ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

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

COMME EXIGENCE À L'OBTENTION DU
DOCTORAT EN GÉNIE DE L'ÉTS
ET DU
DOCTORAT EN INFORMATIQUE DE L'UNIVERSITÉ DE VERSAILLES
SAINT-QUENTIN EN YVELINES EN FRANCE

PAR
HICHAM DJENIDI

ARCHITECTURES LOGICIELLES DYNAMIQUES DÉDIÉES AUX APPLICATIONS MULTIMODALES

MONTRÉAL, LE 25 JUILLET 2007

CETTE THÈSE A ÉTÉ ÉVALUÉE
PAR UN JURY COMPOSÉ DE :

Madame Rita Noumeir, présidente du jury,
Professeure à l'École de technologie supérieure, Département de Génie Électrique

Madame Isabelle Borne, membre du jury et rapporteur,
Professeure à l'Université de Bretagne Sud, France

Madame Sylvie Ratté, membre du jury et rapporteur,
Professeure à l'École de technologie supérieure, Département de Génie Logiciel et des TI

Monsieur Amar Ramdane-Chérif, membre du jury,
Maître de conférence à l'Université de Versailles Saint-Quentin en Yvelines, Laboratoire PRISM

Madame Nicole Levy, directrice de thèse,
Professeure à l'Université de Versailles Saint-Quentin en Yvelines, Laboratoire PRISM

Monsieur Chakib Tadj, directeur de thèse,
Professeur à l'École de technologie supérieure, Département de Génie Électrique

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

LE 28 AOÛT 2007.

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

ARCHITECTURES LOGICIELLES DYNAMIQUES DÉDIÉES AUX APPLICATIONS MULTIMODALES

Hicham Djenidi

SOMMAIRE

Les systèmes multimédia multimodaux combinent des modes naturels en entrée, comme la parole, le toucher, le geste, etc. et, génèrent une ou plusieurs commandes, envoyée(s) sur un ou plusieurs systèmes de présentation en sortie. L'objectif principal de nos travaux de recherche est double. En premier lieu, nous proposons une architecture logicielle (AL) nouvelle qui rassemble les propriétés récurrentes implicitement employées dans les dialogues multimodaux. Cette AL est modélisée, spécifiée formellement et raffinée par l'emploi de réseaux de Petri colorés temporisés stochastiques. Elle réalise un moteur de fusion fission multimodal. En second lieu, nous fournissons une méthodologie basée sur le concept de scénario permettant la reconfiguration architecturale du moteur de fusion/fission multimodal tout en garantissant un profil de qualité choisi. Ces deux propositions constituent une méthodologie qui a pour finalité un comportement intelligent de ce type d'application. En effet, elles permettent au concepteur de systèmes multimodaux de réaliser deux propriétés fondamentales : l'accès universel et la mobilité. L'accès universel constitue la première et majeure motivation du développement d'interfaces multimodales plus flexibles. En effet, la multimodalité permet à une plus vaste population (plus diversifiée et non spécialisée) d'utilisateurs d'employer les systèmes informatiques. Un autre avantage majeur est la possibilité d'étendre le contexte d'utilisation viable d'une application pour y inclure des modifications dynamiques des paramètres d'utilisation et des traitements dans des cas de mobilité. Nous proposons donc une approche générique qui répond aux carences des solutions architecturales traitées dans la littérature : l'architecture est dédiée à tout type de média, elle permet la fusion/fission multimodale à tous les niveaux d'abstractions (niveaux sémantique, syntaxique, etc.…) L'architecture proposée est multiagent. Elle est 'reconfigurable' dynamiquement et sa reconfiguration est réalisée pour maintenir des caractéristiques d'un profil de qualité.
Dans un but de validation de notre approche, la méthodologie proposée a été appliquée pour le développement de prototypes.

DYNAMIC SOFTWARE ARCHITECTURES FOR MULTIMODAL APPLICATIONS

Hicham Djenidi

ABSTRACT

With the growth in technology, many applications supporting more transparent and flexible human-computer interactions have emerged. This has resulted in an increasing need for more powerful communication protocols, especially when several media are involved. Multimedia multimodal applications are systems combining two or more natural input modes, such as speech, touch, manual gestures, lip movements, etc. Thus, a comprehensive command or a meta-message is generated by the system and sent to multimedia output devices. Each application is based on a software architecture combining modalities to match and elaborate on the relevant multimodal information. Such applications remain strictly based on previous results, however, and there is limited synergy among parallel ongoing efforts. Today, for example, there is no agreement on the software architectures that support a dialog implementation, independently of the application type. The main objective of this thesis is twofold.

First, we propose new architectural paradigms for analyzing and extracting the collective and recurrent properties implicitly used in such dialogs. These paradigms use the agent architecture concept to achieve their functionalities and unify them into software architecture modeled, specified and refined by stochastic timed colored Petri nets.

Second, we propose a new methodology to show the ways in which agents can be introduced at the architectural level and how such agents improve some quality attributes by adapting the initial architecture via multimodal scenarios.

Examples of multimodal prototypes are sketched to illustrate the proposals.

PRÉFACE

Ce mémoire de doctorat a été réalisé dans le cadre d'une thèse de cotutelle entre le Laboratoire des Architectures du Traitement de l'Information et du Signal (LATIS) de l'École de Technologie Supérieure (ÉTS) de l'Université du Québec à Montréal et le groupe de recherche Système d'Informations et Architectures Logicielles (SIAL) du Laboratoire du Parallélisme, des Réseaux, des Systèmes et de la Modélisation (PRiSM) de l'Université de Versailles Saint Quentin en Yvelines (UVSQ) en France. Les deux institutions ont collaboré grâce à l'initiative de Hicham Djenidi de ses directeurs de Thèse Nicole Levy et Chakib Tadj, mais aussi avec l'appui et l'expertise du Docteur Amar Ramdane-Cherif, Maître de Conférence au PRiSM.

L'auteur de ce mémoire est le premier doctorant en cotutelle de ces deux institutions, mais il n'est certainement pas le dernier. En effet, suite au succès fructueux de cette collaboration qui s'est traduit par diverses publications dans des revues avec comités de lectures et par des projets franco-québécois, d'autres thésards ont déjà emboîté le pas à Hicham Djenidi.

Par ailleurs, une retombée parallèle à ce travail de recherche doctorale est le développement d'un nouveau cours pour ingénieur intitulé 'Systèmes Interactifs Multimodaux'. Ce cours et ses laboratoires ont été élaborés et sont enseignés régulièrement par Hicham Djenidi à l'ÉTS.

REMERCIEMENTS

Ce travail de recherche, ce mémoire, ne pourraient exister sans un certain nombre de personnes auxquelles je tiens à exprimer ici ma sincère gratitude.

Tout d'abord, je remercie vivement Madame Rita Noumeir de me faire l'honneur de présider le jury de ma soutenance.

Je remercie également Mesdames Isabelle Borne et Sylvie Ratté de me faire l'honneur de leurs participations au jury mais aussi pour leurs examens minutieux de mon travail. Je leurs suis extrêmement reconnaissant pour le travail de lecture critique de ce mémoire puis pour celui d'écriture de rapports sur mon travail de thèse.

Un immense merci va à Nicole Levy et Chakib Tadj pour leur confiance et leur accueil, pour la qualité de leur encadrement, pour la pertinence de leurs directives, et surtout pour les nombreuses discussions qui m'ont permis de faire évoluer mon travail.

Je dois ma profonde reconnaissance à Amar Ramdane-Cherif pour son expertise et son soutien, lui que j'ai tant 'harcelé' de mes questions et qui a contribué à la finalisation de ce document par ses relectures minutieuses et ses recommandations avisées.

Je remercie tous les membres des Laboratoires LATIS et PRISM pour la qualité du cadre de travail qu'ils m'ont offert durant ces quelques années.

À ce propos, je tiens à dire un merci tout particulier à mes collègues des laboratoires, mais aussi à mes amis personnels, qui ont tous su, chacun à leur façon, apporter un indispensable souffle à ce travail : Amanda, Kamel, Madjid, Said, Salvina et Samir.

Enfin, je tiens à dire : "Merci à toi ma très chère épouse pour ta patience et ton appui continuel durant tous ces jours que j'ai passés à travailler sur ce document! Merci à

vous, mes très chers parents, ma très précieuse famille et mes inestimables amis qui avaient grandement contribué à faire de moi ce que je suis aujourd'hui!"

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS ET SIGLES

A:	Agent
ACP :	Agent de Contrôle Parallèle
AdL :	Agent du Langage
AE:	Agent Expert
AL:	Architecture Logicielle
ATAM:	Architecture Tradeoff Analysis Method (Méthode d'analyse par compromis de l'architecture)
BC :	Base de Connaissances
C:	Contrôle
Co:	Composant
Con :	Connexion
CORBA:	Common Object Request Broker Architecture (Architecture d'objets communs courtiers)
CPM:	Communication Personne Machine,
CPN:	Colored Petri Nets (Réseaux de Petri colorés)
CPU :	Central Process Unit (unité centrale d'un processeur)
CSP:	Communicating Sequential Processes (Processus communicants séquentiels)
CCS:	Calculus Communicating Systems (Systèmes communicants de calculs)
E :	Environnement
É:	État mental
F:	Fusion/Fission
FA :	File d'Attente
Fr:	Fragment
G:	Grammaire
Gn:	Génération
HS :	Hiérarchie Sous-jacente

I :	Interaction
IA:	Intelligence Artificielle
IAD:	Intelligence Artificielle Distribuée
In :	Interprétation
L:	Langage
LOTOS:	Language Of Temporal Ordering Specifications (Langage de spécification ordonnée temporisée)
M :	Module
Msg :	Message
N :	Noeud
ML :	Méta Langage
New :	Nouveau, nouvelle
O :	Organisation
OAA :	Open Agent Architecture (Architecture ouverte de type agent)
P:	Phrase
R:	Redondance
RTA:	Réseau de Transitions Augmentées
RP :	Réseau de Petri
RPCT :	Réseaux de Petri Colorés Temporisés
RPCTS :	Réseaux de Petri Colorés Temporisés Stochastiques
PAC:	Présentation Abstraction Contrôle
PDA :	Personnal Digital Assistant (ordinateur personnel de poche)
Pr :	Processus
PVM :	Parallel Virtual Machine (Machine virtuelle parallèle)
S:	Sémantique
SAAM :	Software Architecture Analysis Method (Méthode d'analyse de l'architecture logicielle)
SC :	Sources de Connaissances
Se:	Sérialisation

Sg:	Signal
SGR :	Système de Gestion de Règles
SMA:	Systèmes Multiagent
St :	Stochastique
T:	Temps
TA:	Table d'Aiguillage
TI :	Technologie de l'Information
UML:	Unified Modeling Language (Langage de modélisation unifié)
V:	Vocabulaire

INTRODUCTION

1.1 Avant-propos

Ce mémoire présente nos travaux de recherche qui consistent en une étude des interactions multimodales dans les applications 'multimédia'[1] en vue de proposer des architectures logicielles génériques dynamiques pour ces applications.

Ce projet de recherche impose des expertises dans divers domaines scientifiques connexes comme les applications multimodales, les architectures logicielles et leurs méthodes d'analyse et de spécification, ainsi que les paradigmes 'agent' et 'systèmes multiagent[2]' employés en 'Intelligence Artificielle'.

Depuis le premier système, rudimentaire mais pertinent, de Bolt et al. (Bolt 1980), différentes applications multimodales ont vu le jour, (Cohen 1997; Crowley 1997; McGee 2000; Oviatt 2000-a). Chacune de ces applications est basée sur une architecture du dialogue[3] (Moeschler 1989; Villing 2006; Manchón 2006-a) combinant les modalités en entrée afin d'élaborer l'information multimodale la plus pertinente à présenter en sortie.

Même si nous invoquerons, souvent la multimodalité en entrée des systèmes et le dialogue multimodal qui en découle, nous ne pourrons éviter de nous placer dans une perspective 'Entrée-sortie' (Figure 1.) Il va en effet nous falloir, analyser plusieurs formes de coopérations ('personne machine'), impliquant à la fois l'entrée et la sortie du système (principe du retour immédiat de l'information incluant le parallélisme et/ou l'entrelacement des traitements entre l'entrée et la sortie.)

---

[1] Le mot 'multimédia' sera employé comme adjectif invariable dans ce document car il constituera à chaque fois une élision de l'expression : 'du type multimédia.' Le mot 'média' est le pluriel du nom latin 'médium'. Dans ce document, il sera invariablement employé tel quel au singulier comme au pluriel.

[2] Le mot 'multiagent' sera employé comme adjectif invariable dans ce document car il constituera à chaque fois une élision de l'expression : 'du type multiagent.'

[3] Le dialogue est un échange d'information qui suppose l'intercompréhension des acteurs (systèmes, utilisateurs) de cet échange. Cette recherche d'une compréhension mutuelle implique l'existence d'une

Il apparaît, dès les prémices de la multimodalité, qu'il ne suffit pas d'avoir, à sa portée, des fonctionnalités et des capacités matérielles pour pouvoir développer des applications multimodales. Aujourd'hui encore, de nombreuses questions se posent quant à la résolution de l'interaction multimodale personne-machine aussi bien au niveau conceptuel et logiciel qu'au niveau matériel. Les solutions proposées sont souvent ad hoc et relativement peu de travaux offrent des propositions 'd'architectures logicielles' pour de telles applications. La raison principale qui semble justifier cette allégation est que ces deux disciplines (Architectures logicielles et Applications multimodales) ont constitué deux courants de recherche qui se sont développés durant la même période et qui ont atteint tous deux un début de maturité à la fin des années 90.

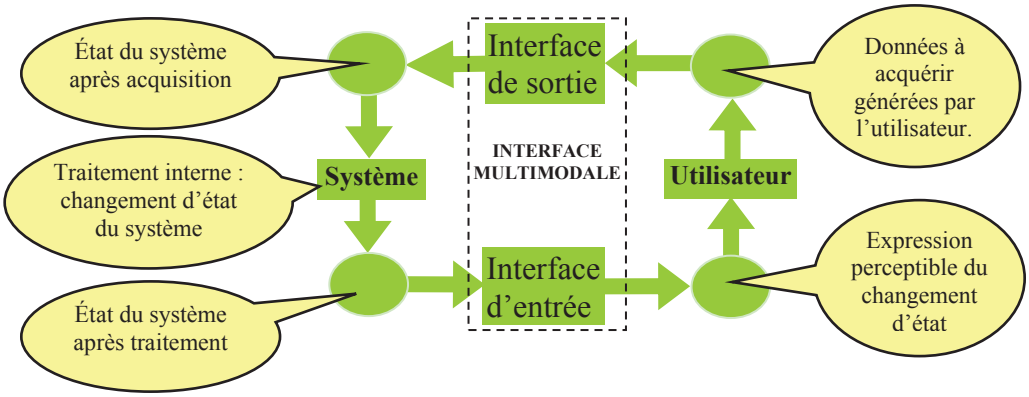

Figure 1 Principe du processus cyclique faisant intervenir la distinction entre une interface d'entrée et une interface de sortie

Lors de la conception d'une application 'intelligente' multimodale il est nécessaire d'analyser les tâches effectuées par l'utilisateur[4] (ou qu'il aura à effectuer), d'identifier ses besoins et de spécifier toutes les contraintes éventuellement humaines, matérielles

propriété générale : l'interaction. L'interaction existe si les actions de l'un des acteurs sont influencées, au moins, par celles d'un autre. L'interaction est donc un résultat du dialogue.

[4] Le mot '*utilisateur*' au masculin est aussi employé ici pour désigner le mot féminin '*utilisatrice*.' Dans un souci de simplification du texte, les mots du genre masculin se rapportant à une personne, seront désormais employés pour désigner à la fois le masculin et le féminin dans tout le reste de ce document.

et/ou autres (exemples : environnement bruité, accès difficiles, comportement d'un utilisateur 'moyen' etc.) Ceci permet d'élaborer les différents liens qu'entretiennent les messages en entrée du système et, de les combiner de façon 'intelligente' dans une architecture du dialogue 'personne-machine-personne'[5] (Watzlawick 1972).

Ces architectures doivent s'adapter continuellement aux changements dus aux perturbations externes ou aux actions de l'utilisateur. Elles sont donc assujetties à des contraintes en cours d'utilisation (en temps réel) lors du dialogue de l'application avec l'utilisateur.

Avec l'essor des applications dites 'intelligentes', un des défis de la recherche concernant la multimodalité est l'élaboration de solutions architecturales qui répondent et qui s'adaptent à ces types de contraintes.

1.2 Définition de notre problématique

De nos jours, nombreuses, sont les applications proposant des interactions 'homme machine' plus flexibles et plus transparentes. Les applications multimédia font appel à des protocoles de communication plus élaborés lorsque plusieurs média sont impliqués. Les systèmes multimédia multimodaux combinent des modes naturels en entrée, comme la parole, le toucher, le geste, etc. et, génèrent une ou plusieurs commandes, envoyée(s) sur un ou plusieurs systèmes de présentation en sortie. Depuis le premier système, rudimentaire mais pertinent, du « Mets ça ici » (Bolt 1980), différentes applications multimodales ont apporté des réponses quand les applications classiques étaient insuffisantes (Bellik 1994; Crowley 1997; McGee 2000). Chacune de ces applications est basée sur une architecture du dialogue combinant les modalités en entrée afin d'élaborer l'information multimodale la plus représentative de la requête de l'utilisateur. De telles élaborations sont appelées fusion, d'où le nom de 'moteurs de fusion' attribué à ce type d'architecture. Pour de telles élaborations, les développeurs se basent souvent

[5] En 1972, Watzlawick et al. avaient déjà défini le type d'interaction : 'personne-machine-personne' et l'avaient différencié de l'interaction 'personne-machine' selon le contexte de la communication (séquencement, bidirectionnalité, modes synchrone ou asynchrone).

sur les expériences passées des autres designers. Aujourd'hui, il n'y a pas de consensus sur des architectures logicielles génériques dynamiques reflétant l'implémentation du dialogue multimodal. Cette absence de consensus constitue le fondement de notre problématique de recherche. Il s'agit donc de tenter d'apporter des solutions aux problèmes posés par les différentes questions suivantes :

a. Quelles propriétés récurrentes et caractéristiques pertinentes, sont implicitement employées dans les dialogues multimodaux, et à quels niveaux et selon quels points de vues devons-nous les considérer dans une architecture logicielle?
b. Quels concepts et quelles théories, sont les mieux adaptés pour définir et décrire ces propriétés.
c. Comment, ces propriétés et caractéristiques, peuvent être mises en œuvre dans des paradigmes architecturaux réutilisables?
d. Quels outils ou méthodologies de modélisation, seront les mieux adaptés au design de ces architectures, en terme de « *généricité* », de modularité et de robustesse (prise en charge des 'erreurs' utilisateurs)?
e. Quelles stratégies de modélisation permettront d'aboutir à une architecture offrant des qualités au logiciel comme, par exemple, la fiabilité, l'adaptabilité dynamique et la garantie de ses performances (Perry 1992).
f. Quels sont alors les sous modèles que nous pouvons proposer pour ces architectures?
g. Quelles simulations et/ou applications pratiques pouvons-nous proposer pour la validation de notre approche?

Les questions a et b concernent l'aspect analyse du problème que pose notre recherche, alors que les questions c à f concernent l'aspect modélisation. La réponse à la dernière question offre les exemples pratiques qui appuient et valident nos propositions.

1.3 Objectifs de recherche

L'objectif principal de notre recherche est de proposer des modèles d'architectures logicielles réutilisables qui rassembleront les propriétés récurrentes implicitement employées dans les dialogues multimodaux. Ces architectures auront pour finalité un comportement intelligent car elles auront pour finalité la réalisation de deux propriétés fondamentales : l'accès universel et la mobilité.

L'accès universel constitue la première et majeure motivation du développement d'interfaces multimodales plus flexibles. En effet, la multimodalité permettrait à une plus vaste population (plus diversifiée et non spécialisée) d'utilisateurs d'employer les systèmes informatiques. Les capacités intellectuelles, cognitives et motrices lors de la communication avec des systèmes multimodaux ainsi que les préférences et choix d'utilisation des modes de communication varient de manière significative entre les individus. De ce fait, les applications multimodales devraient permettre une communication 'personne-machine-personne' plus naturelle et ce pour des utilisateurs de tous âges, niveaux d'expertises et cultures ainsi que pour des personnes ayant des déficiences physiques, sensorielles ou intellectuelles.

Un autre avantage majeur des interfaces multimodales est qu'elles permettent d'étendre le contexte d'utilisation viable d'une application pour y inclure la possibilité de modification dynamique des paramètres d'utilisation et des traitements lorsque l'application est mobile (Zouinar 2004). En particulier, elles laissent les utilisateurs libres de changer de mode durant des conditions d'utilisation variables. Comme les modes en entrée d'une interface multimodales peuvent être complémentaires, leurs combinaisons peuvent procurer une plus grande variété d'action selon différents contextes changeants.

Un exemple typique est l'utilisation de la voix pour commander un système lorsque l'utilisateur doit employer ses mains à d'autres tâches dans un contexte de pilotage.

L'originalité des travaux envisagés peut se résumer par les différentes étapes suivantes que nous avons réalisées.

1. Nous proposons une approche nouvelle qui répond aux carences des solutions architecturales traitées dans la littérature :

a. l'architecture est dédiée à tout type de média;
b. l'architecture permet la fusion et fission multimodale à tous les niveaux d'abstractions (niveaux sémantique, syntaxique, etc.);
c. l'architecture est 'reconfigurable' dynamiquement grâce au système multiagent que nous proposons;
d. l'architecture vérifie des attributs de qualités adaptés aux problèmes de la multimodalité.

2. Des propositions de spécifications motivées et originales du système multiagent sont proposées pour modéliser l'architecture multimodale, suite à une analyse comparative des modélisations RPCT, par rapport aux autres types de modélisations. Cette analyse est menée dans le but d'expliciter les apports de ces méthodes de modélisation et de spécification pour les applications multimodales :

a. par l'emploi des RPCT stochastiques;
b. par des validations de notre approche qui sont réalisées par des outils de simulation et par le développement d'applications.

3. Enfin, nous proposons une méthodologie multiagent de reconfiguration dynamique de l'architecture multimodale basée sur des scénarios (correspondants à des propriétés de caractéristiques d'attributs de qualités) permettant de maintenir et d'améliorer un modèle de qualité logiciel.

1.4 Contenu et structure de ce document

Nous offrons dans ce document une revue de la littérature qui nous permet de proposer des approches architecturales multimédia multimodales originales et nouvelles pour la résolution des problématiques énoncées précédemment. Il est donc nécessaire de réaliser, au préalable, une synthèse bibliographique qui prend en compte trois aspects :

a. la caractérisation des interactions multimodales;
b. les types d'architectures qu'elles impliquent;
c. les méthodes de modélisation et de spécification de ces architectures.

Cet état de l'art fait l'objet de la première partie de ce document (chapitre 2.) Il définit l'interaction multimodale et présente un résumé synthétique accompagné d'une analyse critique des travaux relatifs à ces différents domaines de recherche (a, b et c) depuis le début des années 90. Cette analyse critique met en évidence les aspects peu abordés par la littérature et nous permet de nous positionner par rapport à ces recherches. Cette partie pose ainsi les fondements de l'approche que nous proposons.
Nous décrivons alors notre problématique de recherche en termes d'objectifs à atteindre. Nous mettons l'accent sur l'originalité de la contribution que nous désirons apporter à ce domaine de recherche. Ceci fait l'objet du chapitre 3, où nous détaillons alors la méthodologie envisagée pour résoudre notre problématique en décrivant l'approche qui permet de réaliser, spécifier formellement et raffiner (de manière descendante ou ascendante) une architecture logicielle multimodale qui répond aux carences et lacunes de celles proposées dans la littérature (présentées au chapitre1). Nous mettons alors l'accent sur l'originalité de notre contribution à ce domaine de recherche. La description de notre problématique, de notre méthodologie, ainsi que de ses atouts et aboutissants pour les architectures logicielles multimodales constitue les chapitres deux et trois de ce document.

Le chapitre 4 expose des exemples d'applications concrètes et didactiques de la méthodologie développée au chapitre 3. Nous présentons des applications multimodales dont les architectures logicielles sont spécifiées et raffinées par réseaux de Petri colorés de haut niveau.

Le chapitre 5 propose une méthodologie multiagent pour reconfigurer dynamiquement l'architecture logicielle multimodale en tenant compte d'un profil de qualité logiciel décrit par le concept du *scénario*. Il s'agit en fait d'un agent expert qui contrôle et reconfigure l'architecture logicielle multimodale.

Le chapitre 6 développe les aspects structurels et fonctionnels de l'agent expert présenté au chapitre précédent et donne un exemple de reconfiguration. Un profil de qualités logicielles, adapté à un prototype multimodal dédié aux handicapés moteurs, permettra de reconfigurer l'architecture du dit prototype.

Nous conclurons ce mémoire par un résumé de nos contributions et une description des perspectives que nous envisageons.

CHAPITRE 2

PRÉSENTATION ET ANALYSE CRITIQUE DE L'ÉTAT DE L'ART RELATIF À NOTRE PROBLÉMATIQUE DE RECHERCHE

2.1 Caractérisation des interactions multimodales

2.1.1 Introduction

La multimodalité représente le plus souvent, pour les informaticiens, la capacité d'un système interactif à utiliser plusieurs canaux de communication lors de l'interaction entre l'utilisateur et le système, et cela au cours d'une même session.

Lorsqu'un système multimodal, reçoit des informations en entrée, il en fait une *analyse* (par simple combinaison ou par organisation selon une sémantique, par exemples) pour en donner *une interprétation*. Le résultat de l'interprétation est conservé par le système ou complété par de nouvelles informations nécessaires à l'élaboration d'une réponse qui sera présentée à l'utilisateur. 'Cette capacité *d'interprétation* et de *compréhension* des informations en provenance de différentes entrées, constitue la principale différence entre les systèmes multimodaux et les systèmes multimédia' (Bellik 1995). Pouvons-nous, pour autant, dire que les systèmes multimodaux sont 'intelligents'? Nous tenterons de répondre à cette question, de façon argumentée, lorsque nous aborderons plus en détail, le paradigme 'Agent' pour la modélisation des architectures du dialogue multimodal. En attendant, nous nous permettons de qualifier ces systèmes 'd'intelligents', simplement du fait de leurs capacités d'*interprétation* et de *compréhension* (capacités attribuées à un comportement intelligent.)

En plus de pouvoir acquérir, stocker et restituer des informations de natures différentes (texte, images, sons, séquences vidéo...) comme le font les systèmes multimédia, les systèmes *multimédia multimodaux* ont donc la capacité de combiner ces informations

pour leur donner un sens (Nigay 1996). De ce fait, la modalité est un concept qui fait intervenir celui du sens et celui du média. Le terme « sens » représente ici l'organe de perception (l'ouïe, le toucher, la vue, l'odorat ou le goût) incluant ses fonctionnalités d'analyse de l'information. En résumé, '*si l'information dans un système multimédia est l'objet de la tâche, elle sert au contraire à contrôler la tâche dans un système multimodal'* (Azémard 1995).

Afin de clarifier les concepts de modalité et de média, nous présentons au paragraphe 2.1.2 des définitions et terminologies pertinentes issues de la littérature. Le paragraphe 2.1.3 présente trois exemples de taxonomies qui répondent à une nécessité d'intégration des modalités dans les systèmes multimodaux. Le dernier paragraphe traite de la représentation des événements multimodaux dans la littérature. Nous préciserons à chaque fois notre point de vue et/ou notre choix pour notre problématique.

2.1.2 Définition et terminologie du 'média' et de la modalité

2.1.2.1 Le média

Dans le dictionnaire académique de la langue française (Hachette 2004), le **média** est un procédé technique permettant la distribution, la diffusion ou la communication des œuvres de l'esprit écrites, sonores ou visuelles (depuis la presse imprimée jusqu'à l'ordinateur…) De façon générale et en termes de Communication Personne-Machine (CPM), la littérature scientifique définit un média comme étant un dispositif servant de support d'information. Cette définition a été affinée par différents auteurs pour répondre à plusieurs types de finalités (Nigay 1996), dans le cadre de la CPM.

Tableau I

Comparaison des définitions du média

Auteur(s)-Année-Référence	**Définition- Exemple**	***Finalité de la définition***			
		Technique		**'Technico-humaine'**	
		Physique	**Logiciel**	**Couplage** Dispositif physique et qualités sensorielles humaines	**Système représen-tationnel**
Sens Commun ramené à la CPM-1991-(Hachette 2004)	**Média :** support physique d'information. **Exemples:** Cd-rom, disquette.	X			
	Média physique : dispositif physique capteur ou effecteur d'un système informatique. **Exemples:** souris = capteur, écran = effecteur.	X			
Blattner -1990-(Blattner 1990)	**Média logiciel** : toute forme logique non matérielle ou physique qui véhicule de l'information. **Exemple:** message électronique.	X	X		
Arens et al.-1993 -(Arens 1993) Frohlich- 1991-(Frohlich 1991)	**Média**: système représentationnel. **Exemples :** un texte en langage naturel, un graphique, un système logiciel de présentation de données, bref tout ce qui fait appel à l'interprétation et à la compréhension.				X
Bernsen -1994 -(Bernsen 1994)	**Média :** couplage d'un ensemble de qualités perceptives avec le dispositif sensoriel humain nécessaire à la perception de ces qualités. **Exemples:** qualités visuelles et graphiques couplées à la vision.			X	
Bourguet- 1992 –(Bourguet 1992)	**Média** : dispositif qui a la capacité matérielle de véhiculer des informations d'une certaine ***substance*** organisée en une ***forme***.- ***substance*** d'une expression = sa réalité physique observable en tant que matériau non analysé. **Exemples:** son, image, etc. **- *forme*** d'une expression = structure attribuée à cette expression. **Exemples**: pièce musicale, images animées,…	X			X

a. Une finalité purement matérielle : le média est vu comme un capteur ou un effecteur d'un système informatique. Il est communément appelé dispositif d'entrée/sortie;
b. Une finalité purement technique : le média est considéré comme un procédé physique et/ou logiciel utilisable comme véhicule ou support d'information;.
c. Une finalité 'technico-humaine' à deux niveaux d'abstraction : le média est défini comme un couplage d'un dispositif physique, aux qualités sensorielles humaines, et aussi comme un système représentationnel.

Le Tableau I résume les contributions de différents auteurs pour la définition du média.
Ce qui apparaît dans ce Tableau est que chaque auteur possède sa propre conception du média.
Du point de vue perception humaine, certaines définitions, comme celles de Bernsen et de Frohlich sont presque à l'opposée l'une de l'autre, indépendamment de leurs finalités. Pour Frohlich le média n'a aucune qualité de perception humaine : il nécessite l'interprétation et la compréhension de la part de l'être humain. Par contre, Bernsen considère que le média comporte une composante de perception humaine (la vision ou l'audition, etc.) et fait donc appel uniquement à 'l'interprétation.' Ceci s'explique par le fait que Bernsen propose une taxonomie pour les systèmes de présentation en sortie, alors que de Frohlich permet une classification entrée/sortie.
Cette absence de consensus impose de toujours préciser, au préalable, la définition choisie lorsque nous parlons d'applications faisant intervenir des média.
Pour notre part, nous n'attribuons pas au média la capacité de perception et d'extraction du contenu sémantique du message conceptuel qu'il transporte qu'il capte ou qu'il émet.

2.1.2.2 La modalité

Comme pour le média, la littérature offre plusieurs définitions de la modalité. Chaque définition dépend du contexte de son utilisation. Il n'y a évidemment pas d'unanimité sur la terminologie. De ce fait, comme pour le cas du terme 'média', les chercheurs sont

contraints pour éviter tout quiproquo de donner explicitement leurs propres définitions dans chacune de leurs publications ou encore d'employer des exemples qui permettent une définition implicite de ce terme. Ce sont les définitions de la modalité en psychologie et en linguistique qui sont à l'origine des définitions employées en CPM. Le Tableau II résume trois définitions de la modalité qui caractérisent les trois principales tendances de la littérature.

Tableau II

Comparaison des définitions de la modalité

Auteur(s)-Année-Référence	***Définition***	***Finalité de la définition***		
		Système	**Utilisateur**	**Conjointe**
Martin--1995- (Martin 1995)	Processus informatique d'***analyse*** et de ***synthèse,*** défini sur des ensembles de données d'entrée-sortie.	X		
Bellik -1995- (Bellik 1995)	***Structure*** des informations échangées, telle qu'elle est perçue par l'être humain		X	
Duke et all.-1994 -(Duke 1994)	***Technique d'interaction*** dépendant: - des capacités sensorielles de l'utilisateur, - des dispositifs physiques ou logiques engagés dans l'interaction.			X

Les approches ou finalités pouvant être envisagées sont les suivantes.

a. une finalité purement utilisateur où l'échange d'information se fait de la machine vers l'homme et où la modalité est définie par rapport au point de vue humain et non pas en fonction de l'emploi de l'information par la machine;
b. une finalité purement système où l'échange d'information se fait de l'humain vers la machine et où la modalité se définit par rapport à l'emploi de l'information par la machine;
c. une vision conjointe du système et de l'utilisateur.

C'est cette dernière vision que nous prenons en compte dans nos travaux.

2.1.3 Taxonomies des modalités pour l'intégration multimodale

Pour réaliser l'intégration des modalités dans des applications multimodales de nombreux auteurs ont proposé un classement des systèmes multimodaux pour répondre aux questions suivantes :

a. Qu'est ce que la multimodalité?
b. Comment un système particulier donné peut-il supporter différentes modalités?
c. Comment mettre en relation différents types d'informations produites par différentes modalités et comment les combiner?

Nous choisissons de présenter trois contributions qui répondent ou tentent de répondre à ces questions.

2.1.3.1 Approche de Martin et al.

Une première taxonomie des modalités (due à Martin et al. (Martin 1993; Martin 1995-a; Martin 1997)) qui a tenté de répondre à ces questions, est basée sur deux dimensions : les types et les buts entre les modalités. Les modalités sont définies comme étant 'la manière d'exploiter un système physique particulier permettant ainsi l'échange

d'information entre l'utilisateur et un système informatique.' La multimodalité est définie comme étant 'des coopérations entre plusieurs modalités, afin d'améliorer la communication personne machine'. Les 'buts des coopérations' décrivent les pré requis et les conditions nécessaires à l'interface 'personne machine' en termes d'amélioration de l'interaction pour la rendre plus précise, plus intuitive, plus efficace et adaptative à différents utilisateurs et environnements. Selon les travaux de l'auteur et ses collègues (Martin 1995-a), nous distinguons six types de coopérations possibles :

a. complémentarité: l'utilisateur dit "mets ça ici", en pointant un objet, puis un lieu.
b. redondance: un client dit "je veux un second item à droite", simultanément il pointe vers cette direction;
c. équivalence: l'option de choix dans un menu par sélection via une commande vocale ou un clic de souris;
d. spécialisation: un kiosque d'information offrant différents services sélectionnés en touchant les boutons correspondants. (Elle se manifeste par les préférences de l'utilisateur);
e. concurrence : l'utilisation parallèle de plusieurs modalités pour des actions distinctes;
f. transfert: interfaces hypermédia où un clic de souris conduit à l'affichage d'une image.

Cette taxonomie ne fait pas apparaître les aspects de parallélisme (pour une même commande) et les types de données fusionnées. Les travaux de Nigay (Nigay 1993; Nigay 2001) et Coutaz (Nigay 1994) mettent en valeur ces aspects.

2.1.3.2 Approche de Coutaz et Nigay

Coutaz et Nigay (Nigay 1993; Nigay 1994; Nigay 2001) caractérisent les systèmes multimodaux à travers trois dimensions qui sont : le niveau d'abstraction du traitement, l'emploi des modalités et le type de fusion.

Le niveau d'abstraction représente un des multiples niveaux de traitement des données par un système particulier. Ces niveaux d'abstraction s'étendent depuis le niveau signal jusqu'au niveau sémantique de l'information. Au niveau signal, il n'y a pas d'interprétation. La lecture sur les lèvres est un exemple d'intégration de modalités au niveau signal. Dans ce cadre, les signaux élémentaires audio et image sont associés à un bas niveau d'abstraction pour obtenir une interprétation plus précise de l'événement issu de leur combinaison. À l'opposé, l'exemple classique du 'mets ça ici', accompagné de la désignation d'un objet, (Bolt 1980) nécessite l'interprétation séparée des événements geste et parole pour aboutir à l'émergence d'une information au niveau sémantique (à un niveau d'abstraction élevé.) Cette distinction de l'intégration des modalités selon le niveau d'abstraction du traitement opéré est équivalente à l'approche dite du *couplage fort* versus celle du *couplage lâche* des modalités (voir l'exemple dans (Sarukkai 1997).)
Deux types d'applications multimodales sont concernés dans cette distinction.
Premièrement, nous avons les applications dédiées aux systèmes des reconnaissances 'vocale-labiale', 'gestuelle-vocale' et 'scripturale-vocale' (Cosi 1994; Jourlin 1998; Chibelushi 2002) qui sont basées sur des architectures réalisant l'intégration des modalités, au niveau signal ('fusion lexicale'[6] de fragment de mots, phonèmes, gestes élémentaires, caractères et symboles écrits élémentaires.) La combinaison se fait de façon incrémentale et, généralement, une modalité dominante (la parole ou l'écriture etc.) est renforcée par une modalité secondaire, au cours du processus de reconnaissance. Les architectures associées peuvent intégrer des réseaux de neurones dédiés aux

[6] L'expression 'fusion lexicale' est définie un peu plus loin dans le texte.

différentes modalités, des chaînes de Markov cachées ou, enfin, réaliser des combinaisons architecturales hybrides des deux types de réseaux.
En second lieu, il s'agit des applications dédiées à l'aide des utilisateurs susceptibles de changer de modalité à cause de certaines conditions contraignantes et/ou à cause de leurs états physiques, psychologiques, etc. Ces applications gèrent les actions de l'utilisateur comme des événements à fusionner. Elles concernent l'aide aux handicapés (Farcy 2004; Jacquet 2004), la mobilité (Oviatt 2000-a; Oviatt 2000-b), etc.

L'emploi des modalités concerne la disponibilité temporelle des modalités qui peuvent être employées séquentiellement et en parallèle.
Le type de fusion est la possibilité de combiner plusieurs événements en entrée du système. Coutaz et Nigay distinguent trois types de fusions selon le niveau d'abstraction de l'information.

La fusion sémantique : C'est une combinaison de commandes pour obtenir une autre commande ou fonction. Les commandes prises séparément ont un sens et combinées entre elles donnent un autre sens.
Exemple : Dans l'éditeur de dessin 'VoicePaint' (Gourdol 1992), deux commandes (dessiner une droite et changer l'épaisseur d'un objet) sont combinées en une seule pour obtenir une droite à plusieurs épaisseurs.

La fusion syntaxique : C'est une combinaison d'unités d'information pour obtenir une commande ou un effet. Considérées de façon isolée, les unités d'information ne sont pas porteuses de sens.
Exemple : Dans le logiciel NoteBook (Nigay 1994), la combinaison des informations pour obtenir la commande complète 'insérer une note' avec un lieu d'insertion correspond à un cas de fusion syntaxique. L'unité d'information 'insérer une note' n'a de sens qu'après sa combinaison avec une autre unité indiquant le lieu d'insertion dans ce logiciel.

La fusion lexicale : C'est une combinaison d'actions physiques pour obtenir des unités d'information au niveau signal.
Exemple (Nigay 1996) : Dans le Macintosh, l'enfoncement conjoint de la touche 'shift' et d'un click de souris sont fusionnés par le système en un seul évènement. Les informations acquises et fusionnées constituent les lexèmes[7] du langage utilisé. Cette fusion est qualifiée de bas niveau puisque chaque information considérée isolément ne constitue pas un lexème.
Pour chaque type de fusion le processus inverse (fission) peut être envisagé.

2.1.3.3 Les critères de fusion selon Bellik

Bellik a affiné et complété la classification de Coutaz et Nigay en distinguant des critères de fusion selon trois paramètres (Bellik 1995) :

a. La production des énoncés : qui indique si les énoncés (en entrée ou en sortie) doivent être produits séquentiellement ou s'il est possible que plusieurs énoncés indépendants soient produits en parallèle;
b. l'usage des média : qui indique si l'usage des média doit être exclusif, c'est à dire qu'à un instant donné un seul média peut être utilisé ou si au contraire, il est possible d'en utiliser plusieurs simultanément;
c. le nombre de média par énoncé : qui indique si lors de la production d'un énoncé, il faut utiliser un seul média ou s'il est possible d'en utiliser plusieurs.

Dans ce dernier cas, il est nécessaire de procéder à la fusion des différentes informations provenant des différents média, en entrée.
Ces trois paramètres définissent huit types de multimodalités. Logiquement, il existe une combinaison qui ne peut être produite. En effet, nous ne pouvons avoir d'usage

[7] Le lexème est la plus petite unité d'information qui porte un sens.

simultané des média, si la production des énoncés doit être séquentielle avec l'utilisation d'un seul média par énoncé. Il est possible de représenter ces types par les sommets d'un cube dans un espace à 3 dimensions (Figure 2.)

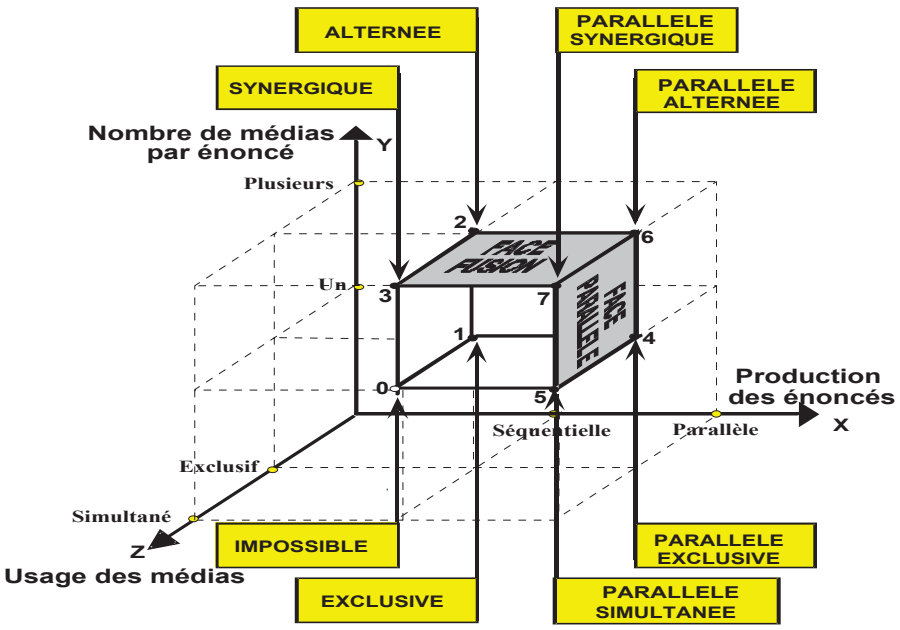

Figure 2 Les différents types de multimodalités selon Bellik (Bellik 1995)

Plus récemment, les chercheurs combinent les stratégies de fusions/fissions multimodales pour tenir compte des avantages de chacune d'elles.

2.1.3.4 Stratégies hybrides pour la fusion multimodale

Les auteurs des références (Pérez 2005; Manchón 2006-b) proposent une stratégie hybride qui est le mixage de deux stratégies. La première est une fusion basée sur une grammaire composée d'entrées multimodales (combinaison de modalités) et sur des contraintes temporelles et modales. Cette première stratégie ne fait intervenir aucun autre processus de décision dans la fusion. La seconde stratégie est basée sur un

processus de dialogue multimodal et implique des entrées grammaticales unimodales, des contraintes temporelles et modale le tout accompagné d'un processus de décision fondé sur le niveau du dialogue et employant :

a. le déplacement du dialogue sur différents niveaux du processus de fusion;
b. le type de modalité;
c. le temps séparant les événements en entrée;
d. l'ordre des applications du contrôle du dialogue sur les différents niveaux;
e. les déplacements du dialogue déjà exécutés;
f. l'historique des déplacements du processus du dialogue.

Le but de cette stratégie hybride est de faire en sorte d'employer les avantages des stratégies la composant tout en inhibant les inconvénients que chacune d'elles entraine lorsqu'elles sont employées indépendemment l'une de l'autre.

Il reste maintenant à représenter les évènements multimodaux en entrée des systèmes pour les fusionner. Pour ce faire, différentes approches ont été proposées.

2.1.4 Représentation et spécification des événements multimodaux

2.1.4.1 Introduction

Avant de présenter les approches proposées dans la littérature pour la spécification et la représentation des événements des dialogues multimodaux, nous commençons par donner une définition sommaire de la 'spécification formelle,' puis nous ferons une analyse détaillée des différentes approches afin de pouvoir effectuer un choix justifié pour la résolution de notre problématique.
La spécification consiste en l'établissement du cahier des charges et de la description des contraintes d'un système.

Afin de clarifier la notion de spécification formelle en génie logiciel, nous reprenons les sept mythes attribués aux méthodes formelles et énoncés dans l'article de Hall (Hall 1990).

'L'utilisation des méthodes formelles produit un logiciel parfait'. Ceci est un non-sens car une spécification formelle est un modèle du monde réel et peut donc inclure des erreurs, des omissions et des malentendus.

'L'utilisation des méthodes formelles signifie faire de la preuve de programme'. La spécification formelle d'un système est valable sans vérification formelle des programmes, car elle force, à une analyse détaillée et à une simulation, dans le cycle de développement.

'Les méthodes formelles ne sont justifiables que pour des systèmes critiques'. L'expérience industrielle montre que les coûts de développement sont réduits pour tous les types de systèmes.

'Les méthodes formelles sont pour les mathématiciens.' C'est un non-sens car les mathématiques employées sont élémentaires (de niveau pré universitaire) pour la majorité d'entre-elles.

'Les méthodes formelles augmentent les coûts de développement'. Ceci est faux car il y a plutôt un déplacement des coûts vers les premières phases et donc, possiblement, un moindre coût.

'Les clients ne peuvent comprendre les spécifications formelles'. Ils le peuvent : il suffit de les représenter graphiquement, de les paraphraser en langage naturel ou de les prototyper.

'Les méthodes formelles ne sont utilisées que pour les systèmes triviaux, académiques.' Il existe de nos jours de nombreux projets industriels concernant des systèmes non triviaux qui ont été mis en oeuvre.

Voyons maintenant les méthodes formelles et informelles employées dans la littérature pour spécifier ou représenter les événements multimodaux.

2.1.4.2 'Typed feature Structures'

Cette approche (Cohen 1997; Johnston 1997) permet de transformer des événements multimodaux en entrée, en structures typées représentant les contributions sémantiques des différentes modalités. Les structures typées obtenues sont alors combinées dans des opérations d'unification. La composante temporelle n'apparaît pas dans la spécification proposée par cette approche.

2.1.4.3 Représentation syntaxique

Dans cette proposition (Faure 1993), les événements multimodaux en entrée sont représentés par des triplets (verbe, objet, lieu.) Cette représentation est suffisante pour la parole en entrée avec des références exprimées par le geste (désignations de la main) mais elle reste insuffisante lors de la généralisation à d'autres événements multimodaux.

2.1.4.4 Structures d'actions partielles

Chaque modalité en entrée est interprétée séparément puis analysée et transformée en structures ('*frames'*) sémantiques contenant des labels spécifiant des paramètres de commande ('*parameter slots'*) (Vo 1996; Vo 1997; Vo 1998). Les informations dans ces structures d'actions partielles sont rassemblées par un algorithme appelé '*Domain-independent frame-merging algorithm*'. Cet algorithme combine les '*frames*' partielles

en une '*frame*' complète. À chaque groupement de séquences d'événements d'entrée est assigné un taux correspondant à leurs informations mutuelles. Un algorithme dynamique (similaire à l'algorithme de recherche de Viterbi ou encore à l'algorithme *Dynamic Time Warping* employé dans les systèmes de reconnaissance vocale (Rabiner 1993)) détermine alors la meilleure séquence d'événements interprétés qui correspond à l'événement multimodale global en entrée. L'architecture du moteur de fusion correspondant est appelée: '*Multi-state mutual information network*'.

2.1.4.5 Le langage de spécification pour les applications multimodales

Ce langage (Martin 1995-a) présente un moyen de spécification simple pour décrire les coopérations entre modalités dans une application multimodale. Pour chaque tâche, le langage spécifie quelles modalités peuvent être employées pour exprimer certains paramètres, et comment ces paramètres son transmis au module d'exécution de la commande multimodale. Pour chaque modalité, une liste, d'événements élémentaires, décrit quelle portion d'information peut s'exprimer par cette modalité et de quelle manière. Une information est employée par des modules pour définir des vocabulaires. Cependant les paramètres temporels ne sont pas spécifiés par ce langage. En effet, un serveur de modalités '*MServer'* (Bourdot 1995) a été développé dans le but de dater et trier chronologiquement les événements détectés par les périphériques en tenant compte de leurs différences de temps de traitement. Ce serveur est un processus maître qui fait transiter des informations entre des processus gérant les périphériques et une ou plusieurs applications multimodales. Les événements fournis par ce serveur de modalités, sont ensuite intégrés et interprétés. TYCOON (Martin 1995-a), un noyau multimodal, a été développé dans ce but. Il comprend ce langage de spécification permettant la déclaration de plusieurs types et buts de coopérations entre modalités. Ce langage ne permet donc pas de paramétrer le temps dans un but de spécifier de façon générique les événements multimodaux.

2.1.4.6 Technique du creuset

Cette technique (Nigay 1995) permet d'effectuer la fusion sur la base de trois critères: la structure des objets à combiner, le temps et le contexte. À chacun de ces critères, correspond un traitement spécifique de la fusion. Ces traitements sont appelés respectivement micro fusion, macro fusion et fusion contextuelle.

En complément, il convient de considérer dans cette technique l'anti-critère suivant: si les structures logiques des unités à fusionner sont incompatibles, leur fusion est impossible. La complémentarité structurelle est donc une condition nécessaire mais non suffisante à la fusion, que le critère soit micro, macro ou contextuel.

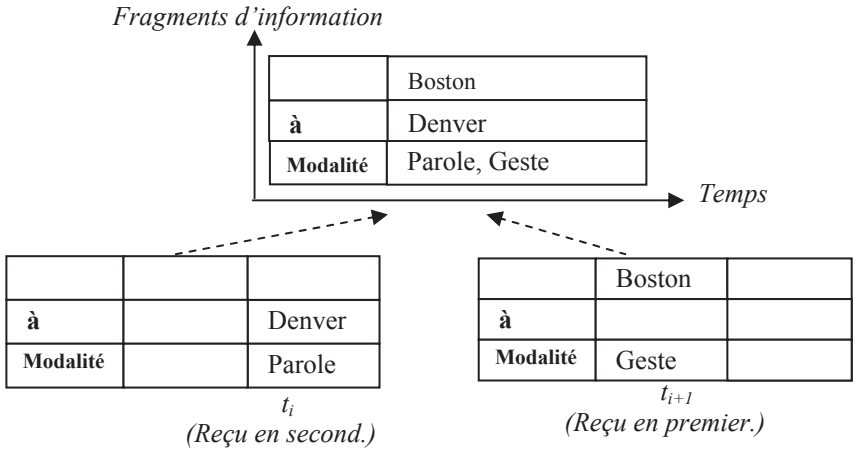

Figure 3 Technique du creuset selon Coutaz et Nigay (Nigay 1995)

Ces trois conditions sont présentées, ci-après, dans l'ordre de priorité décroissante de leur vérification au cours du processus de fusion.

a. Le critère de micro fusion temporelle s'appuie sur le temps et combine des informations, structurellement complémentaires ayant des intervalles de temps qui se chevauchent (parallélisme ou pseudo parallélisme) ;

b. Le critère de macro fusion temporelle s'appuie sur le temps et combine des informations structurellement complémentaires qui appartiennent à une fenêtre temporelle ('épaisseur du présent' ou proximité temporelle);
c. Le critère de fusion contextuelle utilise uniquement le contexte et fusionne des informations structurellement complémentaires.

La Figure 3 montre un exemple de macro fusion temporelle employant des contraintes d'événements : l'évènement parole "Denver" est associé à l'évènement de désignation "Boston" (l'utilisateur pointe Boston sur une carte) pour une requête : "De Boston à Denver."
La technique du creuset permet, selon Coutaz et Nigay, la fusion aux niveaux lexical, syntaxique et sémantique. Cependant, aucune spécification formelle ne lui est associée.

2.1.4.7 XSPECIMEN

XSPECIMEN (Krus 1993) est un éditeur graphique et interactif des réseaux de transitions augmentés (RTA), utilisé pour décrire les interactions dans un dialogue multimodal. L'éditeur assiste l'utilisateur dans sa tâche de conception des interfaces. Il est notamment le garant de la syntaxe et de la grammaire[8] employées dans le dialogue. L'application XSPECIMEN a été développée suivant un modèle orienté objet en utilisant le langage C++. Les structures définies dans cet éditeur ne sont pas dynamiques puisqu'elles passent par la déclaration d'objets C++ qui, a priori, ne peuvent être modifiés après leur création. Un tel dynamisme du graphe, s'il existait, permettrait, par exemple, de faire évoluer le dialogue en fonction du comportement de l'utilisateur ou de l'état interne de l'application. Au niveau de la phase de spécification logicielle du dialogue multimodal, cette façon de procéder conduit à des modèles fonctionnels mais statiques en termes de reconfigurations dynamiques (lors de l'emploi ou de l'abandon d'une nouvelle modalité par l'utilisateur ou encore lorsque les conditions doivent être

[8] L'Annexe 1 donne une définition formelle des grammaires.

modifiées.) Une solution pour répondre à cette contrainte du dynamisme serait de prévoir, à l'avance, toutes les combinaisons possibles dans le graphe de transition décrivant le dialogue selon une grammaire donnée. Par ailleurs, cet outil ne permet pas de spécifier le temps dans le dialogue multimodal.

2.1.4.8 Analyse et choix pour notre problématique

Même si les techniques formelles de description datent déjà de plus de 35 ans en ingénierie logicielle (comme c'est le cas pour (Parnas 1969)), peu d'entre elles ont été adaptées et employées pour décrire les systèmes multimodaux. Nous pouvons, néanmoins, citer les travaux de Duke et Harrison (Duke 1997) ou ceux de McColl et Carrington (MacColl 1998), qui présentent, respectivement, comment les méthodes de spécification en Z (Davies 1996) et par les CSP[9] (Schneider 2000) ont été adaptées et employées pour modéliser un cas particulier de dialogue multimodal : Il s'agit d'un système d'information dédié à la réservation de vols en avion (système appelé MATIS (Coutaz 1995).)

Les systèmes interactifs multimodaux comportent des propriétés et des caractéristiques intrinsèques que les techniques de descriptions formelles, employées en génie logiciel, ne peuvent prendre en charge. Par exemple, l'interactivité nécessite une modélisation de comportements événementiels avec parallélisme difficile à capturer par des descriptions en Z (Davies 1996). Par ailleurs, les contraintes temporelles sont fondamentales pour la plupart de ces applications. Beaucoup d'entre elles les emploient dans le parallélisme et le dynamisme (en temps réel.) Par exemple, les actions des utilisateurs peuvent se produire simultanément sur plusieurs périphériques en entrée et le mécanisme de fusion doit se produire en temps réel pour combiner ces actions.

Les descriptions formelles employant des sémantiques entrelacées, comme les algèbres de processus classiques (CSP, CCS[10](Milner 1989) ou LOTOS[11] (Bolognesi 1988)) ne

[9] Acronyme de Communicating Sequential Processes.
[10] Acronyme de 'Calculus Communicating Systems'.
[11] Acronyme de 'Language of Temporal Ordering Specifications'.

sont pas capables de modéliser de tels comportements avec un parallélisme réel (appelé aussi parallélisme non déterministe car il permet la simultanéité parfaite dans le temps, par opposition au parallélisme entrelacé.) Par ailleurs, l'emploi de fenêtres temporelles, dans les mécanismes de fusion multimodales, requiert la prise en compte du temps d'une manière quantitative par les méthodes de spécifications. Par exemple, à l'arrivée d'un événement donné il faut que la méthode de description puisse allouer un intervalle de temps durant lequel le processus de fusion est autorisé à se produire avec un autre événement survenant pendant cette durée.

Des problèmes de gestion temporelle surviennent également, lorsqu'il est nécessaire de prendre en compte l'instant absolu auquel l'utilisateur commence à parler par exemple. Ceci permet de détecter des 'coréférences' avec coïncidence temporelle (Bellik 1995). Pour faire face à ce type de problèmes, la spécification doit permettre de dater et trier chronologiquement les événements détectés en entrée, en tenant compte de leurs différences de temps de traitement par chacun des périphériques.

De plus, l'évolution d'un système multimodal dépend des conditions relatives aux variations externes des données Ces variations ne peuvent pas être modélisées par les réseaux de transition augmentés (RTA.) Le talon d'Achille des RTA est l'absence d'un langage formel, associé aux réseaux, pour spécifier ces actions. La Figure 4 représente un dialogue personne machine incluant une phrase sémantique (composée de mots prononcés par l'utilisateur et de clics de souris.)

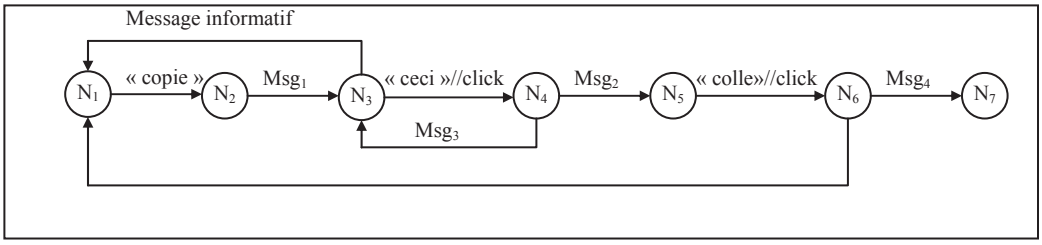

Figure 4 Exemple de modélisation par Réseaux de Transitions Augmentés (repris de (Bellik 1995))

Dans la Figure 4, Msg_i représente un message i, N_i (nœud i) représentant l'état i, et les flèches labellisées représentent des actions. Une expression entre guillemets représente une commande vocale prononcée par l'utilisateur. Le symbole // est un opérateur binaire indiquant des actions qui se produisent en parallèle, sans aucune indication sur leurs durées d'exécution. L'absence de langage se traduit par l'absence de variables typées et d'une syntaxe pour modéliser les attributs (temporels, de formes, etc.) des événements, se produisant dans ces systèmes.

En contrepartie, le méta langage fortement typé des réseaux de Petri colorés[12] temporisé stochastiques (CPN-Tools (Jensen 1995; Jensen 1997-a; Jensen 1997-b; University of Aarhus 2006)), peut être utilisé pour réaliser ces spécifications. De ce fait, les RTA peuvent être employés uniquement pour spécifier une syntaxe de grammaire contextuelle ou hors contexte durant l'interaction multimodale (voir l'exemple de la Figure 4.)

La gestion de ces interactions est réalisée de façon externe et les différents états du système peuvent être déterminés à partir des variables dans le code uniquement.

Il est par ailleurs toujours possible de modéliser les interactions dans les architectures multimodales par les outils offerts par UML (Unified Modeling Langage (http://www.rational.com/uml/index.jsp 2006).) Certains de ces outils sont présentés ci-après (Muller 1997).

Les diagrammes de collaboration décrivent des interactions entre objets en insistant sur la structure spatiale statique permettant la mise en collaboration d'un groupe d'objets. L'interaction est représentée par l'envoi de messages. Les spécifications des séquences, niveau d'emboîtement de l'envoi de messages, et de leurs synchronisations sont possibles.

Les diagrammes de séquence montrent les interactions entre objets selon le point de vue temporel. Le contexte des objets n'est pas représenté de manière explicite comme dans

[12] L'annexe 2 présente les Réseaux de Petri.

les diagrammes de collaboration. Le temps n'est pas pris en compte de manière quantitative.

Les diagrammes d'états transitions représentent des automates à états finis, du point de vue des états et des événements.

Les diagrammes d'activité sont une variante des diagrammes d'états transitions par rapport aux actions. Ils permettent de réaliser le comportement interne d'une méthode ou d'un cas d'utilisation uniquement via le stéréotype d'activité. Ce comportement est modélisé sous la forme d'étapes regroupées séquentiellement dans des branches parallèles de flots de contrôle. Les synchronisations entre les flots de contrôle sont réalisées au moyen du stéréotype de barres de synchronisation. Ils permettent d'ouvrir et de fermer des branches parallèles au sein d'un flot d'exécution d'une méthode ou d'un cas d'utilisation.

Les places dans les réseaux de Petri colorés peuvent modéliser des états, des files d'attentes ou des ressources contenant des symboles (marques ou jetons.) Ces symboles peuvent être retirés de ces places et d'autres symboles peuvent être générés dans ces mêmes places ou dans d'autres. Les diagrammes d'états transitions ne permettent pas de modéliser conjointement des symboles (données ou objets) et leurs réceptacles (files d'attente, etc.), comme le font les réseaux de Petri. Ils ne permettent pas de modéliser à la fois la structure architecturale d'un système (réceptacles : ressources, files d'attentes, etc.) son contenu (symboles : objets, variables, etc.), ainsi que ses différents états et événements produisant ces états dans la structure de l'architecture. Les diagrammes d'états transitions réalisent uniquement la modélisation des états et des événements produisant ces états. Les systèmes à automates finis comme les réseaux de Petri colorés temporisés et stochastiques (Jensen 1995; Jensen 1997-a; Jensen 1997-b) s'avéreraient donc plus pertinents du fait qu'ils sont très bien adaptés aux protocoles de communications statiques et surtout dynamiques présents dans les applications multimodales. Ils permettent ainsi d'observer les interactions événementielles conjointes en tenant compte des contextes structurel, spatial, temporel et événementiel.

Parmi toutes ces méthodes de spécification, nous privilégierons donc les Réseaux de Petri Colorés Temporisés (RPCT) dont nous validerons l'applicabilité dans le domaine de la modélisation des dialogues multimodaux par des exemples de modélisations. Ces réseaux permettent l'élaboration de modèles très complexes intégrant à la fois une approche centrée 'individu' et/ou centrée système tout en s'affranchissant d'une étape de programmation généralement très lourde et en restant dans le cadre d'un formalisme bien défini.

Les travaux de doctorat de Bastide (Bastide 1992) (coauteur de la référence (Accot 1996)) sur les réseaux de Petri ont consistés à étudier le formalisme du concept 'objet dans le réseau de Petri' versus le concept ' réseau de Petri dans l'objet'. Notons que cet auteur ne fait aucunement référence au concept d'architecture logicielle dans ses travaux de doctorat. Par la suite, la modélisation des événements multimodaux a déjà été réalisée par des réseaux de Petri de haut niveau dans le cadre d'une application dédiée à l'interaction à deux mains (employant simultanément une souris et une boule de commande[13] (Accot 1996).)

Cependant, l'application (Accot 1996) est non générique. Par ailleurs, les autres modélisations d'interaction multimodales par réseaux de Petri de ce auteur et ses collègues (Navarre 2002) ne font pas appel aux concepts d'AL et de reconfiguration architecturales qui sont nos principaux propos. Notre approche emploiera à la différence des travaux de Navare, Palanque et Bastide (Navarre 2002), les RPCT pour spécifier le dialogue générique d'une *architecture logicielle* multimodale qui sera elle-même reconfigurable. Puis, nous proposerons une approche de reconfiguration dynamique de cette AL multimodale. Ceci ne constitue en aucun cas le propos des travaux de cet auteur et de son équipe de recherche sur les réseaux de Petri appliqués à la problématique de la multimodalité et de l'interaction multimodale comme le montre la référence (Bastide 1995).

[13] Traduction de 'Trackball.'

2.1.5 Conclusion

L'intérêt des espaces de conception et des taxonomies est qu'ils peuvent être utilisés, a priori, pour guider les choix de modélisation. L'espace de conception peut se rapporter à des niveaux d'abstraction (niveau signal, niveau sémantique, etc.) auxquels correspondent des décisions et des choix d'architectures logicielles.

Il faudra ensuite employer une modélisation et une spécification adaptées au problème traité. Dans le cadre de notre approche, nous avons donné au paragraphe 2.1.4 des arguments qui justifient la spécification du dialogue multimodal par les RPCT.

En guise de synthèse, nous résumons les possibilités offertes par les différentes approches, outils ou méthodes de spécification, quant aux possibilités de description des propriétés et caractéristiques qui nous semblent fondamentales dans un dialogue multimodal, par le Tableau III. Les RPCT, que nous employons, sont des réseaux de haut niveau intégrant les concepts de hiérarchie, de temps, etc. Nous n'avons pas présenté dans ce Tableau toutes les méthodes de spécifications qui existent en génie logiciel. Seules, les méthodes employées pour la multimodalité dans la littérature, et celles évoquées au paragraphe 2.1.4, ont été mentionnées dans ce Tableau. Il est évident que certaines méthodes sont plus avantageuses que d'autres lorsqu'il s'agit de faire un choix, en fonction des besoins de spécification d'un système et quand il faut se positionner à des niveaux d'abstractions pertinents pour une architecture donnée. Les frontières entre ces niveaux d'abstractions doivent être clairement définies. Les liens entre ces niveaux définissent des implications sur les décisions concernant les propositions d'architectures qui modélisent la fusion multimodale en entrée et la présentation de l'information multimodale en sortie. Le Tableau III justifie notre choix des RPCT.

Tableau III

Comparaison des formalismes de spécification du dialogue multimodal

Propriétés et caractéristiques permises ou offerte par la spécification / formalismes pour la spécification de l'interaction des événements multimodaux		Prise en compte des paramètres temporels pour le dialogue multimodal :			Langage formel			Représentation :										Gratuité des outils
		dater les événements	trier l'arrivée des événements	durée de traitement des événements	oui	non	+ logique formelle	des données échangées dans le dialogue (objets symboles, messages, …)	des supports et des structures de données (files d'attentes ressources canaux, localités, …) et de leurs répartitions et interconnexions.	des aspects dynamiques du dialogue :				distri-buée:		graphique intuitive	hiérarchisée	
										Communication		Parallélisme		fonctionnelle	spatiale			
										asynchrone	synchrone	déterministe	non-déterministe					
Algèbres des processus[14] (Xavier 1992)	CSP				X		X	X			X	X						X
	CCS				X		X	X			X	X						X
	LOTOS				X		X	X			X	X				X	X	X
UML	diagrammes de collaboration					X		X	X						X	X		
	diagrammes de séquence	X	X	X		X				X	X	X				X		
	diagrammes d'états transitions			X		X		X		X	X	X				X	X	
	diagrammes d'activité					X				X	X		X			X		

[14] Il existe des variantes temporelles des algèbres de processus. Par exemples : '*Les algèbres TeCCS, TiCCS et U-LOTOS sont définies relativement à un domaine temporel réel. Les algèbres TCSP et ACP sont définies explicitement sur un domaine temporel dense (les réels ou les rationnels.) la syntaxe des algèbres TPCCS et de TPL impose, comme celle d'ATP, l'utilisation d'un domaine temporel discret (les entiers naturels.)'* Encore faut-il qu'elles permettent de prendre en compte les aspects temporels du Tableau III et que des outils logiciels soient disponibles.

Tableau III

(suite)

Z				X		X	X	X								X	X
RTA (Krus 1993)							X	X	X		X				X		
RPCT (Jensen 1995; Jensen 1997-a; Jensen 1997-b)	X	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X
Structures des caractéristiques typées					X	X	X										
Le langage de spécification des applications multimodales				X		X	X										
Technique du creuset	X	X	X	X			X	X	X	X	X						
Structures d'actions partielles				X			X	X									
Représentation Syntaxique				X			X										

2.2 Les types d'architectures pour les applications multimodales

Il est nécessaire de présenter les définitions et les caractéristiques de l'architecture en génie logiciel et de monter leurs importances relativement à notre approche. Pour ce faire, nous donnons ici une analyse de la littérature qui met en évidence les aspects permettant de définir et caractériser au mieux l'Architecture Logicielle (AL) dans le cadre des applications multimédia multimodales (paragraphe 2.2.1.)

Puis, nous exposons, au paragraphe 2.2.2 un aperçu synthétique des architectures pour l'intégration multimodale. Cet aperçu est également jumelé à une analyse qui précise leurs forces et faiblesses et définit ainsi les aspects et les points importants relatifs à notre approche.

2.2.1 L'architecture logicielle

2.2.1.1 Introduction

Il n'existe pas de consensus unanime pour la définition d'une AL. Plusieurs définitions sont proposées dans la littérature. Ces définitions se rejoignent et se complètent pour affirmer que la conception d'une architecture d'un logiciel a pour objet de donner à un effort de développement, du dit logiciel, les moyens de se réaliser. Plus spécifiquement, l'AL se focalise principalement sur la forme globale d'une application. 'Cette forme est mise en évidence lors de la conception et va au delà des structures des données et des algorithmes' (Garland 1993). En effet, l'AL se préoccupe de l'intégrité, de l'uniformité de la simplicité et de l'esthétisme.

Il n'existe aucune architecture universelle adaptée à tous les types d'applications. La communauté scientifique ne considère pas avoir atteint une maturité suffisante dans ce domaine, malgré les percées et les résultats importants obtenus. Le sujet reste d'actualité et fait l'objet de nombreuses recherches. Les chercheurs s'accordent sur le fait qu'il est possible de concevoir des architectures réutilisables dans les limites d'un domaine particulier (un exemple serait les systèmes interactifs multimodaux.) Concevoir une architecture adaptée et adéquate serait la clé du succès d'un développement logiciel donné.

Une AL devrait donc permettre d'offrir une vision globale de l'application en décrivant les choix stratégiques qui, d'une part, déterminent des attributs de qualité du logiciel comme la fiabilité, l'adaptabilité ou la garantie d'autres performances et d'autre part laisse le champ libre aux décisions tactiques des développeurs.

Traditionnellement, le développement de logiciels est conduit par la nécessité de répondre aux besoins de leurs utilisateurs. Cependant, beaucoup de personnes autres que les utilisateurs finaux sont intimement concernés et impliqués dans le processus de développement de l'AL (développeurs, testeurs, opérateurs, travailleurs de la maintenance, réparateurs, financeurs, etc.) Toutes ces personnes sont parties prenantes

du processus de réalisation de l'AL. Le terme anglo-saxon qui leurs est consacré dans la norme ANSI/IEEE Std-1471-2000 (http://www.ieee.org/web/standards/home/index.html 2006) est '*stakeholders*[15]'. Une bonne architecture est par conséquent celle qui répond aux objectifs, buts, intérêts et besoins des '*intervenants'*. L'AL sera accompagnée d'une description architecturale qui démontrera à ces derniers que leurs objectifs sont atteints. Cette description architecturale constitue la documentation de l'AL.

Les sections qui suivent présentent les principales définitions de l'AL, depuis l'avènement de ce thème de recherche dans les milieux académiques[16] au début des années 90 jusqu'à nos jours. Une analyse pertinente est proposée afin de mettre en évidence l'intérêt de ces définitions pour les systèmes interactifs multimodaux.

2.2.1.2 Les fondements de l'AL

Perry et Wolf sont les fondateurs de l'AL en tant que discipline à part entière du génie logiciel. Leur définition de l'AL (Perry 1992) s'inspire de la définition de l'architecture des édifices et constructions (immeubles, bâtisses, etc.) employée en génie civil. Ils proposent un modèle représenté par l'ensemble suivant :

AL = {éléments architecturaux, forme architecturale, motivations de l'architecte}

Les éléments architecturaux sont soit des éléments de données, soit des éléments de transformation des données ou encore les éléments de connexion entre les éléments précédents.

La forme architecturale correspond à la fois à des propriétés et à des relations. Propriétés et relations définissent respectivement des contraintes sur les éléments architecturaux

[15] Nous choisissons le terme *intervenants* comme traduction du terme anglo-saxon *stakeholders* dans le texte car la traduction *partie prenante* concerne le cadre legislatif et non celui du génie logiciel.

[16] Selon la référence (Bosh 200), il y a une différence entre la perception académique et la perception industrielle de l'AL.

pris individuellement et des contraintes sur leurs dispositions conjointes. Ces contraintes sont pondérées et classées selon leur importance.

Les motivations économiques (coût, temps…), techniques (performances…), conceptuelles (évolutivité…), etc. permettent à l'architecte de faire un choix pour une architecture donnée plutôt qu'une autre.

Les auteurs regroupent les caractéristiques architecturales et les décisions de conception communes en ce qu'ils appellent des styles architecturaux. Un style architectural n'est pas une architecture mais un concept qui rassemble des architectures ayant les mêmes aspects clés permettant de les définir.

Un style architectural est l'expression schématique structurelle et abstraite d'une organisation des systèmes logiciels. Il offre un ensemble de types prédéfinis, spécifie leurs responsabilités générales et inclut les grandes lignes et règles d'interrelation et d'organisation entre ces types. Le style offre un ensemble de principes organisationnels de l'intégralité du système par opposition à une description détaillée d'une de ses parties. Le style architectural comprend généralement les différents types d'éléments de l'architecture, leurs interfaces et les types de connecteurs. Le tout est organisé, combiné et relié selon des contraintes bien précises.

Les styles architecturaux sont employés durant le processus de définition de l'architecture logicielle. Un style peut constituer :

a. une solution à appliquer directement pour le design d'un système;
b. un style de base à adapter aux contraintes particulières du système à développer;
c. une aide qui peut inspirer l'architecte, si le style ne répond pas au problème à résoudre;
d. une motivation qui conduira l'architecte à créer un style nouveau;

Enfin, il est possible de combiner plusieurs styles pour offrir des solutions aux systèmes complexes comme les systèmes interactifs multimodaux.

Ils proposent également la notion de vues architecturales qui donnent des représentations de certains aspects structurels d'une même architecture. Ces représentations multiples sont séparées mais interdépendantes. Elles illustrent la façon dont l'AL répond aux intérêts et besoins des intervenants. Perry et Wolf (Perry 1992) proposent trois vues architecturales: la vue des données, la vue de transformation et la vue de connexion, relativement aux trois natures différentes des éléments architecturaux. Ils définissent l'analyse de l'AL en des spécifications permettant de fournir une documentation claire et précise sur l'architecture ainsi qu'en des analyses automatiques de cette documentation, dans le but de mettre en évidence divers problèmes. Les auteurs distinguent les analyses de cohésion des analyses de couplage.

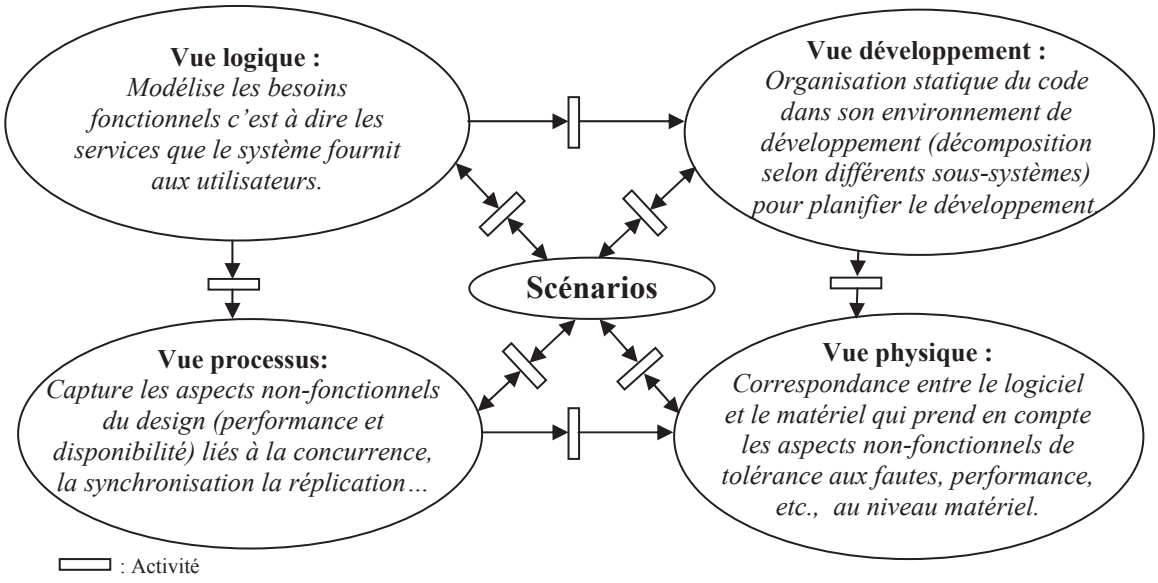

Figure 5 Les '4+1 vues' de l'architecture logicielle selon Kruchten adaptées de (Kruchten 1995) sous la forme d'un réseau de Petri (les rectangles représentent des activités et les flèches le sens du flux informationnel)

Tout en souscrivant aux concepts proposés par ces fondateurs de l'AL, nous nous sommes proposés d'élargir leur définition en y incluant des 'vues plurielles' qui regroupent deux ou trois des vues proposées par Perry et Wolf.

Nous pensons que pour certaines applications une vue conjointe des données et de leurs transformations et/ou de leurs connexions peut apporter une vision utile aux concepteurs de l'application (dans le cas d'architecture du dialogue multimodal et/ou lorsque la reconfiguration architecturale dynamique est de mise.) Nous rejoignons ainsi l'idée des '4+1 vues' de Kruchten (Kruchten 1995) qui utilise une cinquième vue pour regrouper les quatre autres qu'il définit.

Cette vue conjointe permet d'illustrer le fonctionnement de l'architecture à partir de scénarios lors de la phase de conception (Figure 5.)

Elle offre aussi une base pour la mise en place des tests de validation de l'AL. Sur chacune des vues de la Figure 5, le modèle de l'AL de Perry et Wolf peut s'appliquer avec une notation et une syntaxe particulière donnant lieu à différents 'styles architecturaux.' L'adaptation sous forme de RP des '4+1 vues' à la Figure 5 permet de mettre en évidence les liens dynamiques qu'entretiennent ces différentes vues.

En second lieu, nous considérons que l'intérêt principal de l'architecture que nous nous proposons d'élaborer ne doit pas servir uniquement à la documentation, comme dans le cas des architectures existantes, mais doit également permettre d'effectuer des analyses automatisées. Cet aspect fera l'objet de la partie validation de notre approche lorsque nous proposerons des scénarios d'interactions multimodales que nous simulerons par l'outil CPN-Tools (University of Aarhus 2006).)

2.2.1.3 Vision de l'AL selon Garlan et Perry

En se basant sur des travaux existants, Garlan et Perry proposèrent en 1995 un synthèse qui donne une vision de l'AL en termes de structure de composants d'un programme/système, de leurs interrelations, de leurs principes et de leurs lignes de

conduites gouvernant leurs évolutions dans le temps (Garlan 1995). L'AL, telle que décrite par les auteurs, possède les caractéristiques suivantes :

a. elle fournit un cadre formel, des notations et des outils pour l'étude de l'organisation des structures de haut niveau d'un système;
b. elle offre une description modulaire, à base de composants et connecteurs, qui permet de réaliser de l'implémentation modulaire (construire un module à partir des services d'autres modules) et de l'interaction modulaire (définir des protocoles de communication entre les modules);
c. elle offre une alternative aux méthodes de conception classiques car elle permet d'offrir aux concepteurs,à la fois, (Figure 6):

    i. un cadre pour résoudre certains types de problèmes posés par les besoins tout en s'affranchissant de l'implémentation du code source : algorithmique et structures des données;
    ii. un cadre pour faire ressortir ces besoins au niveau implémentation.

La Figure 6[17] présente de façon symbolique la différence entre les méthodes de conception classiques et celle incluant une AL structurée et organisée en modules ayant des éléments en commun.

Les propositions de Garlan et Perry montrent que l'AL permet de répondre à des questions qui lui sont propres (relatives aux composants et connecteurs.) Ainsi, elle ne se place pas au plan de l'algorithmique et des structures de données mais fournit des choix et des compromis architecturaux pour résoudre certains types de besoins. La modularité que peut apporter une AL est pertinente pour les applications multimodales. Coome le montre la Figure 6, nous considérons que, disposer d'une structure hiérarchique

[17] Comme le(a) lecteur(rice) a pu le remarquer, les figures de ce document, décrivant un concept ou une abstraction, sont, dans la mesure du possible, représentées sous forme de réseau de Petri, par souci d'homogénéité envers un même formalisme, mais aussi car nous pensons qu'une représentation qui met en

modulaire, à base de composants, permettra de cloisonner les besoins des applications multimodales. L'AL devra également décrire l'interaction entre les composants. Ceci permettra d'élaborer les scénarios de reconfigurations architecturales à un niveau d'abstraction élevé (par modification des connexions entre composants par exemple), sans se soucier de ce qu'il y a à l'intérieur des composants.

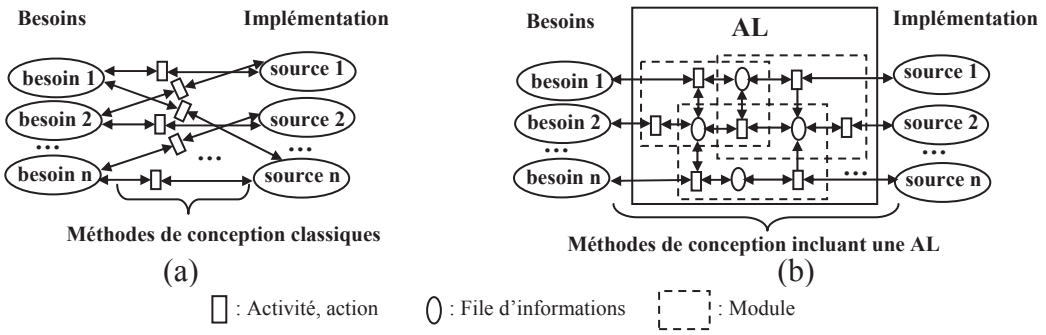

Figure 6 Distinction symbolique entre : (a) Méthodes de conception classiques et (b) AL

À la différence des auteurs, nous considérons que les méthodes classiques tout comme celles employant l'AL ont la même finalité: cerner les besoins tout en faisant en sorte de permettre de les inclure dans l'implémentation du code. La nuance provient du fait que les besoins pris en compte par l'AL diffèrent de ceux traités par les méthodes classiques. Ainsi, les procédures et approches pour résoudre les problèmes posés par ses besoins le sont aussi.

2.2.1.4 Prise en compte des critères de choix de l'AL

La proposition de Gacek et al. (Gacek 1995) considère que les définitions proposées pour l'AL concernent la représentation et la description de structures. Elle souligne que les critères de choix qui permettent de décider si une structure est une AL ne sont pas

---

évidence des activités, des ressources et les résultats dans des files d'informations de ces activités est plus pertinente pour le lecteur.

suffisamment détaillés dans ces définitions. Ils recommandent de prendre en considération ces critères aussi bien que les structures de l'architecture.

> *«A software system architecture comprises:*
> *- A collection of software and system components, connections and constraints.*
> *- A collection of system stakeholders' need statements.*
> *- A rationale which demonstrates that the components, connections and constraints define a system that, if implemented, would satisfy the collection of system stakeholders' need statement. »* (Gacek 1995)

Nous adhérons au fait qu'une approche 'AL' doit permettre de considérer les besoins des acteurs et les critères de choix de l'architecture. Ceci est tout à fait pertinent si nous souhaitons qu'une architecture soit générique au sens de réutilisable.
Les auteurs proposent alors d'introduire des possibilités de vues multiples pour répondre aux besoins des acteurs. L'intérêt des vues architecturales multiples sera également d'à propos, lorsqu'il s'agira de faire des propositions nouvelles pour les systèmes multimodaux. Il faudra cependant définir les différents acteurs leurs besoins et des critères de choix pour ces architectures multimodales.
Un autre aspect important soulevé par Gacek & al. dans (Gacek 1995), est le point de vue architectural centré sur le processus du cycle de vie. Cet aspect est important et aidera à la conduite de modifications lors de la phase de conception, sans avoir à affecter les performances du logiciel (au contraire elles seront maintenues ou améliorées.)

2.2.1.5 Structure ou structures d'un système

En 1998, Bass et ses coauteurs définissent l'AL dans un contexte de pratique industrielle (Bass 1998) comme étant '*la structure ou les structures d'un système*' qui incluent des composants logiciels, '*leurs propriétés visibles externes*' et les relations que ces composants entretiennent. Dans leur ouvrage, ils présentent l'AL comme une '*abstraction de la réalité'* qui décrit principalement l'interaction entre les composants. Nous cautionnons leur définition mais nous admettons que l'abstraction offerte par une AL ne concerne pas forcément l'interaction entre les composants. L'architecte doit avoir

le champ libre pour prendre en compte tous les aspects architecturaux qu'il souhaite mettre en valeur en fonction des besoins.

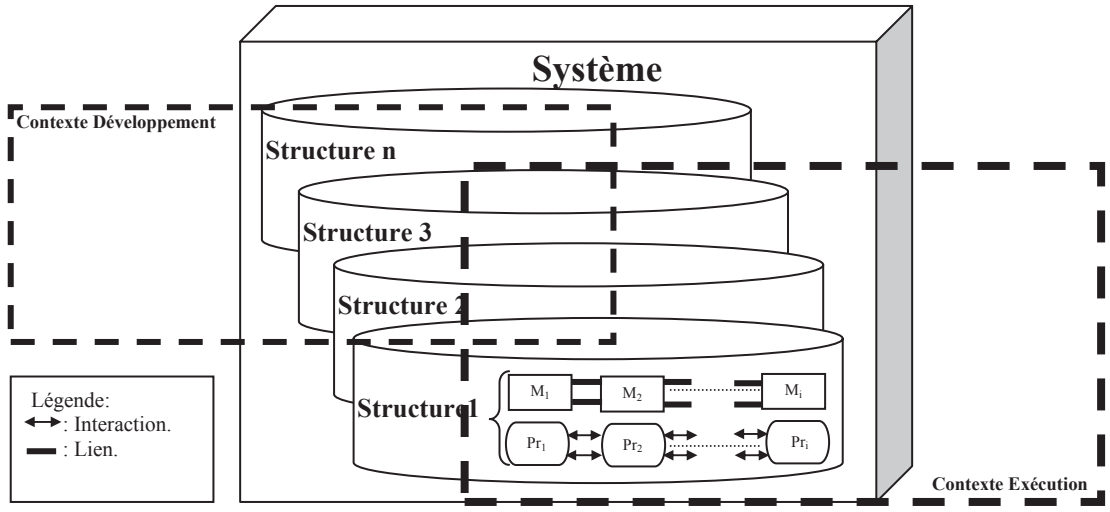

Figure 7 Vue symbolique de l'AL selon Bass et al. (M_i: module i, Pr_i: processus i)

Leurs travaux différencient les '*vues multiples'* des '*structures multiples'* : à une structure donnée peut correspondre plusieurs vues. Des vues multiples permettent de présenter sous différents angles une même structure. La structure est une configuration particulière de différents éléments de l'architecture (composants, etc.)

Structures et vues ont pour finalité l'aide à la conception, à la compréhension et aux analyses architecturales. Ils insistent sur le fait qu'un système peut comporter plusieurs structures, sans qu'aucune d'entre elles ne constitue pour autant une définition de l'AL.

Dans le même ordre d'idée, ils expliquent qu'il existe plusieurs types de '*composants'* ('*modules'* et '*processus'*), qui peuvent réaliser plusieurs types '*d'interactions'* ('*subdivisions*, *synchronisations'*), dans divers '*contextes'* (voir Figure 7.)

Selon Bass et ses coauteurs, le comportement de chaque composant permet de déterminer l'interaction des autres composants susceptibles d'interagir avec lui.

Comme les auteurs allèguent que ce comportement fait partie intégrante de l'AL, ils avancent alors qu'une 'boxologie' informelle (employant des boîtes reliées par des lignes) ne constitue pas une AL si elle ne s'accompagne pas des descriptions comportementales des éléments qui la composent.

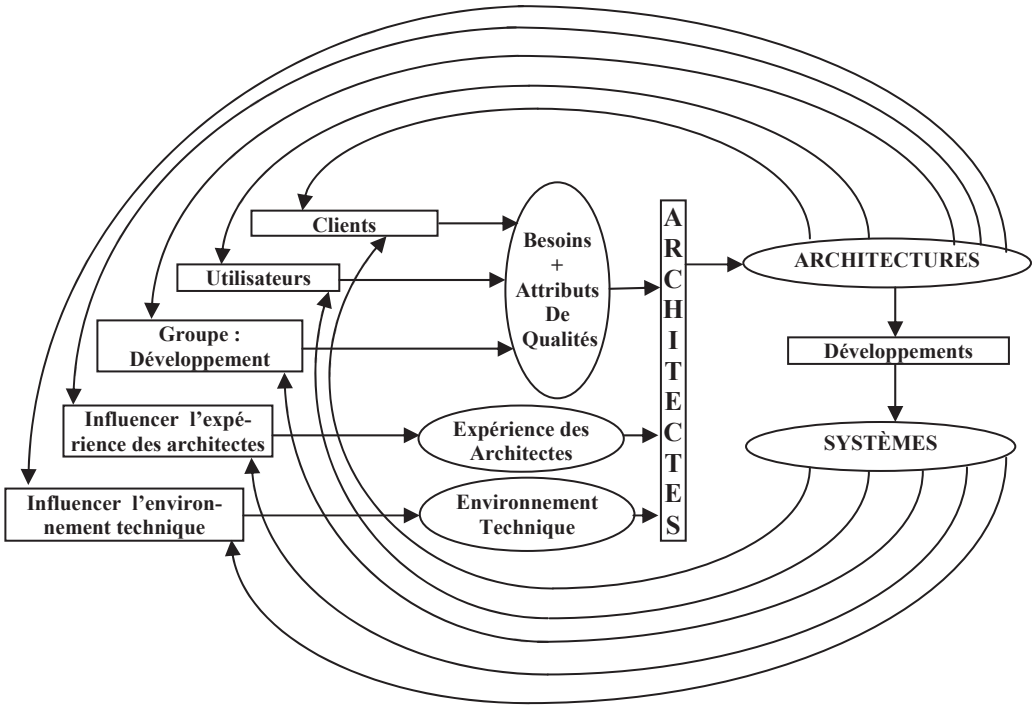

Figure 8 '*Architecture Business Cycle*' sous forme de réseau de Petri

Pour notre part, nous considérons aussi qu'une description formelle des comportements et des interactions entre les composants est fondamentale. Cependant, nous ne dénions pas pour autant que des descriptions détaillant moins ces aspects, ne fassent pas partie intégrante de l'AL. Au contraire, elles pourront en faire partie car ce sont les besoins qui en détermineront le degré de précision.

Ils définissent aussi '*l'Architecture Business Cycle*' (Figure 8.) C'est un cycle décrivant les facteurs qui influencent l'architecte et où, ces mêmes facteurs, sont à leurs tours

influencés par l'architecture. Nous identifierons des éléments de ce cycle dans le contexte des architectures multimodales, en précisant quels attributs de qualités nous souhaitons mettre en avant dans le cadre de notre approche.

