox:isDirectSuperClassOf(?m, IntendedLocation) → sqwrl:selectDistinct(?m)

ActionForMovableObject(?x) ∧ IntendedObject(?y) ∧ IntendedLocation(?z) ∧ hasNextM08(?x, ?y) ∧ hasNextM08(?y, ?z) ∧ tbox:isDirectSuperClassOf(?m, ActionForMovableObject) ∧ tbox:isDirectSuperClassOf(?m, IntendedObject) ∧ tbox:isDirectSuperClassOf(?m, IntendedLocation) → sqwrl:selectDistinct(?m)

ActionForMovableObject(?x) ∧ IntendedObject(?y) ∧ IntendedLocation(?z) ∧ MovableObject(?a) ∧ hasNextM09(?x, ?y) ∧ hasNextM09(?y, ?z) ∧ hasNextM09(?z, ?a) ∧ tbox:isDirectSuperClassOf(?m, ActionForMovableObject) ∧ tbox:isDirectSuperClassOf(?m, MovableObject) ∧ tbox:isDirectSuperClassOf(?m, IntendedObject) ∧ tbox:isDirectSuperClassOf(?m, IntendedLocation) → sqwrl:selectDistinct(?m)

ActionForMovableObject(?x) ∧ IntendedObject(?y) ∧ hasNextM10(?x, ?y) ∧ tbox:isDirectSuperClassOf(?m, ActionForMovableObject) ∧ tbox:isDirectSuperClassOf(?m, IntendedObject) → sqwrl:selectDistinct(?m)

ActionForMovableObject(?x) ∧ IntendedLocation(?z) ∧ MovableObject(?y) ∧ hasNextM11(?x, ?z) ∧ hasNextM11(?z, ?y) ∧ tbox:isDirectSuperClassOf(?m, ActionForMovableObject) ∧ tbox:isDirectSuperClassOf(?m, MovableObject) ∧ tbox:isDirectSuperClassOf(?m, IntendedLocation) → sqwrl:selectDistinct(?m)

ActionForPerson(?x) ∧ Person(?y) ∧ hasNextM12(?x, ?y) ∧ tbox:isDirectSuperClassOf(?m, ActionForPerson) ∧ tbox:isDirectSuperClassOf(?m, Person) → sqwrl:selectDistinct(?m)

ActionForMovableObject(?x) ∧ IntendedPerson(?y) ∧ IntendedObject(?z) ∧ hasNextM17(?x, ?y) ∧ hasNextM17(?y, ?z) → hasNextM17(?x, ?z)

ActionForPerson(?x) ∧ Person(?y) ∧ MovableObject(?z) ∧ hasNextM13(?x, ?y) ∧ hasNextM13(?y, ?z) ∧ tbox:isDirectSuperClassOf(?m, ActionForPerson) ∧ tbox:isDirectSuperClassOf(?m, Person) ∧ tbox:isDirectSuperClassOf(?m, MovableObject) → sqwrl:selectDistinct(?m)

tbox:isDirectSuperClassOf(?m, ActionForPerson) ∧ tbox:isDirectSuperClassOf(?m, Person) ∧ tbox:isDirectSuperClassOf(?m, IntendedLocation) ∧ tbox:isDirectSuperClassOf(?m, Location) → sqwrl:selectDistinct(?m)

ActionForPerson(?x) ∧ Person(?y) ∧ IntendedLocation(?z) ∧ hasNextM15(?x, ?y) ∧ hasNextM15(?y, ?z) ∧ tbox:isDirectSuperClassOf(?m, ActionForPerson) ∧ tbox:isDirectSuperClassOf(?m, Person) ∧ tbox:isDirectSuperClassOf(?m, IntendedLocation) → sqwrl:selectDistinct(?m)
ActionForPerson(?x) ∧ Person(?y) ∧ IntendedLocation(?z) ∧ hasNextM16(?x, ?y) ∧ hasNextM16(?y, ?z) ∧ tbox:isDirectSuperClassOf(?m, ActionForPerson) ∧ tbox:isDirectSuperClassOf(?m, Person) ∧ tbox:isDirectSuperClassOf(?m,

IntendedLocation) → sqwrl:selectDistinct(?m)

ActionForNonMovableObject(?x) ∧ IntendedObject(?y) ∧ hasNextM18(?x, ?y) ∧ tbox:isDirectSuperClassOf(?m, ActionForNonMovableObject) ∧ tbox:isDirectSuperClassOf(?m, IntendedObject) → sqwrl:selectDistinct(?m)

ActionForPerson(?x) ∧ Person(?y) ∧ IntendedObject(?z) ∧ hasNextM17(?x, ?y) ∧ hasNextM17(?y, ?z) ∧ tbox:isDirectSuperClassOf(?m, ActionForPerson) ∧ tbox:isDirectSuperClassOf(?m, Person) ∧ tbox:isDirectSuperClassOf(?m, IntendedObject) → sqwrl:selectDistinct(?m)