Un autre aspect qui nous semble important et qui a été explicité par ces chercheurs, concerne les méthodes permettant l'analyse d'une AL pour déterminer si elle répond bien aux besoins des concepteurs et des acteurs d'un système. Pour ce faire, ils détaillent, dans une méthodologie appelée ATAM (*Architecture Tradeoff Analysis Method* (Bass 2001)), les étapes à suivre pour analyser et déterminer les compromis à faire dans une AL, si cette dernière doit vérifier plusieurs attributs de qualité. Même si cette méthode est dédiée aux gros systèmes d'informations ('*intensive systems*') ses principes restent applicables pour les applications multimodales qui font intervenir différents types de données et utilisateurs.

2.2.1.6 CORBA

Common Object Request Broker Architecture (CORBA) (Vinoski 1997) est un ensemble de concepts architecturaux qui sont le fait d'un effort de standardisation de la conception orientée objet pour les environnements hétérogènes distribués et pour la communication dans ces environnements de traitements informatiques. CORBA vise un plus vaste ensemble d'applications ('MiddleWare') que les applications multimodales. Cependant, comme la programmation orientée objet apparaît extrêmement adéquate pour les systèmes multimodaux hautement modulaires, les futures applications multimodales devraient de plus en plus faire appel à une implémentation CORBA. Il n'y a pas, à notre connaissance, d'architecture multimodale CORBA.

2.2.1.7 Synthèse pour notre approche

En résumé, même s'il n'y a pas de consensus unanime pour la définition de l'AL, il apparaît que les chercheurs sont plus ou moins en accord sur ses grandes lignes.

La norme (http://www.ieee.org/web/standards/home/index.html 2006) ANSI/IEEE Std-1471-2000 qui offre des recommandations pour la description architecturale des gros systèmes, est un exemple d'effort réalisé pour tenter de réunir les approches et d'offrir un standard de l'AL. Cette diversité des définitions n'est pas gênante pour notre approche car nous voulons définir une AL qui se limite aux problèmes et besoins posés par les applications multimédia multimodales. Cette liberté de choix d'une définition de l'AL pour les systèmes interactifs multimodaux est plutôt un avantage.

Nous proposerons une définition de l'AL dédiée aux applications multimédia multimodales à partir des définitions précédentes (dans le chapitre suivant.) Mais préalablement, nous allons présenter une revue des propositions de la littérature sur les architectures multimédia multimodales.

2.2.2 Architectures multimodales

Les architectures multimodales doivent vérifier des qualités et principes fonctionnels, en plus des qualités architecturales communes aux interfaces standard. Ces qualités et principes fonctionnels sont définis dans (Hill 1992) comme suit:

a. modalités combinées : l'utilisateur doit pouvoir utiliser simultanément toute combinaison possible de modalités qu'il souhaite employer;
b. inclusion de l'ambiguïté : le système doit être capable de prendre en charge tout emploi ambigu des modalités d'entrée;
c. protocole de coopération : l'utilisateur doit être capable d'interrompre les entrées et les sorties en tout temps;
d. accès total à l'historique de l'interaction en ligne ou après avoir fini une session;
e. évolutivité : les interfaces doivent pouvoir être évolutives durant les interactions.

Par ailleurs, un aspect important à prendre en compte du point de vue architectural est la distribution qui peut être spatiale et/ou fonctionnelle (Tabeling 2002) pour ce type d'application.
La littérature offre un ensemble d'alternatives architecturales résumées par les exemples ci-après.

2.2.2.1 PAC-Amodeus

Cette architecture (Nigay 1993; Nigay 1994) supporte le mécanisme de fusion générique du creuset[18]. Cette architecture permet une plateforme globale réutilisable et applicable au développement de toute application multimodale supportant la coopération synergique parallèle des modalités (et par conséquent, toutes les autres, comme le montre la Figure 9.)

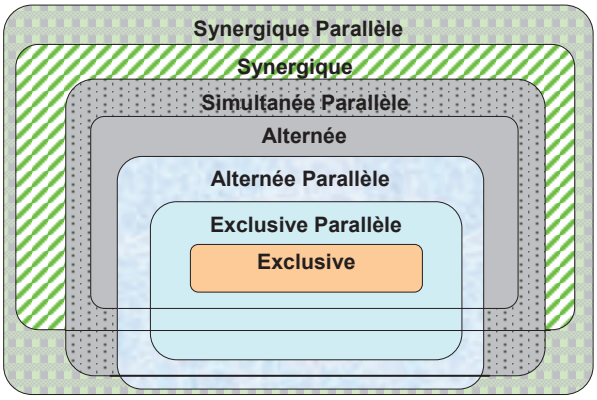

Figure 9 Relations d'inclusion des multimodalités selon Bellik (Bellik 1995)

Le cœur du composant PAC-Amodeus est le Contrôleur de Dialogue – un ensemble d'agents coopérant qui réalise le parallélisme et le traitement de l'information à plusieurs niveaux d'abstraction. Cependant, comme l'a mentionné Bellik dans (Bellik

[18] Voir paragraphe 2.1.4.6.

1995), cette approche ne définit pas clairement le rôle des agents. Il ajoute que son caractère trop général fait qu'en l'absence d'une méthode précise de décomposition, l'identification des agents et des rôles qui doivent leurs être assignés reste un problème important.Par ailleurs, la spécification par la méthode du creuset ne permet pas une validation automatique des scénarios de dialogue avant la phase d'implémentation du prototype.

2.2.2.2 Open Agent Architecture (OAA)

OAA (Moran 1997) est offerte en tant que méthodologie de programmation qui donne accès à des applications multiagent via des interfaces utilisateur multiagent intelligentes, coopératives et distribuées. Elle supporte le langage parlé, l'écriture manuelle, les gestes 2D en plus des modalités d'entrée standard (clavier et souris.) Dans ce type d'architecture, seule la première interface utilisateur multiagent, doit s'exécuter sur un ordinateur local. De ce fait, cette architecture autorise les applications multimodales mobiles comme les assistant personnels numériques (*PDA personal digital assistant*.) L'OAA est une architecture qui a été utilisée pour le développement d'applications multimodales, incluant les assistants de bureau, les guides touristiques basés sur des cartes interactives, l'information sur les vols offerts par les compagnies d'aviation, le contrôle de multi-robots et les systèmes à réponse émergente de l'intelligence artificielle, démontrant ainsi sa caractéristique fondamentale de 'réutilisabilité'. Cependant cette architecture est dédiée à des types de modalité prédéfinis et ne constitue pas une méthodologie de spécification générique mais offre la possibilité de réutilisation dans certains cas précis de blocs déjà préprogrammés et les méthodes pour les relier.

2.2.2.3 MIAMI PVM

C'est une architecture (Schomaker 1995) qui supporte l'aspect multi-tâches avec le package logiciel Tcl/Tk[19] du domaine public PVM (Parallel Virtual Machine). L'aspect multi-tâches est crucial dans les applications multimodales. Nous pouvons reprocher à cette architecture le fait qu'elle soit spécifique à un package logiciel particulier. Par ailleurs, elle n'offre pas d'outil de spécification logicielle.

2.2.2.4 Quick Set

Quick Set est un panel d'outils basé sur une architecture d'agents collaborateurs agissant dans un système multimodal sans fil ('wireless') permettant à plusieurs utilisateurs de réaliser et contrôler des simulations militaires par l'utilisation de la voix, du geste et en attribuant aux utilisateurs (qui sont considérés comme des entités), des comportements et des rôles. Cette application, offre une architecture centralisée (basée sur le concept du Tableau noir : '*blackboard*[20']) d'agents collaborateurs. (Cohen 1997; Cohen 1997; http://www.cse.ogi.edu/CHCC/QuickSet/mainProj.html 2006) Cependant, étant une application militaire, les auteurs donnent peu d'information sur leur architecture et sur la façon dont est spécifié le dialogue.

2.2.2.5 SPECIMEN

Cette Architecture, proposée par Bellik (Bellik 1995), est représentée par la Figure 10 ci-après. Sur cette figure les petites ellipses représentent des files d'informations. L'architecture est construite autour d'un contrôleur de dialogue modélisé par un réseau de transitions augmentées. Il communique en amont avec un ensemble de blocs qui

---

[19] Tcl/Tk regroupe la boîte à outils Tk (Tool Kit) et le langage Tcl (Tool Command Language) qui permet d'instancier les éléments de Tk ; Tcl/Tk permet de développer des interfaces graphiques.

[20] Voir le paragraphe 2.3.2 de ce document.

permettent le cheminement du dialogue dans le réseau. Ces blocs réalisent la fusion d'événements en provenance de différents média grâce à des agents.

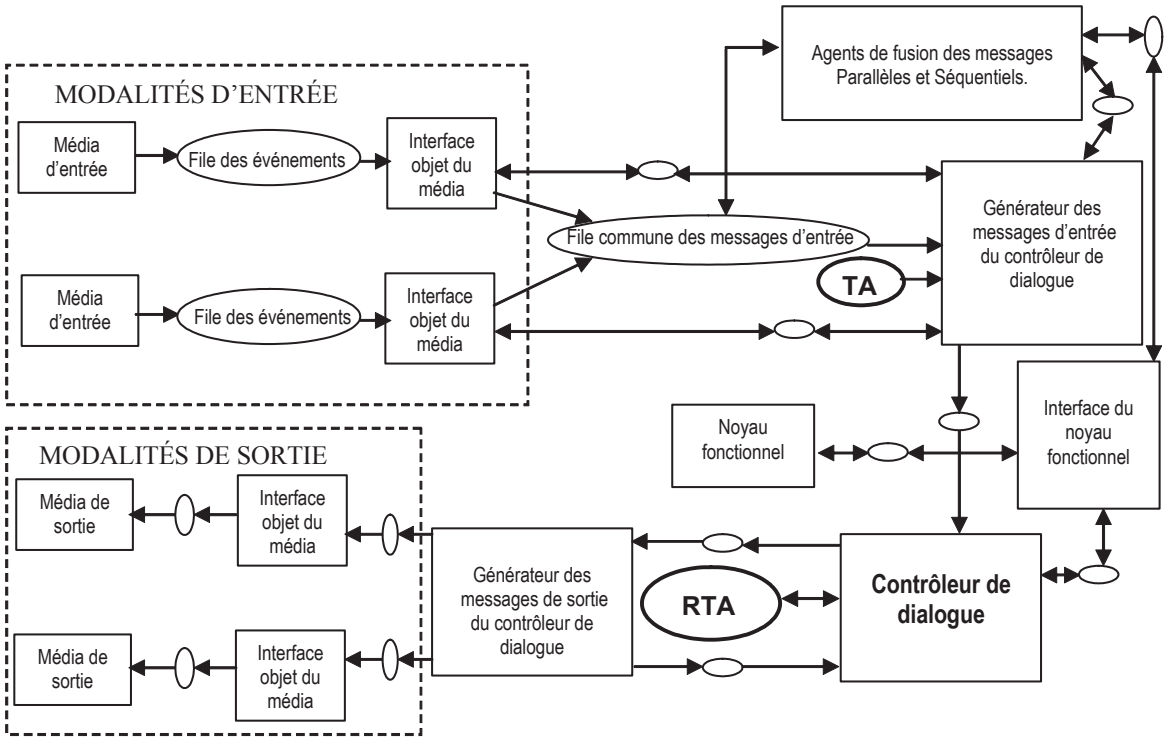

Figure 10 Architecture de SPECIMEN (adaptée de (Bellik 1995) sous forme de réseau de Petri) où le contrôleur de dialogue est modélisé par un réseau de transitions augmentées (RTA) et où une Table d'Aiguillage (TA) est employée

Dans cette architecture, la gestion des interactions avec l'utilisateur est toujours faite à l'extérieur du 'contrôleur de dialogue,' par le 'noyau fonctionnel[21]' de l'application (code dans un langage de programmation évolué comme C++, par exemple.) Par conséquent, les variables indicatrices des différents états (indicateurs du temps, etc.) du système se trouvent noyées dans le code (Bellik 1995) et n'apparaissent pas dans la

[21] Le noyau fonctionnel est la partie du code qui contient les données et les fonctions spécifiques de l'application. Celles-ci sont employées pour la gestion et l'organisation des tâches que doit réaliser un logiciel en fonction des états du système.

spécification de l'architecture. Nous pouvons aussi noter que, la grammaire qui permet de définir les énoncés sémantiques du langage multimodal, est spécifiée par le RTA de façon explicite pour générer le dialogue avec l'utilisateur en sortie.Elle est employée par la TA de manière implicite lors du processus de fusion des messages d'entrée.

2.2.2.6 Analyse et issues pour notre problématique

Dans l'idéal, les interactions d'un dialogue multimodal générique, devraient faire intervenir, aussi bien, des modalités associées au langage (parole, langage gestuel, langage écrit, etc.) que des modalités associées à l'action (désignation par le regard, le touché sur un écran tactile, un clic sur le bouton d'une souris, etc.) et ce, possiblement dans une même interface. Cette dernière s'appuierait sur un modèle (dynamique et 'reconfigurable' en temps réel) de la combinaison anthropomorphique[22] et physique des modalités. Pour ce faire, il est important de tenir compte du temps dans la gestion du dialogue multimodal. En effet, si nous souhaitons proposer une **spécification générique** de ce dialogue, il faut que cette spécification permette de tenir compte conjointement de la granularité temporelle (comme la datation des événements multimodaux, par exemple) et de la grammaire employée pour organiser ces évènements selon une sémantique (la sémantique sera ainsi elle-même sujette au temps.) Bellik (Bellik 1995) a présenté les problèmes relatifs au temps lors de la génération d'un énoncé multimodal et ce, par des exemples extrêmement clairs. Il a démontré ainsi son importance pour la compréhension d'un énoncé multimodal par certains systèmes. Son architecture (Figure 10) n'offre cependant pas une vue du dialogue qui spécifie conjointement la gestion du temps et celle de la grammaire (employée pour représenter un énoncé multimodal) dans un même modèle. Sa proposition de l'architecture SPECIMEN a consisté, au contraire, à offrir un modèle original hybride basé sur les RTA modélisant un dialogue basé sur une grammaire et sur un système de gestion des messages par des agents employant une Table d'Aiguillage (TA.)

[22] C'est à dire relative aux comportements et/ou aux attributs humains.

Nous estimons que le rôle d'une AL générique est d'offrir des vues documentées par des spécifications et qu'il est préférable d'employer une spécification de l'interaction multimodale avec un outil autre que les RTA, si nous désirons permettre à l'architecture du dialogue multimodal, de tenir compte à la fois du temps et d'une grammaire. Nous voulons faire en sorte que l'architecture offre explicitement une spécification de la 'grammaire temporelle' employée par le dialogue 'personne machine'. Ainsi l'architecture sera totalement générique (au sens de paramétrable au niveau temporel et grammatical à la fois) et nous pourrons l'adapter à tous les cas de configurations. Nous désirons, également, que l'architecture permette la validation automatique de scénarios de dialogue avant la phase d'implémentation et de programmation.

Nous souhaitons autant que possible, faire en sorte que l'architecture multimodale soit réutilisable selon tous les types de grammaires, quelles soient temporelles ou non, et cela quelle que soit la fusion et/ou la fission envisagées. Cet aspect nous semble extrêmement important si l'AL doit être générique.

D'autre part, il apparaît important que le type d'architecture, que nous souhaitons proposer, puisse offrir des caractéristiques relatives à la reconfiguration dynamique (dépendante de la vérification de contraintes temporelles et/ou sémantique de deux ou plusieurs évènements) pour permettre un langage 'humain machine' le plus naturel possible. Autrement dit, l'architecture doit offrir la possibilité de gérer l'adaptation de l'architecture à la vitesse avec laquelle l'utilisateur communique avec l'application (c'est à dire la vitesse avec laquelle l'utilisateur génère des évènements.)

Nous considérons que la prise en compte du temps dans la gestion des événements multimodaux se révèlera d'autant plus fondamentale au niveau architectural, si ces caractéristiques doivent se vérifier.

2.2.3 Conclusion

En résumé, nous constatons que la littérature offre, soit des solutions ad hoc dédiées à des cas précis, soit des architectures multiagent génériques, mais à des niveaux

d'abstraction élevés ou ne faisant pas une identification pertinente des agents, une description détaillées de leurs rôles et des interactions qu'ils entretiennent. Ou encore, si elles décrivent de façon détaillée l'interaction et le dialogue, ces architectures ne font pas état de la 'reconfiguration dynamique' architecturale (dans le cas de l'emploi ou de l'abandon d'une modalité, dans des situations de mobilité par exemple.) Enfin, pour certaines d'entre elles, il n'y a pas de spécification de tous les aspects importants pour la gestion multimodale générique des événements. Nous osons ne pas leurs attribuer une dénomination *d'architecture logicielle générique*, mais celles de *vues architecturales dédiées.*

Enfin, la mobilité est un aspect important des architectures multimodales et a été étudiée par Oviatt (Oviatt 2000-b) et appliqués dans différents cas par différents auteurs (Zouinar 2004; Baillie 2005-a; Baillie 2005-b). Cet aspect rejoint les systèmes multiagent mobiles décrits dans (Van Belle 2001; Serrano 2006).

En fait, il apparaît dans les exemples d'architectures présentés précédemment l'émergence de l'emploi du concept architectural d'agent intelligent (Jennings 1998-a; Weiss 1999). Ce concept est un aspect fondamental de la recherche en intelligence artificielle. Ceci explique l'engouement des concepteurs d'architectures logicielles multimodales intelligentes pour la théorie des systèmes multiagent (Oviatt 2003). La théorie des systèmes multiagent, employée pour la synthèse de vues architecturales multimodales, nécessite aussi l'emploi d'outils et de méthodes de modélisation. Certaines de ces méthodes sont présentées au paragraphe suivant, dans le cadre des interactions multimodales.

2.3 Les outils et méthodes de spécification et de prototypage des architectures multimodales

Peu d'outils de spécification et de prototypage ont été proposés pour les architectures multimodales dans la littérature.

2.3.1 Outils et approches pour les architectures multimodales

2.3.1.1 Outils de design d'Interfaces Multimodales Usagers

Cette boîte à outil (Kamio 1994) supporte un prototypage rapide des interfaces multimodales. Les objets *User Interface* peuvent être placés sur un panel et les liens entre les objets décrivent des scénarios 'plan buts' (du type : Que faire quand certains événements d'entrée surviennent?). L'outil de design génère alors un script qui dirige l'interface multimodale. Cette boîte à outil est dédiée uniquement aux applications basées sur le langage parlé.

2.3.1.2 Outils grammaticaux multimodaux

Il s'agit une boîte à outils (Vo 1997) pour le développement d'applications multimodales. Elle est composée d'un ensemble d'outils supportant les spécifications des applications multimodales employant des grammaires hors contexte (Pierrel 1987) et des transformations automatiques de ces grammaires multimodales en des fichiers de configuration nécessaires pour implémenter des applications comportant des composants multimodaux et des modules d'intégration. Des outils graphiques permettent à l'interface développée de spécifier ces grammaires par des interactions du type 'glisser et poser' réalisables dans une interface graphique utilisateur. Cette boîte à outil est limitée en type et en nombre de média. Par ailleurs, nous pouvons aussi noter que les grammaires hors contexte ne permettent pas de prendre en compte la composante temporelle qui est un paramètre fondamental dont il faut tenir compte dans les applications multimodales.

2.3.1.3 Reconfiguration architecturale dynamique

Comme précisé précédemment, un aspect important de la modélisation des applications multimodales est la reconfiguration dynamique au niveau architectural pour les systèmes mobiles. Cet aspect a été présenté dans la référence (Ramdane-Cherif 2002) qui présente une approche de reconfiguration dynamique des architectures logicielles sans lien avec les problèmes que posent les interactions multimodales mais qui se trouve être une piste intéressante que nous exploiterons dans le cadre de notre problématique de recherche, aux chapitres 4 et 5.

2.3.2 Agents et Systèmes multiagent pour les AL multimodales

2.3.2.1 Introduction

Aujourd'hui, les agents et systèmes multiagent (SMA) constituent une discipline à part entière qui permet de fournir des solutions originales lorsque des situations complexes ne peuvent être résolues par des méthodes classiques (orientées objets, etc.) Même si les méthodologies dites *agent* ne constituent pas forcément des outils de spécifications formelles, nous avons choisi de les présenter dans le paragraphe 2.3 car elles offrent des moyens adéquats et des techniques originales pour décrire les comportements complexes. Elles permettent de résoudre les questions relatives aux architectures intelligentes comme celles posées dans les systèmes multimédia multimodaux.

Les définitions et notions que nous présentons ici sont essentiellement issues de l'article de Chaib-draa (Chaib-draa 2001) sur les principes généraux des SMA. Elles résument également les travaux des grands noms de littérature sur la théorie des agents et SMA : comme Brooks (Brooks 1991-a; Brooks 1991-b) , Jennings (Jennings 1995; Jennings 1998-a; Jennings 1998-b), Sycara (Sycara 1989; Sycara 1990-a; Sycara 1990-b), etc. Comme pour les paragraphes précédents, nous tenterons, par une analyse, de mettre en

valeur ce qui nous semble pertinent pour notre approche architecturale dans le cadre de la multimodalité. Cependant, nous reprendrons au chapitre 5 et 6 de ce document une analyse plus détaillée (et adaptée à notre approche) des concepts 'agent' et 'SMA' issus de la littérature lorsque nous décrirons notre méthodologie de reconfiguration dynamique de l'AL multimodale par un agent expert et élaboré constituant lui-même un SMA.

2.3.2.2 Le paradigme Agent

Il n'y a pas de définition unique de l'Agent. La littérature offre une multitude de définitions semblables mais qui se différencient par le type d'application pour laquelle un agent est conçu. En 1995, Jennings et al. (Jennings 1995) ont proposé la définition suivante :

'*Un agent est un système informatique, situé dans un environnement, et qui agit d'une façon autonome et flexible pour atteindre les objectifs pour lesquels il a été conçu.*'

Les notions de "situation", "d'autonomie" et "flexibilité" sont détaillées comme suit :

a. *situation* : l'agent est capable d'agir sur son environnement à partir des entrées sensorielles qu'il reçoit de ce même environnement. Exemples : systèmes de contrôle de processus, systèmes embarqués, etc.;
b. *autonomie* : l'agent est capable d'agir sans l'intervention d'un tiers (humain ou agent) et contrôle ses propres actions ainsi que son état interne;
c. *flexibilité* : l'agent dans ce cas est :

   i. *capable de répondre à temps* : l'agent doit être capable de percevoir son environnement et élaborer une réponse dans les temps requis par l'application;
   ii. *proactif* : l'agent doit exhiber un comportement proactif et opportuniste (capable de prendre la bonne initiative au "bon" moment);

iii. *social* : l'agent doit être capable d'interagir avec les autres agents (entités logicielles et humaines) quand la situation l'exige afin de compléter ses tâches ou aider ces agents à accomplir les leurs.

Cette définition met en avant des propriétés clés comme l'*autonomie*, l'*action*, la *perception* et la *communication*. Cependant d'autres propriétés peuvent être attribuées aux agents selon le contexte de l'application. Chaib-draa (Chaib-draa 2001) cite en particulier la *réactivité*, la *rationalité*, l'*engagement* et l'*intention*.

Il faut également noter que cette définition est conforme à une vision de l'entité 'agent cognitif' (ou délibératif) capable de planifier et de prendre des décisions en fonction du modèle symbolique qu'il a de son environnement. Elle permet aussi de classer un agent selon une typologie prenant en compte ses attributs primaires, comme représentée à la Figure 11.

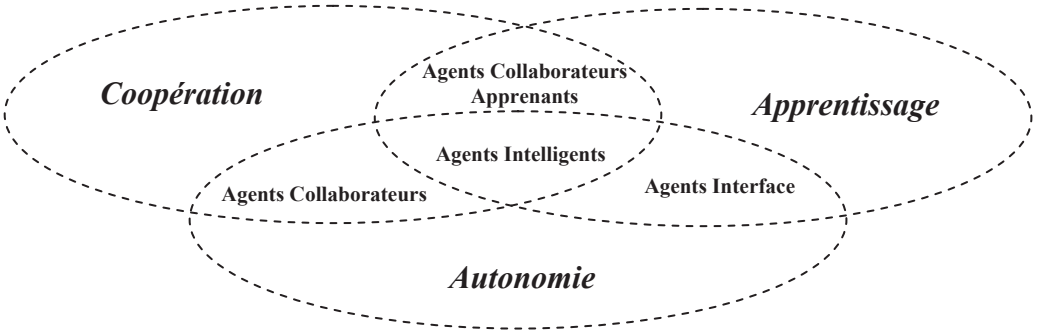

Figure 11 Vue partielle de la typologie d'un agent d'après (Nwana 1996)

À l'opposé des agents cognitifs, les agents réactifs qui n'ont pas de modèles symboliques internes de leur environnement, agissent selon un comportement élémentaire de type stimulus/réponse en fonction de l'état de l'environnement dans lequel ils sont embarqués (Ferber 1994; Nwana 1996). Même si la vision de l'agent délibératif est celle qui est la plus acceptée dans la discipline des SMA, cela ne signifie

pas pour autant qu'il faille ignorer une approche réactive. Tout dépend de ce que nous souhaitons faire.

Certaines démarches dites hybrides, combinent l'approche cognitive à l'approche réactive pour structurer un agent en niveaux (ou couches.) Les chercheurs (Chaib-draa 1996; Fischer 1999) s'accordent pour définir trois couches :

a. une couche 'bas niveau' réactive où les décisions sont prises à partir des données brutes provenant des senseurs de l'agent;
b. une couche de niveau intermédiaire où les données traitées se situent au niveau des connaissances de l'environnement;
c. une couche supérieure qui se charge des aspects sociaux de l'environnement (raisonnement tenant compte des autres agents.)

En résumé, l'intelligence artificielle (IA) a commencé à se préoccuper des composantes d'un agent autonome en les considérant de façon séparée : le but étant l'élaboration de solutions partielles par des 'résolveurs' (Chaib-draa 2001) indépendants. La réunion des composantes permettrait, par la suite, de construire un agent de façon directe. Ces préoccupations ont constitué les prémices de la discipline de l'IA distribuée (IAD.) Puis, la recherche en IA s'est focalisée sur l'aspect des interactions, entre divers comportements simples, au sein d'un même agent (approche de l'agent réactif.) Par la suite, des combinaisons réactivo-cognitives ont donné naissance au paradigme de l'agent autonome hybride.

L'emploi des agents autonomes dans les applications, est principalement réalisé via trois types d'agents :

a. l'agent interface (Maes 1994; Maes 1994) : il prend l'initiative, dans certaines circonstances, pour aider l'utilisateur dans ses tâches, tout en observant et en apprenant (Figure 11) du comportement de son interlocuteur (i.e. : l'utilisateur.)

b. l'agent mobile (Wayner 1995; Gray 1998) : il est supposé se déplacer d'un système informatique à un autre via un réseau pour exécuter, par exemple, un code et ne pas surcharger le réseau;
c. l'agent émotionnel : comme son nom l'indique, il est supposé refléter une 'émotion humaine,' dans certaines situations.

L'agent interface génèrerait ce que nous appelons un contexte 'd'intelligence ambiante[23]' qui constituerait une aide très appréciable dans le cas des interfaces multimodales (Djenidi 2004-b). L'intérêt des agents émotionnels (représentés par des visages animés parlants, par exemples) dans le cadre des applications multimédia multimodales, a été discuté dans les références (Brennan 1994; Walker 1994; Takeuchi 1995). Ces travaux montrent que l'emploi de ces agents permet d'améliorer la productivité et la performance de l'utilisateur.
La recherche en IAD a ensuite évolué vers la coordination des efforts et des connaissances, plans, buts des agents autonomes (possiblement préexistants et hétérogènes) pour agir et résoudre un problème commun. Ceci constitue la finalité des SMA présentés au paragraphe suivant.

2.3.2.3 Les systèmes multiagent

Un SMA est un système distribué composé d'un ensemble d'agents. 'Contrairement aux systèmes de l'IA, qui simulent dans une certaine mesure les capacités du raisonnement humain, les SMA sont conçus et implantés idéalement comme un ensemble d'agents interagissant, le plus souvent, selon des modes de *coopération*, de *concurrence* ou de *coexistence* (Chaib-draa 2001). Plus spécifiquement, un SMA est caractérisé par le fait que:

[23] C'est-à-dire la capacité de prendre en compte des informations provenant du contexte (localisation, historique d'interaction, autres utilisateurs, etc.) pour faciliter et enrichir l'interaction de l'utilisateur, en l'élargissant à des modalités et des objets qui peuvent rendre cette interaction plus naturelle et mieux intégrée dans l'environnement ambiant des utilisateurs.

a. chaque agent a un point de vue partiel des informations et/ou des capacités de résolution limitées des problèmes;
b. il n'y a aucun contrôle global du SMA;
c. les donnés sont décentralisées;
d. le calcul est asynchrone.

Les SMA ont donc un objectif qui pourrait être doublement pertinent pour les AL multimodales. D'une part, ils permettraient de faire une analyse théorique et expérimentale des mécanismes se produisant lorsque plusieurs entités interagissent. D'autre part, ils faciliteraient la réalisation de programmes distribués capables d'accomplir des tâches complexes via la coopération et l'interaction (communication, coopération et coordination d'actions.)

Plusieurs modèles de SMA ont été proposés dans la littérature et s'inspirent des sciences cognitives, sociales et naturelles pour générer des modèles qui 's'auto-organisent'. Dans tous ces modèles, les agents doivent faire un compromis entre l'interaction et la coopération qui se situe entre *l'antagonisme total* et la *coopération totale*. Cependant, ce compromis existe toujours dans le but de la réalisation d'un objectif commun. Pour implanter la coopération entre les agents, il faut maintenir quatre propriétés génériques importantes (Durfee 1989) :

a. il faut augmenter le taux de finalisation des tâches grâce au parallélisme;
b. il faut augmenter le nombre de tâches réalisables grâce au partage de ressources (informations, expertises, dispositifs physiques, etc.);
c. il faut augmenter les chances de finaliser des tâches en les dupliquant et en utilisant éventuellement des modes de réalisation différents;
d. Il faut diminuer les interférences entre les tâches en évitant les interactions négatives (qui conduisent à des blocages.)

Ces quatres propriétés fondementales se prêtent bien à une modélisation par les RPCTS.
Dans le cadre des applications multimodales, c'est une interaction entre agents coopératifs qui prévaudra en général pour réaliser le dialogue 'personne machine', plutôt qu'une interaction entre agents égocentrés. Cela peut s'expliquer par le fait que cette dernière est basée sur la théorie des jeux (Vlassis 2003) où l'utilité est le seul paramètre employé par les agents qui sont considérés comme omniscients (ils connaissent les actes de tous les autres agents) avec des utilités secrètes (les unes pour les autres.) Ou bien encore, cette interaction s'appuie sur les modèles de conventions collectives (employant trois types d'agents : *agent syndicat*, *agent compagnie* et *agent médiateur*) (Sycara 1989; Sycara 1990-a; Sycara 1990-b).
Néanmoins, les agents égocentrés et interagissant pour résoudre des conflits, seraient viables dans les systèmes multimodaux quand il s'agit de prendre des décisions basées sur le rôle joué par le paramètre *temps* pour réaliser une entente finale (Kraus 1991; Kraus 1995).
Dans les approches des SMA, diverses questions doivent être abordées. Il s'agit entre autres de déterminer (Ferber 1995) :

a. avec quels agents un agent doit-il coordonner ses actions?
b. quand et où ces actions de coordination doivent-elles être accomplies?
c. comment détecter et traiter les interactions entre actions (conflits et renforcement)?

Afin de résoudre ces problèmes, de nombreuses méthodologies ont été proposées. Nous résumons les principales d'entre elles par le Tableau IV. Différentes stratégies ressortent de ce tableau.
Certaines d'entre elles sont importantes pour les applications multimodales, si nous souhaitons, par exemple, faire une modélisation agent de notre architecture. Il s'agit des propositions numérotées : (1), (2), (4), (6), (7) et (9). Elles offrent des alternatives intéressantes pour les architectures multimodales statiques et dynamiques (en termes de

reconfigurations architecturales.) Les propositions (5) et (8) s'averrent aussi pertinentes dans le cadre d'une approche agent pour la reconfiguration architecturale dynamique.

Les détracteurs de la méthodologie objet peuvent avancer qu'il n'est pas nécessaire d'employer une méthodologie agent. En effet, comme les agents, les objets encapsulent leur état interne (leurs données.) Ils peuvent également poser des actions sur cet état par le biais de leurs méthodes et ils communiquent en s'envoyant des messages.

Il faut cependant faire la différence entre les concepts Agent et Objet. Ils diffèrent de part leur degré d'autonomie. Une méthode doit être invoquée par un autre objet pour pouvoir accomplir ses effets. Un agent, pour sa part, décidera de son propre gré s'il doit poser ou non une action quand il reçoit une requête. Une seconde différence provient du caractère flexible (réactif, proactif et social) du comportement d'un agent. Il est, certes, possible de faire de la programmation orientée objet qui intègre ces caractéristiques. Cependant, le modèle standard d'un objet ne donne aucune explication ou détail pour ces types de comportements.

Enfin dans une méthodologie agent, les agents sont considérés comme des sources de contrôle au sein du système alors que dans un système orienté objet, la source de contrôle est unique.

L'effort de standardisation réalisé par la FIPA[24] (http://www.fipa.org/ 2006) a permis à des chercheurs de développer des spécifications 'UML agents'(Odell 2000). Ils expliquent que l'analyse orientée agent emploie un riche ensemble de concepts qui rendent difficile la compréhension de tous les aspects du modèle d'analyse selon un seul point de vue. Il est plus simple de définir plusieurs sous modèles qui mettront en valeur, différents aspects du modèle complet.

[24] Foundation for Intelligent Physical Agents : Organisation internationale à but non lucratif dont le but est de produire des standards pour les inter-operations entre agents logiciels hétérogènes. http://www.fipa.org/

Tableau IV

Synthèse des principales approches de la communication entre agents

Approche	Auteurs-Référence-Année	Proposition - (numéro)	Principe
Coordination entre agents	Malone-(Malone 1990)-1990	Gestion de l'allocation de ressources rares et de la communication de résultats intermédiaires - (1)	*Travail des agents par petits groupes : un gros groupe est divisé en petits groupes contrôlés et coordonnés par des superviseurs selon une hiérarchie*
	Jennings-(Jennings 1995)-1995	Centralité des engagements et des conventions - (2)	*Engagements = promesses =>réaliser des actions Conventions = moyens =>suivi de ces engagements dans des circonstances changeantes.*
Négociations	Parunak-(Parunak 1996)-1996	Contrats - (3)	*Des agents gestionnaires décomposent les tâches en sous contrats qui seront attribuées aux agents contractants après négociations*
	Cammarata et al.-(Cammarata 1988)-1988	Centralisation des tâches - (4)	*La négociation permet de décider quel est l'agent le plus apte à devenir le planificateur centralisé de la tâche.*
	Sycara- (Sycara 1989)-1989	Agent Médiateur externe - (5)	*Un agent externe vient résoudre les problèmes non solubles par le SMA lui-même.*
Planification dans un environnement multiagent	Martial-(Martial 1992)-1992	Planification dirigée par les tâches et coordination de plans - (6)	*Coordination de plans pour la résolution de conflits basée sur : les divers types de relations (négatives et positives) entre les plans et sur un protocole de communication pour la synchronisation des plans.*
	Lesser-(Lesser 1991)-1991	'Functionally Accurate/Cooperative mode' - (7)	*Les agents construisent des plans et partagent leurs plans au cours de l'interaction.*
Communication entre agents	Kreifelts et al.-(Kreifelts 1991)-1991	Transfert de Plan et de messages - (8)	*Transfert et délégations des plans en totalité entre les agents*
	Nii-(Nii 1989)-1989	Tableau noir - (9)	*Permet de partager une ressource commune aux agents lors de l'échange de leurs messages*
	Rousseau et al.-(Rousseau 1995)-1995	Actes de discours de la conversation et états mentaux des agents. - (10)	*Offrir à l'agent la possibilité de reconnaître les types d'actes de discours primitifs et des états mentaux sous forme logique*
	Finin et al.-(Finin 1994)-1994	Langage entre agents - (11)	*KQML : langage de requête et d'assertions et de commandes, développé de façon ad hoc.*
	Vongkasem et al. –(Vongkasem 1999)-1999		*ACL (Agent Communication Langage) : Langage muni d'une sémantique et de protocoles explicites.*

2.3.2.4 Conclusion

Nous opterons pour une approche orientée agent de l'AL multimédia car nous croyons qu'une telle approche est pertinente pour ce type de système. Nous pouvons avancer différents arguments qui motive une approche orienté agent. Une approche orientée agent est avantageuse dans des situations où des types complexes/divers de communication sont mis en jeux. Cette idée rejoint les notions déjà vues qui concernent les AL (paragraphe 2.2.1.) Nous l'adopterons dans notre problématique. Communication orientée humain ou interaction entre entités hétérogènes en sont des exemples. Ce type d'interaction est fréquent dans les applications multimodales. Une approche orientée agent est bénéfique quand le système doit être efficace et performant dans des situations où il n'est pas pratique/possible de spécifier leurs comportements en se basant sur une étude cas par cas. Par exemple, le système sera déployé dans un environnement inconnu où plusieurs solutions sont possibles.

En fait un agent a un comportement 'orienté but' qui n'est pas spécifié en termes d'association des entrées aux sorties : c'est à dire, en termes de « quoi faire? » En contre partie, son comportement est spécifié en termes de « comment décider quoi faire? » Cette approche permet d'aboutir à des systèmes flexibles qui peuvent adapter leurs comportements dans des circonstances changeantes afin de satisfaire les buts dont ils ont la responsabilité.

Ces types de systèmes cadrent très biens avec les applications multimodales intelligentes. Une approche orientée agent est profitable dans des situations impliquant des négociations, des coopérations et la compétition entre plusieurs entités. Par exemple, quand différentes tâches à buts conflictuels doivent être prises en charge de manière concurrente, ce type de système est nécessaire pour servir les multiples tenants des enjeux.

Lors de la prise en compte du comportement orienté buts et de l'adaptation aux négociations, une approche orientée agent est importante quand le système agit de façon

autonome par exemple à la place de l'utilisateur. Nous nous plaçons ici dans le cadre des systèmes multimodaux intelligents qui génèrent une ambiance ubiquitaire.

Les Systèmes multiagent sont intrinsèquement fortement modulaires. Cela est fondamental quand le système peut être étendu ou modifié où lorsque l'objectif du système peut changer.

Il est possible d'avancer bien d'autres arguments pour un tel choix (distributivité, etc.)

2.4 Conclusion du chapitre 2

Nous avons présenté dans ce chapitre un état de l'art synthétique relatif à notre problématique de recherche : Étude des interactions multimodales dans les applications multimédia : en vue de proposer des architectures logicielles nouvelles pour ces applications. (voir Figure 12.) Cet état de l'art est doublé d'une analyse qui pose les fondements de notre approche.

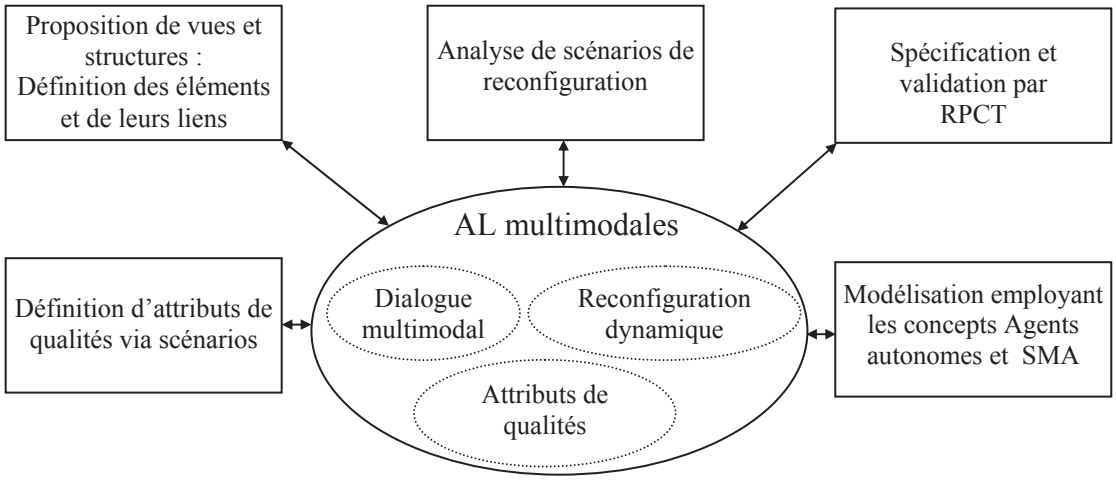

Figure 12 Vue résumant les points importants de notre approche

Il nous montre que les architectures multimédia multimodales présentes dans la littérature sont, pour la plupart, dédiées à des applications particulières pré-établies dépendantes de certains types de matériels et de protocoles de communication.

Tableau V

Comparaison des tendances architecturales les plus représentatives de la littérature avec l'approche que nous souhaitons proposer

Nom de l'architecture - Concepteur(s)-Année	Évolutivité de l'architecture		Description de l'architecture multimodale						Aspect Générique de l'architecture multimodale	
			Spécification du dialogue			Aspects Intelligents en termes d'Agents				
	Sta--tique	Dyna-mique	RTA	RPCTS	UML	Agents Identifiés sans ambiguïté	Rôles des agents définis sans ambi-guïté	Scénarios d'inter-actions établis sans ambiguïté	Générique (tous les types de fusion, fission et de média)	Dédiée à des média particuliers et à des types de fusion et de fission
PAC_AMODEUS – Nigay,Coutaz et al. - 1995	X					X			X	
SPECIMEN - Bellik -1995	X		X			X	X	X		X
OAA - Moran et al.- 1997		X				X	X	X		X
MIAMI PVM - Schomaker et al. - 1995					X					X
QUICKSET - Cohen et al. -1997		X			X	X				X
CORBA - Vinoski - 1997		X			X					
Notre approche- - 2003 à 2006.		X		X		X	X	X	X	

Elles sont donc difficilement modifiables, en cours d'utilisation, pour l'emploi de nouveaux média par exemple.

Si elles sont génériques ces caractéristiques ne sont alors définies que pour des niveaux d'abstractions élevées. Par ailleurs, la reconfiguration dynamique d'une structure multimodale multiagent où l'identification des agents, leurs rôles et les interactions qu'ils entretiennent reste un important domaine à explorer pour ces architectures.

Comme indiqué par le Tableau V, ces aspects sont encore mal étudiés et ne font pas encore l'objet de méthodologies formelles. Ceci nous permettra d'établir notre travail de recherche relatif aux applications multimodales. Nous proposerons des AL génériques (statique et dynamique) permettant une reconfiguration en cours d'utilisation, à base d'agents intelligents.

La Figure 12 replace notre travail dans le cadre de la documentation et de l'analyse, bref de la spécification de ce type d'architecture et devra faire appel à un ensemble de théories et/ou de disciplines. Cette Figure est une vue qui résume les points importants de notre approche.

La description de la première première partie de nos travaux de recherche fait l'objet du chapitre suivant. L'aspect de reconfiguration dynamique fera quant à lui l'objet du chapitre 5.

.

CHAPITRE 3

PROPOSITION D'ARCHITECTURES LOGICIELLES MULTIMODALES

3.1 Introduction et démarche de recherche

La méthodologie que nous suivons reprend, dans un premier temps, les différentes premières étapes du cycle de vie d'un logiciel. La Figure 13 indique les étapes de développement d'une AL, représentées sous la forme d'un réseau de Petri.

Dans une première étape, nous allons rassembler les besoins relatifs aux applications multimodales. Cette première étape est déjà largement amorcée dans le chapitre 2 où nous avons mis en évidence ces besoins et les différentes méthodes par lesquelles la capture de ces besoins peut s'effectuer. Nous ne ferons donc ici que synthétiser les points fondamentaux relatifs aux exigences qu'implique notre approche.

Cependant nous allons capturer ces besoins selon un style d'architecture en commençant à un niveau d'abstraction élevé. Cette première décomposition du système en modules, représentant des agents, permettra l'identification des composantes génériques de base de l'architecture ainsi que de leurs liens. Elle constituera un premier niveau de granularité architectural. Par la suite nous ferons une décomposition de la structure de l'architecture en éléments plus détaillés. Cette décomposition offrira une modélisation de ces besoins à travers la description des flux de données et de leurs traitements grâce aux RPCTS.

Nous montrerons les apports du choix stratégique de l'association des RPCT et du paradigme agent. Les caractéristiques de l'architecture multimodale proposée seront détaillées en termes de généricité (maniable, compréhensible et extensible à tout type de fusion, fission et modalités) et en termes d'une représentation centrée sur les évènements (réactif en temps réel tout en prenant en compte la concurrence.)

Nous montrerons comment notre proposition permet de faire des validations au niveau du dialogue multimodal et quelles sont les qualités et propriétés de ce type d'architecture (gestion des erreurs de l'utilisateur pour la robustesse du système, etc.)

En résumé, nous présenterons une première proposition d'architecture, du dialogue multimédia multimodal, générique mais statique (en termes de reconfiguration architecturale.) Pour ce faire, des paradigmes architecturaux pour la fusion et la fission multimodales seront proposés.

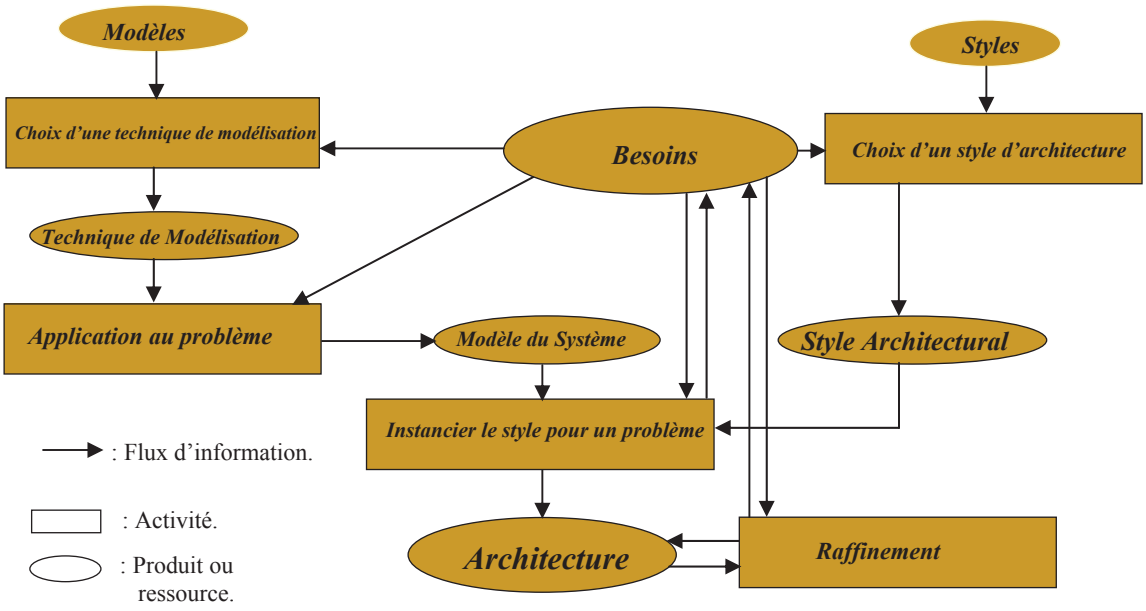

Figure 13 Réseau de Petri des phases du développement logiciel

Ces paradigmes emploieront le concept architectural d'agent pour réaliser les fonctionnalités afférentes au dialogue et les unifier en des structures réutilisables statiques. La modularité des structures proposées permettra un monitoring efficace de la structure globale. Cette méthodologie (Djenidi 2002-a; Djenidi 2002-b; Djenidi 2004-b) est donnée ci-après dans les paragraphes 3.2. à 3.6. Le paragraphe 3.7 est un exemple de simulation employant cette méthodologie.

3.2 Une première approche

3.2.1 Introduction

Cette première approche concerne une proposition d'architecture multiagent, multimodale à base de RPCT. Nous exposons d'abord les besoins des AL multimodales. Puis nous déployons des aspects architecturaux pertinents de la fusion/fission. Nous proposons des caractéristiques et propriétés pour notre proposition d'AL. Nous expliquons comment la gestion des erreurs est réalisée pour notre proposition et nous terminons par un exemple classique modélisé par RPCTS.

3.2.2 Besoins des architectures multimodales

Le dialogue multimodal est au cœur des préoccupations des besoins posés par les systèmes multimodaux (Figure 14.)

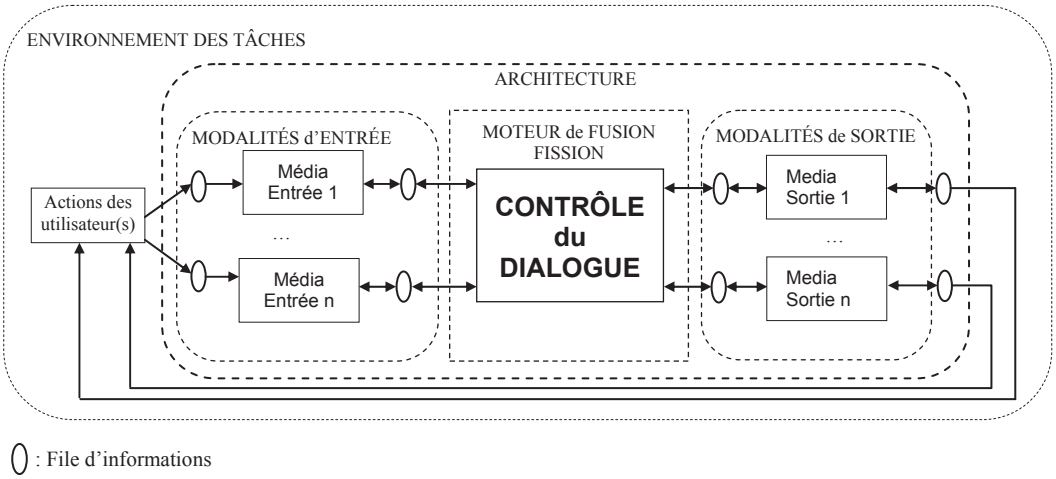

Figure 14 Contrôle du dialogue au cœur de l'architecture

Les contraintes environnementales influencent le dialogue et donc l'AL. Il est donc normal de déterminer les besoins qui permettent les interactions et la communication entre l'utilisateur et le système dans cet environnement des tâches. Les contraintes imposées par cet environnement et par l'utilisateur (situé dans cet environnement) peuvent conduire à des exigences fortes qui peuvent induire une complexité architecturale beaucoup plus importante que celle des applications multimédia classiques. Dans ce contexte, un premier cadre a été introduit dans (Hutchins 1986) pour classifier les interactions selon deux dimensions (l'engagement et la distance.)

Ainsi, il est possible de décomposer le dialogue 'système utilisateur' selon quatre variantes possibles, dépendamment de l'engagement et de la distance entre l'utilisateur et le système (Tableau VI.)

Tableau VI

Interaction dans les Systèmes.

Engagement de l'utilisateur ou du système	**Distance : utilisateur système**	**Type de Système**
Conversation	faible	Langage haut niveau
Conversation	Grande	Langage bas niveau
modèle de l'environnement	Petite	Manipulation directe
modèle de l'environnement	Étendue	'Environnement' bas niveau

L'engagement caractérise la profondeur de l'implication de l'utilisateur vis à vis du système. L'utilisateur ressent qu'un sous-système intermédiaire réalise la tâche, dans le cas de la 'conversation', alors qu'il peut agir directement sur les composants de ce système dans le cas d'un modèle de l'environnement. La distance représente l'effort cognitif réalisé par l'utilisateur. Ce classement rejoint l'idée préconisant deux architectures multimodales possibles. La première réalise des fusions basées sur la reconnaissance des détails caractéristiques des signaux : les étapes du processus de reconnaissance d'une modalité guident et influencent les autres modalités dans leurs propres étapes de reconnaissance. La seconde architecture utilise des systèmes de

reconnaissance indépendants pour chacune des modalités. Un système supplémentaire réalise alors la fusion sémantique des éléments reconnus individuellement. Une architecture hybride, combinant le niveau sémantique de l'information au niveau des détails caractéristiques des signaux, est également possible. Le design de la fusion de l'information provenant des média d'entrée constitue l'un des principaux défis dans les systèmes multimodaux et doit tenir compte de besoins que nous réunissons de façon synthétique sur la Figure 15.

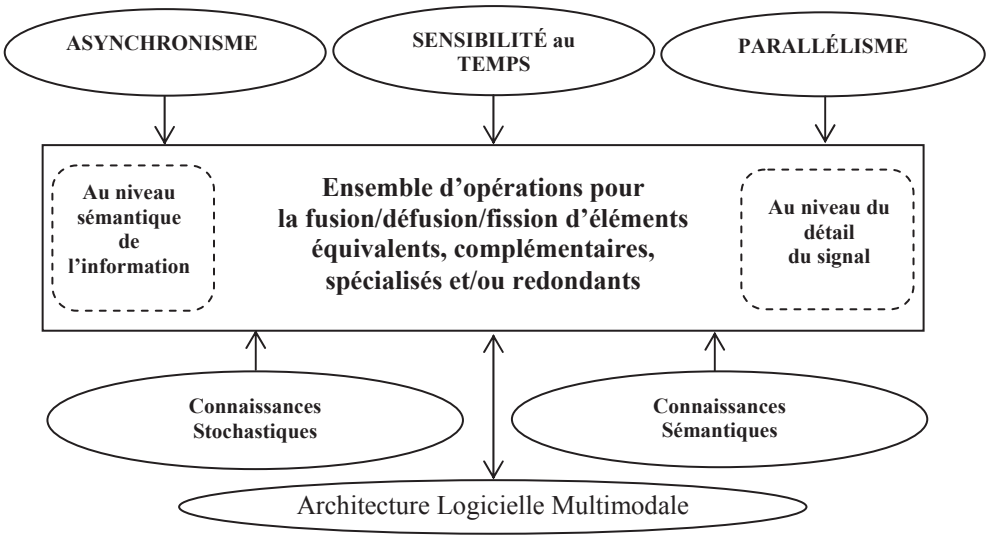

Figure 15 Principaux besoins pour une architecture du dialogue multimodal

Les modalités d'entrée et de sortie peuvent être équivalentes, complémentaires, spécialisées ou redondantes comme décrit dans (Coutaz 1994). Un système, conçu avec l'une des architectures (niveau sémantique et/ou niveau de détail du signal), nécessite l'intégration de différents besoins. Comme le montre la Figure 15, cinq caractéristiques correspondantes aux peuvent être employées dans les deux niveaux où se produisent les opérations de fusions et de fission.

La fusion dite lexicale, se produit au niveau du détail des fragments du signal. La fusion, dite sémantique, combine des éléments comme des commandes et des données pour réaliser d'autres commandes plus complexes.

Sur la Figure 15, la sensibilité au temps dont doit tenir compte une AL multimodale se traduit par la nécessité de l'étiquetage temporel des événements multimodaux, de leurs durées et du choix des intervalles temporels durant lesquels les processus de combinaison (fusion/défusion/fission) des fragments d'informations multimodales sont autorisés à se produire dans l'application. L'AL doit aussi prendre en compte le temps de traitement des différents dispositifs (média) employés dans l'application. Toutes ces informations temporelles qu'il faut établir et prévoir lors de la construction de l'AL multimodale correspondent au besoin : 'sensibilité au temps' de la Figure 15. La sensibilité au temps dans les architectures multimodales ne sert pas uniquement à la prise en compte du recouvrement des signaux. Elle est aussi employée pour décider si deux signaux peuvent être considérés comme des actions unimodales séparées (durée importante entre deux événements.) Ainsi, les systèmes multimodaux sont à même d'éviter et de retrouver des erreurs que des systèmes unimodaux ne pourraient prendre en considération. Il en résulte un langage 'personne machine' plus robuste.

Une autre propriété est que la croissance des combinaisons sémantiques possibles entraîne celle du nombre de formulations équivalentes de la même commande. Par exemple, ['copie ça' (clic)] et ['copie' (clic) 'ça'] sont deux énoncés (combinaisons sémantiques) de la même commande. Il s'agit de la copie d'un objet depuis un lieu donné, la parole et le clic de souris étant les deux modalités employées. Cette redondance accroît, également, la robustesse en terme d'interprétation des erreurs. L'ensemble des énoncés sémantiques possibles et les grammaires multimodales qui régissent leurs combinaisons constituent les connaissances sémantiques (Figure 15) nécessaires au bon fonctionnement des applications multimodales.

Les connaissances stochastiques (Figure 15) sont, par exemple, les informations qui permettent de caractériser par des lois stochastiques les instants d'arrivée des différents événements produits par les modes de l'application. Par exemple, la fréquence d'arrivée

des symboles représentant les mots prononcés par l'utilisateur peut suivre une loi exponentielle. En effet, l'utilisateur peut dicter des commandes dans le microphone d'un système de reconnaissance vocale et le temps entre deux arrivées successives de mots peut, par exemple, avoir une moyenne de 20 millisecondes. Le temps inter-arrivée des mots suit alors une distribution exponentielle de paramètre = 1/20 dont il faut tenir compte dans l'architecture pour pouvoir faire des tests. Tous les aspects faisant intervenir des lois stochastiques lors des processus de décisions pris durant le dialogue multimodal doivent aussi faire parti des connaissances stochastiques.

La propriété d'asynchronisme donne à l'architecture la possibilité de traiter des événements externes pendant que des opérations de fusion ou de fission sont en cours. Les connaissances sont utilisées dans les processus de sélection (du type d'information pour la fusion) et de décision (de la meilleure commande parmi différentes fusions possibles et de du meilleur choix de représentation selon le contexte lors du processus de fission.)

Le prallélisme est aussi un besoin dont il faut tenir compte lors de la conception d'une AL logiciel multimodale. Il faut en effet permettre une modélisation de la gestion du parallélisme déterministe et non déterministe dans ce type d'application ou l'emploi synergique des modalités peut se produire. L'ensemble des besoins présentés à la Figure 15 suggère fortement une architecture multiagent.

3.2.3 Approche d'architecture multiagent pour l'interaction multimodale

Une solution courante employée pour gérer la complexité des systèmes est le concept d'abstraction. Il s'agit de la définition d'un modèle partiel ou simplifié de l'AL ou de ses sous parties qui met en valeur certaines propriétés et en dissimule d'autres. Cela permet au concepteur de traiter et prendre en compte les caractéristiques majeures au dépend des moins significatives. Ceci permet aussi de réaliser une implémentation logicielle par étapes : depuis les parties les plus grandes vers les plus petites (implémentation

descendante ou ‘top-down’) ou inversement (implémentation ascendante ou ‘bottom up’).

Nous avons choisi de proposer une approche architecturale multiagent descendante. Au niveau le plus élevé, des abstractions de haut niveau (de chacun des agents) seront employées. Ainsi, chaque agent peut avoir plusieurs ports d’entrée pour recevoir des messages et/ou plusieurs ports de sortie pour en envoyer. Notons que ces deux approches, ascendantes et descendantes, peuvent s’adapter aux fusions et fission sémantique et lexicale pour les cas d’applications multimédia multimodales. Tous les combinaisons (lexico-sémantique) entre ces approches de fusion et fission, sont aussi réalisables.

Afin de définir les agents qui composent l’AL que nous souhaitons proposer, nous pouvons remarquer que, dans la plupart des applications, la parole, comme modalité d’entrée, offre souvent la rapidité d’interaction, un large spectre d’information et une relative facilité d’emploi. Par exemple, lors de situations de pilotage, elle laisse les mains et le regard de l’utilisateur, libres d’œuvrer à d’autres occupations.

Mieux que cela, la parole met en œuvre un modèle langagier générique pour la communication ‘homme machine.’ Ce modèle générique langagier est décrit par une grammaire avec des règles de production. Ces règles génèrent des phrases par sérialisation des symboles produits par l’utilisateur et appartenant à un vocabulaire. Le vocabulaire peut être un ensemble de mots, phonèmes ou autres fragments de signaux, dépendamment du niveau de détail du et du type de système de reconnaissance.

Le but du système de reconnaissance est d’identifier les fragments de signaux. Nous alléguons qu’un agent peut organiser ces fragments en une phrase ou un énoncé. La phrase est composée de fragments classés selon les connaissances grammaticales et temporelles de l’agent. Nous appelons se regroupement sérialisation ou regroupement en série car les éléments qui constituent la phrase sont placé les un à la suite des autres et forment une série ordonnée (en fonction de la grammaire et des étiquettes temporelles portées par les fragments.) Ces fragments produits correspondent à une seule modalité. L’agent correspondant à cette modalité peut questionner les autres agents sur une

éventuelle possibilité de fusion de défusion ou de fission, à chaque étape du regroupement ou déregroupement des éléments de la phrase. L'interaction globale peut se synthétiser en un premier agent d'architecture générique, représenté à la Figure 16. Cet agent est appelé Agent du Langage (AdL). Chaque modalité d'entrée est associée à un AdL. Pour les modalités simples, comme le clic d'une souris, la complexité de l'AdL est fortement réduite.

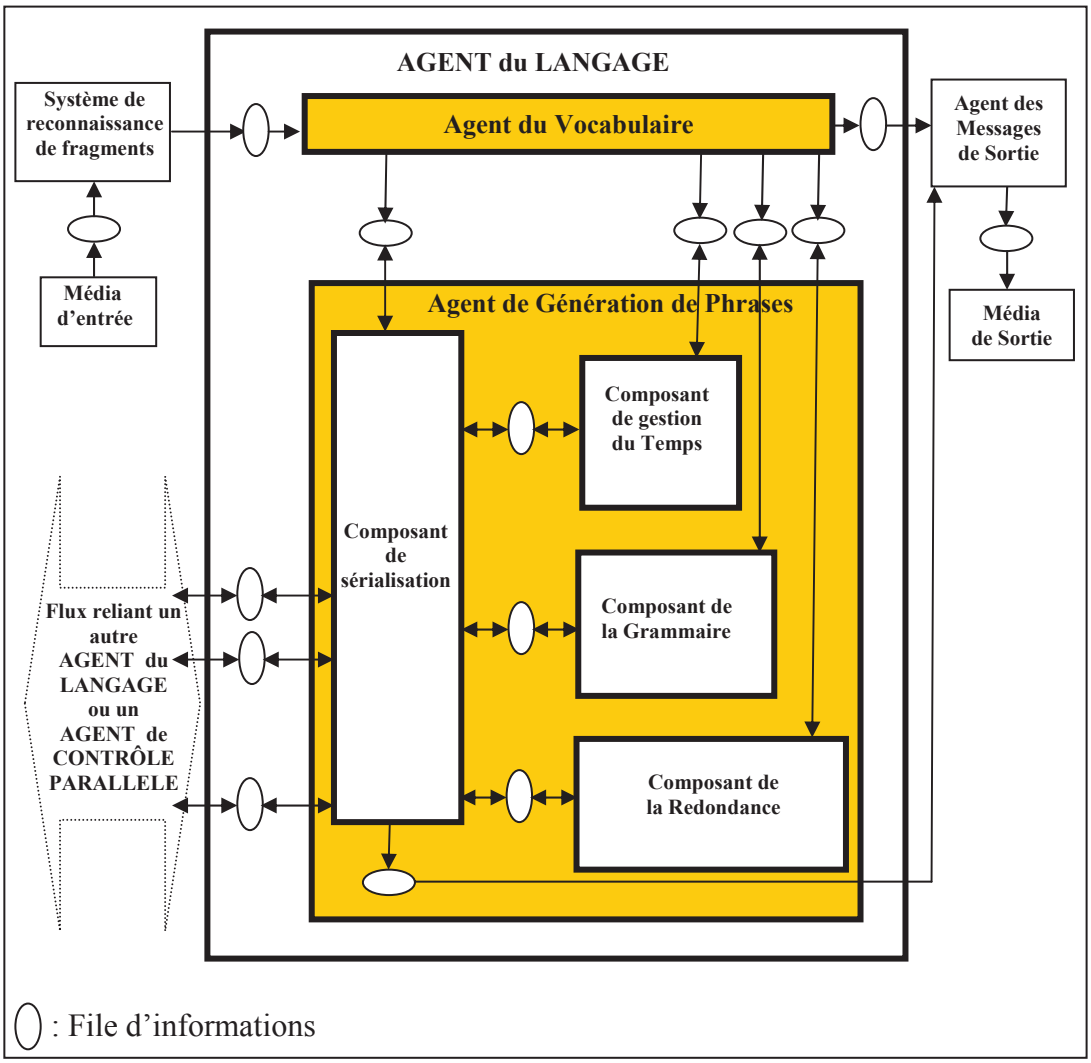

Figure 16 Agent générique du langage correspondant à une modalité en entrée

L'agent du vocabulaire qui vérifie si un fragment appartient au vocabulaire, n'est plus nécessaire dans ce cas. L'agent de génération de phrases est réduit à une simple file d'attente des événements, où un autre agent externe de contrôle pourra prendre en main les fonctions des composants de redondance et du temps. Ces deux composants sont des vérifient, respectivement, la redondance et la proximité temporelle des fragments durant le processus de regroupement en série.

Le composant de sérialisation génère ce regroupement (Figure 16.) Ainsi, selon le type de modalité en entrée d'un système multimodal, l'AdL peut être assimilé à un système complexe ou à une simple file d'attente.

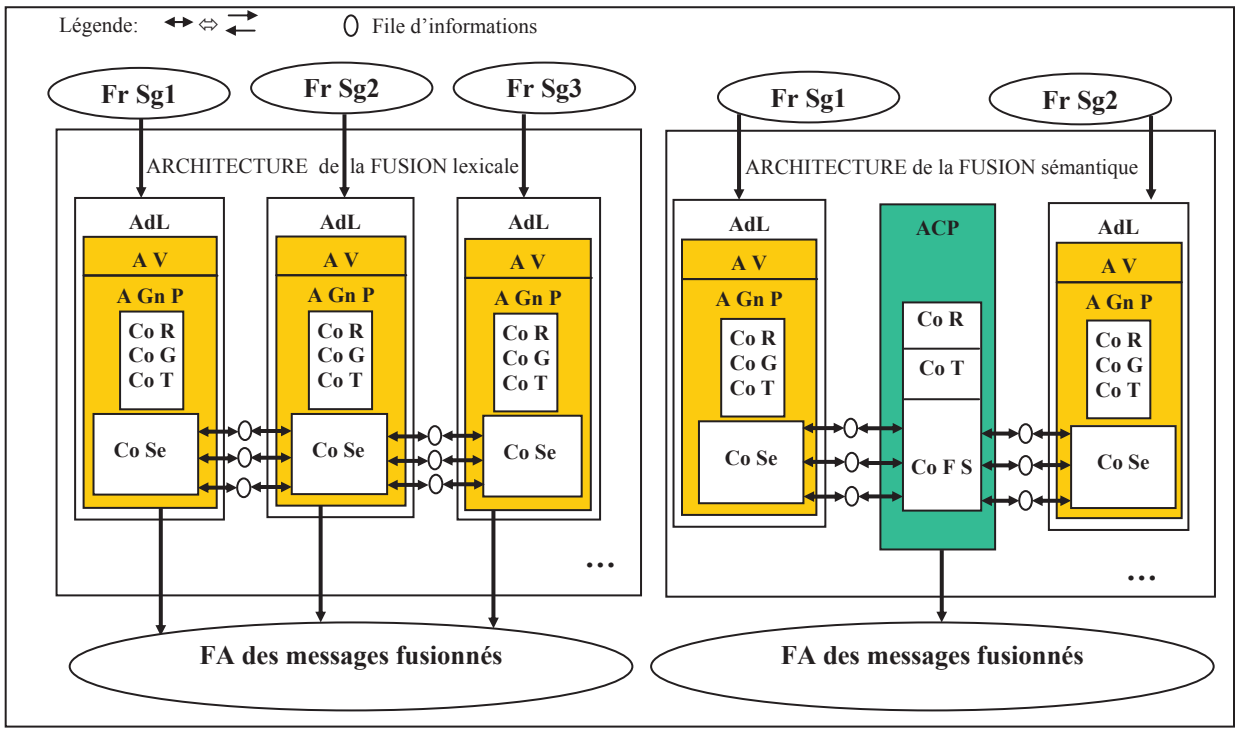

Figure 17 Vues des architectures des fusions lexicale et sémantique (P: parallèle, C: contrôle, A: agent, AdL : Agent du Langage, G: grammaire, R: redondance, S: sémantique, T: Temps, P: phrase, Gn: génération, F: fusion, Se: sérialisation, V: Vocabulaire, Co: composant, Fr: Fragment, Sg: Signal, FA : File d'attente)

Deux ou plusieurs AdL peuvent communiquer directement pour une fusion lexicale (Figure 17 à gauche) ou via un autre agent pour une fusion sémantique (Figure 17 à droite). Dans le premier cas, le composant grammatical, de l'un des AdL doit comporter des connaissances supplémentaires pour la fusion parallèle. Ces connaissances peuvent également être distribuées entre les composants grammaticaux des AdL, comme symbolisé par la Figure 17 (gauche).

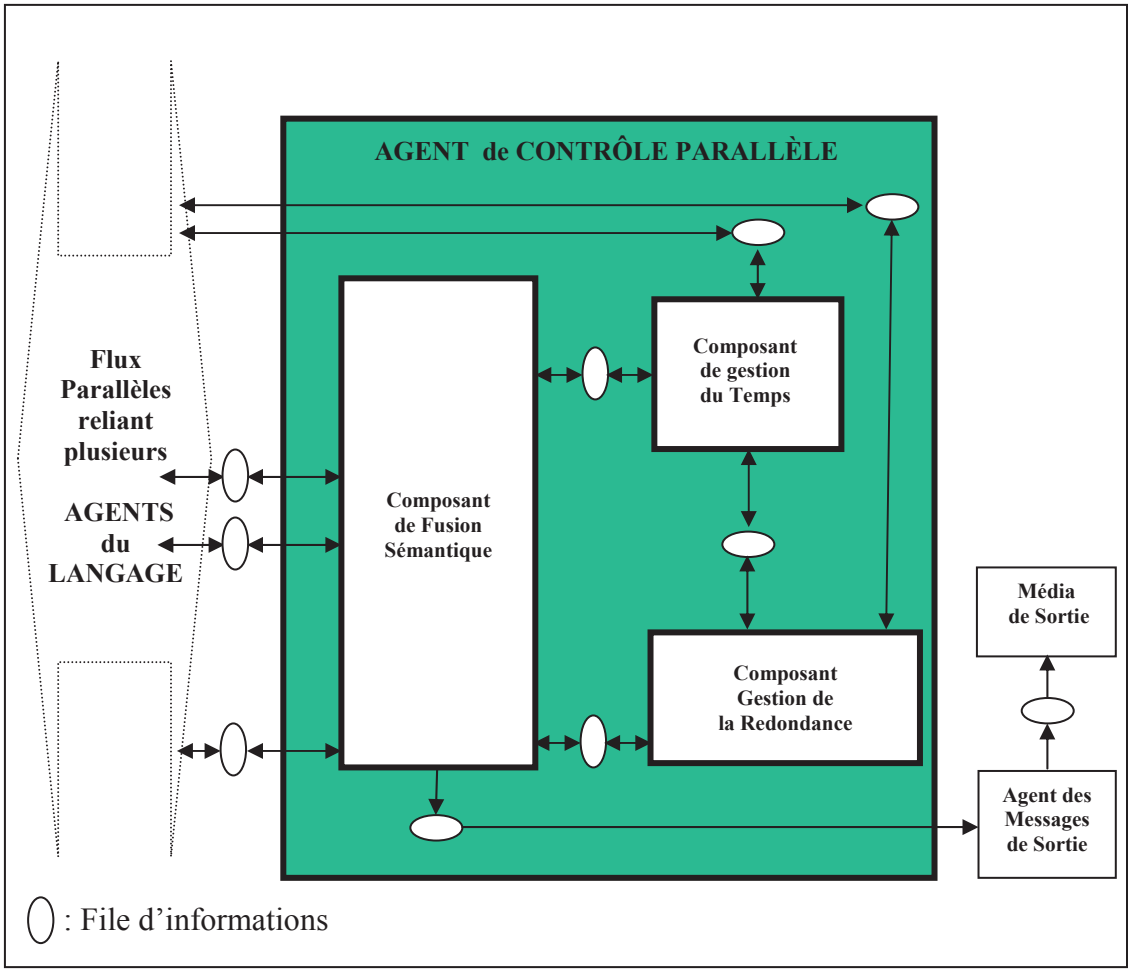

Figure 18 Agent générique pour le contrôle central des fusions parallèles

Par exemple, plusieurs composants de sérialisation partagent ainsi leurs informations communes jusqu'à ce que l'un d'entre eux donne le résultat de la fusion parallèle en sortie. Dans l'autre cas (Figure 17 à droite), un Agent de Contrôle Parallèle (ACP) traite et centralise les fusions parallèles de l'information en provenance des différents AdL.

Pour ce faire, l'ACP a deux composants gérant, respectivement, la redondance et le temps (Figure 18.) Ces composants échangent de l'information avec les autres composants de sérialisation (Figure 16 et Figure 17 droite) des AdL pour élaborer des décisions. Puis, les autorisations générées sont envoyées au Composant de Fusion/Fission Sémantique (Co F S). Ces autorisations permettent au Co F S de poursuivre les étapes du processus sémantique de fusion/fission. Le Co F S réalise donc la fusion selon une grammaire temporelle qui correspond à un ensemble des énoncés sémantiques multimodaux permis par cette même grammaire. Une autre grammaire permet de réaliser la fission sémantique en sortie. Elle est également embarquée par le Co F S.

Comme indiqué sur la Figure 18, Les composants de gestion du temps et de la redondance, se procurent l'information temporelle et sur la redondance via le Co F S ou directement depuis des AdL (cela dépend de la complexité de l'architecture et des choix des développeurs.)

Les paradigmes proposés dans cette section constituent un pas important du cycle du développement des AL multimodales.

Les composants de gestion du temps des AdL ont été omis sur la Figure 18 par soucis de simplification.

Cependant, il est possible et même avantageux d'intégrer la gestion du temps dans les Co Se et les Co F S lorsque la modélisation est effectuées par les RPCT. Ceci fera l'objet du paragraphe suivant

En résumé, les Figures 16, 17 et 18 sont des vues multiples abstraites d'une AL multimodale générique à un haut niveau d'abstraction (niveau des composants) qui font apparaître plusieurs aspects importants :

a. elles modélisent une partie des besoins fonctionnels présentés au paragraphe précédent : les services que le système fournit aux utilisateurs permettent la fusion multimodale lexicale et sémantique ainsi que la fission sémantique;
b. elles offrent les grandes lignes d'une première organisation statique (décomposition selon différents sous-systèmes) pour planifier le développement;
c. elles définissent, des contraintes sur les éléments architecturaux pris individuellement et des contraintes sur leurs dispositions conjointes. Ces contraintes sont pondérées et classées selon leur importance de façon hiérarchique;
d. elles offrent une modélisation dynamique à un haut niveau d'abstraction du fait de l'emploi des réseaux de Petri.

Nous présentons aux paragraphes suivants des éléments de raffinement de ces vues architecturales génériques.

3.3 Modélisation des agents par les réseaux de Petri

Nous pensons que les RPCT stochastiques offrent la possibilité de prendre en compte des contraintes temporelles, de parallélisme, dans une structure dynamique modélisant le dialogue multimodal. Pour ce faire, chaque modalité d'entrée est assimilée à une file d'attente de fragments d'un signal. Les entrées multimodales constituent un ensemble de files d'attente parallèles correspondantes à un environnement variable. Cet environnement décrit les différents états internes du système. Dans ce contexte, les systèmes multiagent sont de type « multi-files d'attentes » : chaque agent a le contrôle d'une ou plusieurs files d'attentes.

Les agents observent les états d'une ou plusieurs files pour lesquelles ils ont été conçus. Puis, ils exécutent les actions qui modifient l'environnement.

La description qui suit définit les conventions de notations employées par les réseaux de Petri pour modéliser les agents qui composent les architectures de la Figure 18. Cette

convention rejoint le paradigme ‘objet dans les réseaux de Petri’ présenté dans (Bastide 1995) et permet en plus la prise en compte de la composante temporelle.

Les réseaux de Petri sont des diagrammes fluents composés de places (représentées par des ellipses) et de transitions (rectangles) interconnectées par des arcs et gérées par un ensemble de règles. (Voir Annexe 2 pour plus de détails théoriques sur les réseaux de Petri.) Les règles déterminent quand l’occurrence d’une transition peut se produire et spécifie comment cette occurrence change l’état des places, par changement de leurs marquages colorés. Une place contenant des marques modélise une file d’attente car il est possible de faire en sorte que chaque marque porte l’information sur l’instant de son arrivée dans la file grâce à son attribut de couleur. L’ensemble des marques colorées dans toutes les places avant une occurrence du réseau (franchissement d’une transition) est équivalent à une séquence d’observation du système multiagent. Chaque couleur correspond, par exemple, à un type de fragment de signal et peut donc porter l’information sur ses propriétés pertinentes (temps, et autres attributs permettant de spécifier le symbole : fusion réalisée, priorité, etc.). La couleur est simplement un type de donné qui peut être une variable entière, réelle, Booléenne, chaîne de caractères ou encore un type plus complexe comme une énumération, etc. Un RPCT constitue donc une structure graphique associée à des énoncés d’un langage de programmation.

Une petite présentation formelle sur les agents (adaptée de (Weiss 1999)) s’impose ici, pour démontrer la relation entre les deux formalismes : agent et réseau de Petri.

Si $\mathbf{Actions = \{a_1; a_2; \dots\}}$ (2.1),

et $\mathbf{Observations = \{ o_1; o_2; \dots\}}$ (2.2),

représentent, respectivement, les ensembles des actions et des observations d’un agent,

et si, $\mathbf{States = \{ s_1; s_2; \dots\}}$ (2.3),

est l'ensemble des états par lequel l'environnement est décrit (incluant les états intermédiaires), alors un réseau de Petri modélise deux types d'activités décrites par les fonctions

$$\textbf{Observation\_fonction : States} \rightarrow \textbf{Observations} \qquad (2.4),$$

$$\textbf{Environnement\_fonction : States x Actions} \rightarrow 2^{S} \qquad (2.5).$$

La première fonction décrit ce qu'un agent observe lorsque le système est dans un certain état $\mathbf{s}_i$. La seconde décrit comment l'environnement développe l'état si une action $\mathbf{a}_i$ est exécutée.

Le réseau de Petri modélise aussi les actions des agents décrites par la fonction

$$\textbf{Action\_fonction : Observations} \rightarrow \textbf{Actions} \qquad (2.6).$$

Le comportement caractéristique de l'action d'un agent dans un environnement est l'ensemble 'Historique':

$$\textbf{Historique} = \{ \mathbf{h_1}, \mathbf{h_2}, \ldots \mathbf{h_i}, \ldots \} \qquad (2.7)$$

de toutes les séquences des observations, défini par :

$$\mathbf{h_i}: (\mathbf{s_0}) \text{—}a_0\text{—>} (\mathbf{s_1}) \text{—}a_1\text{—>} \ldots (\mathbf{s_i}) \text{—}a_i\text{—>} \ldots \qquad (2.8)$$

avec $$\mathbf{a_i} = \textbf{Action\_fonction } (<\mathbf{s_0}, \ldots, \mathbf{s_i}>), \quad \forall\, i \qquad (2.9)$$

et $$\mathbf{s_i} = \textbf{Environnement\_fonction } (\mathbf{s_{i-1}}, \mathbf{a_{i-1}}), \quad \forall\, i,\ i \neq 0 \qquad (2.10)$$

($\mathbf{s_0}$ est l'état initial du système).

En résumé le précédent formalisme nous montre que le réseau de Petri doit modéliser les fonctions (2.4), (2.5) et (2.6). Nous pouvons noter la correspondance entre ce formalisme et les propriétés des réseaux de Petri. Par exemple, le graphe d'accessibilité d'un réseau correspond à l'ensemble **Historique** des agents. Mais le réseau peut aussi modéliser les files d'attentes des symboles contenant les fragments de signaux générés par les média d'entrée et manipulés par les agents. Ces files de fragments de signaux sont modélisées par des ellipses appelées Fr Sg dans les Figures 18, 19, 20 et 24.)

Ainsi, une transition représentée par un rectangle peut modéliser un agent. La fonction d'observation de l'agent est alors modélisée à la fois par les instructions associées aux arcs entrant dans la transition et par les conditions entre crochets sous cette même transition. Les inscriptions sur les arcs entrants spécifient les données qui doivent exister (les marques présentes dans les places d'entrée) pour que la transition soit franchie et elles spécifient, éventuellement, le temps de franchissement. Ce temps est symbolisé dans la Figure 19 par l'écriture (en langage CPN-ML (Jensen 1995)) **@+temps**. Une fois la transition franchie, une ou plusieurs marques sont retirées de chacune des places d'entrée (selon les spécifications sur les arcs entrants). Puis, l'action réalisée par l'agent (modélisé par la transition) consiste en la modification des types (ou couleur) associés à chacune des nouvelles marques qui sont générées dans les places de sorties. Les instructions sur les arcs quitant la transition spécifient le nombre de marques et leurs couleurs respectives. Ces instructions spécifient donc les marques produites dans chacune des places de sortie. Ainsi l'état du système change par addition d'une ou plusieurs marques aux places de sortie.

Si des modifications de couleurs se produisent, elles peuvent être exécutées par le programme qui peut être associé à chaque transition. Dans ce mémoire, le script ML (Méta Langage (Jensen 1995)) de ce programme est écrit dans des rectangles en pointillés à proximité de la transition. Les inscriptions se trouvant sur les arcs entrant et sortant de chaque transition sont elles-mêmes du script CPN-ML. Cependant, celles sur

les arcs sortants sont, en quelque sorte, des données générées alors que celles sur les arcs entrant représentent des types des données qui doivent exister dans les places entrantes. Ainsi, les instructions des arcs quittant une transition peuvent aussi contenir du code écrit en ML qui s'exécute lors du franchissement de la dite transition. Ceci permet le changement de l'attribut de la nouvelle marque (c'est-à-dire la valeur de la donnée générée dans la ou les places de sortie). Des conditions et contraintes sur la couleur des jetons peuvent être spécifiées sur les arcs sortants par l'emploi de code en ML (if…then… else, case, appel de fonctions, etc.)

L'étiquette temporelle attribuée à une marque franchissant une transition retourne le temps courant du réseau modélisé.

Le symbole HS (hiérarchie sous-jacente) dans une transition indique que cette dernière peut être remplacée par un sous réseau. Ceci permet de modéliser la hiérarchie de l'architecture et permet ainsi de donner une indication au développeur sur les niveaux d'encapsulation qu'il aura dans son code.

La Figure 19 représente une modélisation générique par RPCT de l'ACP permettant la fusion. Le réseau de la Figure 19 permet de modéliser une fusion parallèle.

Nous le transposons aux cas de la fusion série et de la fusion 'série parallèle' comme le montre la Figure 20 par l'emploi judicieux de la syntaxe du ML.

La modélisation à la Figure 19 met en valeur plusieurs aspects importants pour la spécification architecturale :

a. l'ACP peut être réparti selon une distribution fonctionnelle dans le réseau de Petri qui modélise l'architecture;
b. l'ACP peut être distribué spatialement dans le réseau qui modélise l'architecture; dans certaines localités du système distribué (représentant une vue de l'architecture multimodale) cet agent aura un rôle particulier (la fusion parallèle), dans d'autres localités ses rôles seront différents.

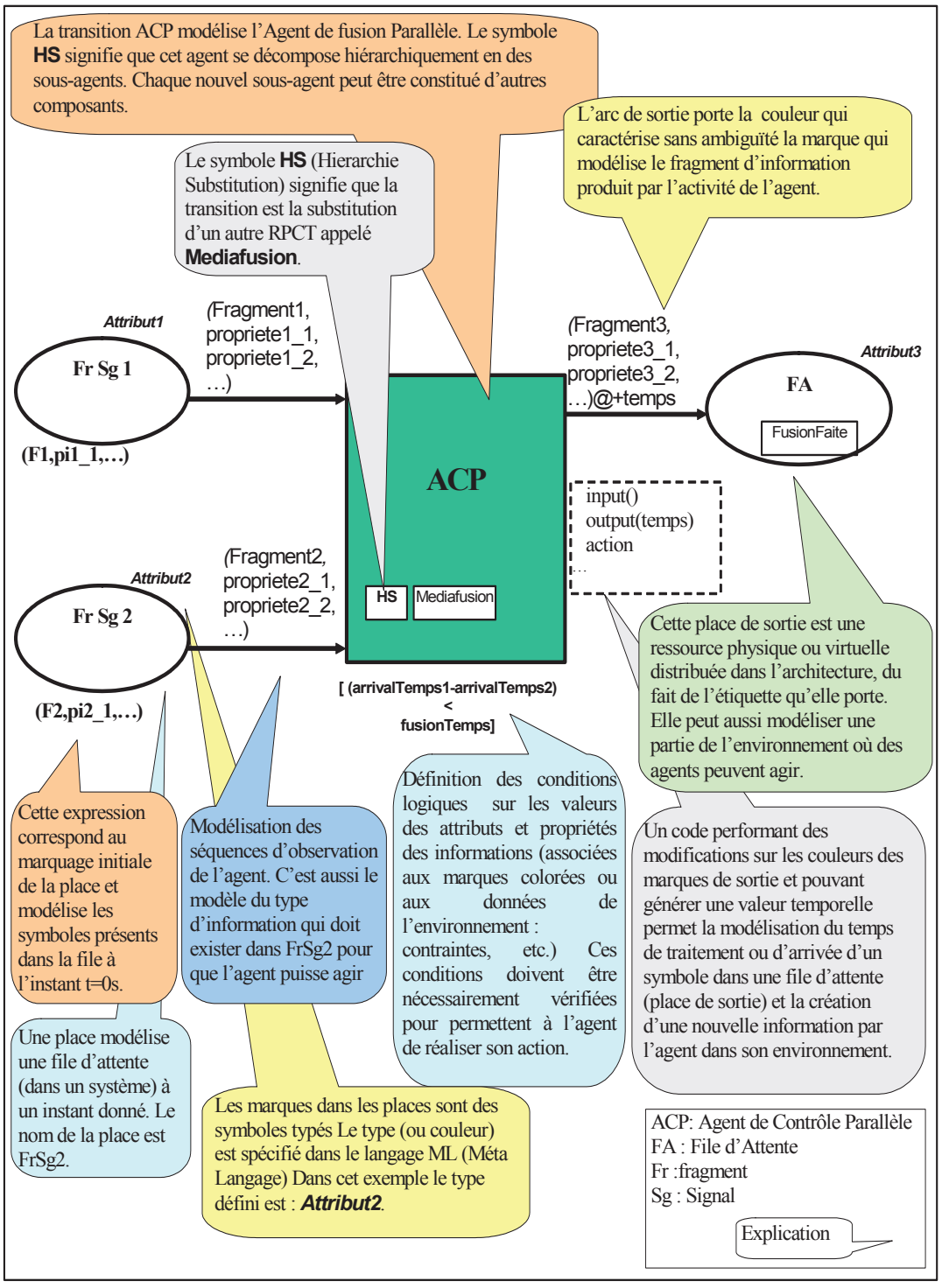

Figure 19 Principes de modélisation de l'ACP par RPCT

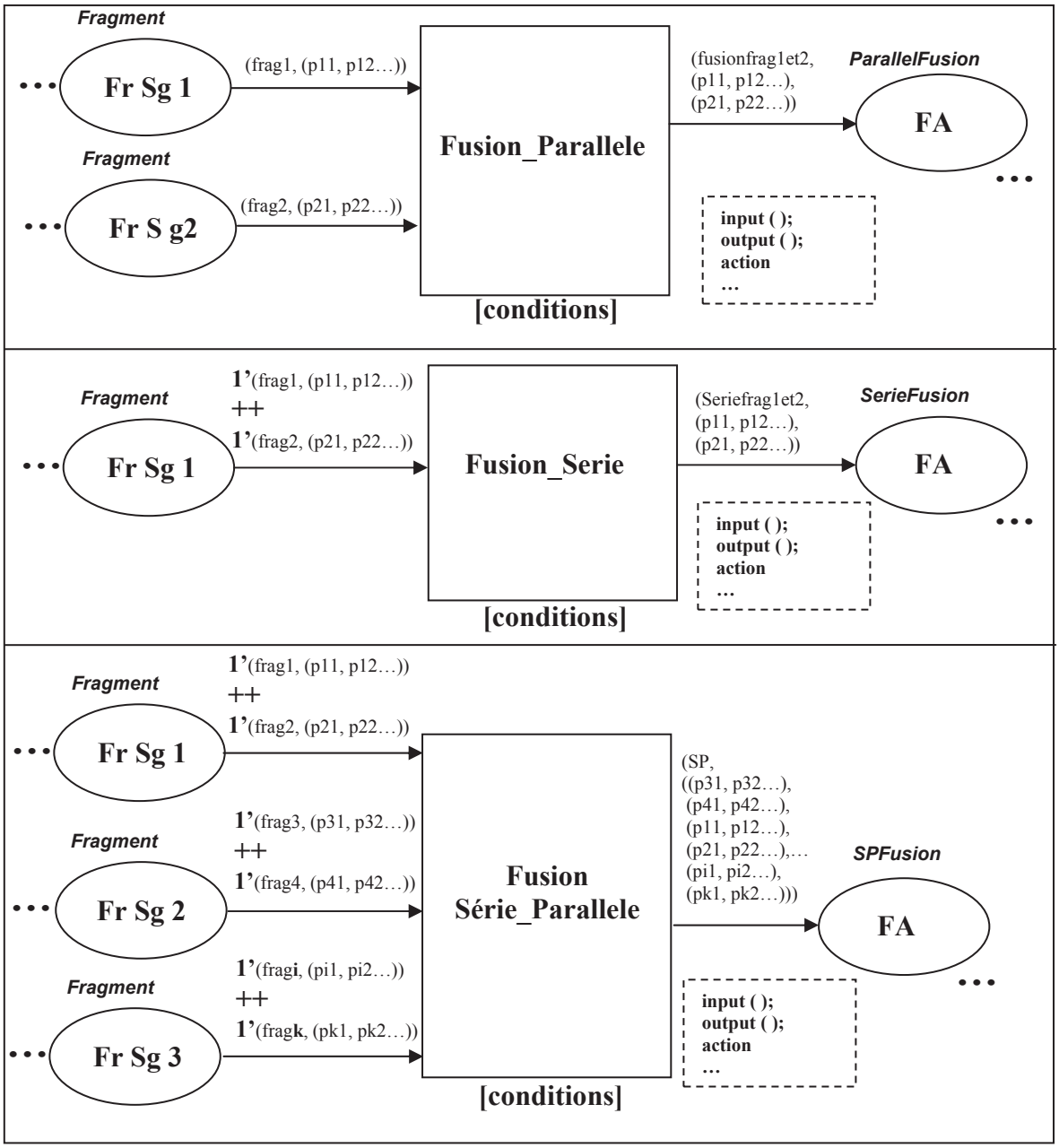

Figure 20 Principes des fusions parallèle, série et 'série parallèle' modélisées par des réseaux de Petri (FA : File d'Attente, fragi: fragment i d'information traitée, pi: propriété i du fragment, Fr : fragment, Sg : Signal, pi: propriété i du fragment)

Les Figures 19 et 20 constituent des vues architecturales génériques des agents de fusion. La décomposition en couches permet ainsi la réduction de la perception des mécanismes internes des agents et facilite ainsi l'extension de l'architecture à de nouvelles modalités. Par ailleurs, la hiérarchie de la structure est extrêmement pertinente pour un développeur qui souhaite encapsuler les comportements des agents.