ActionForNonMovableObject(?x) ∧ Location(?y) ∧ hasNextM19(?x, ?y) ∧ tbox:isDirectSuperClassOf(?m, ActionForNonMovableObject) ∧ tbox:isDirectSuperClassOf(?m, Location) → sqwrl:selectDistinct(?m)

tbox:isDirectSuperClassOf(?m, ActionForNonMovableObject) ∧ tbox:isDirectSuperClassOf(?m, NonMovableObject) ∧ tbox:isInDomainOf(?m, hasNextM20) → sqwrl:selectDistinct(?m)

MovableObject(?x) ∧ hasNextM25(?x, ?y) ∧ hasNextM25(?y, ?z) → hasNextM25(?x, ?z)

ActionForNonMovableObject(?x) ∧ NonMovableObject(?y) ∧ IntendedLocation(?z) ∧ hasNextM21(?x, ?z) ∧ hasNextM21(?z, ?y) ∧ tbox:isDirectSuperClassOf(?m, ActionForNonMovableObject) ∧ tbox:isDirectSuperClassOf(?m, NonMovableObject) ∧ tbox:isDirectSuperClassOf(?m, IntendedLocation) → sqwrl:selectDistinct(?m)

ActionForNonMovableObject(?x) ∧ IntendedObject(?y) ∧ NonMovableObject(?z) ∧ hasNextM22(?x, ?y) ∧ hasNextM22(?y, ?z) ∧ tbox:isDirectSuperClassOf(?m, ActionForNonMovableObject) ∧ tbox:isDirectSuperClassOf(?m, IntendedObject) ∧ tbox:isDirectSuperClassOf(?m, NonMovableObject) → sqwrl:selectDistinct(?m)

ActionForNonMovableObject(?x) ∧ NonMovableObject(?y) ∧ Location(?z) ∧ hasNextM23(?x, ?y) ∧ hasNextM23(?y, ?z) ∧ tbox:isDirectSuperClassOf(?m, ActionForNonMovableObject) ∧ tbox:isDirectSuperClassOf(?m, NonMovableObject) ∧ tbox:isDirectSuperClassOf(?m, Location) → sqwrl:selectDistinct(?m)

ActionForLocation(?x) ∧ MovableObject(?y) ∧ hasNextM24(?x, ?y) ∧

tbox:isDirectSuperClassOf(?m, ActionForLocation) ∧ tbox:isDirectSuperClassOf(?m, MovableObject) ∧ tbox:isInDomainOf(?m, hasNextM24) → sqwrl:selectDistinct(?m)

ActionForLocation(?x) ∧ MovableObject(?y) ∧ IntendedLocation(?z) ∧ Location(?a) ∧ hasNextM25(?x, ?y) ∧ hasNextM25(?y, ?z) ∧ hasNextM25(?z, ?a) ∧ tbox:isDirectSuperClassOf(?m, ActionForLocation) ∧ tbox:isDirectSuperClassOf(?m, MovableObject) ∧ tbox:isDirectSuperClassOf(?m, IntendedLocation) ∧ tbox:isDirectSuperClassOf(?m, Location) → sqwrl:selectDistinct(?m)

ActionForLocation(?x) ∧ IntendedLocation(?y) ∧ hasNextM27(?x, ?y) ∧ tbox:isDirectSuperClassOf(?m, ActionForLocation) ∧ tbox:isDirectSuperClassOf(?m, IntendedLocation) → sqwrl:selectDistinct(?m)

ActionForLocation(?x) ∧ MovableObject(?z) ∧ IntendedLocation(?y) ∧ hasNextM28(?x, ?y) ∧ hasNextM28(?y, ?z) ∧ tbox:isDirectSuperClassOf(?m, ActionForLocation) ∧ tbox:isDirectSuperClassOf(?m, MovableObject) ∧ tbox:isDirectSuperClassOf(?m, IntendedLocation) → sqwrl:selectDistinct(?m)

ActionForLocation(?x) ∧ NonMovableObject(?z) ∧ IntendedLocation(?y) ∧ hasNextM29(?x, ?y) ∧ hasNextM29(?y, ?z) ∧ tbox:isDirectSuperClassOf(?m, ActionForLocation) ∧ tbox:isDirectSuperClassOf(?m, NonMovableObject) ∧ tbox:isDirectSuperClassOf(?m, IntendedLocation) → sqwrl:selectDistinct(?m)