3.4 Autres vues architecturales génériques

Pour modéliser un SMA multimodal il est pertinent de se baser sur l'approche 'Voyelle' (Ricordel 2001) qui propose quatre dimensions (Figure 21.)

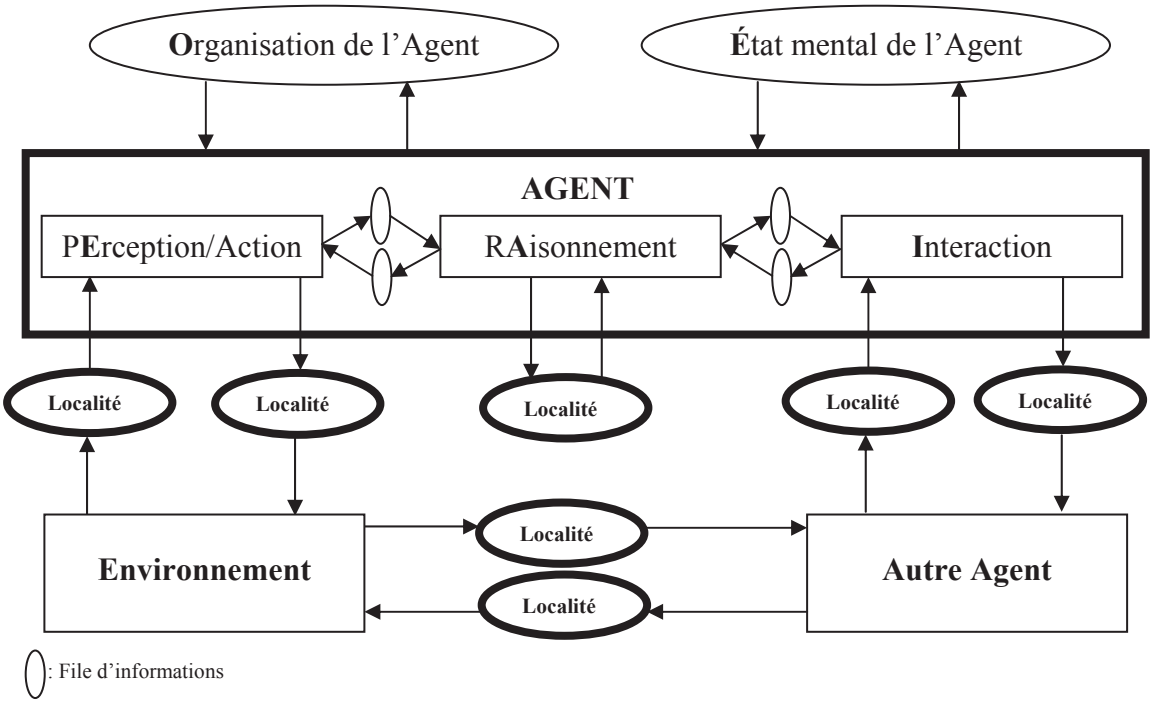

Figure 21 Dimensions AEIOÉ d'un agent

Ces dimensions sont: le raisonnement de l'Agent (A), sa perception et son action sur Environnement (E), son Interaction (I) avec d'autres agents et son Organisation (O). Il

convient d'ajouter à ces dimensions l'État mental (É) de l'agent. Cet état mental (Shoham 1993) est une représentation préalable et partielle (car incomplète) des plans des autres agents. Il s'agit, de ne pas se limiter à un échange d'événements générés de manière purement réactive et de faire en sorte qu'un agent puisse tenir compte dans le processus de génération du dialogue de l'état mental de l'autre agent. Cette dimension est importante pour la reconfiguration dynamique du dialogue multimodale.
Dans l'AL, les distributions fonctionnelle et spatiale de l'agent sont deux dimensions orthogonales à chacune de ces quatre dimensions :

a. la dimension **A** indique toutes les fonctionnalités internes du raisonnement d'un agent qui seront distribuées de façon spatiale (plusieurs transitions portant le même nom se trouvant à différents endroits du réseau) et de façon fonctionnelle (chaque transition aura un raisonnement élémentaire encapsulé[25] dans l'activité réalisée par la transition);
b. la dimension **E** réunit les fonctionnalités relatives aux capacités de perception et d'action de l'agent dans l'environnement; elles sont modélisées entre autres par les inscriptions sur les arcs entrants (pour la perception) sur les transitions du réseau ainsi que par le code associé aux transitions et par les inscriptions sur les arcs qui en sortent;
c. La dimension **I** réunit les fonctionnalités relatives aux interactions de l'agent avec les autres agents (interprétation des primitives du langage de communication, gestion de l'interaction et des protocoles de communication) et la structure du RPC, où chaque transition peut modéliser un agent global décomposé en composants distribués dans un sous RPC (avec ses variables initiales et ses procédures fonctionnelles), correspondent à cette dimension.
d. Les dimensions **O** et **É** sont les plus difficiles à mettre en œuvre avec des RPCT. Elles concernent les fonctions et les représentations relatives aux capacités de restructuration et de gestion des relations entre agents pour faire des changements

[25] Ce raisonnement peut être raffiné par la décomposition hiérarchique de la transition en un autre réseau.

architecturaux dynamiques. Le concepteur du réseau doit prévoir toutes les configurations possibles à l'avance pour une modélisation par RPCT de ces dimensions ou encore il doit lui permettre de choisir une solution de reconfiguration lorsque l'agent est soumis à des stimuli que le concepteur n'a pas prévu. Dans ce cas l'agent est assimilable à un système expert.

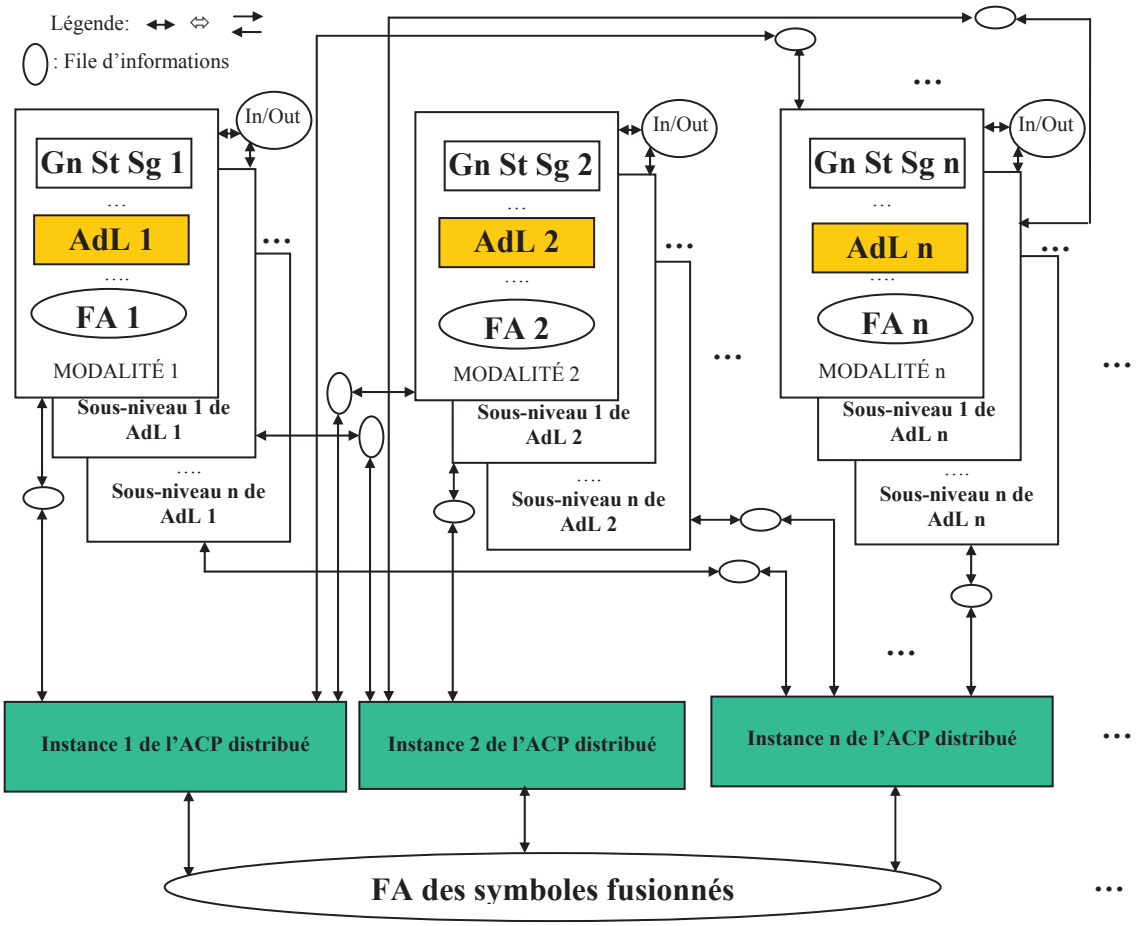

Figure 22 Modélisation par réseau de Petri d'une vue architecturale générique multiagent d'un moteur de fusion et de fission composé d'AdL et d'ACP (A: agent, P: phrase, Gn: génération, F: fusion, Fr: Fragment, Sg: Signal, St : stochastique, FA : File d'attente et d: du)

Sur la Figure 21 nous avons modélisé ces dimensions comme des ressources propres à l'agent, auxquelles il fait appel dès qu'il agit.

Le SMA proposé est représenté par des vues architecturales génériques d'un moteur de fusion multimodale. Nous avons montré comment modéliser par les RPCT les vues de la fusion à la Figure 20. (La Figure 24 montre les vues pour la fission.)

Ce SMA permet de réaliser tous les cas intermédiaires entre l'architecture de la fusion lexicale, complètement décentralisé, et celle de la fusion sémantique, fortement centralisée.

La Figure 22 fait apparaître une vue intermédiaire du modèle de fusion. Nous remarquons que cette vue prend en compte la génération stochastique des flux des symboles à fusionner. Dans un réseau de Petri temporisé, une durée fixe est associée à chaque place ou à chaque transition du réseau. Les modèles obtenus sont bien adaptés pour l'étude des systèmes où les durées opératoires sont fixes. Mais il existe des cas de modélisation qui doivent faire intervenir des durées qui sont des variables aléatoires. Pour cela un temps aléatoire peut être associé au franchissement des transitions. Selon le choix de résolution de conflits, les réseaux de Petri stochastiques peuvent alors être déduits des réseaux temporisés. Dans le cas des modèles que nous proposons, les RPCT sont stochastiques car ils comportent des transitions qui jouent le rôle de générateurs stochastiques d'événements. Ces générateurs apparaissant à la Figure 22 sont modélisés par une transition appelée **Gn St Sg**. Dans l'exemple de la Figure 23, le RPCT stochastique permet de modéliser des temps de traitements qui suivent des lois de probabilités aléatoires (voir codes associés aux transitions '**Reconnaissance'** et '**DebutExecution'** dans les rectangles en lignes discontinues) et, dans ce même réseau, la transition **Gn St Sg** associée à la place **AttendEvenement** modélise un générateur aléatoire stochastique. Lorsque cette transition est franchie après un temps qui suit une loi aléatoire exponentielle (voir code associé à la transition **Gn St Sg** dans le rectangle en lignes discontinues à proximité de la transition), un jeton de couleur (ou type) *ObjetAttribut* spécifié par l'expression en langage CPN-ML par **(objet,n,intTime())** est généré dans la place **FA** (au même moment les jetons **n+1** et **event** sont générés dans les

places **AttendEvenement** et **Valeurcourante**, respectivement). Le jeton dans **FA** est composé de trois attributs. Le premier, **objet**, symbolise la nature du fragment d'information arrivé dans **FA**. Le deuxième est la valeur du numéro d'arrivée et le troisième est la valeur de l'instant ou le jeton arrive dans **FA**.

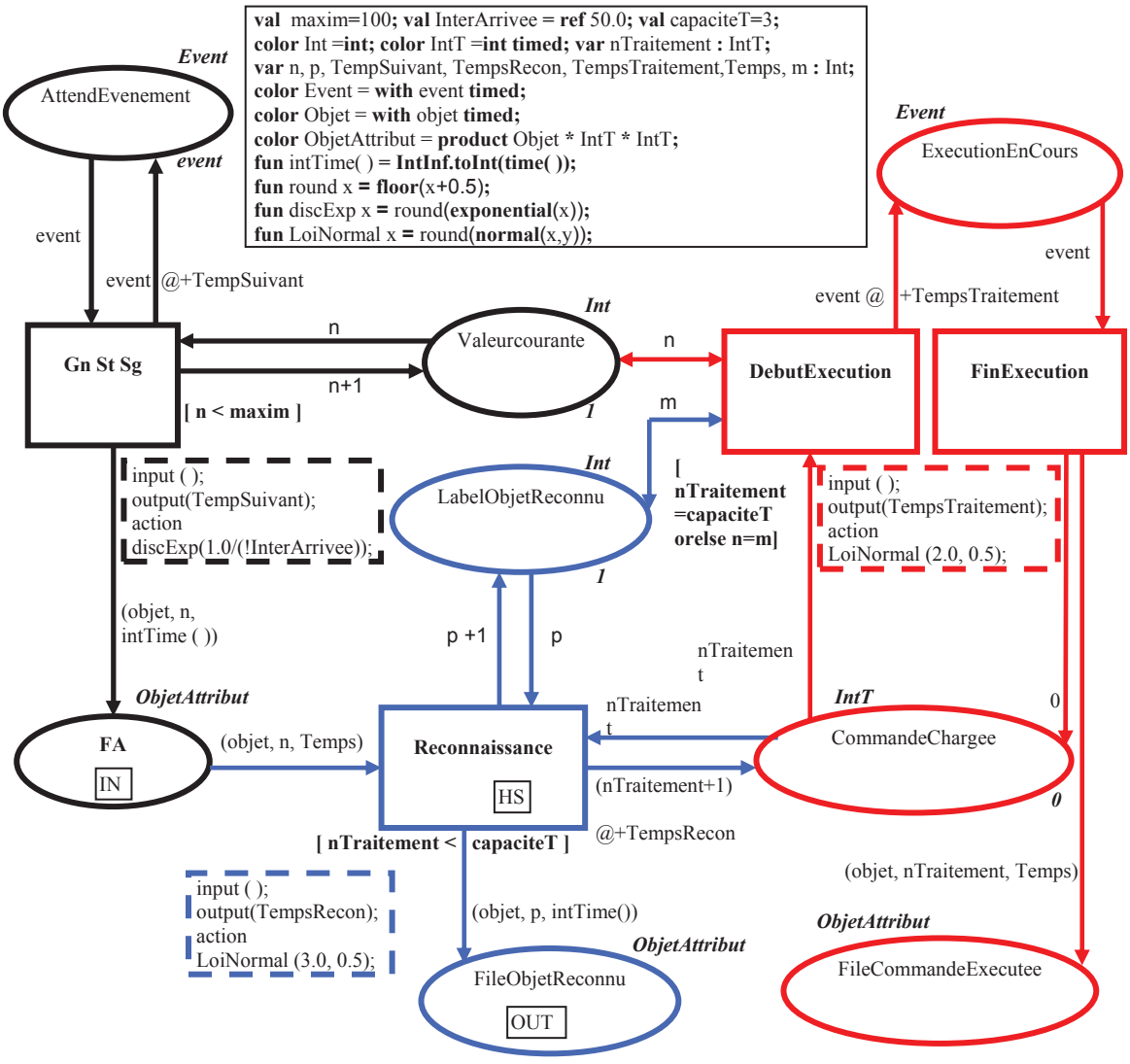

Figure 23 Générateur aléatoire de symboles, système de reconnaissance et capacité de traitement modélisés par RPCTS

Ces modélisations d'événements et d'activités avec des temps aléatoires constituent des aspects stochastiques dans les modèles en RPCT présentés dans ce document.
Il est facile de déduire des modèles de RPCT de fissions à partir des modèles de RPCT de fusions vu que la fission constitue, du point de vue modèle, le processus inverse de celui de la fusion.
Des propriétés architecturales importantes pour la généricité de l'AL sont mises en évidences par la vue multiagent de la Figure 22. Elles sont présentées au paragraphe suivant.

3.5 Caractéristiques génériques proposées pour l'AL

Architecture de Fusion/défusion/fission : Cette vue générique de l'AL permet aussi bien de spécifier la fusion que la défusion (phénomène inverse qui permet de 'défusionner' des symboles -Figure 24 (a)-.) La défusion est réalisée par les ACP et/ou les AdL et permet de gérer les annulations de commandes réalisées par l'utilisateur et de revenir sur le résultat intermédiaire d'une commande possible, au cours du processus de fusion (qu'il soit sémantique ou syntaxique.)
La fission est un processus qui consiste à transformer une information de sortie en deux ou plusieurs informations qui peuvent être complémentaires, redondantes, spécialisées ou équivalentes. La modélisation des fissions est identique à celle des défusions.

Architecture distribuée: Nous choisissons de distribuer l'ACP dans l'architecture, comme indiqué à la Figure 22. À chaque instance de l'ACP, sont associées les dimensions d'action, de perception et d'interaction, dépendamment de la localisation contextuelle de l'agent dans la structure architecturale. (Rappelons que la distribution est double : spatiale et fonctionnelle.) Le fait de distribuer cet agent permet de séparer les différents problèmes posés auxquels sera confronté l'agent. Cela permet aussi de décomposer la conduite globale de l'agent en comportements élémentaires.

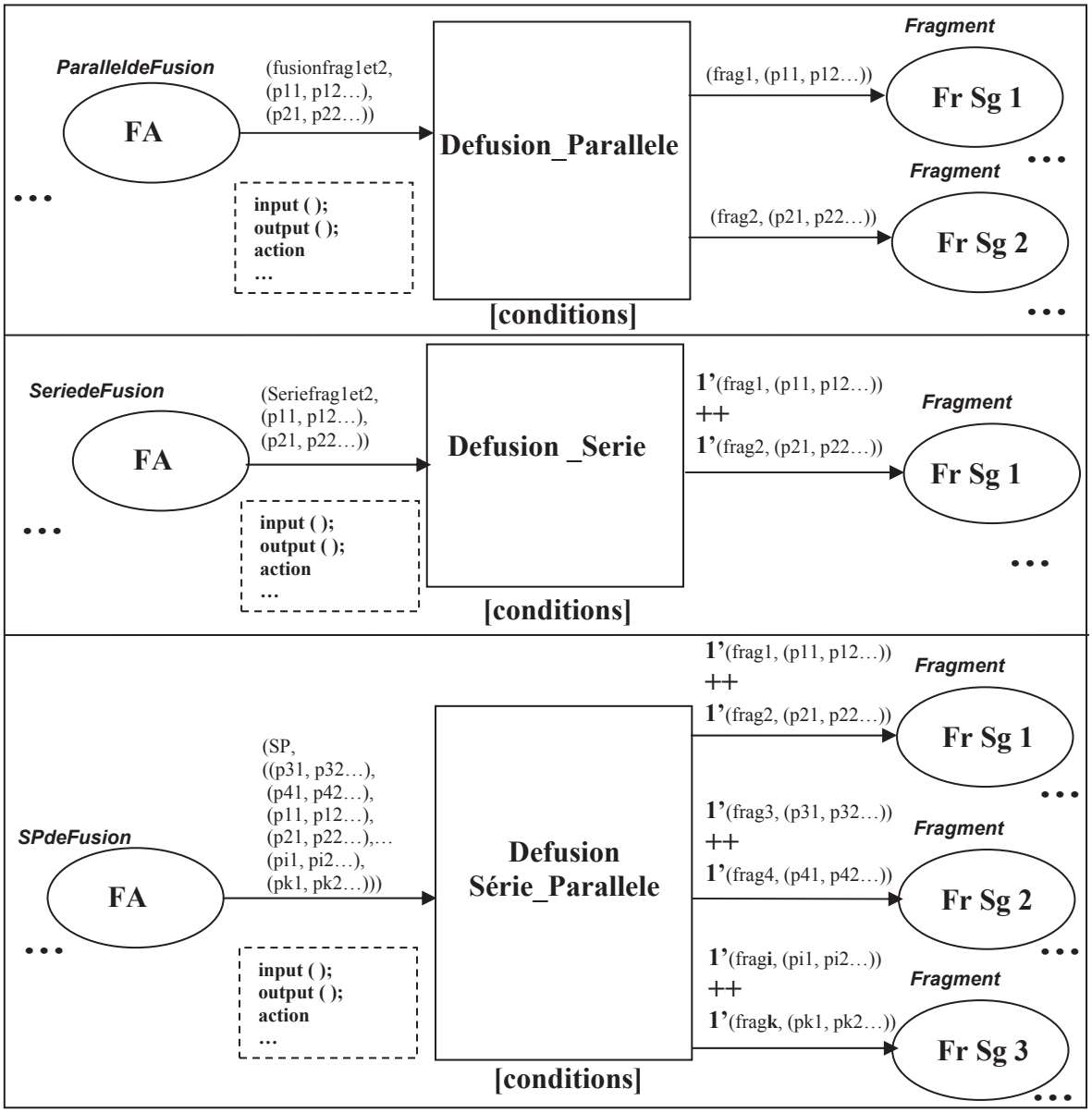

Figure 24 (a) Défusions multimodales : principes inverses des fusions parallèle, série et série-parallèle (FA : File d'Attente, pi: propriété i du fragment, Fr : fragment, Sg : Signal, fragi: fragment i d'information traitée, pi: propriété i du fragment)

Architecture extensible[26] et hiérarchisée: Pour garantir un aspect générique, l'architecture permet l'ajout de nouvelles modalités sans perturber le fonctionnement du reste du système. Pouvoir décomposer chacun des AdL en niveaux permet d'assister le développeur dans la génération du code par simple adaptation modulaire des nouveaux composants à des niveaux spécifiques.

Architecture Parallèle: Le Parallélisme offre la possibilité de faire fonctionner l'application en faisant en sorte que chaque AdL soit une entité gérée sur un système matériel séparé. Ainsi, il sera possible d'activer ou d'inhiber ces agents (dans le cas d'une reconfiguration architecturale dynamique) sans perturber le fonctionnement global de l'application au cours de son exécution.

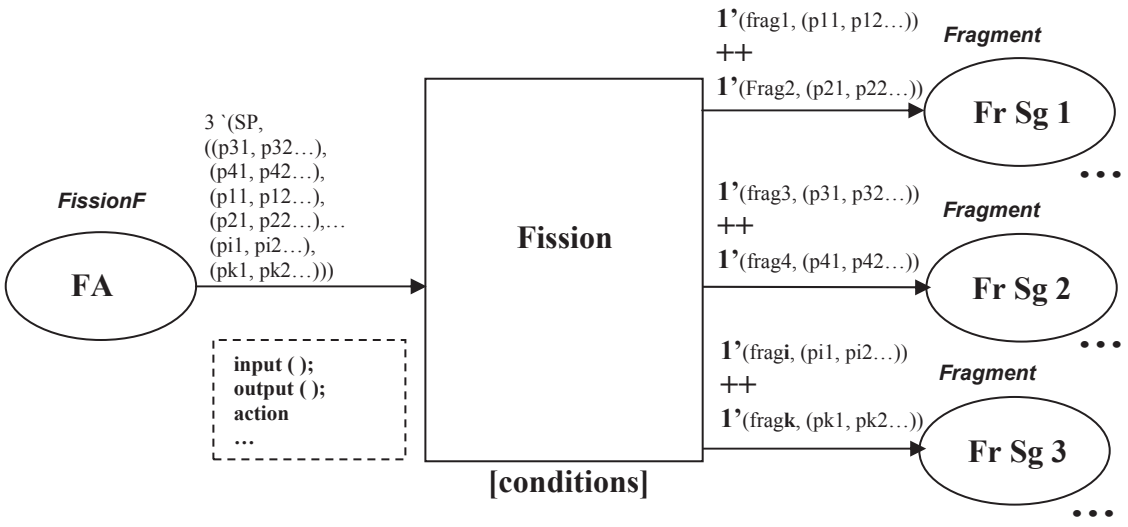

Figure 24 (b) Exemple de principe de la fission (cf. Figure 24 (a) pour la légende)

Architecture Pipe-line: Avec plusieurs entrées et plusieurs flux de données internes et externes au système, la tâche du contrôle et du suivi de l'évolution des données et des événements de cette architecture multimodale se trouve grandement améliorée en termes

de robustesse (la diminution et prise en charge du nombre d'erreurs de l'utilisateur durant le dialogue.) Les différentes instances des ACP peuvent prendre en charge le diagnostic de ces files de données pour prévenir les erreurs.
Une variante de la modélisation de la fission est donnée Figure 24 (b).

3.6 Gestion des erreurs

L'AL parallèle, distribuée, pipe-line, modélisée par RPCT, permet une gestion efficace des erreurs à différents niveaux du processus de fusion. Chaque file de signaux (ou fragments de signaux) est contrôlée. Le contrôle des erreurs peut être opéré directement par l'ACP ou par un autre agent assistant l'ACP dans ses tâches. Dans un souci de généricité et d'ergonomie pour le développement du code source, nous préconisons un RPCT dual du réseau responsable de la fusion/fission pour la gestion des erreurs. Le rôle de ce réseau sera de purger les files des signaux indésirables qui ne correspondent pas à une commande unimodale ou qui n'ont pas été fusionnés après un temps choisi par le concepteur. Ce temps est spécifié quantitativement dans le réseau à tous les niveaux d'intervention de l'agent de purge distribué. La complexité du réseau de gestion des erreurs peut égaler ou excéder celle d'un ACP. Ce réseau comporte aussi des agents responsables d'avertir l'utilisateur quand une fusion ou une commande est avortée et/ou lorsque le système ne comprend pas le message de l'utilisateur.
La structure architecturale proposée permet aussi d'employer deux modalités qui s'enrichissent mutuellement de leurs données sémantiques ou syntaxiques pour réduire l'ambiguïté du message de l'utilisateur. L'AL proposée procure un contexte permettant ce genre d'interaction durant le processus d'intégration. Durant l'intégration chacun des AdL interprète les attributs des données accumulées dans les files d'attentes. Simultanément, leurs alter ego réalisent la gestion des erreurs. La durée de l'intégration (fenêtre temporelle allouée pour attendre un signal complémentaire à celui déjà

[26] Traduction du terme anglo-saxon : '*scalable*'.

partiellement reconnu) est aussi un important critère directement spécifié quantitativement dans le RPCT qui sert à gérer les erreurs de l'utilisateur.
Cette information contextuelle est aussi employée par l'ACP distribué pour confirmer, auprès de l'utilisateur, l'exécution d'une commande. Cette confirmation est envoyée sur une modalité de sortie uniquement si l'agent a besoin de l'accord de l'utilisateur pour faire des prévisions. La structure de l'AL est donc très fortement adaptée à la gestion des erreurs de l'utilisateur.

3.7 Exemple de simulation

L'outil CPN-Tools (University of Aarhus 2006), permet de faire des simulations pour valider des scénarios de fusion multimodales. L'architecture de la fusion sémantique proposée est validée ici par l'exemple typique du 'copier coller.' Ce dernier implique un AdL de haut niveau pour la parole et un AdL rudimentaire pour le clic de la souris.
Nous donnons dans ce qui suit les caractéristiques de l'exemple modélisé.
Le Tableau VII donne symboliquement le vocabulaire et la grammaire employés par l'AdL de parole, dans cet exemple. Chaque mot du vocabulaire a une étiquette employée dans le réseau. Dans ce tableau, des expressions régulières symboliques sont employées pour représenter les éléments sémantiques. L'opérateur flèche dans l'expression (ouvrir→clic) signifie le mot prononcé 'ouvrir' suivi d'un clic de souris dans le temps. Les codes (Tableau VII) sont des étiquettes définies pour transporter les informations grammaticales et sémantiques utilisées dans le réseau.
Le mot 'annuler' est une commande qui annule la dernière action (mot prononcé ou dernière commande.) Selon un critère de proximité temporelle, le mot suivant la commande 'annuler' pourra être considéré ou non comme une nouvelle commande.
Le réseau associé au système modélise deux générateurs aléatoires correspondant aux arrivées des mots et des clics avec des intervalles de temps (inter-arrivée) suivant des lois exponentielles (fonction **ExpLaw** sur la Figure 25, en haut à droite et à gauche.)

Sur la Figure 25, les places et les transitions représentées en traits fins discontinus modélisent ces générateurs (**Gn St Sg clic** et **Gn St Sg MotDit**).

Le temps entre 2 clics (respectivement mots prononcés) arrivant successivement dans la file d'attente '**FA Clics**' (respectivement **'FA Mots'**) a une moyenne égale à '**ArriveeClic'** (respectivement '**ArriveeMot**'). Si la proximité temporelle entre un événement 'mot' et un événement 'clic' est inférieure à **TempsProxy** et si ces deux événements vérifient des conditions grammaticales et sémantiques ces deux événements sont alors fusionnés en une seule commande et une marque supplémentaire est ajoutée dans la '**FA Mediafusiones.'**

La déclaration des types (ou colorsets dans le jargon des RPCTS), variables et fonctions pour les réseaux des Figures 25 et 26 est donnée à l'Annexe 3.

Les transitions dessinées avec un trait épais discontinu modélisent les composants de l'ACP distribués sur tout le réseau. Les transitions et places en traits épais modélisent les AdL ou composants des AdL.

Toutes les places du réseau jouent le rôle de files d'attente ou de ressources communes. Les places portant une étiquette identique ne sont, en fait, qu'une et même place (une file ou ressource commune distribuée spatialement). Les places remplies d'une texture sont équivalentes à des files d'attentes avec autant de ports de sortie (respectivement d'entrée) qu'il y a d'arcs sortants (entrants) dans la place. L'ensemble du réseau est représenté sur les Figures 25 et 26. Les symboles IN et/ou OUT dans des places indiquent qu'elles jouent le rôle de ports d'entrée et/ou de sortie d'un niveau hierarchique à l'autre de l'architecture ainsi modélisée.

La simulation du réseau de Petri à l'aide de la boîte à outils (University of Aarhus 2006) permet de visualiser les différents états du système (marquages du réseau) au cours du dialogue multimodal. La Figure 27 donne une représentation synthétique des résultats obtenus pour des valeurs moyennes des inter arrivées des événements (clics ou mots) de 0,5s et une valeur du critère de proximité des évènements représentée par la variable **tempsProxy** = 1s (Figures 25 et 26).

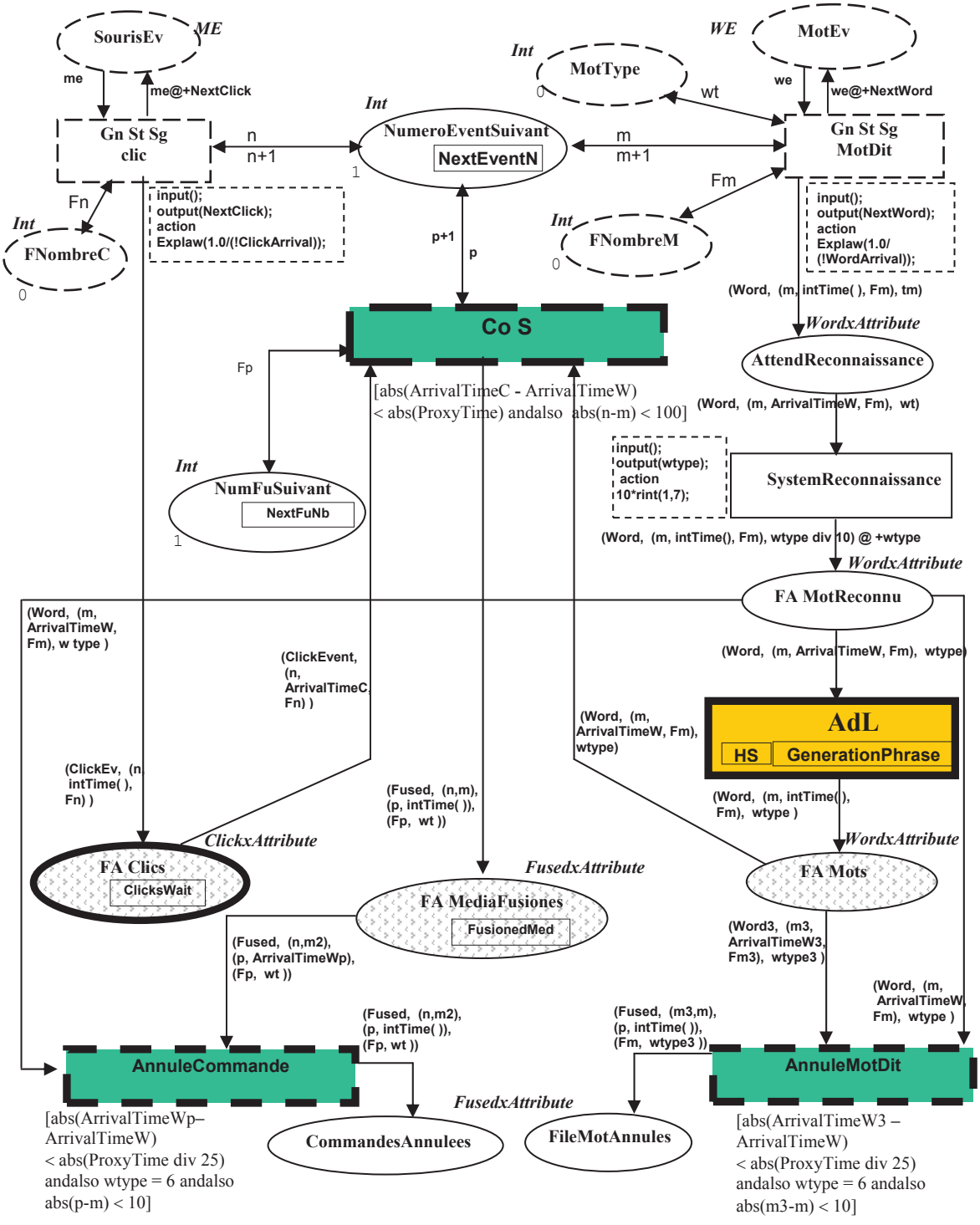

Figure 25 Premier niveau de l'architecture du dialogue bimodal (A : agent, AdL : agent du langage, Co : composant, FA : file d'attente, Gn : générateur, S : sémantique, St : stochastique, Sg : signal)

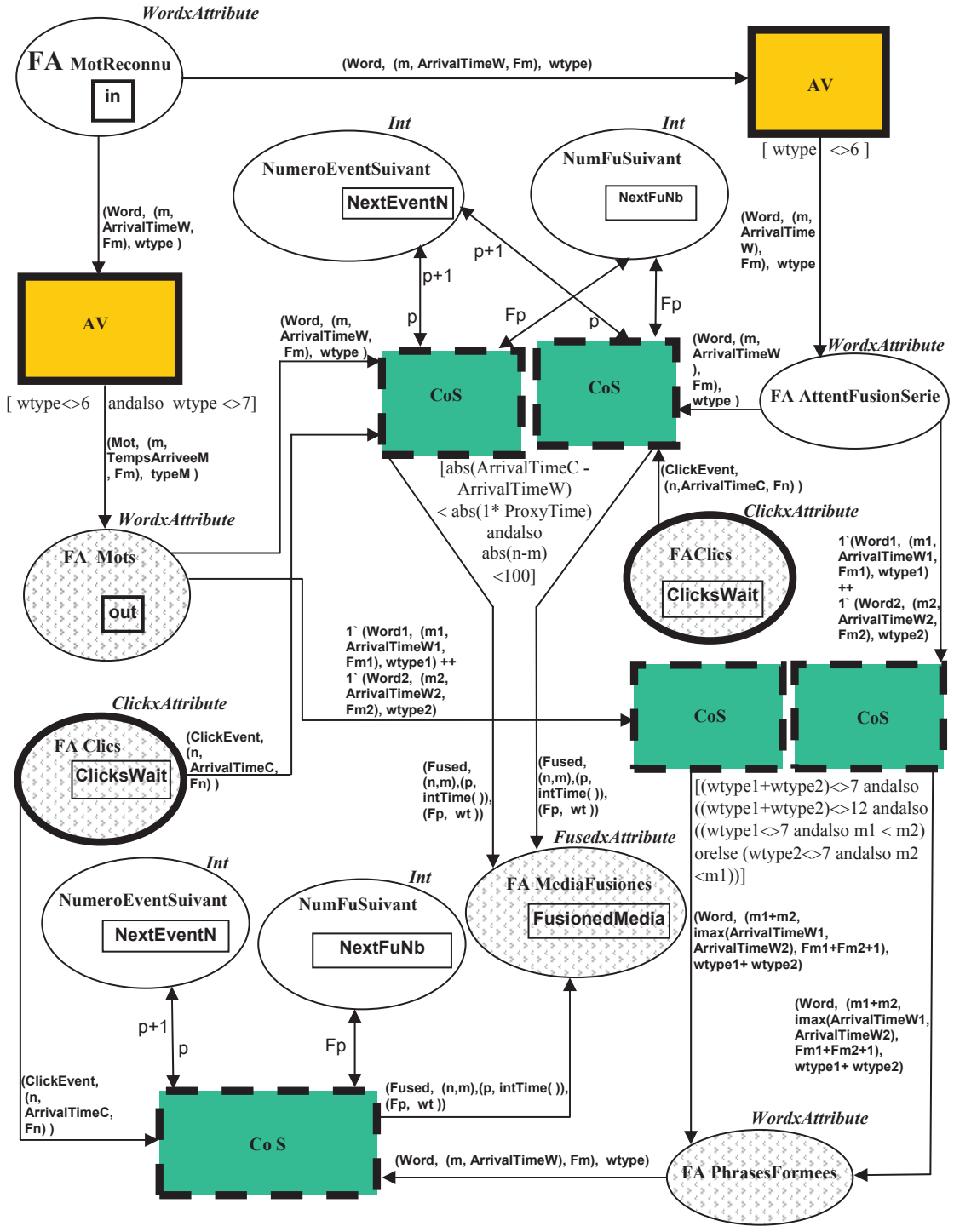

Figure 26 Second niveau du dialogue bimodal (A : agent, Co : composant, FA : file d'attente, S : sémantique, V : vocabulaire)

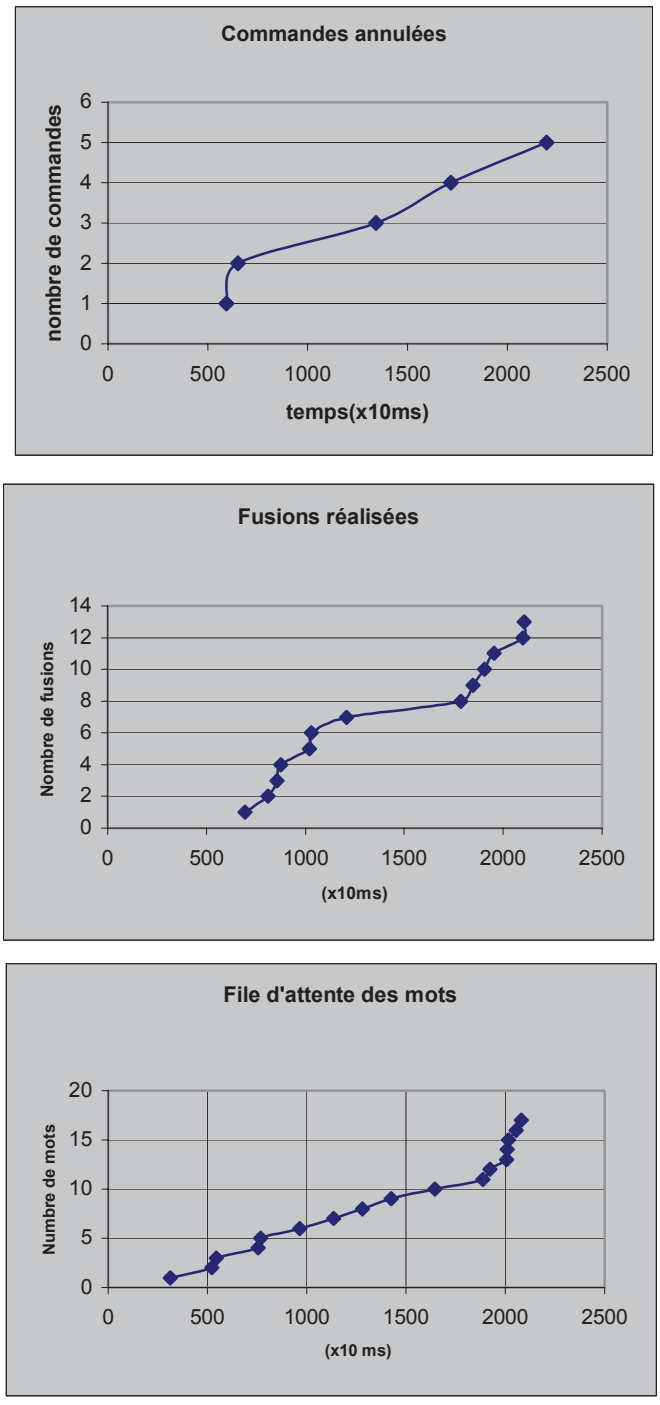

Figure 27 Résultats de la simulation

Les résultats de la Figure 27 permettent de suivre, d'un seul coup d'œil, l'évolution, au cours du temps, des différents événements : fusion, commandes annulées et mots prononcés Elle sert d'illustration et montre informellement (d'après l'allure des 3 courbes) l'existence d'une corrélation entre les événements 'mots dits', 'modalités fusionnées' et 'commandes annulées.' L'intérêt d'une modélisation par RPCT est multiple. Il permet en premier lieu de valider une structure architecturale de moteur de dialogue multimodal en tenant compte de la composante temporelle.

Tableau VII

Ensemble des phrases permises par la grammaire pour la modalité parole.

Ensemble de phrases	Signification de la commande	Ensemble des codes sémantiques correspondant	Ensemble des codes correspondant
{ (ouvrir→ça); (ouvrir) }	Ouvre l'objet désigné	{ (1→7); (1) }	{ (8); (1) }
{ (fermer→ça); (fermer) }	Ferme l'objet désigné	{ (2→7); (2) }	{ (9); (2) }
{ (effacer→ça); (effacer).}	Efface l'objet désigné	{ (3→7); (3) }	{ (10); (3) }
{ (coller) }	Colle l'objet désigné	{ (5) }	{ (5) }
{ (copier→ça); (copier) }	Copie l'objet désigné	{ (4→7); (4) }	{ (11); (4) }
{ (annuler)}	Annule la dernière commande	{ (6) }	{ (6) }

Le temps séparant l'arrivée de deux événements consécutifs, ainsi que le temps de traitement par un agent (ou franchissement d'une transition) est paramétrable dans le réseau. Ce paramétrage peut suivre des lois de probabilité (exponentielle, erlang, de Poisson, etc.) Par ailleurs les données sémantiques qui sont les éléments du dialogue (représentées par des marques colorées dans les places du RPCTS) portent une information temporelle, une information sur leur ordre d'arrivée dans une file, une information sur le degré de fusion ou de défusion dans la grammaire multimodale.

En second lieu, le RPCTS est propice à une modélisation répartie où chaque transition assume le rôle d'un agent spécialisé. La fonction de chaque agent est aisément

modifiable (changement des conditions de franchissement) et réglable (ajustement des valeurs de proximité temporelle.)

Par ailleurs, La modélisation d'une 'fission' (processus inverse de la fusion) est simple à mettre en œuvre car les couleurs des marques issues d'une fusion peuvent conserver toutes les informations nécessaires à cet effet.

Par exemple, sur les Figures 25 et 26, la donnée portée par l'arc entrant dans la place '**FA MediaFusiones**' comporte les numéros d'événements m et n correspondant à un mot prononcé et un clic, respectivement.

Enfin, de part sa nature, un RPCTS se prête au parallélisme et au traitement de la concurrence (tâches indépendantes en parallèle) présents dans les systèmes multimodaux.

3.8 Conclusion du chapitre 3

Nous avons présenté notre première proposition d'architecture logicielle multimodale multiagent générique, à base de réseaux de Petri colorés temporisés stochastiques. Nous avons mis en valeur ses atouts et caractéristiques architecturales les plus importantes notament en terme de réutilisabilité et ce pour tout système faisant intervenir l'interaction multimodale. Nous avons montré l'intérêt d'une modélisation de type 'agent' par RPCTS pour ce type d'architecture. Le choix d'une modélisation agent est d'autant plus pertinent pour la suite de nos travaux. En effet, la Figure 28 donne une représentation du principe du monitorage par un agent de la reconfiguration dynamique architecturale. Cet aspect de reconfiguration dynamique fera l'objet du deuxième aspect de notre approche. Nous allons donc aborder un aspect de la reconfiguration architecturale qui requiert l'emploi d'un agent expert externe à l'architecture (Figure 28) et dont les aspects fonctionnels et structurels seront détaillés.

Nous resterons donc dans le cadre de la modélisation 'agent.' Nous montrerons alors en quoi notre proposition est originale et se distingue des autres travaux concernant le dynamisme des AL (chapitre 5).

Mais au préalable nous présentons au chapitre 4 deux prototypes multimodaux développés selon la méthodologie proposée dans ce chapitre 3.

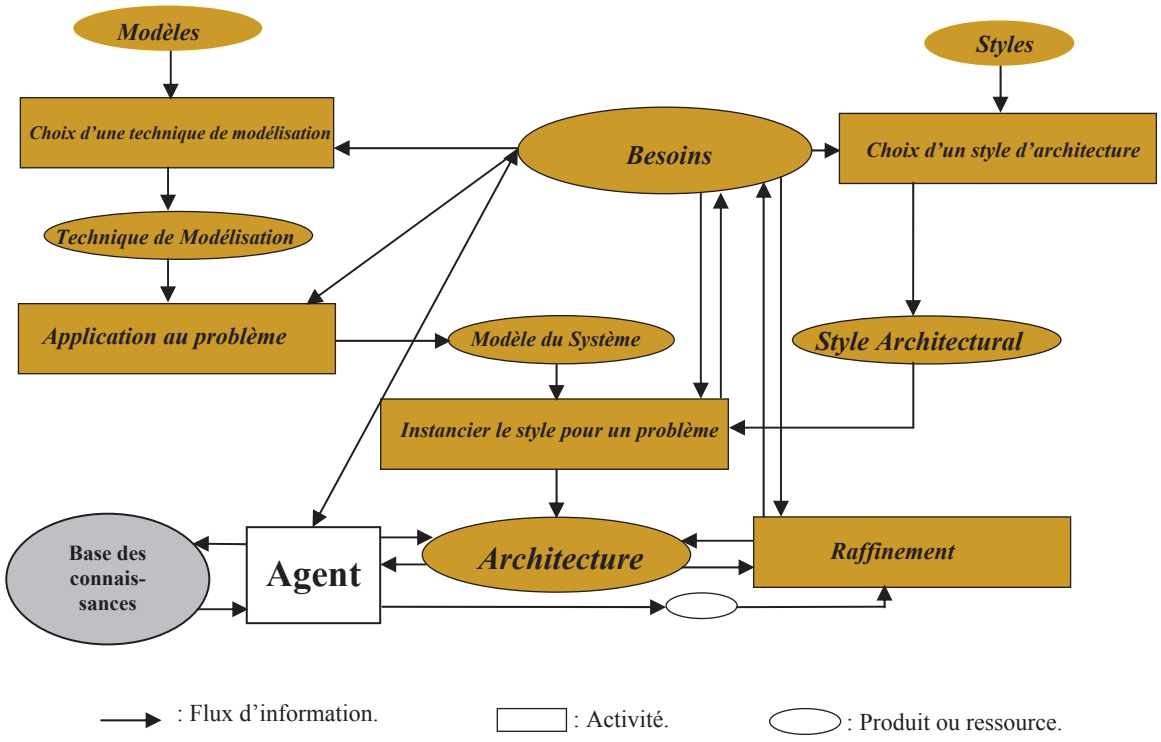

Figure 28 Réseau de Petri des phases du développement logiciel avec reconfiguration par agent

CHAPITRE 4

EXEMPLES D'APPLICATIONS : JEUX INTERACTIFS MULTIMODAUX

4.1 Introduction

Au chapitre 3, nous avons exposé une approche pour modéliser et spécifier formellement des AL tenant compte des critères et besoins des applications multimodales. La modélisation utilise le concept d'agent qui nous sera profitable lorsque nous aborderons la reconfiguration dynamique. Nous avons également montré comment raffiner formellement ces paradigmes architecturaux jusqu'à un niveau d'interaction multimodale souhaité par l'architecte (ce niveau peut être rafiné jusqu'au niveau lexème ou encore signal).

Dans ce nouveau chapitre, nous dévoilons des applications multimodales concrètes employant l'approche proposée. Ces applications sont des jeux interactifs multimodaux modélisés, spécifiés et développés dans le cadre de deux projets réalisés à l'École de Technologie Supérieure(Bisson 2006; Gagnon 2006). Les architectures des dialogues multimodaux de ces applications sont d'abord construites selon des modèles en RP qui font apparaître les éléments qui, à niveau d'abstraction élevé, vont intervenir dans l'interaction multimodale. Puis, ces RP sont raffinés en des RPCTS où les contraintes fonctionnelles temporelles et stochastiques ainsi que l'interaction dynamique sont mises en valeur. Les réseaux obtenus permettent de vérifier qu'il n'y a aucun blocage dans les dialogues mis en jeu. De plus, ces modèles d'AL offrent aux développeurs la connaissance de la liste des différentes fonctions ou procédures (les méthodes) mises en jeux dans l'application à développer et ce qu'elles doivent réaliser. Ces fonctions ou procédures correspondent aux activités des transitions dans ces réseaux. Ils donnent aussi la liste des attributs (les données et leurs types ou couleurs) manipulées par les transitions qui les récupèrent ou les génèrent dans des places. Également, les liens entre

les transitions via les places donnent des informations sur la façon dont sont liées ces fonctions et procédures et indiquent comment elles communiquent (la transmission ou la génération de l'information dans les réseaux est souvent conditionnée par des expression employant des opérateurs logique, de comparaison, etc.). Toutes ces informations permettent aux intervenants du projet logiciel de développer les applications à partir de modèles d'architectures en RPCTS. Ces deux applications ont été développées en langage évolué après que les différentes méthodes et attributs des classes ont été définis à partir des modèles en RPCTS. Les paragraphes 4.2 et 4.3 présentent ces applications et démontre comment, des architectures sont spécifiées et raffinées selon notre approche. Les exemples de jeux présentés ici n'ont aucune prétention commerciale. Ils constituent par contre des exemples didactiques qui illustrent les concepts du chapitre précédent.

4.2 TUX-XMEN

4.2.1 Présentation du jeu

Le projet suivant s'inspire d'une application de jeu non multimodal déjà existante le 'Mario Party' (http://marioparty.com/ 2006) et utilise une série d'images représentant des pingouins, personnages de Linux, communément appelés TUX[27] et issus du site (http://tux.crystalxp.net 2006). Les TUX peuvent être employés gratuitement dans un but non commercial. Nous voulions principalement développer un jeu :

a. multimodal : trois modalités en entrée et deux en sortie;
b. multi-utilisateurs : trois joueurs où le rôle des joueurs peut différer et où, les joueurs peuvent entrer en compétition les uns contre les autres;
c. permettant la fusion sémantique en entrée et la fission sémantique en sortie.

[27] Nous emploierons ce nom propre 'TUX' avec la même graphie invariablement au singulier comme au pluriel.

L'application développée est appelée TUX-X-MEN. Le but de ce jeu est de retrouver des pingouins appelés TUX déguisés en différents personnages des séries de supers héros 'X-Men', 'Batman', etc., des bandes déssinées 'Marvel's Comics' (Figure 29(a)).

Figure 29 (a) Exemples de personnages TUX déguisés en personnages de Marvel'Comics

Les TUX à trouver sont disposés dans une file d'attente par le maître du jeu. Les joueurs se déplacent sur une grille dont chaque case contient un personnage TUX déguisé. La grille se trouve sous la file d'attente et les joueurs peuvent s'y déplacer dans toutes les directions case par case à leurs guises. Il s'agit pour les joueurs de trouver les personnages TUX, un par un, le plus rapidement possible en se déplacant sur la grille et en se positionnant sur le TUX recherché. Ce dernier est le premier personnage se trouvant dans la file d'attente. Dès qu'un joueur trouve (avant son adversaire) le premier TUX choisi par le maître du jeu, il marque un point. La recherche se poursuit alors pour le prochain TUX de la liste choisie par le maître de jeu. Le maître du jeu peut à sa guise perturber la recherche des deux joueurs par différentes actions multimodales.

Le maître du jeu emploie le microphone et la souris et les deux joueurs sont au clavier.

Un des joueurs utilise le pavé numérique du clavier (à droite du clavier) tandis que l'autre se positionne à gauche du clavier et utilise des lettres du clavier pour jouer.

La modélisation par réseau de Petri de l'interaction multimodale donnée plus loin dans ce chapitre peut être étendue à plus de trois joueurs.

Il est, en effet, possible d'augmenter le nombre de joueurs et aussi avoir plusieurs maîtres de jeux. La modélisation par réseaux de Petri deviendra alors plus que nécessaire

si nous souhaitons éviter les cas de blocages dans l'interaction multimodale. Les preuves formelles que permettra une modélisation par Réseau de Petri conduiront alors à un développement sur des bases saines.

Cependant, pour le cas présenté ici nous nous sommes limités à un exemple simple composé de trois joueurs pour illustrer les principes méthodologiques présentés au chapitre précédent.

4.2.1.1 Architecture générale du jeu

La Figure 29 (b) représente l'architecture multimodale et les différentes interactions entre les périphériques de l'application.

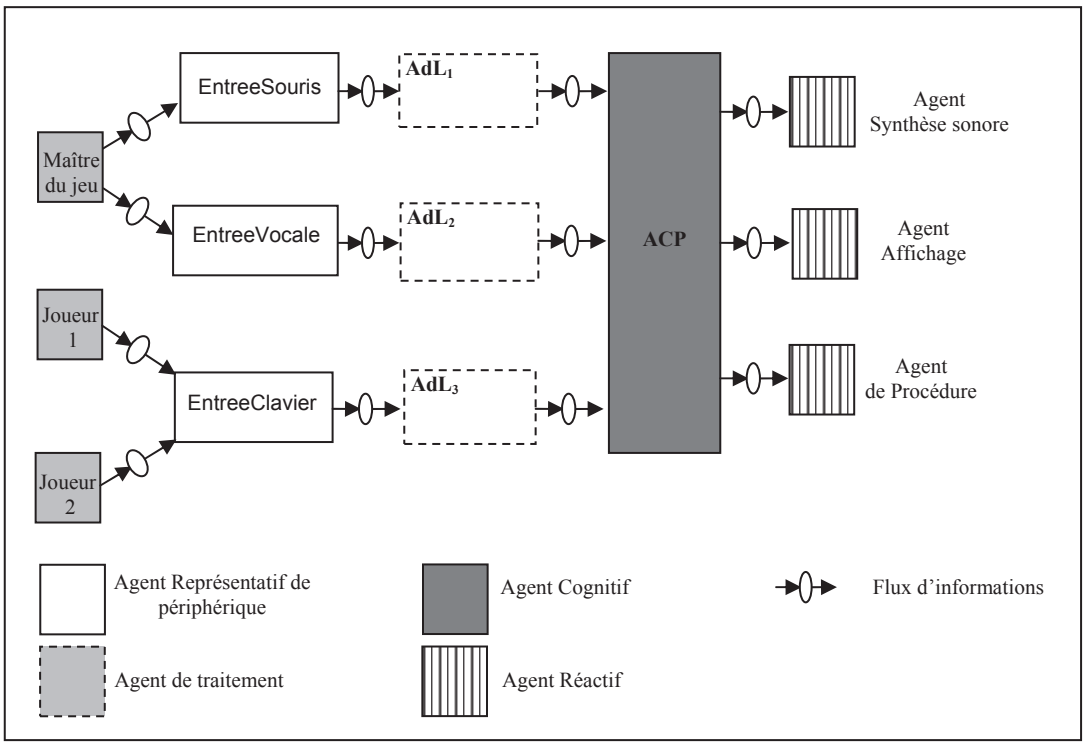

Figure 29 (b) Interactions entres les modalités

L'application permet donc d'employer les différents concepts vus au chapitre précédent, en intégrant par exemple la fusion de modalités combinant la voix et le clic de souris pour activer une fonctionnalité du jeu.
Trois AdL sont employés en entrée de l'architecture un ACP réalise l'interaction multimodale et la fission de l'information sur deux modalités de sortie (Figure 29(b)).

4.2.1.2 Requis système et outil de développement

Ce jeu a été développé en Visual Basic. Les requis système pour l'application sont :

a. Microsoft .Speech SDK 5.0
b. Microsoft Visual Studio 2005.

4.2.2 Fonctionnalités de TUX-XMEN

L'application offre plusieurs possibilités de jeu que nous appellons fonctionnalités. Les fonctionnalités sont donc activées par différentes entrées (le microphone, la souris et le clavier) pour ensuite générer des événements en sortie sur l'écran et sur des haut-parleurs. Il est à noter que l'utilisation des différentes commandes vocales nécessite l'emploi d'un préfixe qui est dans ce cas-ci, le mot « TUX ». Ceci permet d'éviter des ambiguïtés que le système de reconnaissance vocale pourrait générer.
En analysant le cheminement de l'information dans le réseau de Petri représentatif du raffinement de l'architecture globale de l'application (modélisée en Figure29 (b)), nous allons décrire plus en détails les fonctionnalités suivantes :

a. ajout d'un item dans la file du maître du jeu;
b. retrait d'un item dans la file du maître du jeu;
c. trouver un item de la file du maître du jeu;
d. activation d'une commande spéciale par le maître du jeu.

4.2.2.1 Ajout d'un item dans la file

Cette fonctionnalité est activée par le maître du jeu quand celui-ci veut ajouter un nouvel item dans la liste afin que cet item soit par la suite recherché par les joueurs. L'activation de la fonctionnalité se fait par la voix (Microsoft Speech SDK 5.1, une API gratuite, est employée pour la reconnaissance et la synthèse vocale). La commande vocale est par la suite reconnue si celle-ci est contenue dans la grammaire. Nous pouvons voir à la Figure 30 le parcours que l'information de la commande vocale effectue dans le réseau de Petri. Pour ce faire, un générateur de type de commande a été modélisé dans le réseau afin de pouvoir simuler les trois types de commandes que peut effectuer le maître du jeu.

L'exemple de la Figure 30 détaille ce qui se produit lors de l'activation de cette fonctionnalité :

1. le maître du jeu dit clairement à l'aide du micro la commande 'TUX STORM';

a. un jeton est généré par la transition *EntreeVocale* et est inséré dans la place *FileVocale;*

2. l'application reconnaît la commande 'TUX STORM' comme étant une commande valide et de type 1;

a. le jeton est retiré par la transition *ReconnaissanceVocale* et est inséré dans la place *FileCommandeV;*
b. l'application insère le personnage 'TUX STORM' dans la liste d'items recherchés par les joueurs (fonctionnalité de l'*Agent Procédure* de la Figure 29 (b));

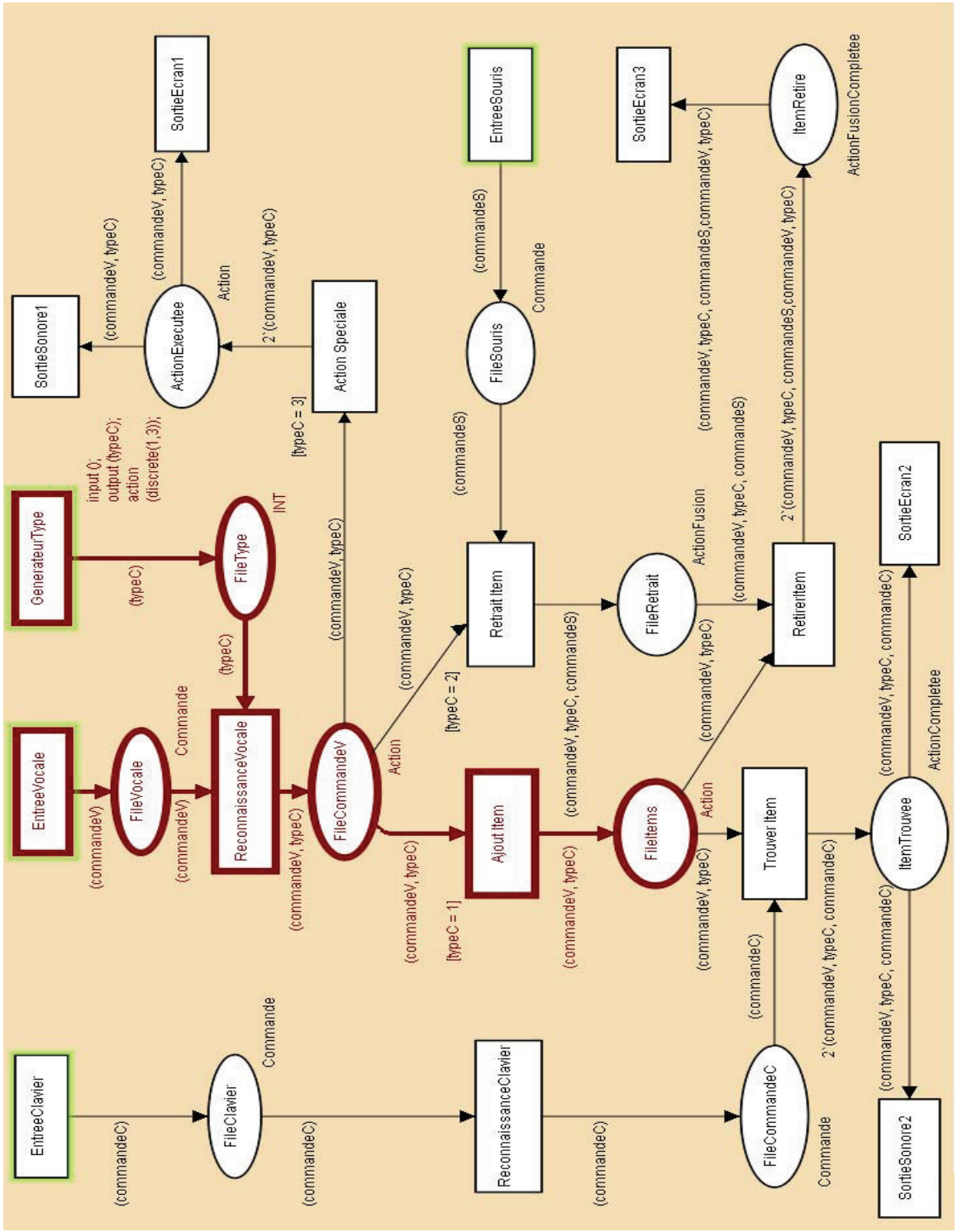

Figure 30 Ajout d'un item dans la liste (partie du réseau en trait gras)

c. le jeton est retiré par la transition *AjoutItem* et est inséré dans la place *FileItems*.

L'ensemble des différentes commandes vocales reconnues par la reconnaissance pour cette fonctionnalité est le suivant :

{ 'TUX STORM', 'TUX WOLVERINE', 'TUX MAGNETO', 'TUX ROGUE',
'TUX PROFESSOR', 'TUX CYCLOPS'}.

Lorsqu'une de ces commandes vocales est reconnue, l'*Agent procédure* ajoute le personnage correspondant à cette commande vocale dans la file des items recherchés par les deux joueurs.

Les déclarations nécessaires à la simulation du réseau de la Figure 30 sous CPN-Tools sont données à l'Annexe 4.

4.2.2.2 Retrait d'un item dans la file

Cette fonctionnalité requière, au préalable, des ajouts dans la file des items recherchés par le maître du jeu. Elle permet principalement à ce dernier d'enlever des items déjà existants dans la file, rendant ainsi le jeu un peu plus interactif avec les joueurs.

Cette fonctionnalité conduit à une interaction de deux modalités différentes : la reconnaissance vocale pour indiquer le désir de retirer des items de la liste et la souris pour pointer sur l'item qui doit être précisément retiré. L'ACP effectue donc la fusion sémantique des signaux en provenance de ces deux modalités différentes.

Nous présentons à la Figure 31 un exemple détaillé de ce qui se produit lors de l'activation de cette fonctionnalité.

1. Le maître du jeu clique sur un des items de l'interface de jeu représentant un des TUX affiché dans la liste des TUX à identifier (liste des items à rechercher par les joueurs) et dit clairement (dans le microphone) la commande : 'TUX REMOVE'.

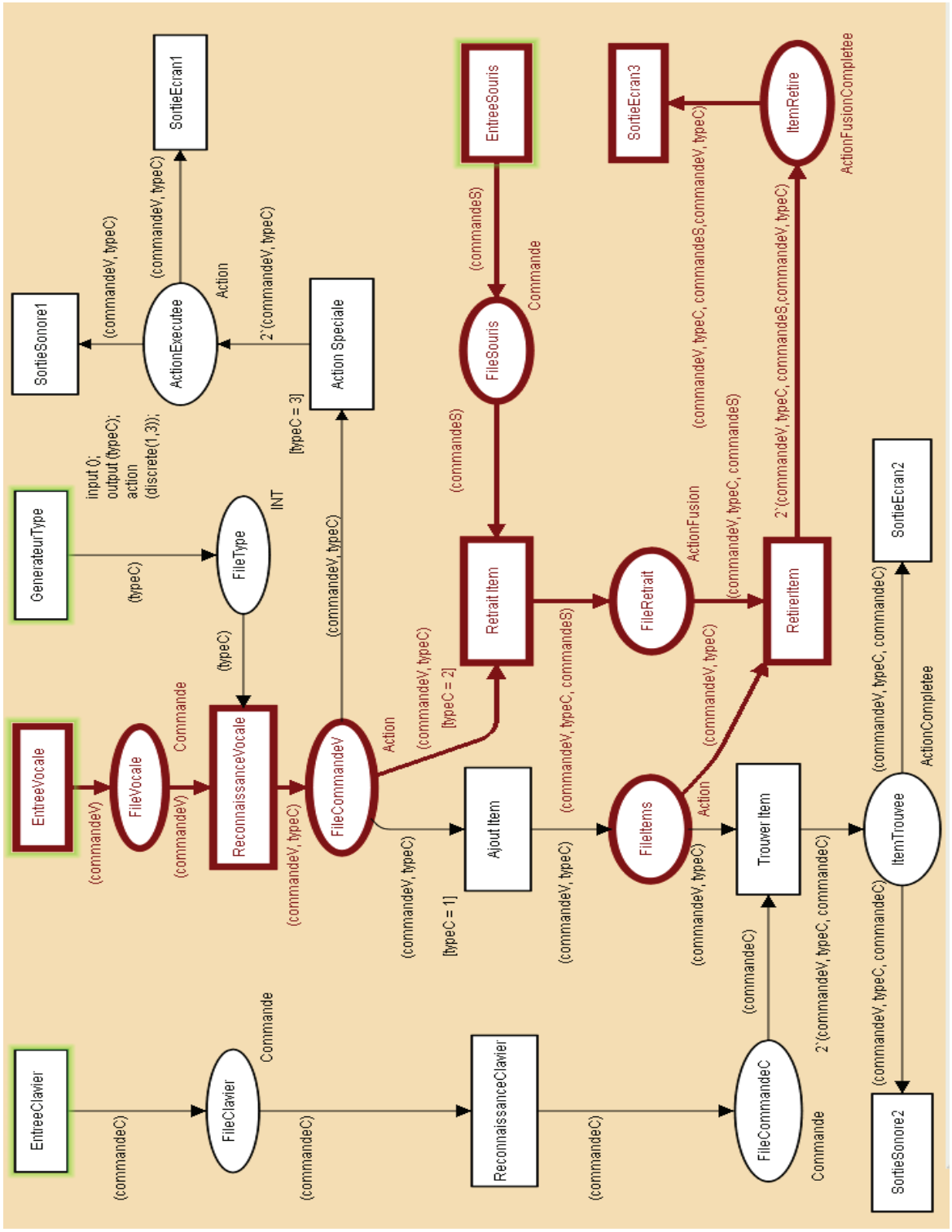

Figure 31 Retrait d'un item dans la liste (partie du réseau en gras)

a. Un jeton est généré à partir de la transition *EntreeVocale* et est inséré dans la place *FileVocale*.
b. Un jeton est généré à partir de la transition *EntreeSouris* et est inséré dans la place *FileSouris*.

2. L'ACP reconnaît la commande 'TUX REMOVE' comme étant une commande valide de type 2.

a. Le jeton de la place *FileVocale* passe par la transition *ReconnaissanceVocale* et est inséré dans la place *FileCommandeV*.

3. L'ACP retire le TUX sélectionné à l'aide de la souris dans la liste d'item recherché par les joueurs.

a. Le jeton de la place *FileCommandeV* et celui de la place *FileSouris* passent par la transition *RetraitItem* et sont combinés pour former un nouveau jeton dans la place *FileRetrait*.
b. Le jeton de la place *FileRetrait* et celui de la place *FileItems* passent par la transition *RetirerItem* et sont combinés pour former un nouveau jeton dans la place *ItemRetire*.
c. Le jeton de la place *ItemRetire* passe par la transition SortieÉcran2.

La commandes vocale reconnues pour cette fonctionnalité est 'TUX REMOVE' : L'*Agent Procédure* retire un TUX sélectionné avec la souris dans la liste des items recherchés.

4.2.2.3 Trouver un item de la liste

Cette fonctionnalité constitue l'objectif principal des joueurs. Elle permet aux joueurs qui sont positionnés sur le bon TUX d'utiliser les touches S ou 5 du clavier pour

indiquer qu'ils ont trouvé le TUX dans la grille de jeu. Afin de faciliter la compréhension du réseau de Petri nous avons omis de représenter la gestion du positionnement des joueurs dans la grille par rapport au TUX entré par le maître du jeu. Une fois le TUX trouvé dans la grille, l'ACP envoie un message sonore indiquant quel joueur vient de marquer le point, le pointage est mis à jour sur l'écran et le TUX est simultanément enlevé de la liste.
Voici le détail (Figure 32) de ce qui se produit lors de l'activation de cette fonctionnalité.

1. Un des joueurs indique, en appuyant sur la touche « S » ou « 5 », qu'il a trouvé le TUX recherché. Un jeton est créé à partir de la transition *EntreeClavier* et est inséré dans la place *FileClavier.*

2. L'ACP reconnaît la commande du clavier comme étant une commande valide. Le jeton 'passe' la transition *ReconnaissanceClavier* et est inséré dans la place *FileCommandeC.*

3. L'ACP valide que le positionnement du joueur correspond au premier TUX dans la file d'items (celui présentement recherché). Le joueur marque des points et le TUX est enlevé de la liste d'items.

a. Le jeton de la place *FileCommandeC* est fusionné au jeton de la place *FileItems* et ils passent par la transition *TrouverItem* pour être insérés dans la place *ItemTrouvé* sous la forme de deux jetons.
b. Les deux nouveaux jetons de la place *ItemTrouvé* 'passent' respectivement par les transition *SortieSonore2* et *SortieEcran2* pour réaliser la fission multimodale en sortie via la sortie écran et la sortie sonore.

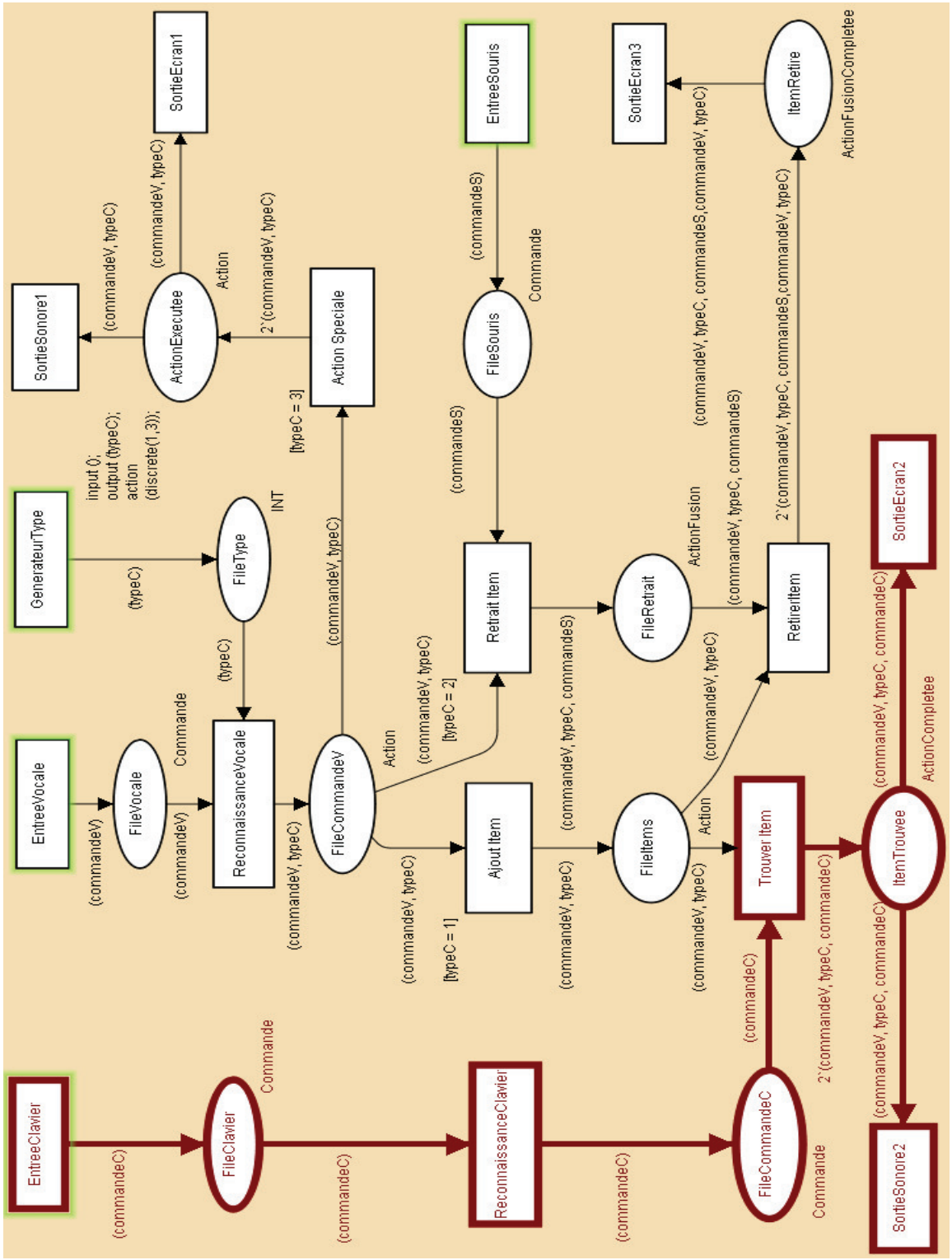

Figure 32 Trouver un item dans la liste (partie du réseau en trait gras)

La Figure 33 présente les deux (2) interfaces physiques du clavier employées par les deux joueurs. Les différentes commandes de type clavier reconnues par la reconnaissance pour cette fonctionnalité sont les suivantes : S et 5 qui permettent d'indiquer que le TUX trouvé est sélectionné.
Il est à noter que les commandes de déplacement n'ont pas été intégrées au réseau de Petri, puisque celles-ci n'interagissent pas avec les autres fonctionnalités directement.
Le joueur 1 peut se déplacer avec les touches Q-W-E-A-D-Z-X-C pour reproduire le pavé numérique 7-8-9-4-6-1-2-3.

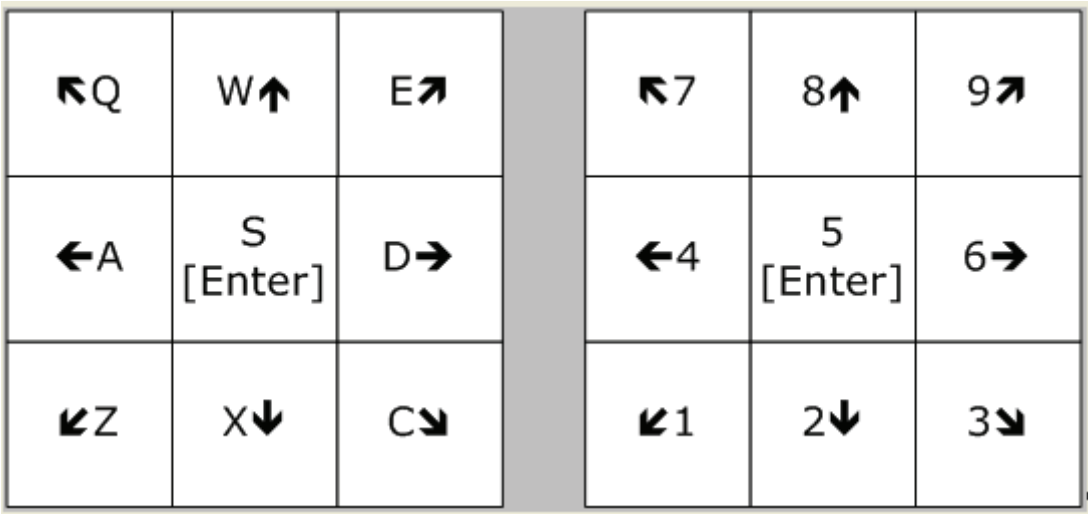

Figure 33 Touches utilisées par les joueurs

En utilisant par exemple la touche W ou 8 nous pouvons nous déplacer d'une case vers le haut dans la grille de jeu.

4.2.2.4 Activation d'une commande spéciale

Cette fonctionnalité regroupe toutes les commandes autres que le maître du jeu peut utiliser, mais qui n'ont pas d'effet sur la liste d'items.

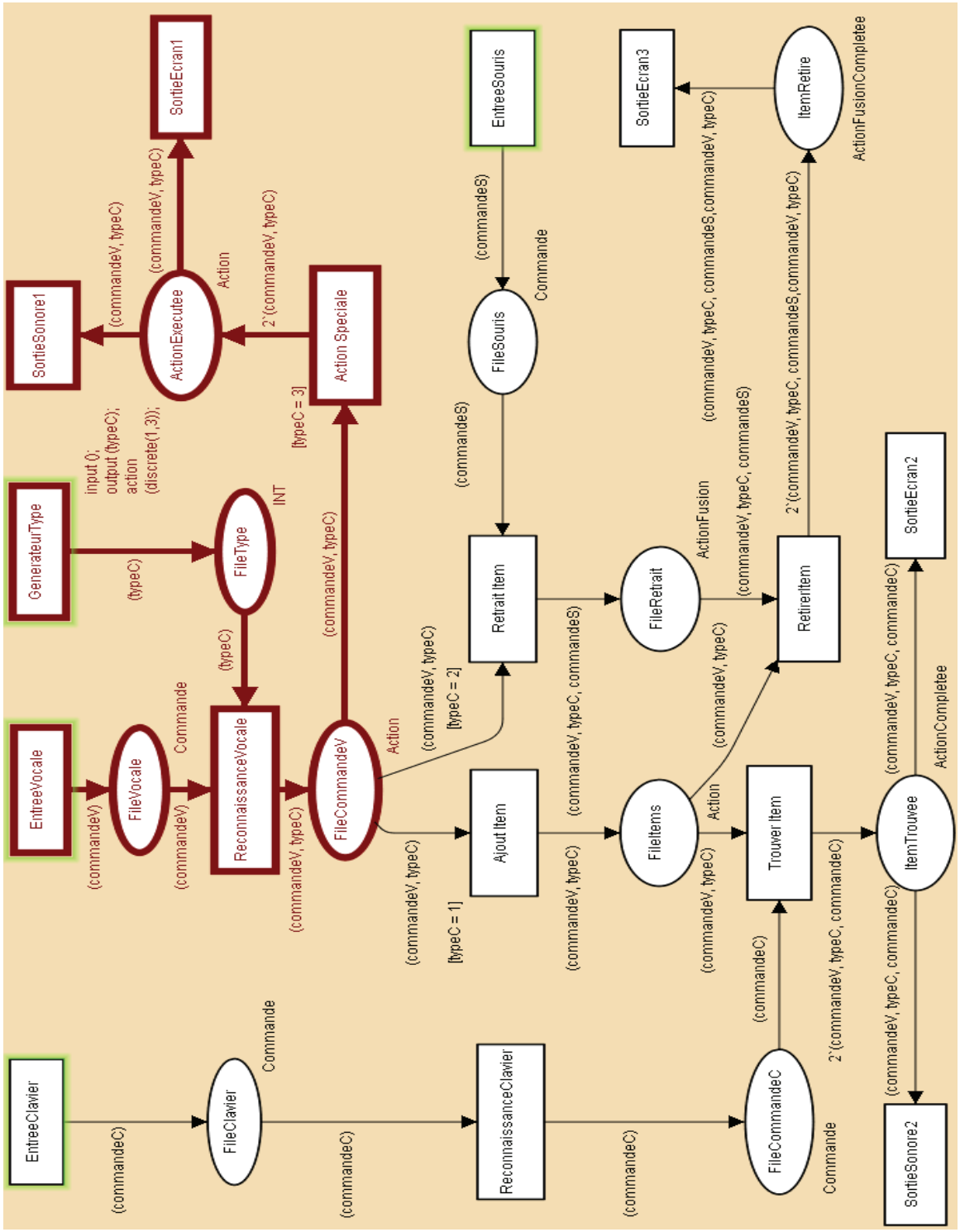

Figure 34 Activation d'une commande spéciale (partie pleine du réseau)

Nous pouvons par exemple, utiliser la commande ‘TUX SPIN’ pour mélanger les items dans la grille et ainsi rendre la partie plus difficile pour les joueurs.
Voici un exemple détaillé (Figure 34) de ce qui se produit lors de l’activation de cette fonctionnalité :

1. Le maître du jeu utilise le micro et prononce une commande vocale valide comme ‘TUX SPIN’. Un jeton est créé à partir de la transition *EntreeVocale* et est inséré dans la place *FileVocale* avec un *typeC =3*.

2. L’application reconnaît la commande comme étant une commande valide et elle interprète la commande.

a. Le jeton passe par la transition *ReconnaissanceVocale* pour être inséré dans la place *FileCommandeV*.
b. Le jeton passe par la transition *ActionSpeciale* et est inséré dans la place *ActionExécutée.*
c. Le jeton passe par la transition *SortieSonore1* et par la transition *SortieEcran1*pour réaliser une fission sur les deux modalités de sortie.

Les différentes commandes clavier reconnues pour cette fonctionnalité sont les suivantes :

a. ‘TUX BEGIN’ : Elle permet d’afficher la grille de jeu et de démarrer une partie (cette fonction active aussi le compteur sur le côté de l’interface). Cette fonctionnalité est aussi accessible via un bouton sur l’interface, à des fins de tests principalement.
b. ‘TUX SPIN’ : Elle permet de mélanger les TUX de la grille; c’est-à-dire que chacune des images sera déplacée au hasard à un autre endroit dans la grille.

c. ‘TUX PARTY’ : Idem à ‘TUX SPIN’.
d. ‘TUX VADER’ : Affiche une image du personnage ‘TUX VADER’ qui s’agrandit au fil du temps afin de bloquer la vue de la grille aux joueurs et joue une musique solennelle afin de perturber les joueurs.

e. ‘TUX XMEN’ : Affiche plusieurs TUX qui se déplace sur la grille afin de bloquer la vue de la grille aux joueurs.

4.2.3 Ergonomie et Interface de TUX-XMEN

L’interface de l’application est divisée en 5 sections distinctes (Figure 35). Les deux plus simples sont celles destinées à afficher les points des joueurs. La section à gauche, en rouge, affiche les points cumulés par le premier joueur. La section à droite, en vert, affiche les points cumulés par le second joueur. Les curseurs des deux joueurs portent les mêmes couleurs que leurs pointages respectifs afin de mieux les identifier lors du jeu. La section du haut est réservée au maître du jeu, elle est constituée d’une barre d’énergie, d’une liste de TUX et d’un TUX courant (celui qu’il faut trouver sur la grille). (Le bouton est implémenté à des fins de tests seulement, il n’est pas obligatoire de l’afficher). La barre d’énergie (disposée au dessus de la liste dans l’interface) augmente progressivement et diminue lorsque le maître du jeu active des commandes spéciales.
Il ne peut donc utiliser ces commandes que s’il possède assez d’énergie (cette condition est ajoutée à la reconnaissance vocale lors de l’activation des commandes spéciales). Ceci constitue une condition supplémentaire à respecter par le maître du jeu pour réaliser des commandes spéciales et permet de limiter ainsi son action sur les deux autres joueurs.
La liste de TUX est remplie par le maître du jeu. En utilisant la commande ‘TUX ROGUE,’ par exemple, le TUX ROGUE est ajouté dans la file d’items dans la section du maître du jeu (dans les boîtes carrées).

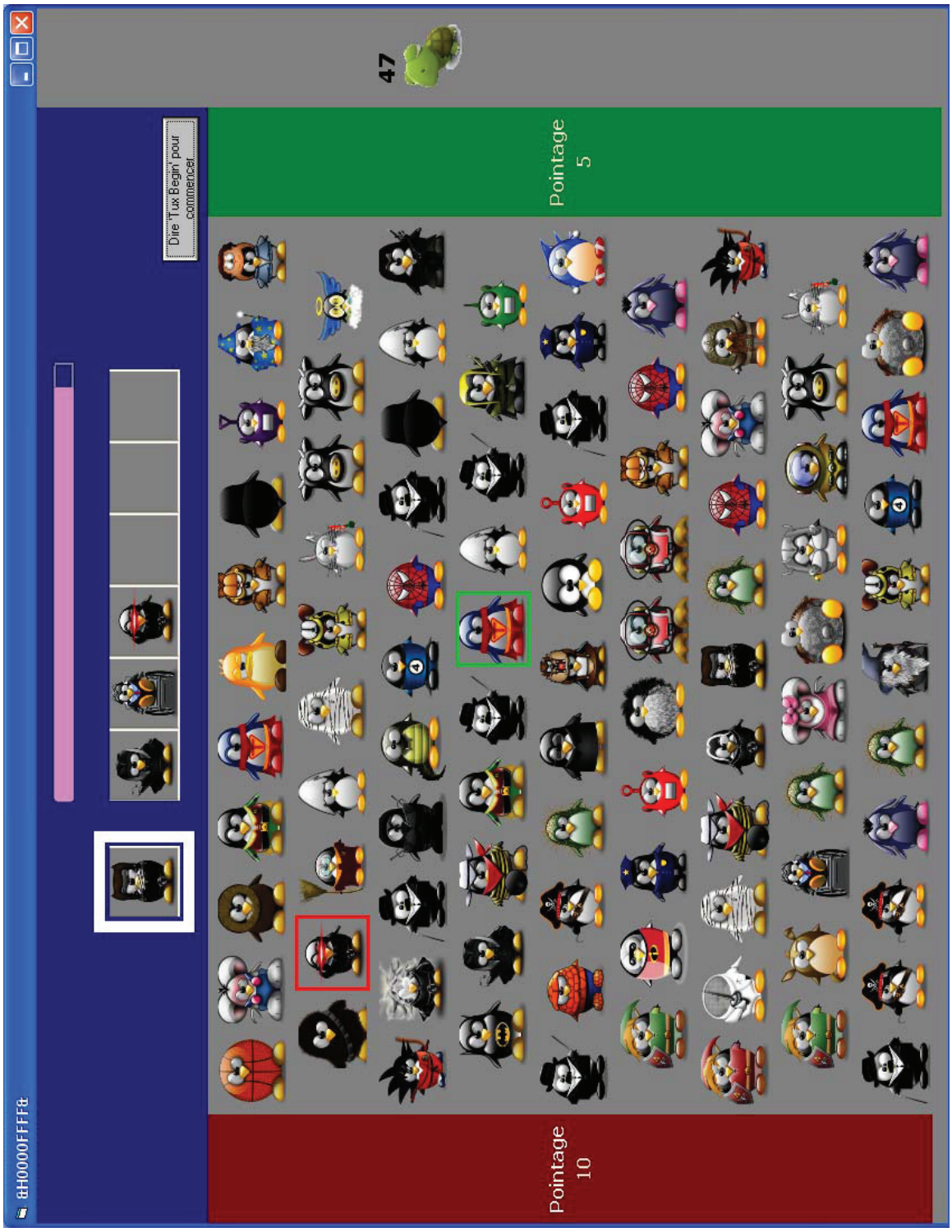

Figure 35 Interface de l'application TUX-XMEN

Le TUX courant est celui présentement recherché par les joueurs, le plus à gauche de l'écran dans un encadré plus prononcé. Lorsqu'il est trouvé, il est remplacé par le suivant dans la liste et chacun des TUX de la liste avance d'une case vers la gauche.La grille de jeu au centre affiche l'ensemble des TUX, et permet aux joueurs de se déplacer d'un TUX à l'autre pour se diriger vers celui recherché. Cette grille peut évidemment changer de contenu, à la discrétion du maître du jeu, pour rendre la partie plus amusante.

4.2.4 Conclusion

L'application multimodale TUX-XMEN constitue un exemple de la méthodologie développée au chapitre précédent. L'intérêt de cette méthodologie est pertinent dans cet exemple pour deux raisons. Elle permet de réaliser une AL spécifiée formellement par RPCT et rafinée jusqu'au niveau de l'interaction multimodale. Cette AL met en évidence sans ambiguité les phénomènes de fusion et fission qui entre en jeu dans les fonctionalités de l'architecture.
Dans les réseaux de Petri de cette aplication nous n'avons pas introduit la temporisation. Cette denière peut être facilement ajoutée au réseau modélisant le dialogue multimodal pour permettre au concepteur de tester des contraintes temporelles multimodales additionnelles avant même la phase du développement du logiciel.
L'exemple du paragraphe suivant prend en compte le temps et des aspects stochastiques dans les modèles formels de l'architecture qui y est proposée.

4.3 Jeux de mémoire

4.3.1 Introduction

Cette application emploie l'interaction personne/machine d'une manière multimodale avec comme modalités en entrée un système de reconnaissance vocale (moteur de voix conçu par Interop) et la souris. Les modalités de sortie sont la synthèse vocale et l'affichage de messages sur la fenêtre principale de l'application. Le but du jeu est

simple. L'utilisateur doit trouver chaque binôme pour chaque carte à jouer affichée sur la planche de jeu de la fenêtre principale de l'application. La planche de jeu est composée d'une matrice contenant 16 cartes à jouer. Donc, il existe huit binômes possibles. Une fois que le joueur a trouvé toutes les combinaisons, le jeu est terminé. Pour remplir les objectifs multimodaux, nous permettons à l'usager d'utiliser la parole, la souris et la combinaison de la voix et de la souris pour dicter les commandes à l'interface.

Nous débutons par le descriptif du guide d'utilisation du jeu. Le guide d'utilisation nous présente les requis du système et les fonctionnalités de base de l'application. Puis, nous présentons une analyse de cette application qui fournit les fonctionnalités du jeu via un raffinement de l'architecture globale présentée à la Figure 36.

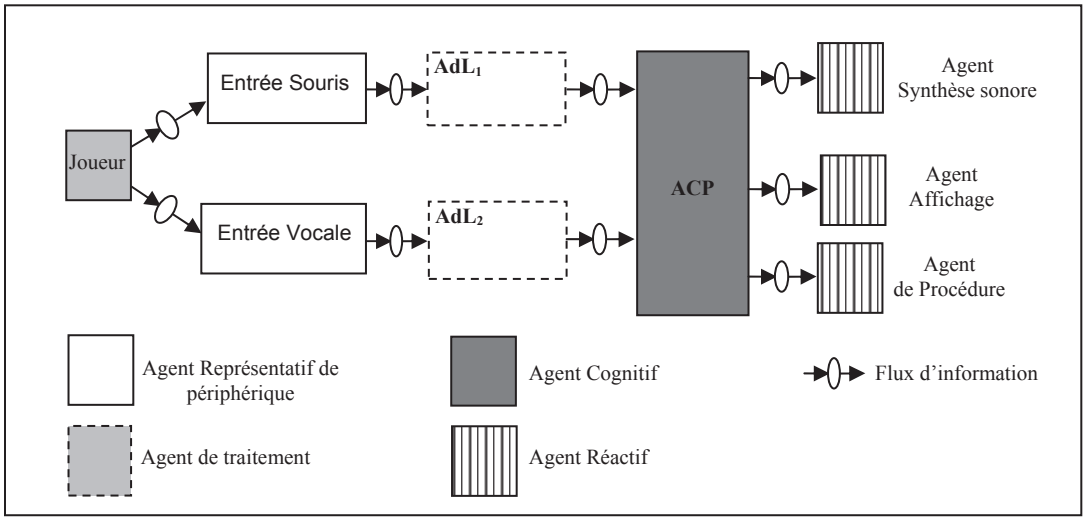

Figure 36 Architecture de l'application Jeu de mémoire

Nous décrirons donc successivement les fonctionnalités du jeu, le vocabulaire et la grammaire multimodale de l'application et enfin la modélisation par RPCT du dialogue multimodal (raffinement de l'ACP de la Figure 36).

4.3.2 Guide d'utilisation

4.3.2.1 Requis système et outil de développement

Ce jeu a été développé en Visual C++. Les requis système pour l'application sont :

c. La bibliothèque dynamique d'Interop;
d. L'outil Microsoft .NET framework 2.0
e. L'outil Microsoft Visual Studio 2005.

4.3.2.2 Comment utiliser l'application

Une fois exécutée, la fenêtre principale de l'application sera affichée comme le montre la Figure 37 (a). La fenêtre principale du jeu de mémoire est divisée en 3 sections :

a. La section supérieure est une section d'information sur l'état du jeu et sur l'état de la réception du micro.
b. La section *table de jeu* est composée d'une matrice (4 par 4) dans laquelle seront affichée les cartes à jouer lors du démarrage de la partie
c. Finalement la section inférieure est composée des boutons d'action.

Pour débuter une partie, l'utilisateur peut cliquer sur le bouton 'New Game' ou il peut dire la commande vocale « New Game ». Une fois le bouton pressé ou la commande vocale « New Game » reconnue la partie peut débuter. La Figure 37 (b) est un exemple de fenêtre affichée lorsque la partie est en cours.
La commande « New Game » initialise une matrice de cartes à jouer. Rappelons que le le but du jeu est de trouver les paires de cartes à jouer qui sont identiques.

Il est possible avec la souris de débuter une nouvelle partie, de retourner les cartes lorsqu'elles sont sélectionnées, d'annuler une commande et de quitter l'application. Ces commandes peuvent être exécutées avec l'aide de la parole.

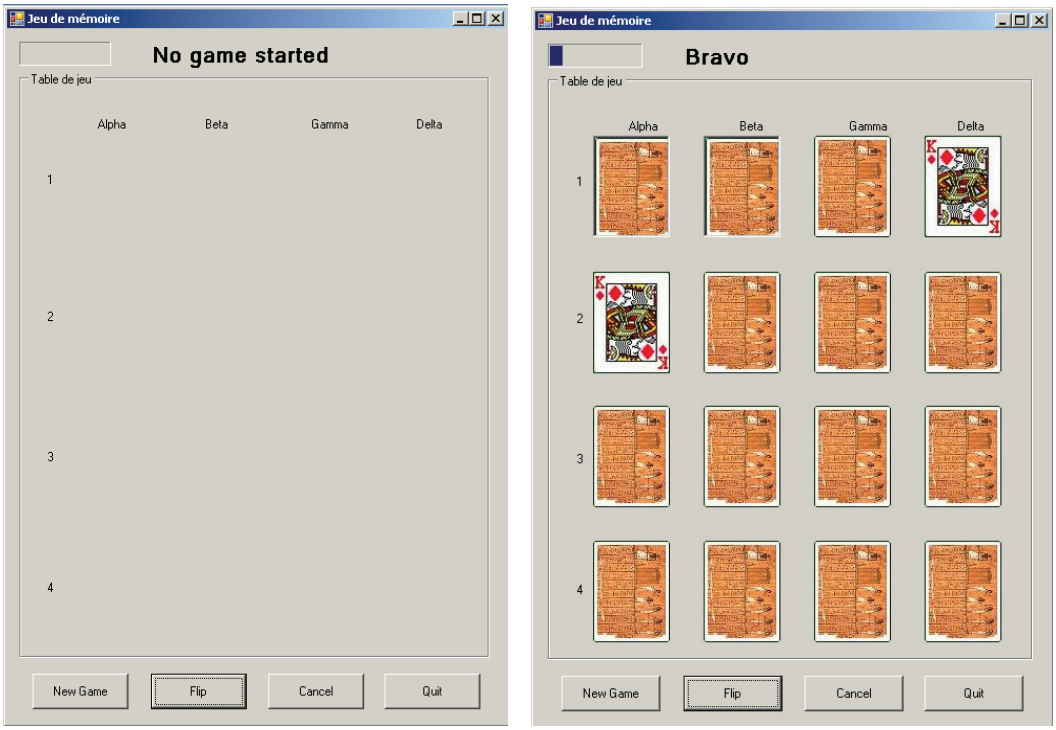

Figure 37 (a) Fenêtre principale du jeu de mémoire à gauche (b) Fenêtre lorsque la partie est en cours à droite

Par exemple en exprimant la commande vocale « Alpha one» le système sélectionne la carte à jouer qui est située à la ligne 1 et à la colonne alpha, cette opération peut être réalisée en cliquant sur la carte désirée.

Lorsque deux cartes sont sélectionnées, l'utilisateur peut exprimer la commande vocale « Flip » ou appuyer sur le bouton équivalent. Cette commande aura pour effet de retourner les deux cartes sélectionnées.

Si les deux cartes sélectionnées sont identiques elles resteront affichées. Par contre, si les cartes ne sont pas identiques, elles ne resteront affichées que quelques secondes pour

permettre à l'utilisateur de les mémoriser. Puis, après ces quelques instants, elles sont à nouveau automatiquement dissimulées.

Finalement, une fois toutes les cartes reconnues, l'utilisateur peut recommencer une partie ou quitter l'application. Par ailleurs, l'utilisateur peut en tout temps débuter une nouvelle partie ou quitter l'application en invoquant la commande vocale « New Game » ou « Close application ». Il peut également, lorsqu'une ou deux cartes sont sélectionnées, les désélectionner en invoquant la commande vocale « Cancel ». Toutes ces opérations peuvent être effectuées avec la souris en cliquant sur le bouton correspondant à la commande, « New Game » pour une nouvelle partie, « Cancel » pour annuler une sélection et « Quit[28] » pour quitter l'application.

4.3.3 Analyse

Dans cette section nous allons présenter l'analyse fonctionnelle nécessaire au développement de l'application.

4.3.3.1 Fonctionnalités du système :

L'utilisateur peut réaliser les différentes actions suivantes.

a. Débuter une partie : une fois l'application démarrée, l'utilisateur peut démarrer une partie en dictant la commande vocale « New Game » ou en appuyant sur le bouton 'New Game'; l'utilisateur peut à tout moment réinitialiser une matrice de cartes à jouer en cliquant sur le bouton ou en dictant la commande vocale.

[28] La commande vocale pour quitter le jeu « Close application » est différente du libellé du bouton correspondant « Quit », d'une part pour éviter que le système de reconnaissance vocal ne confonde le mot « Quit » avec le mot « Flip » et d'autre part parce que un libellé aussi long que « Close application » serait inesthétique sur le bouton de l'interface.

b. Annuler une sélection : Lorsque qu'une carte est sélectionnée dans la table de jeu, l'utilisateur peut la désélectionner en cliquant sur le bouton 'Cancel' ou en dictant la commande vocale « Cancel »; si deux cartes sont sélectionnées les deux cartes seront désélectionnées.
c. Quitter le jeu : pour quitter l'application, l'utilisateur peut dicter la commande vocale «Close application» ou cliquer sur le bouton 'Quit'; cette action peut être faite à tout moment durant le jeu.
d. Sélectionner une carte : pour sélectionner une carte l'utilisateur peut utiliser la souris ou utiliser une commande vocale composée de deux mots. Il doit, pour sélectionner une carte avec l'aide de la parole, dicter la colonne et la ligne où est située la carte. Cette opération peut aussi être effectuée avec la souris en cliquant sur la carte désirée.
e. Retourner les cartes : pour retourner les cartes et afficher leurs valeurs, l'utilisateur peut utiliser la commande vocale « flip » ou il peut cliquer sur le bouton « Flip »; deux cartes doivent être sélectionnées pour que la commande flip fonctionne.

Le réseau de Petri de la Figure 38 (a) décrit les actions possibles que l'utilisateur effectue sur l'application. Rappelons que lorsqu'une place du réseau comporte une étiquette, comme la place **Count** (portant l'étiquette **Total**), alors toutes les autres places portant la même étiquette représentent une seul et même place distribuée spatiallement dans le réseau. Ainsi, **Count**, **Count0**, **Count1**, **Count11**, **Count2**, **Count22**, **Count3** et **Counter** représentent une seule place ayant le meme marquage en tout temps.

1. Comme le montre ce réseau, après le lancement de l'application (tir de la transition **Game**) le système tombe en mode « **Wait** » (nom de la place) et donne à l'utilisateur deux possibilités :

a. quitter le jeu (tir de **QuitGame**);
b. commencer une nouvelle partie par une action qui distribue les 16 cartes retournées (tir de **NewSet**).

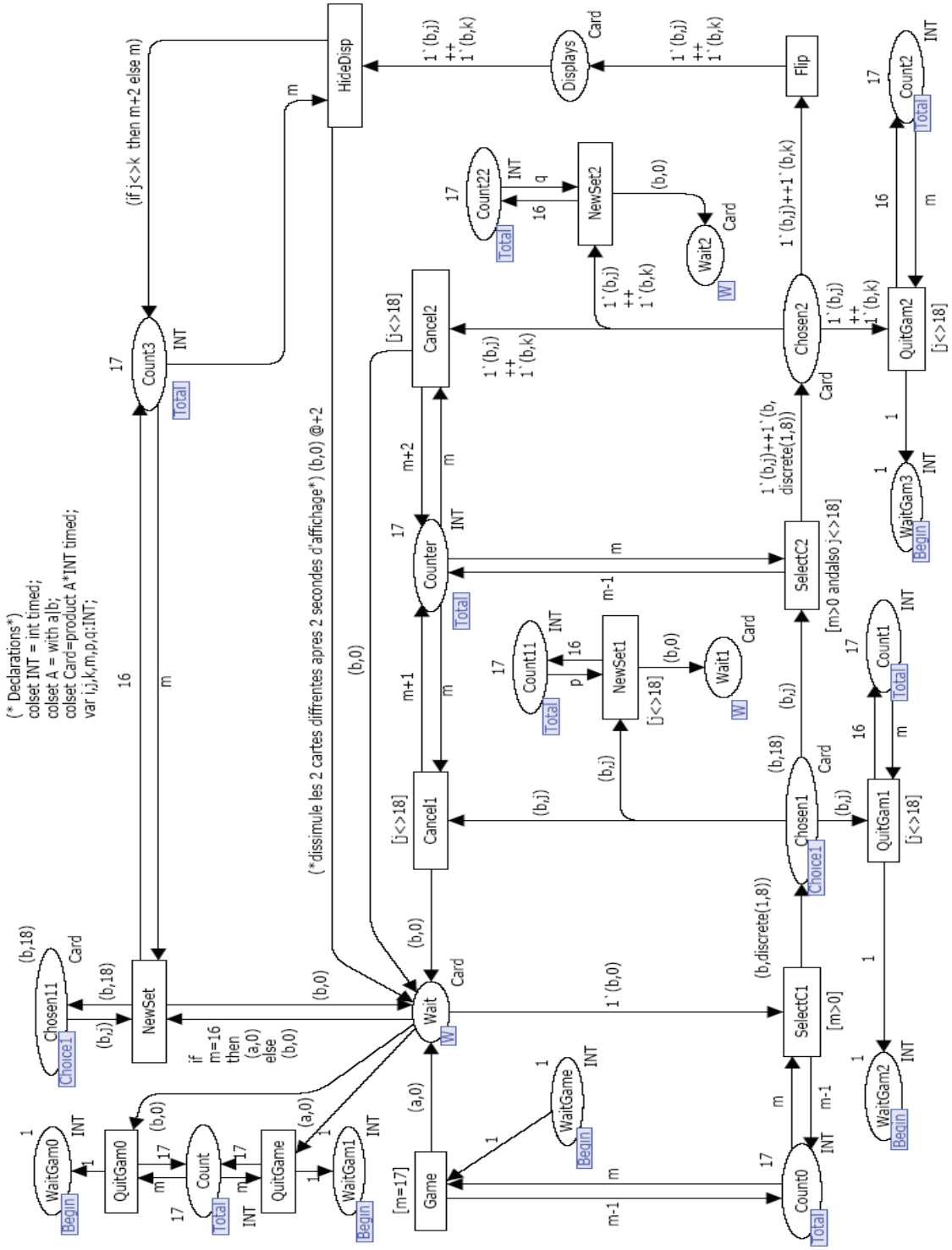

Figure 38 (a) Réseau de Petri temporisé de l'application jeu de mémoire

2. Une fois **NewSet** franchie l'utilisateur reste encore en attente (place **Wait**) et peut alors :

c. quitter le jeu (tir de **QuitGam0**)
d. recommencer la partie (tir de **NewSet1**)
e. sélectionner une carte (tir de **SelectC1**).

3. Si l'utilisateur sélectionne la carte (étape 2. e. précédente.) alors il se retrouve dans l'état **Chosen1** et peut :

f. quitter le jeu (tir de **QuitGam1**)
g. recommencer la partie (tir de **NewSet1**)
h. sélectionner une seconde carte (tir de **SelectC2**).

3. Si l'utilisateur sélectionne la carte (étape 2. h. précédente.) alors il se retrouve dans l'état **Chosen2** et peut :

i. quitter le jeu (tir de **QuitGam2**)
j. recommencer la partie (tir de **NewSet2**)
k. retourner les deux cartes (tir de **Flip**).

Si les deux cartes sélectionnées, sont affichées par l'utilisateur (état **Displays**). L'application les dissimule après un délai de 2 seconde (Symbole **(b,0) @+2** sur l'arc quittant la transition **HideDisp** de la Figure 38 (a)). Simultanément la valeur du jeton (de type entier) de la place **Count3** qui a été décrémentée d'une unité à chaque sélection de carte est incrémentée de deux unités par la transition **HideDisp** car les deux cartes sélectionnées sont justement différentes. Si les deux cartes sont identiques, alors le jeton reste de la place Count3 n'est pas incrémenté. Ceci est indiqué par l'inscription (**if j<>k then m+2 else m**) de l'arc reliant **HideDisp** à **Count3**.

Lorsque les deux cartes retournées ont la même valeur et que toutes les cartes sur la planche de jeu sont retournées, alors la partie est terminée. L'utilisateur se retrouve à l'étape 2 précédente. La Figure 38 (a) constitue la grammaire sémantique multimodale temporelle de l'application. Dans le cas où les deux cartes sont identiques elles restent retournées et le joueur peut continuer la partie.

Dans les états **Wait**, **Chosen1** et **Chosen2** le joueur peut à chaque fois quitter le jeu ou encore recommencer une nouvelle partie. La valeur du jeton de la place portant l'étiquette **Total**, est testée par différentes transitions pour, entre autres, limiter à 8 le nombre de paires de cartes retournées au cours d'une même partie. Notons qu'il est possible de se passer de la distribution spatiale des places pour tester le fonctionnement du réseau. Cependant, cela nuirait à la lisibilité du modèle architectural (voir Figure 38 (b)) qui n'est pas destiné uniquement à la vérification de l'interaction mais qui sert aussi comme base architectural pour développer l'application.

L'outil CPN-Tools permet une analyse automatique du réseau pour vérifier ses propriétés comportementales. À titre d'exemple, nous donnons à l'Annexe 5 le résultat de la vérification de ces propriétés pour le réseau de la Figure 38 (a).

Nous donnons dans la suite les RPCTS complets de cette application modélisés avec CPN-Tools.

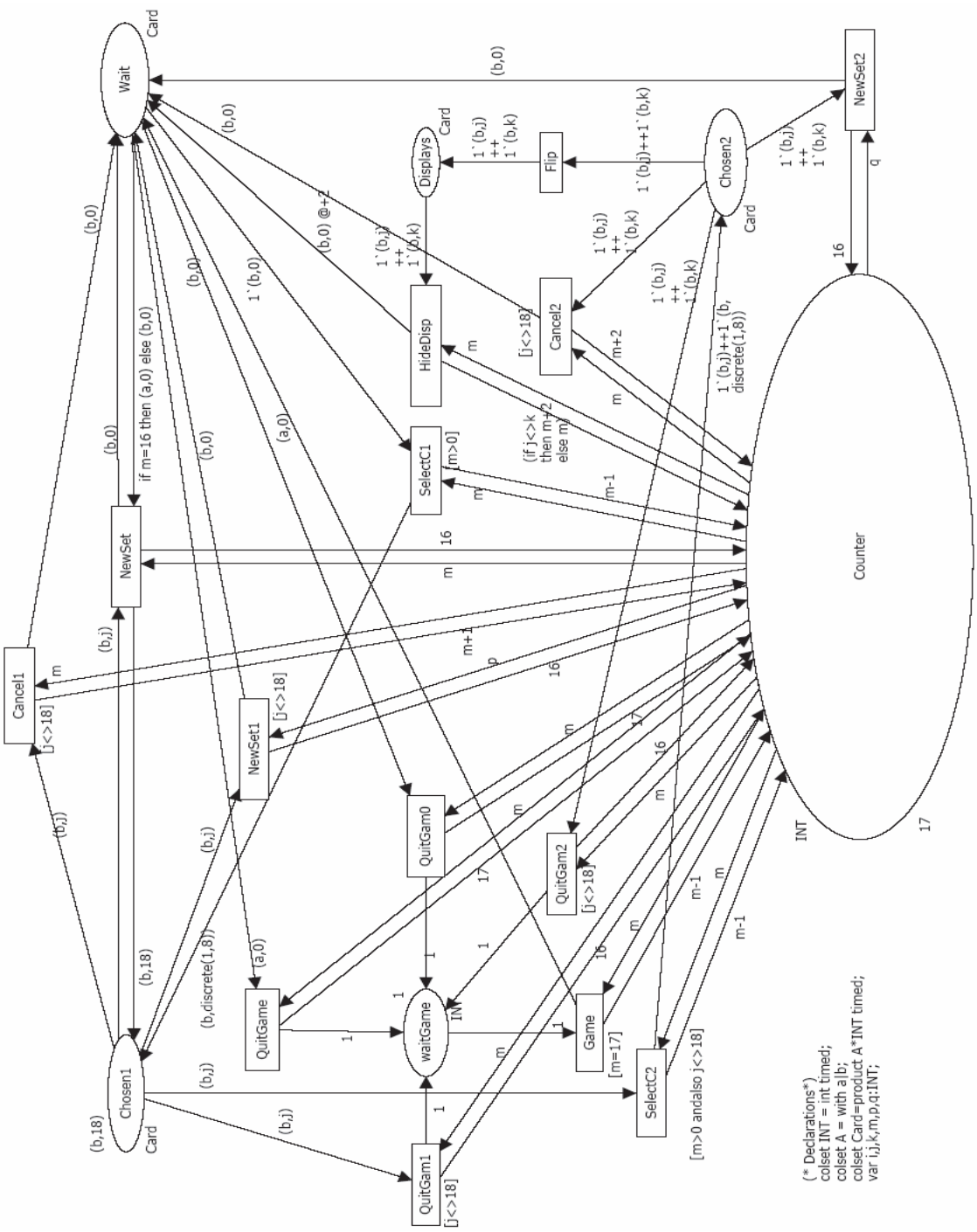

Figure 38 (b) Réseau de Petri temporisé de l'application jeu de mémoire sans distribution spatiale des places

4.3.3.2 Reconnaissance vocale

Le vocabulaire, la grammaire et les phrases que l'utilisateur peut exprimer pour dicter des commandes vocales à l'application sont décrits ci-après.
Le vocabulaire est constitué 12 mots. Les mots que l'utilisateur peut prononcer sont:

{« *Alpha*», «*Beta*», «*Gamma*», «*Delta*», «O*ne*», «*Two*», «*Three*», «*Four*», «*Flip*», «*Close application*», «*Cancel*», «*New Game*»}.

L'ensemble: {«*Alpha*» «*Beta*», «*Gamma*» «*Delta*»} représente les colonnes de la matrice.
L'ensemble : {«O*ne*», «*Two*», «*Three*» et «*Four*»} représente les lignes de la matrice des cartes à jouer. Les mots «*Flip*», «*Close application*», «*Cancel*», et «*New Game*» sont les actions que l'utilisateur peut dicter vocalement à l'application.
Nous donnons la grammaire à contexte libre pour l'application à la Figure 39.

1	COLONNE	→ alpha | beta | gamma | delta
2	LIGNE	→ 1 | 2 | 3 | 4
3	POSITION	→ COLONNE LIGNE
4	INIT_ACTION	→ newGame | quit
5	CARD_ACTION	→ cancel | flip
6	ACTIONS	→ INIT_ACTION | CARD_ACTION | ξ
7	SELECT	→ POSITION (ACTIONS)?
8	PARTIE	→ INIT_ACTION (SELECT (SELECT)?)$^+$

ξ représente l'ensemble vide $(S)^+$ = une ou plusieurs occurrences
(S)* = zéro ou plusieurs occurrences (S)? = une ou zéro occurrence

Figure 39 Grammaire indépendante du contexte pour l'application jeu de mémoire (Les symboles non terminaux sont écrits en majuscules et les symboles terminaux en minuscules)

Cette grammaire permet de reconnaître la structure des séquences possibles de phrases que l'utilisateur peut dicter à l'interface (voir l'Annexe 1 pour la définition des grammaires). Cette grammaire présente sept règles. Une carte à jouer est identifiée par une position. Une position est composée d'une colonne et d'une ligne. Une colonne et une ligne sont identifiées par les ensembles que nous avons décrits dans notre vocabulaire. Il existe deux types d'actions que l'utilisateur peut effectuer. Les actions d'initialisation sont : le démarrage d'une partie avec la commande «New Game» et l'action de quitter l'application avec la commande « Close application ».

Quand l'application est démarrée, l'utilisateur peut dire deux commandes, soit « New Game » ou « Close Application » car lors du démarrage, la table de jeu n'est pas encore initialisée.

Les actions qui peuvent être exécutées sur les cartes sont : l'action « Cancel » qui désélectionne toute les cartes sélectionnées et l'action « Flip » qui retourne les deux cartes qui sont sélectionnées.

Finalement, une partie doit absolument commencer par une action d'initialisation. Ensuite l'utilisateur doit sélectionner des cartes, les retourner pour visualiser leurs valeurs et répéter cette étape jusqu'à ce qu'il n'ait plus de cartes à retourner et que la partie se termine. Par contre, il est possible en tout temps de quitter le jeu ou de redémarrer la partie.

Comme nous l'avons mentionné une carte est identifiée par une position. Alors pour sélectionner une carte à jouer l'utilisateur doit faire la combinaison d'un mot de l'ensemble colonne et d'un mot de l'ensemble ligne.

Le Tableau VIII présente les combinaisons possibles pour sélectionner toutes les cartes sur le plateau de jeu.

Tableau VIII

Ensemble de phrases permises par la grammaire pour la modalité parole (+ : représente l'opérateur de sérialisation temporel)

Lignes	**Colonne**			
	Alpha	**Beta**	**Gamma**	**Delta**
One	{«Alpha» + «1»}	{«Beta» + «1»}	{«Gamma»+«1»}	{«Delta»+«1»}
Two	{«Alpha» + «2»}	{«Beta» + «2»}	{«Gamma»+«2»}	{«Delta»+«2»}
Three	{«Alpha» + «3»}	{«Beta» + «3»}	{«Gamma»+«3»}	{«Delta»+«3»}
Four	{«Alpha» + «4»}	{«Beta» + «4»}	{«Gamma»+«4»}	{«Delta»+«4»}

La Figure 40, quant à elle, présente la grammaire pour la modalité de reconnaissance vocale de l'application sous forme de réseau de Petri. Pour simplifier le diagramme, nous utilisons les termes 'Position 1' et 'Position 2' pour faire référence à une commande vocale du type colonne/ligne du Tableau VIII. Une fois la table de jeu initialisée, le joueur peut choisir entre trois options vocales : a) dicter la position de la première carte, b) quitter l'application c) ou redémarrer une nouvelle partie.

Si l'utilisateur dicte la position de la première carte le système peut accepter quatre actions :

a. quitter l'application;
b. démarrer une nouvelle partie;
c. dicter la position de la deuxième carte;
d. exécuter la commande « Cancel ».

S'il choisit de dire « Cancel » la première carte sera désélectionnée. Par contre, s'il dicte la deuxième position il pourra demander à l'application de retourner les cartes avec la

commande vocale « Flip ». Àvant de dicter « Flip », l'utilisateur peut également dicter la commande vocale « Cancel » et les deux cartes seront désélectionnées.

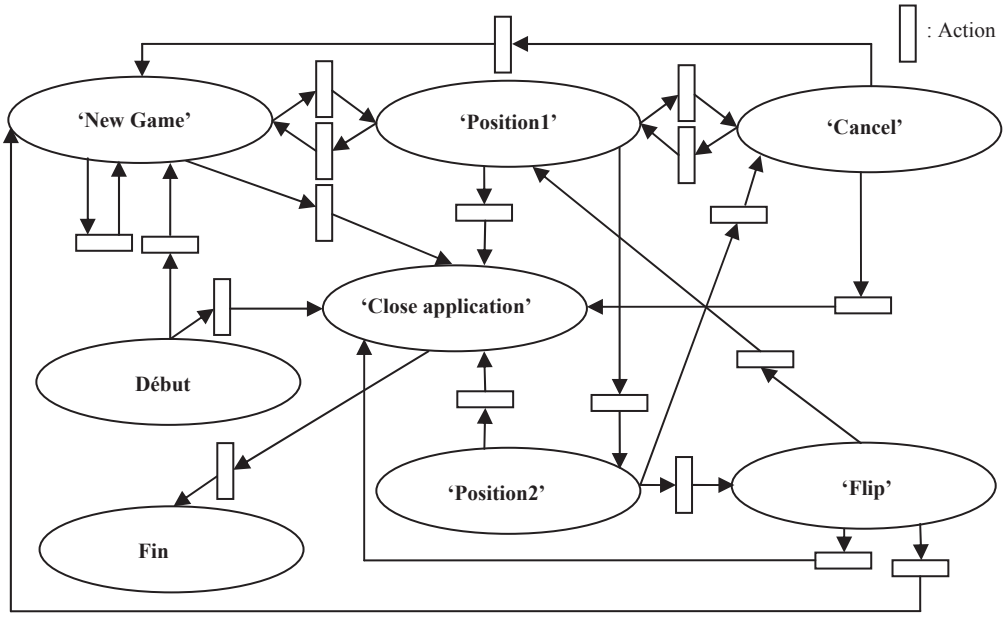

Figure 40 Grammaire pour la modalité de reconnaissance vocale du jeu de mémoire sous forme de réseau de Petri

Comme l'utilisateur fait afficher des cartes plusieurs fois, ces étapes sont nécessairement répétées.

4.3.3.3 Modélisation du dialogue multimodal du jeu de mémoire

La modélisation du dialogue multimodal par un RPCT du jeu de mémoire est représentée sur plusieurs pages (Figures 41 à 45). Cette modélisation permet de simuler le déroulement d'une partie. Les actions de *démarrer une nouvelle partie* et *de fermer l'application* ne sont pas modélisées. En effet, ces deux actions ont pour effet de 'vider' le réseau de Petri ce qui n'a pas un grand intérêt du point de vue de la modélisation.

4.3.3.3.1 Commandes vocales

La Figure 41 montre la génération aléatoire des commandes vocales. Les commandes possibles sont le nom d'une colonne (symbolisé par le Label de 1 à 4), le nom d'une rangée (Label de 5 à 8), retourner une carte (Label 9), annuler une commande (Label 10) et une mauvaise commande (Label 11).

La génération des mots est réalisée selon une loi de probabilité discrète équiprobable et le temps d'identification suit une loi normale. Les paramètres des lois sont paramétrables selon le type de système de reconnaissance vocale et également selon le débit moyen de l'utilisateur. Différentes lois de probabilités peuvent également être choisies à la guise de l'utilisateur.

4.3.3.3.2 Commandes de la souris

La Figure 42 démontre les commandes de souris. Il s'agit de la même logique que pour les commandes vocales. En effet, dans l'application, chaque commande vocale a son 'équivalent souris'.

La différence se situe dans le fait qu'un événement de souris spécifie directement une carte du jeu et non sa position en colonne et rangée. Les commandes sont l'une des 16 cartes du jeu (Label 3 à 18), retourner une carte (*Label* 1), annuler une commande (*Label* 2) et une mauvaise commande (*Label* 19).

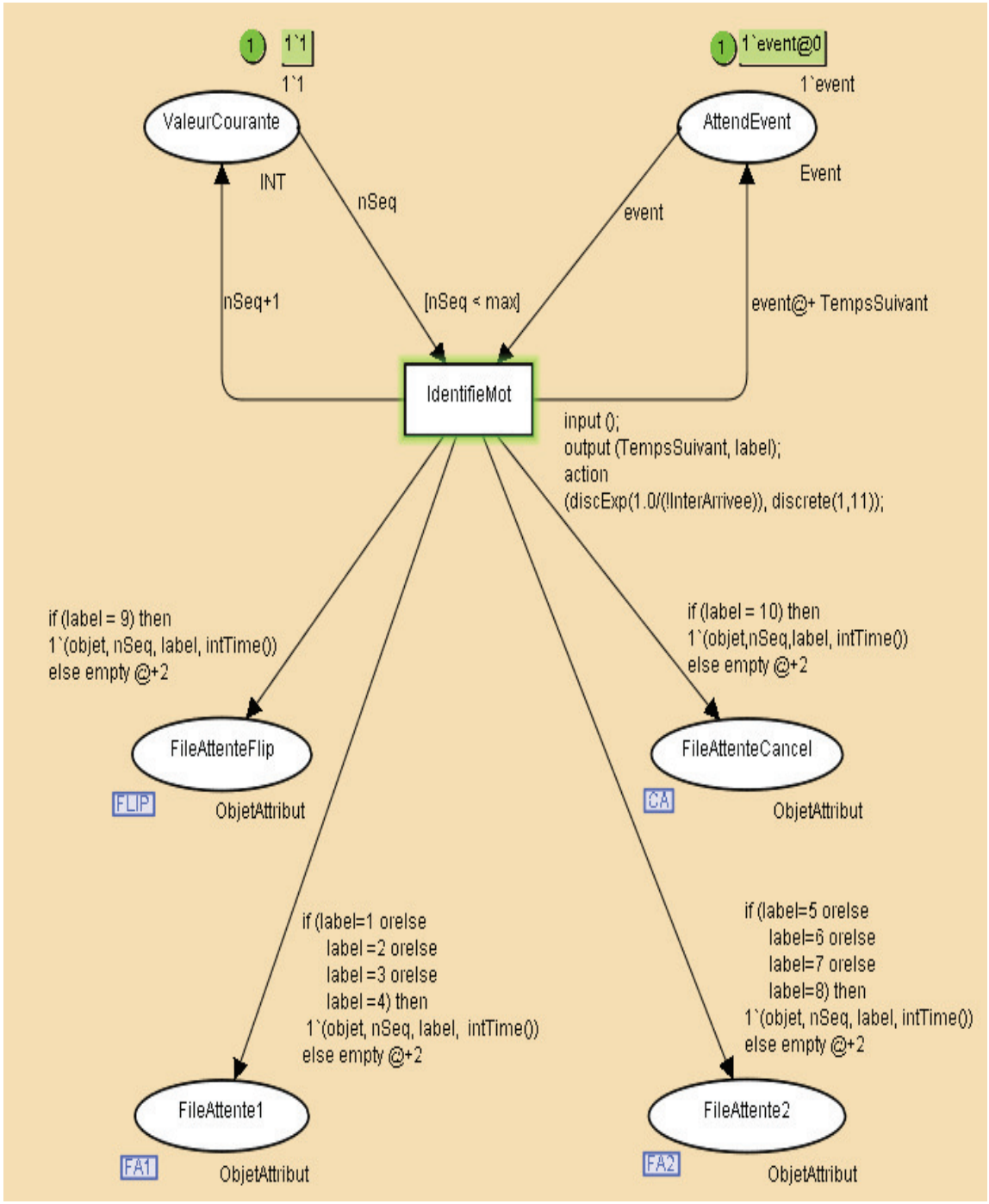

Figure 41 Commandes vocales

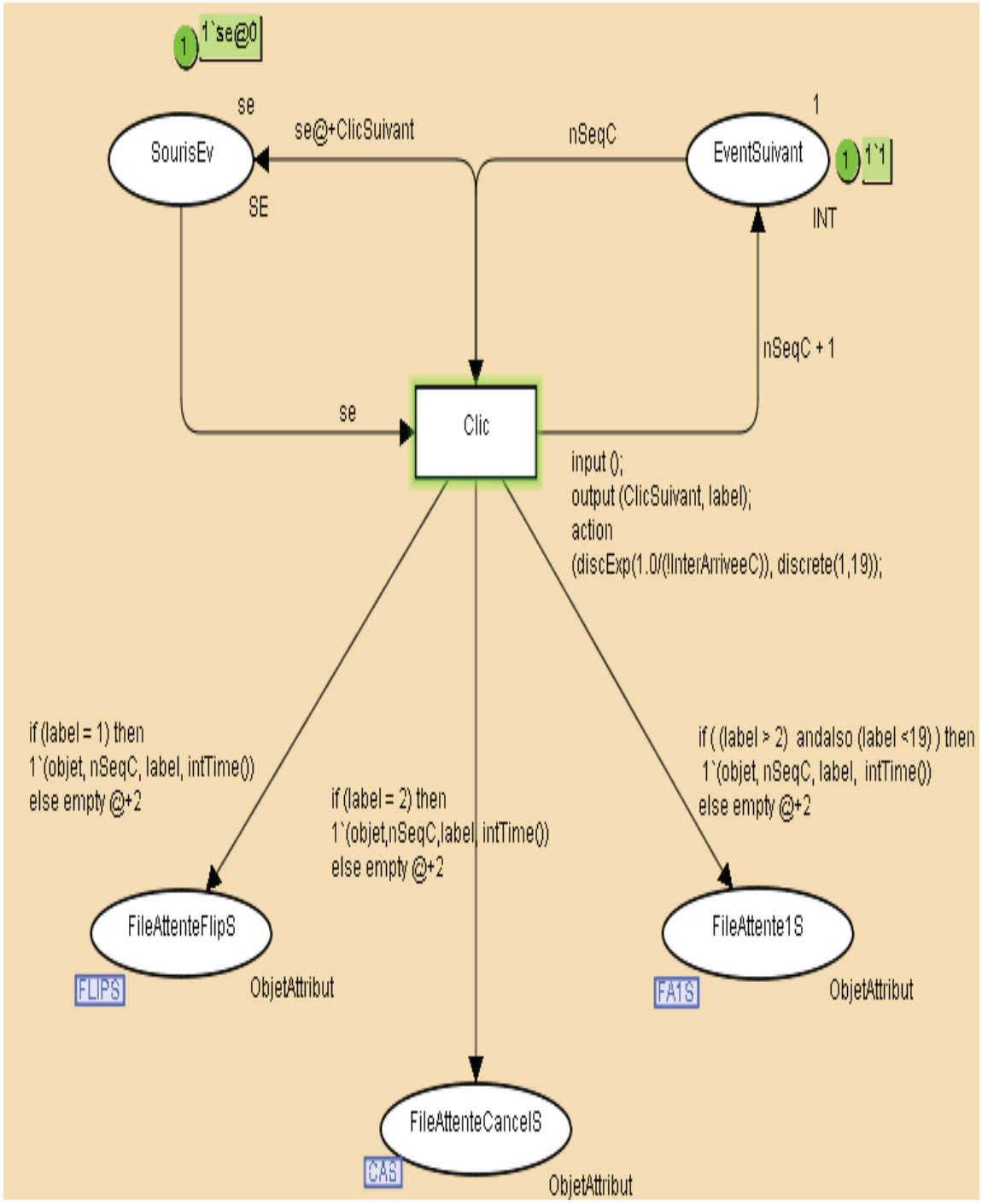

Figure 42 Commandes de la souris

4.3.3.3.3 Combinaison et fusion des modalités

La fusion des modalités en entrée se fait au niveau des choix de cartes et de la commande '*Flip'*. Nous pouvons observer avec la Figure 43 que la fusion des cartes est assez complexe. En effet, avant de fusionner les cartes sélectionnées avec la voix et la souris, il faut combiner sémantiquement les commandes vocales pour 'recréer' une carte. Il s'agit donc de mettre une colonne et une rangée ensemble : c'est le rôle de la transition **Combinaison 2A** sur le RPCT à la Figure 43. Finalement, nous réalisons une défusion dans deux listes distinctes des cartes reçues pour pouvoir distinguer la première et la deuxième carte sélectionnées.

Pour faire la distinction des cartes dans les deux files d'attente, une place nommée **CompteurA** et l'autre **CompteurB,** ont été utilisée. En effet, l'un de ces compteurs est incrémenté à chaque nouvelle carte indiquée par l'utilisateur. Nous utilisons la fonction *modulo* pour décider de la destination de la carte sélectionnée.

La fusion de la commande '*Flip'* est beaucoup plus simple. Il s'agit simplement de fusionner la commande vocale et la commande de souris dans une même file d'attente (**FileAttenteFlipFusion**). Il aurait même été possible de sauter cette étape de fusion. En effet, les commandes *Flip* venant de la souris et de la voix ont le même type de jetons. Il aurait donc été possible d'envoyer directement les commandes *Flip* dans la même place. C'est cette stratégie qui a été retenue pour les commandes de type *Cancel*.

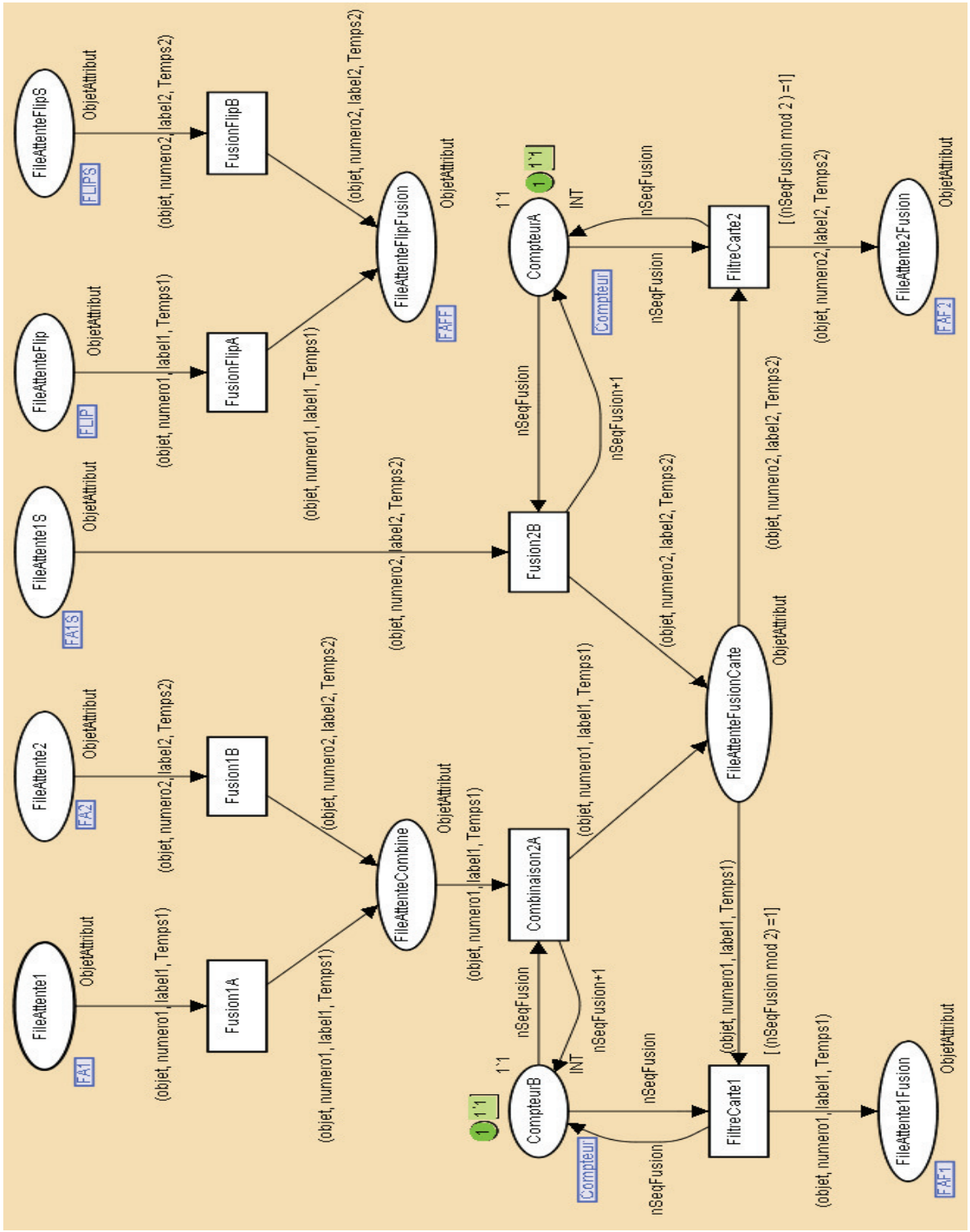

Figure 43 Combinaison et Fusion des modalités

La reconnaissance des commandes est constituée des deux cartes sélectionnées et de la commande '*Flip'*. Le critère de fusion est basé sur le temps d'arrivée des jetons dans les places.

En effet, il faut s'assurer que la carte 1 a été créée avant la carte 2 et aussi que la commande '*Flip'* a été réalisée après la carte 2.

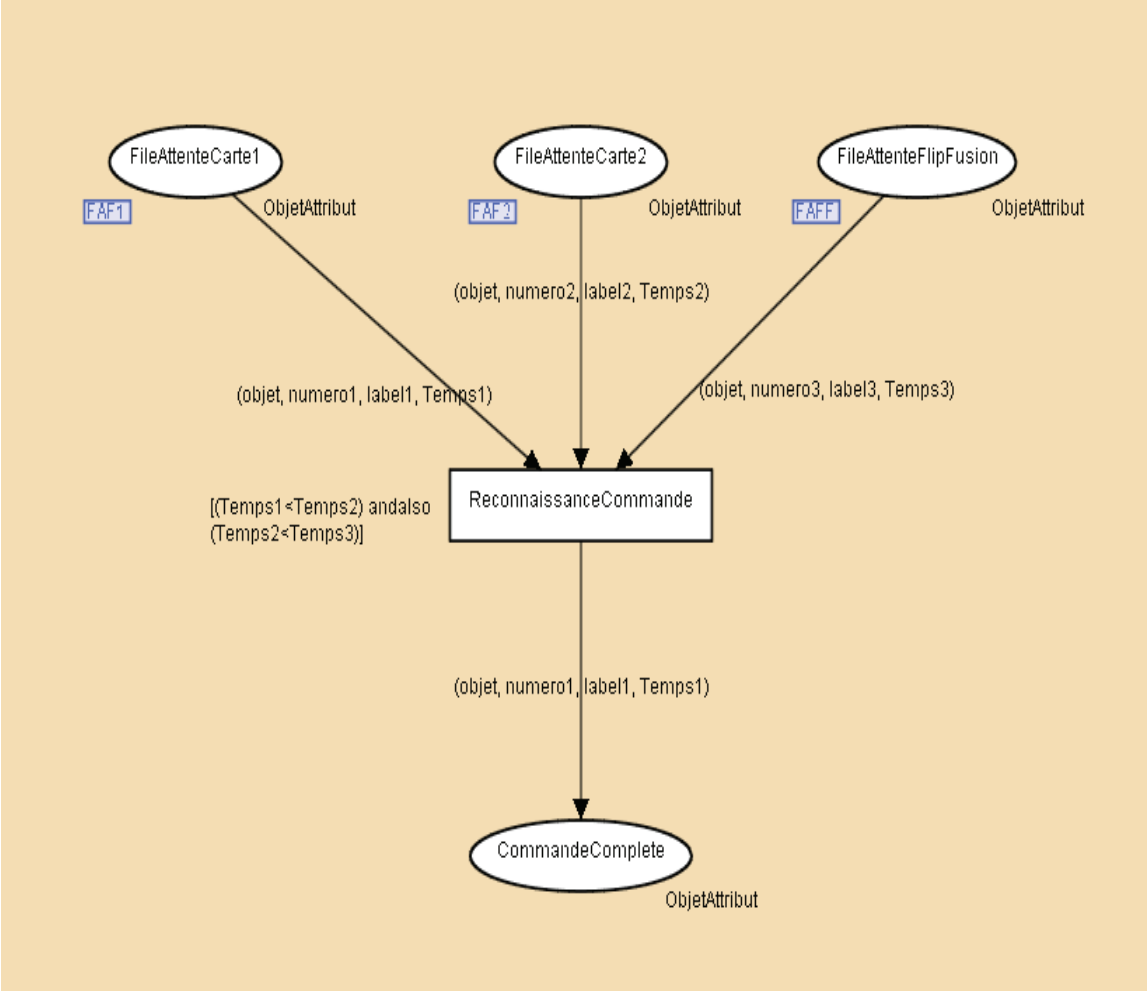

Figure 44 Reconnaissance des commandes

Le tout est démontré avec la modélisation représenté à la Figure 44.

4.3.3.3.3 Annulation de commande

Il y a deux cas possibles d'annulation de commande (Figure 45.)

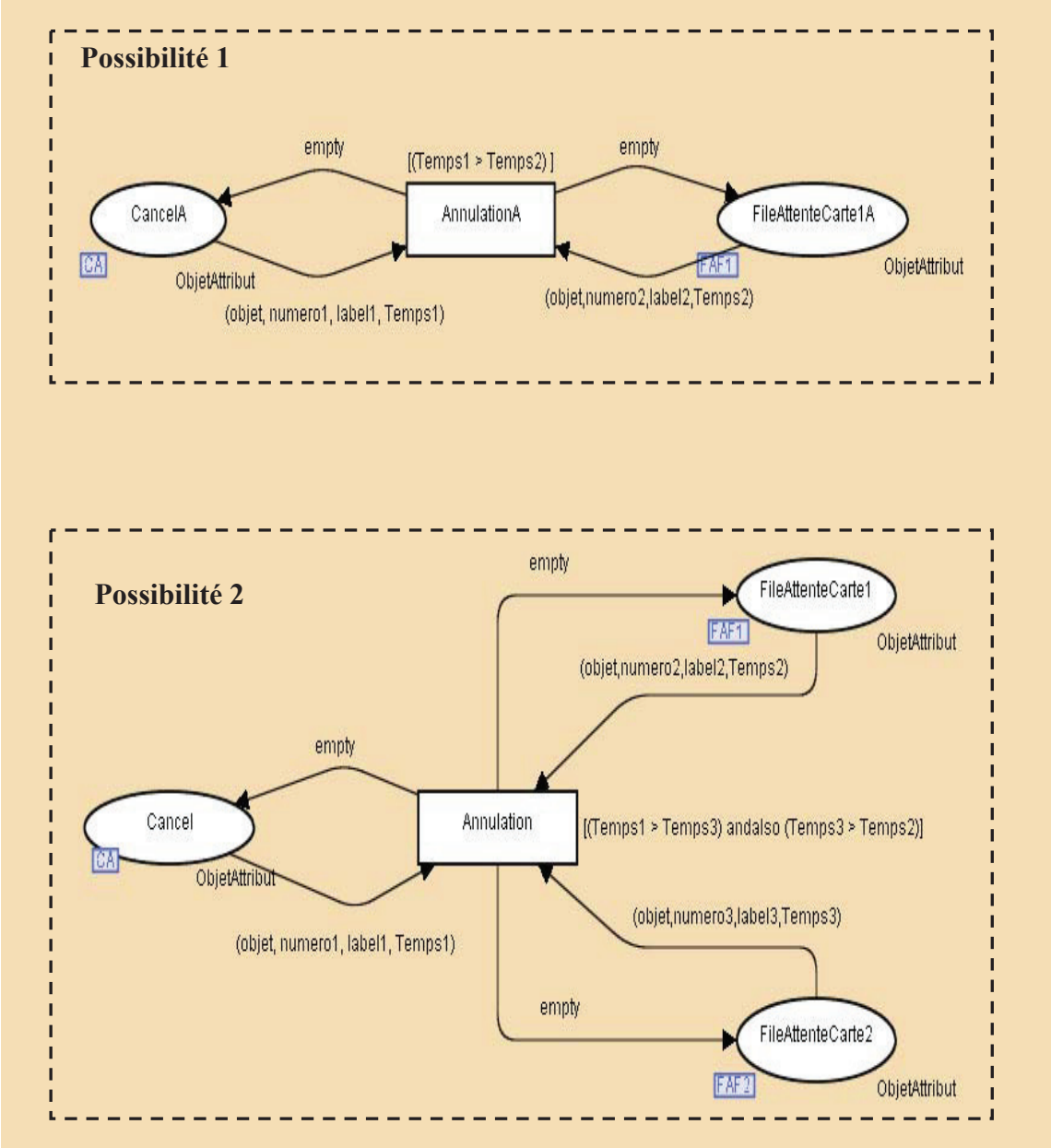

Figure 45 Annulation d'une commande

a. La première annulation consiste à recevoir la commande *Cancel* après qu'une seule carte ait été sélectionnée. Dans ce cas, il faut enlever une carte uniquement dans la file d'attente de la première carte (**FileAttenteCarte1A**).
b. La deuxième possibilité arrive lorsque la commande *Cancel* est donnée après que deux cartes aient été sélectionnées. Dans ce cas, il faut enlever une carte dans les deux listes d'attente (**FileAttenteCarte1** et **FileAttenteCarte2**).

Les déclarations nécessaires à la simulation sous CPN-Tools des réseaux des Figure 41 à 45 précédentes sont données à l'Annexe 4.

4.3.4 Conclusion

L'application développée dans le cadre de ce projet a permis de mettre en évidence plusieurs concepts. En effet, l'application permet l'utilisation de deux modalités en entrée et en sortie et intègre une grammaire multimodale temporelle et l'emploi de lois de probabilités pour la génération des symboles.
L'application multimodale implique l'emploi d'une grammaire temporelle qui embarque des processus de défusion, adaptée à la situation. Il est beaucoup plus aisé de modéliser la fusion de plusieurs modalités avec un réseau de Petri que par l'utilisation de diagrammes d'activités (diagrammes d'états ou de séquences).

La plus grande limitation de l'application développée se situe au niveau de la reconnaissance vocale. Ceci peut s'expliquer par la gratuité de l'API employé. Il arrive fréquemment qu'il faille répéter une même commande vocale pour que l'application la comprenne avec exactitude. Si il y avait eu plus de temps ou de moyens financiers disponibles pour la réalisation du projet, il aurait été intéressant de tester plusieurs bibliothèques de reconnaissance vocale non gratuites pour observer s'il y en avait une qui aurait été plus performante pour nos besoins.

4.4 Conclusion du chapitre 4

Ces exemples d'applications multimodales nous ont permis d'explorer un domaine qui est présentement très prospère : l'industrie du jeu. Il est vrai que ces applications simplistes ne sont pas assez élaborées pour pouvoir être mise sur le marché. Mais, ces exemples démontrent qu'il y a beaucoup de potentiels à exploiter dans le domaine des nouvelles technologies reliées à la multimodalité et aux jeux. En effet, étant donné la demande grandissante dans le domaine du divertissement, il est possible d'automatiser complètement une interface d'un jeu à l'aide de commandes vocales combinées à d'autres commandes. Le lien de la référence (http://edwardtse.com/edwardtse-videos.html 2006) est un exemple de jeu existant, 'Warcraft III', modifié par des étudiants d'une université pour être fonctionnel sur une table digitale. Cette table digitale permet de cliquer directement avec les doigts sur l'image et d'entrer des commandes vocalement pour activer les unités du jeu.

Ces deux exemples nous ont montré l'intérêt de notre approche architecturale employant une modélisation par RPCT : l'intoduction de contraintes grammaticales temporelles et la possibilité de vérifier les propriétés comportementales de l'interaction multimodale, de façon automatique, grâce à l'outil CPN-Tools. En effet, ce dernier est capable de vérifier les propriétés comportementales du réseau de Petri (et fournit un rapport sur ces propriétés après une analyse automatique du réseau). À titre d'exemple, nous donnons à l'Annexe 5 le listage des résultats de la vérification de ces propriétés pour les réseaux des application 'TUX_X-MEN' et jeu de mémoire (réalisés avec l'outil CPN-Tools). Ces propriétés vérifées par l'outil sont suffisantes pour affirmer que les modèles RP des AL multimodales proposées ne comportent pas de blocage (présences de transitions qui deviennent non franchissable ou boucles infinies dans le réseau qui ne sont pas prévues par les concepteurs de l'interaction.) Il est aussi possible de vérifier ces propriétés comportementales formellement pour chacun des réseaux présenté dans ce document. Pour cela il faut écrire les matrices d'incidence $\boldsymbol{C}$ de chacun des réseaux puis faire une

analyse formelle. Nous ne le faisons pas car l'outil CPN-Tools le fait de façon automatique pour nous. Une perspective de ce travail de recherche serait la vérification formelle des propriétés comportementales des réseaux présentés ici. Cependant, nous considérons que les résultats de l'analyse automatique par l'outil CPN-Tools sont suffisants comme «preuve» pour la vérification du comportement de l'AL.

La concurrence entre les entreprises qui produisent des jeux est impitoyable. Aussi, nous pensons que la multimodalité est probablement l'avenue du futur pour cette industrie.

Nous allons maintenant nous intéresser, au chapitre suivant, à la reconfiguration dynamique d'AL multimodales.

CHAPITRE 5

AGENT EXPERT POUR LA RECONFIGURATION ARCHITECTURALE DYNAMIQUE VIA SCÉNARIOS

5.1 Introduction

La spécification des architectures logicielles et de leurs évolutions dynamiques est un thèmes qui requiert un intérêt croissant en recherche (Ramdane-Cherif 2001; Ramdane-Cherif 2002). En effet, de plus en plus de systèmes intègrent la composante dynamique pour répondre à de nouvelles exigences d'adaptabilité, liées à l'évolution du contexte, à des défaillances internes ou à la détérioration des qualités. Ceci est d'autant plus vrai dans le cas des interfaces multimodales qui doivent grandement tenir compte du contexte de l'application.

Le contexte recouvre un ensemble d'informations (et/ou conditions) dans lequel il est possible de situer (et/ou d'interpréter) une information donnée. Cet ensemble dépend entre autres des objectifs de l'application multimodale. L'environnement qui constitue une des composantes du contexte est l'ensemble des modalités, de leurs interactions, et des tâches multimodales que l'utilisateur est susceptible de réaliser. Il peut y avoir un ou plusieurs environnements sur une plateforme physique, par exemple : mise à l'essai et production. Un environnement a des particularités et des caractéristiques uniques qui dictent comment il peut être administré de façon distincte. Mais, pour notre cas d'étude, nous supposons que l'application n'est pas distribuée sur un réseau.

Ce chapitre est consacré à la présentation d'une approche, qui consiste à utiliser un agent expert (AE) pour modifier dynamiquement la configuration d'une AL multimodale. Cet AE modélisé avec les RPCTS est constitué d'une partie décisionnelle qui lui offre des capacités d'autogestion et d'organisation de l'architecture multimodale et d'une partie applicative affectée à la réalisation des reconfigurations sur cette même architecture

multimodale. Une approche de reconfiguration architecturale plus générale que celle présentée dans ce chapitre a fait l'objet d'une thèse au laboratoire PRiSM (Benarif 2007). À la différence de cette approche notre approche est dédiée aux problèmes inhérents à l'interaction multimodale (comme la prise en compte du temps, de la grammaire multimodale, des attributs architecturaux multimodaux, etc.) Notre approche met en avant la spécification de l'AE par des RPCTS pour réaliser la validation formelle des propriétés comportementales de l'AE et également dans un but de simulation, avant la phase de développement, pour l'analyse et l'interprétation de l'activité de l'AE en fonction de différents paramètres (comme le temps.)

Après une présentation générale de l'approche, nous découvrons la structure générale de l'AE via sa description préliminaire sous forme de RP. Cette description est simplificatrice mais offre les différentes fonctionnalités de base dont il est doté. L'aspect 'agent' et interaction avec l'architecture selon un modèle de qualités seront traités plus en détails dans le chapitre suivant.

Puis, après avoir présenté cette première ébauche qui consiste en une description de l'AE à un niveau d'abstraction élevé, nous donnons un peu plus d'informations sur le modèle de l'AE. Nous nous intéressons en particulier au concept « scénario » utilisé dans notre approche pour exprimer les besoins en qualité de l'architecture. La structure et les propriétés du scénario y sont traitées. Nous proposons une définition du scénario qui est modélisable par RPCTS.

Nous décrivons alors brièvement la méthodologie développée pour la conception et l'implémentation des scénarios au cœur de l'AE. Enfin, nous terminons ce chapitre par une conclusion qui synthétise nos contributions.

5.2 Présentation générale

La reconfiguration dynamique de l'architecture multimodale permet de faire face à des variations de l'environnement en cours d'exécution, dans le but de satisfaire les besoins nouveaux et imprévus des utilisateurs ou éventuellement du système. Notre approche

repose donc sur l'amélioration de la qualité logicielle par une reconfiguration architecturale.

La modélisation du domaine d'application de l'AE permet d'identifier de façon générale la problématique et les solutions proposées. La Figure 46 présente une modélisation du domaine avec les principales fonctionnalités de l'AE.

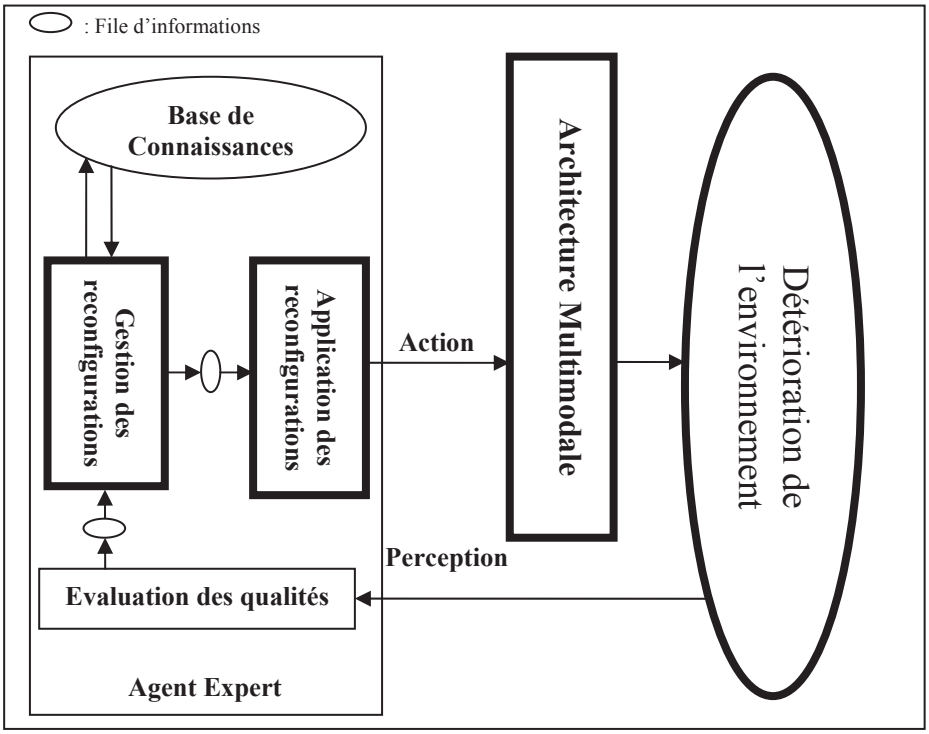

Figure 46 Approche suivie

La détérioration des qualités du système constitue la problématique que notre approche doit résoudre à travers les fonctions de base suivantes :

a. spécification des besoins : elle consiste à recueillir les besoins des intervenants, puis à orienter les différents besoins en scénarios de reconfiguration;

b. décryptage des besoins : le décryptage des besoins consiste à interpréter les scénarios de reconfiguration; ainsi, toutes les informations relatives aux contextes et aux solutions architecturales proposées lors de la spécification des besoins, vont être analysées puis sauvegardées dans la base de connaissance de l'AE;
c. gestion des reconfigurations : cette fonctionnalité est incluse dans l'AE; elle permet la sélection des scénarios de reconfiguration les plus pertinents;
d. application des reconfigurations : cette fonctionnalité, également incluse dans l'AE, consiste à appliquer le scénario choisi par l'AE; l'exécution du scénario est réalisée grâce à la décomposition du scénario en actions simples comme l'inhibition ou la stimulation d'un composant.
e. évaluation des qualités : afin de parvenir à maintenir le niveau de qualité souhaité, l'AE dispose d'une fonctionnalité de surveillance du système; la détérioration de la qualité déclenche immédiatement une recherche du scénario adéquat pouvant apporter une solution architecturale à ce problème.

5.3 Première approche pour l'AE

Nous proposons dans ce paragraphe une approche qui ébauche des fonctionnalités de base de l'AE en termes de reconfiguration architecturale. En fait, nous amorçons ici une esquisse de cet AE. Par la suite nous le raffinerons et lui attribuerons une structure plus complexe qui lui permettra de réaliser de la reconfiguration dynamique architecturale via 'scénario'. Ce dernier aspect fait l'objet de la suite du paragraphe 5.4 dans ce document.

5.3.1 Travaux relatifs à la reconfiguration architecturale dynamique

Le modèle architectural d'un système multimodal doit fournir une description abstraite qui permet à la fois d'analyser le système et ses composants et d'évaluer ses attributs de qualités. Tout système a une architecture et celle-ci est la base des processus systématiques de développement et d'évolution du logiciel.

Les premiers travaux sur la description et l'analyse des structures architecturales ont porté sur la spécification d'architectures statiques. Récemment, le besoin de spécifier les aspects dynamiques des architectures se fait de plus en plus ressentir (Kramer 1996-a; Oreizy 1998). Ce besoin a donné lieu à de nombreuses publications. Plusieurs auteurs ont développé des approches qui introduisent le dynamisme dans les architectures en séparant le comportement dynamique de la reconfiguration de l'aspect architectural statique.

Ces approches augmentent la propriété de réutilisation (*'réutilisabilité')* de certains composants du système. Dans (Allen 1997), les auteurs ont utilisé le langage de spécification Wright pour prendre en compte les aspects dynamiques. Certains auteurs (Taylor 1996) proposent l'ajout d'un langage complémentaire pour exprimer les modifications et les contraintes d'un style architectural basé sur les messages. Une approche semblable est utilisée dans le système Darwin (Kramer 1996-b) où un gestionnaire contrôle la reconfiguration désirée en utilisant un langage de script. D'autres recherches ont abordé la question de la reconfiguration dynamique pour des types d'applications particulières. Par exemple, Polylith (Purtilo 1994) est un environnement de programmation distribuée basé sur un bus de logiciel qui permet les changements structurels des systèmes hétérogènes répartis. Dans Polylith, la reconfiguration peut se produire uniquement à certains moments particuliers appelés des points de reconfiguration. Ceux-ci sont explicitement identifiés dans le code source de l'application. Ce mécanisme très contraint rend Polylith peu utilisable pour la reconfiguration dynamique. L'environnement de programmation Durra (Barbacci 1993) propose un mécanisme de déclenchement des reconfigurations par des événements. L'inconvénient de cet environnement est que le traitement des reconfigurations doit être défini dans le code source de l'application et que le programmeur doit considérer tous les événements possibles pouvant déclencher une reconfiguration. Argus (Bloom 1993) propose une autre approche basée sur un système d'opérations transactionnelles. L'application doit se conformer à un modèle de programmation spécifique. Cette approche n'est donc pas appropriée pour traiter l'hétérogénéité et l'interopérabilité.

L'approche Conic (Kramer 1990) propose un mécanisme indépendant des applications où les changements de configuration affectent les interactions entre composants. Chaque action de reconfiguration peut être mise en œuvre si et seulement si les composants sont dans un état bien déterminé. La mise en place de cette approche tend à verrouiller une grande partie de l'application causant ainsi des interruptions importantes. De nouveaux langages formels pour spécifier la mobilité des composants ont été proposés (Ciancarini 1998; DeNicola 1998). En particulier dans (Van Belle 2001) un nouveau cadre expérimental est utilisé pour étudier deux problèmes des systèmes à base de composants. Le premier concerne le développement des architectures robustes à base de composants mobiles. Le deuxième concerne l'écriture du code pour ces systèmes. Cette analyse indique clairement qu'une nouvelle architecture qui permet la reconfiguration, l'adaptation et l'évolution dynamique de l'architecture, tout en assurant l'intégrité de l'application, est nécessaire. Tous ces travaux montrent l'importance grandissante que revêt la reconfiguration dynamique pour les AL que les concepteurs souhaitent rendre mobiles et évolutives. Cependant, notons que, dans l'état de nos connaissances, la littérature ne fait pas mention de travaux sur la reconfiguration dynamique pour des architectures multimodales. Aussi, nous pensons que les questions posées et les problèmes mis en évidence par ces différents travaux restent pertinents si nous souhaitons faire de la reconfiguration dynamique architecturale dédiée aux applications multimodales.

5.3.2 Première approche de reconfiguration

5.3.2.1 Services de Reconfiguration

Nos premiers travaux (Djenidi 2002-c; Djenidi 2002-d; Djenidi 2002-e; Djenidi 2004-a) se sont orientées vers le paradigme agent dont le principe consiste à introduire un AE intelligent afin d'appliquer des reconfigurations sur l'architecture (voir Figure 47); ces processus obéissent à ce que nous avons appelé des règles de reconfiguration. L'idée

proposée est d'inclure un nouveau type d'agent intelligent (Ramdane-Cherif 2002-a; Ramdane-Cherif 2002-b) dans l'architecture multimodale. Cet agent agit avec autonomie et dynamisme pour adapter l'application sans aucune intervention externe de l'utilisateur. Cet agent fait un monitoring de l'architecture et réalise la reconfiguration.

L'architecture multimodale est décomposée en éléments structurels maintenus dans un environnement composé entre autres de *fragments d'architecture* (Figure 47.) Si l'AL contrôlée par l'AE est elle-même structurée en parties (composées d'agents travaillant sur une ou plusieurs localités où se trouvent des médias), l'AE responsable de la reconfiguration sera aussi structuré en parties et peut être considéré comme un SMA. Chacun des agents le composant s'occupe de gérer une ou plusieurs localités (pouvant être modélisées par des places) et communique à travers un réseau de communication avec les autres agents pour réaliser ainsi le monitorage global de l'architecture. Sur la Figure 47, les différents composants i $\mathbf{Co_i}$, d'un fragment d'architecture correspondent aux composants d'un AdL (ou d'un ACP.)

Dans la représentation symbolique de la Figure 47 (a), les environnements sont identiques. Cependant, nous les avons distinguées en zones environnementales différentes; chacune des zones pouvant être surveillée et reconfigurée par une instance du même AE. L'AE représenté à la Figure 47 (b) est employé pour prendre en charge la reconfiguration au niveau architectural.

Les adaptations dynamiques sont des changements qui se produisent en temps réel et elles dépendent du contexte d'exécution de l'application.

Les services de reconfiguration offerts par l'AE consistent en des opérations répétitives dans tous les cas d'évolution architecturale. Ceci explique aussi l'intérêt d'une modélisation par une machine à états comme les RP. À un niveau d'abstraction élevé ces opérations réalisées sont caractérisées de la façon suivante. L'agent est responsable:

a. de la 'création' et la 'suppression' d'instances de composants;
b. de la stimulation ou de l'inhibition des liens entre les composants;

c. du transfert des états d'une configuration donnée aux différents composants instanciés et préexistants.

Des exigences sont ajoutées à l'emploi de ces opérations pour préserver les contraintes architecturales et procurer une garantie de sécurité supplémentaire. Les principaux problèmes qui surviennent (lorsque les attributs architecturaux de qualité, comme la 'facilité de modification' ou la 'facilité de maintenance'[29], doivent être vérifiés) sont de deux ordres.

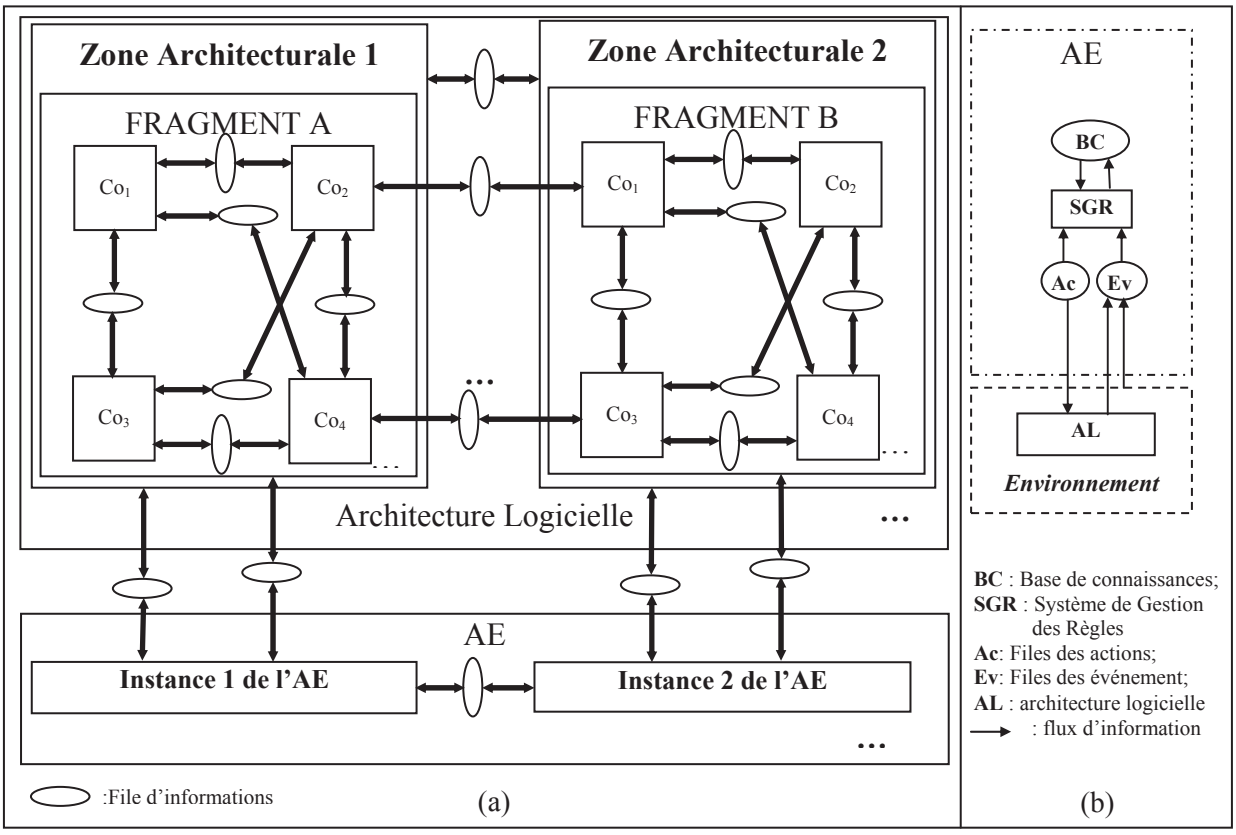

Figure 47 (a) Architecture basée sur l'Agent Expert (Coi : composant i) (b) Vue partielle symbolique de l'Agent Expert

[29] Traduction du terme anglo-saxon : '*maintainability*.'

a. Il faut pouvoir évaluer les changements qui déterminent les propriétés de l'architecture qui sont affectées et les incohérences qui peuvent en résulter.
b. Il faut également pouvoir gérer les changements pour assurer la protection des propriétés globales de l'architecture lorsque de nouveaux composants et connexions sont activés ou inhibés dynamiquement.

Afin d'apporter une solution à ces problèmes nous proposons deux types d'agents : l'Agent Interface et l'Agent d'Expertise.

5.3.2.2 Principe des agents employés dans notre approche

La Figure 48 montre l'Agent Interface qui permet de réaliser le lien entre chaque instance de l'Agent d'Expertise (responsable de la reconfiguration architecturale d'une AL) et son environnement (contenant l'AL). Il est défini par un ensemble d'actions et d'évènements. La Figure 48 (a) représente ce principe sous la forme d'un réseau de Petri.

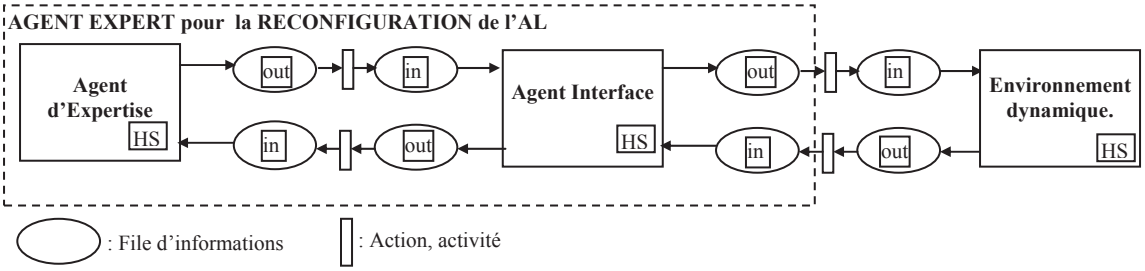

Figure 48 (a) Décomposition de l'agent pour la reconfiguration de l'AL en un Agent d'Expertise et en un Agent Interface

Nous associons à l'AE un mécanisme dicté par les règles du type : évènement/condition/action. Ces règles lui permettent de réagir aux activités de l'architecture et de l'environnement architectural et de réaliser ses activités. Réaliser une

activité signifie invoquer dynamiquement une ou plusieurs méthodes accompagnées des paramètres adéquats, nécessaires à la reconfiguration architecturale. De cette façon un agent peut:

a. rassembler des informations sur l'architecture et sur l'environnement;
b. déclencher des mécanismes de reconfiguration suite aux événements en provenance de l'architecture et de l'environnement;
c. prendre des décisions personnelles par l'emploi de mécanismes selon des règles;

5.3.2.3 Règles comportementales

Nous introduisons dans ce paragraphe une vue architecturale du comportement de l'Agent d'expertise (Figure 48 (b)). Cette vue met en évidence sa relation avec l'agent interface et l'environnement. La vue proposée est modélisée par un réseau de Petri à la Figure 48 (b) qui est le raffinement de la vue Figure 48 (a).
L'agent dispose d'un ensemble de règles écrites dans une notation élémentaire qui permet de raisonner. Nous distinguons trois catégories de règles:

a. celles qui décrivent la réaction de l'agent à des événements;
b. celles permettant l'interconnexion des dimensions structurelles;
c. et celles permettant l'interconnexion des dimensions fonctionnelles.

Ces règles sont spécifiées par le comportement et l'activité des RPCTS, par ses liens dynamiques et par les expressions en CPN-ML sur ces réseaux. Des exemples, sur la fçon dont sont appliquées ces règles, sont présentés dans les cas de reconfiguration dynamique au paragraphe 6.4.6. de ce document.
Chaque dimension décrit les variations d'une caractéristique architecturale. Les valeurs de ces dimensions correspondent à des alternatives d'exigences ou à des choix de design.

L'Agent d'Expertise dispose ses actions en fonction de trois types d'états et de deux types de comportements (Figure 48 (b)).

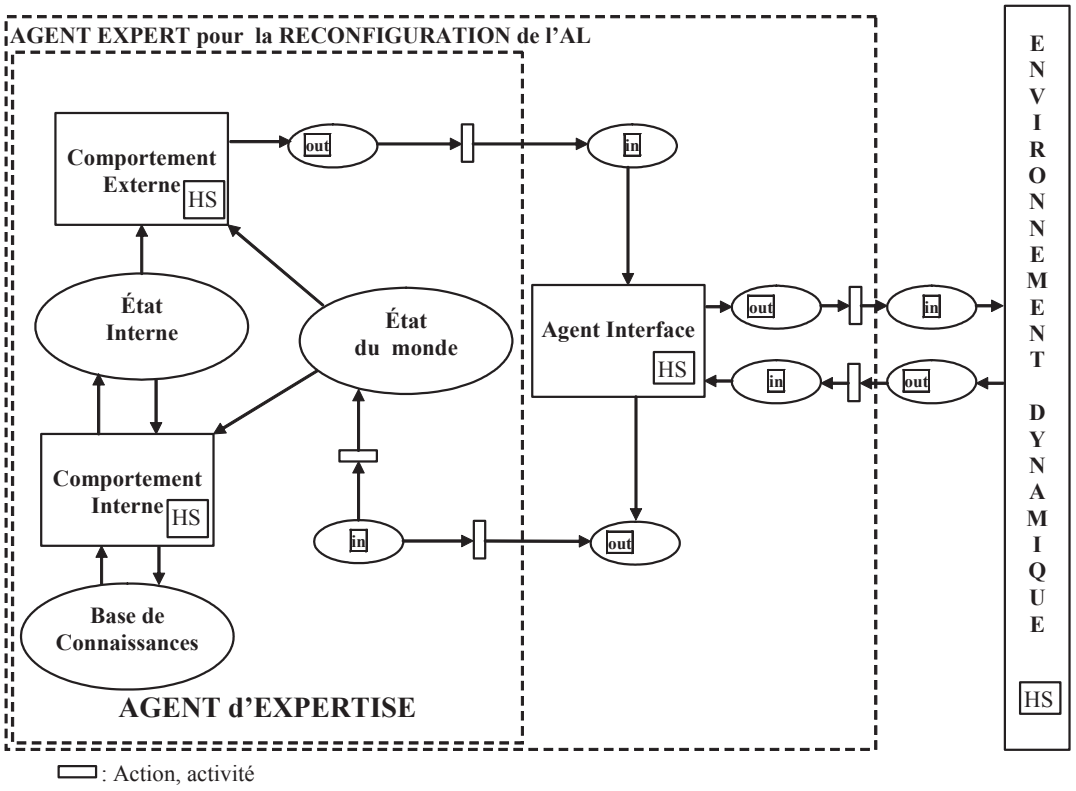

Figure 48 (b) Vue comportementale cyclique de l'architecture de l'Agent d'Expertise, modélisée par réseau de Petri

'L'**État du monde**' de l'agent est un état qui provient de la mise à jour des informations interprétées par les capteurs de l'agent. Il conduit à des comportements internes et externes de l'**Agent d'Expertise** qui reconfigure l'architecture. L'**état interne** stocke les variables internes à l'agent qui sont la conséquence des états courants ou précédents du monde de l'agent. L'agent met à jour son **état interne** lorsqu'il détecte des événements

via son interface. L'état de sa **base de connaissance** qui définit les règles flexibles de l'agent n'est accessible que via des **comportements internes** de l'agent (Figure 48 (b)).
Les **comportements internes** sont exécutés à partir des informations sur l'**état interne** courant, sur l'**état du monde** et sur la **base de connaissance** de l'agent. Les **comportements externes** de l'agent sont des activités qui emploient les **états internes** du monde et sélectionnent des actions que l'agent va réaliser. Ces actions affectent l'architecture (qui se trouve elle-même dans l'environnement) et donc, les futurs préceptes et états du monde de l'agent. Le **comportement externe** considère uniquement l'**état du monde** et l'**état interne** sans avoir un accès direct à la **base des connaissances** (voir Figure 48 (b)).
Dans le cas de la reconfiguration architecturale d'un système, l'architecture proposée inclut un mécanisme de base permettant l'orchestration de la coordination tout en garantissant la cohérence et l'exactitude du processus en temps réel. Elle assure aux agents la capacité de communiquer d'analyser et, de manière plus générale, de raisonner sur la modification qu'ils doivent opérer.

5.3.2.4 Connaissances de l'Agent

Un agent capture différentes connaissances pour évaluer et gérer les changements architecturaux. Ces connaissances font partie de la base de connaissance de l'agent. Par exemple, il est possible de gérer l'introduction d'un nouveau composant dans l'architecture de plusieurs manières. Elle peut se faire directement ou par l'emploi de comportements déjà existants ou suite à la création de nouveaux comportements. L'agent ne considère que la partie de l'architecture sujette à la reconfiguration et procède par étapes. Il capture les propriétés **Pi** directement reliées au nouveau composant **i**. Ces propriétés correspondent, par exemple, aux paramètres de temps, de charge du processeur (CPU), de l'information sémantique, etc. transportés par un événement généré par l'emploi d'une nouvelle modalité **i** inactive alors dans l'architecture. Puis, l'agent :

a. sonde toutes les propriétés $\mathbf{P_d}$ (des autre composants) reliées aux $\mathbf{P_i}$ et les structures architecturales qu'elles affectent;
b. il détermine toutes les incompatibilités à vérifier dans le cadre des propriétés $\mathbf{P_i}$ et/ou $\mathbf{P_d}$;
c. il détermine toutes les incompatibilités avec le composant récemment ajoutés;
d. il génère l'ensemble des composants/connecteurs et des propriétés pertinentes nécessitant des nouvelles évaluations.

5.3.3 Principe élémentaire de reconfiguration dynamique

Dans cet exemple un nouveau média est employé dans une application multimodale en cours d'utilisation. Le système est décrit par la Figure 49 (a). Le système initial est composé de deux AdL (voir chapitre 2).

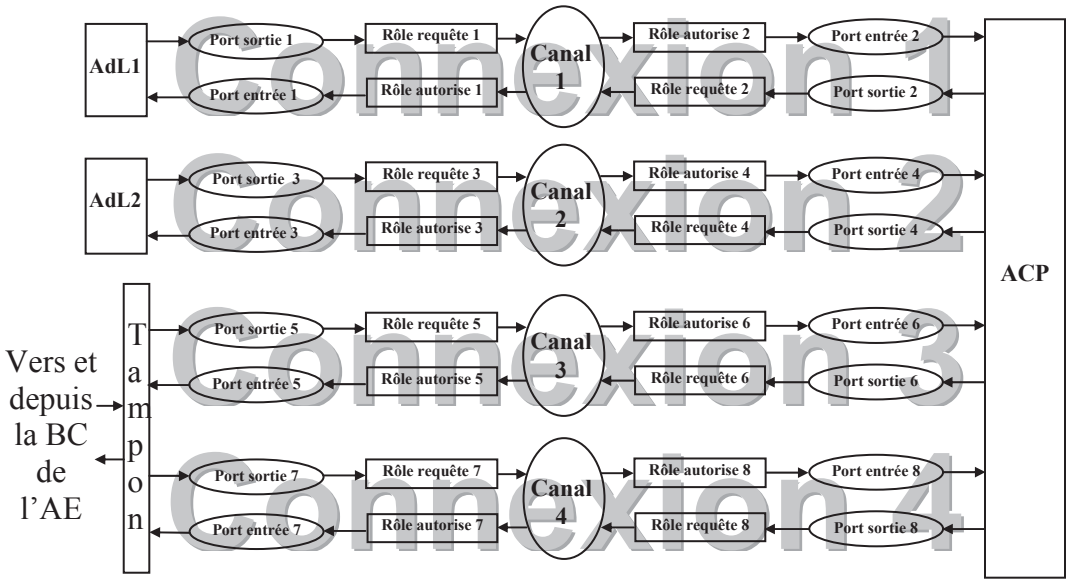

Figure 49 (a) Architecture initiale (BC: Base de Connaissances)

La fusion est réalisée par un ACP. L'attribut de qualité qui mesure l'efficacité de la reconfiguration en un temps donné est supposé le plus important. Afin d'améliorer cet attribut, la reconfiguration doit se faire graduellement.
L'adaptation doit être réalisée de façon sécuritaire pour assurer l'intégrité globale de l'architecture. À la réception d'un événement (du type nouvelle modalité employée par exemple) l'agent suit la stratégie qui consiste en l'application de quelques règles opérationnelles. L'AE de reconfiguration architecturale:

a. se crée une nouvelle base de connaissances pour la nouvelle modalité (New BC);
b. remplace progressivement l'ACP par un nouveau ACP (New ACP) et l'ancienne BC par la nouvelle;
c. prend des décisions pour inhiber chacune des vieilles connexions **Con j** après avoir testé leur statut d'inactivité.Si une connexion est passive, l'agent l'inhibe et active celles nouvellement créées pour y transférer les derniers états en cours.

Ceci est réalisé par l'AE en temps réel jusqu'à ce que la configuration désirée remplace l'ancienne (Figure 49 (b)).
L'agent proposé dans ce paragraphe se caractérise par une autogestion des reconfigurations. Cependant, l'augmentation croissante des règles de reconfiguration et la distribution des applications limitent son emploi. Nous nous sommes donc orientés vers une structure plus complexe capable de suivre la distribution et la complexité des architectures logicielles multimodales. Cette structure sera également constituée d'un système multiagent (SMA), développé sous la forme d'un AE logiciel.
Nous considérons que cette nouvelle structure multiagent de l'AE est une version améliorée et raffinée de l'AE présenté ici. L'utilisation de cet AE logiciel permettra sa réutilisation car sa structure et son organisation interne (le SMA) restent valables pour tous types d'architectures multimodales.

Seule une mise à jour de sa base de connaissance est nécessaire. Quand à notre choix d'un SMA pour l'implémentation de l'AE, il est motivé par les raisons énoncées ci-après.

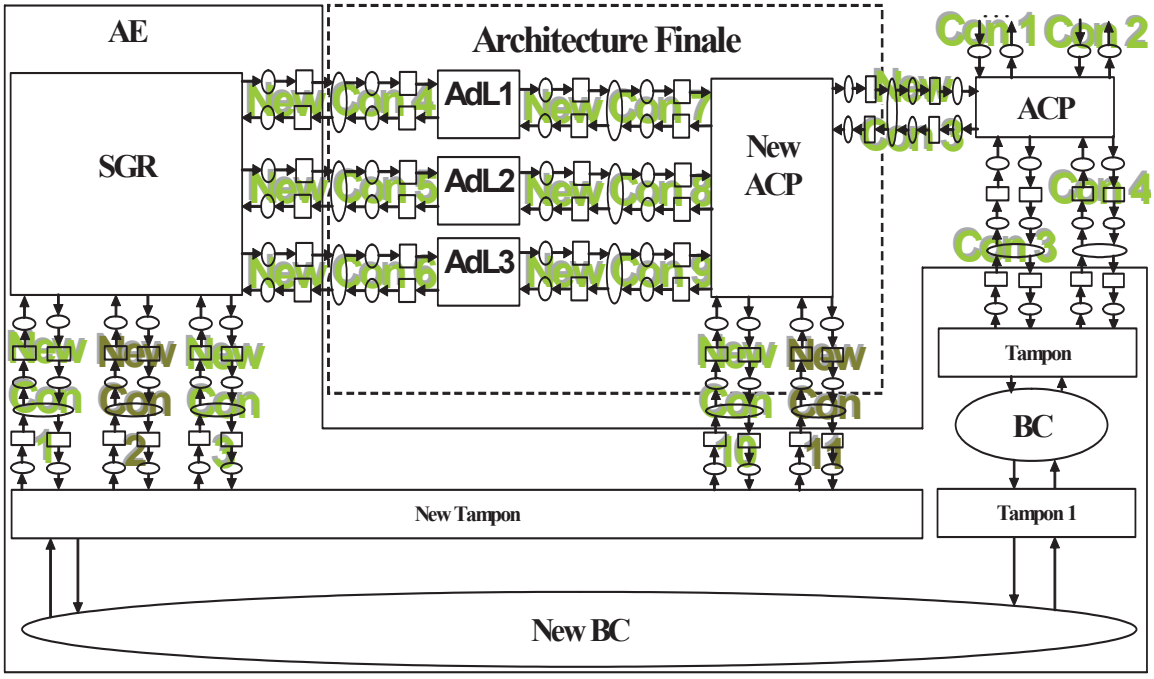

Figure 49 (b) Architecture finale désirée (New: nouveau ou nouvelle, Con : connexion, BC: Base de Connaissances, SGR : Système de Gestion des Règles)

a. La décomposition des tâches : les SMA permettent de résoudre des problèmes complexes en les décomposant en un ensemble de tâches simples. Chaque agent (ou groupe d'agents) résout la tâche qui lui est assignée de telle sorte que l'ensemble des sous solutions forme une solution globale dite de coopération entre agents.
b. L'autonomie des agents : le paradigme agent appartient au domaine de l'intelligence artificielle. L'utilisation d'agents dits « intelligents », donc dotés de capacités cognitives, permet d'automatiser les processus de reconfiguration et admet une autogestion des agents.

c. La réactivité des agents : les agents sont réactifs à tout stimulus grâce à leurs aptitudes à percevoir leur environnement. Les informations perçues par les agents peuvent être ainsi échangées et distribuées. Une centralisation de l'information permet une analyse globale du système.
d. La flexibilité des SMA : les systèmes multiagent permettent une grande évolutivité dans leurs structures (remplacement d'agents) ou dans leurs fonctions (mise à jour des bases de connaissances et des rôles.)
e. La stabilité des SMA : l'organisation dans un SMA décrit toutes les règles de comportement des agents, ainsi que leurs modes de communication. La stabilité dans les SMA est conditionnée par la rigueur de son organisation. Cette rigueur ne peut se concrétiser qu'avec une modélisation formelle comme celle que nous faisons avec les RPCT.