ActionForMovableObject(?x) ∧ Person(?y) ∧ ObjectForLiquid(?z) ∧ Liquid(?a) ∧ hasNextM30(?x, ?y) ∧ hasNextM30(?y, ?z) ∧ hasNextM30(?z, ?a) ∧ tbox:isDirectSuperClassOf(?m, ActionForMovableObject) ∧ tbox:isDirectSuperClassOf(?m, ObjectForLiquid) ∧ tbox:isDirectSuperClassOf(?m, Liquid) ∧ tbox:isDirectSuperClassOf(?m, Person) → sqwrl:selectDistinct(?m)

Modality(?m) ∧ Handicap(?h) ∧ hasUserContext(?m, ?h) ∧ hasHandicap(?h, ?han) ∧ swrlb:equal(?han, 1) → sqwrl:selectDistinct(?m)

SmallObject(?x) ∧ IntendedLocation(?y) ∧ Location(?z) ∧ hasNextM06(?x, ?y) ∧ hasNextM06(?y, ?z) → hasNextM06(?x, ?z)
Modality(?m) ∧ Light(?l) ∧ hasLight(?m, ?l) ∧ hasLightnessLevel(?l, ?level) ∧ swrlb:lessThan(?level, 4) → sqwrl:selectDistinct(?m)

Modality(?m) ∧ LocationContext(?loc) ∧ hasLocationContext(?m, ?loc) ∧ isAtRoad(?loc,

?loca) ∧ swrlb:equal(?loca, 1) → sqwrl:selectDistinct(?m)

Modality(?m) ∧ Noise(?l) ∧ hasNoise(?m, ?l) ∧ hasNoiseLevel(?l, ?level) ∧ swrlb:greaterThan(?level, 6) → sqwrl:selectDistinct(?m)

Liquid(?y) ∧ SmallObject(?x) ∧ hasLiquid(?x, ?y) → hasNextM30(?x, ?y)

Person(?x) ∧ IntendedLocation(?y) ∧ Location(?z) ∧ hasNextM14(?x, ?y) ∧ hasNextM14(?y, ?z) → hasNextM14(?x, ?z)

MovableObject(?x) ∧ IntendedLocation(?y) ∧ Location(?z) ∧ hasNextM25(?x, ?y) ∧ hasNextM25(?y, ?z) → hasNextM25(?x, ?z)

ActionForLocation(?x) ∧ IntendedLocation(?y) ∧ MovableObject(?z) ∧ hasNextM28(?x, ?y) → hasNextM28(?x, ?z)

ActionForLocation(?x) ∧ IntendedLocation(?y) ∧ NonMovableObject(?z) ∧ hasNextM29(?x, ?y) → hasNextM29(?x, ?z)

BIBLIOGRAPHIE

Babar, M. A., et I. Gorton. (2004). Comparison of Scenario-Based Software Architecture Evaluation Methods In Asia-Pacific software engineering conference (APSEC, 2004). Busan , south Corea**:** p. 600-607.

Barboni E., C. S., Navarre D., Palanque P. (2006). Model-Based Engineering of Widgets, User Applications and Servers Compliant with ARINC 661 Specification. Design Specification and Verification of Interactive Systems (DSV-IS 2006), LNCS, pp. 25-38.

Bellik, Y. (1997). Media integration in multimodal interfaces. Proceedings of IEEE Workshop on Multimedia Signal Processing, Princeton, NJ.

Bernsen, N. (1994). Modality Theory in support of multimoda interface design. Proc. of Intelligent Multi-Media Multi-Modal Systems (1994), 37-44.

Bolt, R. A. (1980). "Put-that-there": Voice and gesture at the graphics interface ACMSIGGRAPH Computer Graphics, v.14 n.3, p.262-270, July 1980.

Bouchet, J., Nigay, L. (2004). ICARE: A Component-Based Approach for the Design and Development of Multimodal Interfaces. In Extended Abstracts CHI'04 (2004). ACM Press, 1325-1328.

Bouchet, J., Nigay, L., and Ganille, T. (2004). ICARE software components for rapidly developing multimodal interfaces. InProceedings of the 6th international Conference on Multimodal interfaces (State College, PA, USA, October 13 - 15, 2004). ICMI '04.ACM, New York, NY, 251-258.

Bourguet, M. L. (2002). A Toolkit for Creating and Testing Multimodal Interface Designs. Proceedings of UIST'02, , Paris. 2002, pp. 29--30. .

Bourguet, M. L. (2002). A Toolkit for Creating and Testing Multimodal Interface Designs. Proceedings of UIST'02, Paris. 2002, pp. 29--30.

Brézillon, P., et J. C. Pomerol. (1999). Contextual knowledge sharing and cooperation in intelligent assistant systems. Le Travail Humain, vol. 62, no 3, p. 223-246.

Brisson, L. T. (2004). Mesures d'intérêt subjectif et représentation des connaissances. Report ISRN I3S/RR-2005-35-FR, Université de Nice.