5.4 Structure générale de l'AE

L'AE présentée au paragraphe 5.3 devient vite insuffisant pour traiter des cas complexes de reconfiguration nécessitant la gestion d'un ensemble de qualités architecturales.
Nous le remplacons donc par un SMA à couches. L'Agent d'Expertise et l'Agent Interface deviennent maintenant des couches. Chacune d'elles est peuplée[30] d'agents spécialisés (Figure 50).
Nous distinguons deux principales couches dont les caractéristiques dépendent du type d'agent qu'elles comprennent. La couche la plus évoluée est appelé couche décisionnelle. Les agents y présentent des fonctionnalités cognitives, offrant ainsi la capacité d'organiser un groupe d'agents situés dans la couche inférieure. La seconde est chargée d'appliquer des activités de reconfigurations à l'architecture. C'est la couche réactive de l'AE. Elle enferme des agents dits réactifs.

[30] Il est commun d'employer un vocabulaire anthropomorphique pour les SMA, comme le(a) lecteur(rice) de ce mémoire a déjà pu le constater.

Ces deux couches sont reliées par des canaux de communication. Ce réseau de communication entre les couches permet aux agents l'échange de messages et une meilleure coopération.

Nous nous focalisons dans ce chapitre sur l'aspect fonctionnel de cette AE ainsi que sur les méthodes utilisées pour la reconfiguration dynamique des architectures, le maintien et l'évaluation des attributs de qualités.

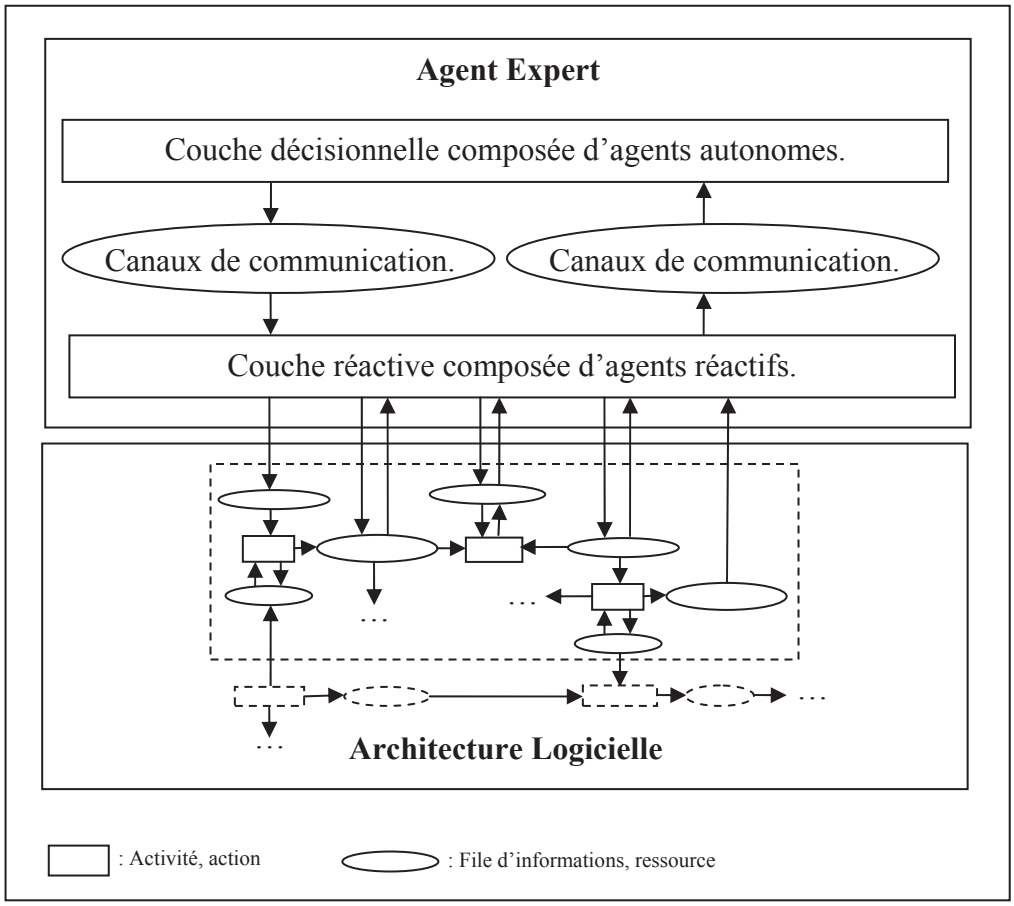

Figure 50 Structure générale de l'AE

5.5 Fonctionnalités générales de l'AE

L'AE offre deux fonctionnalités de base : la surveillance et la reconfiguration de l'architecture. Le processus de surveillance de l'architecture a été modélisé pour répondre au plus vite à une situation critique comme la détérioration d'une qualité logicielle.
La réponse de l'AE consiste à appliquer des modifications spécifiques sous forme de solutions architecturales sur la configuration initiale de l'architecture. Mais avant de décrire ces fonctionnalités, voici quelques conditions préliminaires à son utilisation.

Connaissance de l'architecture : C'est l'une des principales conditions pour pouvoir faire la moindre modification sur l'architecture multimodale candidate. Une bonne connaissance des éléments de l'architecture, de sa structure et de son organisation permet de mieux analyser son comportement dans des situations dites naturelles ou exceptionnelles et d'anticiper son comportement normal ou anormal.
L'objectif à long terme et de pouvoir transformer l'architecture candidate sans pour autant affecter son intégrité. Ces connaissances permettent aux intervenants, participant à la conception de l'architecture multimodale et de l'AE, de mettre au point des scénarios de reconfiguration.

Décomposition des tâches : l'objectif de l'AE est d'atteindre la ou l'ensemble des qualités souhaitées par les intervenants à travers des reconfigurations complexes de l'architecture. Tout le défi réside dans la description de la solution architecturale via des étapes simples comme, par exemple, l'ajout ou la suppression d'une instance d'un composant : c'est ce que nous entendons par 'décomposition des tâches'.

5.5.1 Fonctionnalité de reconfiguration

5.5.1.1 Principe général

L'AE opère par actions ou modifications sur l'architecture grâce à sa couche inférieure dite couche réactive. C'est à elle qu'incombe la responsabilité de modifier les éléments de l'architecture. Les modifications opérées par les agents au niveau architectural incluent :

a. l'addition ou la suppression d'instances de composants;
b. la modification de la configuration et/ou l'extension de l'architecture en termes d'activation ou d'inhibition des connecteurs entre les composants.

5.5.1.2 Exemple

La Figure 51 présente un exemple symbolique d'une configuration primaire. Sur cette figure, par souci de clarté, chaque ellipse annotée du nom '**Composant x**', (**x** appartenant à {**a**, **c**, **d**, **e** }) symbolise en fait une file pouvant contenir une instance du composant **x**. Cette convention de représentation restera valable pour la Figure 52.
Afin d'alléger la figure, seules les portions du réseau de Petri mises en jeux à chaque étape du processus de configuration ont été représentées. Le processus modélisé par la Figure 51 est l'extension de l'architecture initiale composée de deux composants **e** et **c**.
Pour cela l'AE doit générer une instance d'un nouveau composant **a** rattachée au composant **c**. La procédure nécessite l'utilisation de deux agents réactifs, qui seront générés pour les besoins de la reconfiguration. L'opération se déroule en trois étapes :

a. création (ou génération) d'instances d'agents de la couche réactive;
b. ajout d'une instance du composant **a**;
c. activation du connecteur **1** et enfin activation du composant **a**.

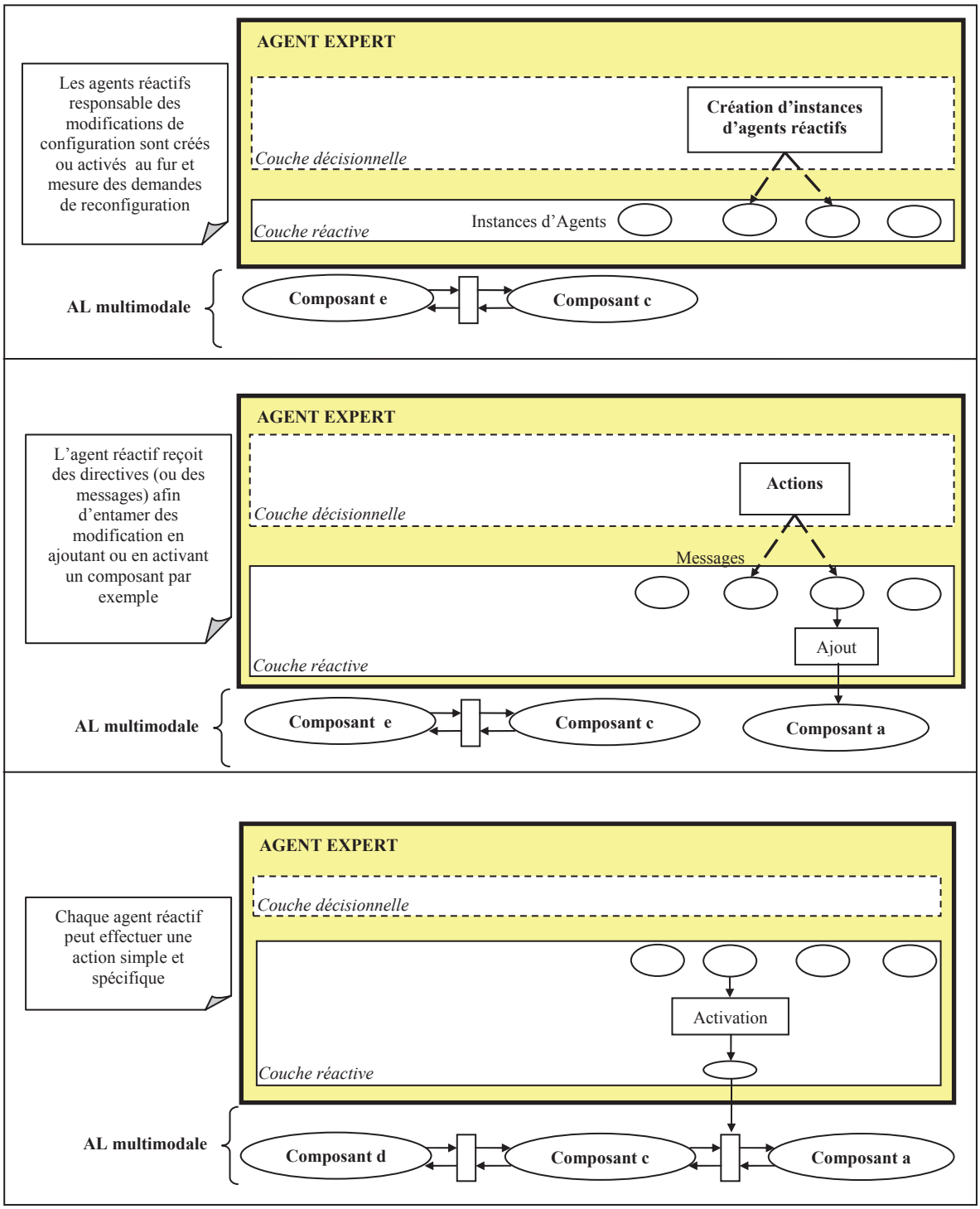

Figure 51 Principe de configuration sur une architecture candidate

5.5.1.3 Processus de Monitoring des architectures

5.6.1.3.1 Avant-propos

La surveillance est un processus largement utilisé en informatique. Il a été souvent utilisé pour améliorer la fiabilité des systèmes critiques grâce à l'emploi de différentes techniques comme celle de la tolérance aux erreurs. Un système de surveillance a pour fonction l'évaluation de certains paramètres devant être maintenus ou améliorés. La collecte de ses paramètres est souvent basée sur une métrique.

L'utilisation d'un processus de surveillance est une pratique extrêmement délicate. Dans la référence (Littlewood 2000), l'auteur explique l'intérêt de la surveillance dans l'industrie de l'aviation civile. Il nous apprend que « *rapporter les erreurs et les fautes permet de les corriger rapidement. L'utilisation des processus de détections et de signalisations des erreurs a permis à l'industrie de l'aviation civile la rectification de ses équipements et l'amélioration de ses procédures de fabrication et donc de la fiabilité de sa production*».

L'implémentation d'un système de surveillance diffère d'une approche à une autre. Par exemple, dans la référence (Asterio 2003), les chercheurs ont implémentés un processus de détection des erreurs au cœur du composant. De ce fait, la détection des erreurs est localisée et la réaction du système devient plus rapide. Cependant, l'implémentation de nouveaux processus de détection implique des modifications importantes dans le code du composant et le redéploiement de l'application.

Les systèmes de surveillance peuvent être utilisés comme gestionnaires d'alertes pouvant déclencher des processus pré programmés. Cette technique a été largement utilisée pour l'adaptation et la maintenance des systèmes critiques. Par exemple, Garlan utilise cette technique dans ses travaux (Garlan 2003). Il y explique qu'« *afin d'illustrer l'idée d'adaptation dynamique de l'architecture logicielle, la surveillance du comportement du système par rapport à sa performance déterminera l'autoréparation de l'application Web service basée sur une architecture client/serveur»*. Dans cette

article, le système de monitoring appelé Remos (pour '*Resource Monitoring System*') (Garlan 2001) et développé par le Computing Media and Communication Laboratory de la prestigieuse université de Carnegie Mellon, permet la surveillance des architectures de type Client/Serveur. Ce outil est utilisé pour le test et la mesure des ressources réseaux disponibles. Il produit des résultats métriques et graphiques.

L'approche de Garlan consiste à utiliser plusieurs '*frameworks'* pour l'adaptation et la reconfiguration des architectures Client/Serveur afin de maintenir la fiabilité de ces systèmes. Le système Remos détecte les problèmes liés à la performance et à la fiabilité des architectures et déclenche une alerte afin de procéder aux adaptations. Pour notre part nous emploierons des modes de surveillances décrits au paragraphe suivant.

5.6.1.3.2 Principes employés dans l'AE

La couche inférieure de l'AE est chargée de la perception des événements provenant de l'architecture et de son environnement. Les éléments surveillés de l'architecture et de son environnement sont définis préalablement dans les scénarios de reconfiguration. D'autre part, cette couche joue le rôle de couche réactive de l'AE. Diverses fonctionnalités sont attribuées à la couche inférieure de l'AE.

Le traitement d'un grand nombre d'événements est la première fonctionnalité de cette couche. Les agents de cette couche réactive sont des agents spécialisés. Chaque agent est conçu pour la perception d'un type d'évènement spécifique provenant d'un élément architectural précis. Le traitement des signaux se fait avec une forte granularité du parallélisme (d'où l'intérêt d'une modélisation par réseaux de Petri). L'AE peut ainsi traiter un grand nombre de signaux en même temps.

Une autre fonctionnalité est l'implémentation de différents moyens de détection. Les processus de mesure et de test de l'architecture peuvent être adaptés et peuvent évoluer à l'intérieur de l'AE. Cela se réalise par la mise à jour des connaissances des agents ou encore par l'ajout d'instances de nouveaux agents. L'automatisation des mécanismes de réplique est une fonctionnalité qui permet à l'AE de réagir en se basant sur les

informations reçues de son environnement. Il intègre des mécanismes de réaction spécifiques à chaque type de signal qu'il reçoit via ses agents sensitifs. L'archivage des mesures est une fonctionnalité non négligeable. Les signaux reçus par les agents réactifs de la couche inférieure peuvent être sauvegardés dans un historique par l'AE.

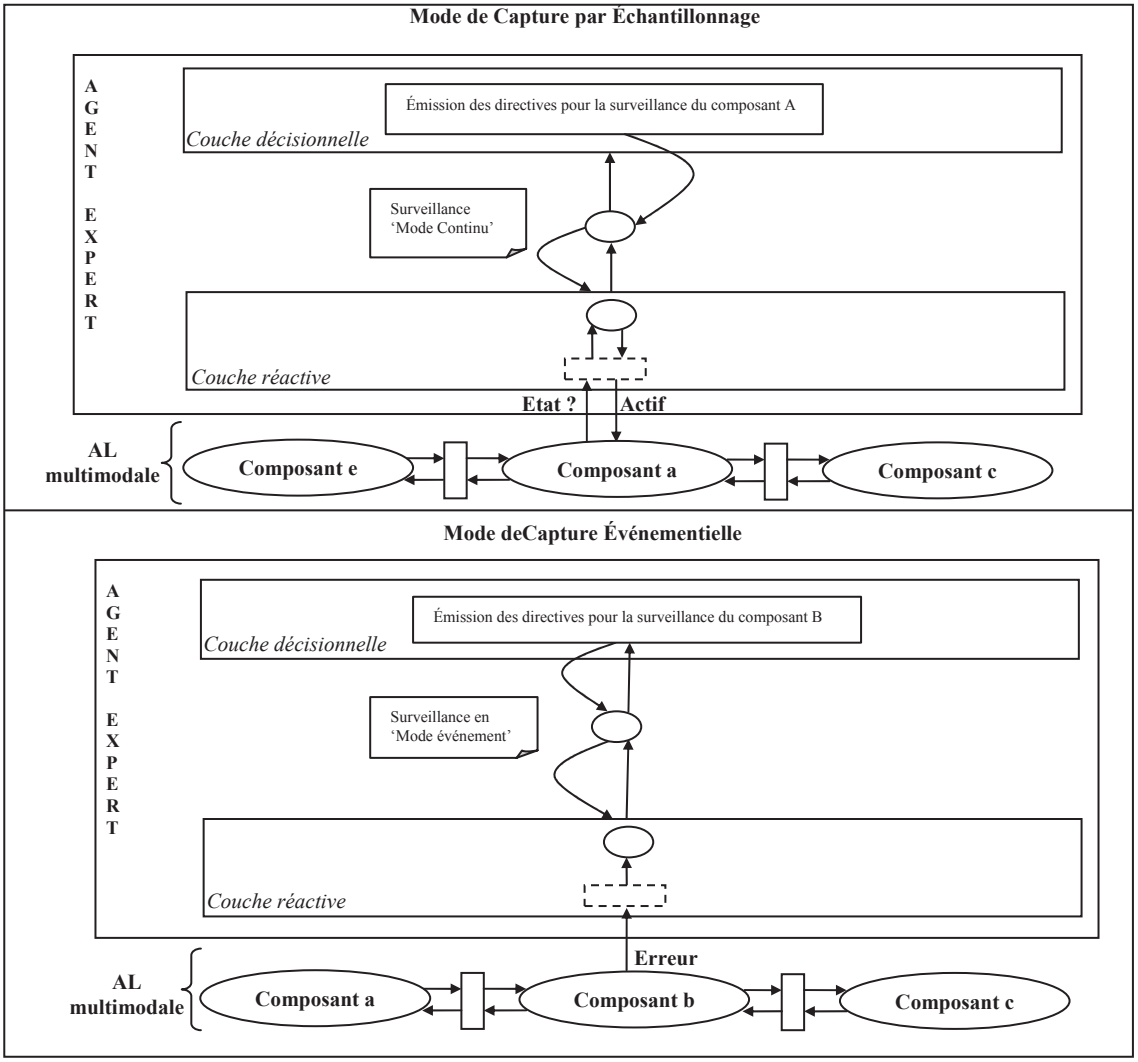

Figure 52 Principe des modes de surveillance de l'architecture

La technique de monitoring (ou surveillance permanente) permet de maintenir et d'évaluer le niveau des qualités d'une architecture candidate. La surveillance permanente de certaines zones prédéfinies de l'architecture ou zones dites sensibles permet de réagir plus rapidement aux exigences en qualité. Pour ce faire nous proposons deux modes d'observation (Figure 52): l'un est dit par capture d'évènement l'autre par échantillonnage.

La surveillance par capture d'événements repose sur l'interception d'événements prédéfinis, sous forme de messages ou d'exceptions (dans le cas d'erreurs).

La surveillance par échantillonnage d'une architecture candidate utilise une connexion directe avec ses éléments. Généralement, l'échantillonnage consiste à vérifier l'état d'un composant ou d'un connecteur à un intervalle de temps régulier (période d'échantillonnage). La réponse retournée est sous forme booléenne (valeur vraie ou fausse). La fréquence d'échantillonnage est paramétrable.

5.5.2 Processus pour maintenir des critères de qualité

5.5.2.1 Introduction

L'autonomie de l'AE repose sur sa partie décisionnelle. Elle a à sa disposition deux fonctionnalités de base : la capacité de surveiller et celle de modifier l'architecture

5.5.2.2 Mise en place des processus de raisonnement

C'est la couche supérieure de l'AE qui constitue sa partie décisionnelle. Les types d'agents se distinguent par leurs aptitudes à résoudre des problèmes donnés en effectuant la mobilisation d'une population d'agents via des capacités de gestion et d'organisation. Nous nous contenterons dans cette section de décrire les processus de raisonnement (Figure 53) utilisés par les agents de la couche supérieure afin de résoudre

les problèmes proposés sans aborder le type de coopération et le protocole de communication entre les agents.

Dans notre cas, le problème complexe que doit résoudre l'agent intelligent est la maintenance de critères de qualités. Chaque critère est conditionné par des contraintes que doit respecter l'architecture. Si celles-ci ne sont pas satisfaites, une succession d'actions est déclenchée (Figure 53).

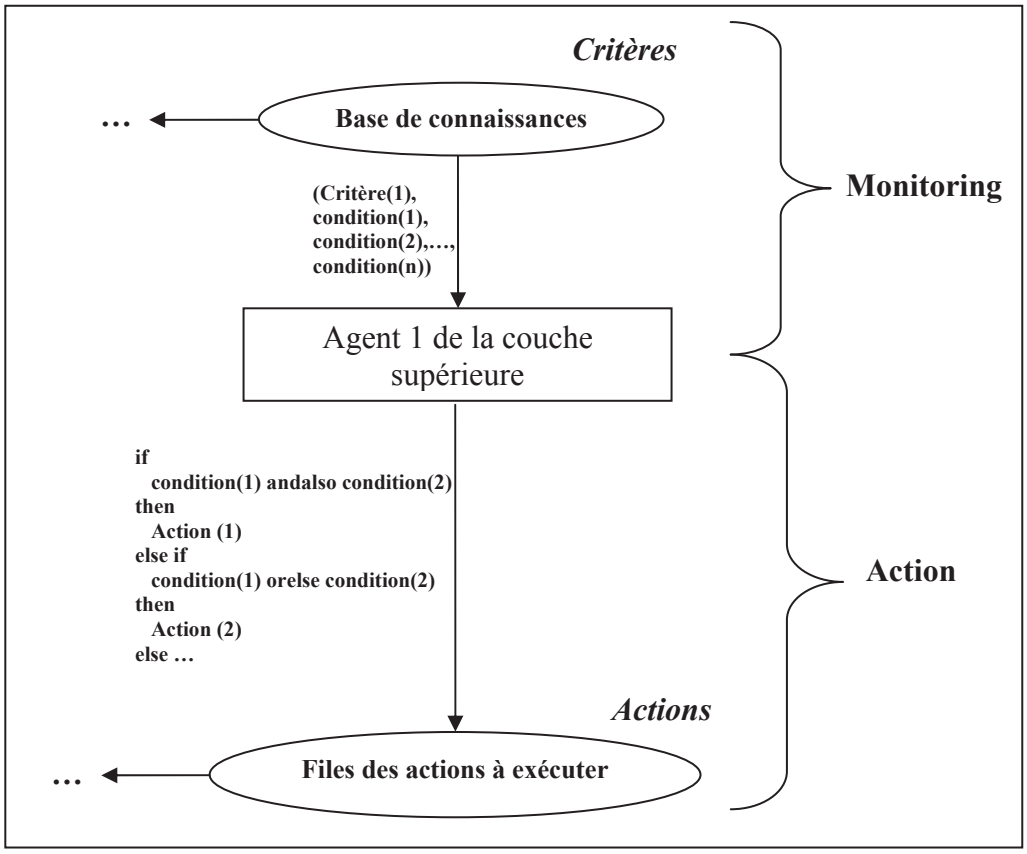

Figure 53 Spécification de la base de connaissance et du processus de raisonnement de l'agent

Les contraintes et les actions à entreprendre sont dictées par les intervenants, qui décrivent :

a. les conditions : c'est à dire les circonstances ou le moment d'intervention de l'AE (dans le cas contraire il restera passif);
b. les actions qui retracent le déroulement des modifications sur l'architecture afin de passer d'une configuration initiale vers une nouvelle configuration de l'architecture permettant ainsi de satisfaire les objectifs de qualité.

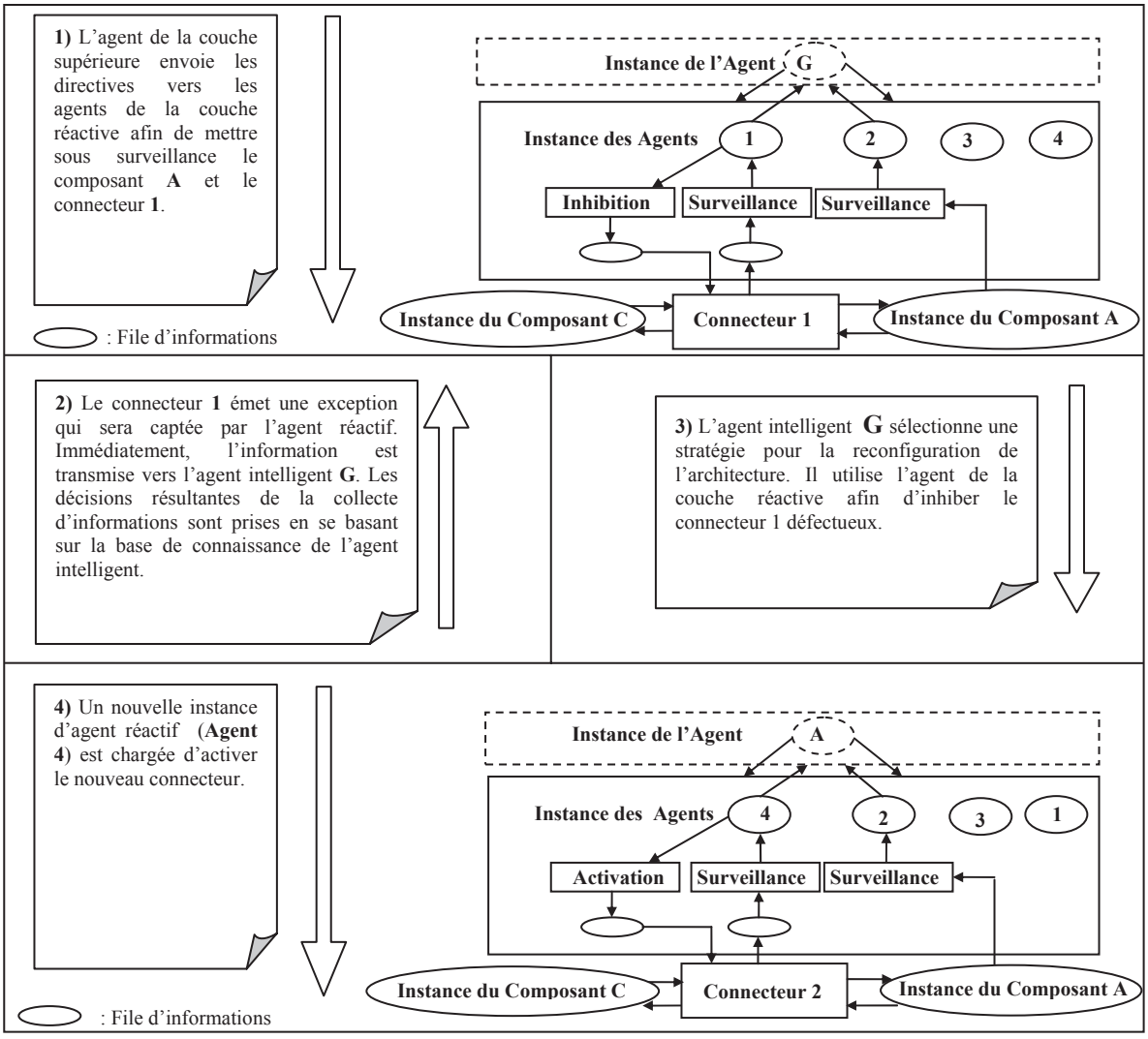

Figure 54 Exemple en 4 étapes d'un processus de reconfiguration d'une architecture candidate

Pour ce faire deux transitions (une d'activation et l'autre d'inhibition) sont prévues pour agir sur chaque connecteur.

Ces transitions ne sont pas systématiquement représentées à la Figure 53 pour des raisons de clarté. Les ellipses symbolisant les agents ou composants sont des files d'instances d'agents et/ou de composants.

La solution architecturale proposée est sauvegardée dans la base de connaissance d'un agent de la couche supérieure.

Un exemple de processus de reconfiguration de l'architecture est décrit à la Figure 54. Dans cet exemple, chaque connecteur ou composant de l'architecture est surveillé selon le mode événementiel. De plus, tous les connecteurs peuvent être inhibés ou activés.

5.5.3 Conclusion

Nous avons présenté les fonctionnalités générales qu'offrait l'AE. La partie réactive de celui-ci, parachevée par sa partie décisionnelle, oriente notre approche vers le dynamisme intelligent des architectures multimodales. Afin de maintenir les qualités d'une architecture multimodale, nous avons utilisé la notion de critères conditionnels dans l'élaboration de la base de connaissance de l'agent intelligent. C'est sur celle-ci que les stratégies de reconfiguration vont être adoptées par l'AE. Cette technique pose les bases nécessaires pour décrire toutes les propriétés du système, l'état de l'environnement et les caractéristiques de la qualité. Elle constitue aussi une modélisation, à un niveau d'abstraction élevé, qui permet au développeur de comprendre les processus dynamiques à implémenter. Dans le paragraphe suivant nous présentons l'élément de base utilisé par l'AE pour son fonctionnement : le scénario multimodaux.

5.6 Reconfiguration basée sur les scénarios

5.6.1 Approche basée sur le scénario

Des mécanismes comme l'ajout, la suppression ou le fait de raffiner des composants/ou des connecteurs sont suffisants pour une simple reconfiguration (paragraphe 5.4). Cependant, les opérations de reconfigurations sont beaucoup plus complexes. Le processus de modification d'une architecture doit d'une part modifier une configuration donnée, mais aussi garantir l'intégrité globale de l'application et donc de l'application d'autres scénarios déjà en cours d'exécution.

Le fait d'employer des critères comme connaissances de base des agents permet de comprendre le fonctionnement global de l'AE et les processus engagés par ses agents. Cependant, la description apportée par des critères ne tient pas compte de tous les paramètres entrant en jeu lors d'une reconfiguration. Par exemple, l'impact des modifications sur les qualités déjà existantes ou encore, la coexistence de plusieurs qualités n'est pas prise en compte. Des questions concernant les critères restent sans réponses pertinentes quant aux contextes de la reconfiguration et au déploiement de la solution architecturale.

Pour apporter des solutions à ces questions, des méthodes d'évaluation comme l'ATAM ou la SAAM (Kazman 1996; Bass 2001; Clements 2002) analysent toutes les interactions entre les intervenants et le système puis, déduisent les modifications pouvant être adaptées à l'architecture. Il s'agit d'atteindre une qualité à travers une solution architecturale. Le succés de ces méthodes réside dans l'utilisation de scénarios pour la description de la problématique et des différentes solutions envisageables. Les avantages d'adopter des méthodes de ce type sont :

a. une meilleure spécification des besoins en reconfiguration : le scénario décrit une interaction entre le système et les intervenants (ceci permet à chaque intervenant d'exprimer ses besoins, selon le type d'interaction qu'il a avec le système);

b. une maîtrise de tous les paramètres de contexte et de modification : les méthodes d'évaluation utilisent les scénarios dans le but de décrire l'environnement du système, les qualités souhaitées et la configuration de l'architecture pour les atteindre;
c. une facilité d'expression et d'utilisation : le scénario est un moyen relativement simple pour exprimer des besoins.

Les scénarios semblent donc la solution adéquate pour l'expression des besoins en qualité. Utilisés dans notre approche, ils apportent une meilleure description de la problématique et des solutions architecturales. Nous utiliserons donc les scénarios mais selon un approche différente de celle des méthodes SAAM et ATAM. Notre approche ne diffère pas fortement dans la définition du concept de scénario, mais plutôt dans le processus de son intégration au sein de l'AE et également dans son mode d'application pour maintenir un profil de qualité dans une AL multimodale. Ces deux points constituent la nouveauté de notre approche

5.6.2 Définition d'un scénario de reconfiguration

Plusieurs définitions du scénario existent. Nous avons sélectionné les deux plus pertinentes :

> *« Nous utilisons le concept de scénario pour décrire quelles sont les qualités que nous devons atteindre. Le scénario décrit comment l'architecture répond à certain stimulus » (Bass 2001).*

> *« Le scénario est un court rapport, décrivant l'interaction entre les intervenants et le système. Le client décrit son utilisation du système pour achever certaines tâches, son scénario ressemble assez à un cas d'utilisation. L'équipe de maintenance décrit les changements opérés sur le système, comme la mise à jour, ou l'ajout de nouvelles fonctionnalités. Les architectes exposent l'architecture du système et prédisent ses performances. Le client décrit comment l'architecture va être réutilisé pour un second produit. » (Bass 2001).*

Nous allons donc adapter le concept du scénario à notre approche en élargissant sa définition donnée dans la littérature. Nous définissons le scénario de reconfiguration architecturale comme suit.

Le scénario de reconfiguration architecturale est un processus dynamique à travers lequel tous les états internes ou externes du système sont pris en compte afin de définir son comportement -une ou plusieurs séquences d'actions successives et/ou en parallèles- face à une situation bien précise -un ensemble d'états et de ressources contextuels et environnementaux- dans le but d'atteindre une qualité prédéfinie, sans pour autant en détériorer totalement une autre.

La définition que nous proposons se prête parfaitement à une modélisation par les RPCTS : les séquences d'actions sont modélisées par des transitions ; les états, ainsi que les ressources sont modélisées par des places.
Par ailleurs, elle implique des conditions que les différents scénarios doivent respecter pour maintenir un compromis entre les qualités logicielles de l'architecture.

5.6.3 Structure des scénarios destinés à l'AE

5.6.3.1 Introduction

L'avantage de l'approche proposée réside dans la relative facilité d'utilisation des scénarios. Les scénarios que nous allons implémenter dans l'AE diffèrent des scénarios classiques, principalement par leurs structures mais aussi par certains aspects. Les scénarios implémentés dans l'AE sont destinés à une AL multimodale: d'où la nécessité de développer un modèle pour les scénarios.
Par ailleurs, ces scénarios proposent une solution architecturale à un problème de qualité, à un instant « t », lors de l'exécution du système. Cette solution architecturale, reste relativement temporaire et peut être changée par une autre lorsque le contexte

diffère, en temps réel d'exécution. Nous restons donc dans le cadre d'une spécification qui tient compte du paramètre 'temps' en employant les RPCTS pour modéliser l'AE et l'exécution de scénarios par ce dernier.

5.6.3.2 Organisation et structure des scénarios

L'expression des besoins par les intervenants au début de la conception de l'architecture reste (pour notre approche) identique à celles des méthodes classiques (ATAM et SAAM (Clements 2002)). Elle prend la forme de questions ou de phrases simples obtenues par l'utilisation de questionnaires ou de check-lists (voir Figure 55), pendant des séances de prospections d'idées.
Ces besoins évoluent en scénarios de reconfiguration relatifs à un profil de qualité. En effet, chaque interrogation des intervenants peut se traduire par un scénario d'interaction avec le système dans le but de cibler une caractéristique d'une qualité.
Enfin, chaque scénario général est analysé par rapport aux solutions architecturales qu'il apporte, afin d'obtenir un scénario de reconfiguration qui tient compte d'un profil de qualité. La transformation des scénarios généraux en scénarios de reconfiguration permet d'avoir la structure finale des scénarios utilisés par l'AE. Il s'agit donc d'un processus de raffinement du scénario général en un scénario plus détaillé qui permettra la reconfiguration selon un modèle de qualités.
Ceci est possible grâce à l'organisation des scénarios selon une structure offrant aux intervenants le moyen d'uniformiser l'expression de leurs besoins et, à l'AE, des solutions implémentables.
Nous définissons une structure globale pour le scénario composée de six principaux paramètres (voir Figure 56).

Le **stimulus** est l'élément déclencheur du scénario. Le signal est interprété par la partie décisionnelle de l'AE comme l'amorceur de l'exécution des modifications. Il informe les agents intelligents de la faiblesse de certains des critères de qualités surveillés. Le

stimulus conduit donc l'AE à entamer toutes les procédures nécessaires afin de rétablir la situation.

L'expression conditionnelle qui suit le « si » (Comment réagit le système **si** …?) identifie le stimulus. Ce dernier peut représenter un évènement, un état ou une valeur calculée.

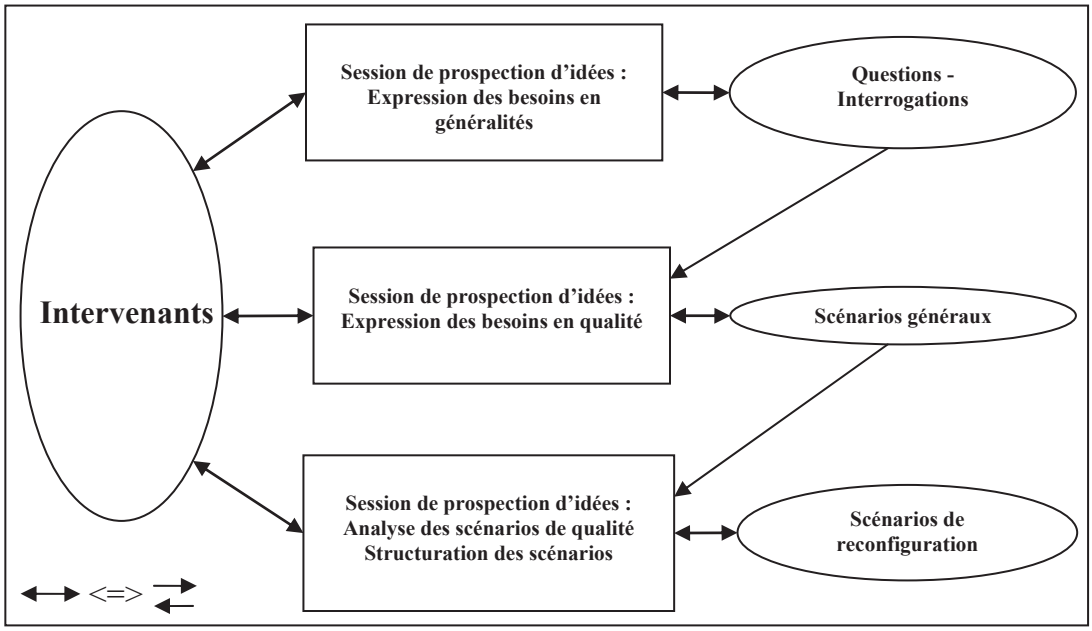

Figure 55 Travail préliminaire des intervenants pour définir les scénarios

Un évènement peut provenir de l'architecture ou de son environnement. Il s'agit d'événements internes (par exemple : une exception) ou d'événements externes au système (message provenant de l'environnement, d'un autre système ou de l'utilisateur).

Un état décrit le comportement d'un élément de l'architecture (par exemple : l'état du composant **a** qui est désactivé).

Une valeur est généralement obtenue par une mesure ou calculée par une expression mathématique (par exemple : le temps de traitement d'un composant de reconnaissance vocale).

Le **contrôle** est défini par l'identification des points à risque ou sensibles de l'architecture du système ou de son environnement (Guizzardi 2003). Le contrôle permet la surveillance des points définis par le scénario. Il donne une information sur l'état externe (utilisateur, autre système, environnement, etc.) ou interne (composants dispositifs et connecteurs relatifs à une modalité) de l'architecture.

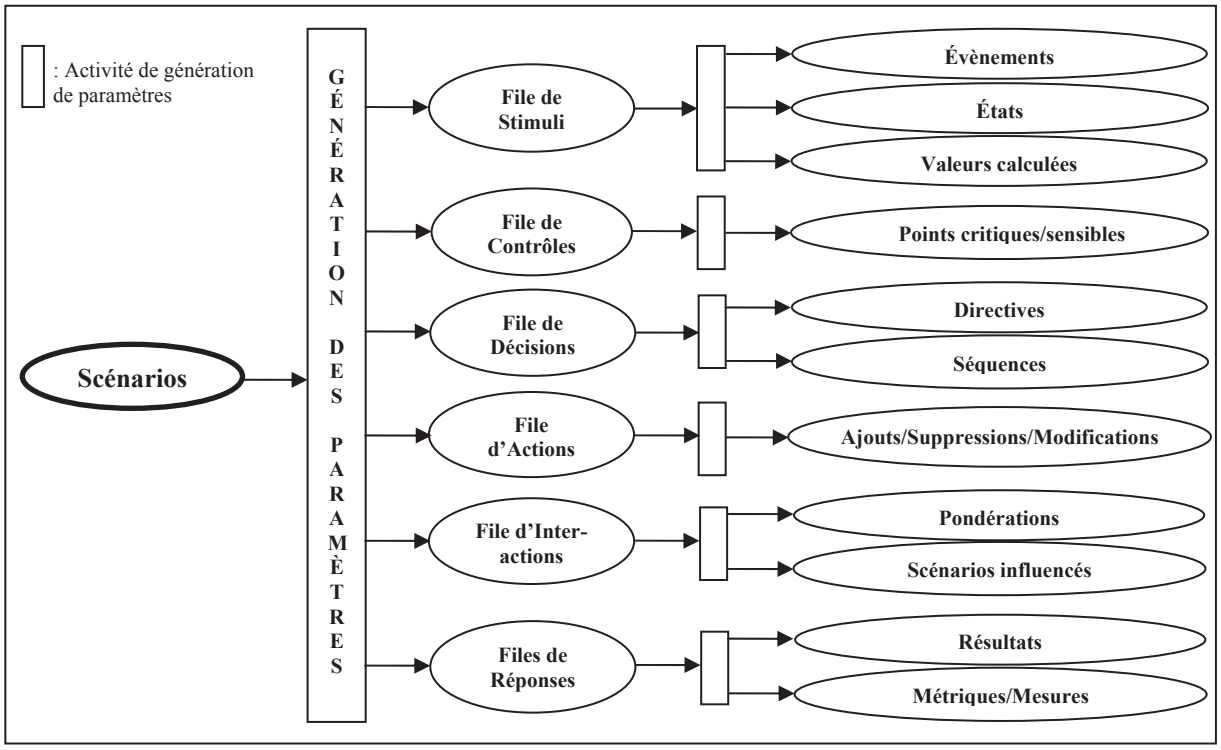

Figure 56 Modèle de génération des paramètres du scénario

Les **actions** représentent les actions simples sur les éléments de l'architecture comme l'ajout/modification/suppression/inhibition/activation d'un composant, d'un dispositif ou d'un connecteur relatifs aux modalités de l'architecture.

Les **décisions architecturales** correspondent à celles proposées par les intervenants. Les informations apportées par ce paramètre décrivent les étapes de modification de l'architecture candidate.

Les **interactions** sont des paramètres qui définissent les points d'interférence possibles entre les scénarios. Elles décrivent les scénarios pouvant s'influencer négativement ou ayant des objectifs contradictoires.

La **réponse** est le résultat apporté par le scénario. L'AE produit des résultats : une documentation textuelle du système (qui contribue à la traçabilité), des résultats de mesure[31] ou une métrique[32] suite à une simulation d'une reconfiguration.

5.6.3.3 Interactions entre les scénarios

Les scénarios sont utilisés de façon simultanée ou individuelle par l'AE. La gestion de cet ensemble dépend du type de priorité entre les scénarios. L'AE devra donc faire cohabiter ses scénarios de telle sorte qu'ils n'interfèrent pas négativement les uns avec les autres. Nous avons identifié trois types d'interactions entre les scénarios.

a. Scénarios d'interférences : les scénarios d'interférences sont des scénarios ciblant l'amélioration d'une qualité mais dont le résultat détériore une autre qualité (par exemple, performance versus sécurité);

b. Scénarios imbriqués ou entremélés : les scénarios imbriqués sont identifiés indépendamment de leurs objectifs de qualité. L'entremêlement de scénarios, exprime le fait que des scénarios peuvent s'exécuter en même temps, mais dont les modifications sur l'architecture impliquent les mêmes éléments architecturaux;

[31] Action d'utiliser une unité de mesure pour évaluer et mesurer une sous caractéristique d'une qualité d'un système informatique.
[32] Indicateur permettant de mesurer la sous caractéristique d'une qualité.

c. Scénarios homogènes : les scénarios homogènes ne produisent pas de conflit à gérer par l'AE. Ces types de scénarios s'influencent positivement. Cependant ils sont tout de même mis en évidence afin de mieux exploiter l'affinité entre scénarios et améliorer leurs apports.

5.6.3.4 Organisation des scénarios

L'organisation des scénarios a pour objectif de permettre à l'AE de gérer un grand nombre de scénarios sans pour autant affecter son efficacité. En effet, l'organisation des scénarios évite les problèmes liés à la détérioration des rendements relatifs à certains critères de qualité tout en veillant à l'intégrité globale du système. Nous attribuons à chaque sous caractéristique d'un attribut de qualité un possible scénario. Ainsi l'organisation des scénarios tente de se calquer à la structure organisationnelle du SMA constituant l'AE. Nous prônons donc une organisation des scénarios qui sera arborescente tout comme la structure d'un modèle de qualité qui se décompose en qualité, caractéristiques et sous caractéristiques.

5.6.3.5 Facteurs d'influence des scénarios

5.6.3.5.1 Détections des scénarios imbriqués

Dans un ensemble de scénarios, les actions apportées par l'AE ne peuvent pas concerner une même modalité en même temps : d'où l'importance de développer un système de détection des scénarios imbriqués avant même leurs importations par l'AE. À travers les descriptions des éléments de l'architecture apportées par les scénarios, nous pouvons aisément identifier les noms des composants pouvant faire l'objet de modification simultanément.

5.6.3.5.2 Identification des scénarios d'interférence

L'identification des scénarios d'interférence est une tâche qui est intégrée dans l'AE. Cette identification est déteminée par les intervenants au début de la phase de conception de l'architecture de l'AE. Ces derniers sont, en effet, les mieux habilités à juger si leur système doit mieux respecter une qualité plutôt qu'une autre dans un contexte déterminé.

5.6.3.5.3 Pondération des scénarios

Chaque scénario est pondéré par les intervenants selon un contexte donné. Cette pondération a son importance vis à vis de tous les scénarios. Elle est ensuite embarquée dans l'AE sous forme de paramètres de priorité. L'AE peut ainsi choisir, dans des circonstances bien précises, le meilleur scénario à exécuter par rapport au contexte dans lequel se trouve l'application.

5.6.3.6 Modèle global du scénario

La mise en place d'un modèle pour les scénarios nous permet de construire un « profil des qualités » de l'architecture candidate de l'AE. La Figure 57 présente le modèle des qualités pour un système quelconque. Ce modèle comporte les différents éléments suivants.

a. le profil de qualité : c'est la liste des qualités souhaitées;
b. les attributs de qualité : chaque qualité, nommée plus haut, est maintenue par un groupe de scénarios classés par type l
c. e scénario : il décrit les paramètres du scénario et correspond à une caractéristique d'une qualité donnée;

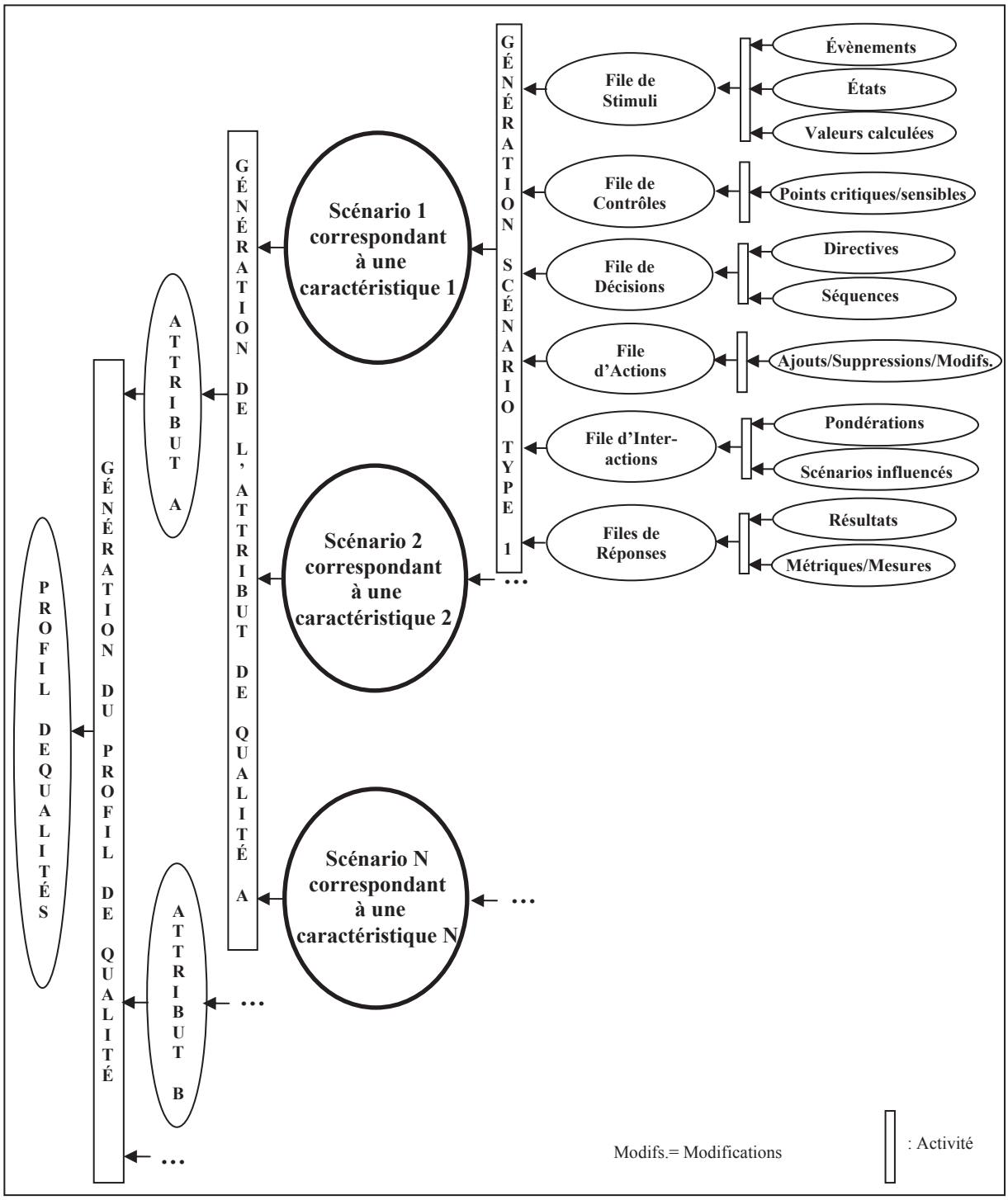

Figure 57 Modèle d'organisation et de génération des profils de qualité

5.7 Conclusion du chapitre 5

Nous avons décrit dans ce chapitre les fonctionnalités générales de l'AE et, en particulier, ses aptitudes en termes de raisonnement et d'autogestion. Nous utilisons l'AE comme processus externe à l'architecture multimodale afin de modifier sa configuration quand cela devient nécessaire. Les situations d'engagement des reconfigurations sont conditionnées par la définition des niveaux des qualités du système. Nous devons pour cela décrire le système selon un modèle général de profil de qualité, rattaché à un ensemble de scénarios de reconfiguration. L'AE ne se contente pas d'appliquer les scénarios de reconfiguration : il les administre aussi afin de gérer toutes les interférences que peuvent avoir certains scénarios sur le profil de qualité globale du système.

Dans la suite de ce mémoire, nous présentons la mise en œuvre de l'AE. L'étude porte sur le SMA, avec ses aspects fonctionnels et structurels. Cette étude présente une application à un exemple multimodal d'un cas choisi de profil de qualité.

CHAPITRE 6

RECONFIGURATION SELON UN PROFIL DE QUALITÉ D'UNE APPLICATION MULTIMODALE DÉDIÉE AUX HANDICAPÉS MOTEURS : ANALYSES STRUCTURELLES ET FONCTIONNELLES DE L'AGENT EXPERT

6.1 Introduction

Ce chapitre présente la structure déjà esquissée au chapitre précédent et dédiée à la gestion dynamique des architectures logicielles multimodales. La mise en œuvre architecturale de cette structure constitue elle-même un AE. La compréhension de cette structure et de son organisation permet d'expliciter le fonctionnement de l'AE par rapport à un processus dynamique de reconfiguration architecturale multimodale, jusqu'à celle de la simulation de ce processus.

Le paragraphe 6.2 de ce chapitre donne l'organisation individuelle des agents qui constituent l'AE. Il détaille la structure générale de l'AE et les architectures de ses agents (propriétés des agents de chaque couche, leurs rôles, etc.) Ces dernières sont décrites par des modèles en 'réseaux de Petri'.

Le paragraphe 6.3 suivant est employé à la présentation de l'organisation de l'AE à travers l'analyse de ses parties fonctionnelles et structurales ainsi que celle de ses paramètres. D'autres aspects du SMA y sont abordés comme : la coordination des actions et la communication entre les agents lors de l'exécution des scénarios de reconfiguration.

Enfin, au pararagraphe 6.4, un exemple de reconfiguration dynamique, selon un profil de qualité, est exposé.

6.2 Système multiagent à couches hiérarchisées

6.2.1 Introduction

Rappelons que nos recherches dans le domaine des systèmes multiagent dédiés à la multimodalité poursuivent deux objectifs majeurs. Le premier concerne l'analyse théorique des mécanismes qui ont lieu lorsque plusieurs entités autonomes interagissent. Le second s'intéresse à la réalisation de modèles de processus distribués coopérant, capables d'accomplir des tâches complexes, telles que le maintien des qualités d'une architecture multimodale quelconque via des reconfigurations dynamiques. Notre position est donc double : d'une part nous nous plaçons au sein des sciences cognitives et des sciences sociales pour construire des modèles auto organisés; d'autre part nous présentons une technique qui vise la réalisation de systèmes multimodaux dynamiques à partir des concepts d'agent, de communication, de coopération et de coordination d'actions entre agents.

6.2.2 Structure générale de l'AE

La modélisation du problème abordé permet d'entrevoir les caractéristiques et les fonctionnalités de notre SMA. Elle nous permet aussi de proposer la structure la mieux adaptée à la problématique déjà amorcée au chapitre précédent. La Figure 58 présente la modélisation du domaine d'application. L'AE embarque les scénarios de reconfiguration architecturale. Ces informations alimentent sa base de connaissance et cela lui permet de mettre à jour ses règles de comportement. À partir de ces règles et de ces connaissances, l'AE construit les mécanismes d'évaluation des qualités et d'autogestion des reconfigurations de l'architecture. Nous pouvons identifier quatre fonctionnalités de base des agents de l'AE, le choix du type d'agent utilisé étant guidé par des critères sur sa fonction et son rôle :

a. des agents pour l'analyse des informations produites par 'la mesure' d'attributs de qualité;
b. des agents réactifs pour la perception et l'action sur l'architecture;
c. des agents cognitifs pour le raisonnement et la prise de décisions basée sur les connaissances et les règles de comportement;

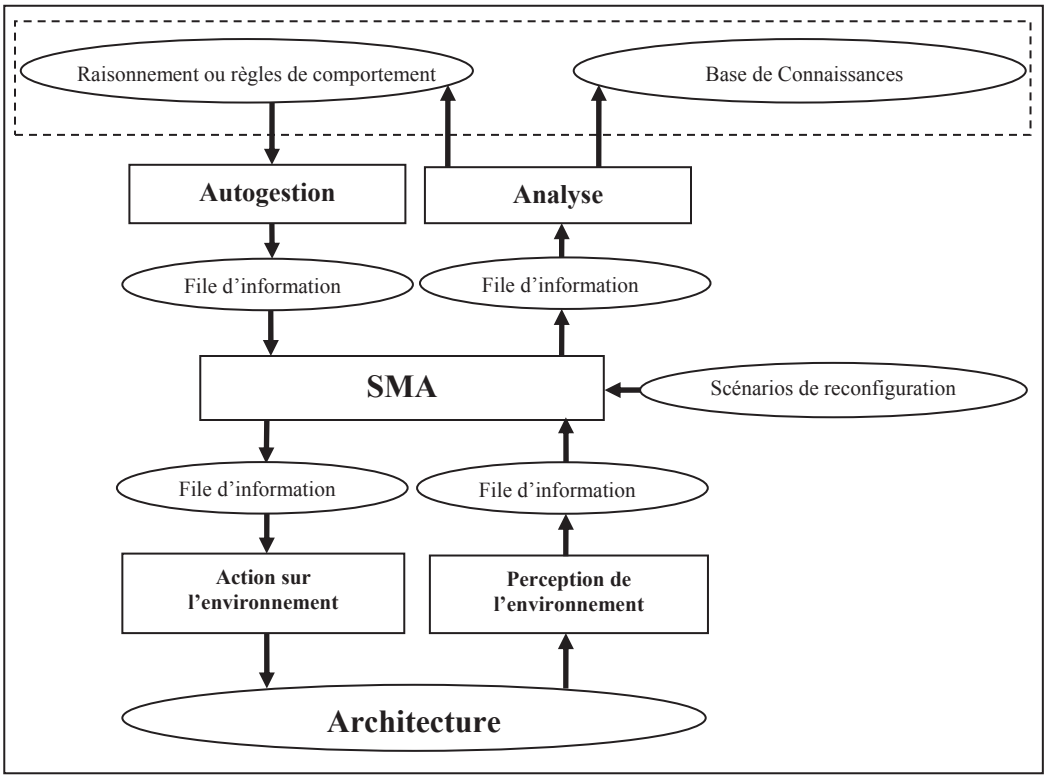

Figure 58 Modélisation du domaine d'application

L'AE est donc un SMA. Le SMA que nous proposons est hétérogène car il utilise trois types d'agents spécialisés et disposés sur des couches spécifiques selon leurs types. Les agents interagissent et communiquent entre eux, mais ne sont pas égalitaires. Ils font partie d'un système hiérarchique appelé « squelette d'agents ».

Les agents cognitifs (Gestionnaire et Décideurs) sont placés au sommet du squelette. Puis, en dessous, se trouvent les agents Réactifs (voir Figure59). Nous qualifions notre SMA de « SMA à couches hiérarchisées, constituées d'agents Gestionnaires, Décideurs et Réactifs». Sur la couche supérieure dite de 'Décision', des agents Gestionnaires et plusieurs agents cognitifs Décideurs y sont implantés.

Chaque agent Décideur dispose de plusieurs agents Réactifs situés dans la couche Réactive de l'AE.

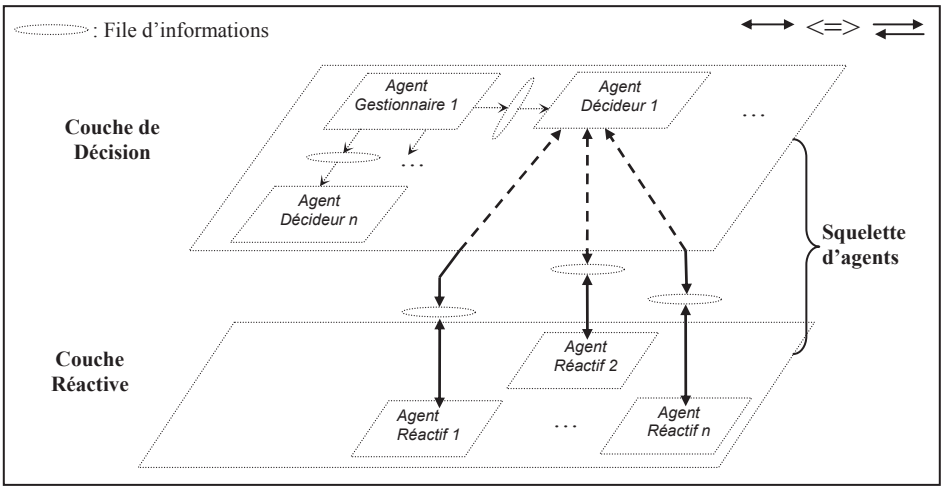

Figure 59 Structure de l'AE

6.2.3 Organisation individuelle des agents

L'organisation au niveau individuel est le point de départ de la constitution d'organisations collectives. Chaque individu d'un système multiagent a sa propre organisation interne. Celle-ci permet de cerner le rôle et le comportement de l'individu dans le système. L'analyse structurelle d'un agent ressemble à celle d'une organisation quelconque, à ceci près que les composants d'un agent ne sont généralement pas des agents eux-mêmes, mais des modules (horizontaux ou verticaux) plus ou moins

spécialisés. Le terme généralement employé pour décrire l'organisation interne d'un agent est celui d'architecture, par analogie avec la structure des ordinateurs. Cependant, dans le contexte d'utilisation de notre SMA nous préférons parler de structure afin de nous distinguer de l'AL multimodale.

Les structures multiagent présentent un éventail très riche de solutions pour la conception de l'organisation interne d'un agent (voir paragraphe 2.3.2). Cette pluralité de solutions est fonction du comportement exhibé par l'agent. Un agent cognitif dispose naturellement d'une structure plus élaborée que celle d'un agent réactif. Mais la division cognitif/réactif n'épuise absolument pas le débat sur les structures mixtes ('réactivo-cognitives'). En effet, il existe une vaste gamme d'agents cognitifs et réactifs, mais surtout il est tout à fait possible de réaliser des agents réactifs à partir de structures initialement destinées à être des agents cognitifs et vice-versa.

Parmi l'ensemble des structures multiagent possibles, seul un petit nombre d'entre-elles a conduit à des implémentations suffisamment pertinentes pour pouvoir être catégorisées. Il s'agit des structures à base : de modules horizontaux, (ou structures modulaires), de tableaux noirs, de subsomptions, de tâches compétitives, de règles de production, de classificateurs, de systèmes dynamiques de type multiagent et de structures connexionnistes. Notons que la majorité de ces types de structures emploient une approche fonctionnelle. Pour la conception des agents de l'AE, nous nous sommes inspirés de trois types de structures d'agents que nous allons étudier : la structure modulaire horizontale, celle à base de tableau noir et celle de production.

6.2.3.1 Structure modulaire horizontale

Ce type de structure est certainement l'une des plus répandues, qu'il s'agit de travaux théoriques ou d'applications pratiques (Ferguson 1992). La plupart des structures proposées pour la définition d'agents cognitifs sont fondées sur la notion d'ensembles de modules horizontaux. Les structures modulaires horizontales, que nous appellerons

modulaires (par souci de simplification du texte) sont conçues comme un assemblage de modules.

Chacun des modules réalise une fonction horizontale particulière. Les fonctions les plus courantes sont :

a. les fonctions perceptives et motrices s'il y a lieu;
b. l'émission et l'interprétation des communications;
c. la génération de la base des croyances de l'agent comprenant la représentation de l'environnement et celle des autres agents;
d. la gestion des engagements;
e. les expertises du domaine de compétence;
f. la gestion des buts et de la prise de décision;
g. la planification des actions.

Dans ce type de structure toutes les liaisons sont fixes : c'est-à-dire que le mode de circulation des informations est prédéfini par le concepteur.

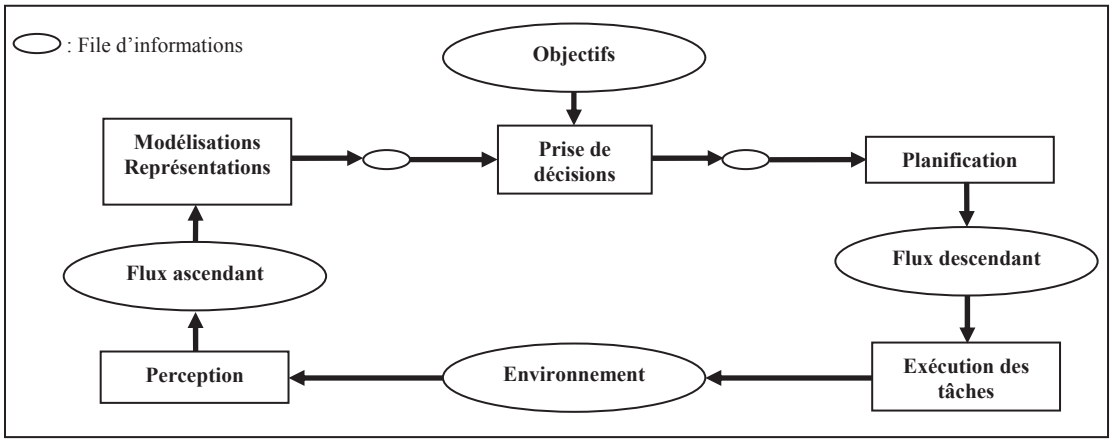

Figure 60 Architecture modulaire horizontale

La Figure 60 montre un exemple typique d'une telle structure, caractérisée par un flux d'information ascendant puis descendant. Dans la phase ascendante, les signaux provenant de l'extérieur issus de capteurs, ou de boîtes aux lettres contenant des messages, sont filtrés de manière à obtenir une information dont la nature est de plus en plus abstraite. Ceci se produit jusqu'à ce qu'elle puisse s'intégrer aux modélisations de l'agent. La fonction la plus élevée est effectuée par le module de prise de décision qui agit à partir des informations qu'il reçoit et des objectifs qui lui sont propres. La phase descendante correspond, quant à elle, à la mise en application des décisions. Le module de planification ordonne les actions à effectuer pour satisfaire l'objectif choisi. Celles-ci sont ensuite transmises au module d'exécution.

6.2.3.2 Les structures à base de tableaux noirs

La structure à base de tableau noir (Nii 1989) est l'une des plus utilisées dans les systèmes multiagent cognitifs symboliques. Elle a donné lieu à une abondante littérature. Originellement développée par le courant scientifique de l'IA traditionnelle pour la reconnaissance de la parole avec le système Hearsay II (Erman 1980), la structure à tableau noir s'est rapidement imposée en IAD comme une structure suffisamment souple et puissante pour pouvoir implémenter les mécanismes de raisonnement et de calculs intervenant à l'intérieur des agents, notamment avec le système DVMT (Lesser 1988). Le modèle du tableau noir est fondé sur un découpage en modules indépendants qui ne communiquent aucune information directement, mais qui interagissent indirectement en partageant des informations.
Ces modules, appelés sources de connaissance (‘*Knowledge Sources’*), travaillent sur un espace qui comprend tous les éléments nécessaires à la résolution d'un problème. La structure d'un système à base de tableau noir comprend trois sous-systèmes (voir Figure 61) :

a. les sources de connaissance (SC);

b. la base partagée (le tableau proprement dit) qui comprend les états partiels d'un problème en cours de résolution, les hypothèses et les résultats intermédiaires et, d'une manière générale, toutes les informations que s'échangent les SC : ces bases sont décomposées en hiérarchies (conceptuelles, partitives, causales etc.) qui structurent la modélisation du domaine d'application comme l'espace des hypothèses/solutions;
c. un dispositif de contrôle qui gère les conflits d'accès entre les SC : ces conflits interviennent de manière "opportuniste" c'est-à-dire sans être déclenchés effectivement par un système centralisé de contrôle.

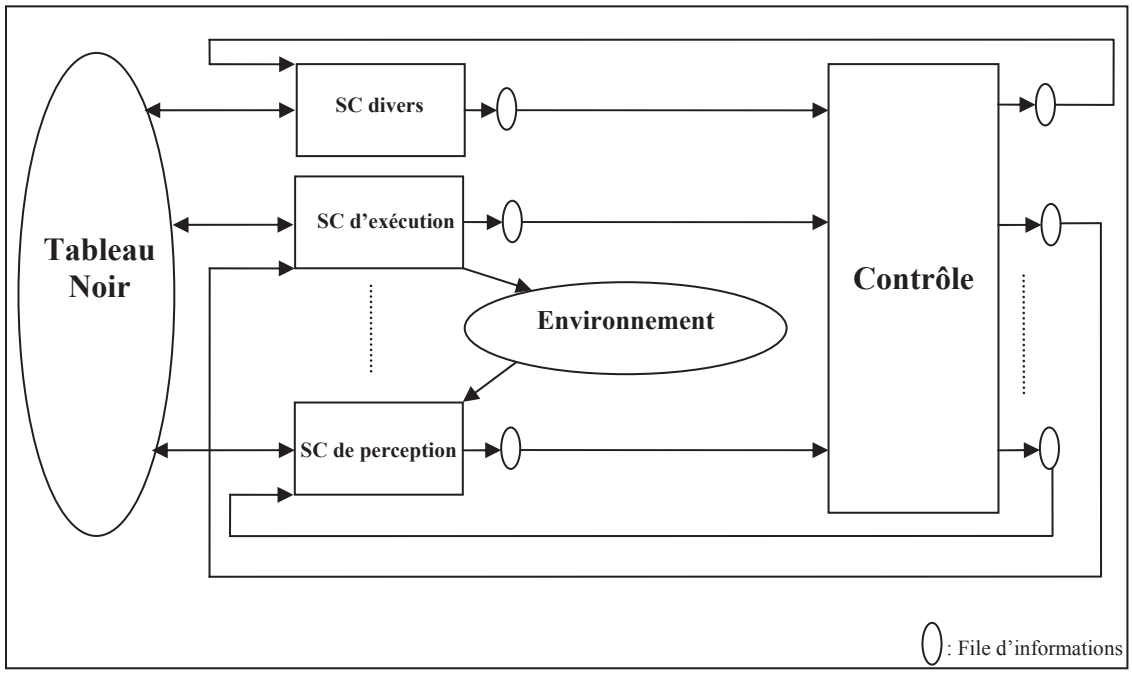

Figure 61 Une structure de SMA à base de tableau noir (SC : source de connaissances)

Notons que c'est la partie du dispositif de contrôle qui a connu le plus de modifications au cours de l'évolution de ces structures.

6.2.3.3 Les systèmes de production

Les systèmes de production font certainement partie des structures les plus connues de l'intelligence artificielle (Muller 1996).

Un système de production est défini par la combinaison d'une base de faits, d'une base de règles de production et d'un interprète appelé moteur d'inférence (Figure 62). Malgré la grande variété de syntaxes pour la définition de règles de production, ces dernières sont généralement données sous la forme suivante :

si <liste-conditions> alors <liste-actions> (6.1)

où <liste-conditions> est associée à des éléments de la base de faits, et <liste-actions> comprend des actions élémentaires telles qu'ajouter ou supprimer des éléments de la base de faits. Certaines actions peuvent aussi directement activer les commandes d'exécution de l'agent. Lorsqu'une règle peut valider chacune des conditions de sa liste, elle exécute les actions correspondantes.

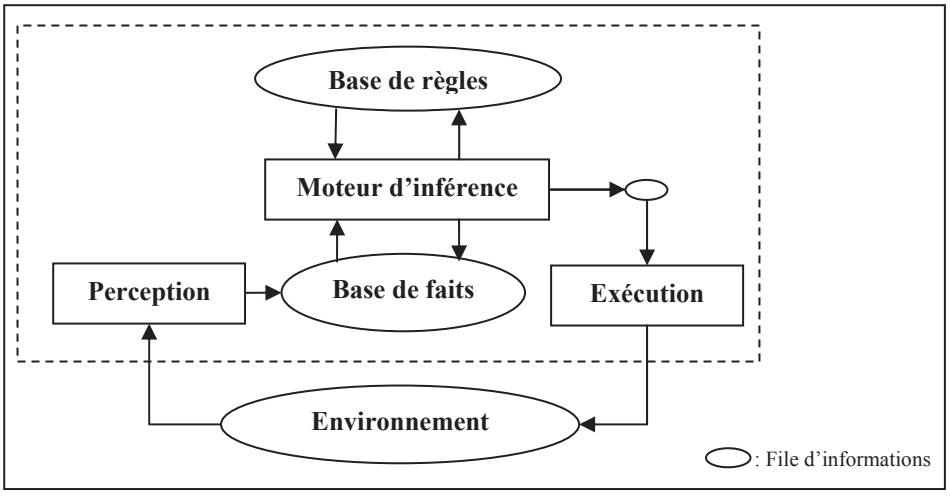

Figure 62 Agent à base de systèmes de production

Si plusieurs règles peuvent être activées simultanément, elles sont dites en conflit. Alors le système de contrôle du moteur d'inférence déclenche la règle qu'il considère comme la plus prioritaire (en fonction de paramètres internes à la règle, ou tout simplement en prenant la première).

Dans le cadre d'un système multiagent, chaque agent est représenté sous la forme d'un système de production, muni des fonctions de perception et d'exécution. La fonction de perception se charge de placer les informations perçues ou les messages à l'intérieur de la base de faits pendant le fonctionnement du moteur d'inférence, pour que la base de règles puisse les prendre en compte directement.

6.2.3.4 Structures des agents de l'AE

6.2.3.4.1 Agent Gestionnaire

Les agents Gestionnaires sont situés dans la couche de décision de l'AE. Ils sont dotés d'une structure dite de traitement (voir Figure 63).

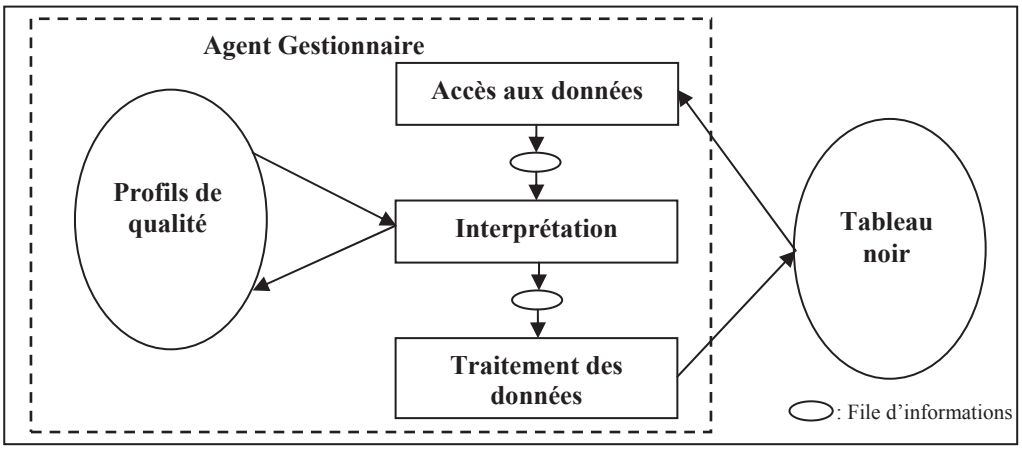

Figure 63 Structure de l'Agent Gestionnaire

Un agent Gestionnaire dispose de capacité de raisonnement et communique avec les autres agents de la couche de décision, généralement via le tableau noir et parfois par voie directe. L'agent a pour fonction l'accès aux données (générés par mesures ou collectes d'informations) qui correspondent à des sous caractéristiques de propriétés d'attributs de qualités. De l'interprétation de ces informations seront mis en branle le ou les scénarios de reconfigurations le ou les plus adéquats.

L'Agent Gestionnaire reformule les informations extraites en données cohérentes et compatibles avec le système de traitement des agents Décideurs. Ces données sont stockées dans le tableau noir qui constitue une base de données commune à tous les agents Décideurs.

6.2.3.4.2 Agents Décideurs

Les Agents Décideurs sont dotés d'une structure de production adaptée à nos besoins (voir la Figure 64).

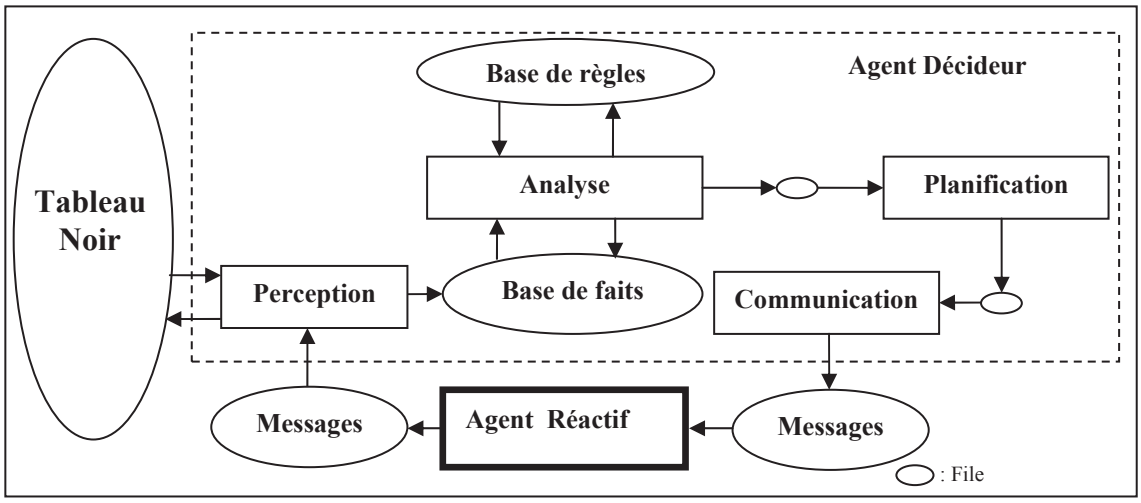

Figure 64 Architecture de l'agent supérieur

Deux aspects majeurs apparaissent dans sa structure par rapport au cas classique (Figure 61) :

a. l'agent ne perçoit pas l'environnement extérieur, mais échange des informations avec le tableau noir et communique avec ses Agents Réactifs pour percevoir ou agir sur l'environnement;
b. l'agent n'exécute pas d'action mais engendre une planification des tâches que d'autres agents vont exécuter.

Ces fonctionnalités permettent à l'Agent Décideur de pouvoir mettre à jour ses règles et donc, d'acquérir de nouvelles solutions de reconfiguration via le tableau noir.
Autre avantage majeur de la structure, elle permet à l'Agent Décideur de consulter l'état des autres Agents Décideurs (toujours via le tableau noir), mais aussi de diminuer l'effort de conception de sa structure de production puisque le module d'exécution n'existe plus : l'Agent Décideur distribue les plans mais ne les exécute pas.

6.2.3.4.3 Agents de la couche réactive

L'Agent Réactif possède une structure dépouillée (Figure 65). Son rôle est d'activer les mécanismes dont il est doté. Nous admettrons que les mécanismes sont de deux types : action et perception. Ces mécanismes sont activés par de simples stimuli via un commutateur de type marche/arrêt. Chaque action ou perception est validée par l'agent Décideur.

6.2.3.5 Approche 'voyelle' pour la description des agents

Dans cette section, nous décrivons les agents de l'AE à travers leurs propriétés de raisonnement et d'interaction. Nous analysons dans ce qui suit les propriétés de chaque

agent grâce à l'approche 'voyelle' que nous avons présentée au paragraphe 3.4 de ce document.

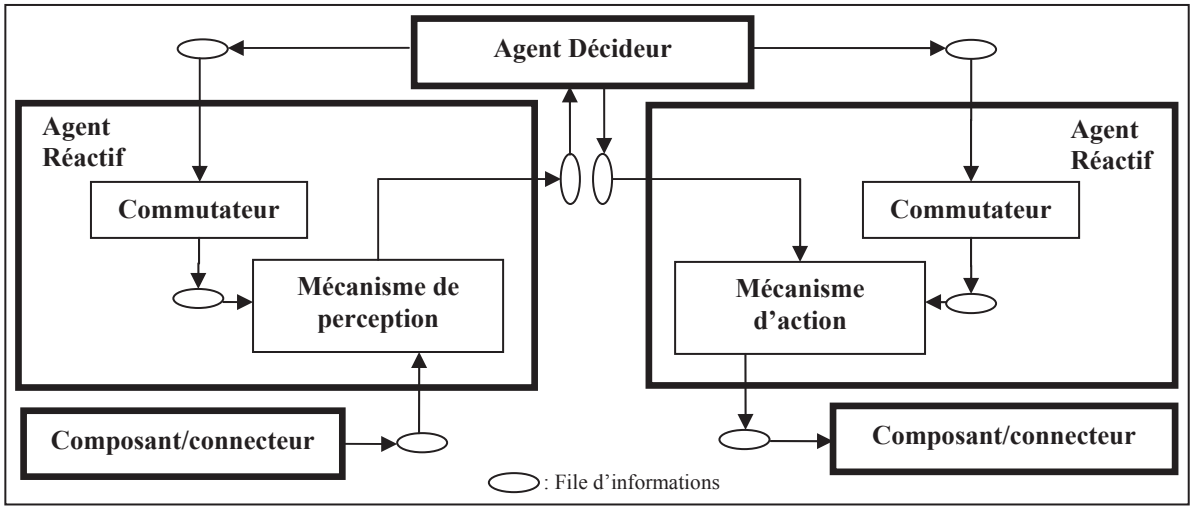

Figure 65 Architecture de l'Agent Réactif

6.2.3.5.1 Agent Gestionnaire

L'Agent Gestionnaire est un agent dit de traitement, sa description selon l'approche voyelle est donnée à la Figure 66.

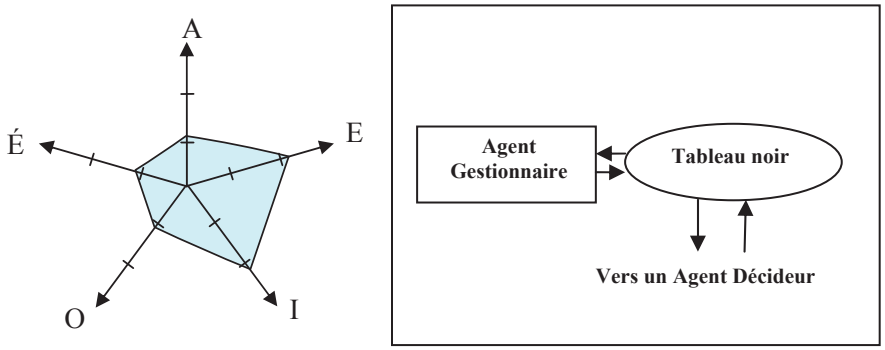

Figure 66 Description de l'Agent Gestionnaire selon l'approche 'voyelle'

Il analyse les informations du tableau noir via un protocole défini et envoie les résultats vers les agents de la couche décisionnelle, toujours via le tableau noir. Il est fortement caractérisé par ses dimensions (E) et (I).

6.2.3.5.2 Les Agents Décideurs

Les agents de la couche de décision sont fortement caractérisés par leurs fonctionnalités de raisonnement (dimension A). Les autres caractéristiques fortes de ces agents sont leurs capacités de mobiliser les agents de la couche réactive. Ceci explique des dimensions (O) et (I) prononcées sur la Figure 67.

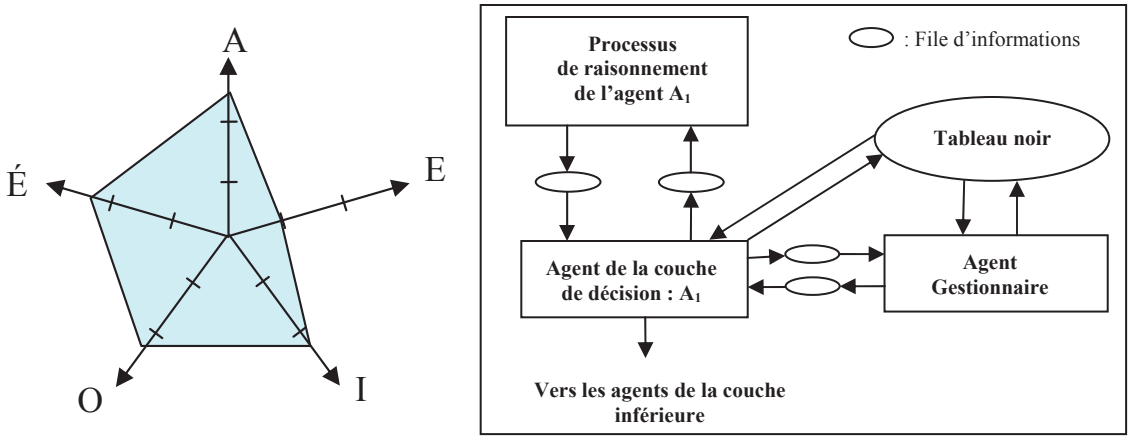

Figure 67 Description d'un agent A_1 de la couche de décision selon l'approche 'voyelle'

Sur la Figure 67 et les figures suivantes, les petites ellipses représentent des files d'informations.

6.2.3.5.3 Les Agents Réactifs

Les agents de la couche réactive sont utilisés pour l'action ou la perception. Ils répondent à de simples stimuli. Ces agents sont en contact avec l'architecture d'où une forte dimension (E) comme le montre la Figure 68.

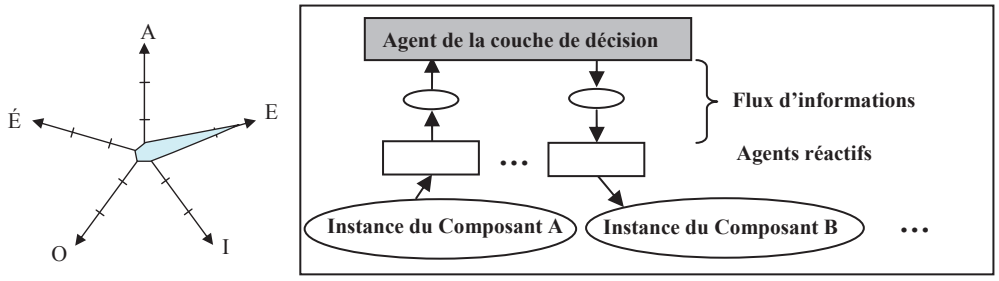

Figure 68 Description des agents de la couche réactive selon l'approche 'voyelle'

6.3 Organisation du SMA

L'organisation est l'un des concepts de base des systèmes multiagent. L'organisation suppose qu'il existe un ensemble d'entités formant une entité plus complexe et dont les différents éléments sont subordonnés entre eux dans un ensemble solidaire et dans une ou plusieurs activité convergente.

Définition traduite de (Bond 1998) :

« Une organisation peut être définie comme un agencement de relation entre composants ou individus qui produit une unité, ou système, doté de qualités inconnues au niveau des composants ou individus. L'organisation lie selon une interrelation des éléments ou événements, ou individus divers qui dès lors deviennent les composants d'un tout. Elle assure solidarité et solidité relative et donc garantit au système une certaine possibilité de durée en dépit de perturbations aléatoires ».

Dans un SMA il existe de nombreuses interrelations entre les agents, par le biais de délégations de tâches, de transferts d'informations, d'engagements et de synchronisations d'actions (Bond 1998). Ces interrelations ne sont possibles qu'à l'intérieur d'une organisation. Mais, réciproquement, les organisations exigent l'existence de ces interrelations. Les organisations constituent donc à la fois le support et la manière dont se produisent ces interrelations. C'est-à-dire, la façon dont sont réparties les tâches, les informations, les ressources et la coordination des actions.
Le paragraphe suivant donne une analyse complète de l'organisation de l'AE, en passant par sa structure, ses modes de communications et sa stratégie de coordination de ses actions.

6.3.1 Étude de l'organisation de l'AE

6.3.1.1 Introduction

L'organisation du SMA que constitue notre AE est basée sur l'analyse des scénarios de reconfiguration, puis leurs exécutions et leurs gestions. Nous distinguerons donc trois étapes importantes dans l'organisation de l'AE :

a. l'initialisation des couches : implémentation des scénarios de reconfiguration dans l'AE;
b. le test de l'architecture logicielle : évaluation des qualités de l'architecture basée sur les paramètres des scénarios;
c. l'activation des scénarios de reconfiguration : phase d'application des solutions architecturales basées sur les scénarios de reconfiguration.

La Figure 69 décrit les trois volets de l'organisation sous forme d'un réseau de Petri.
L'étude de l'organisation du système multiagent de l'AE sera focalisée sur les aspects que nous jugeons les plus importants et que nous classons sous trois volets :

a. l'analyse fonctionnelle : Elle décrit les fonctions de l'organisation multiagent dans ses différentes dimensions;
b. l'analyse structurale : qui distingue les différentes formes d'organisation possibles et identifie quelques paramètres structuraux essentiels de l'AE;
c. les paramètres de concrétisation : qui traite du passage d'une structure organisationnelle à une organisation concrète et pose la difficile question de la réalisation effective d'une organisation multiagent.

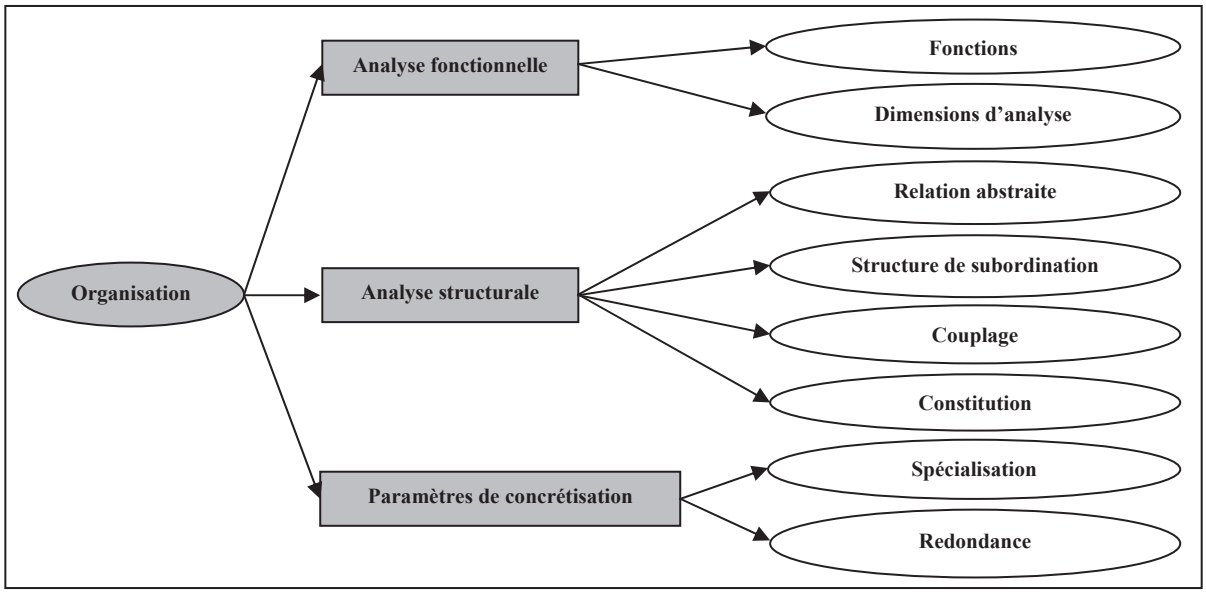

Figure 69 Étude d'une organisation

6.3.1.2 Analyse fonctionnelle

L'analyse fonctionnelle d'une organisation conduit à l'identification des principales fonctions que les composants de notre organisation doivent remplir (Ferber 1995). Dans le cadre d'une telle analyse, notre organisation peut être vue comme un système de rôles. Chaque rôle est défini par l'ensemble des propriétés que les agents (possédant ce rôle) manifestent dans cette organisation.