Brown, P. J. (1995). The stick-e document: a framework for creating context-aware

applications. Electronic Publishing Origination, Dissemination and Design, vol. 8, no 2 (JUNE & SEPTEMBER 1995), p. 259-272.

Brown, P. J., J. D. Bovey et X. Chen. (1997). Context-aware applications: from the laboratory to the marketplace. IEEE Personal Communications, vol. 4, no 5 (October 1997), p. 58-64.

Bruno Dumas, D. L., Rolf Ingold (2008). Démonstration : HephaisTK, une boîte à outils pour le prototypage d!interfaces multimodales.

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. Data Min. Knowl. Discov. 2(2), 121–167.

Calvanese, D., G. D. Giacomo, et al. (2001). Reasoning in expressive description logics. Handbook of automated reasoning, Elsevier Science Publishers B. V.**:** 1581-1634.

Calvanese D., J. C., Giacomo GD, J.Hendler, I.Berman, B. Parsia, P.F. Patel-Schneider, A.Ruttenberg, U.Sattler and M.Schneider (2009). OWL2 Web ontology Languages Profiles.

Carlos Duarte, L. s. C. (2006). A conceptual framework for developing adaptive multimodal applications. Proceedings of the 11th international conference on Intelligent user interfaces. Sydney, Australia, ACM**:** 132-139.

Chen, G., et D. Kotz. (2000). A Survey of Context-Aware Mobile Computing Research. TR2000-381. Dartmouth: Dartmouth College Computer Scienc.

Choudhury, T., Rehg, J.M., Pavlovic, V., Pentland, A. (2002). Boosting and structure learning in dynamic bayesian networks for audiovisual speaker detection. In: The 16th International Conference on Pattern Recognition, vol. 3, pp. 789–794. Quebec.

Clay, A. (2009). La branche émotion, un modèle conceptuel pour l'intégration de la reconnaissance multimodale d'émotions dans des applications interactives : application au mouvement et à la danse augmentée. PhD Thesis, ESTIA, Université Bordeaux 1. 2009. 204 pages.

Cohen, P. R., Johnston, M., McGee, D., Oviatt, S., Pittman, J., Smith, I., Chen, L., and Clow, J. (1997). QuickSet: multimodal interaction for distributed applications. In Proceedings of the Fifth ACM international Conference on Multimedia. MULTIMEDIA '97. ACM, New York, NY, 31-40.

Coutaz, J., Caelen, J. (1991). A taxonomy for multimedia and multimodal user interfaces. Proceedings of the 1st ERCIM Workshop on Multimodal HCI, pp. 143-148, Lisbon: INESC.

Cuenca Grau, B. I. H., Boris Motik, Bijan Parsia, Peter Patel-Schneider, and Ulrike Sattler (2008). OWL 2: The next step for OWL. Journal of Web Semantics **6**(4).

D L McGuinness, F. V. H. (2004). OWL Web Ontology Language Overview. http://www.w3.org/TR/owl-features/.

Dey, A. K. (2001). Understanding and using context. Personal and ubiquitous computing, vol. 5, p. 4-7.

Dmitry Tsarkov , I. H. (2006). FaCT++ description logic reasoner: System description n Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006).

Duarte, C. a. C., L. (2006). A conceptual framework for developing adaptive multimodal applications. In Proceedings of the 11th international Conference on intelligent User interfaces (Sydney, Australia, January 29 - February 01, 2006). IUI '06. ACM, New York, NY, 132-139.

Duarte, L. e. a. J., M. and Bangalore (2005). Finite-state multimodal integration and understanding. Nat. Lang. Eng. 11, 2 (Jun. 2005), 159-187.

Dumas, B., D. Lalanne, et al. (2009). HephaisTK: a toolkit for rapid prototyping of multimodal interfaces. Proceedings of the 2009 international conference on Multimodal interfaces. Cambridge, Massachusetts, USA, ACM**:** 231-232.

Dumas, B., Lalanne, D., Guinard, D., Koenig, R., and Ingold, R. (2008). Strengths and weaknesses of software architectures for the rapid creation of tangible and multimodal interfaces. In Proc. of the 2nd int. Conf. 159on Tangible and Embedded interaction (Bonn, Germany, 2008). TEI '08. ACM, 47-54.

Dumas B., L. D. O. S. (2009). Multimodal Interfaces: A Survey of Principles, Models and Frameworks. D. Lalanne and J. Kohlas (Eds.): Human Machine Interaction, LNCS 5440, pp. 3–26, 2009. Springer-Verlag Berlin 2009.

Engel, R. P., Norbert. (2006). Modality Fusion.SmartKom: Foundations of Multimodal Dialogue Systems. Springer, Berlin.

Evren Sirin, B. P., Bernardo Cuenca Grau , Aditya Kalyanpur , Yarden Katz (2007). Pellet: A Practical OWL-DL Reasoner. Web Semantics: Science, Services and Agents on the World Wide Web **5**(2): 51-53.

Flippo, F., Krebs, A., and Marsic, I. (2003). A framework for rapid development of multimodal interfaces. In Proceedings of the 5th international Conference on Multimodal interfaces (Vancouver, British Columbia, Canada, November 05 - 07, 2003). ICMI '03. ACM, New York, NY, 109-116.

Gennari, J. H., Musen, M. A., Fergerson, R. W., Grosso, W. E., Crubézy, M., Eriksson H., Noy N. F. et Tu S. W. (2002). The Evolution of Protégé: An Environment for Knowledge-Based Systems Development. T. Report. Stanford, CA.

Gomez Perez A., V. R. B. (1999). Applications of Ontologies and Problem-Solving Methods. AI-Magazin Vol 20. n°1. AAAI Press. pp 119-122. ISSN 0738-4602.

Gruber, T. R. (1993). Towards Principles for the Design of Ontologies Used for Knowledge Sharing in Formal Ontology in Conceptual Analysis and Knowledge Representation. Kluwer Academic Publishers.

Guarino, N. (1998). Formal ontology and information systems. In N.Guarino (Ed.),Proceedings of the 1st International conference. T. N. IOS Press Amsterdam, , ISBN: 9051993994 Trento, Italy**:** p. 337.

Guha, B. a. (2004). RDF Vocabulary Description Language 1.0: RDF Schema. World Wide Web Consortium, Recommendation REC-rdf-schema.

Gurevych, I., Merten, S., and Porzel, R. (2003). Automatic creation of interface specifications from ontologies. In: Proceedings of the HLT-NAACL Workshop on Software Engineering and Architecture of Language Technology Systems (SEALTS), Edmonton, Canada.

Hall, D. L., Llinas, J. (1997). An introduction to multisensor fusion. In: Proceedings of the IEEE: Special Issues on Data Fusion, vol. 85, no. 1, pp. 6–23.

Heckmann, D. (2005). Ubiquitous User Modeling. Computer Science Department. Germany, Saarland University.

Henricksen, K., J. Indulska et A. Rakotonirainy. (2002). Modeling Context Information in Pervasive Computing Systems. In Proc. of the First International Conference on

Pervasive Computing, Pervasive'2002 (August 2002). Vol. 2414, p. 167-180. Zurich, Switzerland: Lecture Notes in Computer Science, Springer Verlag, LNCS.

Herzog, B., Kirchmann, H., Merten, S., Ndiaye, S., Poller, and a. B. P., T. (2003). MULTIPLATFORM Testbed: An Integration Platform for Multimodal Dialog Systems. In: Proceedings of the HLT-NAACL Workshop on Software Engineering and Architecture of Language Technology Systems (SEALTS), Edmonton, Canada.

Holzapfel, H., Nickel, K., and Stiefelhagen, R. (2004). Implementation and evaluation of a constraint-based multimodal fusion system for speech and 3D pointing gestures. In

Proceedings of the 6th international Conference on Multimodal interfaces, State College, PA, USA., ICMI '04. ACM, New York, NY, 175-182.

Holzapfel, H., Nickel, K., and Stiefelhagen, R. I. (2004). mplementation and evaluation of a constraint-based multimodal fusion system for speech and 3D pointing gestures. In Proceedings of the 6th international Conference on Multimodal interfaces (State College, PA, USA. ICMI '04. ACM, New York, NY, 175-182.

http://www.ontoknowledge.org/index.shtml (2003). OIL. Ontology Inference Layer. (Last access:10. August 2003).

Ipek Ozkaya, R. K., & Mark Klein (Mai 2007). Quality-Attribute-Based Economic Valuation of Architectural Patterns. SEI, CMU/SEI-2007-TR-003.

Jaimes, A., Sebe, (2007). Multimodal human-computer interaction: A survey. In Computer Vision and Image Understanding 108, 1-2 (Oct. 2007), Elsevier, pp.116-134 (2007).

Jensen, K. (1998). A Brief Introduction to Coloured Petri Nets. In Workshop on the Applicability of Formal Models (June 1998). p. 55-58. Aarhus, Denmark.

JEONG Gu-Beom , K. G.-B. (mai 2006). A Study on Software Architecture Evaluation. Congrès Computational science and its applications, .

Jobs, S. P. e. a. (2008). Touch Screen Device, Method, and Graphical User Interface for Determining Commands by Applying Heuristics. United States Patent Application 20080122796. Kind Code A, May 29, 2008.

Johnston (2009). Building Multimodal Applications with EMMA Proceedings of the 2009 international conference on Multimodal interfaces - ICMI-MLMI 09 Pages: 47-54.

Johnston, M., Cohen, P. R., McGee, D., Oviatt, S. L., Pittman, J. A., and Smith, I. (1997). Unificationbased multimodal integration. In Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics, (Morristown, NJ, USA, 1997).