Les rôles décrivent la position des agents au sein d'une organisation ainsi que l'ensemble des activités qu'ils sont censés exercer afin que l'organisation puisse accomplir ses objectifs. Ils caractérisent ainsi les fonctions que les agents remplissent, indépendamment de leur structure interne et des mécanismes mis en oeuvre. Par exemple, des agents de la couche de décision gèrent les demandes d'exécution et les répartissent aux agents réactifs compétents. D'autres agents de la couche supérieure font des choix parmi un ensemble d'actions possibles. Ils ont le rôle de décideur. Ces rôles sont définis par l'ensemble des fonctions nécessaires à notre organisation pour qu'elle s'adapte à ses finalités (et en particulier à nos objectifs et à nos besoins) et à son environnement. Nous donnons dans les trois paragraphes qui suivent les fonctions principales d'une organisation et quelques uns des rôles associés.

6.3.1.2.1 La fonction représentationnelle

La fonction représentationnelle comprend l'ensemble des fonctions de modélisation de l'environnement et des autres organisations ainsi que la mémorisation des événements qui ont pu affecter l'organisation. Cette fonction est particulièrement développée dans les organisations individuelles cognitives (c'est-à-dire les agents cognitifs) qui disposent d'une représentation interne de leur environnement qui leurs est possible de manipuler afin de planifier les actions qu'elles devront entreprendre. Dans une organisation collective, la fonction représentationnelle peut être, comme pour les autres fonctions, distribuée parmi tout un ensemble d'agents. (À ce propos nous renvoyons le lecteur aux notions de distribution fonctionnelles et spatiales présentées au paragraphe 3.4.) Les agents de la couche supérieure disposent d'une représentation restreinte de l'architecture qui ne dépasse pas le cadre de leurs scénarios de reconfiguration. Cette représentation restreinte est partagée dans chaque squelette d'agents et concerne une zone de reconfiguration sur l'AL multimodale (Figure 70). Elle est obtenue grâce à la description du scénario des points sensibles, à risque et ceux de l'évaluation de l'architecture. Un autre type de description concerne la représentation des agents entre eux. Du fait de la

hiérarchisation du SMA, certains des agents de l'AE disposent de manière directe ou indirecte d'une représentation partielle ou totale des autres agents avec qui ils interfèrent. Cette fonctionnalité leurs permet de communiquer entre eux. Cette fonctionnalité est plus particulièrement attribuée aux Agents Décideurs qui disposent d'une représentation entre eux via le tableau noir.

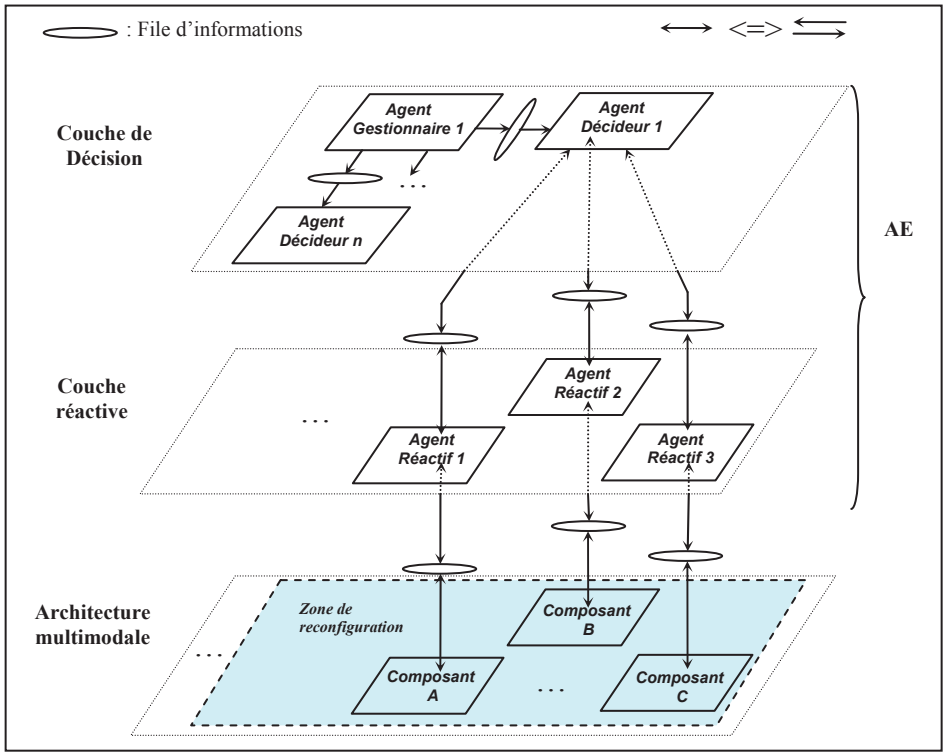

Figure 70 Représentation symbolique de l'architecture par zone de reconfiguration

De même chaque Agent Décideur possède une représentation directe de ses Agents Réactifs.

6.3.1.2.2 La fonction organisationnelle

La fonction organisationnelle se rapporte à tout ce qui a trait à la gestion de l'activité d'une organisation et, en particulier, à la planification, l'allocation et le suivi de tâches, la coordination des actions ainsi que la gestion des engagements.

Cette fonction est évidemment essentielle et c'est celle qui dans le cas des SMA a été le plus particulièrement étudiée (voir paragraphe 2.3.2.3) . De nombreux rôles sont associés à cette fonction : en particulier ceux du client, fournisseur, médiateur, planificateur et coordinateur. La fonction organisationnelle de chaque agent dépend de la phase dans laquelle se trouve l'AE :

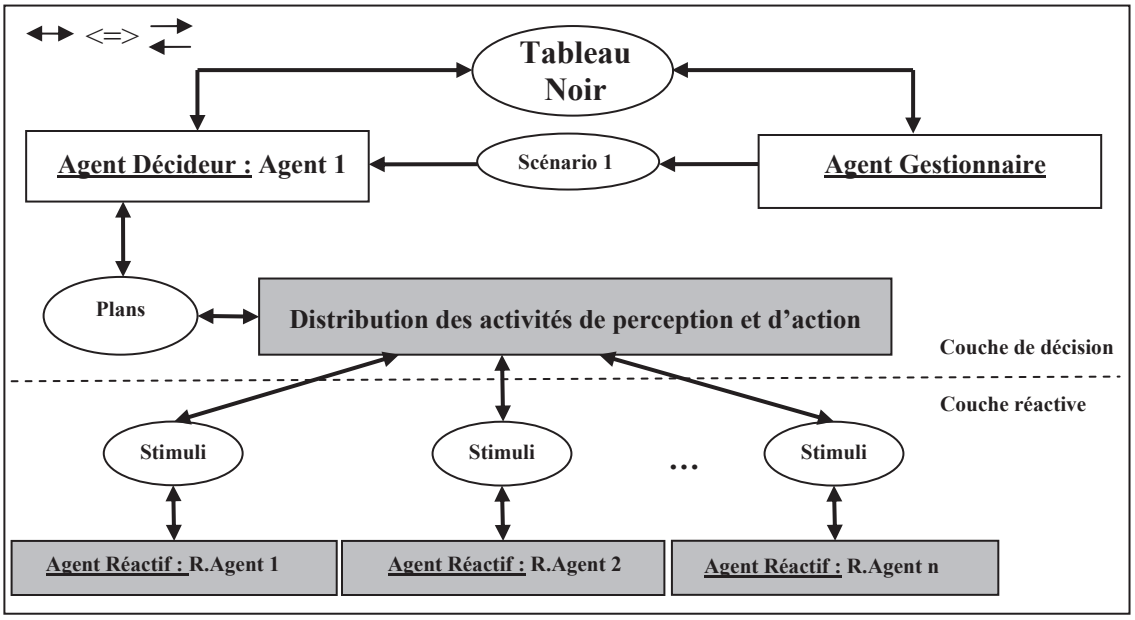

Figure 71 Fonction organisationnelle dans la phase d'initialisation

Initialisation des couches (Figure 71) : dans cette phase, la distribution des scénarios est effectuée par l'Agent Gestionnaire. Chaque Agent Décideur reçoit un scénario et en est responsable. Les agents de la couche de décision utilisent le tableau noir comme base de connaissances commune. Elle inclut des informations importantes comme l'identifiant

d'agent, l'identifiant de scénario, l'identifiant de l'attribut, l'identifiant du profil, le poids du scénario, etc. Les Agents Décideurs extraient ensuite les données fournies par leurs scénarios et réalise les plans d'évaluation et de reconfiguration de l'architecture, à partir de leurs bases de faits et de leurs bases de règles. Ces plans seront transférés aux agents réactifs.

Évaluation de l'architecture : Lorsqu'un Agent Décideur reçoit les informations qui correspondent aux directives d'un scénario, ce dernier engage immédiatement le processus de dialogue avec l'agent Gestionnaire, via le tableau noir pour l'avertir qu'il évalue l'architecture en activant les Agents Réactifs de perception.

Activation des scénarios de reconfiguration (Figure 72) : les données provenant de la perception de l'architecture permettent de juger de son niveau de qualité, sur la base des critères fournis par le scénario. C'est l'Agent Décideur qui engage les processus de reconfiguration, avec l'autorisation de l'agent gestionnaire. L'application du scénario commence par l'envoi de directives pour stimuler les Agents Réactifs d'action.

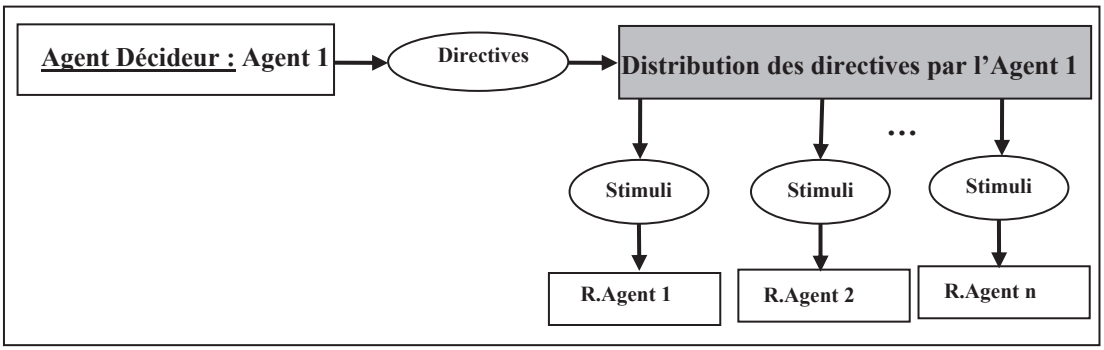

Figure 72 Fonction organisationnelle dans la phase d'activation

6.3.1.2.3 La fonction interactionnelle

La fonction interactionnelle sert à faire le lien entre une organisation et ce qui l'entoure. C'est elle qui gère les activités d'interactions avec l'environnement ainsi que l'ensemble des communications avec les autres organisations. Elle assure ainsi toutes les fonctions d'entrées-sorties et d'interface. Elle se compose de deux sous fonctions : a) la fonction perceptive qui se charge de l'acquisition d'informations provenant de l'extérieur; b) et la fonction exécutrice qui s'occupe de l'accomplissement des actions choisies par le système conatif et mises en oeuvre par le système organisationnel.
L'AE dispose de fonctions interactionnelles lui offrant la capacité d'interagir avec l'architecture via une évaluation ou une reconfiguration. Enfin, citons les espaces d'interaction interne permettant aux agents de l'AE de communiquer entre eux pour l'échange d'information ou pour l'organisation des actions. Nous distinguons trois espaces d'interaction et de communication :

a. Espace d'interaction α : L'espace d'interaction dans la couche supérieure de décision comprend les interactions externe (pour l'initialisation d'informations concernant la reconfiguration) et interne (entre les agents de la couche décisionnelle via le tableau noir) à l'AE.
b. Espace d'interaction β: situé entre la couche réactive et la couche de décision, il fait le lien entre chaque Agent de Décision et un à plusieurs Agents Réactifs.
c. Espace d'interaction γ : cette espace est réservée à l'interaction des Agents Réactifs avec les éléments de l'architecture multimodale via des connecteurs par exemple.

6.3.1.3 Analyse structurale

L'organisation multiagent apparaît comme un système d'une grande complexité. L'analyse structurale tente de donner un ordre à l'ensemble des interactions possibles

entre les agents en dégageant les relations abstraites qui les relient et la manière dont elles évoluent au cours du temps.

6.3.1.3.1 Agents et tâches des agents

La décomposition d'un problème en termes multiagent, aborde la délicate mais pertinente question de savoir ce que représente véritablement un agent en relation avec le problème à résoudre. Il s'agit plus exactement de définir le rapport entre la fonction productive individuelle de l'agent et la fonction productive de l'organisation à laquelle il appartient. En effet, la définition d'une organisation multiagent doit mettre en évidence :

a. l'identification de ce qu'est un agent pour un problème donné; l'analyse s'effectuant à partir d'une approche fonctionnelle (centrée sur la fonction) ou objet (centrée sur l'individu et le produit);
b. les tâches à accomplir et leurs répartitions entre les agents;
c. la redondance des agents par rapport aux tâches.

Un agent se caractérise bien évidemment par son architecture et son comportement. Mais pour le concepteur d'un SMA, l'agent est surtout caractérisé par ce qu'il peut faire, c'est-à-dire par sa fonction productive et donc, par son apport dans une organisation multiagent. Il se définit aussi par ce qu'il n'est pas autorisé à faire. Il s'agit alors d'identifier ce que nous considérons comme 'agent' par rapport à un problème considéré. Ceci doit être fait à partir d'un point de vue à la fois conceptuel et méthodologique qui ordonne toutes les questions futures. La tâche de chaque agent dans l'AE est bien définie mais peut éventuellement changer au cours du temps ou de l'évolution de l'environnement. Cela se produit si le scénario est mis à jour.

L'Agent Gestionnaire : Nous avons assigné à cet agent la tâche d'extraction des données de scénarios. C'est une tâche non évolutive, elle consiste à répertorier les scénarios et de classifier les informations par thème selon les étapes suivantes (Figure 73):

a. le profil de qualité : Il consiste à associer à chaque qualité un ensemble de scénarios et à chaque scénario un Agent Décideur;
b. identification du scénario : l'Agent Gestionnaire utilise cette donnée afin d'instancier les agents de l'AE : pour chaque scénario de reconfiguration un Agent Décideur est instancié ainsi qu'un certains nombre d'Agents Réactifs;
c. identification des règles : l'Agent Gestionnaire sépare les différents mécanismes d'évaluation et de reconfiguration et identifie les règles des scénarios car ils permettrons aux Agents Décideurs d'établir les plans et les directives.

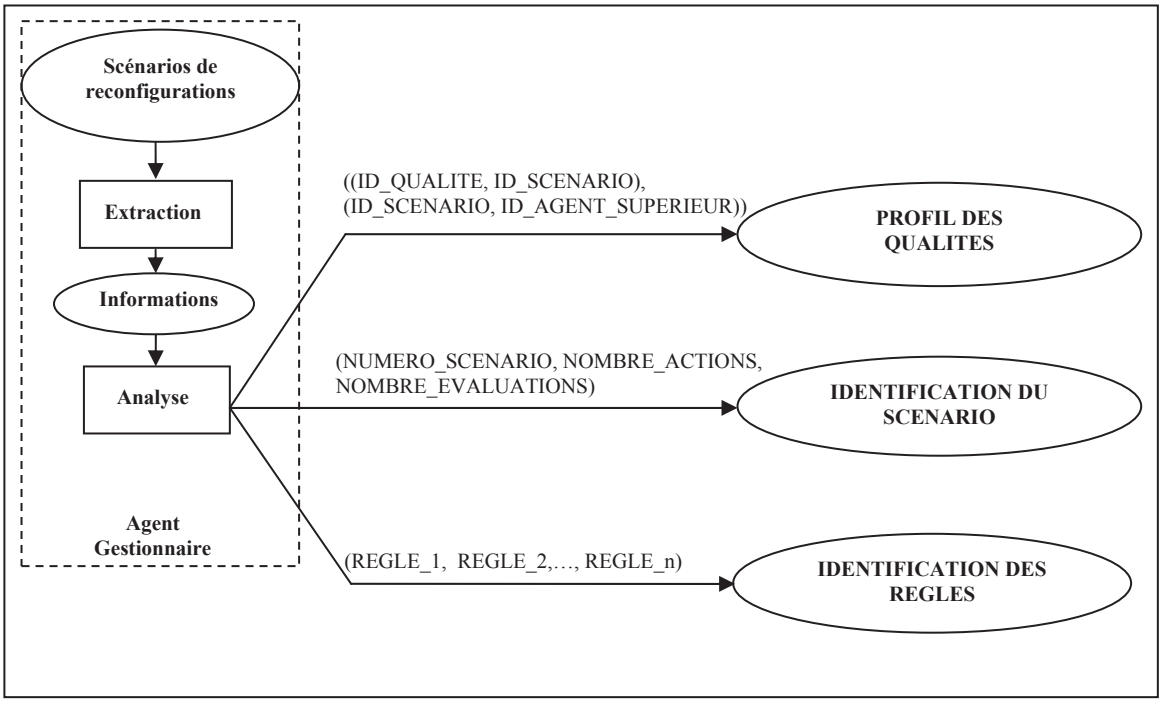

Figure 73 Tâches d'extraction des données de l'Agent Gestionnaire

Les informations récoltées par l'Agent Gestionnaire sont envoyées vers les Agents Décideurs de façon indirecte via le tableau noir.

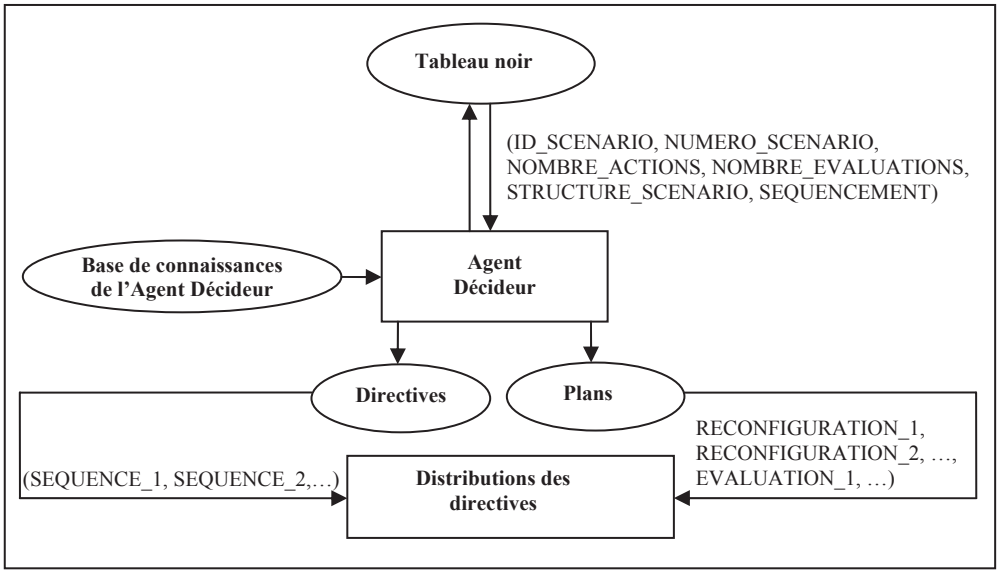

Figure 74 Tâche de l'agent Décideur

L'Agent Décideur : Chaque scénario est dédié à un seul Agent Décideur, celui-ci peut disposer de deux sources d'informations : le tableau noir ainsi que sa propre base de connaissances (qui correspond à l'ensemble des deux bases : la base de faits et la base de règles à la Figure 64). En tenant compte de ces informations, cet agent peut envoyer un plan de reconfiguration sous forme de directives assignées aux agents Réactifs (Figure 74). Le plan comprend les processus d'évaluation et de reconfiguration de l'architecture. L'initialisation de l'Agent comprend l'envoi d'un plan à un processus de 'Distribution des directives' afin de préparer l'exécution du scénario de reconfiguration. Si les conditions sur la priorité de l'exécution du scénario sont validées, alors les séquences de reconfigurations sont accomplies par les Agents Réactifs.

Les Agent réactifs : Lorsque les agents réactifs sont dédiée à la perception, leurs rôles consistent à envoyer des valeurs métriques de leurs mesures par de simple message. Cependant, si ces valeurs exigent l'intervention de l'AE, l'Agent de Décision applique les directives du scénario en envoyant des stimuli vers les Agents Réactifs correspondant (Figure 75) tout en respectant le séquencement généré par l'Agent Décideur.

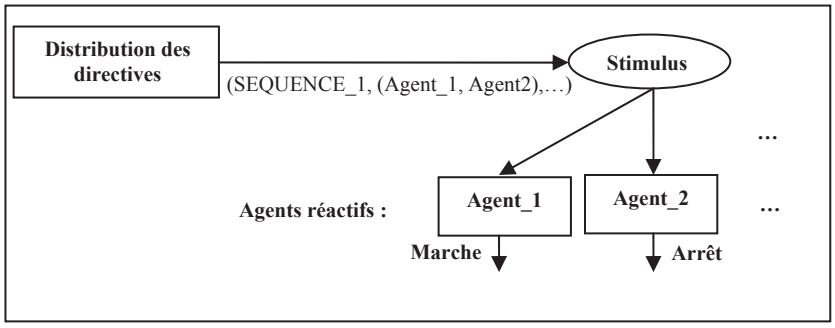

Figure 75 Tâche de distribution des directives

6.3.1.3.2 Relations abstraites

Les relations organisationnelles décrivent au niveau abstrait les formes d'interrelations qui peuvent avoir lieu entre les différente catégories fonctionnelles d'agents : c'est-à-dire entre les différents rôles que peuvent jouer ces agents.

Nous distinguons deux types de relations organisationnelles : a) les relations statiques, qui forment le squelette d'une organisation, caractérisent la structure organisationnelle préexistante à tout fonctionnement et sont définies en dehors de toute exécution; b) et les relations dynamiques, qui constituent la 'substance' des organisations et décrivent les relations qui sont modifiées lors des interactions. Cette substance sera modélisée symboliquement par des jetons typés qui se déplacent en changeant de 'couleur' dans des RPCTS. L'AE dispose d'une base de connaissances évolutive, mais les relations entre les agents restent fixes. Nous pouvons y distinguer trois types de relation (Figure 76) :

a. Une relation d'accointance qui se produit entre deux agents Décideurs de la couche décisionnelle et qui indique que l'un d'entre eux *connaît* l'autre, qu'il en a une représentation et qu'il peut lui adresser indirectement ses messages. Il s'agit de la relation minimale entre deux agents cognitifs de la couche supérieure de l'AE. Elle sert généralement de support à toutes les autres relations.
b. L'existence d'une relation communicationnelle, ou plus exactement d'un canal de communication, entre un Agent Gestionnaire et un Agent Décideur. Cette dernière indique que l'Agent Gestionnaire peut envoyer et recevoir des messages à l'Agent décideur.

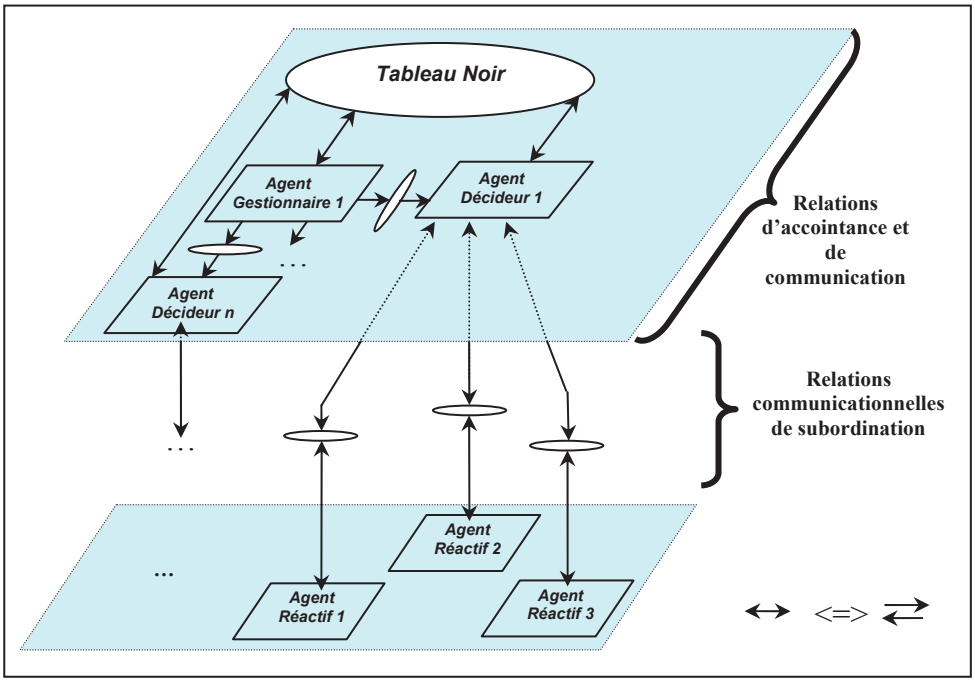

Figure 76 Types de relations abstraites dans l'AE

c. Les relations de subordination décrivent un transfert d'exécution entre un Agent de Décideur demandeur et un agent réactif exécutant. Cette relation est statique, cela

signifie que l'Agent Réactif ne peut pas refuser la demande de l'Agent Décideur et donc que ces deux entités sont dans une relation maître/esclave.

6.3.1.3.3 Constitution des structures organisationnelles

Les structures organisationnelles peuvent être constituées de deux manières. Elles peuvent être définies, a priori, par le concepteur et il s'agit alors d'organisations prédéfinies : ce qui signifie que les relations abstraites, qu'elles soient statiques ou dynamiques, sont déterminées à l'avance. Ou bien, elles peuvent être constituées d'organisations évolutives, décrivant l'ensemble des relations abstraites possibles ainsi que l'ensemble des transformations connues à l'avance.
L'AE évolue avec l'architecture, mais la structure et le rôle de ses agents restent les mêmes. Néanmoins nous pouvons parler d'émergence dans ce type d'organisation prédéfinie variable ou évolutive. Cette émergence est toujours contrôlée par des schémas de communication bien établis et les organisations qui en résultent sont toujours décrites par des structures organisationnelles très précises. L'évolution de l'AE est initiée par de nouvelles combinaisons de scénarios de reconfiguration selon différents profils, puis par l'Agent Gestionnaire (car après analyse de chaque scénario de reconfiguration un nombre précis d'agents est instancié). La règle d'instanciation des agents et plus précisément celle du squelette d'agents est réalisée dans l'ordre suivant :

a. un Agent Décideur est instancié pour chaque scénario de reconfiguration;
b. des Agents Réactifs correspondant aux nombres d'éléments de l'architecture évalués (points sensibles et à risques) et d'éléments reconfigurables sont instanciés.

6.3.2 Coordination des actions dans l'AE

6.3.2.1 Introduction

Lorsque plusieurs agents travaillent ensemble, il faut gérer un certain nombre de tâches supplémentaires qui ne sont pas directement productives mais servent à améliorer l'accomplissement des activités qui elles le sont. Ces tâches supplémentaires font partie de la structure organisationnelle et sont appellées tâches de coordination. Celles-ci sont indispensables dès que le système se trouve en présence d'un ensemble d'agents autonomes qui poursuivent leurs propres buts ou un but commun. La réalisation de tâches productives entraînent avec elle tout un cortège de tâches de coordination sans lesquelles les premières ne peuvent être accomplies. La coordination d'actions est nécessaire pour quatre raisons principales que nous citons ci-après.

a. Les agents ont besoin d'informations et de résultats que seuls d'autres agents peuvent fournir.
b. Les ressources sont limitées. L'attention prêtée aux actions des autres n'est parfois due qu'aux faibles ressources dont nous disposons et au fait que plusieurs entités utilisent ces ressources réduites. Qu'il s'agisse de temps, d'espace, d'énergie, d'argent ou d'outils, les actions, pour être accomplies, ont besoin de ressources qui n'existent pas en quantité inépuisable. Et, la coordination est d'autant plus importante que les ressources sont faibles. Il faut donc partager ces ressources de manière à optimiser les actions à effectuer (éliminer les actions inutiles, améliorer le temps de réponse, diminuer les coûts, etc.) tout en essayant d'éviter les conflits éventuels (conflits d'accès, actions contradictoires, etc.)
c. Les coûts doivent être optimisés. Coordonner des actions permet aussi de diminuer les coûts en éliminant les actions inutiles et en évitant les redondances d'action.
d. Il faut tenter de tirer avantage de l'interdépendance des agents. Nous voulons permettre à des agents ayant des objectifs distincts mais dépendants les uns des

autres de satisfaire ces objectifs et d'accomplir leur travail en tirant éventuellement parti de cette dépendance.

6.3.2.2 Relation entre les actions

Lorsque les agents accomplissent leurs actions, certaines actions sont particulièrement difficiles à gérer, parce que leurs exécutions simultanées entraînent des conflits. Au contraire, d'autres actions conduisent à une amélioration des performances. F. Von Martial (Von martial 1992) a classé en deux grandes catégories les relations pouvant exister entre les actions lorsqu'elles sont accomplies simultanément par plusieurs agents : les relations négatives et positives (Figure 77).

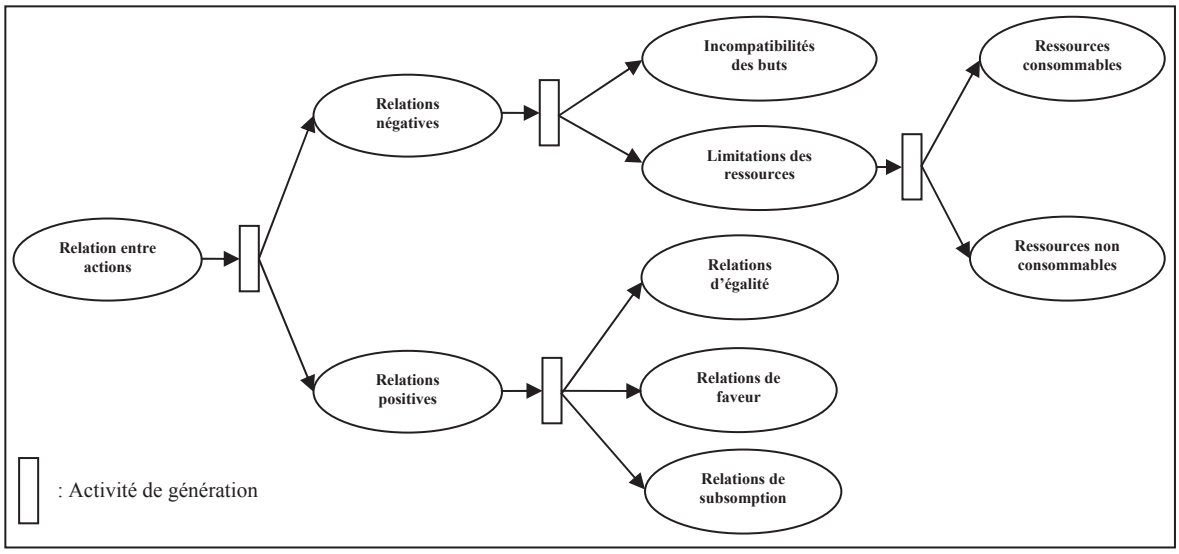

Figure 77 Les types de relations entre actions

Les relations négatives ou conflictuelles sont celles qui gênent ou empêchent plusieurs actions de s'accomplir simultanément. Cet aspect négatif peut évidemment être dû soit à l'incompatibilité des objectifs soit à la limitation des ressources. Au contraire, les relations positives, ou synergiques, sont celles qui favorisent les actions en les faisant

bénéficier les unes des autres et donc tendent à une plus grande efficacité (comme si les actions étaient réalisées de manière indépendante). La relation positive peut être de trois types : d'égalité, de faveur et de subsomption. La relation d'égalité signifie que certaines actions ne sont pas liées à un agent particulier et peuvent être réalisées par un autre agent. La relation de subsomption indique que l'action *a* d'un agent *A* fait partie des actions *b* d'un agent *B*, et donc qu'en accomplissant *b*, *a* se réalise également. La relation de faveur précise qu'une action, en s'accomplissant, favorise la possibilité d'accomplir une autre action.

Dans notre domaine d'application, les relations négatives entre les actions découlent des interférences entre les scénarios, nous parlerons alors d'incompatibilité entre les buts.

Détecter qu'une relation existe entre deux actions constitue un enjeu majeur de l'organisation d'un ensemble d'actions, mais aussi un problème crucial pour l'organisation et l'amélioration de la coordination des actions. Les agents supérieurs de l'AE disposent de mécanismes de détection des relations négatives, dans le but d'obtenir une action globale cohérente du SMA. Les Agents Décideurs ont la capacité d'éviter les conflits entre les actions grâce à des règles de comportement implantées dans le tableau noir. Quant aux attributs qui leurs permettent d'appliquer les règles, ils sont fournis par les scénarios. L'Agent Décideur effectue une activité de gestion des conflits dans le SMA selon des mécanismes prédéfinis. Cette activité est déclenchée lorsque l'Agent Décideur s'apprête à envoyer les directives de reconfiguration. Nous appelons cette action « consultation ». Comme son nom l'indique, l'Agent Décideur consulte la base espace de données commune de l'AE (tableau noir) afin de vérifier l'exécution des autres scénarios et les possibles interactions avec le sien. La liste des scénarios d'interférence est fournie par l'Agent Décideur lui-même. Si la liste des scénarios d'interférence ne correspond à aucun scénario en cours d'exécution, l'Agent Décideur exécute les reconfigurations et signale son activité à travers le tableau noir. Au contraire si des scénarios d'interférence existent et sont en cours d'exécution, l'agent compare le poids de son scénario avec les poids des autres scénarios d'interférence (c'est l'activité de comparaison).

Deux cas se présentent : a) si le poids du scénario de l'agent est plus grand que tous les autres, l'agent met à jour la base de connaissance en signalant son activité et exécute en premier ses reconfigurations (prioritairement sur les autres reconfigurations); b) dans le cas contraire, l'agent arrête et n'effectue pas ses reconfigurations même si ses propres critères sur la qualité ne sont pas respectés par l'application. L'emploi des RPCTS pour modéliser ces comportements prend encore une fois tout son sens ici. En effet, l'utilisation de cet outil permet de gérer de manière automatique les conflits ou les blocages que pourraient occasionner l'exécution des scénarios d'interférence.

6.3.2.3 Déroulement d'un scénario de reconfiguration

Afin de mieux percevoir la coordination des actions des agents dans l'AE, nous présenterons dans cette section le déroulement d'un scénario de reconfiguration à travers des modèles de réseaux de Petri à un haut niveau d'abstraction. Nous décomposons le déroulement du scénario en trois principales étapes détaillées aux paragraphes suivants.

6.3.2.3.1 Phase d'initialisation de l'AE

C'est lors de cette phase que l'AE construit sa représentation de l'architecture. Les scénarios fournissent à la fois des informations sur les éléments sensibles et à risque de l'architecture, ainsi que les critères choisis par les intervenants sur le niveau de ses qualités.

Dans le réseau de la Figure 78, nous décrivons les activités des agents lors de l'initialisation de l'AE. Cette phase consiste à employer des données afin d'en extraire les scénarios de reconfigurations et de préparer les objectifs de l'AE. Le flux de données est descendant, c'est-à-dire qu'il commence à la couche décisionnelle et finit au niveau de la couche réactive. Cette phase se termine par l'activation des Agents Réactifs responsables de la surveillance de l'architecture.

Afin de mieux décrire le rôle des agents, nous détaillons l'activité de chaque agent de l'AE. L'agent Gestionnaire (Figure 79) est responsable de l'importation des données depuis le tableau noir et de leurs distributions. L'analyse réalisée par l'Agent Gestionnaire lui permet de connaître la configuration structurelle de l'AE. L' « Analyse des scénarios » permet en effet d'instancier le squelette d'agents pour chaque scénario et permet de stocker les premières données dans la base de connaissances commune (tableau noir).

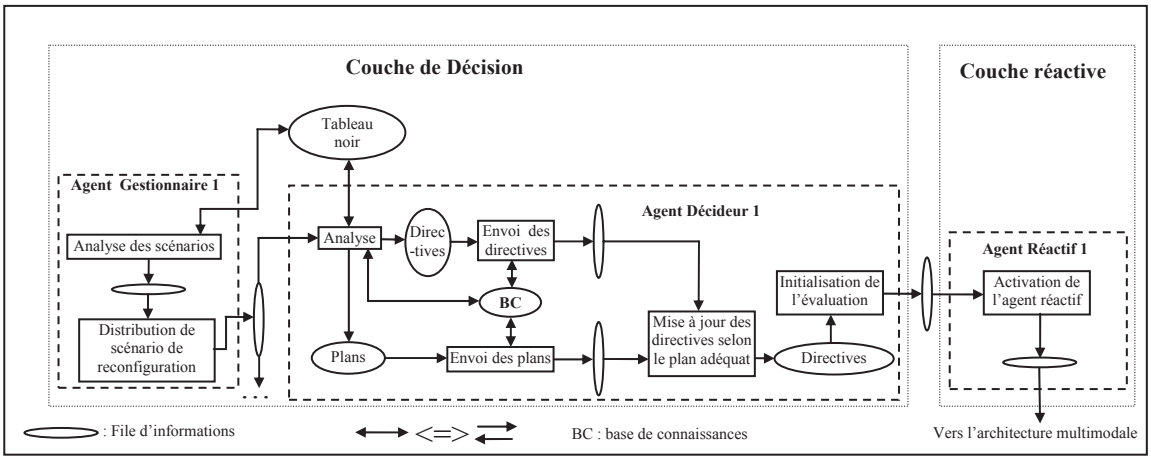

Figure 78 Réseau de Petri des agents de l'AE lors de la phase d'initialisation

La « Distribution de scénario de reconfiguration » est l'activité de l'agent responsable de la diffusion des scénarios aux Agents Décideurs. L'envoi des messages utilise un adressage (identification) pour cibler les agents.

Dès la réception du message d'initialisation de l'Agent Gestionnaire, l'Agent Décideur en extrait la structure afin d'initialiser tous les paramètres d'évaluation (les conditions sur le stimulus, les points sensibles…etc.), de reconfiguration (les directives de reconfiguration et les séquences de directives) et d'interaction (la liste des scénarios d'interférence, le poids de chacun d'eux, etc.) Cette activité est appelée « Analyse » sur la Figure 78. Lors de l'analyse, chaque directive de reconfiguration est constituée d'une

suite de paramètres formant une ou plusieurs séquences. De plus, à chaque suite ordonnée de séquences est associée un plan (correspondant à un ensemble d'actions). Les activités « Envoi de plans » et « Envoi de directives » se chargent d'envoyer les plans et les directives du scénario vers un composant (ou sous-agent de l'Agent Décideur). Le plan comprend également la liste des éléments de l'architecture soumis à la surveillance, ainsi que les plans de reconfiguration de l'architecture (c'est-à-dire un ensemble d'actions de reconfiguration).

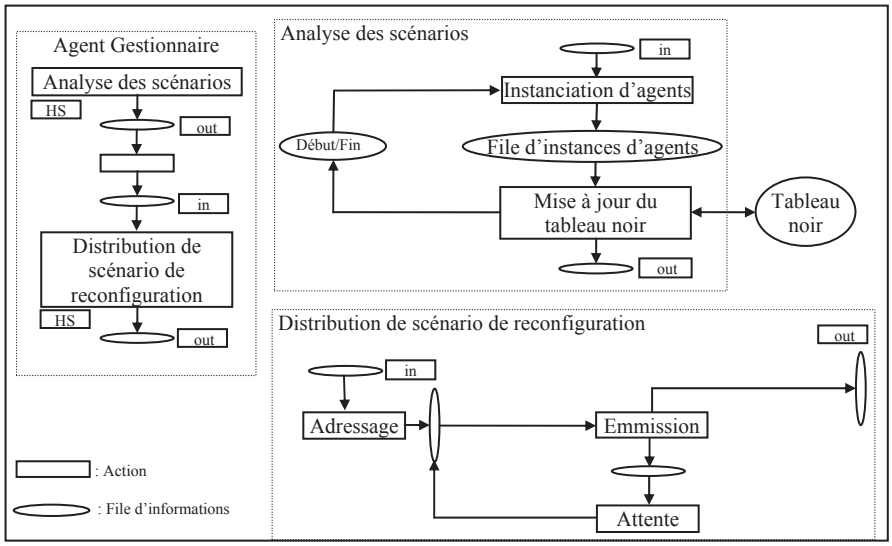

Figure 79 Réseau de Petri de l'Agent Gestionnaire lors de l'initialisation

Son activité d'initialisation permet de mettre à jour les directives de gestion des Agents Réactifs. En même temps, il envoie les stimuli vers les Agents Réactifs responsables de l'évaluation de l'architecture.

Chacun des Agents Réactifs d'évaluation s'active en recevant le stimulus de l'Agent de Décision. Rappelons que lors de l'activation du processus de test de l'architecture, deux types de tests sont disponibles :

a. le mode ‘continu’, pour lequel l’agent teste l’élément de l’architecture et envoie ses résultats à une fréquence précise;
b. le mode ‘détection’ pour lequel l’agent envoie les résultats dès qu’il détecte une exception ou une erreur de fonctionnement de l’élément de l’architecture.

6.3.2.3.2 Remontée de l’information

La remontée des résultats de l’évaluation de l’AL multimodale permet aux Agents Décideurs de juger si ils interviennent sur l’architecture ou pas. La Figure 80 montre la partie du réseau de Petri concerné par la remontée de l’information. Il s’agit de la partie surlignée du réseau qui est impliquée dans ce processus.

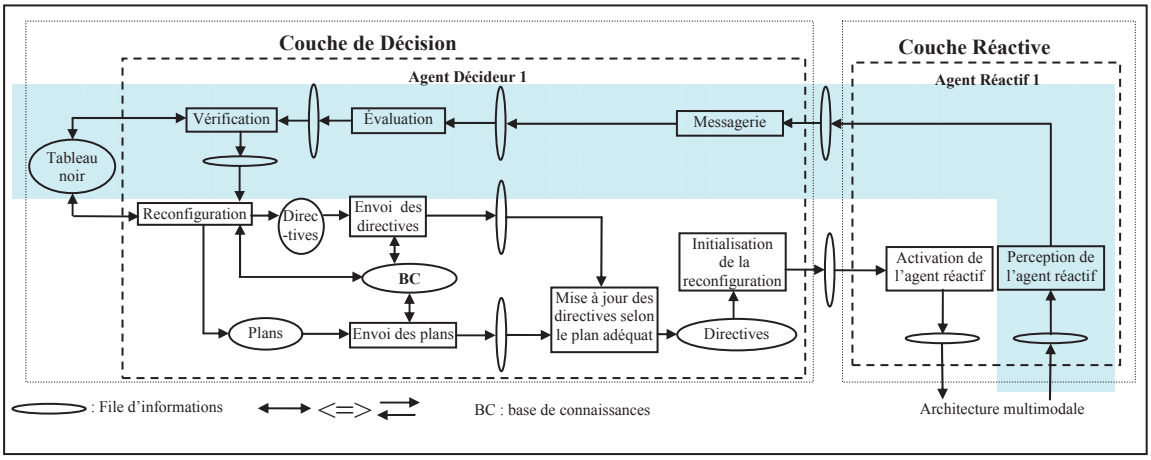

Figure 80 L’AE lors de la phase de la remontée de l'information

Donc, chaque Agent Décideur reçoit les informations demandées (points à risque, sensibles) afin de les comparer aux critères de qualité de son propre scénario. Le processus de remontée de l’information débute à partir des Agents Réactifs, passe par l’Agent de Décideur qui évalue et Vérifie avec l’accord de l’agent Gestionnaire qu’il peut commencer la reconfiguration. En effet, l’Agent Décideur évalue les informations reçues. Si les conditions sur les critères évalués pour l’activation de son scénario ne sont

pas respectées l'agent Décideur entame la procédure de reconfiguration. Cependant, et en même temps, il vérifie si son scénario à la priorité, par rapport aux autres scénarios de reconfiguration en cours d'activité, en faisant appel aux informations actualisées dans le tableau noir par l'Agent Gestionnaire.
Autrement dit, l'Agent Décideur doit faire deux vérifications (activité : « Vérification » sur le réseau de Petri de la Figure 80) avant d'entamer une action sur l'AL multimodale :

a. Vérifier si les critères de son scénario ne sont plus respectés;
b. Vérifier que son activité est toujours prioritaire sur les autres scénarios actifs.

Le composant de l'Agent Décideur employé pour l'initialisation n'est pas forcément le même que celui qui réalise la surveillance et la reconfiguration. Il en va de même pour les Agents Réactifs qui sont différenciés pour leurs tâches de surveillance et reconfiguration.

6.3.2.3.3 Phase de reconfiguration de l'architecture

La phase de reconfiguration est caractérisée par un flux de données descendant (Figure 81). Ce flux correspondant à la prise de décision dans la couche décisionnelle et à une application des actions par la couche réactive.
Si la condition sur le poids de priorité du scénario est satisfaite, l'Agent Décideur envoie les séquences de reconfiguration au agents réactifs. Chaque scénario de reconfiguration respecte un séquençage bien précis afin de respecter l'intégrité de l'application lors de la reconfiguration dynamique de l'architecture. C'est l'Agent Décideur qui est responsable de faire appliquer les séquences d'actions dans le bon ordre.
Grâce aux directives dont il dispose, l'Agent Décideur peut stimuler les Agents Réactifs concernés. C'est-à-dire qu'il applique les directives dans un ordre temporel précisé, en stimulant chaque agent réactif séparément. L'Agent Réactif est de structure très simple, dès qu'il est stimulé, il agit sur l'architecture par instanciation, inhibition, ou modification d'un élément de l'architecture.

Sur la Figure 81, la partie surlignée du réseau de Petri concerne le processus de reconfiguration.

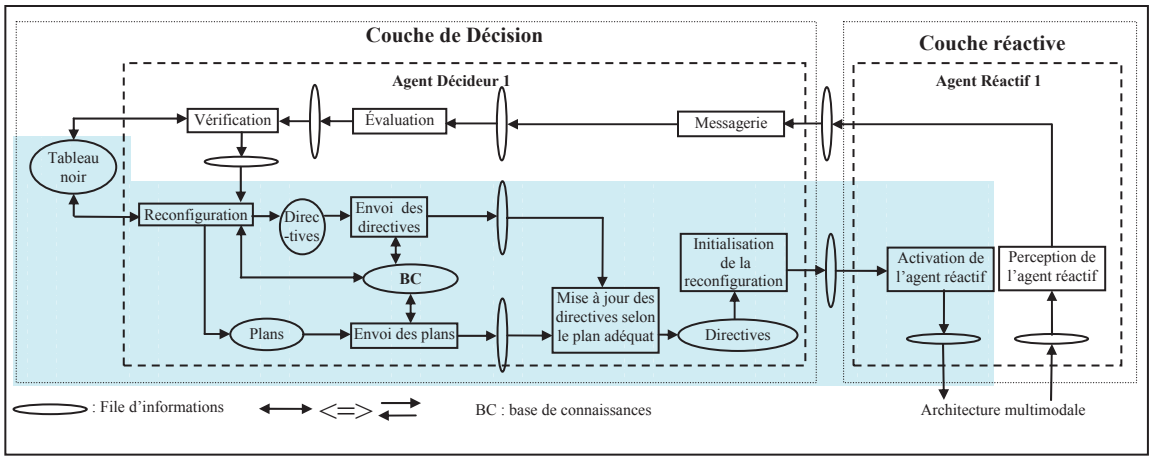

Figure 81 Diagramme d'état des agents de l'AE lors de phase de reconfiguration

6.3.3 Communications

6.3.3.1 Aspects de la communication

Les communications, dans les systèmes multiagent comme chez les humains, sont à la base des interactions et de l'organisation sociale (Genesereth 1994). Sans communication, l'agent n'est qu'un individu isolé, sourd et muet aux autres agents, enfermé sur sa boucle perception/délibération/action. C'est parce que les agents communiquent qu'ils peuvent coopérer, coordonner leurs actions, réaliser des tâches en commun et devenir ainsi de véritables 'êtres sociaux'. La communication s'exprime comme une forme d'interaction dans laquelle la relation dynamique entre les agents se manifeste par l'intermédiaire de médiateurs : les signaux, qui, une fois interprétés, vont produire des effets sur ces agents.

6.3.3.2 Définition et modèles de communication

Il existe de nombreuses théories de la communication mais elles reposent essentiellement sur des variantes de la théorie de la communication issue des recherches en télécommunications des années 40 développées par Shannon et Weaver (Shannon 1963). Dans ce modèle, l'acte de communication consiste en une transmission d'information d'un émetteur vers un récepteur (ou destinataire), cette information étant codée à l'aide d'un langage et décodée à l'arrivée par le destinataire. Cette information est transmise dans un canal (ou médium) de transmission.

Nous utiliserons pour l'AE plusieurs canaux de transmission correspondant aux espaces d'interaction et de communication entre les agents, ils sont aux nombres de trois :

a. au sein de la couche de décision;
b. entre la couche de décision et la couche réactive;
c. entre la couche réactive et l'architecture multimodale.

L'Agent Gestionnaire dispose de plusieurs ports de communication qui le relient au tableau noir (par transmission simple) et aux Agents Décideurs (par transmission simple).
Les Agents Décideurs dédient deux types de ports internes à l'AE pour la transmission des données vers le tableau noir ou des communications avec l'Agent Gestionnaire, et utilisent deux types ports à l'AE pour communiquer avec les Agents Réactifs. Enfin les Agents Réactifs disposent de deux types de ports. L'un est dédié à la transmission des résultats d'évaluation et l'autre est un connecteur pour les communications avec l'environnement architectural.

6.3.3.3 Catégories de communication

Malgré toutes les critiques prononcées à son égard, le schéma classique de la communication est apparu comme suffisamment pertinent par nombre de linguistes pour qu'ils l'intègrent dans leur propre description des faits de langues en classant les manières de communiquer par rapport aux relations qui s'expriment entre les différentes entités concernées par l'acte de communication. Ces relations portent sur :

a. la liaison émetteur\destinataire;
b. la nature du médium;
c. l'intention de communiquer.

6.3.3.3.1 Liaison émetteur-destinataire

Lorsque le destinataire est connu de l'émetteur, ce dernier peut lui adresser des messages en particulier et ainsi instaurer une communication individuelle. Nous parlons alors de communications qui sont effectuées selon un mode point à point. C'est ce type de communication qui est généralement le plus employé dans les agents cognitifs. Nous noterons pour les communications entre l'Agent Gestionnaire et l'Agent Décideur le protocole :

(ID_message, émetteur, destinataire, énoncé) (6.10)

Où 'ID_mesage' est l'identifiant du message, 'émetteur' est l'agent émetteur du message, 'destinataire' est l'agent à qui est destiné le message. Enfin 'énoncé' correspond au contenue du message (directives ou séquence d'actions). Par la suite, le décryptage du message permet l'activation des processus interne des agents.

6.3.3.3.2 Nature du médium

La nature du canal de communication joue aussi un rôle important dans la communication. De manière générale, il existe trois sortes de mécanismes d'acheminement de message : l'acheminement direct, l'acheminement par propagation d'un signal et l'acheminement par voie d'affichage. L'AE dispose de deux types de canaux de communication :

a. L'acheminement direct est le mode de communication le plus simple. Lorsqu'un agent désire envoyer un message par ce mode, il dépose le message dans un canal de communication qui conduit le message directement à son destinataire (ou à tous les destinataires potentiels dans le cas d'un message diffus) sans tenir compte de la 'distance' et sans donner la possibilité à d'autres agents intermédiaire de le recevoir au cours de l'acheminement. Ce type de canal est utilisé lors des communications entre agents Gestionnaire/Décideur, Décideur/ Réactifs.
b. Le mode d'acheminement par voie d'affichage est le moins utilisé dans les systèmes multiagent. C'est typiquement le mécanisme de communication des "petites annonces". Un agent, s'il désire communiquer, place son message dans un espace commun, appelé tableau d'affichage, tableau noir ou "ether" (Kornfeld 1979), visible par tous les agents (ou tous ceux d'une classe particulière). Ce mode de transport combine les caractéristiques des messages directs (rien n'empêche de mettre un destinataire en en-tête) et ceux des messages diffus. L'AE utilise ce type de communication entre ses Agents Décideurs. La consultation du tableau noir permet aux Agents Décideurs de communiquer leurs propres états, mais aussi de s'informer sur l'état des autres agents Décideurs.

6.3.3.3.3 L'intention de communiquer

Lorsqu'une personne envoie un message à un tiers, la communication peut être qualifiée d'intentionnelle. En ce sens, l'émetteur décide résolument de communiquer quelque chose à son (ou ses) destinataire(s). Les communications intentionnelles procèdent par choix. Elles expriment une volonté ou tout au moins une décision d'action. Au contraire les communications incidentes sont effectuées sans que l'émetteur y prenne une part active. Toutes les communications des agents de l'AE découlent d'une intention. Elles font partie de la coordination des agents et sont soumises à des règles de déclenchement strictes. L'intention de communiquer n'est pas une affaire de tout ou rien, mais un système gradué qui dépend des capacités cognitives des émetteurs.

Les études des communications animales ont notamment permis de dégager les rapports complexes existant entre les capacités de représentation d'un individu et les formes de communications qu'il utilise (Vauclair 1992). Dennett (Dennett 1983) a proposé un ensemble de niveaux d'intentionnalité hiérarchisés selon une échelle de complexité qui part de 0 pour aller jusqu'à un niveau théorique n. Ceci reste valable pour les agents des différentes couches de l'AE.

Par exemple, un Agent Réactif possède un ordre d'intentionnalité de 0. Il décrit les situations dans lesquelles l'émetteur envoie un signal parce qu'il présente un certain état interne ou parce qu'il reçoit un certain stimulus (perception d'une exception d'un composant). La production du signal repose alors uniquement sur un rapport stimulus/réponse et ne dépend aucunement d'une quelconque délibération de la part de l'Agent Réactif. C'est ce que nous pouvons exprimer en disant que :

$$\boldsymbol{X}\textbf{ envoie le message }\boldsymbol{M}\textbf{ parce que }\boldsymbol{X}\textbf{ perçoit }\boldsymbol{S} \qquad (6.11)$$

S est un signal. Dans ce cas, ***M*** est appelé 'message incident'. La relation (6.11) se modélise facilement à l'aide des RPCT (voir Figure 82 (a).)

Dans le cas l'Agent Décideur, l'intentionnalité est d'ordre 1, elle suppose l'existence d'une volonté directe de l'émetteur pour obtenir un effet du récepteur. La relation entre le message et l'intention s'exprime de la manière suivante :

X* envoie le message *M* à *Y* parce que *X* souhaite que *Y* effectue une action *Z (6.12)

Ces différents niveaux d'intentionnalité ne sont pas des caractéristiques externes qu'il est possible de déterminer par observation. En effet, si un agent ***X*** envoie un signal à ***Y*** et qu'une réaction de ***Z*** est observée alors il n'est pas possible d'en inférer un à un quelconque niveau d'intentionnalité.

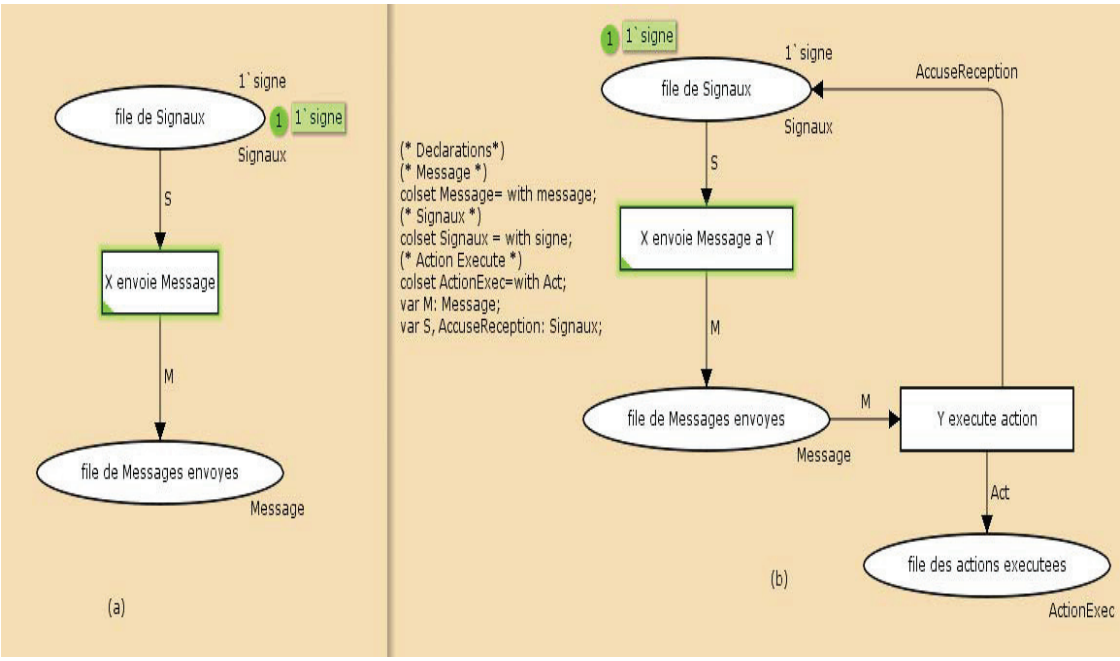

Figure 82 (a) Intentionnalité d'ordre 0 et (b) d'ordre 1 lors de l'envoi de messages

Il se peut tout simplement que ***X*** possède une intentionnalité d'ordre 0 et que ***Y*** soit un agent réactif directement déclenché par le message de ***X***. Il est impossible de trancher simplement entre les deux situations. Pour en avoir le cœur net, il faut avoir accès à

l'architecture de ***X*** et ***Y*** et savoir quels sont leurs états mentaux respectifs. Cependant, les RPCT permettent de spécifier ces niveaux d'intentionnalité comme le montre la Figure 82.

Le Tableau IX donne les différents modes de communication dans les SMA.

Tableau IX

Comparatif des modes de communication dans les SMA

Types de message	Mode de communication	Acheminement	Intentionnalité
Message symbolique	Point à point	Direct	Généralement intentionnel
Message symbolique	Diffusion générale	Direct	Généralement intentionnel
Annonce	Point à point diffusion générale	Tableau d'affichage	Généralement intentionnel
Signal	Diffusion	Propagation	Incident

Dans le paragraphe suivant nous détaillerons le processus de communication entre les agents constituant le SMA AE, via un réseau de Petri, pour un exemple de processus de reconfiguration architectural.

6.4 InterAct

6.4.1 Introduction

InterAct (pour **Inter**action multimodale **Act**ive) est une application multimédia multimodale destinée aux handicapés. Elle a été développée en collaboration avec Monsieur Samir Benarif, doctorant au PRiSM, dans le cadre d'un projet de recherche de coopération franco-québécoise et dont nous avons été le principal initiateur. Le titre du

projet est : Prototypage d'une application Internet multimédia multimodale évolutive générique en vue d'aide aux handicapés (Djenidi 2005).

Ce qui la caractérise, c'est une structure multiagent générique. En effet, du fait de l'emploi de l'AL multimodale présentée précédement au chapitre 3, pour sa conception, InterAct peut accepter n'importe quelle nouvelle modalité en entrée ou en sortie. L'adaptation de la nouvelle modalité est alors réalisée sans grand effort de modélisation. La structure d'InterAct offre des avantages comme la performance et le dynamisme, du fait qu'un AE a été greffé à l'application. Ainsi, il est possible, par exemple, d'activer ou de désactiver les modalités en entrée en mode exécution, ce qui rend le logiciel adaptable au handicap de l'utilisateur.

Dans la littérature, nous rencontrons diverses applications multimodales dédiées aux handicapées comme celles développées au laboratoire LIMSI dans le cadre du projet NICAM (http://www.limsi.fr/Individu/bellik/ 2005) :

a. SeeWeb : adaptation dynamique des pages Web pour les non-voyants
b. EasyWeb : navigateur Web multimodal pour non-voyants
c. MEDITOR : éditeur de textes multimodal pour non-voyants

Cependant, il s'agit d'applications pour les non-voyants alors qu'InterAct Software est dédiée aux handicapés moteurs. InterAct Software est destiné à une catégorie de personnes souffrant de certains types de handicaps moteurs (paralysie entière ou partielle des membres supérieurs et/ou des mains).

Il est important de distinguer deux domaines des TI dédiés aux personnes handicapées. Le premier est celui des personnes ayant des besoins conformes à ceux de tous les utilisateurs 'standard' des TI : accès à Internet, traitement texte, etc. Il consiste à adapter les interface de ces applications pour les rendre accessibles à ces personnes. Le second domaine est la technologie d'assistance. Dans ce cas, la technologie vient jouer le rôle d'une aide offerte aux personnes pour leurs permettre de réaliser des tâches que des personnes non handicapées réaliseraient sans avoir besoin de faire appel à un outil ou

une technologie. InterAct est une application qui s'inscrit dans le premier domaine des TI. Néanmoins l'approche proposée dans nos travaux pourraient s'appliquer au second. L'utilisation de plusieurs modalités dans InterAct permet de gérer plusieurs types de handicaps moteurs afin de faciliter l'accès à l'outil informatique. InterAct Software offre la possibilité à l'utilisateur de choisir les modalités qui lui conviennent. Ainsi, le logiciel s'adapte automatiquement au type du handicap moteur de l'utilisateur et non le contraire (emploi de la voix et/ou de la souris infra-rouge, et/ou etc.) Dans notre application actuelle nous avons testé InterAct Software avec plusieurs modalités en entrée comme le système de détection du regard, la parole (en utilisant des *Speech APIs)*, l'écran tactile, une télécommande infra-rouge pour PC et un clavier virtuel propre à l'application.

Une étude sur les qualités de l'AL permet d'obtenir un logiciel plus fiable, plus convivial et plus simple à utiliser. Dans notre cas, l'effort est porté sur des attributs de qualités comme la performance, la 'maintenabilité' (la gestion des erreurs) et enfin 'l'utilisabilité' (l'aide contextuel, possibilité d'annuler des tâches, facilité d'utilisation, etc.) qui rendent le logiciel plus fiable, plus interactif et plus 'réactif' aux actions de l'utilisateur.

InterAct est également capable de gérer les applications se trouvant sur le système d'exploitation Windows ©. Le développement de processus experts pouvant générer de façon logicielle des événements souris ou clavier, mais aussi des commandes compréhensibles par le système d'exploitation rendent possible l'utilisation des modalités en entrée avec n'importe quel logiciel (de Microsoft sous Windows ©). En fait, nous nous sommes efforcés à généraliser l'utilisation du logiciel InterAct Software sans le confiner uniquemement à sa fonctionnalité de départ, à savoir: la navigation sur le Web.

Nous présentons dans ce paragraphe un rappel sur le principe de conception de la présentation multimodale d'InterAct. Nous détaillons alors l'AL multimodale et l'implantion par l'AE d'un modèle de qualité dédié à InterAct.

6.4.2 Principe de présentation multimodale en sortie

Rappelons qu'une des différences entre la conception d'un système multimodal en entrée et en sortie réside, entre autres, dans le choix pertinent des modalités. Dans le cas de la multimodalité en entrée, c'est l'utilisateur qui choisit les modalités qu'il emploie, à condition que le système offre un choix. Nous pouvons alors considérer que le modèle est centré utilisateur. Au contraire, dans le cas de la multimodalité en sortie, le système doit choisir la ou les modalités les plus adéquates pour communiquer l'information à l'utilisateur en fonction du contexte et de l'environnement. Le modèle est alors centré sur le contexte d'utilisation composé de l'application multimodale et de son utilisateur tous deux baignés dans un environnement.

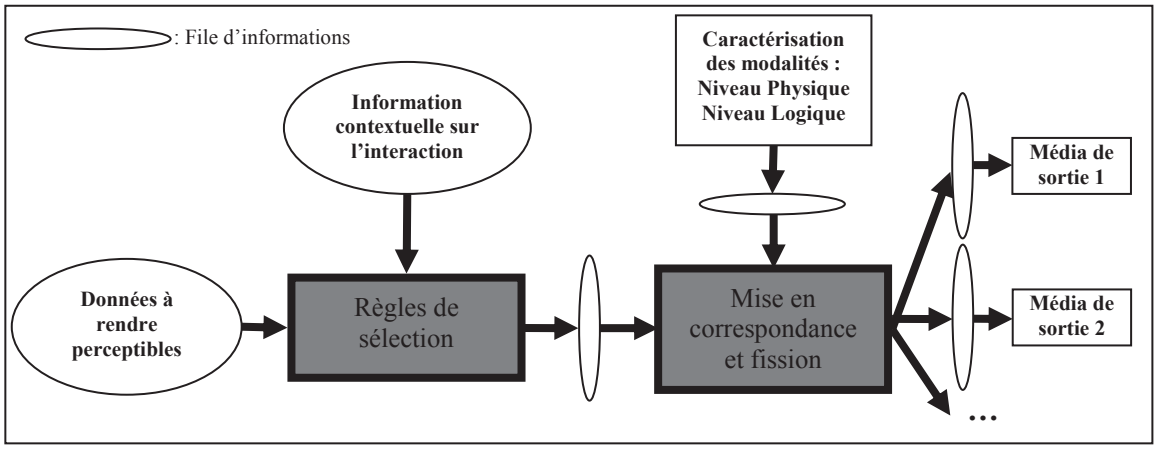

Figure 83 Principe de conception d'une présentation multimodale

À la Figure 83, nous mettons en relation le contexte de l'interaction, l'ensemble des modalités de sortie et de leurs caractéristiques, les données à rendre perceptibles et les règles de sélection et de composition pour définir le principe de la présentation multimodale d'InterAct.

6.4.3 AL multimodale d'InterAct

Pour l'architecture multimodale d'InterAct (Djenidi 2003), nous avons employé une vue simplifiée de la structure multiagents générique du chapitre 3 sur la Figure 84.

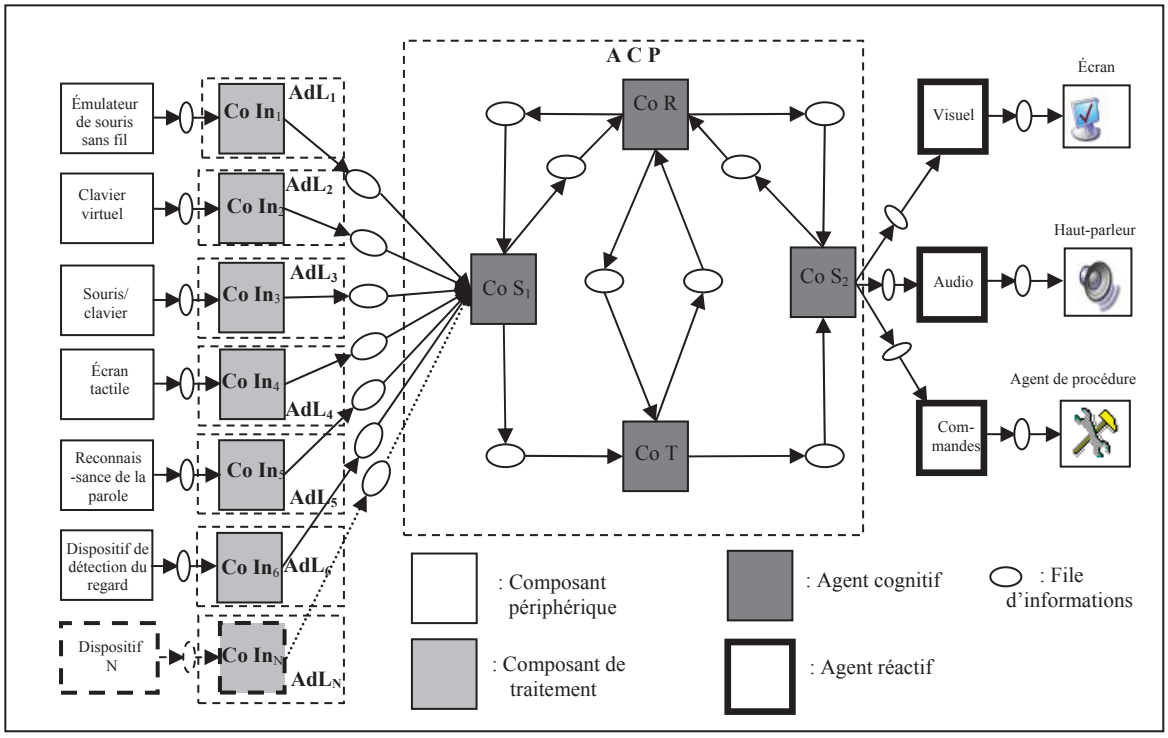

Figure 84 Architecture du dialogue multimodal d'InterAct (A C P : Agent de Contrôle Parallèle, A d L : Agent du Langage, R: Redondance, S: Sémantique, T: Temps, Co: Composant, In : Interprétation)

Peu importe la modalité employée, la structure du système multiagent proposée reste toujours valable. L'aspect dynamique du système sera pris en compte, puisque le système doit s'adapter aux éventuels changements de configuration des modalités en entrée, en mode dynamique, afin de toujours répondre aux requêtes de l'utilisateur. Nous avons opté pour une structure composée d'agents spécialisés qui constituent des instances des AdL et de l'ACP. L'ensemble est défini dans des espaces d'interactions,

évoluant dans un système strict (où les règles et les protocoles sont définis dans la phase de conception). L'introduction d'agents dits intelligents dans la structure va procurer à notre système une autonomie complète et plus de fiabilité grâce entre autres, à une gestion des erreurs de l'utilisateur. Les agents purement réactifs forment la partie applicative de notre système. Leur principale fonction est l'application des requêtes.

Les composants d'interprétation de la Figure 84 sont des agents de traitement des informations des dispositifs d'entrée : d'où leur positionnement à l'entrée du système. Ils ont pour rôle la traduction des messages provenant des transitions représentant les périphériques, afin que tous ces messages soient compris par les agents du système. Ces composants correspondent au composant 'système de reconnaissance' de la Figure 16. Nous les avons intégrés aux AdL sur la Figure 84.

L'ajout de nouvelles modalités non prévues auparavant par le système, n'affecte pas la structure générale du système il suffit de développer les composants d'interprétation spécialisés pour ces nouveaux types de périphériques ou dispositifs.

Les composants sémantiques Co S_1 et Co S_2 embarquent, respectivement, les grammaires multimodales de fusion et de fission. L'ACP réalise une fusion multimodale en entrée et une fission multimodale en sortie. Les intervalles de temps pour la fusion, la défusion et la fission des informations sont choisis par l'architecte dans la modélisation par réseau de Petri en fonction de la nature des données traitées. Elles sont gérées au niveau du composant du Temps (Co T), mais sont aussi distribuées sur les composants sémantiques du réseau.

Différents composants d'interprétation sont décrits ci-après.

Co In_6 :

Ce composant, dédié au système de détection de la position du regard, est lui-même constitué de plusieurs composants (Figure 85). Le composant de calibration est employé pour ajuster le système de détection à l'utilisateur (définition de l'écran et position de l'œil). Le processus de calibration est activé à la première utilisation de l'aplication par un utilisateur donné. Le processus de suivi du regard capture, à une fréquence fixe, la

position du regard sur l'écran et l'exprime en pixels. Ces positions sont stockées dans la Base de données. Le rôle du convertisseur est de collecter la position sauvegardée dans la base de données et de l'envoyer sous forme d'un message normalisée au Co S_1.

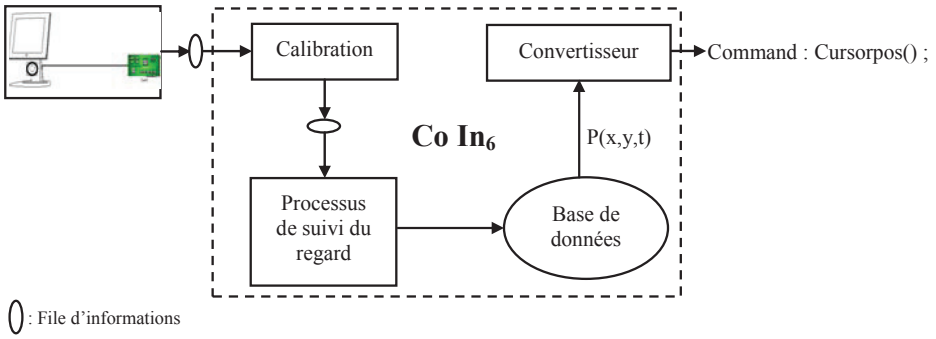

Figure 85 Composant Interprète du système de détection de la position du regard

Co In_5 :

Nous décrivons à la Figure 86 le composant interprète de la parole.

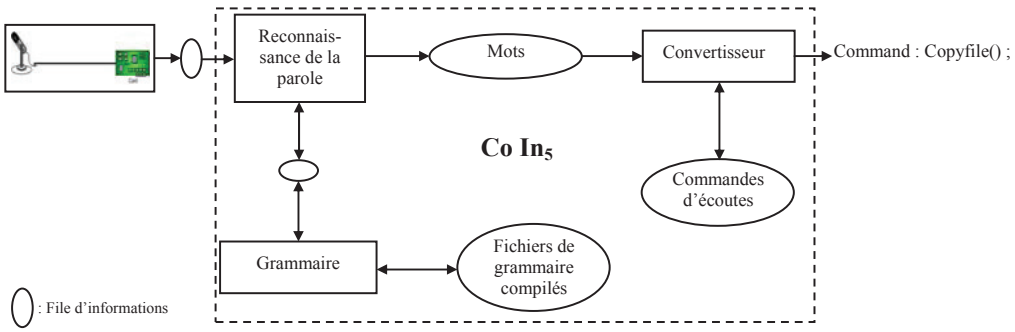

Figure 86 Composant Interprète de la parole

Le convertisseur associe les mots, engendrés par le composant de reconnaissance de la parole, aux commandes se trouvant dans la ressource 'commandes d'écoutes' afin de

générer un message. Le message produit peut correspondre directement à une commande, comme indiqué sur la Figure 86 avec la commande : 'Copyfile'.

6.4.4 Scénarios de reconfiguration d'InterAct

Rappelons que notre approche consiste à atteindre des qualités logicielles multimodales grâce à l'intervention d'un AE. Pour ce faire, des scénarios sont employés pour décrire une interaction que des intervenants ont eue avec l'AL multimodale sous différentes considérations tenant compte des qualités logicielles.

Par exemple, l'évaluation de l'attribut de qualité logiciel 'utilisabilité' est un processus complexe qui nécessite avant tout de définir ce qu'est cet attribut de qualité et quelles propriétés de cet attribut nous souhaitons améliorer. Selon la définition du standard ISO 9241-11 (http://www.ieee.org/web/standards/home/index.html 2006), '*l'utilisabilité est la dimension mesurable (de manière quantitative ou qualitative) pour laquelle un produit logiciel peut être employé par des utilisateurs spécifiques dans l'objectif d'atteindre des buts spécifiques avec efficacité*[33]*, efficience*[34] *et en ayant le sentiment le satisfaction, dans un contexte d'utilisation bien spécifié.'*

La norme ISO 9241-11 explique comment identifier l'information à prendre en compte lors de la spécification et de l'évaluation de l'utilisabilité en termes de mesures de la performance de l'utilisateur et de sa satisfaction. Cette norme donne des directives générales sur la manière de décrire le contexte d'utilisation du produit de façon explicite. Elle indique comment l'utilisabilité d'un produit peut être spécifiée et évaluée comme un élément d'un système de qualité (qui soit conforme par exemple à ISO 9001.) Il existe beaucoup de travaux concernant l'utilisabilité et la multimodalité (Emery 2003; Vitense 2003), (Jacko 2003; Kaiser 2003; Lin 2006). Cependant peu d'entre eux donne une méthodologie générique de la prise en compte de l'utilisabilité ou de tout autre attribut de qualité dans la phase de conception d'une architecture logicielle multimodale.

---

[33] Efficacité comme permettant d'atteindre les objectifs fixés ou d'obtenir les effets attendus.
[34] Efficience comme permettant de les atteindre ou de les obtenir au moindre coût (temps passé, difficultés, aléas).

Nous pouvons citer les travaux de Salber et de ses collègues (Balbo 1993; Salber 1994) ou encore plus récemment ceux de Lin et Imamiya (Lin 2006) qui vont dans le sens de notre approche, vu qu'ils posent les fondements nécessaires à l'élaboration des scénarios pour l'observation et l'étude de l'utilisabilité dans les applications multimodales.

Pour notre part, nous avons transposé, adapté et appliqué les recommandations de la norme ISO 9241-11 au cas de notre prototype architectural multimodal dédié aux handicapés et à son AE.

La Figure 87 montre les différents choix de scénarios que nous avons intégrés pour améliorer les quatre sous caractéristiques de l'utilisabilité qui sont les suivants.[35]

a. *Understandability* : Facilité de compréhension de l'utilité du logiciel (le degré avec lequel le but, l'intérêt et l'utilité du système est clair pour l'utilisateur.)
b. *Learnability* : Facilité d'apprentissage du logiciel. (caractéristiques du logiciel qui concernent l'effort des utilisateurs pour apprendre son application : commandes, entrées, rendement.)
c. *Attractiveness :* Attribut d'un logiciel qui donne lieu à la satisfaction des désirs et préférences d'un utilisateur latent, via des services, le comportement et la présentation.
d. *Operability* : Facilité d'opérer sur le système (attributs du logiciel qui concernent l'effort des utilisateurs pour l'opération et la commande d'opérations.)

La caractéristique de *learnability* peut à son tour être décomposée en cinq sous caractéristiques: *familiarity, consistency*[36]*, generalizability, predictability* et *simplicity.*

De même, une seule caractéristique peut s'améliorer grâce à différents scénarios. Par conséquent (nous l'avons vu aussi au chapitre 5), afin de pouvoir être traité par l'AE,

[35] Nous donnons ici les terminologies anglo-saxonnes.
[36] Lorsqu'un langage de commandes ou un ensemble d'actions est ordonné, descriptible par des règles peu nombreuses et donc facile à retenir on parle alors d'interface logicielle avec la sous caractéristique de : *consistency*.

chaque scénario s'insère dans un processus de génération d'un profil de qualité (Voir pour cela la Figure 57 de ce document).

Rappelons que chaque scénario est géré par un Agent Décideur.

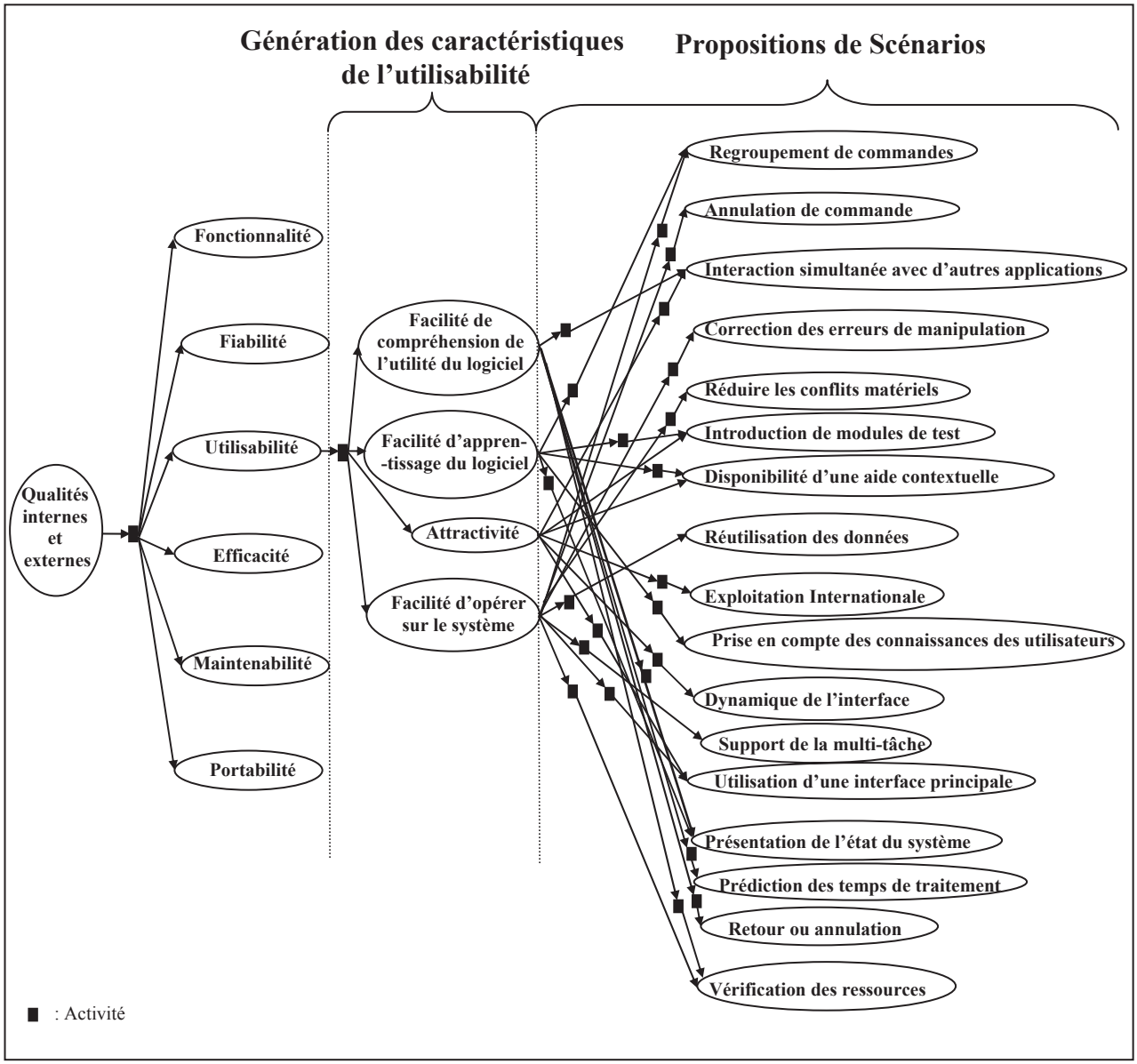

Figure 87 Proposition de scénarios de propriétés d'un attribut de qualité: l'utilisabilité, pour le prototype architectural d'InterAct Software.

Comme le montre la Figure 87, plusieurs scénarios peuvent améliorer plusieurs caractéristiques d'un même attribut de qualité. L'attribution de la priorité d'exécution d'un scénario par rapport à un autre est la tâche de l'agent gestionnaire. Un agent gestionnaire a donc la tâche de gérer un profil de qualité.
Dans le paragraphe qui suit, nous présentons un exemple de profil de qualités construit à partir de scénarios tenant compte :

a. d'une caractéristique de l'attribut utilisabilité ;
b. de la performance du logiciel InterAct : caractéristique de l'efficacité ;
c. d'un compromis entre ces deux attributs de qualité.

6.4.5 Exemple de profil de qualités pour InterAct

Nous donnons dans ce paragraphe un exemple qui permet de comprendre comment le profil de qualités est intégré et appliqué par l'AE sur l'AL multimodale.

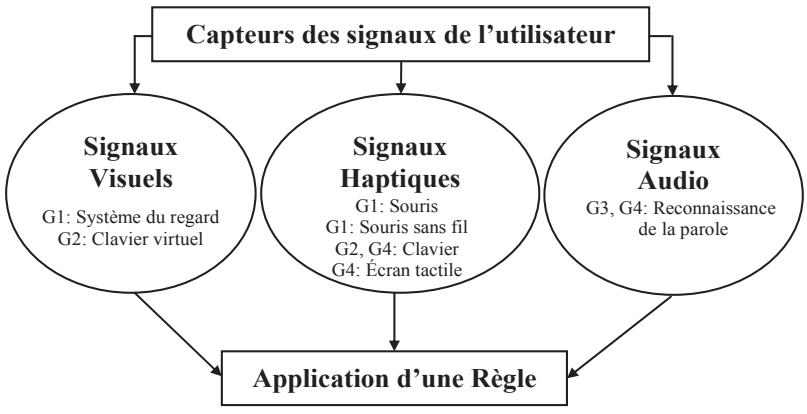

Figure 88 Signaux captés à l'entrée d'InterAct.

Un emploi synergique et parallèle des modalités en entrée d'InterAct peut conduire à des erreurs sur l'interprétation des signaux captés en entrée de l'application. En effet,

différentes modalités ont été introduites dans 'InterAct Software' pour simuler les événements souris et clavier (voir Figure 88).
Par exemple, l'écran tactile et le système de détection de la parole peuvent simuler les fonctionnalités de sélection de la souris. De même, le clavier virtuel et le clavier physique peuvent entrer en conflit lorsque l'utilisateur tente d'agir sur les deux dispositifs simultanément. Pour résoudre ce type de conflit, nous imposons une règle de gestion de l'activation des modalités d'entrée.
Pour ce faire, les modalités sont classées en quatre groupes :

a. Groupe 1 (simulateur ou générateur de mouvement de la souris): souris, système de détection du regard sur l'écran et souris sans fil de contrôle par infrarouges.
b. Groupe 2 : clavier et clavier virtuel.
c. Groupe 3 (commande vocale): Système de reconnaissance de la parole (sans la prononciation du mot 'select.')
d. Groupe 4 (sélection d'un objet) : pointage sur un écran tactile, appui de la touche 'Enter' (ou 'retour chariot') du clavier, action de cliquer sur la souris et reconnaissance vocale du mot 'select'.

La proposition architecturale correspondante à l'application du scénario de gestion de l'emploi des modalités est 'symbolisée' à la Figure 89. C'est le scénario qui embarque la règle d'activation des modalités. Cette règle d'activation des modalités en entrée est simple : l'utilisateur ne peut pas activer simultanément deux modalités appartenant au même groupe (par exemple : le système de détection du regard et le mouvement de la souris). Cependant, toute modalité peut être employée conjointement avec une autre modalité pourvu qu'elles appartiennent respectivement à deux groupes distincts (par exemple l'écran tactile et le système de reconnaissance de la parole). L'application de cette règle d'activation est le résultat de l'application d'un scénario suite à la capture de stimuli par l'AE. En effet, dès qu'une nouvelle modalité est activée par l'utilisateur,

l'AE emploi le scénario pour gérer de manière autonome les modalités d'entrée et évite ainsi des erreurs à l'utilisateur. Nous appelons le scénario qui permet d'éviter les conflits matériels à l'entrée de l'application : scénario_1. Par exemple, si une nouvelle modalité est activée en entrée, son agent réactif correspondant (se trouvant dans la couche réactive de l'AE) capture l'évènement et l'envoie à la couche de décision.

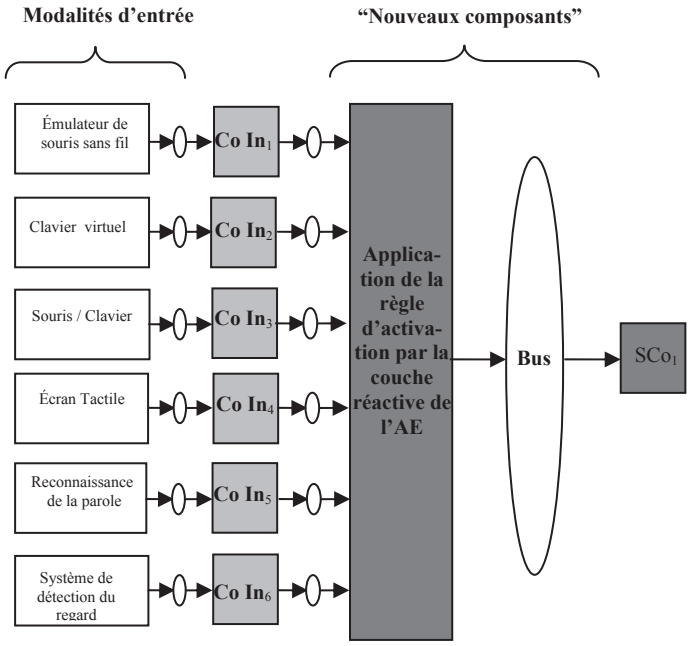

Figure 89 Représentation symbolique de la règle d'activation des modalités dans InterAct.

L'AE teste si la modalité activée est employée au même moment qu'une autre modalité appartenant au même groupe. Si la nouvelle modalité cause conflit, l'AE établit un plan basé sur le scénario_1. Le plan consiste en la désactivation de la nouvelle modalité selon une règle. Cependant, l'application de ce plan dépend du paramètre de priorité du scénario_1 par rapport aux autres paramètres de priorité assignés aux autres scénarios courants et présentement actifs. Si les contraintes de priorité sont respectées, la couche réactive désactive alors le Co In concerné et sa connexion avec le S Co_1. La modalité nouvellement activée est ainsi automatiquement inhibée.

Afin de décrire les paramètres et informations qui définissent, identifient et caractérisent les scénarios nous nous référons à la Figure 57 de ce document. Ces informations et paramètres sont simplement saisis dans des fichiers textes (par les intervenants architectes de l'application) selon un format donné dans l'exemple suivant:

(*====Exemple de fichier de paramètres pour le scénario_1====*)
(*==*)
Numéro du scénario : '**1**'
Nom du scénario : '**scenario_1**'
But du scénario : '**improve usability by reducing input modalities' inconsistency**'
Priorité du scénario : '**8/10**'
Attributs visés par le scénario: '**usability**'
(*===========Stimuli et réponses du scénario_1======*)
Nom du stimulus : '**Modality_6_activated**'
Instance du stimulus : '**STIMULUS(1) = "active"** ' ;
Réponse désirée: '**1 second to validate the modality**'
Fonction donnant le temps de réponse: ' **TimeResponse (1) = CounterStart (1)** '
(*=====Monitorage pour le scénario_1=======*)
Numéro de monitorage : '**1**'
Monitorage # 1: '**eye gaze modality**'
Type de monitorage : '**component**'
Nom des composants ou connecteurs monitorés : '**Co_In6**'
Mode de monitorage : '**tracing**'
Instance du monitorage du composant impliqué par le stimulus STIMULUS(1) : **MONITOR_S(1) = 'eye_gaze.state;**'
(*==========================*)
Numéro de monitorage : '**2**'
Monitorage # 2: '**mouse modality**'
Type de monitorage : '**component**'

Nom des composants ou connecteurs monitorés : '**Co In_4**'

Mode de monitorage : '**tracing**'

Instance du monitorage du composant impliqué par le stimulus STIMULUS(1) : **MONITOR_S(2) = 'mouse.state;'**

(*==========*)

Numéro de monitorage : '**3**'

Monitorage # 3: '**wireless mouse modality**'

Type de monitorage : '**component**'

Nom des composants ou connecteurs monitorés : '**Co In_1**'

Mode de monitorage : '**tracing**'

Instance du monitorage du composant impliqué par le stimulus STIMULUS(1) : **MONITOR_S(2) = 'Wmouse.state;'**

(*==========Raisonnement suivi pour le scénario_1=========*)

'Vérifie si le point de risque est évité'

'Vérifie la règle'

'Désactive la dernière modalité activée'

'Envoie un message à l'utilisateur'

Raisonnement (1): Execute(Action(1,1));
Execute(Action(1,2));
Execute(Action(1,3));

(*========Actions pour l'exécution du scénario_1=========*)

Action: Désactive les modalités en conflit

Action(1,1): Rule();

Action(1,2): deactivate();

Action(1,3): message();

(*==========Points à risque du scénario_1==========*)

Points à risques: '**empty**'

(*========Points de compromis du scénario_1========*)

Points de compromis: perturbe la performance

Compromis #: '**1'**
Attributs impliqués dans le compromis: '**efficiency** and **usability'**
Contrainte du monitorage du compromis: **CPU.charge > 60%;**
TradeOff(1): Apply.Scenario(1);
(*========Fin du scénario_1========*)

Dans la section des 'Points de compromis du scénario_1' de l'exemple précédent, la performance qui est une caractéristique de l'attribut de qualité efficacité doit être maintenu par l'AE. Le scénario_1 doit donc tenir compte de la performance. Le scénario concernant la performance que nous appellons scénario_2 est donné dans l'exemple ci-après.