Johnston, M. a. B. (2005). S. Finite-state multimodal integration and understanding. Nat. Lang. Eng. 11, 2 (Jun. 2005), 159-187.

KG, R. S. G. C. (2007). RacerPro User's Guide, http://www.racer-systems.com/products/racerpro/users-guide-1-9-2-beta.pdf.

Kittler, J., Hatef, M., Duin, R.P., Matas, J.: On combining (1998). classifiers. IEEE Trans. Pattern Anal. Mach. Intell. 20(3), 226–239.

Koons, D. B., Sparrell, C. J., and Thorisson, K. R. (1993). Integrating simultaneous input from speech, gaze, and hand gestures. In intelligent Multimedia interfaces, M. T. Maybury, Ed. American Association for Artificial Intelligence, Menlo Park, CA, 257-276.

Krahnstoever, N., Kettebekov, S., Yeasin, M., and Sharma, R. (2002). A Real-Time Framework for Natural Multimodal Interaction with Large Screen Displays. International Conference on Multimodal Interfaces. IEEE Computer Society, Washington, DC.

Kristensen, K. J. a. L. M. (2009). Coloured Petri Nets – Modelling and Validation of Concurrent Systems.

Lalanne, D., L. Nigay, et al. (2009). Fusion engines for multimodal input: a survey. Proceedings of the 2009 international conference on Multimodal interfaces. Cambridge, Massachusetts, USA, ACM: 153-160.

Lalanne, D., L. Nigay, et al. (2009). Fusion engines for multimodal input: a survey. Proceedings of the 2009 international conference on Multimodal interfaces. Cambridge, Massachusetts, USA, ACM**:** 153-160.

Lassila, O. (2000). The Resource Description Framework. IEEE Intelligent Systems **15**(6).

Latoschik, M. E. (2002). Designing transition networks for multimodal VR-interactions using a markup language Multimodal Interfaces. In Proceedings of the Fourth IEEE International Conference on Multimodal Interfaces, IEEE, 411-416. 2002.

Latoschik, M. E. (2002). Designing transition networks for multimodal VR-interactions using a markup language. Multimodal Interfaces, 2002. In Proceedings of the Fourth IEEE International Conference on Multimodal Interfaces. IEEE, 411-416.

Lenat, D. B. (1990). Building Large Knowledge-Based Systems: Representation and Inference of the CYC Project, Addison-Wesley.

M. Johnston, P. B., D. C. Burnett, J. Carter, D. A. Dahl, G. McCobb, and D. Raggett. (2009). EMMA: Extensible MultiModal Annotation markup language, W3C Recommendation.

Magalhaes, J., Ruger, S. (2007). Information-theoretic semantic multimedia indexing. In: International Conference on Image and Video Retrieval, pp. 619–626. Amsterdam, The Netherlands.

Manola, F. M., Eric. (2004). RDF Primer. World Wide Web Consortium, Recommendation REC-rdf-primer.

Mansoux, B., Nigay, L., Troccaz, J. (2006). Output Multimodal Interaction: The Case of Augmented Surgery. In Proceedings of HCI'06, Human Computer Interaction, People and Computers XX, the 20th BCS HCI Group conference. (London, UK,11-15). Springer Publ, 177-192.

Mansoux, B., Nigay, L., Troccaz, J. (2006). Output Multimodal Interaction: The Case of Augmented Surgery. In Proceedings of HCI'06, Human Computer Interaction, People and Computers XX, the 20th BCS HCI Group conference, (London, UK,11-15 September, 2006). Springer Publ, 177-192.

Martin, J. C. (1998). TYCOON: Theoretical Framework and Software Tools for Multimodal Interfaces. Intelligence andMultimodality in Multimedia interfaces. (ed.) John Lee, AAAI Press.

Martin O'Connor, A. D. (2009). SQWRL: a Query Language for OWL, Stanford Center for Biomedical Informatics Research, Stanford, CA 94305 martin.oconnor@stanford.edu.

Mattsson, M., H. Grahn et F. Mårtensson. (2006). Software Architecture Evaluation Methods for Performance, Maintainability, Testability, and Portability. In Second International Conference on the Quality of Software Architectures (QoSA 2006). Västerås, Sweden. .

McNeill, D. (1992). Hand and Mind: What Gestures Reveal About Thought, Univ. of Chicago Press, Chicago, IL (1992).

Melichar, M. a. C., P. (2006). From vocal to multimodal dialogue management. In Proceedings of the 8th international Conference on Multimodal interfaces, ICMI '06. ACM, New York, 59-67.

Melichar, M. a. C., P. (2006). From vocal to multimodal dialogue management. In Proceedings of the 8th international Conference on Multimodal interfaces 2006. ICMI '06. ACM, New York, 59-67.

Milota, A. D. (2004). Modality fusion for graphic design applications. In Proceedings of the 6th international Conference on Multimodal interfaces ICMI '04. ACM, New York, NY, 167-174.

Milota, A. D. (2004). Modality fusion for graphic design applications. In Proceedings of the 6th international Conference on Multimodal interfaces. ICMI '04. ACM, New York, NY, 167-174.

Minsky, M. (1974). A Framework for Representing Knowledge. Technical Report. UMI Order Number: AIM-306., Massachusetts Institute of Technology.

Navarre, D., Palanque, P., Basnyat, S. (2008). Usability Service Continuation through Reconfiguration of Input and Output Devices in Safety Critical Interactive Systems. The 27th International Conference on Computer Safety, Reliability and Security (SAFECOMP 2008), LNCS 5219, pp. 373–386, 2008.

Navarre, D., Palanque, P., Bastide, R., Schyn, A., Winckler, M.A., Nedel, L., Freitas, C. (2005). A Formal Description of Multimodal Interaction Techniques for Immersive Virtual Reality Applications. In: INTERACT 2005: IFIP TC13 International Conference. Rome, Italy, Springer-Verlag LNCS 3585, p. 170-185.

Neal, J. G., Thielman, C. Y., Dobes, Z., Haller, S. M., and Shapiro, S.C. (1989). Natural language with integrated deictic and graphic gestures. In Proceedings of the 1989 Workshop on Speech and Natural Language. Human Language Technology Conference. Association for Computational Linguistics, Morristown, NJ, 410-423.

Nigay, L., Coutaz, J., Salber, D. MATIS (1993). A multimodal airline travel information system. E. B. A. SM/WP10.

Nigay, L. a. C., J. (1993). A design space for multimodal systems: concurrent processing and data fusion. In Proceedings of the INTERCHI '93 Conference on Human Factors in Computing Systems. S. Ashlund, A. Henderson, E. Hollnagel, K. Mullet, and T. White, Eds. IOS Press, Amsterdam, 172-178.

Nigay, L. a. C., J. 5. (1995). A generic platform for addressing the multimodal challenge. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (Denver, Colorado, United States, May 07 - 11, 1995). Conference on Human Factors in Computing Systems. ACM Press, New York, NY, 98-10.

Obrst, L. (2003). Ontologies for semantically Interoperable Systems. Conference on information ans knowledge Management. Proceedings of the twelfth international conference on Information and knowledge management. New Orleans. LA, USA, ACM Press, New York, NY, USA. ISBN: 1-58113-723-0**:** Pp 366-369.

OCL (2010). Object Constraint Language formal v2.2, Object modeling Group, http://www.omg.org/spec/OCL/2.2.

ODM (2008). Ontology Definition Metamodel. Object modeling Group.

Pan, H., Liang, Z.P., Anastasio, T.J., Huang, T.S. (1999). Exploiting the dependencies in information fusion. In CVPR, vol. 2, pp. 407-412 (1999).

Panton, K., Cynthia Matuszek, Douglas Lenat, Dave Schneider, Michael Witbrock, Nick Siegel, and Blake Shepard (2006). Common Sense Reasoning – From Cyc to Intelligent Assistant In Y. Cai and J. Abascal (Eds.): Ambient Intelligence in

Everyday Life. LNAI 3864, pp. 1 – 31, 2006, Springer-Verlag Berlin Heidelberg 2006.

Pareiras F.Silva, T. W., S.Staab, C.Saathoff, T.Franz (2009). APIs agogo: Automatic Generation of Ontology APIs. In Proceedings of the , 2nd IEEE International Conference on Semantic Computing. Santa Clara.

Parsia, B., Sirin E, Grau BC, Ruckhaus E, Hewlett D (2005). Cautiously Approaching SWRL. Science.

Pascoe, J. (1997). The Stick-e Note Architecture: Extending the Interface Beyond the User. In International Conference on Intelligent User Interfaces. p. 261-264.

Pfleger, N. (2004). Context based multimodal fusion. In: ACM International Conference on Multimodal Interfaces, pp. 265–272. State College.

Pfleger, N. (2004). Context based multimodal fusion. In Proceedings of the 6th international Conference on Multimodal interfaces (State College,PA, USA, October 13 - 15, 2004). ICMI '04. ACM, New York,NY, 265-272.

Portillo, P. M., García, G. P., and Carredano, G. A. (2006). Multimodal fusion: a new hybrid strategy for dialogue systems. In Proceedings of the 8th international Conference on Multimodal interfaces. (Banff, Alberta, Canada), ICMI '06. ACM, New York, NY, 357-363. .

Portillo, P. M., García, G. P., and Carredano, G. A.. (2006). Multimodal fusion: a new hybrid strategy for dialogue systems. In Proceedings of the 8th international Conference on Multimodal interfaces (Banff, Alberta, Canada, November 02 - 04, 2006). ICMI '06. ACM, New York, NY, 357-363.