(*====Exemple de fichier de paramètres pour le scénario_2====*)
(*==========*)
Numéro du scénario : '**2**'
Nom du scénario : '**scenario_2**'
But du scénario : '**Improve the performance of the application**'
Priorité du scénario : '**10/10**'
Attributs visés par le scénario: '**efficiency**'
(*==========Stimuli et réponses du Scénario_2======*)
Nom du stimulus : '**empty**'
Instance du stimulus : '**empty**'
Réponse désirée: '**1 second to apply the scenario**'
Fonction donnant le temps de réponse: ' **TimeResponse (2) = CounterStart(1)** '
(*=====Monitorage pour le Scénario_2=======*)
Numéro de monitorage : '**1**'
Monitorage # 1: '**eye gaze modality**'
Type de monitorage : '**component**'
Nom des composants ou connecteurs monitorés : '**Co_In6**'

Mode de monitorage : '**tracing**'

Instance du monitorage du composant impliqué par le stimulus STIMULUS(1) : **MONITOR_S(1) = 'eye_gaze.state;**'

(*==========*)

Numéro de monitorage : '**2**'

Monitorage # 2: '**mouse modality**'

Type de monitorage : '**component**'

Nom des composants ou connecteurs monitorés : '**Co In_4**'

Mode de monitorage : '**tracing**'

Instance du monitorage du composant impliqué par le stimulus STIMULUS(1) : **MONITOR_S(2) = 'mouse.state;'**

(*==========*)

Numéro de monitorage : '**3**'

Monitorage # 3: '**wireless mouse modality**'

Type de monitorage : '**component**'

Nom des composants ou connecteurs monitorés : '**Co In_1**'

Mode de monitorage : '**tracing**'

Instance du monitorage du composant impliqué par le stimulus STIMULUS(1) : **MONITOR_S(2) = 'Wmouse.state;'**

(*==========Raisonnement suivi pour le Scénario_2=========*)

'Vérifie si le point de risque est évité'

'Active la modalité qui consomme le moins de ressources'

'Désactive les autre modalités du même groupe'

'Envoie un message à l'utilisateur'

Raisonnement (2): Execute(Action(2,1));
Execute(Action(2,2));
Execute(Action(2,3));

(*========Actions pour l'exécution du scénario_2=========*)

Action: Désactive les modalités en conflit

Action(2,1): BestModality();

Action(1,2): deactivate();

Action(1,3): message();

(*==========Points à risque du scénario_2==========*)

Points à risques: '**empty**'

(*========Points de compromis du scénario_2========*)

Points de compromis:

Compromis #: '**empty**'

Attributs impliqués dans le compromis: '**empty**'

Contrainte du monitorage du compromis: '**empty**'

(*========Fin du scénario_2========*)

Ces scénarios contiennent les précieuses informations qui nous permettent de réaliser la modélisation par RPCTS de l'AE qui exécute la reconfiguration architecturale dynamique de l'AL multimodale. Nous pouvons noter que le scénario_2 ne fait aucun compromis, vu que sa priorité est de 10/10. Ceci explique pourquoi la section des points de compromis de ce scénario est vide ('**empty**'). L'emploi de chacune des modalités peut en effet consommer 'beaucoup' des ressources CPU.

Ainsi, lorsque les ressources systèmes employées par l'activation d'une modalité en entrée dépassent 60% des ressources CPU, le scénario_2 est exécuté au détriment du scénario_1.

Le Tableau X donne un exemple de l'emploi du temps CPU par les modalités d'entrée. Ce sont ces grandeurs quantitatives qui sont exploitées par l'AE pour la sélection et l'application du scénario prioritaire.

Ainsi le scénario concernant l'emploi des resources systèmes (scénario_2) est appliqué en priorité sur le scénario concernant un attribut de l'utilisabilté. Ce compromis entre attribut de qualités est géré de manière autonome par l'AE. La séquence d'actions réalisée par l'AE lors de l'exécution du scénario_1 (après vérification des priorités) est

décrite par le réseau de Petri à la Figure 90. Sur cette figure, le cheminement de l'information est représenté symboliquement par une numérotation des arcs entrants sur les places du réseau.

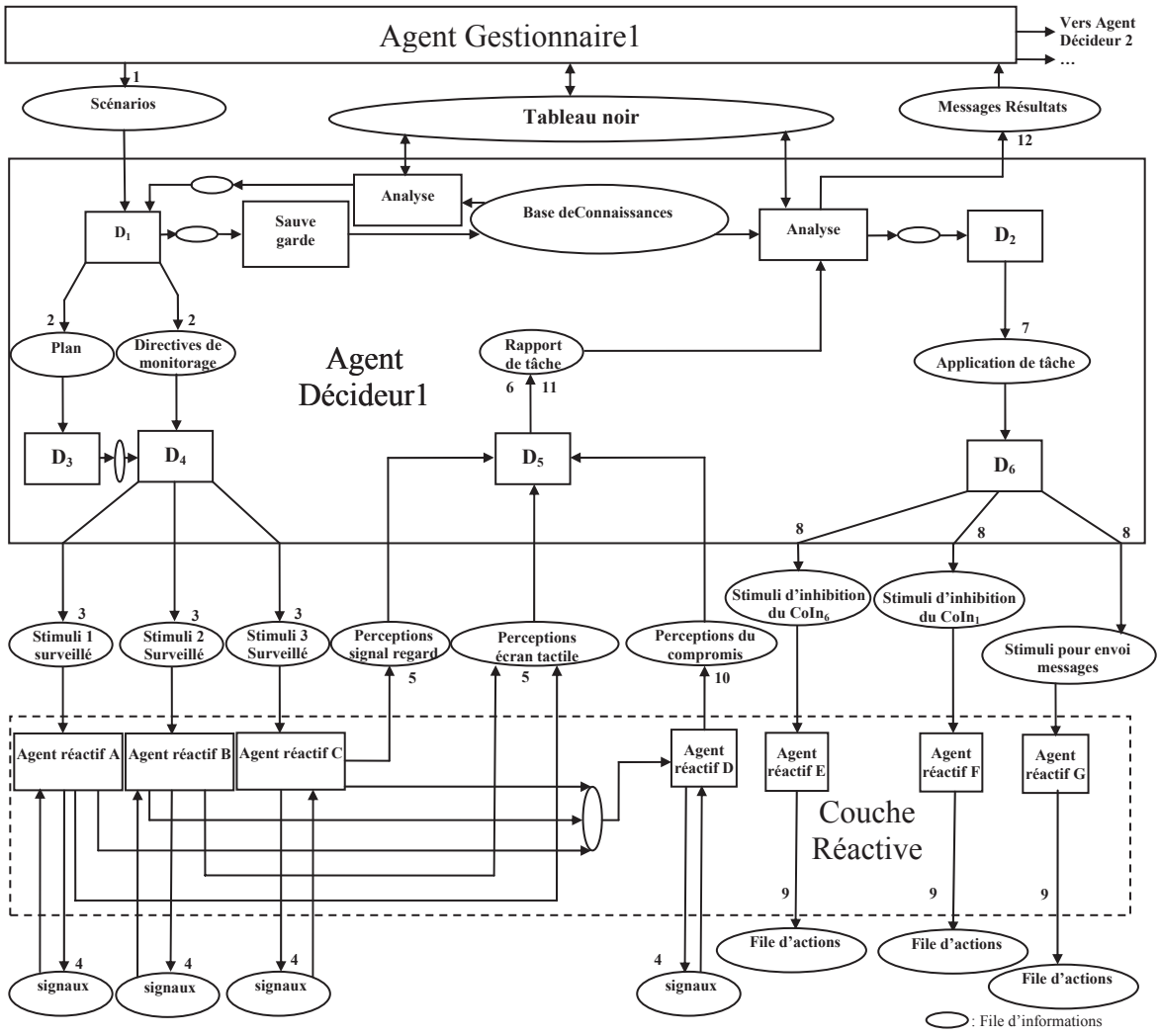

Figure 90 Principe de processus d'application du scénario au sein de l'AE

Cette numérotation de la séquence d'actions va de l'étape 1 à l'étape 12 (voir Figure 90). Sur cette figure, les files d'attente des signaux des différents média (signal du regard, écran tactile) impliqués dans le compromis et les signaux du compromis lui-même sont

clairement spécifiés dans des files d'attentes représentées par des places. À l'étape 8 de la séquence d'actions exécutées par ce réseau, les files d'attente contiennent les signaux d'application de la consigne pour le compromis (inhibitions des composants 1 et 6 et messages informationnels.) Vu que l'Agent Gestionnaire 1 est relié à d'autres agents décideurs non représentés sur le réseau (voir en haut à droite de la Figure 90), la Figure 90 est donc une représentation partielle de l'AE.

Tableau X

Exemple d'utilisation des Ressources systèmes dans InterAct

Modalité	**Charge du CPU : Pentium III (850 MHz)**	**Charge du CPU : Pentium VI (2 GHz)**
Système de détection du regard	60%	35%
Écran tactile	20%	8%
Souris télécommande à infra rouge	5%	1%
Reconnaissance vocale (commandes vocales simples : d'au plus trois mots.)	10%	10%

Le profil de qualités choisi dans cet exemple est donc intégré dans l'AE qui l'applique de manière autonome et automatique pour reconfigurer l'AL en cours d'utilisation.
L'avantage d'un modèle en réseau de Petri de l'AE qui se greffe sur le modèle de l'AL multimodale (lui-même en RP) reste que les invariants du système peuvent être vérifiés au niveau de chaque transition du réseau de Petri comme condition de leurs franchissements. Pour ce faire, l'outil CPN-Tools dispose d'une fonctionnalité qui permet de faire une analyse automatique pour déterminer les propriétés du RP.

6.4.6 Simulation expérimentale de l'application de scénarios

Les réseaux aux Figures 91 et 92 ont servi à simuler, respectivement, les scénarios 1 et 2 présentés au paragraphe précédent. L'Annexe 6 donne la liste des déclarations nécessaires à la simulation de ces réseaux.

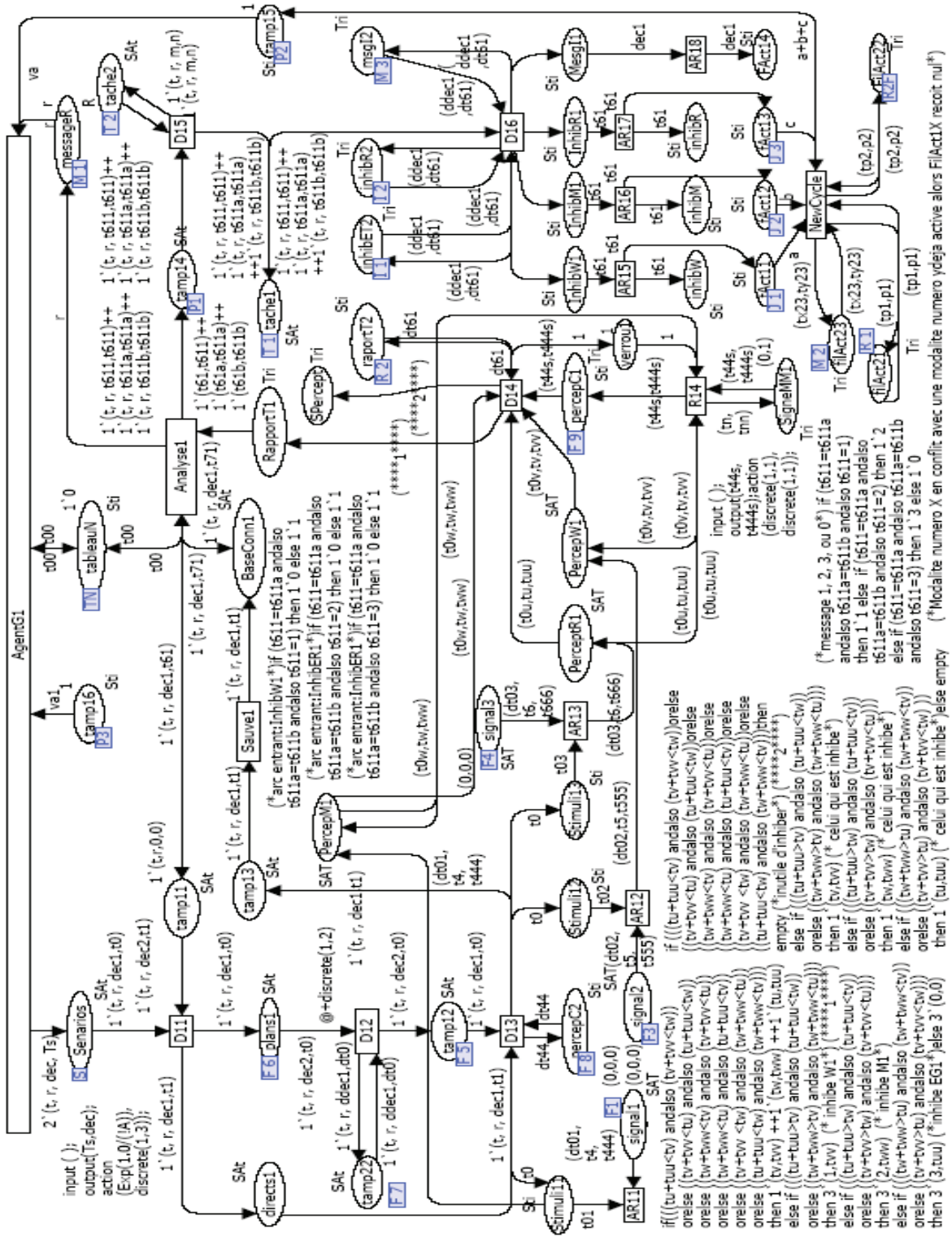

Figure 91 AE composé des agents Gestionnaire, 1 Décideur 1 et de 7 agents réactifs

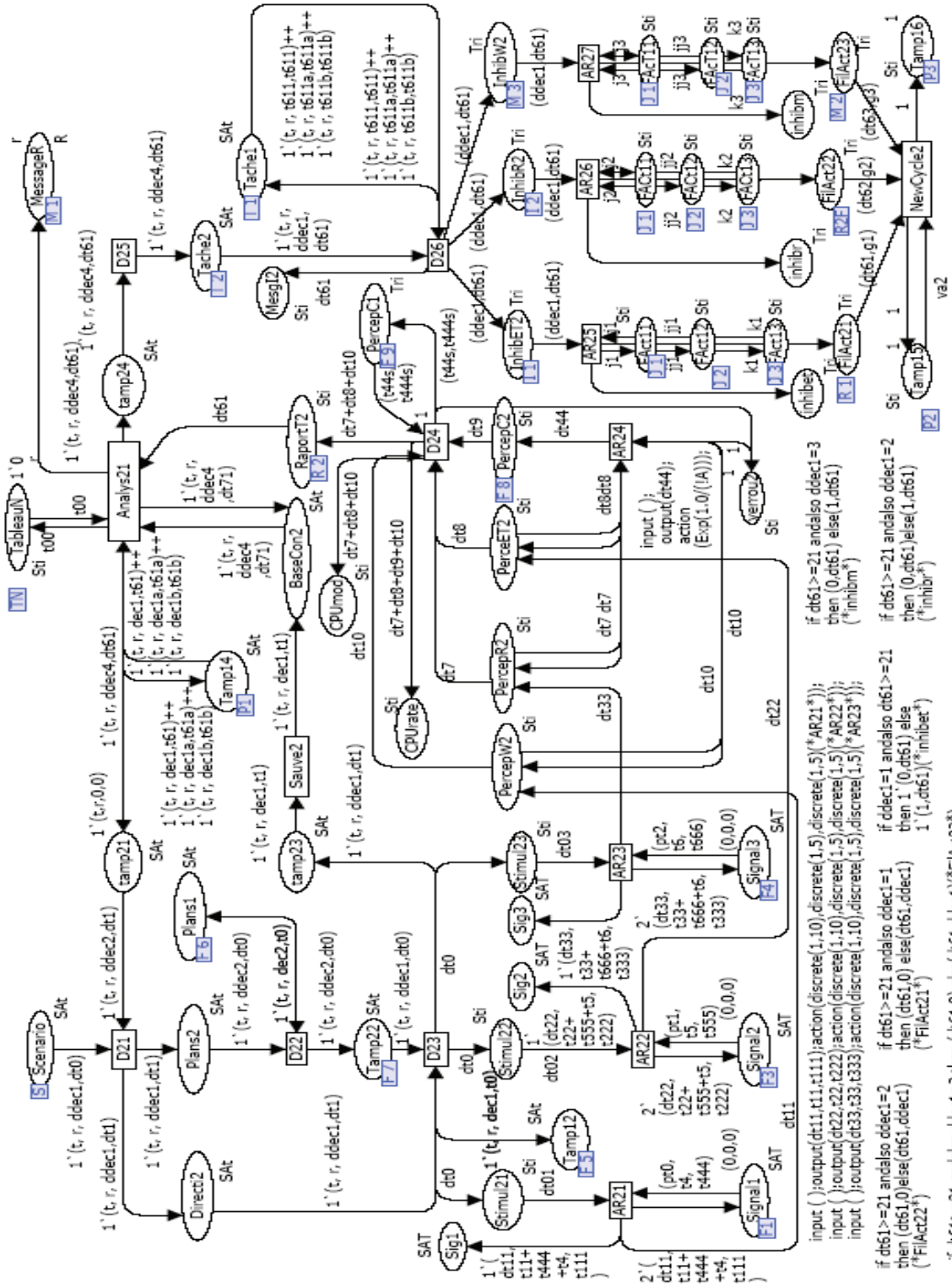

Figure 92 AE composé de l'agent Décideur 2 et de 7 agents réactifs

Sur la Figure 91 (respectivement 92) précédente, les transitions commençant par D1 (respectivement D2) modélisent des activités de l'agent Décideur 1 (respectivement 2).
Les agents réactifs sont modélisés par les transitions dont le nom commence par 'AR'.
Les résultats de la simulation du scénario_2 sont présentés aux figures 93 à 96. L'agent Distributeur 2 inhibe aléatoirement une des trois modalités mises en jeux dans le processus décrit par le scénario 2, lorsque la charge du processeur dépasse 100%. Le réseau a été conçu pour toujours faire en sorte que le processeur ne soit pas saturé au cours du temps lorsque les événements multimodaux se produisent (en parallèle ou en séquentiel). Il remplit donc sa fonction de monitorage et de contrôle de manière 'intelligente' et automatique. 'L'intelligence' provient du fait que nous savons que l'agent va réaliser sa mission, même si nous ne sommes pas capable de prévoir, à l'avance, à chaque instant, de façon procédurale, laquelle des trois modalités sera inhibée par l'agent. L'automatisme provient des comportements cycliques invariants dans le réseau qui permettent la surveillance l'AL multimodale.
Simultanément à l'exécution du scénario_2 et de manière ininterrompue, l'AE surveille et contrôle les modalités d'entrée de l'AL pour l'application du scénario_1 (Figure 91). La partie de l'AE, chargée de l'application du scénario_1, tient compte pour cela de l'instant de début des événements captés sur les modalités d'entrée, ainsi que de leurs durées respectives. Ceci permet d'appliquer la règle d'activation et/ou d'inhibition sur les modalités conflictuelles. Comme pour le scénario_2, l'architecture du réseau de la Figure 92 est un automate qui applique le scénario_1 en respectant des invariants cycliques. Les actions appliquées par l'AE sur l'AL multimodale, qui inhibent les signaux des modalités, sont accompagnées de messages émis par l'AE au niveau de chacun des agents Décideurs.
Lors de l'exécution pas à pas du réseau, représenté aux deux figures 91 et 92 précédentes, nous observons le comportement conforme de l'AE quant au scénario_1 et au scénario_2. Les quatre figures suivantes (93 à 96) montre comment réagit l'AE au cours du temps, lors de la variation des signaux : signal 1, signal 2 et signal 3 des Figures 91 et 92.

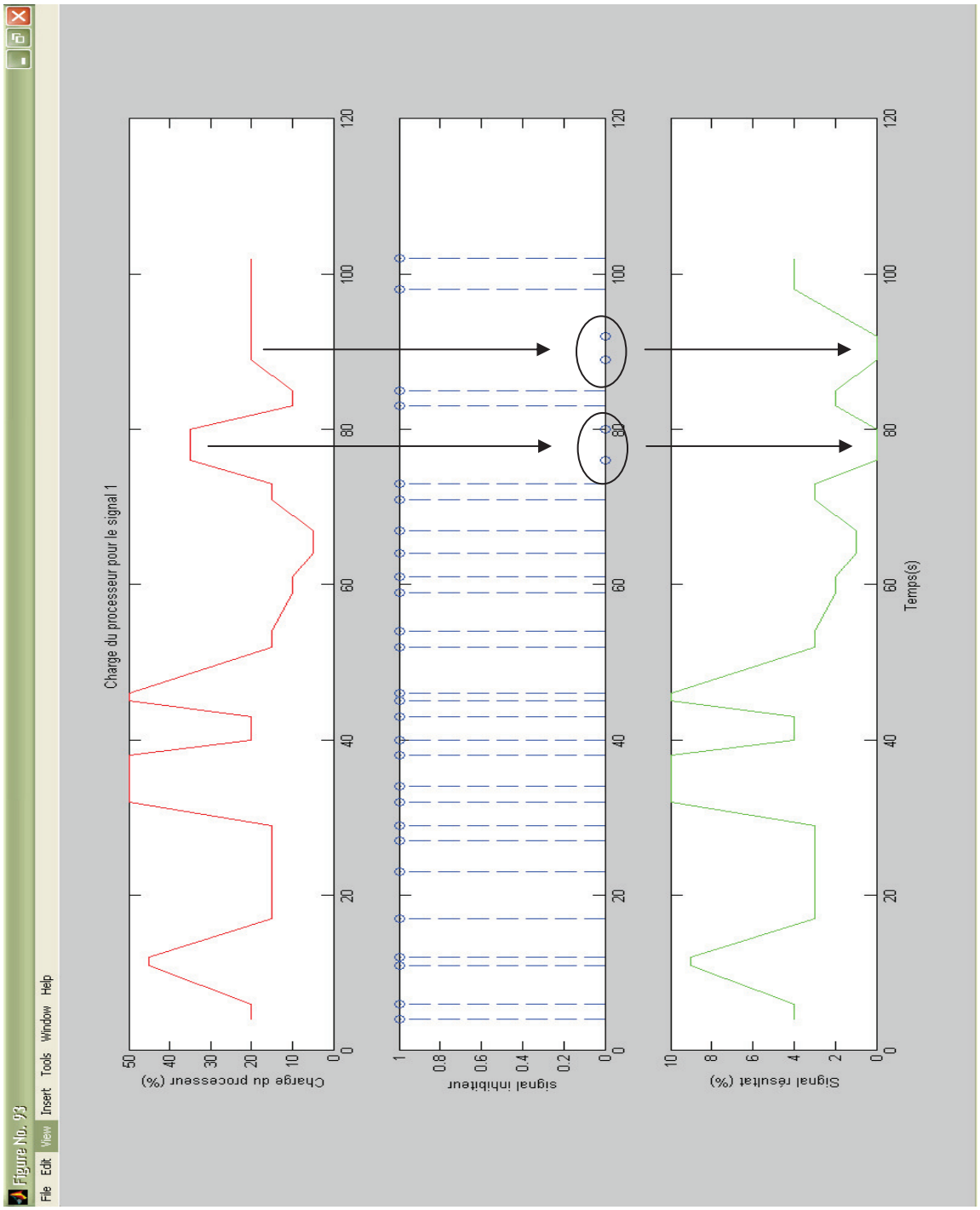

Figure 93 Résultats du contrôle, par l'agent Distributeur 2, du signal, correspondant à la modalité 1, capté par les agents Distributeurs 1et 2 via leurs couches réactives, en fonction du temps

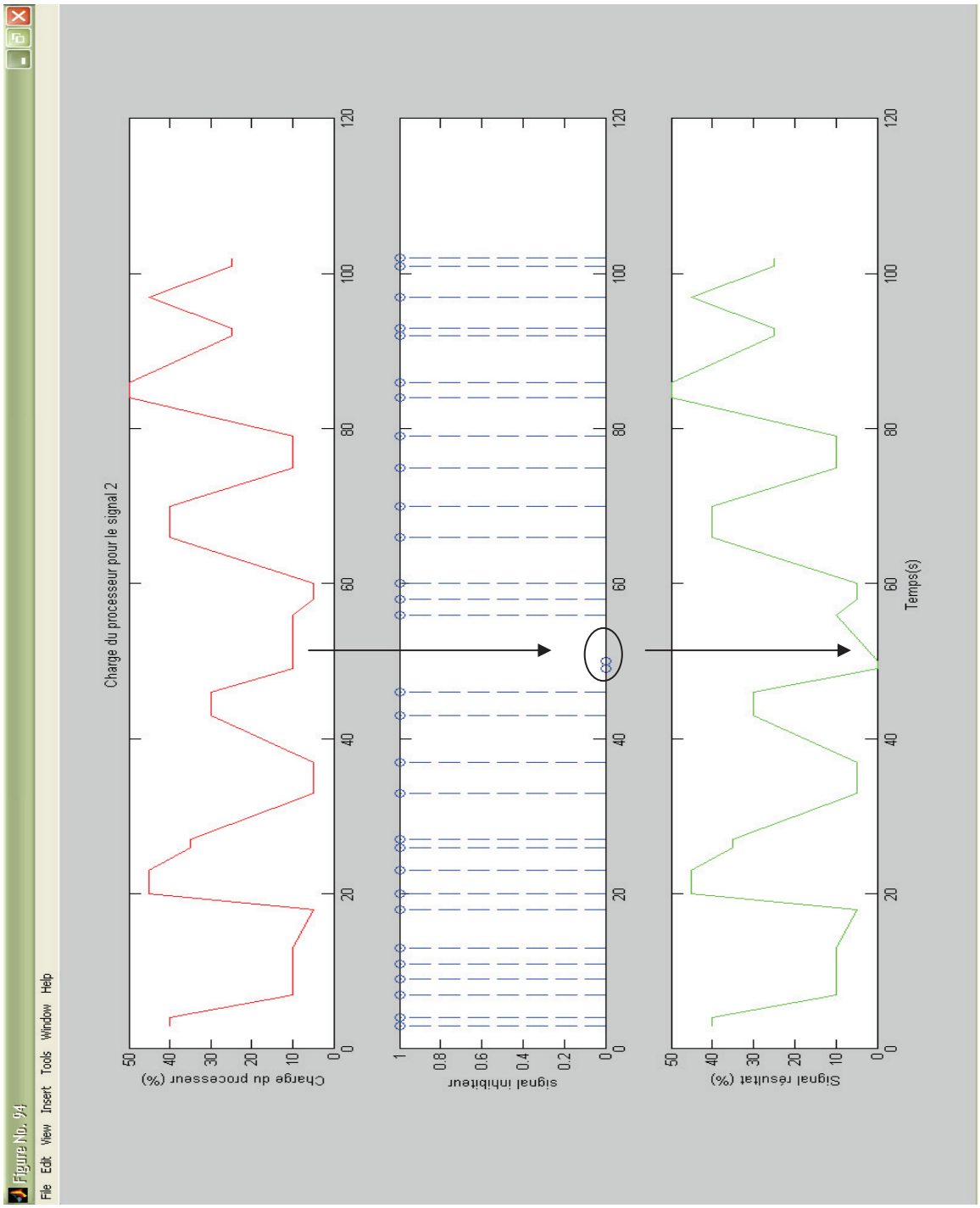

Figure 94 Résultats du contrôle, par l'agent Distributeur 2, du signal, correspondant à la modalité 2, capté par les agents Distributeurs 1et 2 via leurs couches réactives, en fonction du temps

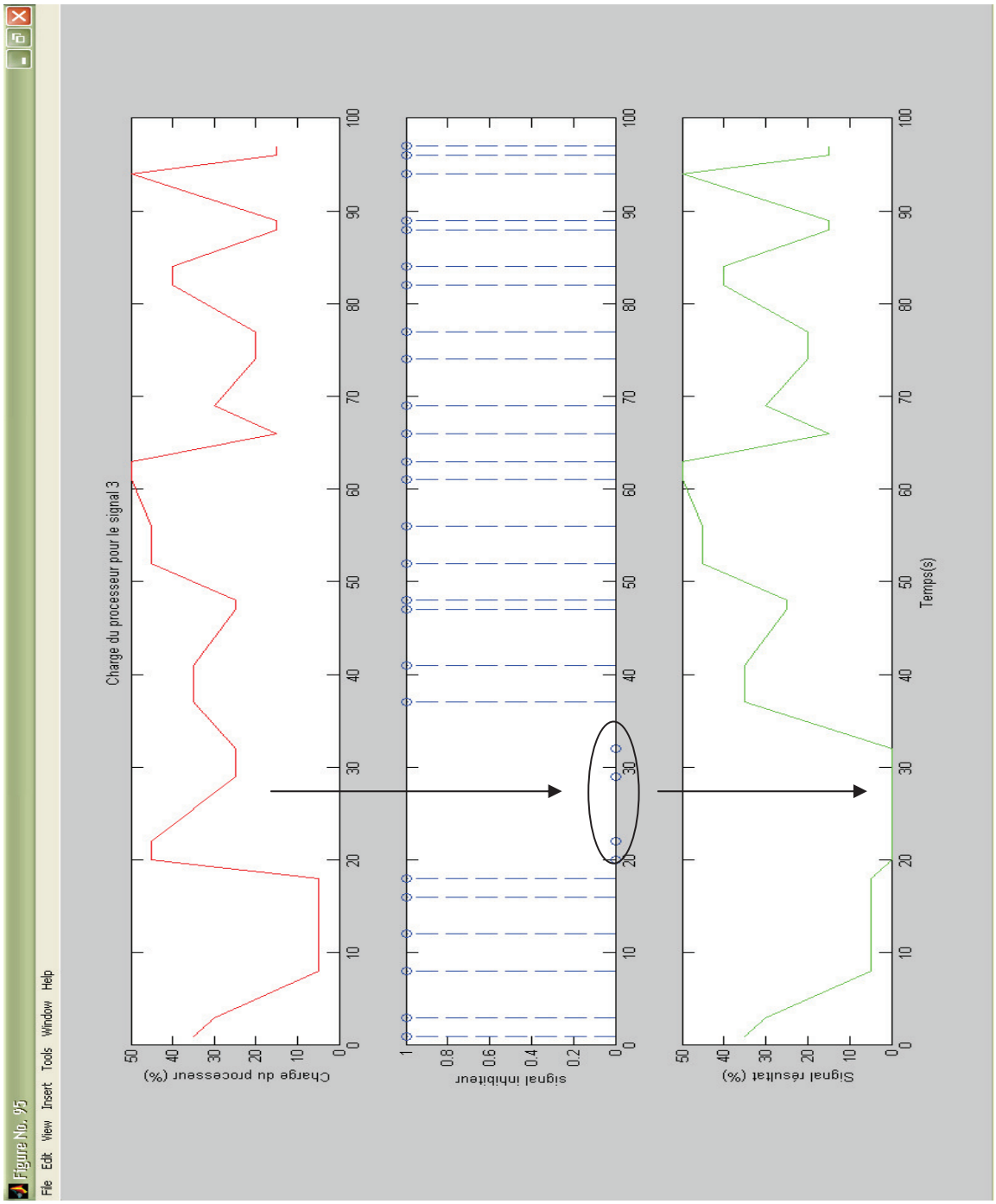

Figure 95 Résultats du contrôle, par l'agent Distributeur 2, du signal, correspondant à la modalité 3, capté par les agents Distributeurs 1et 2 via leurs couches réactives, en fonction du temps

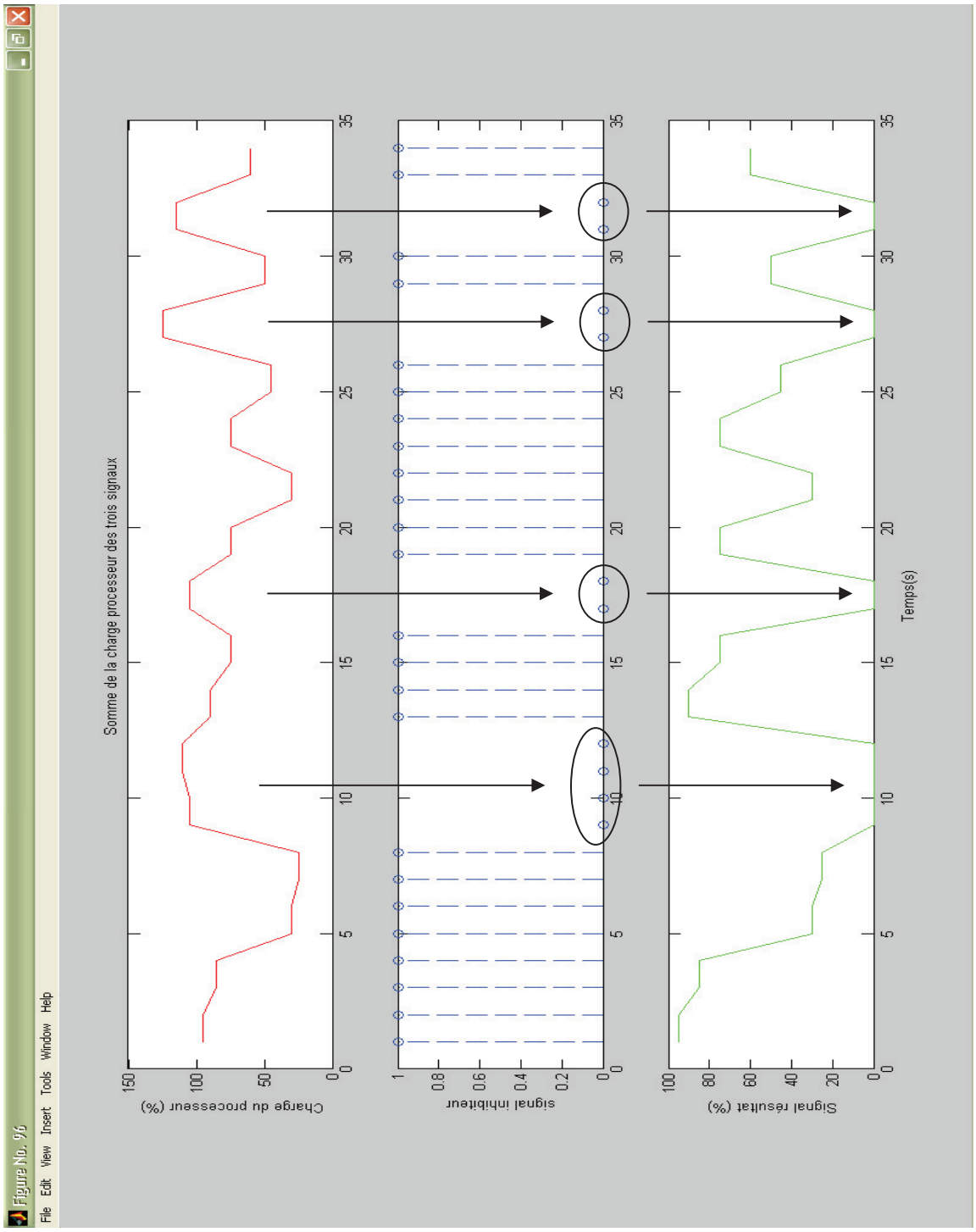

Figure 96 Résultats du contrôle, par l'agent Distributeur 2, de la somme des 3 signaux, correspondant aux 3 modalités, captés par les agents Distributeurs 1et 2 via leurs couches réactives, en fonction du temps

Les informations en CPN-ML sur les différents arcs, au niveau des codes exécutés par les transitions et dans les conditions gardées ainsi que les liens dynamiques (arcs) entre les différents agents constituent les spécifications des actions de raisonnements de l'AE en fonction de ses propres règles qu'il élabore.
L'Annexe 7 donne le rapport de l'analyse automatique réalisée par CPN pour les propriétés comportementales des RPCTS donnés aux les Figure 91 et 92.

6.5 Conclusion du chapitre 6

Nous avons vu, tout au long de ce chapitre qu'il est difficile de concevoir et de bâtir un système multiagent. En effet, la construction de ce type de système comporte toutes les difficultés inhérentes aux systèmes répartis, auxquelles s'ajoute le caractère flexible et sophistiqué des interactions entre agents. À cela s'ajoute le fait que la conception des SMA fait face à l'absence de méthodologie systématique qui permettrait de spécifier et de structurer une application multiagent. L'intérêt d'une modélisation par RPCTS prend ici tout son sens car elle offre un cadre formel dont les avantages restent identiques à ceux déjà cités lors de nos modélisations d'AL de l'interaction multimodale. Le manque d'outils commerciaux pour bâtir des SMA est encore une fois un handicap auquel nous remédions par l'approche que nous proposons. Nous avons décrit tout le travail de modélisation que nous avons entrepris, ainsi que la mise au point de l'organisation de l'AE face aux problèmes des qualités de l'architecture logicielle. Le modèle de l'AE sous forme de RPCTS constitue donc une représentation stratégique car elle tient compte des différents éléments de la structure architecturale de l'AE (tableau noir, différents agents, différents liens, différentes données et leurs types, etc.) et met en avant son comportement pour l'exécution du profil de qualités choisi. La compréhension de ces mécanismes nous permet de mieux appréhender comment l'AE peut gérer de façon autonome les scénarios de reconfiguration sur une AL multimodale. Enfin, il nous permet de simuler la structure de l'AE pour un profil de qualité choisi avant la phase de développement de l'application, en plus de pouvoir faire une vérification automatique

des propriétés comportementales de l'AE greffé sur l'AL multimodale, avant cette même phase de développement.

CONCLUSION

Cette thèse s'inscrit dans le cadre d'une approche sur des modèles multiagent d'architectures logicielles (AL) multimodales et sur la validation automatique de ces modèles aux niveaux statique et dynamique.

En effet, dans une première étape de recherche, nous avons proposé des paradigmes architecturaux modélisés par les réseaux de Petri colorés stochastiques (RPCTS) et dédiés à la problématique des applications multimodales. Nous avons unifié ces paradigmes en des AL nouvelles et génériques. Ces AL sont génériques au sens qu'elles prennent en compte tous les aspects fonctionnels de l'interaction multimodale, à tous les niveaux d'abstractions et qu'elles permettent l'ajout ou la suppression de tout nouveau média d'entrée et/ou de sortie sans modification importante de la structure architecturale. Nous avons présenté comment raffiner ces AL pour tenir compte des contraintes et besoins fonctionnels des concepteurs. Enfin, nous avons montré les intérêts du choix de notre spécification et modélisation par les RPCTS en répondant aux insuffisances des AL multimodales de la littérature. Un des principaux intérêts de ce type de modélisation, par réseaux de Petri, est la possibilité de valider automatiquement et par des simulations l'architecture quand à ses propriétés comportementales. Ainsi, il nous est possible de valider que tous les états de l'AL qui correspondent à l'exécution de commandes multimodales sont effectivement réalisables sous différentes contraintes (temporelles, stochastiques, de complémentarité structurelle et grammaticale de l'information). Cette approche méthodologique de conception d'AL multimodale a, par la suite, été illustrée par deux exemples didactiques d'applications de jeux multimodaux.

Dans une seconde étape de recherche, nous avons apporté des solutions sous forme de modèles dynamiques d'un Agent Expert (AE) permettant le monitorage et la reconfiguration dynamique de l'AL multimodale proposée précédemment. Pour rester dans le prolongement cohérent de la première étape de notre travail de recherche, nous avons également montré comment modéliser avec les RPCTS cet AE. De même, nous avons exposé comment cet AE vient se connecter à une AL multimodale, toujours grâce

aux modèles en réseaux de Petri. Cet agent prend en charge la réalisation de scénarios pour surveiller et maintenir un profil de qualité logiciel pour une AL multimodale. Aussi, nous avons redéfini le concept de scénario de telle sorte qu'il puisse être reproduit par un modèle en réseau de Petri tenant compte d'attributs de qualité logiciels multimodaux. Nous avons également montré, par l'exemple, comment construire le modèle de l'AE pour qu'il puisse surveiller maintenir et reconfigurer l'AL afin de vérifier les propriétés de caractéristiques d'attributs de qualités inscrites elles-mêmes dans un modèle en réseau de Petri, du profil de qualité multimodal.

Les perspectives offertes par notre recherche seraient l'exploration d'autres modèles de profils de qualité que celui proposé dans ce travail. Par ailleurs, l'aspect, la forme et la structure du comportement intelligent élaboré ici, dans le modèle agent, pourraient être raffinés et modifiés afin de permettre à l'AE de faire de l'apprentissage incrémental avant de reconfigurer l'AL. Par ailleurs, des études expérimentales sur l'emploi de différents dispositifs dans l'interaction multimodale par l'utilisateur moyen ou par l'utilisateur spécialisé (enfant, handicapé, expert, etc.), sur les temps de réponses de ces mêmes dispositifs et sur le comportement des utilisateurs face à l'alternative de l'interaction multimodale en entrée comme en sortie, offriraient des données qui permettraient de mieux caractériser l'AL et le champ d'action de l'AE pour une reconfiguration plus intelligente de l'application. Une autre perspective de nos travaux serait de mener une analyse théorique sur les paradigmes architecturaux modélisés en RPCTS et de faire la preuve formelle de leurs propriétés comportementales.

ANNEXE 1

Les grammaires

Cette annexe est un bref rappel sur la notion de grammaire et sur la hiérarchie de Chomsky (Abeillé 2000)

1.1 Notion de grammaire

Une **grammaire** est définie comme un quadruplet :

$$G = <V_T, V_A, S, R>$$

où V_T représente le **vocabulaire terminal,**
V_A représente le **vocabulaire auxiliaire,**
S représente l'**axiome** de la grammaire ($S \in V_A$),
R représente un ensemble de **règles de réécriture.**

Chaque règle est de la forme :

$$\varsigma \rightarrow \sigma$$

où $\varsigma \in (V_A \cup V_T)^*$ et $\sigma \in (V_A \cup V_T)^*$,
ς est appelée **partie gauche** de la règle,
σ est appelée **partie droite** de la règle.

Ces notions sont définies avec les conventions suivantes :

a. $(X)^*$: zéro ou plusieurs occurrences de X;
b. $(X)^+$: une ou plusieurs occurrences de X;

c. (X) ? : une ou zéro occurrence de X.

Voici un exemple de grammaire G= <V_T, V_A, P, R> comprenant 3 règles :

P → GN v GN

GN → npropre

GN → det nom

Les éléments de V_T correspondent aux catégories des mots de la phrase à analyser (i.e. ici : v, npropre, det, nom), les éléments de V_A correspondent aux catégories pouvant apparaître comme partie gauche d'une règle de réécriture (ici : P, GN), l'axiome correspond à la structure que nous cherchons à analyser (le plus souvent il s'agit donc d'une phrase : P).

Par convention, les éléments du vocabulaire terminal débutent en général par une lettre minuscule tandis que ceux du vocabulaire auxiliaire débutent par une majuscule.

1.2 Hiérarchie de Chomsky ("Structures syntaxiques" 1957)

Cette présentation de la hiérarchie de Chomsky est extraite de (Abeillé 2000).

La hiérarchie de Chomsky comprend 4 classes de grammaires, définies en fonction de la forme de leurs règles de réécriture :

a. Les grammaires régulières, également appelées grammaires de type 3 ou grammaires de Kleene. Toutes les règles d'une telle grammaire sont de la forme : $A \rightarrow a$ ou $A \rightarrow a\ B$ où $a \in V_T$, $A \in V_A$, $B \in V_A$.
b. Les grammaires hors contextes ou indépendantes du contexte (CFG), également appelées grammaires de type 2. Toutes les règles d'une telle grammaire sont de la forme : $X \rightarrow \varsigma$ où $X \in V_A$ et $\varsigma \in (V_T \cup V_A)^*$ (donc longueur (X)=1).

c. Les grammaires contextuelles (CSG), également appelées grammaires de type 1. Toutes les règles d'une telle grammaire sont de la forme : $aXb \rightarrow a \varsigma b$ où $a \in (V_A \cup V_T)^*$, $b \in (V_A \cup V_T)^*$, $X \in V_A$ et $\varsigma \in (V_A \cup V_T)^+$.
d. Les grammaires non contraintes, également appelées grammaires de type 0. Toutes les règles d'une telle grammaire sont de la forme $\varsigma \rightarrow \sigma$. La seule contrainte est que ς ne soit pas vide et comprenne au moins un élément de V_A.

Il s'agit d'une hiérarchie car ces 4 classes de grammaires sont en relation d'inclusion :

Grammaires régulières $\subset$ Grammaires hors contexte $\subset$ Grammaires contextuelles $\subset$ Grammaires non contraintes

Ainsi, toute grammaire régulière appartient à la classe des grammaires hors contexte, à la classe des grammaires contextuelles et à la classe des grammaires non contraintes.

Les grammaires régulières sont les moins puissantes en termes de langages qu'elles permettent de générer et les grammaires non contraintes sont les plus puissantes.

ANNEXE 2

Rappel sur les réseaux de Petri.

Cette Annexe reprend quelques notions de base sur les réseaux de Petri (RP.)

2.1 Les réseaux de Petri

Les réseaux de Petri ont été définis en 1962 par Carl Adam Petri (Petri 1962) dans sa thèse "Kommunikation mit Automaten". Ils permettent, en particulier, de modéliser et d'analyser des systèmes de processus concurrents et parallèles évoluant dans le temps.

2.1.1 Définition du Réseau de Petri

Un réseau de Petri (Reisig 1985; Vidal-Naquet G. 1992) est la donnée d'un ensemble fini de places, d'un ensemble fini de transitions et d'une fonction dite fonction de poids. Ceci définit la structure statique du système. L'état de celui-ci se modélise à l'aide d'un marquage que nous faisons évoluer en franchissant des transitions. Ceci correspond à exécuter les actions qui leurs sont associées. Formellement, un réseau de Petri peut être représenté par un quadruplet $RP = (Pe, Et, W, m_0)$ où :

$Pe = \{p_1, p_2, p_3, \ldots p_n\}$ est l'ensemble des places,

$Et = \{t_1, t_2, t_3, \ldots t_n\}$ est l'ensemble des transitions,

$W : (Pe \times Et) \cup (Et \times Pe) \rightarrow N$ est la fonction de poids et

$m : Pe \rightarrow N$ est la fonction de marquage.

Le marquage initial est donné par m_0.

$m(p) = k$ signifie que la place p contient k marques (jetons).

Nous disons aussi que le marquage de p est k.
$W(p,t)=k$ signifie que la transition t utilise k jetons dans la place p; de façon duale, $W(p,t)=k$ signifie que la transition t génère les jetons dans la place p.

2.1.2 Représentation graphique

Un réseau de Petri peut être vu (Reisig 1985; Vidal-Naquet G. 1992) comme un graphe biparti où :

a) les places sont représentées par des ellipses,
b) et les transitions par des rectangles (ou traits).

Un arc relie une place p à une transition t ssi[37] $W(p,t) \neq 0$.
Un arc relie une transition t à une place p ssi $W(t,p) \neq 0$.

1) Règles de franchissement d'une transition :
Les règles suivantes permettent de représenter l'évolution des systèmes à modéliser
Une transition t est dite franchissable pour le marquage m si pour toute place p telle que $W(p,t) \neq 0$ est marquée d'au moins $W(p,t)$ jetons, c'est-à-dire si

$$\forall p \in Pe, m(p) \geq W(p,t).$$

Nous notons ceci $m(t)$.
Une transition t est franchie en retirant $W(p,t)$ jetons de chaque place p telle que $W(p,t) \neq 0$, et en ajoutant $W(t,p')$ jetons à chaque place p' telle que $W(t,p') \neq 0$, c'est-à-dire si t est franchissable pour m, alors le franchissement de t fait passer le marquage m au marquage m' de la façon suivant :

$$\forall p \in Pe, m'(p) = m(p) - W(p,t) + W(t,p).$$

Nous notons ceci $m(t)m'$.

La notion de franchissement de transitions peut alors être étendue aux séquences de transitions.

2.1.2.1 Définition :

Soit $s = t_1t_2\ldots t_n$ une séquence de transitions. La séquence s est franchissable à partir de m et conduit au marquage m' ce qui sera noté $m(s)m'$ ssi il existe des marquages $m_0 = m_1, \ldots, m_n = m'$ tels que :

$$\forall i : 0 \leq i \leq n-1 : m_i(t_{i\neq 1})m_{i\neq 1}$$

2.1.3 Représentation matricielle

Il est possible de représenter par des matrices la fonction W (Vidal-Naquet G. 1992).
Nous appelons matrice précondition *pré*, la matrice de dimensions (p, n) à coefficients dans l'ensemble des entiers naturels, où p est le nombre de places, n est le nombre de transitions et

$$pré(i,j) = W(p_i,t_j).$$

Nous appelons matrice postcondition la matrice *post* de dimension (p,n) à coefficients dans N définie par

$$post(i,j) = W(t_j,p_i).$$

La matrice $C = post - pré$ est appelée matrice d'incidence.
$pré(i,j)$ indique le nombre de marques que doit contenir la place p_i, pour que la transition t_j soit franchissable.
De façon duale, $post(i,j)$ contient le nombre de marques déposées dans la place p_i à la suite du franchissement de la transition t_j.
Un marquage m sera alors représenté par un vecteur de dimension p à coefficients dans N. Nous le notons $\overline{m}$.

---

[37] Acronyme de 'si et seulement si'.

2.1.3.1 Notations pour la matrice pré.

Nous notons

$$W(.,t_i) = pré(.,i)$$

le i^{ème} vecteur colonne de la matrice *pré* et

$$W(p_j,.) = pré(j,.)$$

le j^{ème} vecteur ligne de la matrice *pré*.

2.1.3.2 Définition du vecteur caractéristique d'une transition

Le vecteur caractéristique (Vidal-Naquet G. 1992) de la transition t_i noté $\bar{t}_i$, est défini comme suit :

$\bar{t}_i(j) = 0$ pour $j \neq i, 1 \leq j \leq l$ où l est le nombre de transitions,

$\bar{t}_i(i) = 1$.

Si $\bar{m} \geq (pré.\bar{t}) = W(.,t)$, où le "." Dans $pré.\bar{t}$ dénote le produit matriciel, alors t est franchissable pour m. Le franchissement de t aboutit au marquage m' défini par : $\bar{m}' = \bar{m} + post\,.\,\bar{t} - pré\,.\,\bar{t} = \bar{m} + C\,.\,\bar{t}$

2.1.3.3 Définition du vecteur caractéristique d'une séquence

Soit s = $t_1 t_2 ... t_n$ une séquence finie (Vidal-Naquet G. 1992) de transitions franchissables (séquence de tir licite) à partir de m. Nous appelons *vecteur caractéristique* de s, le vecteur $\bar{s}$ dont la composante $\bar{s}(i)$ détermine le nombre d'occurrences de la transition t_i dans s. Si $m(s)m'$ alors : $\bar{m}' = \bar{m} + C\,.\,\bar{s}$

2.1.3.4 Notation pour la séquence franchissable.

Soit *t* une transition, nous notons

$\Delta(t) = C \,.\, \bar{t} = (W(t, p_1) - W(p_1, t) - ...W(t, p_n) - W(p, t_n))$.

De plus, si $t_1, t_2, \ldots t_k$ sont des transitions alors

$\Delta(t_1, t_2, ...t_k) = \Delta(t_1) + \Delta(t_2) + ...\Delta(t_k)$.

Nous notons $m(s)$ le fait qu'une séquence *s* (finie ou infinie) soit franchissable à partir de *m*.

2.2 Propriétés des réseaux de Petri

Une fois qu'un réseau de Petri a été défini, nous pouvons toujours l'analyser, pour déterminer ses propriétés. Celles-ci dépendent de la topologie et du marquage initial du réseau. Nous présentons dans cette section certaines propriétés. Nous invitons le lecteur à consulter (Reisig 1985; Vidal-Naquet G. 1992) pour découvrir d'autres propriétés des réseaux de Petri.

2.2.1 Définition d'un réseau k-borné :

Un réseau de Petri est dit *k-borné* (Vidal-Naquet G. 1992) ssi le nombre de jetons dans n'importe quelle place ne peut dépasser *k*.

2.2.1.1 Définition. d'un réseau sauf :

Un réseau de Petri est dit sauf (Vidal-Naquet G. 1992) ssi il est 1-*borné*.
Le concept de réseau *k-borné* est généralement utilisé pour modéliser les systèmes réels dont les ressources sont nécessairement finies. De plus, lorsqu'il s'agit d'un système logique, il est commode de le modéliser avec un réseau *sauf*.

La propriété de monotonie, définie ci-dessous, traduit l'existence de conditions minimales de franchissement d'une transition. Elle est liée au fait que les conditions pour qu'une action soit possible découlent exclusivement de la présence de ressources en nombre suffisant.

2.2.1.2 Propriété de monotonie

Si $m' \geq m$ alors pour toute séquence $s, m(s)m_1 \Rightarrow (\exists m_1' : m'(s)m_1' \; et \; m_1' \geq m_1)$ (Vidal-Naquet G. 1992)

Une méthode naturelle pour examiner un système consiste à étudier tous les états possibles de ce système. S'il est fini, nous pouvons par une étude exhaustive déterminer les propriétés du système.

2.2.1.3 Définition du marquage accessible

Un marquage *m* est dit *accessible* (Vidal-Naquet G. 1992) à partir du marquage m_0 ssi il existe une séquence de tirs telle que

$$m_0(s>m:$$

Nous notons $Acc(RP; m_0)$ l'ensemble des marquages du réseau *RP* accessibles à partir de m_0.

Une des méthodes les plus naturelles pour analyser un réseau de Petri, consiste à le faire évoluer systématiquement à partir du marquage initial, pour déterminer quelles propriétés sont satisfaites ou non, par exploration exhaustive de tous les marquages accessibles. Ceci nous amène à construire un graphe de marquages.

2.2.1.4 Graphe des marquages

Soient un *RP* et m_0 son marquage initial. Le graphe des marquages $GA(RP; m_0)$ est défini par les deux conditions suivantes (Vidal-Naquet G. 1992) :

a. l'ensemble des sommets est l'ensemble $Acc(RP; m_0)$;

b. il existe un arc étiqueté par t de m à m_0 ssi $m(t \succ m'$.

Le graphe des marquages n'est pas toujours constructible. En effet, $Acc(RP;m_0)$ peut être trop grand, voire même infini, lorsque le réseau n'est pas borné. Cependant, il existe une alternative à ce graphe, le graphe de couverture (Reisig 1985), qui n'est autre qu'une "compression" du graphe des marquages. Il permet à chaque marquage accessible d'être soit représenté explicitement par un nœud du graphe, soit "couvert" par un nœud. En premier lieu et avant de donner la définition de ce graphe, il convient de définir ω qui représente un nombre infini.

2.2.1.5 Définition de 'oméga'

Soit $\omega \notin$ N (ensemble des entiers naturels) et $N_\varpi = N \bigcup \{\omega\}$. Les opérations +,- et la relation < sont étendues à N_ϖ de la manière suivante (Vidal-Naquet G. 1992):

pour tout $n \in N$,

$$n < \omega$$
$$n+\omega = \omega+\omega = \omega+n = n$$
$$\omega - n = n$$
$$n - \omega \text{ n'est pas défini.}$$

Nous étendons +, _ et < aux vecteurs de N_ω^m de manière analogue à celle utilisée pour étendre +, _ et < aux vecteurs de N^m. L'addition de deux vecteurs donne un vecteur dont les composantes sont égales à la somme des composantes correspondantes des deux vecteurs. La soustraction est similaire à l'addition à l'exception évidemment que nous soustrayons au lieu d'additionner. Pour la comparaison, nous examinons les composantes correspondantes entre elles : si toutes les composantes du premier vecteur sont inférieures aux composantes correspondantes du deuxième vecteur alors le premier

vecteur est inférieur au deuxième et vice-versa. Si certaines composantes sont inférieures et certaines ne le sont pas alors les deux vecteurs ne sont pas comparables.
Sans nous appesantir sur les détails, nous pouvons maintenant définir le graphe de couverture.

2.2.1.6 Algorithme de construction du graphe de couverture

Soit le réseau de Petri *RP* = (*Pe; Et*; *W*; m_0), comportant *n* places et ayant comme marquage initial m_0. Nous notons *Gc*(*RP*;m_0) = (*Sn*;X) le graphe de couverture de R construit comme suit (Uchihira 1990):

1. *Sn* est un ensemble de nœuds étiquetés par des éléments de N_{ω}^{n}; $Sn := \{m_0\}$.
2. *X* est un ensemble d'arcs (*x; x'*) étiquetés par des éléments de *Et*; $X := \phi$.
3. Répéter l'étape 4 jusqu'à ce que tous les nœuds soient traités.
4. Soit $x = (x_1; x_2; \ldots; x_n)$ un nœud non traité. Pour chaque transition *t* franchis-sable à partir de *x*, faire :

a. Créer un nouveau nœud candidat $x' = (x_1', x_2', \ldots, x_n')$

b. Pour tout *i*, $1 \le i \le n$, poser $x_i' := x_i - W(p_i, t) + W(t, p_i)$.

c. S'il existe un nœud *r* sur un chemin allant de la racine m_0 à *x* tel que $(\forall i : 1 \le i \le n : r_i \le x_i')$ alors, pour chaque *i* tel que $r_i \le x_i'$ poser $x_i' := \omega$

d. Si le nœud *x'* n'appartient pas à *Sn*, alors ajouter le nœud *x'* à *Sn* et l'arc (*x; x'*) étiqueté par *t* à *X*, sinon ajouter seulement l'arc (*x; x'*) étiqueté par *t* à *X*.

2.2.1.7 Remarques

1. Nous pouvons remarquer qu'à l'étape 4.b que nous avons $x_i' = \omega \quad si \quad x_i = \omega$
2. L'algorithme de la construction du graphe de couverture s'arrête toujours, car :

Nous ne pouvons avoir aucune branche de longueur infinie. En effet, d'après le lemme de Karp et Miller (Karp 1969), toute suite infinie de vecteurs formés d'entiers positifs ou nuls $v_1, v_2, ..., v_k, ...$ est telle qu'elle contient au moins deux éléments (en fait une infinité de couples) v_i et v_j avec $i < j$ telles que $v_i \leq v_j$.

Le nombre de branches est fini, car pour chaque marquage le nombre de transitions franchissables est fini (inférieur ou égal au nombre de transitions du RP).

Nous suggérons au lecteur intéressé par plus d'aspects théoriques sur les RP de consulter les références citées dans cette annexe.

ANNEXE 3

Page des déclarations globales nécessaires à la simulation du réseau des Figures 25 et 26 par l'outil CPN-Tools

Cette page de déclaration est écrite dans le langage CPN-ML(University of Aarhus 2006).

```
(*GLOBAL DECLARATION PAGE*)
(*=========================*)
(*Proximity time between two events*)
(*=========================*)
val ProxyTime = 100;
 (*Average Inter_arrival*)
(*================*)
val ClickArrival = ref 1.0;
val WordArrival = ref 10.0;
 (* Color sets *)
(*========*)
color Int =int;
color Attribute = product Int * Int * Int;
 (* Color sets for mouse event *)
(************************)
color MouseClick = with ClickEvent;
color ClickxAttribute = product MouseClick * Attribute timed;
color ME = with me timed;
(* Color sets for speech*)
(******************)
color WordSaid = with Word;
color WordxAttribute = product WordSaid * Attribute * Int timed;
```

```
color WE = with we timed;
 (* Color sets for fusion event*)
(************************)
color FusedEvents = with Fused;
color FAttrib =product Int * Int;
color FusedxAttribute = product FusedEvents * FAttrib * FAttrib * FAttrib timed ;
 (*Variables*)
(*=======*)
var Word1, Word2,Word3 :WordSaid;
var n, Fn, Fm, Fm1, Fm2, Fm3, m, m1, m2, m3, p, Fp : Int;
 (* Variables for Time *)
(******************)
var NextClick, NextWord, ArrivalTimeC, ArrivalTimeW1, ArrivalTimeW2,
        ArrivalTimeW3, ArrivalTimeWp, ArrivalTimeW: Int ;
 (* Variables for word labels *)
(************************)
var wt, wtype, wtype1, wtype2, wtype3 : Int;
 (* Functions *)
 (*========*)
fun intTime()=IntInf.toInt(time());
fun round x =floor(x+0.5);
fun ExpLaw x= round(exponential(x));
```

ANNEXE 4

Pages des déclarations globales nécessaires à la simulation des réseaux de Petri du chapitre 4

Ces pages de déclaration sont écrites dans le langage CPN-ML(University of Aarhus 2006).

```
(*TUX_XMEN GAME' DECLARATION PAGE*)
(*==================================*)
color   INT = int;
color   Commande = with commande timed;
color   ActionFusion = product Commande * INT * Commande;
color   ActionFusionCompletee = product Commande * INT * Commande * Commande
        * INT;
color   Action = product Commande * INT;
color   ActionCompletee = product Commande * INT * Commande;
var     typeC, typeI : INT;
var     commandeC, commandeV, commandeS : Commande;

 (*MEMORY GAME' DECLARATION PAGE*)
(*==================================*)
color INT = int;
color   Event = with event timed;
color   Objet = with objet timed;
color   ObjetAttribut= product Objet * INT * INT *INT timed;
color   SE = with se timed;
```

```
color   ObjetS = with objetS timed;
color   ObjetAttributFusion = product Objet*ObjetS*INT*INT;
var     n, nSeq, nSeqC, nSeqFusion, lab, nCap, nTraite, label , numero1, numero2,
numero3, label1, label2, label3, Temps1, Temps2, Temps3, TempsRecon,
TempsSuivant, ClicSuivant, tempsMots, tempsClic: INT;
var     objAttribut:ObjetAttribut;
fun     intTime()=IntInf.toInt(time());
fun     round x =floor(x+0.5);
fun     LoiNormale (x,y) = round(normal(x,y));
fun     discExp x = round(exponential(x));
val     max =4;
val     InterArrivee = ref 5.0;
val     InterArriveeC = ref 5.0;
```

ANNEXE 5

Listage des rapports de vérification des propriétés des réseaux de Petri du jeu 'Tux_XMEN' et du jeu de mémoire, du chapitre 4, générés automatiquement par l'outil CPN-Tools.

```
CPN Tools state space report for:
C:\ TuxXMEN.cpn
Report generated: Wed Jul 25 00:45:07
2007

Statistics
------------------------------------------------
----------------------
  Occurrence Graph
    Nodes:  17701
    Arcs:   50483
    Secs:   300
    Status: Partial

  Scc Graph (Srongly connected
components of the state space graph)

    Nodes:  16376
    Arcs:   47300
    Secs:   12

 Boundedness Properties
------------------------------------------------
----------------------

  Best Integers Bounds          Upper
Lower
  Application'ActionExecutee 1    2
0
  Application'FileClavier 1       9          0
  Application'FileCommandeC 1     4
0
  Application'FileCommandeV 1     3
0
  Application'FileItems 1         2          0
  Application'FileRetrait 1       1          0
  Application'FileSouris 1        10         0
  Application'FileType 1          10         0
  Application'FileVocale 1        9          0
  Application'ItemRetire 1        0          0
  Application'ItemTrouvee 1       1          0

  Best Upper Multi-set Bounds
  Application'ActionExecutee 1
                2`(commande,3)
  Application'FileClavier 1
                9`commande
  Application'FileCommandeC 1
                4`commande
  Application'FileCommandeV 1

3`(commande,1)++3`(commande,2)++3
`(commande,3)
  Application'FileItems 1
                2`(commande,1)
  Application'FileRetrait 1

1`(commande,2,commande)
  Application'FileSouris 1
                10`commande
  Application'FileType 1
                6`1++5`2++7`3
  Application'FileVocale 1
                9`commande
  Application'ItemRetire 1
                empty
  Application'ItemTrouvee 1

1`(commande,1,commande)
  Best Lower Multi-set Bounds
  Application'ActionExecutee 1
                empty
```

Application'FileClavier 1
empty
Application'FileCommandeC 1
empty
Application'FileCommandeV 1
empty
Application'FileItems 1
empty
Application'FileRetrait 1
empty
Application'FileSouris 1
empty
Application'FileType 1
empty
Application'FileVocale 1
empty
Application'ItemRetire 1
empty
Application'ItemTrouvee 1
empty

Home Properties

Home Markings: Initial Marking is not a home marking

Liveness Properties

Dead Markings: 8918 [9999, 9998, 9997, 9996, 9995,...]

Dead Transitions Instances:
Application'RetirerItem 1
Application'SortieEcran2 1
Live Transitions Instances: None

Fairness Properties

Application'ActionSpeciale 1 Fair
Application'AjoutItem 1 No Fairness
Application'EntreeClavier 1 No Fairness
Application'EntreeSouris 1 No Fairness
Application'EntreeVocale 1 Impartial
Application'GenerateurType 1 Impartial
Application'ReconnaissanceClavier 1 No Fairness
Application'ReconnaissanceVocale 1 Impartial
Application'RetirerItem 1 Fair
Application'RetraitItem 1 No Fairness
Application'SortieEcran1 1 No Fairness
Application'SortieEcran2 1 Fair
Application'SortieSonore1 1No Fairness
Application'SortieSonore2 1 Fair
Application'TrouverItem 1 Fair

CPN Tools state space report for:
C:\ MemoryGt.cpn
Report generated: Wed Jul 25 00:45:07 2007

Statistics

State Space
Nodes: 83753
Arcs: 312849
Secs: 300
Status: Partial

Scc Graph (Srongly connected components of the state space graph)

Nodes: 61521
Arcs: 253423
Secs: 49

Boundedness Properties

Best Integer Bounds

	Upper	Lower
Jeu'Chosen1 1	2	1
Jeu'Chosen11 1	2	1
Jeu'Chosen2 1	2	0
Jeu'Count0 1	1	1
Jeu'Count 1	1	1
Jeu'Count1 1	1	1
Jeu'Count11 1	1	1
Jeu'Count2 1	1	1
Jeu'Count22 1	1	1
Jeu'Count3 1	1	1
Jeu'Counter 1	1	1
Jeu'Displays 1	2	0
Jeu'Wait 1	1	0
Jeu'Wait1 1	1	0
Jeu'Wait2 1	1	0
Jeu'WaitGam0 1	1	0
Jeu'WaitGam1 1	1	0
Jeu'WaitGam2 1	1	0
Jeu'WaitGam3 1	1	0
Jeu'WaitGame 1	1	0

Best Upper Multi-set Bounds

Jeu'Chosen1 1 1`(b,1)++
1`(b,2)++
1`(b,3)++
1`(b,4)++
1`(b,5)++
1`(b,6)++
1`(b,7)++
1`(b,8)++
1`(b,18)
Jeu'Chosen11 1 1`(b,1)++
1`(b,2)++
1`(b,3)++
1`(b,4)++
1`(b,5)++
1`(b,6)++
1`(b,7)++
1`(b,8)++
1`(b,18)
Jeu'Chosen2 1 2`(b,1)++
2`(b,2)++
2`(b,3)++
2`(b,4)++
2`(b,5)++
2`(b,6)++
2`(b,7)++
2`(b,8)
Jeu'Count0 1 1`0++
1`1++
1`2++
1`3++
1`4++
1`5++
1`6++
1`7++
1`8++
1`9++
1`10++
1`11++

```
1`12++
1`13++
1`14++
1`15++
1`16++
1`17
    Jeu'Count 1         1`0++
1`1++
1`2++
1`3++
1`4++
1`5++
1`6++
1`7++
1`8++
1`9++
1`10++
1`11++
1`12++
1`13++
1`14++
1`15++
1`16++
1`17
    Jeu'Count1 1        1`0++
1`1++
1`2++
1`3++
1`4++
1`5++
1`6++
1`7++
1`8++
1`9++
1`10++
1`11++
1`12++
1`13++
1`14++
1`15++
1`16++
1`17
    Jeu'Count11 1       1`0++
1`1++
```

```
1`2++
1`3++
1`4++
1`5++
1`6++
1`7++
1`8++
1`9++
1`10++
1`11++
1`12++
1`13++
1`14++
1`15++
1`16++
1`17
    Jeu'Count2 1        1`0++
1`1++
1`2++
1`3++
1`4++
1`5++
1`6++
1`7++
1`8++
1`9++
1`10++
1`11++
1`12++
1`13++
1`14++
1`15++
1`16++
1`17
    Jeu'Count22 1       1`0++
1`1++
1`2++
1`3++
1`4++
1`5++
1`6++
1`7++
1`8++
1`9++
```

```
1`10++
1`11++
1`12++
1`13++
1`14++
1`15++
1`16++
1`17
   Jeu'Count3 1       1`0++
1`1++
1`2++
1`3++
1`4++
1`5++
1`6++
1`7++
1`8++
1`9++
1`10++
1`11++
1`12++
1`13++
1`14++
1`15++
1`16++
1`17
   Jeu'Counter 1      1`0++
1`1++
1`2++
1`3++
1`4++
1`5++
1`6++
1`7++
1`8++
1`9++
1`10++
1`11++
1`12++
1`13++
1`14++
1`15++
1`16++
1`17
```

```
 Jeu'Displays 1     2`(b,1)++
2`(b,2)++
2`(b,3)++
2`(b,4)++
2`(b,5)++
2`(b,6)++
2`(b,7)++
2`(b,8)
   Jeu'Wait 1         1`(a,0)++
1`(b,0)
   Jeu'Wait1 1        1`(a,0)++
1`(b,0)
   Jeu'Wait2 1        1`(a,0)++
1`(b,0)
   Jeu'WaitGam0 1     1`1
   Jeu'WaitGam1 1     1`1
   Jeu'WaitGam2 1     1`1
   Jeu'WaitGam3 1     1`1
   Jeu'WaitGame 1     1`1

 Best Lower Multi-set Bounds
-------------------------------------
   Jeu'Chosen1 1      1`(b,18)
   Jeu'Chosen11 1     1`(b,18)
   Jeu'Chosen2 1      empty
   Jeu'Count0 1       empty
   Jeu'Count 1        empty
   Jeu'Count1 1       empty
   Jeu'Count11 1      empty
   Jeu'Count2 1       empty
   Jeu'Count22 1      empty
   Jeu'Count3 1       empty
   Jeu'Counter 1      empty
   Jeu'Displays 1     empty
   Jeu'Wait 1         empty
   Jeu'Wait1 1        empty
   Jeu'Wait2 1        empty
   Jeu'WaitGam0 1     empty
   Jeu'WaitGam1 1     empty
   Jeu'WaitGam2 1     empty
   Jeu'WaitGam3 1     empty
   Jeu'WaitGame 1     empty
```

Home Properties

Home Markings
Initial Marking is not a home marking

Liveness Properties

Dead Markings
3014 [9994,9987,9980,9971,9966,...]

Dead Transition Instances
None

Live Transition Instances
None

Fairness Properties

Jeu'Cancel1 1 Just
Jeu'Cancel2 1 Just
Jeu'Flip 1 Just
Jeu'Game 1 Fair
Jeu'HideDisp 1 Fair
Jeu'NewSet 1 Just
Jeu'NewSet1 1 Just
Jeu'NewSet2 1 Just
Jeu'QuitGam0 1 Just
Jeu'QuitGam1 1 Just
Jeu'QuitGam2 1 Just
Jeu'QuitGame 1 Just
Jeu'SelectC1 1 Just
Jeu'SelectC2 1 Just

ANNEXE 6

Page des déclarations globales nécessaires à la simulation du réseau des Figures 91 et 92 par l'outil CPN-Tools

Cette page de déclaration est écrite dans le langage CPN-ML (University of Aarhus 2006).

```
colset INT = int;
colset T=with t timed;
colset R =with r timed;
colset SAT= product INT*INT*INT timed;
colset SAt= product T * R* INT * INT timed;
colset Sti=INT timed;
colset Tri =product INT *INT timed;
var ,mg:Msg;
var a,b,c,va,va1,va2,h1,h2,h3:Sti;
val InterArrivee = ref  100.0;
var dt,ddec1,ddec2,ddec4,dt0,dt1,dt2,dt7,dt8,dt9,dt10,dt01,dt02,dt03,dt22,dt33,
dt11,dt44,dt61,dt62,dt63,dt71,ts3,ts1,ts2,g1,g2,g3,j1,j2,j3,jj1,jj2,jj3,k1,k2,k3,
dec1,dec2,dec1a,dec1b,dec,Ts,tn,tnn,ttu,tuu,tv,tvv,tw,tww,t0,tu,t0u,t0v,t0w,t00,t01,t02,
t03,t1,t11,t111,t2,t222,t3,t333,t4,t444,t41,t44s,t444s, t5,t555,t6,t61,t611,t61a,t611a,t61b,
t611b,t666,t7,t71,t8,t88,t10,t11,t22,t33,r61,tx23,ty23,m,n,tp1,p1,tp2,p2,pt0,pt1,pt2:INT ;
fun intTime( ) = IntInf.toInt(time( ));
fun round x = floor(x+0.5);
fun Exp x = round(exponential(x));
```

ANNEXE 7

Listage du rapport de vérification des propriétés des réseaux de Petri de l'AE donnés aux Figures 91 et 92, au chapitre 6, généré automatiquement par l'outil CPN-Tools.

```
CPN Tools state space
report for:
C:\ArchiAE16.cpn
Report generated: Wed
Jul 25 15:50:04 2007

 Statistics
----------------------------


  State Space
    Nodes:  22918
    Arcs:   38003
    Secs:   300
    Status: Partial

  Scc Graph (Srongly
connected components
of the state space graph)

    Nodes:  22918
    Arcs:   38003
    Secs:   4


 Boundedness
Properties
----------------------------

Best Integer
Bounds   Upper  Lower

    AgentD1'BaseConn1
1    2         0
    AgentD1'FAct14 1
1         0
    AgentD1'FilAct22 1
1         0
    AgentD1'InhibM1 1
1         0
    AgentD1'InhibR1 1
1         0
    AgentD1'InhibW1 1
1         0
    AgentD1'MesgI1 1
1         0
    AgentD1'PercepM1
1    1          0
    AgentD1'PercepW1
1    1          0
    AgentD1'PerceptR1
1    1          0
    AgentD1'RapportT1
1    3          0
    AgentD1'SPercept 1
1         0
    AgentD1'Senarios 1
2         0
    AgentD1'SigneMM1
1    1          1
    AgentD1'Stimuli11 1
1         0
    AgentD1'Stimuli12 1
1         0
    AgentD1'Stimuli13 1
1         0
    AgentD1'directs1 1
1         0
    AgentD1'fAct11 1
1         0
    AgentD1'fAct12 1
1         0
    AgentD1'fAct13 1
1         0
    AgentD1'filAct21 1
1         0
    AgentD1'filAct23 1
1         0
    AgentD1'inhibET2 1
1         0
    AgentD1'inhibM 1
1         0
    AgentD1'inhibR 1
1         0
    AgentD1'inhibR2 1
1         0
    AgentD1'inhibW 1
1         0
    AgentD1'messageR
1    2          0
    AgentD1'msgI2 1
1         0
    AgentD1'percepC1 1
1         0
    AgentD1'percepC2 1
1         0
    AgentD1'plans1 1
1         0
    AgentD1'raportT2 1
1         0
    AgentD1'signal1 1
2         1
    AgentD1'signal2 1
2         1
    AgentD1'signal3 1
2         1
    AgentD1'tableauN 1
1         1
```

AgentD1'tache1 1
3 0
AgentD1'tache2 1
1 0
AgentD1'tamp11 1
1 0
AgentD1'tamp12 1
1 0
AgentD1'tamp13 1
1 0
AgentD1'tamp14 1
3 0
AgentD1'tamp15 1
1 0
AgentD1'tamp16 1
1 0
AgentD1'tamp22 1
1 0
AgentD1'verrou1 1
1 0
AgentD2'BaseCon2
1 2 0
AgentD2'CPUmod 1
1 0
AgentD2'CPUrate 1
1 0
AgentD2'Directi2 1
1 0
AgentD2'FACt11 1
1 0
AgentD2'FACt12 1
1 0
AgentD2'FACt13 1
1 0
AgentD2'FAcT11 1
1 0
AgentD2'FAcT12 1
1 0
AgentD2'FAcT13 1
1 0
AgentD2'FAct11 1
1 0
AgentD2'FAct12 1
1 0
AgentD2'FAct13 1
1 0
AgentD2'FilAct21 1
1 0
AgentD2'FilAct22 1
1 0
AgentD2'FilAct23 1
1 0
AgentD2'InhibET2 1
1 0
AgentD2'InhibR2 1
1 0
AgentD2'InhibW2 1
1 0
AgentD2'MesgI2 1
1 0
AgentD2'MessageR
1 2 0
AgentD2'PerceET2 1
1 0
AgentD2'PercepC1 1
1 0
AgentD2'PercepC2 1
1 0
AgentD2'PercepR2 1
1 0
AgentD2'PercepW2
1 1 0
AgentD2'Plans1 1
1 0
AgentD2'Plans2 1
1 0
AgentD2'RaportT2 1
1 0
AgentD2'Scenario 1
2 0
AgentD2'Sig1 1
2 0
AgentD2'Sig2 1
2 0
AgentD2'Sig3 1
2 0
AgentD2'Signal1 1
2 1
AgentD2'Signal2 1
2 1
AgentD2'Signal3 1
2 1
AgentD2'Stimul21 1
1 0
AgentD2'Stimul22 1
1 0
AgentD2'Stimul23 1
1 0
AgentD2'TableauN 1
1 1
AgentD2'Tache1 1
3 0
AgentD2'Tache2 1
1 0
AgentD2'Tamp12 1
1 0
AgentD2'Tamp14 1
3 0
AgentD2'Tamp15 1
1 0
AgentD2'Tamp16 1
1 0
AgentD2'Tamp22 1
1 0
AgentD2'inhibet 1
1 0
AgentD2'inhibm 1
1 0
AgentD2'inhibr 1
1 0
AgentD2'tamp21 1
1 0
AgentD2'tamp23 1
1 0
AgentD2'tamp24 1
1 0
AgentD2'verrou2 1
1 0

Best Upper Multi-set Bounds

AgentD1'BaseConn1 1 1`(t,r,1,0)++ 1`(t,r,1,2)++ 1`(t,r,2,0)++ 1`(t,r,2,2)++ 2`(t,r,3,0)++ 1`(t,r,3,1)++ 1`(t,r,3,2)

AgentD1'FAct14 1 1`0++ 1`1++ 1`2++ 1`3

AgentD1'FilAct22 1 1`(6,3)++ 1`(11,3)++ 1`(15,3)++ 1`(16,3)++ 1`(17,3)++ 1`(19,3)++ 1`(20,3)++ 1`(21,3)++ 1`(22,3)++ 1`(23,3)++ 1`(24,3)++ 1`(25,3)++ 1`(27,3)++ 1`(28,3)

AgentD1'InhibM1 1 1`0++ 1`1

AgentD1'InhibR1 1 1`0++ 1`1

AgentD1'InhibW1 1 1`0++ 1`1

AgentD1'MesgI1 1 1`0++ 1`1++ 1`2++ 1`3

AgentD1'PercepM1 1 1`(1,3,2)++ 1`(1,4,1)++ 1`(1,4,3)++ 1`(1,4,4)++ 1`(1,4,5)++ 1`(1,5,1)++ 1`(1,5,3)++ 1`(1,5,4)++ 1`(1,5,5)++ 1`(1,6,1)++ 1`(1,6,2)++ 1`(1,6,3)++ 1`(1,6,5)++ 1`(1,7,1)++ 1`(1,7,2)++ 1`(1,7,4)++ 1`(1,7,5)++ 1`(1,8,2)++ 1`(1,8,3)++ 1`(1,8,4)++ 1`(1,8,5)++ 1`(1,9,2)++ 1`(1,9,3)++ 1`(1,9,4)++ 1`(1,9,5)++ 1`(1,10,1)++ 1`(1,10,2)++ 1`(1,10,3)++ 1`(1,10,5)++ 1`(1,11,1)++ 1`(1,11,2)++ 1`(1,11,3)++ 1`(1,11,5)++ 1`(1,12,1)++ 1`(1,12,5)++ 1`(1,13,4)++ 1`(2,1,4)++ 1`(2,3,2)++ 1`(2,3,4)++ 1`(2,4,2)++ 1`(2,4,3)++ 1`(2,4,4)++ 1`(2,5,1)++ 1`(2,5,2)++ 1`(2,5,3)++ 1`(2,5,4)++ 1`(2,5,5)++ 1`(2,6,2)++ 1`(2,6,3)++ 1`(2,6,4)++ 1`(2,6,5)++ 1`(2,7,1)++ 1`(2,7,2)++ 1`(2,7,3)++ 1`(2,7,4)++ 1`(2,7,5)++ 1`(2,8,2)++ 1`(2,8,3)++ 1`(2,8,5)++ 1`(2,9,1)++ 1`(2,9,3)++ 1`(2,9,4)++ 1`(2,10,1)++ 1`(2,10,2)++ 1`(2,10,4)++ 1`(2,10,5)++ 1`(2,11,1)++ 1`(2,11,2)++ 1`(2,11,3)++ 1`(2,11,4)++ 1`(2,11,5)++ 1`(2,12,1)++ 1`(2,12,3)++ 1`(2,13,1)++ 1`(2,13,2)++ 1`(2,15,1)++ 1`(3,4,4)++ 1`(3,4,5)++ 1`(3,5,5)++ 1`(3,6,1)++ 1`(3,6,3)++ 1`(3,6,4)++ 1`(3,6,5)++ 1`(3,7,1)++ 1`(3,7,2)++ 1`(3,7,3)++ 1`(3,7,4)++ 1`(3,7,5)++ 1`(3,8,2)++

```
1`(3,8,3)++
1`(3,8,4)++
1`(3,8,5)++
1`(3,9,1)++
1`(3,9,2)++
1`(3,9,3)++
1`(3,9,4)++
1`(3,9,5)++
1`(3,10,1)++
1`(3,10,2)++
1`(3,10,3)++
1`(3,10,5)++
1`(3,11,2)++
1`(3,11,3)++
1`(3,11,4)++
1`(3,12,1)++
1`(3,12,3)++
1`(4,2,2)++
1`(4,4,1)++
1`(4,4,3)++
1`(4,5,2)++
1`(4,5,3)++
1`(4,5,4)++
1`(4,5,5)++
1`(4,6,1)++
1`(4,6,2)++
1`(4,6,3)++
1`(4,6,5)++
1`(4,7,1)++
1`(4,7,3)++
1`(4,7,4)++
1`(4,7,5)++
1`(4,8,1)++
1`(4,8,2)++
1`(4,8,3)++
1`(4,8,4)++
1`(4,8,5)++
1`(4,9,1)++
1`(4,9,2)++
1`(4,9,3)++
1`(4,9,4)++
1`(4,9,5)++
1`(4,10,1)++
1`(4,10,3)++
1`(4,10,4)++
1`(4,11,1)++
1`(4,11,2)++
1`(4,11,3)++
1`(4,12,2)++
1`(4,12,3)++
1`(4,12,5)++
1`(5,3,1)++
1`(5,3,4)++
1`(5,4,1)++
1`(5,4,3)++
1`(5,5,3)++
1`(5,5,5)++
1`(5,6,1)++
1`(5,6,2)++
1`(5,6,4)++
1`(5,6,5)++
1`(5,7,1)++
1`(5,7,2)++
1`(5,7,3)++
1`(5,7,4)++
1`(5,8,1)++
1`(5,8,2)++
1`(5,8,3)++
1`(5,8,4)++
1`(5,8,5)++
1`(5,9,1)++
1`(5,9,2)++
1`(5,9,3)++
1`(5,9,4)++
1`(5,9,5)++
1`(5,10,1)++
1`(5,10,2)++
1`(5,10,4)++
1`(5,10,5)++
1`(5,11,1)++
1`(5,11,4)++
1`(5,11,5)++
1`(5,12,1)++
1`(5,12,2)++
1`(5,12,5)++
1`(5,13,2)++
1`(6,1,1)++
1`(6,4,1)++
1`(6,4,2)++
1`(6,4,3)++
1`(6,4,4)++
1`(6,5,1)++
1`(6,5,4)++
1`(6,5,5)++
1`(6,6,1)++
1`(6,6,3)++
1`(6,6,4)++
1`(6,7,1)++
1`(6,7,2)++
1`(6,7,4)++
1`(6,8,1)++
1`(6,8,2)++
1`(6,8,3)++
1`(6,8,4)++
1`(6,8,5)++
1`(6,9,1)++
1`(6,9,2)++
1`(6,9,3)++
1`(6,9,4)++
1`(6,9,5)++
1`(6,10,1)++
1`(6,10,2)++
1`(6,10,3)++
1`(6,11,1)++
1`(6,11,2)++
1`(6,11,3)++
1`(6,11,5)++
1`(6,12,2)++
1`(6,13,2)++
1`(7,4,2)++
1`(7,5,2)++
1`(7,5,5)++
1`(7,6,3)++
1`(7,6,4)++
1`(7,6,5)++
1`(7,7,1)++
1`(7,7,2)++
1`(7,7,3)++
1`(7,7,4)++
1`(7,8,1)++
1`(7,8,2)++
1`(7,8,3)++
```

1`(7,8,4)++
1`(7,8,5)++
1`(7,9,1)++
1`(7,9,2)++
1`(7,9,3)++
1`(7,9,4)++
1`(7,9,5)++
1`(7,10,2)++
1`(7,10,3)++
1`(7,10,4)++
1`(7,10,5)++
1`(7,11,1)++
1`(7,11,2)++
1`(7,11,3)++
1`(7,11,4)++
1`(7,11,5)++
1`(7,12,3)++
1`(7,13,2)++
1`(7,14,1)++
1`(7,15,2)++
1`(8,1,4)++
1`(8,2,3)++
1`(8,4,1)++
1`(8,4,2)++
1`(8,4,4)++
1`(8,4,5)++
1`(8,5,2)++
1`(8,5,5)++
1`(8,6,1)++
1`(8,6,2)++
1`(8,6,3)++
1`(8,7,2)++
1`(8,7,3)++
1`(8,7,4)++
1`(8,7,5)++
1`(8,8,1)++
1`(8,8,2)++
1`(8,8,4)++
1`(8,8,5)++
1`(8,9,1)++
1`(8,9,2)++
1`(8,9,4)++
1`(8,9,5)++
1`(8,10,1)++
1`(8,10,2)++
1`(8,10,3)++
1`(8,10,4)++
1`(8,10,5)++
1`(8,11,1)++
1`(8,11,3)++
1`(8,11,4)++
1`(8,12,5)++
1`(8,13,4)++
1`(8,15,3)++
1`(9,2,4)++
1`(9,3,1)++
1`(9,3,3)++
1`(9,4,1)++
1`(9,4,3)++
1`(9,4,5)++
1`(9,5,2)++
1`(9,5,3)++
1`(9,5,5)++
1`(9,6,2)++
1`(9,6,4)++
1`(9,6,5)++
1`(9,7,1)++
1`(9,7,2)++
1`(9,7,3)++
1`(9,7,5)++
1`(9,8,1)++
1`(9,8,2)++
1`(9,8,3)++
1`(9,8,4)++
1`(9,8,5)++
1`(9,9,1)++
1`(9,9,3)++
1`(9,9,4)++
1`(9,9,5)++
1`(9,10,1)++
1`(9,10,2)++
1`(9,10,3)++
1`(9,10,4)++
1`(9,10,5)++
1`(9,11,1)++
1`(9,11,4)++
1`(9,11,5)++
1`(9,15,3)++
1`(10,2,1)++
1`(10,2,3)++
1`(10,4,2)++
1`(10,4,4)++
1`(10,5,4)++
1`(10,5,5)++
1`(10,6,1)++
1`(10,6,2)++
1`(10,6,3)++
1`(10,6,4)++
1`(10,6,5)++
1`(10,7,1)++
1`(10,7,4)++
1`(10,8,1)++
1`(10,8,2)++
1`(10,8,3)++
1`(10,8,4)++
1`(10,8,5)++
1`(10,9,1)++
1`(10,9,2)++
1`(10,9,3)++
1`(10,9,4)++
1`(10,9,5)++
1`(10,10,1)++
1`(10,10,2)++
1`(10,10,5)++
1`(10,11,2)++
1`(10,11,3)++
1`(10,11,4)++
1`(10,12,4)++
1`(10,12,5)++
1`(10,13,4)

AgentD1'PercepW1 1
1`(1,3,4)++
1`(1,5,1)++
1`(1,5,5)++
1`(1,6,1)++
1`(1,6,2)++
1`(1,6,3)++
1`(1,6,4)++
1`(1,6,5)++
1`(1,7,2)++
1`(1,7,3)++

```
1`(1,7,4)++
1`(1,7,5)++
1`(1,8,1)++
1`(1,8,2)++
1`(1,8,3)++
1`(1,8,4)++
1`(1,8,5)++
1`(1,9,2)++
1`(1,9,3)++
1`(1,9,4)++
1`(1,9,5)++
1`(1,10,1)++
1`(1,10,3)++
1`(1,10,4)++
1`(1,10,5)++
1`(1,11,1)++
1`(1,11,3)++
1`(1,11,4)++
1`(1,11,5)++
1`(1,12,1)++
1`(1,12,2)++
1`(1,12,5)++
1`(1,13,5)++
1`(1,14,2)++
1`(2,2,2)++
1`(2,5,3)++
1`(2,5,5)++
1`(2,6,2)++
1`(2,6,3)++
1`(2,6,4)++
1`(2,6,5)++
1`(2,7,1)++
1`(2,7,2)++
1`(2,7,3)++
1`(2,7,4)++
1`(2,7,5)++
1`(2,8,1)++
1`(2,8,2)++
1`(2,8,3)++
1`(2,8,4)++
1`(2,9,1)++
1`(2,9,2)++
1`(2,9,3)++
1`(2,9,4)++
1`(2,9,5)++
1`(2,10,1)++
1`(2,10,2)++
1`(2,10,3)++
1`(2,10,4)++
1`(2,11,2)++
1`(2,11,3)++
1`(2,12,4)++
1`(2,13,1)++
1`(2,13,4)++
1`(3,3,5)++
1`(3,4,1)++
1`(3,4,3)++
1`(3,6,1)++
1`(3,6,2)++
1`(3,6,3)++
1`(3,6,4)++
1`(3,6,5)++
1`(3,7,1)++
1`(3,7,2)++
1`(3,7,3)++
1`(3,7,5)++
1`(3,8,1)++
1`(3,8,2)++
1`(3,8,3)++
1`(3,8,4)++
1`(3,8,5)++
1`(3,9,2)++
1`(3,9,4)++
1`(3,9,5)++
1`(3,10,1)++
1`(3,10,2)++
1`(3,10,3)++
1`(3,10,4)++
1`(3,10,5)++
1`(3,11,2)++
1`(3,11,3)++
1`(3,11,4)++
1`(3,12,1)++
1`(3,12,2)++
1`(3,12,4)++
1`(3,13,3)++
1`(3,13,4)++
1`(3,14,2)++
1`(3,14,3)++
1`(4,1,3)++
1`(4,3,2)++
1`(4,5,4)++
1`(4,5,5)++
1`(4,6,1)++
1`(4,6,2)++
1`(4,6,3)++
1`(4,6,4)++
1`(4,6,5)++
1`(4,7,2)++
1`(4,7,3)++
1`(4,7,4)++
1`(4,7,5)++
1`(4,8,1)++
1`(4,8,3)++
1`(4,8,4)++
1`(4,8,5)++
1`(4,9,1)++
1`(4,9,2)++
1`(4,9,5)++
1`(4,10,1)++
1`(4,10,2)++
1`(4,10,3)++
1`(4,10,4)++
1`(4,10,5)++
1`(4,11,2)++
1`(4,11,3)++
1`(4,11,5)++
1`(4,12,2)++
1`(4,13,2)++
1`(4,13,3)++
1`(5,1,4)++
1`(5,4,2)++
1`(5,5,5)++
1`(5,6,1)++
1`(5,6,3)++
1`(5,6,4)++
1`(5,6,5)++
1`(5,7,1)++
1`(5,7,2)++
1`(5,7,3)++
1`(5,7,4)++
1`(5,8,1)++
```