R. Kazman, L. B., M. Klein, T. Lattanze, L. Northrop (2005). A Basis for Evaluating Software Architecture Analysis Methods. Software Quality Journal, 13, 2005, 329-335,.

Reisig, W. (1985). Petri nets: An Introduction. In EATCS Monographs on Theoretical Computer Science. Vol. 4. Springer-Verlag.

Ryan, N., J. Pascoe et D. Morse. (1997). Enhanced Reality Fieldwork: The Context-aware Archaeological Assistant. In Computer Applications in Archaeology. Oxford, UK.

Schilit, B. N., N. Adams et R. Want. (1994). Context-Aware Computing Applications. In Proceedings of the Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, (Decenber 1994). p. 85-90. Santa Cruz, CA, : IEEE Computer Society.

Schlomer, T., Poppinga B., Henze N. & Boll S. (2008). Gesture Recognition with a Wii Controller, Proceedings of the 2nd international conference on Tangible and embedded interaction, ACM, 2008.

Schmidt, A., K. Asante Aidoo, A. Takaluoma, U. Tuomela, K. Van Laerhoven et W. Van de Velde. (1999). Advanced Interaction in Context. In Handheld and ubiquitous computing. International symposium No1 (27-29 September 1999). Vol. 1707, p. 89-101. Karlsruhe , ALLEMAGNE Lecture notes in computer science.

Shikler, T. S., Kaliouby, R. and Robinson, P. (2004). Design Challenges in Multi-modal Inference Systems for Human-Computer Interaction. In Proceedings of International Workshop on Universal Access and Assistive Technology, Fitzwilliam College, University of Cambridge, United Kingdom, 22nd-24th March, 2004.

Sowa, J. (1984). Conceptual Structures : Information Processing. Mind and Machine. The Systems Programming Series, Addison-Wesley.

Sowa, J. F. (1999). Knowledge Representation : Logical, Philosophical, and Computational Foundations, Brooks Cole Publishing Co Pacific Grove, CA, USA, p. 594, ISBN: 0-534-94965-7.

Stuckenschmidt, H., Van Harmelen, Frank. (2005). Information sharing on the semantic web. Springer.

Sun, Y., Chen, F., Shi, Y., and Chung, V. (2006). A novel method for multi-sensory data fusion in multimodal human computer interaction. In Proceedings of the 18th Australia Conference on Computer-Human interaction. OZCHI '06, vol. 206. ACM, 401-404.

SWRL (2004). http://www.w3.org/Submission/SWRL/.

Theimer., S. e. (1994). Disseminating Active Map Information to Mobile Hosts. IEEE Network, 8(5), (September/October 1994), p. 22-32.

Toby Segaran, C. E., Jamie Taylor (2009). Programming the Semantic Web Build Flexible Applications with Graph Data, O'Reilly Media.

Tu, S. W., Kahn, M. G. Musen, M. A. (1989). Episodic skeletal-plan refinement based on temporal data. Communications of the ACM, 32(12): 1439-1455.

University of Aarhus, D. (2012). CPN-TOOLS version 2.2.0. http://wiki.daimi.au.dk/cpntools/cpntools.wiki: (site consulté en janvier 2012).

Vo, M. T. a. W., C. (1996). Building an Application Framework for Speech and Pen Input Integration in Multimodal Learning Interfaces. In IEEE Conf. on Acoustics, Speech, and Signal Processing; Proceedings of ICASSP'96, (Atlanta, USA, May 1996). .

Wahlster, W. (1991). User and discourse models for multimodal communication. In intelligent User Interfaces, J. W. Sullivan and S. W. Tyler, Eds. ACM, New York, NY, 45-67.

Wang, J., Kankanhalli, M.S., Yan, W.Q., Jain, R. (2003). Experiential sampling for video surveillance. In: ACM Workshop on Video Surveillance. Berkeley.

Ward, A. J., et A. Hopper. (1997). A New Location Technique for the Active Office. IEEE Personnal Communications, vol. 4, no 5 (October 1997), p. 42-47.

Wu, H. (2003). Sensor data fusion for context-aware computing using dempster–shafer theory. Ph.D. thesis, The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Wu, L., Oviatt, S. L., and Cohen, P. R. (2002). From members to teams to committee-a robust approach to gestural and multimodal recognition. In IEEE Trans Neural Netw, vol. 13, no. 4, 2002. : pp. 972-982.

Wu, L., Oviatt, S., and Cohen, P. R. (1999). Multimodal Integration - A Statistical View. In IEEE Transactions on Multimedia, vol. 1, no. 4, 1999. pp. 334-341.

www.loa-cnr.it/DOLCE.html.

www.ontologyportal.org.