1`(5,8,2)++
1`(5,8,4)++
1`(5,8,5)++
1`(5,9,1)++
1`(5,9,2)++
1`(5,9,3)++
1`(5,9,4)++
1`(5,9,5)++
1`(5,10,1)++
1`(5,10,2)++
1`(5,10,3)++
1`(5,10,4)++
1`(5,10,5)++
1`(5,11,2)++
1`(5,11,4)++
1`(5,12,2)++
1`(5,12,3)++
1`(5,13,2)++
1`(5,13,3)++
1`(5,14,2)++
1`(6,5,2)++
1`(6,5,4)++
1`(6,5,5)++
1`(6,6,1)++
1`(6,6,2)++
1`(6,6,3)++
1`(6,6,4)++
1`(6,6,5)++
1`(6,7,1)++
1`(6,7,2)++
1`(6,7,3)++
1`(6,7,4)++
1`(6,8,1)++
1`(6,8,2)++
1`(6,8,4)++
1`(6,8,5)++
1`(6,9,1)++
1`(6,9,2)++
1`(6,9,3)++
1`(6,9,4)++
1`(6,9,5)++
1`(6,10,1)++
1`(6,10,2)++
1`(6,10,3)++
1`(6,10,4)++
1`(6,10,5)++
1`(6,11,2)++
1`(6,12,1)++
1`(6,12,2)++
1`(6,12,5)++
1`(6,13,1)++
1`(6,14,3)++
1`(7,1,1)++
1`(7,2,3)++
1`(7,4,3)++
1`(7,5,1)++
1`(7,5,2)++
1`(7,5,3)++
1`(7,5,4)++
1`(7,6,1)++
1`(7,6,2)++
1`(7,6,3)++
1`(7,6,5)++
1`(7,7,1)++
1`(7,7,3)++
1`(7,7,4)++
1`(7,7,5)++
1`(7,8,2)++
1`(7,8,3)++
1`(7,8,5)++
1`(7,9,2)++
1`(7,9,3)++
1`(7,9,4)++
1`(7,10,1)++
1`(7,10,2)++
1`(7,10,4)++
1`(7,10,5)++
1`(7,11,1)++
1`(7,11,3)++
1`(7,12,2)++
1`(7,12,3)++
1`(7,12,4)++
1`(7,13,4)++
1`(7,14,1)++
1`(8,1,3)++
1`(8,2,3)++
1`(8,3,5)++
1`(8,4,2)++
1`(8,5,3)++
1`(8,5,5)++
1`(8,6,1)++
1`(8,6,2)++
1`(8,6,4)++
1`(8,6,5)++
1`(8,7,1)++
1`(8,7,2)++
1`(8,7,4)++
1`(8,7,5)++
1`(8,8,1)++
1`(8,8,2)++
1`(8,8,3)++
1`(8,8,4)++
1`(8,8,5)++
1`(8,9,1)++
1`(8,9,2)++
1`(8,9,3)++
1`(8,9,4)++
1`(8,9,5)++
1`(8,10,1)++
1`(8,10,2)++
1`(8,10,3)++
1`(8,10,4)++
1`(8,11,1)++
1`(8,11,5)++
1`(8,12,2)++
1`(8,13,2)++
1`(8,13,3)++
1`(8,14,3)++
1`(9,5,1)++
1`(9,5,3)++
1`(9,5,4)++
1`(9,6,1)++
1`(9,6,2)++
1`(9,6,3)++
1`(9,6,4)++
1`(9,6,5)++
1`(9,7,1)++
1`(9,7,2)++
1`(9,7,4)++
1`(9,8,1)++
1`(9,8,2)++
1`(9,8,3)++

1`(9,8,4)++
1`(9,8,5)++
1`(9,9,2)++
1`(9,9,4)++
1`(9,9,5)++
1`(9,10,2)++
1`(9,10,3)++
1`(9,10,4)++
1`(9,10,5)++
1`(9,11,2)++
1`(9,11,3)++
1`(9,11,4)++
1`(9,12,1)++
1`(9,12,3)++
1`(9,13,2)++
1`(9,13,4)++
1`(9,14,5)++
1`(10,4,2)++
1`(10,5,2)++
1`(10,6,1)++
1`(10,6,3)++
1`(10,6,4)++
1`(10,6,5)++
1`(10,7,1)++
1`(10,7,3)++
1`(10,7,5)++
1`(10,8,2)++
1`(10,8,3)++
1`(10,8,4)++
1`(10,8,5)++
1`(10,9,1)++
1`(10,9,2)++
1`(10,9,3)++
1`(10,9,4)++
1`(10,9,5)++
1`(10,10,1)++
1`(10,10,3)++
1`(10,10,4)++
1`(10,10,5)++
1`(10,11,2)++
1`(10,11,3)++
1`(10,11,4)++
1`(10,12,3)++
1`(10,14,2)

AgentD1'PerceptR1 1
1`(1,4,1)++
1`(1,4,3)++
1`(1,5,1)++
1`(1,5,2)++
1`(1,6,2)++
1`(1,6,4)++
1`(1,6,5)++
1`(1,7,1)++
1`(1,7,5)++
1`(1,8,1)++
1`(1,8,2)++
1`(1,8,3)++
1`(1,8,4)++
1`(1,8,5)++
1`(1,9,1)++
1`(1,9,2)++
1`(1,9,3)++
1`(1,9,4)++
1`(1,9,5)++
1`(1,10,1)++
1`(1,10,2)++
1`(1,10,4)++
1`(1,10,5)++
1`(1,11,1)++
1`(1,11,2)++
1`(1,11,3)++
1`(1,11,4)++
1`(1,11,5)++
1`(1,12,1)++
1`(1,12,2)++
1`(1,12,3)++
1`(1,12,4)++
1`(1,12,5)++
1`(1,13,1)++
1`(1,13,2)++
1`(1,13,3)++
1`(1,13,5)++
1`(1,14,1)++
1`(2,3,5)++
1`(2,4,4)++
1`(2,5,3)++
1`(2,5,4)++

1`(2,6,2)++
1`(2,6,3)++
1`(2,6,4)++
1`(2,7,1)++
1`(2,7,3)++
1`(2,7,5)++
1`(2,8,1)++
1`(2,8,2)++
1`(2,8,3)++
1`(2,8,4)++
1`(2,8,5)++
1`(2,9,1)++
1`(2,9,2)++
1`(2,9,3)++
1`(2,9,5)++
1`(2,10,1)++
1`(2,10,2)++
1`(2,10,3)++
1`(2,10,4)++
1`(2,10,5)++
1`(2,11,1)++
1`(2,11,2)++
1`(2,11,3)++
1`(2,11,4)++
1`(2,11,5)++
1`(2,12,1)++
1`(2,12,3)++
1`(2,12,4)++
1`(2,12,5)++
1`(2,13,2)++
1`(2,13,5)++
1`(3,2,1)++
1`(3,3,5)++
1`(3,4,1)++
1`(3,5,1)++
1`(3,5,4)++
1`(3,6,1)++
1`(3,6,2)++
1`(3,6,3)++
1`(3,6,5)++
1`(3,7,1)++
1`(3,7,2)++
1`(3,7,5)++
1`(3,8,1)++

1`(3,8,2)++
1`(3,8,3)++
1`(3,8,4)++
1`(3,9,1)++
1`(3,9,2)++
1`(3,9,4)++
1`(3,9,5)++
1`(3,10,1)++
1`(3,10,2)++
1`(3,10,4)++
1`(3,11,1)++
1`(3,11,2)++
1`(3,11,3)++
1`(3,11,5)++
1`(3,12,1)++
1`(3,12,2)++
1`(3,12,3)++
1`(3,12,5)++
1`(3,13,1)++
1`(3,13,2)++
1`(4,3,4)++
1`(4,5,4)++
1`(4,5,5)++
1`(4,6,1)++
1`(4,6,4)++
1`(4,6,5)++
1`(4,7,1)++
1`(4,7,3)++
1`(4,7,5)++
1`(4,8,2)++
1`(4,8,3)++
1`(4,8,4)++
1`(4,8,5)++
1`(4,9,1)++
1`(4,9,2)++
1`(4,9,3)++
1`(4,9,4)++
1`(4,9,5)++
1`(4,10,1)++
1`(4,10,2)++
1`(4,10,3)++
1`(4,10,5)++
1`(4,11,1)++
1`(4,11,2)++
1`(4,11,4)++
1`(4,12,1)++
1`(4,12,2)++
1`(4,12,3)++
1`(4,12,4)++
1`(4,12,5)++
1`(4,13,3)++
1`(4,13,4)++
1`(4,13,5)++
1`(5,2,1)++
1`(5,2,5)++
1`(5,4,1)++
1`(5,4,2)++
1`(5,4,4)++
1`(5,5,2)++
1`(5,5,4)++
1`(5,6,1)++
1`(5,6,2)++
1`(5,6,3)++
1`(5,6,5)++
1`(5,7,1)++
1`(5,7,2)++
1`(5,7,4)++
1`(5,7,5)++
1`(5,8,2)++
1`(5,8,4)++
1`(5,8,5)++
1`(5,9,2)++
1`(5,9,3)++
1`(5,9,4)++
1`(5,9,5)++
1`(5,10,1)++
1`(5,10,2)++
1`(5,10,4)++
1`(5,10,5)++
1`(5,11,3)++
1`(5,11,4)++
1`(5,12,1)++
1`(5,12,2)++
1`(5,12,3)++
1`(5,12,4)++
1`(5,12,5)++
1`(5,13,1)++
1`(5,13,3)++
1`(5,13,4)++
1`(5,13,5)++
1`(5,14,4)++
1`(6,1,1)++
1`(6,2,3)++
1`(6,4,2)++
1`(6,4,4)++
1`(6,4,5)++
1`(6,6,2)++
1`(6,6,3)++
1`(6,6,4)++
1`(6,6,5)++
1`(6,7,1)++
1`(6,7,2)++
1`(6,7,3)++
1`(6,7,4)++
1`(6,7,5)++
1`(6,8,1)++
1`(6,8,3)++
1`(6,8,4)++
1`(6,9,1)++
1`(6,9,2)++
1`(6,9,3)++
1`(6,9,4)++
1`(6,9,5)++
1`(6,10,1)++
1`(6,10,2)++
1`(6,10,3)++
1`(6,10,4)++
1`(6,10,5)++
1`(6,11,1)++
1`(6,11,2)++
1`(6,11,3)++
1`(6,11,4)++
1`(6,11,5)++
1`(6,12,1)++
1`(6,12,3)++
1`(6,12,4)++
1`(6,13,1)++
1`(6,13,2)++
1`(6,13,4)++
1`(7,2,5)++
1`(7,4,2)++
1`(7,4,3)++

1`(7,4,5)++
1`(7,5,3)++
1`(7,6,5)++
1`(7,7,4)++
1`(7,8,2)++
1`(7,8,3)++
1`(7,8,4)++
1`(7,8,5)++
1`(7,9,2)++
1`(7,9,3)++
1`(7,9,4)++
1`(7,10,1)++
1`(7,10,2)++
1`(7,10,3)++
1`(7,10,4)++
1`(7,11,1)++
1`(7,11,2)++
1`(7,11,3)++
1`(7,12,1)++
1`(7,12,2)++
1`(7,12,3)++
1`(7,12,4)++
1`(7,12,5)++
1`(7,13,1)++
1`(7,13,3)++
1`(7,13,4)++
1`(7,13,5)++
1`(7,14,1)++
1`(8,1,4)++
1`(8,3,2)++
1`(8,4,1)++
1`(8,4,2)++
1`(8,4,3)++
1`(8,6,1)++
1`(8,6,2)++
1`(8,6,4)++
1`(8,7,1)++
1`(8,7,2)++
1`(8,7,4)++
1`(8,7,5)++
1`(8,8,1)++
1`(8,8,2)++
1`(8,8,3)++
1`(8,8,4)++
1`(8,8,5)++
1`(8,9,1)++
1`(8,9,2)++
1`(8,9,3)++
1`(8,9,5)++
1`(8,10,1)++
1`(8,10,2)++
1`(8,10,3)++
1`(8,10,4)++
1`(8,10,5)++
1`(8,11,1)++
1`(8,11,2)++
1`(8,11,3)++
1`(8,11,5)++
1`(8,12,3)++
1`(8,12,4)++
1`(8,13,3)++
1`(8,13,4)++
1`(9,4,2)++
1`(9,4,4)++
1`(9,5,5)++
1`(9,6,3)++
1`(9,6,5)++
1`(9,7,1)++
1`(9,7,2)++
1`(9,7,4)++
1`(9,7,5)++
1`(9,8,1)++
1`(9,8,2)++
1`(9,8,3)++
1`(9,8,5)++
1`(9,9,2)++
1`(9,9,4)++
1`(9,9,5)++
1`(9,10,1)++
1`(9,10,2)++
1`(9,10,3)++
1`(9,10,4)++
1`(9,11,2)++
1`(9,11,4)++
1`(9,12,1)++
1`(9,12,3)++
1`(9,12,4)++
1`(9,13,1)++
1`(9,13,3)++
1`(9,13,4)++
1`(9,13,5)++
1`(9,14,3)++
1`(10,3,2)++
1`(10,5,1)++
1`(10,5,3)++
1`(10,6,1)++
1`(10,6,4)++
1`(10,6,5)++
1`(10,7,1)++
1`(10,7,3)++
1`(10,7,4)++
1`(10,7,5)++
1`(10,8,1)++
1`(10,8,2)++
1`(10,8,5)++
1`(10,9,1)++
1`(10,9,2)++
1`(10,9,3)++
1`(10,9,5)++
1`(10,10,1)++
1`(10,10,2)++
1`(10,10,3)++
1`(10,10,4)++
1`(10,10,5)++
1`(10,11,1)++
1`(10,11,2)++
1`(10,11,3)++
1`(10,11,4)++
1`(10,11,5)++
1`(10,12,2)++
1`(10,12,3)++
1`(10,12,5)++
1`(10,13,1)++
1`(10,13,2)++
1`(10,13,3)++
1`(10,13,4)

AgentD1'RapportT1 1
3`(0,0)++
3`(1,2)++
3`(2,1)++
3`(2,3)++

3`(3,5)

AgentD1'SPercept 1
1`(2,1)++
1`(2,2)++
1`(2,3)++
1`(3,5)

AgentD1'Senarios 1
2`(t,r,1,3)++
2`(t,r,1,4)++
2`(t,r,1,8)++
2`(t,r,1,13)++
2`(t,r,1,20)++
2`(t,r,1,24)++
2`(t,r,1,36)++
2`(t,r,1,43)++
2`(t,r,1,63)++
2`(t,r,1,80)++
2`(t,r,1,113)++
2`(t,r,1,191)++
2`(t,r,1,201)++
2`(t,r,1,206)++
2`(t,r,1,217)++
2`(t,r,1,226)++
2`(t,r,1,232)++
2`(t,r,1,461)++
2`(t,r,2,2)++
2`(t,r,2,16)++
2`(t,r,2,21)++
2`(t,r,2,28)++
2`(t,r,2,29)++
2`(t,r,2,32)++
2`(t,r,2,33)++
2`(t,r,2,55)++
2`(t,r,2,79)++
2`(t,r,2,90)++
2`(t,r,2,94)++
2`(t,r,2,111)++
2`(t,r,2,125)++
2`(t,r,2,143)++
2`(t,r,2,149)++
2`(t,r,2,305)++
2`(t,r,2,636)++
2`(t,r,3,25)++
2`(t,r,3,28)++
2`(t,r,3,31)++
2`(t,r,3,34)++
2`(t,r,3,36)++
2`(t,r,3,48)++
2`(t,r,3,49)++
2`(t,r,3,58)++
2`(t,r,3,67)++
2`(t,r,3,69)++
2`(t,r,3,79)++
2`(t,r,3,89)++
2`(t,r,3,93)++
2`(t,r,3,107)++
2`(t,r,3,110)++
2`(t,r,3,115)++
2`(t,r,3,117)++
2`(t,r,3,128)++
2`(t,r,3,176)++
2`(t,r,3,316)++
2`(t,r,3,336)++
2`(t,r,3,451)++
2`(t,r,3,548)

AgentD1'SigneMM1 1
1`(0,1)++
1`(1,1)

AgentD1'Stimuli11 1
1`2++
1`3++
1`8++
1`13++
1`16++
1`21++
1`24++
1`25++
1`28++
1`31++
1`32++
1`33++
1`36++
1`43++
1`48++
1`55++
1`58++
1`63++
1`69++
1`79++
1`89++
1`93++
1`107++
1`113++
1`115++
1`125++
1`128++
1`143++
1`149++
1`191++
1`206++
1`217++
1`226++
1`305++
1`336++
1`451++
1`461++
1`548++
1`636

AgentD1'Stimuli12 1
1`2++
1`3++
1`8++
1`13++
1`16++
1`21++
1`24++
1`25++
1`28++
1`31++
1`32++
1`33++
1`36++
1`43++
1`48++
1`55++
1`58++
1`63++

1`69++
1`79++
1`89++
1`93++
1`107++
1`113++
1`115++
1`125++
1`128++
1`143++
1`149++
1`191++
1`206++
1`217++
1`226++
1`305++
1`336++
1`451++
1`461++
1`548++
1`636

AgentD1'Stimuli13 1
1`2++
1`3++
1`8++
1`13++
1`16++
1`21++
1`24++
1`25++
1`28++
1`31++
1`32++
1`33++
1`36++
1`43++
1`48++
1`55++
1`58++
1`63++
1`69++
1`79++
1`89++
1`93++
1`107++
1`113++
1`115++
1`125++
1`128++
1`143++
1`149++
1`191++
1`206++
1`217++
1`226++
1`305++
1`336++
1`451++
1`461++
1`548++
1`636

AgentD1'directs1 1
1`(t,r,1,0)++
1`(t,r,1,2)++
1`(t,r,1,3)++
1`(t,r,2,0)++
1`(t,r,2,2)++
1`(t,r,3,0)++
1`(t,r,3,1)++
1`(t,r,3,2)++
1`(t,r,3,3)

AgentD1'fAct11 1
1`0++
1`1

AgentD1'fAct12 1
1`0++
1`1

AgentD1'fAct13 1
1`0++
1`1

AgentD1'filAct21 1
1`(6,3)++
1`(11,3)++
1`(15,3)++
1`(16,3)++
1`(17,3)++
1`(19,3)++
1`(20,3)++
1`(21,3)++
1`(22,3)++
1`(23,3)++
1`(24,3)++
1`(25,3)++
1`(27,3)++
1`(28,3)

AgentD1'filAct23 1
1`(6,3)++
1`(11,3)++
1`(15,3)++
1`(16,3)++
1`(17,3)++
1`(19,3)++
1`(20,3)++
1`(21,0)++
1`(22,0)++
1`(23,0)++
1`(24,0)++
1`(25,0)++
1`(27,0)++
1`(28,0)

AgentD1'inhibET2 1
1`(3,6)++
1`(3,11)++
1`(3,15)++
1`(3,16)++
1`(3,17)++
1`(3,19)++
1`(3,20)++
1`(3,21)++
1`(3,22)++
1`(3,23)++
1`(3,24)++
1`(3,25)++
1`(3,27)++

1`(3,28)

AgentD1'inhibM 1
1`0++
1`1
AgentD1'inhibR 1
1`0++
1`1

AgentD1'inhibR2 1
1`(3,6)++
1`(3,11)++
1`(3,15)++
1`(3,16)++
1`(3,17)++
1`(3,19)++
1`(3,20)++
1`(3,21)++
1`(3,22)++
1`(3,23)++
1`(3,24)++
1`(3,25)++
1`(3,27)++
1`(3,28)

AgentD1'inhibW 1
1`0++
1`1

AgentD1'messageR
1 2`r

AgentD1'msgI2 1
1`(3,6)++
1`(3,11)++
1`(3,15)++
1`(3,16)++
1`(3,17)++
1`(3,19)++
1`(3,20)++
1`(3,21)++
1`(3,22)++
1`(3,23)++
1`(3,24)++
1`(3,25)++
1`(3,27)++
1`(3,28)

AgentD1'percepC1 1
1`(1,1)

AgentD1'percepC2 1
1`0++
1`1++
1`2++
1`3++
1`4++
1`5++
1`6++
1`7++
1`8++
1`9++
1`10++
1`11++
1`12++
1`13++
1`14++
1`15++
1`16++
1`17++
1`18++
1`19++
1`20++
1`21++
1`22++
1`23++
1`24++
1`25++
1`26++
1`27++
1`28++
1`29++
1`30++
1`31++
1`32++
1`33++
1`34++
1`35++
1`36++
1`37++
1`38++
1`39++
1`40++
1`41++
1`42++
1`43++
1`44++
1`45++
1`46++
1`47++
1`48++
1`49++
1`50++
1`51++
1`52++
1`53++
1`54++
1`55++
1`56++
1`57++
1`58++
1`59++
1`60++
1`61++
1`62++
1`63++
1`64++
1`65++
1`66++
1`67++
1`68++
1`69++
1`70++
1`71++
1`72++
1`73++
1`74++
1`75++
1`76++
1`77++
1`78++
1`79++

1`80++
1`81++
1`82++
1`83++
1`84++
1`85++
1`86++
1`87++
1`88++
1`89++
1`90++
1`91++
1`92++
1`93++
1`94++
1`95++
1`96++
1`97++
1`98++
1`99++
1`100++
1`101++
1`102++
1`103++
1`104++
1`105++
1`106++
1`107++
1`108++
1`109++
1`110++
1`111++
1`112++
1`113++
1`114++
1`115++
1`116++
1`117++
1`118++
1`119++
1`120++
1`121++
1`122++
1`123++
1`124++
1`125++
1`126++
1`127++
1`128++
1`129++
1`130++
1`131++
1`132++
1`133++
1`134++
1`135++
1`136++
1`137++
1`138++
1`139++
1`140++
1`141++
1`142++
1`143++
1`145++
1`146++
1`147++
1`148++
1`149++
1`150++
1`152++
1`153++
1`154++
1`155++
1`156++
1`157++
1`158++
1`159++
1`160++
1`162++
1`163++
1`164++
1`165++
1`167++
1`169++
1`170++
1`172++
1`173++
1`174++
1`177++
1`178++
1`179++
1`180++
1`181++
1`182++
1`183++
1`184++
1`185++
1`186++
1`187++
1`188++
1`189++
1`190++
1`192++
1`197++
1`199++
1`200++
1`201++
1`203++
1`204++
1`205++
1`206++
1`207++
1`208++
1`210++
1`211++
1`212++
1`213++
1`214++
1`215++
1`216++
1`217++
1`218++
1`221++
1`222++
1`224++
1`226++
1`227++
1`228++
1`229++
1`230++
1`231++

1`233++
1`235++
1`236++
1`237++
1`238++
1`239++
1`241++
1`244++
1`247++
1`249++
1`251++
1`253++
1`255++
1`258++
1`264++
1`265++
1`267++
1`268++
1`269++
1`278++
1`280++
1`282++
1`286++
1`289++
1`291++
1`292++
1`299++
1`300++
1`301++
1`306++
1`308++
1`309++
1`311++
1`313++
1`318++
1`320++
1`328++
1`329++
1`331++
1`337++
1`338++
1`340++
1`341++
1`342++

1`356++
1`357++
1`363++
1`364++
1`366++
1`368++
1`374++
1`384++
1`385++
1`388++
1`394++
1`413++
1`415++
1`422++
1`428++
1`437++
1`466++
1`468++
1`472++
1`507++
1`518++
1`533++
1`539++
1`544++
1`591++
1`887++
1`1287

AgentD1'plans1 1
1`(t,r,1,3)++
1`(t,r,1,4)++
1`(t,r,1,8)++
1`(t,r,1,13)++
1`(t,r,1,20)++
1`(t,r,1,24)++
1`(t,r,1,36)++
1`(t,r,1,43)++
1`(t,r,1,63)++
1`(t,r,1,80)++
1`(t,r,1,113)++
1`(t,r,1,191)++
1`(t,r,1,201)++
1`(t,r,1,206)++
1`(t,r,1,217)++

1`(t,r,1,226)++
1`(t,r,1,232)++
1`(t,r,1,461)++
1`(t,r,2,2)++
1`(t,r,2,16)++
1`(t,r,2,21)++
1`(t,r,2,28)++
1`(t,r,2,29)++
1`(t,r,2,32)++
1`(t,r,2,33)++
1`(t,r,2,55)++
1`(t,r,2,79)++
1`(t,r,2,90)++
1`(t,r,2,94)++
1`(t,r,2,111)++
1`(t,r,2,125)++
1`(t,r,2,143)++
1`(t,r,2,149)++
1`(t,r,2,305)++
1`(t,r,2,636)++
1`(t,r,3,25)++
1`(t,r,3,28)++
1`(t,r,3,31)++
1`(t,r,3,34)++
1`(t,r,3,36)++
1`(t,r,3,48)++
1`(t,r,3,49)++
1`(t,r,3,58)++
1`(t,r,3,67)++
1`(t,r,3,69)++
1`(t,r,3,79)++
1`(t,r,3,89)++
1`(t,r,3,93)++
1`(t,r,3,107)++
1`(t,r,3,110)++
1`(t,r,3,115)++
1`(t,r,3,117)++
1`(t,r,3,128)++
1`(t,r,3,176)++
1`(t,r,3,316)++
1`(t,r,3,336)++
1`(t,r,3,451)++
1`(t,r,3,548)

AgentD1'raportT2 1
1`6++
1`11++
1`15++
1`16++
1`17++
1`19++
1`20++
1`21++
1`22++
1`23++
1`24++
1`25++
1`27++
1`28

AgentD1'signal1 1
1`(0,0,0)++
2`(1,3,2)++
2`(1,4,1)++
2`(1,4,3)++
2`(1,4,4)++
2`(1,4,5)++
2`(1,5,1)++
2`(1,5,3)++
2`(1,5,4)++
2`(1,5,5)++
2`(1,6,1)++
2`(1,6,2)++
2`(1,6,3)++
2`(1,6,5)++
2`(1,7,1)++
2`(1,7,2)++
2`(1,7,4)++
2`(1,7,5)++
2`(1,8,2)++
2`(1,8,3)++
2`(1,8,4)++
2`(1,8,5)++
2`(1,9,2)++
2`(1,9,3)++
2`(1,9,4)++
2`(1,9,5)++
2`(1,10,1)++
2`(1,10,2)++
2`(1,10,3)++
2`(1,10,5)++
2`(1,11,1)++
2`(1,11,2)++
2`(1,11,3)++
2`(1,11,5)++
2`(1,12,1)++
2`(1,12,5)++
2`(1,13,4)++
2`(2,1,4)++
2`(2,3,2)++
2`(2,3,4)++
2`(2,4,2)++
2`(2,4,3)++
2`(2,4,4)++
2`(2,5,1)++
2`(2,5,2)++
2`(2,5,3)++
2`(2,5,4)++
2`(2,5,5)++
2`(2,6,2)++
2`(2,6,3)++
2`(2,6,4)++
2`(2,6,5)++
2`(2,7,1)++
2`(2,7,2)++
2`(2,7,3)++
2`(2,7,4)++
2`(2,7,5)++
2`(2,8,2)++
2`(2,8,3)++
2`(2,8,5)++
2`(2,9,1)++
2`(2,9,3)++
2`(2,9,4)++
2`(2,10,1)++
2`(2,10,2)++
2`(2,10,4)++
2`(2,10,5)++
2`(2,11,1)++
2`(2,11,2)++
2`(2,11,3)++
2`(2,11,4)++
2`(2,11,5)++
2`(2,12,1)++
2`(2,12,3)++
2`(2,13,1)++
2`(2,13,2)++
2`(2,15,1)++
2`(3,4,4)++
2`(3,4,5)++
2`(3,5,5)++
2`(3,6,1)++
2`(3,6,3)++
2`(3,6,4)++
2`(3,6,5)++
2`(3,7,1)++
2`(3,7,2)++
2`(3,7,3)++
2`(3,7,4)++
2`(3,7,5)++
2`(3,8,2)++
2`(3,8,3)++
2`(3,8,4)++
2`(3,8,5)++
2`(3,9,1)++
2`(3,9,2)++
2`(3,9,3)++
2`(3,9,4)++
2`(3,9,5)++
2`(3,10,1)++
2`(3,10,2)++
2`(3,10,3)++
2`(3,10,5)++
2`(3,11,2)++
2`(3,11,3)++
2`(3,11,4)++
2`(3,12,1)++
2`(3,12,3)++
2`(4,2,2)++
2`(4,4,1)++
2`(4,4,3)++
2`(4,5,2)++
2`(4,5,3)++
2`(4,5,4)++
2`(4,5,5)++
2`(4,6,1)++

2`(4,6,2)++
2`(4,6,3)++
2`(4,6,5)++
2`(4,7,1)++
2`(4,7,3)++
2`(4,7,4)++
2`(4,7,5)++
2`(4,8,1)++
2`(4,8,2)++
2`(4,8,3)++
2`(4,8,4)++
2`(4,8,5)++
2`(4,9,1)++
2`(4,9,2)++
2`(4,9,3)++
2`(4,9,4)++
2`(4,9,5)++
2`(4,10,1)++
2`(4,10,3)++
2`(4,10,4)++
2`(4,11,1)++
2`(4,11,2)++
2`(4,11,3)++
2`(4,12,2)++
2`(4,12,3)++
2`(4,12,5)++
2`(5,3,1)++
2`(5,3,4)++
2`(5,4,1)++
2`(5,4,3)++
2`(5,5,3)++
2`(5,5,5)++
2`(5,6,1)++
2`(5,6,2)++
2`(5,6,4)++
2`(5,6,5)++
2`(5,7,1)++
2`(5,7,2)++
2`(5,7,3)++
2`(5,7,4)++
2`(5,8,1)++
2`(5,8,2)++
2`(5,8,3)++
2`(5,8,4)++
2`(5,8,5)++
2`(5,9,1)++
2`(5,9,2)++
2`(5,9,3)++
2`(5,9,4)++
2`(5,9,5)++
2`(5,10,1)++
2`(5,10,2)++
2`(5,10,4)++
2`(5,10,5)++
2`(5,11,1)++
2`(5,11,4)++
2`(5,11,5)++
2`(5,12,1)++
2`(5,12,2)++
2`(5,12,5)++
2`(5,13,2)++
2`(6,1,1)++
2`(6,4,1)++
2`(6,4,2)++
2`(6,4,3)++
2`(6,4,4)++
2`(6,5,1)++
2`(6,5,4)++
2`(6,5,5)++
2`(6,6,1)++
2`(6,6,3)++
2`(6,6,4)++
2`(6,7,1)++
2`(6,7,2)++
2`(6,7,4)++
2`(6,8,1)++
2`(6,8,2)++
2`(6,8,3)++
2`(6,8,4)++
2`(6,8,5)++
2`(6,9,1)++
2`(6,9,2)++
2`(6,9,3)++
2`(6,9,4)++
2`(6,9,5)++
2`(6,10,1)++
2`(6,10,2)++
2`(6,10,3)++
2`(6,11,1)++
2`(6,11,2)++
2`(6,11,3)++
2`(6,11,5)++
2`(6,12,2)++
2`(6,13,2)++
2`(7,4,2)++
2`(7,5,2)++
2`(7,5,5)++
2`(7,6,3)++
2`(7,6,4)++
2`(7,6,5)++
2`(7,7,1)++
2`(7,7,2)++
2`(7,7,3)++
2`(7,7,4)++
2`(7,8,1)++
2`(7,8,2)++
2`(7,8,3)++
2`(7,8,4)++
2`(7,8,5)++
2`(7,9,1)++
2`(7,9,2)++
2`(7,9,3)++
2`(7,9,4)++
2`(7,9,5)++
2`(7,10,2)++
2`(7,10,3)++
2`(7,10,4)++
2`(7,10,5)++
2`(7,11,1)++
2`(7,11,2)++
2`(7,11,3)++
2`(7,11,4)++
2`(7,11,5)++
2`(7,12,3)++
2`(7,13,2)++
2`(7,14,1)++
2`(7,15,2)++
2`(8,1,4)++
2`(8,2,3)++
2`(8,4,1)++
2`(8,4,2)++
2`(8,4,4)++

2`(8,4,5)++
2`(8,5,2)++
2`(8,5,5)++
2`(8,6,1)++
2`(8,6,2)++
2`(8,6,3)++
2`(8,7,2)++
2`(8,7,3)++
2`(8,7,4)++
2`(8,7,5)++
2`(8,8,1)++
2`(8,8,2)++
2`(8,8,4)++
2`(8,8,5)++
2`(8,9,1)++
2`(8,9,2)++
2`(8,9,4)++
2`(8,9,5)++
2`(8,10,1)++
2`(8,10,2)++
2`(8,10,3)++
2`(8,10,4)++
2`(8,10,5)++
2`(8,11,1)++
2`(8,11,3)++
2`(8,11,4)++
2`(8,12,5)++
2`(8,13,4)++
2`(8,15,3)++
2`(9,2,4)++
2`(9,3,1)++
2`(9,3,3)++
2`(9,4,1)++
2`(9,4,3)++
2`(9,4,5)++
2`(9,5,2)++
2`(9,5,3)++
2`(9,5,5)++
2`(9,6,2)++
2`(9,6,4)++
2`(9,6,5)++
2`(9,7,1)++
2`(9,7,2)++
2`(9,7,3)++
2`(9,7,5)++
2`(9,8,1)++
2`(9,8,2)++
2`(9,8,3)++
2`(9,8,4)++
2`(9,8,5)++
2`(9,9,1)++
2`(9,9,3)++
2`(9,9,4)++
2`(9,9,5)++
2`(9,10,1)++
2`(9,10,2)++
2`(9,10,3)++
2`(9,10,4)++
2`(9,10,5)++
2`(9,11,1)++
2`(9,11,4)++
2`(9,11,5)++
2`(9,15,3)++
2`(10,2,1)++
2`(10,2,3)++
2`(10,4,2)++
2`(10,4,4)++
2`(10,5,4)++
2`(10,5,5)++
2`(10,6,1)++
2`(10,6,2)++
2`(10,6,3)++
2`(10,6,4)++
2`(10,6,5)++
2`(10,7,1)++
2`(10,7,4)++
2`(10,8,1)++
2`(10,8,2)++
2`(10,8,3)++
2`(10,8,4)++
2`(10,8,5)++
2`(10,9,1)++
2`(10,9,2)++
2`(10,9,3)++
2`(10,9,4)++
2`(10,9,5)++
2`(10,10,1)++
2`(10,10,2)++
2`(10,10,5)++
2`(10,11,2)++
2`(10,11,3)++
2`(10,11,4)++
2`(10,12,4)++
2`(10,12,5)++
2`(10,13,4)

AgentD1'signal2 1
1`(0,0,0)++
2`(1,3,4)++
2`(1,5,1)++
2`(1,5,5)++
2`(1,6,1)++
2`(1,6,2)++
2`(1,6,3)++
2`(1,6,4)++
2`(1,6,5)++
2`(1,7,2)++
2`(1,7,3)++
2`(1,7,4)++
2`(1,7,5)++
2`(1,8,1)++
2`(1,8,2)++
2`(1,8,3)++
2`(1,8,4)++
2`(1,8,5)++
2`(1,9,2)++
2`(1,9,3)++
2`(1,9,4)++
2`(1,9,5)++
2`(1,10,1)++
2`(1,10,3)++
2`(1,10,4)++
2`(1,10,5)++
2`(1,11,1)++
2`(1,11,3)++
2`(1,11,4)++
2`(1,11,5)++
2`(1,12,1)++
2`(1,12,2)++
2`(1,12,5)++
2`(1,13,5)++
2`(1,14,2)++

2`(2,2,2)++
2`(2,5,3)++
2`(2,5,5)++
2`(2,6,2)++
2`(2,6,3)++
2`(2,6,4)++
2`(2,6,5)++
2`(2,7,1)++
2`(2,7,2)++
2`(2,7,3)++
2`(2,7,4)++
2`(2,7,5)++
2`(2,8,1)++
2`(2,8,2)++
2`(2,8,3)++
2`(2,8,4)++
2`(2,9,1)++
2`(2,9,2)++
2`(2,9,3)++
2`(2,9,4)++
2`(2,9,5)++
2`(2,10,1)++
2`(2,10,2)++
2`(2,10,3)++
2`(2,10,4)++
2`(2,11,2)++
2`(2,11,3)++
2`(2,12,4)++
2`(2,13,1)++
2`(2,13,4)++
2`(3,3,5)++
2`(3,4,1)++
2`(3,4,3)++
2`(3,6,1)++
2`(3,6,2)++
2`(3,6,3)++
2`(3,6,4)++
2`(3,6,5)++
2`(3,7,1)++
2`(3,7,2)++
2`(3,7,3)++
2`(3,7,5)++
2`(3,8,1)++
2`(3,8,2)++
2`(3,8,3)++
2`(3,8,4)++
2`(3,8,5)++
2`(3,9,2)++
2`(3,9,4)++
2`(3,9,5)++
2`(3,10,1)++
2`(3,10,2)++
2`(3,10,3)++
2`(3,10,4)++
2`(3,10,5)++
2`(3,11,2)++
2`(3,11,3)++
2`(3,11,4)++
2`(3,12,1)++
2`(3,12,2)++
2`(3,12,4)++
2`(3,13,3)++
2`(3,13,4)++
2`(3,14,2)++
2`(3,14,3)++
2`(4,1,3)++
2`(4,3,2)++
2`(4,5,4)++
2`(4,5,5)++
2`(4,6,1)++
2`(4,6,2)++
2`(4,6,3)++
2`(4,6,4)++
2`(4,6,5)++
2`(4,7,2)++
2`(4,7,3)++
2`(4,7,4)++
2`(4,7,5)++
2`(4,8,1)++
2`(4,8,3)++
2`(4,8,4)++
2`(4,8,5)++
2`(4,9,1)++
2`(4,9,2)++
2`(4,9,5)++
2`(4,10,1)++
2`(4,10,2)++
2`(4,10,3)++
2`(4,10,4)++
2`(4,10,5)++
2`(4,11,2)++
2`(4,11,3)++
2`(4,11,5)++
2`(4,12,2)++
2`(4,13,2)++
2`(4,13,3)++
2`(5,1,4)++
2`(5,4,2)++
2`(5,5,5)++
2`(5,6,1)++
2`(5,6,3)++
2`(5,6,4)++
2`(5,6,5)++
2`(5,7,1)++
2`(5,7,2)++
2`(5,7,3)++
2`(5,7,4)++
2`(5,8,1)++
2`(5,8,2)++
2`(5,8,4)++
2`(5,8,5)++
2`(5,9,1)++
2`(5,9,2)++
2`(5,9,3)++
2`(5,9,4)++
2`(5,9,5)++
2`(5,10,1)++
2`(5,10,2)++
2`(5,10,3)++
2`(5,10,4)++
2`(5,10,5)++
2`(5,11,2)++
2`(5,11,4)++
2`(5,12,2)++
2`(5,12,3)++
2`(5,13,2)++
2`(5,13,3)++
2`(5,14,2)++
2`(6,5,2)++
2`(6,5,4)++
2`(6,5,5)++
2`(6,6,1)++

```
2`(6,6,2)++
2`(6,6,3)++
2`(6,6,4)++
2`(6,6,5)++
2`(6,7,1)++
2`(6,7,2)++
2`(6,7,3)++
2`(6,7,4)++
2`(6,8,1)++
2`(6,8,2)++
2`(6,8,4)++
2`(6,8,5)++
2`(6,9,1)++
2`(6,9,2)++
2`(6,9,3)++
2`(6,9,4)++
2`(6,9,5)++
2`(6,10,1)++
2`(6,10,2)++
2`(6,10,3)++
2`(6,10,4)++
2`(6,10,5)++
2`(6,11,2)++
2`(6,12,1)++
2`(6,12,2)++
2`(6,12,5)++
2`(6,13,1)++
2`(6,14,3)++
2`(7,1,1)++
2`(7,2,3)++
2`(7,4,3)++
2`(7,5,1)++
2`(7,5,2)++
2`(7,5,3)++
2`(7,5,4)++
2`(7,6,1)++
2`(7,6,2)++
2`(7,6,3)++
2`(7,6,5)++
2`(7,7,1)++
2`(7,7,3)++
2`(7,7,4)++
2`(7,7,5)++
2`(7,8,2)++
2`(7,8,3)++
2`(7,8,5)++
2`(7,9,2)++
2`(7,9,3)++
2`(7,9,4)++
2`(7,10,1)++
2`(7,10,2)++
2`(7,10,4)++
2`(7,10,5)++
2`(7,11,1)++
2`(7,11,3)++
2`(7,12,2)++
2`(7,12,3)++
2`(7,12,4)++
2`(7,13,4)++
2`(7,14,1)++
2`(8,1,3)++
2`(8,2,3)++
2`(8,3,5)++
2`(8,4,2)++
2`(8,5,3)++
2`(8,5,5)++
2`(8,6,1)++
2`(8,6,2)++
2`(8,6,4)++
2`(8,6,5)++
2`(8,7,1)++
2`(8,7,2)++
2`(8,7,4)++
2`(8,7,5)++
2`(8,8,1)++
2`(8,8,2)++
2`(8,8,3)++
2`(8,8,4)++
2`(8,8,5)++
2`(8,9,1)++
2`(8,9,2)++
2`(8,9,3)++
2`(8,9,4)++
2`(8,9,5)++
2`(8,10,1)++
2`(8,10,2)++
2`(8,10,3)++
2`(8,10,4)++
2`(8,11,1)++
2`(8,11,5)++
2`(8,12,2)++
2`(8,13,2)++
2`(8,13,3)++
2`(8,14,3)++
2`(9,5,1)++
2`(9,5,3)++
2`(9,5,4)++
2`(9,6,1)++
2`(9,6,2)++
2`(9,6,3)++
2`(9,6,4)++
2`(9,6,5)++
2`(9,7,1)++
2`(9,7,2)++
2`(9,7,4)++
2`(9,8,1)++
2`(9,8,2)++
2`(9,8,3)++
2`(9,8,4)++
2`(9,8,5)++
2`(9,9,2)++
2`(9,9,4)++
2`(9,9,5)++
2`(9,10,2)++
2`(9,10,3)++
2`(9,10,4)++
2`(9,10,5)++
2`(9,11,2)++
2`(9,11,3)++
2`(9,11,4)++
2`(9,12,1)++
2`(9,12,3)++
2`(9,13,2)++
2`(9,13,4)++
2`(9,14,5)++
2`(10,4,2)++
2`(10,5,2)++
2`(10,6,1)++
2`(10,6,3)++
2`(10,6,4)++
2`(10,6,5)++
2`(10,7,1)++
```

2`(10,7,3)++
2`(10,7,5)++
2`(10,8,2)++
2`(10,8,3)++
2`(10,8,4)++
2`(10,8,5)++
2`(10,9,1)++
2`(10,9,2)++
2`(10,9,3)++
2`(10,9,4)++
2`(10,9,5)++
2`(10,10,1)++
2`(10,10,3)++
2`(10,10,4)++
2`(10,10,5)++
2`(10,11,2)++
2`(10,11,3)++
2`(10,11,4)++
2`(10,12,3)++
2`(10,14,2)

AgentD1'signal3 1
1`(0,0,0)++
2`(1,4,1)++
2`(1,4,3)++
2`(1,5,1)++
2`(1,5,2)++
2`(1,6,2)++
2`(1,6,4)++
2`(1,6,5)++
2`(1,7,1)++
2`(1,7,5)++
2`(1,8,1)++
2`(1,8,2)++
2`(1,8,3)++
2`(1,8,4)++
2`(1,8,5)++
2`(1,9,1)++
2`(1,9,2)++
2`(1,9,3)++
2`(1,9,4)++
2`(1,9,5)++
2`(1,10,1)++
2`(1,10,2)++
2`(1,10,4)++
2`(1,10,5)++
2`(1,11,1)++
2`(1,11,2)++
2`(1,11,3)++
2`(1,11,4)++
2`(1,11,5)++
2`(1,12,1)++
2`(1,12,2)++
2`(1,12,3)++
2`(1,12,4)++
2`(1,12,5)++
2`(1,13,1)++
2`(1,13,2)++
2`(1,13,3)++
2`(1,13,5)++
2`(1,14,1)++
2`(2,3,5)++
2`(2,4,4)++
2`(2,5,3)++
2`(2,5,4)++
2`(2,6,2)++
2`(2,6,3)++
2`(2,6,4)++
2`(2,7,1)++
2`(2,7,3)++
2`(2,7,5)++
2`(2,8,1)++
2`(2,8,2)++
2`(2,8,3)++
2`(2,8,4)++
2`(2,8,5)++
2`(2,9,1)++
2`(2,9,2)++
2`(2,9,3)++
2`(2,9,5)++
2`(2,10,1)++
2`(2,10,2)++
2`(2,10,3)++
2`(2,10,4)++
2`(2,10,5)++
2`(2,11,1)++
2`(2,11,2)++
2`(2,11,3)++
2`(2,11,4)++
2`(2,11,5)++
2`(2,12,1)++
2`(2,12,3)++
2`(2,12,4)++
2`(2,12,5)++
2`(2,13,2)++
2`(2,13,5)++
2`(3,2,1)++
2`(3,3,5)++
2`(3,4,1)++
2`(3,5,1)++
2`(3,5,4)++
2`(3,6,1)++
2`(3,6,2)++
2`(3,6,3)++
2`(3,6,5)++
2`(3,7,1)++
2`(3,7,2)++
2`(3,7,5)++
2`(3,8,1)++
2`(3,8,2)++
2`(3,8,3)++
2`(3,8,4)++
2`(3,9,1)++
2`(3,9,2)++
2`(3,9,4)++
2`(3,9,5)++
2`(3,10,1)++
2`(3,10,2)++
2`(3,10,4)++
2`(3,11,1)++
2`(3,11,2)++
2`(3,11,3)++
2`(3,11,5)++
2`(3,12,1)++
2`(3,12,2)++
2`(3,12,3)++
2`(3,12,5)++
2`(3,13,1)++
2`(3,13,2)++
2`(4,3,4)++
2`(4,5,4)++
2`(4,5,5)++

2`(4,6,1)++
2`(4,6,4)++
2`(4,6,5)++
2`(4,7,1)++
2`(4,7,3)++
2`(4,7,5)++
2`(4,8,2)++
2`(4,8,3)++
2`(4,8,4)++
2`(4,8,5)++
2`(4,9,1)++
2`(4,9,2)++
2`(4,9,3)++
2`(4,9,4)++
2`(4,9,5)++
2`(4,10,1)++
2`(4,10,2)++
2`(4,10,3)++
2`(4,10,5)++
2`(4,11,1)++
2`(4,11,2)++
2`(4,11,4)++
2`(4,12,1)++
2`(4,12,2)++
2`(4,12,3)++
2`(4,12,4)++
2`(4,12,5)++
2`(4,13,3)++
2`(4,13,4)++
2`(4,13,5)++
2`(5,2,1)++
2`(5,2,5)++
2`(5,4,1)++
2`(5,4,2)++
2`(5,4,4)++
2`(5,5,2)++
2`(5,5,4)++
2`(5,6,1)++
2`(5,6,2)++
2`(5,6,3)++
2`(5,6,5)++
2`(5,7,1)++
2`(5,7,2)++
2`(5,7,4)++
2`(5,7,5)++
2`(5,8,2)++
2`(5,8,4)++
2`(5,8,5)++
2`(5,9,2)++
2`(5,9,3)++
2`(5,9,4)++
2`(5,9,5)++
2`(5,10,1)++
2`(5,10,2)++
2`(5,10,4)++
2`(5,10,5)++
2`(5,11,3)++
2`(5,11,4)++
2`(5,12,1)++
2`(5,12,2)++
2`(5,12,3)++
2`(5,12,4)++
2`(5,12,5)++
2`(5,13,1)++
2`(5,13,3)++
2`(5,13,4)++
2`(5,13,5)++
2`(5,14,4)++
2`(6,1,1)++
2`(6,2,3)++
2`(6,4,2)++
2`(6,4,4)++
2`(6,4,5)++
2`(6,6,2)++
2`(6,6,3)++
2`(6,6,4)++
2`(6,6,5)++
2`(6,7,1)++
2`(6,7,2)++
2`(6,7,3)++
2`(6,7,4)++
2`(6,7,5)++
2`(6,8,1)++
2`(6,8,3)++
2`(6,8,4)++
2`(6,9,1)++
2`(6,9,2)++
2`(6,9,3)++
2`(6,9,4)++
2`(6,9,5)++
2`(6,10,1)++
2`(6,10,2)++
2`(6,10,3)++
2`(6,10,4)++
2`(6,10,5)++
2`(6,11,1)++
2`(6,11,2)++
2`(6,11,3)++
2`(6,11,4)++
2`(6,11,5)++
2`(6,12,1)++
2`(6,12,3)++
2`(6,12,4)++
2`(6,13,1)++
2`(6,13,2)++
2`(6,13,4)++
2`(7,2,5)++
2`(7,4,2)++
2`(7,4,3)++
2`(7,4,5)++
2`(7,5,3)++
2`(7,6,5)++
2`(7,7,4)++
2`(7,8,2)++
2`(7,8,3)++
2`(7,8,4)++
2`(7,8,5)++
2`(7,9,2)++
2`(7,9,3)++
2`(7,9,4)++
2`(7,10,1)++
2`(7,10,2)++
2`(7,10,3)++
2`(7,10,4)++
2`(7,11,1)++
2`(7,11,2)++
2`(7,11,3)++
2`(7,12,1)++
2`(7,12,2)++
2`(7,12,3)++
2`(7,12,4)++
2`(7,12,5)++

```
2`(7,13,1)++
2`(7,13,3)++
2`(7,13,4)++
2`(7,13,5)++
2`(7,14,1)++
2`(8,1,4)++
2`(8,3,2)++
2`(8,4,1)++
2`(8,4,2)++
2`(8,4,3)++
2`(8,6,1)++
2`(8,6,2)++
2`(8,6,4)++
2`(8,7,1)++
2`(8,7,2)++
2`(8,7,4)++
2`(8,7,5)++
2`(8,8,1)++
2`(8,8,2)++
2`(8,8,3)++
2`(8,8,4)++
2`(8,8,5)++
2`(8,9,1)++
2`(8,9,2)++
2`(8,9,3)++
2`(8,9,5)++
2`(8,10,1)++
2`(8,10,2)++
2`(8,10,3)++
2`(8,10,4)++
2`(8,10,5)++
2`(8,11,1)++
2`(8,11,2)++
2`(8,11,3)++
2`(8,11,5)++
2`(8,12,3)++
2`(8,12,4)++
2`(8,13,3)++
2`(8,13,4)++
2`(9,4,2)++
2`(9,4,4)++
2`(9,5,5)++
2`(9,6,3)++
2`(9,6,5)++
2`(9,7,1)++
2`(9,7,2)++
2`(9,7,4)++
2`(9,7,5)++
2`(9,8,1)++
2`(9,8,2)++
2`(9,8,3)++
2`(9,8,5)++
2`(9,9,2)++
2`(9,9,4)++
2`(9,9,5)++
2`(9,10,1)++
2`(9,10,2)++
2`(9,10,3)++
2`(9,10,4)++
2`(9,11,2)++
2`(9,11,4)++
2`(9,12,1)++
2`(9,12,3)++
2`(9,12,4)++
2`(9,13,1)++
2`(9,13,3)++
2`(9,13,4)++
2`(9,13,5)++
2`(9,14,3)++
2`(10,3,2)++
2`(10,5,1)++
2`(10,5,3)++
2`(10,6,1)++
2`(10,6,4)++
2`(10,6,5)++
2`(10,7,1)++
2`(10,7,3)++
2`(10,7,4)++
2`(10,7,5)++
2`(10,8,1)++
2`(10,8,2)++
2`(10,8,5)++
2`(10,9,1)++
2`(10,9,2)++
2`(10,9,3)++
2`(10,9,5)++
2`(10,10,1)++
2`(10,10,2)++
2`(10,10,3)++
2`(10,10,4)++
2`(10,10,5)++
2`(10,11,1)++
2`(10,11,2)++
2`(10,11,3)++
2`(10,11,4)++
2`(10,11,5)++
2`(10,12,2)++
2`(10,12,3)++
2`(10,12,5)++
2`(10,13,1)++
2`(10,13,2)++
2`(10,13,3)++
2`(10,13,4)

AgentD1'tableauN 1
1`0

AgentD1'tache1 1
3`(t,r,0,0)++
3`(t,r,1,1)++
3`(t,r,2,2)++
3`(t,r,3,3)++
3`(t,r,5,5)

AgentD1'tache2 1
1`(t,r,3,6)++
1`(t,r,3,11)++
1`(t,r,3,15)++
1`(t,r,3,16)++
1`(t,r,3,17)++
1`(t,r,3,19)++
1`(t,r,3,20)++
1`(t,r,3,21)++
1`(t,r,3,22)++
1`(t,r,3,23)++
1`(t,r,3,24)++
1`(t,r,3,25)++
1`(t,r,3,27)++
1`(t,r,3,28)

AgentD1'tamp11 1
1`(t,r,0,0)++
```

1`(t,r,3,0)++
1`(t,r,3,1)++
1`(t,r,3,2)++
1`(t,r,3,3)

AgentD1'tamp12 1
1`(t,r,1,3)++
1`(t,r,1,4)++
1`(t,r,1,8)++
1`(t,r,1,13)++
1`(t,r,1,20)++
1`(t,r,1,24)++
1`(t,r,1,36)++
1`(t,r,1,43)++
1`(t,r,1,63)++
1`(t,r,1,80)++
1`(t,r,1,113)++
1`(t,r,1,191)++
1`(t,r,1,201)++
1`(t,r,1,206)++
1`(t,r,1,217)++
1`(t,r,1,226)++
1`(t,r,1,232)++
1`(t,r,1,461)++
1`(t,r,2,2)++
1`(t,r,2,16)++
1`(t,r,2,21)++
1`(t,r,2,28)++
1`(t,r,2,29)++
1`(t,r,2,32)++
1`(t,r,2,33)++
1`(t,r,2,55)++
1`(t,r,2,79)++
1`(t,r,2,90)++
1`(t,r,2,94)++
1`(t,r,2,111)++
1`(t,r,2,125)++
1`(t,r,2,143)++
1`(t,r,2,149)++
1`(t,r,2,305)++
1`(t,r,2,636)++
1`(t,r,3,25)++
1`(t,r,3,28)++
1`(t,r,3,31)++
1`(t,r,3,34)++
1`(t,r,3,36)++
1`(t,r,3,48)++
1`(t,r,3,49)++
1`(t,r,3,58)++
1`(t,r,3,67)++
1`(t,r,3,69)++
1`(t,r,3,79)++
1`(t,r,3,89)++
1`(t,r,3,93)++
1`(t,r,3,107)++
1`(t,r,3,110)++
1`(t,r,3,115)++
1`(t,r,3,117)++
1`(t,r,3,128)++
1`(t,r,3,176)++
1`(t,r,3,316)++
1`(t,r,3,336)++
1`(t,r,3,451)++
1`(t,r,3,548)

AgentD1'tamp13 1
1`(t,r,1,0)++
1`(t,r,1,2)++
1`(t,r,2,0)++
1`(t,r,2,2)++
1`(t,r,3,0)++
1`(t,r,3,1)++
1`(t,r,3,2)

AgentD1'tamp14 1
3`(t,r,0,0)++
3`(t,r,1,1)++
3`(t,r,2,2)++
3`(t,r,3,3)++
3`(t,r,5,5)

AgentD1'tamp15 1
1`1++
1`2++
1`3

AgentD1'tamp16 1
1`1

AgentD1'tamp22 1
1`(t,r,1,6)++
1`(t,r,1,15)++
1`(t,r,1,16)++
1`(t,r,1,17)++
1`(t,r,1,19)++
1`(t,r,1,20)++
1`(t,r,1,23)++
1`(t,r,1,24)++
1`(t,r,1,25)++
1`(t,r,1,28)++
1`(t,r,2,11)++
1`(t,r,2,15)++
1`(t,r,2,16)++
1`(t,r,2,19)++
1`(t,r,2,21)++
1`(t,r,2,22)++
1`(t,r,2,23)++
1`(t,r,2,24)++
1`(t,r,2,27)++
1`(t,r,2,28)++
1`(t,r,3,0)++
1`(t,r,3,15)++
1`(t,r,3,16)++
1`(t,r,3,17)++
1`(t,r,3,19)++
1`(t,r,3,20)++
1`(t,r,3,21)++
1`(t,r,3,22)++
1`(t,r,3,23)++
1`(t,r,3,24)++
1`(t,r,3,25)++
1`(t,r,3,27)

AgentD1'verrou1 1
1`1

AgentD2'BaseCon2 1
1`(t,r,1,6)++
1`(t,r,1,15)++
1`(t,r,1,17)++
1`(t,r,1,19)++
1`(t,r,1,20)++

1`(t,r,1,23)++
1`(t,r,1,25)++
1`(t,r,1,28)++
1`(t,r,2,11)++
1`(t,r,2,16)++
1`(t,r,2,19)++
1`(t,r,2,21)++
1`(t,r,2,22)++
1`(t,r,2,27)++
1`(t,r,2,28)++
1`(t,r,3,0)++
1`(t,r,3,15)++
1`(t,r,3,16)++
1`(t,r,3,17)++
1`(t,r,3,19)++
1`(t,r,3,20)++
1`(t,r,3,21)++
1`(t,r,3,23)++
1`(t,r,3,24)++
1`(t,r,3,25)++
1`(t,r,3,27)

AgentD2'CPUmod 1
1`6++
1`11++
1`15++
1`16++
1`17++
1`19++
1`20++
1`21++
1`22++
1`23++
1`24++
1`25++
1`27++
1`28

AgentD2'CPUrate 1
1`20++
1`24++
1`27++
1`29++
1`39++
1`50++
1`51++
1`53++
1`54++
1`61++
1`63++
1`70++
1`77++
1`83++
1`84++
1`88++
1`90++
1`91++
1`95++
1`103++
1`120++
1`143++
1`173++
1`212++
1`224++
1`303++
1`380++
1`614

AgentD2'Directi2 1
1`(t,r,1,6)++
1`(t,r,1,15)++
1`(t,r,1,16)++
1`(t,r,1,17)++
1`(t,r,1,19)++
1`(t,r,1,20)++
1`(t,r,1,23)++
1`(t,r,1,24)++
1`(t,r,1,25)++
1`(t,r,1,28)++
1`(t,r,2,11)++
1`(t,r,2,15)++
1`(t,r,2,16)++
1`(t,r,2,19)++
1`(t,r,2,21)++
1`(t,r,2,22)++
1`(t,r,2,23)++
1`(t,r,2,24)++
1`(t,r,2,27)++
1`(t,r,2,28)++
1`(t,r,3,0)++
1`(t,r,3,15)++
1`(t,r,3,16)++
1`(t,r,3,17)++
1`(t,r,3,19)++
1`(t,r,3,20)++
1`(t,r,3,21)++
1`(t,r,3,22)++
1`(t,r,3,23)++
1`(t,r,3,24)++
1`(t,r,3,25)++
1`(t,r,3,27)

AgentD2'FACt11 1
1`0++
1`1

AgentD2'FACt12 1
1`0++
1`1

AgentD2'FACt13 1
1`0++
1`1

AgentD2'FAcT11 1
1`0++
1`1

AgentD2'FAcT12 1
1`0++
1`1

AgentD2'FAcT13 1
1`0++
1`1

AgentD2'FAct11 1
1`0++
1`1

AgentD2'FAct12 1
1`0++

1`1

AgentD2'FAct13 1
1`0++
1`1

AgentD2'FilAct21 1
1`(6,3)++
1`(11,3)++
1`(15,3)++
1`(16,3)++
1`(17,3)++
1`(19,3)++
1`(20,3)++
1`(21,3)++
1`(22,3)++
1`(23,3)++
1`(24,3)++
1`(25,3)++
1`(27,3)++
1`(28,3)

AgentD2'FilAct22 1
1`(6,3)++
1`(11,3)++
1`(15,3)++
1`(16,3)++
1`(17,3)++
1`(19,3)++
1`(20,3)++
1`(21,3)++
1`(22,3)++
1`(23,3)++
1`(24,3)++
1`(25,3)++
1`(27,3)++
1`(28,3)

AgentD2'FilAct23 1
1`(6,3)++
1`(11,3)++
1`(15,3)++
1`(16,3)++
1`(17,3)++
1`(19,3)++
1`(20,3)++
1`(21,0)++
1`(22,0)++
1`(23,0)++
1`(24,0)++
1`(25,0)++
1`(27,0)++
1`(28,0)

AgentD2'InhibET2 1
1`(3,6)++
1`(3,11)++
1`(3,15)++
1`(3,16)++
1`(3,17)++
1`(3,19)++
1`(3,20)++
1`(3,21)++
1`(3,22)++
1`(3,23)++
1`(3,24)++
1`(3,25)++
1`(3,27)++
1`(3,28)

AgentD2'InhibR2 1
1`(3,6)++
1`(3,11)++
1`(3,15)++
1`(3,16)++
1`(3,17)++
1`(3,19)++
1`(3,20)++
1`(3,21)++
1`(3,22)++
1`(3,23)++
1`(3,24)++
1`(3,25)++
1`(3,27)++
1`(3,28)

AgentD2'InhibW2 1
1`(3,6)++
1`(3,11)++
1`(3,15)++
1`(3,16)++
1`(3,17)++
1`(3,19)++
1`(3,20)++
1`(3,21)++
1`(3,22)++
1`(3,23)++
1`(3,24)++
1`(3,25)++
1`(3,27)++
1`(3,28)

AgentD2'MesgI2 1
1`6++
1`11++
1`15++
1`16++
1`17++
1`19++
1`20++
1`21++
1`22++
1`23++
1`24++
1`25++
1`27++
1`28

AgentD2'MessageR 1
2`r

AgentD2'PerceET2 1
1`1++
1`2++
1`3++
1`4++
1`5++
1`6++
1`7++
1`8++
1`9++
1`10

AgentD2'PercepC1 1
1`(1,1)

AgentD2'PercepC2 1
1`0++
1`1++
1`2++
1`3++
1`4++
1`5++
1`6++
1`7++
1`8++
1`9++
1`10++
1`11++
1`12++
1`13++
1`14++
1`15++
1`16++
1`17++
1`18++
1`19++
1`20++
1`21++
1`22++
1`23++
1`24++
1`25++
1`26++
1`27++
1`28++
1`29++
1`30++
1`31++
1`32++
1`33++
1`34++
1`35++
1`36++
1`37++
1`38++
1`39++
1`40++
1`41++
1`42++
1`43++
1`44++
1`45++
1`46++
1`47++
1`48++
1`49++
1`50++
1`51++
1`52++
1`53++
1`54++
1`55++
1`56++
1`57++
1`58++
1`59++
1`60++
1`61++
1`62++
1`63++
1`64++
1`65++
1`66++
1`67++
1`68++
1`69++
1`70++
1`71++
1`72++
1`73++
1`74++
1`75++
1`76++
1`77++
1`78++
1`79++
1`80++
1`81++
1`82++
1`83++
1`84++
1`85++
1`86++
1`87++
1`88++
1`89++
1`90++
1`91++
1`92++
1`93++
1`94++
1`95++
1`96++
1`97++
1`98++
1`99++
1`100++
1`101++
1`102++
1`103++
1`104++
1`105++
1`106++
1`107++
1`108++
1`109++
1`110++
1`111++
1`112++
1`113++
1`114++
1`115++
1`116++
1`117++
1`118++
1`119++
1`120++
1`121++
1`122++
1`123++
1`124++
1`125++
1`126++
1`127++

1`128++
1`129++
1`130++
1`131++
1`132++
1`133++
1`134++
1`135++
1`136++
1`137++
1`138++
1`139++
1`140++
1`141++
1`142++
1`143++
1`145++
1`146++
1`147++
1`148++
1`149++
1`150++
1`152++
1`153++
1`154++
1`155++
1`156++
1`157++
1`158++
1`159++
1`160++
1`162++
1`163++
1`164++
1`165++
1`167++
1`169++
1`170++
1`172++
1`173++
1`174++
1`177++
1`178++
1`179++
1`180++
1`181++
1`182++
1`183++
1`184++
1`185++
1`186++
1`187++
1`188++
1`189++
1`190++
1`192++
1`197++
1`199++
1`200++
1`201++
1`203++
1`204++
1`205++
1`206++
1`207++
1`208++
1`210++
1`211++
1`212++
1`213++
1`214++
1`215++
1`216++
1`217++
1`218++
1`221++
1`222++
1`224++
1`226++
1`227++
1`228++
1`229++
1`230++
1`231++
1`233++
1`235++
1`236++
1`237++
1`238++
1`239++
1`241++
1`244++
1`247++
1`249++
1`251++
1`253++
1`255++
1`258++
1`264++
1`265++
1`267++
1`268++
1`269++
1`278++
1`280++
1`282++
1`286++
1`289++
1`291++
1`292++
1`299++
1`300++
1`301++
1`306++
1`308++
1`309++
1`311++
1`313++
1`318++
1`320++
1`328++
1`329++
1`331++
1`337++
1`338++
1`340++
1`341++
1`342++
1`356++
1`357++
1`363++
1`364++

1`366++
1`368++
1`374++
1`384++
1`385++
1`388++
1`394++
1`413++
1`415++
1`422++
1`428++
1`437++
1`466++
1`468++
1`472++
1`507++
1`518++
1`533++
1`539++
1`544++
1`591++
1`887++
1`1287

AgentD2'PercepR2 1
1`1++
1`2++
1`3++
1`4++
1`5++
1`6++
1`7++
1`8++
1`9++
1`10

AgentD2'PercepW2 1
1`1++
1`2++
1`3++
1`4++
1`5++
1`6++
1`7++
1`8++
1`9++
1`10

AgentD2'Plans1 1
1`(t,r,1,3)++
1`(t,r,1,4)++
1`(t,r,1,8)++
1`(t,r,1,13)++
1`(t,r,1,20)++
1`(t,r,1,24)++
1`(t,r,1,36)++
1`(t,r,1,43)++
1`(t,r,1,63)++
1`(t,r,1,80)++
1`(t,r,1,113)++
1`(t,r,1,191)++
1`(t,r,1,201)++
1`(t,r,1,206)++
1`(t,r,1,217)++
1`(t,r,1,226)++
1`(t,r,1,232)++
1`(t,r,1,461)++
1`(t,r,2,2)++
1`(t,r,2,16)++
1`(t,r,2,21)++
1`(t,r,2,28)++
1`(t,r,2,29)++
1`(t,r,2,32)++
1`(t,r,2,33)++
1`(t,r,2,55)++
1`(t,r,2,79)++
1`(t,r,2,90)++
1`(t,r,2,94)++
1`(t,r,2,111)++
1`(t,r,2,125)++
1`(t,r,2,143)++
1`(t,r,2,149)++
1`(t,r,2,305)++
1`(t,r,2,636)++
1`(t,r,3,25)++
1`(t,r,3,28)++
1`(t,r,3,31)++
1`(t,r,3,34)++
1`(t,r,3,36)++
1`(t,r,3,48)++
1`(t,r,3,49)++
1`(t,r,3,58)++
1`(t,r,3,67)++
1`(t,r,3,69)++
1`(t,r,3,79)++
1`(t,r,3,89)++
1`(t,r,3,93)++
1`(t,r,3,107)++
1`(t,r,3,110)++
1`(t,r,3,115)++
1`(t,r,3,117)++
1`(t,r,3,128)++
1`(t,r,3,176)++
1`(t,r,3,316)++
1`(t,r,3,336)++
1`(t,r,3,451)++
1`(t,r,3,548)

AgentD2'Plans2 1
1`(t,r,1,6)++
1`(t,r,1,15)++
1`(t,r,1,16)++
1`(t,r,1,17)++
1`(t,r,1,19)++
1`(t,r,1,20)++
1`(t,r,1,23)++
1`(t,r,1,24)++
1`(t,r,1,25)++
1`(t,r,1,28)++
1`(t,r,2,11)++
1`(t,r,2,15)++
1`(t,r,2,16)++
1`(t,r,2,19)++
1`(t,r,2,21)++
1`(t,r,2,22)++
1`(t,r,2,23)++
1`(t,r,2,24)++
1`(t,r,2,27)++
1`(t,r,2,28)++
1`(t,r,3,0)++
1`(t,r,3,15)++

1`(t,r,3,16)++
1`(t,r,3,17)++
1`(t,r,3,19)++
1`(t,r,3,20)++
1`(t,r,3,21)++
1`(t,r,3,22)++
1`(t,r,3,23)++
1`(t,r,3,24)++
1`(t,r,3,25)++
1`(t,r,3,27)

AgentD2'RaportT2 1
1`6++
1`11++
1`15++
1`16++
1`17++
1`19++
1`20++
1`21++
1`22++
1`23++
1`24++
1`25++
1`27++
1`28

AgentD2'Scenario 1
2`(t,r,1,3)++
2`(t,r,1,4)++
2`(t,r,1,8)++
2`(t,r,1,13)++
2`(t,r,1,20)++
2`(t,r,1,24)++
2`(t,r,1,36)++
2`(t,r,1,43)++
2`(t,r,1,63)++
2`(t,r,1,80)++
2`(t,r,1,113)++
2`(t,r,1,191)++
2`(t,r,1,201)++
2`(t,r,1,206)++
2`(t,r,1,217)++
2`(t,r,1,226)++
2`(t,r,1,232)++
2`(t,r,1,461)++
2`(t,r,2,2)++
2`(t,r,2,16)++
2`(t,r,2,21)++
2`(t,r,2,28)++
2`(t,r,2,29)++
2`(t,r,2,32)++
2`(t,r,2,33)++
2`(t,r,2,55)++
2`(t,r,2,79)++
2`(t,r,2,90)++
2`(t,r,2,94)++
2`(t,r,2,111)++
2`(t,r,2,125)++
2`(t,r,2,143)++
2`(t,r,2,149)++
2`(t,r,2,305)++
2`(t,r,2,636)++
2`(t,r,3,25)++
2`(t,r,3,28)++
2`(t,r,3,31)++
2`(t,r,3,34)++
2`(t,r,3,36)++
2`(t,r,3,48)++
2`(t,r,3,49)++
2`(t,r,3,58)++
2`(t,r,3,67)++
2`(t,r,3,69)++
2`(t,r,3,79)++
2`(t,r,3,89)++
2`(t,r,3,93)++
2`(t,r,3,107)++
2`(t,r,3,110)++
2`(t,r,3,115)++
2`(t,r,3,117)++
2`(t,r,3,128)++
2`(t,r,3,176)++
2`(t,r,3,316)++
2`(t,r,3,336)++
2`(t,r,3,451)++
2`(t,r,3,548)
AgentD2'Sig1 1
1`(1,3,2)++
1`(1,4,1)++
1`(1,4,3)++
1`(1,4,4)++
1`(1,4,5)++
1`(1,5,1)++
1`(1,5,3)++
1`(1,5,4)++
1`(1,5,5)++
1`(1,6,1)++
1`(1,6,2)++
1`(1,6,3)++
1`(1,6,5)++
1`(1,7,1)++
1`(1,7,2)++
1`(1,7,4)++
1`(1,7,5)++
1`(1,8,2)++
1`(1,8,3)++
1`(1,8,4)++
1`(1,8,5)++
1`(1,9,2)++
1`(1,9,3)++
1`(1,9,4)++
1`(1,9,5)++
1`(1,10,1)++
1`(1,10,2)++
1`(1,10,3)++
1`(1,10,5)++
1`(1,11,1)++
1`(1,11,2)++
1`(1,11,3)++
1`(1,11,5)++
1`(1,12,1)++
1`(1,12,5)++
1`(1,13,4)++
1`(2,1,4)++
1`(2,3,2)++
1`(2,3,4)++
1`(2,4,2)++
1`(2,4,3)++
1`(2,4,4)++
1`(2,5,1)++

1`(2,5,2)++
1`(2,5,3)++
1`(2,5,4)++
1`(2,5,5)++
1`(2,6,2)++
1`(2,6,3)++
1`(2,6,4)++
1`(2,6,5)++
1`(2,7,1)++
1`(2,7,2)++
1`(2,7,3)++
1`(2,7,4)++
1`(2,7,5)++
1`(2,8,2)++
1`(2,8,3)++
1`(2,8,5)++
1`(2,9,1)++
1`(2,9,3)++
1`(2,9,4)++
1`(2,10,1)++
1`(2,10,2)++
1`(2,10,4)++
1`(2,10,5)++
1`(2,11,1)++
1`(2,11,2)++
1`(2,11,3)++
1`(2,11,4)++
1`(2,11,5)++
1`(2,12,1)++
1`(2,12,3)++
1`(2,13,1)++
1`(2,13,2)++
1`(2,15,1)++
1`(3,4,4)++
1`(3,4,5)++
1`(3,5,5)++
1`(3,6,1)++
1`(3,6,3)++
1`(3,6,4)++
1`(3,6,5)++
1`(3,7,1)++
1`(3,7,2)++
1`(3,7,3)++
1`(3,7,4)++
1`(3,7,5)++
1`(3,8,2)++
1`(3,8,3)++
1`(3,8,4)++
1`(3,8,5)++
1`(3,9,1)++
1`(3,9,2)++
1`(3,9,3)++
1`(3,9,4)++
1`(3,9,5)++
1`(3,10,1)++
1`(3,10,2)++
1`(3,10,3)++
1`(3,10,5)++
1`(3,11,2)++
1`(3,11,3)++
1`(3,11,4)++
1`(3,12,1)++
1`(3,12,3)++
1`(4,2,2)++
1`(4,4,1)++
1`(4,4,3)++
1`(4,5,2)++
1`(4,5,3)++
1`(4,5,4)++
1`(4,5,5)++
1`(4,6,1)++
1`(4,6,2)++
1`(4,6,3)++
1`(4,6,5)++
1`(4,7,1)++
1`(4,7,3)++
1`(4,7,4)++
1`(4,7,5)++
1`(4,8,1)++
1`(4,8,2)++
1`(4,8,3)++
1`(4,8,4)++
1`(4,8,5)++
1`(4,9,1)++
1`(4,9,2)++
1`(4,9,3)++
1`(4,9,4)++
1`(4,9,5)++
1`(4,10,1)++
1`(4,10,3)++
1`(4,10,4)++
1`(4,11,1)++
1`(4,11,2)++
1`(4,11,3)++
1`(4,12,2)++
1`(4,12,3)++
1`(4,12,5)++
1`(5,3,1)++
1`(5,3,4)++
1`(5,4,1)++
1`(5,4,3)++
1`(5,5,3)++
1`(5,5,5)++
1`(5,6,1)++
1`(5,6,2)++
1`(5,6,4)++
1`(5,6,5)++
1`(5,7,1)++
1`(5,7,2)++
1`(5,7,3)++
1`(5,7,4)++
1`(5,8,1)++
1`(5,8,2)++
1`(5,8,3)++
1`(5,8,4)++
1`(5,8,5)++
1`(5,9,1)++
1`(5,9,2)++
1`(5,9,3)++
1`(5,9,4)++
1`(5,9,5)++
1`(5,10,1)++
1`(5,10,2)++
1`(5,10,4)++
1`(5,10,5)++
1`(5,11,1)++
1`(5,11,4)++
1`(5,11,5)++
1`(5,12,1)++
1`(5,12,2)++
1`(5,12,5)++
1`(5,13,2)++

1`(6,1,1)++
1`(6,4,1)++
1`(6,4,2)++
1`(6,4,3)++
1`(6,4,4)++
1`(6,5,1)++
1`(6,5,4)++
1`(6,5,5)++
1`(6,6,1)++
1`(6,6,3)++
1`(6,6,4)++
1`(6,7,1)++
1`(6,7,2)++
1`(6,7,4)++
1`(6,8,1)++
1`(6,8,2)++
1`(6,8,3)++
1`(6,8,4)++
1`(6,8,5)++
1`(6,9,1)++
1`(6,9,2)++
1`(6,9,3)++
1`(6,9,4)++
1`(6,9,5)++
1`(6,10,1)++
1`(6,10,2)++
1`(6,10,3)++
1`(6,11,1)++
1`(6,11,2)++
1`(6,11,3)++
1`(6,11,5)++
1`(6,12,2)++
1`(6,13,2)++
1`(7,4,2)++
1`(7,5,2)++
1`(7,5,5)++
1`(7,6,3)++
1`(7,6,4)++
1`(7,6,5)++
1`(7,7,1)++
1`(7,7,2)++
1`(7,7,3)++
1`(7,7,4)++
1`(7,8,1)++
1`(7,8,2)++
1`(7,8,3)++
1`(7,8,4)++
1`(7,8,5)++
1`(7,9,1)++
1`(7,9,2)++
1`(7,9,3)++
1`(7,9,4)++
1`(7,9,5)++
1`(7,10,2)++
1`(7,10,3)++
1`(7,10,4)++
1`(7,10,5)++
1`(7,11,1)++
1`(7,11,2)++
1`(7,11,3)++
1`(7,11,4)++
1`(7,11,5)++
1`(7,12,3)++
1`(7,13,2)++
1`(7,14,1)++
1`(7,15,2)++
1`(8,1,4)++
1`(8,2,3)++
1`(8,4,1)++
1`(8,4,2)++
1`(8,4,4)++
1`(8,4,5)++
1`(8,5,2)++
1`(8,5,5)++
1`(8,6,1)++
1`(8,6,2)++
1`(8,6,3)++
1`(8,7,2)++
1`(8,7,3)++
1`(8,7,4)++
1`(8,7,5)++
1`(8,8,1)++
1`(8,8,2)++
1`(8,8,4)++
1`(8,8,5)++
1`(8,9,1)++
1`(8,9,2)++
1`(8,9,4)++
1`(8,9,5)++
1`(8,10,1)++
1`(8,10,2)++
1`(8,10,3)++
1`(8,10,4)++
1`(8,10,5)++
1`(8,11,1)++
1`(8,11,3)++
1`(8,11,4)++
1`(8,12,5)++
1`(8,13,4)++
1`(8,15,3)++
1`(9,2,4)++
1`(9,3,1)++
1`(9,3,3)++
1`(9,4,1)++
1`(9,4,3)++
1`(9,4,5)++
1`(9,5,2)++
1`(9,5,3)++
1`(9,5,5)++
1`(9,6,2)++
1`(9,6,4)++
1`(9,6,5)++
1`(9,7,1)++
1`(9,7,2)++
1`(9,7,3)++
1`(9,7,5)++
1`(9,8,1)++
1`(9,8,2)++
1`(9,8,3)++
1`(9,8,4)++
1`(9,8,5)++
1`(9,9,1)++
1`(9,9,3)++
1`(9,9,4)++
1`(9,9,5)++
1`(9,10,1)++
1`(9,10,2)++
1`(9,10,3)++
1`(9,10,4)++
1`(9,10,5)++
1`(9,11,1)++
1`(9,11,4)++

1`(9,11,5)++
1`(9,15,3)++
1`(10,2,1)++
1`(10,2,3)++
1`(10,4,2)++
1`(10,4,4)++
1`(10,5,4)++
1`(10,5,5)++
1`(10,6,1)++
1`(10,6,2)++
1`(10,6,3)++
1`(10,6,4)++
1`(10,6,5)++
1`(10,7,1)++
1`(10,7,4)++
1`(10,8,1)++
1`(10,8,2)++
1`(10,8,3)++
1`(10,8,4)++
1`(10,8,5)++
1`(10,9,1)++
1`(10,9,2)++
1`(10,9,3)++
1`(10,9,4)++
1`(10,9,5)++
1`(10,10,1)++
1`(10,10,2)++
1`(10,10,5)++
1`(10,11,2)++
1`(10,11,3)++
1`(10,11,4)++
1`(10,12,4)++
1`(10,12,5)++
1`(10,13,4)

AgentD2'Sig2 1
1`(1,3,4)++
1`(1,5,1)++
1`(1,5,5)++
1`(1,6,1)++
1`(1,6,2)++
1`(1,6,3)++
1`(1,6,4)++
1`(1,6,5)++
1`(1,7,2)++
1`(1,7,3)++
1`(1,7,4)++
1`(1,7,5)++
1`(1,8,1)++
1`(1,8,2)++
1`(1,8,3)++
1`(1,8,4)++
1`(1,8,5)++
1`(1,9,2)++
1`(1,9,3)++
1`(1,9,4)++
1`(1,9,5)++
1`(1,10,1)++
1`(1,10,3)++
1`(1,10,4)++
1`(1,10,5)++
1`(1,11,1)++
1`(1,11,3)++
1`(1,11,4)++
1`(1,11,5)++
1`(1,12,1)++
1`(1,12,2)++
1`(1,12,5)++
1`(1,13,5)++
1`(1,14,2)++
1`(2,2,2)++
1`(2,5,3)++
1`(2,5,5)++
1`(2,6,2)++
1`(2,6,3)++
1`(2,6,4)++
1`(2,6,5)++
1`(2,7,1)++
1`(2,7,2)++
1`(2,7,3)++
1`(2,7,4)++
1`(2,7,5)++
1`(2,8,1)++
1`(2,8,2)++
1`(2,8,3)++
1`(2,8,4)++
1`(2,9,1)++
1`(2,9,2)++
1`(2,9,3)++
1`(2,9,4)++
1`(2,9,5)++
1`(2,10,1)++
1`(2,10,2)++
1`(2,10,3)++
1`(2,10,4)++
1`(2,11,2)++
1`(2,11,3)++
1`(2,12,4)++
1`(2,13,1)++
1`(2,13,4)++
1`(3,3,5)++
1`(3,4,1)++
1`(3,4,3)++
1`(3,6,1)++
1`(3,6,2)++
1`(3,6,3)++
1`(3,6,4)++
1`(3,6,5)++
1`(3,7,1)++
1`(3,7,2)++
1`(3,7,3)++
1`(3,7,5)++
1`(3,8,1)++
1`(3,8,2)++
1`(3,8,3)++
1`(3,8,4)++
1`(3,8,5)++
1`(3,9,2)++
1`(3,9,4)++
1`(3,9,5)++
1`(3,10,1)++
1`(3,10,2)++
1`(3,10,3)++
1`(3,10,4)++
1`(3,10,5)++
1`(3,11,2)++
1`(3,11,3)++
1`(3,11,4)++
1`(3,12,1)++
1`(3,12,2)++
1`(3,12,4)++
1`(3,13,3)++

1`(3,13,4)++
1`(3,14,2)++
1`(3,14,3)++
1`(4,1,3)++
1`(4,3,2)++
1`(4,5,4)++
1`(4,5,5)++
1`(4,6,1)++
1`(4,6,2)++
1`(4,6,3)++
1`(4,6,4)++
1`(4,6,5)++
1`(4,7,2)++
1`(4,7,3)++
1`(4,7,4)++
1`(4,7,5)++
1`(4,8,1)++
1`(4,8,3)++
1`(4,8,4)++
1`(4,8,5)++
1`(4,9,1)++
1`(4,9,2)++
1`(4,9,5)++
1`(4,10,1)++
1`(4,10,2)++
1`(4,10,3)++
1`(4,10,4)++
1`(4,10,5)++
1`(4,11,2)++
1`(4,11,3)++
1`(4,11,5)++
1`(4,12,2)++
1`(4,13,2)++
1`(4,13,3)++
1`(5,1,4)++
1`(5,4,2)++
1`(5,5,5)++
1`(5,6,1)++
1`(5,6,3)++
1`(5,6,4)++
1`(5,6,5)++
1`(5,7,1)++
1`(5,7,2)++
1`(5,7,3)++
1`(5,7,4)++
1`(5,8,1)++
1`(5,8,2)++
1`(5,8,4)++
1`(5,8,5)++
1`(5,9,1)++
1`(5,9,2)++
1`(5,9,3)++
1`(5,9,4)++
1`(5,9,5)++
1`(5,10,1)++
1`(5,10,2)++
1`(5,10,3)++
1`(5,10,4)++
1`(5,10,5)++
1`(5,11,2)++
1`(5,11,4)++
1`(5,12,2)++
1`(5,12,3)++
1`(5,13,2)++
1`(5,13,3)++
1`(5,14,2)++
1`(6,5,2)++
1`(6,5,4)++
1`(6,5,5)++
1`(6,6,1)++
1`(6,6,2)++
1`(6,6,3)++
1`(6,6,4)++
1`(6,6,5)++
1`(6,7,1)++
1`(6,7,2)++
1`(6,7,3)++
1`(6,7,4)++
1`(6,8,1)++
1`(6,8,2)++
1`(6,8,4)++
1`(6,8,5)++
1`(6,9,1)++
1`(6,9,2)++
1`(6,9,3)++
1`(6,9,4)++
1`(6,9,5)++
1`(6,10,1)++
1`(6,10,2)++
1`(6,10,3)++
1`(6,10,4)++
1`(6,10,5)++
1`(6,11,2)++
1`(6,12,1)++
1`(6,12,2)++
1`(6,12,5)++
1`(6,13,1)++
1`(6,14,3)++
1`(7,1,1)++
1`(7,2,3)++
1`(7,4,3)++
1`(7,5,1)++
1`(7,5,2)++
1`(7,5,3)++
1`(7,5,4)++
1`(7,6,1)++
1`(7,6,2)++
1`(7,6,3)++
1`(7,6,5)++
1`(7,7,1)++
1`(7,7,3)++
1`(7,7,4)++
1`(7,7,5)++
1`(7,8,2)++
1`(7,8,3)++
1`(7,8,5)++
1`(7,9,2)++
1`(7,9,3)++
1`(7,9,4)++
1`(7,10,1)++
1`(7,10,2)++
1`(7,10,4)++
1`(7,10,5)++
1`(7,11,1)++
1`(7,11,3)++
1`(7,12,2)++
1`(7,12,3)++
1`(7,12,4)++
1`(7,13,4)++
1`(7,14,1)++
1`(8,1,3)++
1`(8,2,3)++

1`(8,3,5)++
1`(8,4,2)++
1`(8,5,3)++
1`(8,5,5)++
1`(8,6,1)++
1`(8,6,2)++
1`(8,6,4)++
1`(8,6,5)++
1`(8,7,1)++
1`(8,7,2)++
1`(8,7,4)++
1`(8,7,5)++
1`(8,8,1)++
1`(8,8,2)++
1`(8,8,3)++
1`(8,8,4)++
1`(8,8,5)++
1`(8,9,1)++
1`(8,9,2)++
1`(8,9,3)++
1`(8,9,4)++
1`(8,9,5)++
1`(8,10,1)++
1`(8,10,2)++
1`(8,10,3)++
1`(8,10,4)++
1`(8,11,1)++
1`(8,11,5)++
1`(8,12,2)++
1`(8,13,2)++
1`(8,13,3)++
1`(8,14,3)++
1`(9,5,1)++
1`(9,5,3)++
1`(9,5,4)++
1`(9,6,1)++
1`(9,6,2)++
1`(9,6,3)++
1`(9,6,4)++
1`(9,6,5)++
1`(9,7,1)++
1`(9,7,2)++
1`(9,7,4)++
1`(9,8,1)++
1`(9,8,2)++
1`(9,8,3)++
1`(9,8,4)++
1`(9,8,5)++
1`(9,9,2)++
1`(9,9,4)++
1`(9,9,5)++
1`(9,10,2)++
1`(9,10,3)++
1`(9,10,4)++
1`(9,10,5)++
1`(9,11,2)++
1`(9,11,3)++
1`(9,11,4)++
1`(9,12,1)++
1`(9,12,3)++
1`(9,13,2)++
1`(9,13,4)++
1`(9,14,5)++
1`(10,4,2)++
1`(10,5,2)++
1`(10,6,1)++
1`(10,6,3)++
1`(10,6,4)++
1`(10,6,5)++
1`(10,7,1)++
1`(10,7,3)++
1`(10,7,5)++
1`(10,8,2)++
1`(10,8,3)++
1`(10,8,4)++
1`(10,8,5)++
1`(10,9,1)++
1`(10,9,2)++
1`(10,9,3)++
1`(10,9,4)++
1`(10,9,5)++
1`(10,10,1)++
1`(10,10,3)++
1`(10,10,4)++
1`(10,10,5)++
1`(10,11,2)++
1`(10,11,3)++
1`(10,11,4)++
1`(10,12,3)++
1`(10,14,2)

AgentD2'Sig3 1
1`(1,4,1)++
1`(1,4,3)++
1`(1,5,1)++
1`(1,5,2)++
1`(1,6,2)++
1`(1,6,4)++
1`(1,6,5)++
1`(1,7,1)++
1`(1,7,5)++
1`(1,8,1)++
1`(1,8,2)++
1`(1,8,3)++
1`(1,8,4)++
1`(1,8,5)++
1`(1,9,1)++
1`(1,9,2)++
1`(1,9,3)++
1`(1,9,4)++
1`(1,9,5)++
1`(1,10,1)++
1`(1,10,2)++
1`(1,10,4)++
1`(1,10,5)++
1`(1,11,1)++
1`(1,11,2)++
1`(1,11,3)++
1`(1,11,4)++
1`(1,11,5)++
1`(1,12,1)++
1`(1,12,2)++
1`(1,12,3)++
1`(1,12,4)++
1`(1,12,5)++
1`(1,13,1)++
1`(1,13,2)++
1`(1,13,3)++
1`(1,13,5)++
1`(1,14,1)++
1`(2,3,5)++
1`(2,4,4)++

```
1`(2,5,3)++
1`(2,5,4)++
1`(2,6,2)++
1`(2,6,3)++
1`(2,6,4)++
1`(2,7,1)++
1`(2,7,3)++
1`(2,7,5)++
1`(2,8,1)++
1`(2,8,2)++
1`(2,8,3)++
1`(2,8,4)++
1`(2,8,5)++
1`(2,9,1)++
1`(2,9,2)++
1`(2,9,3)++
1`(2,9,5)++
1`(2,10,1)++
1`(2,10,2)++
1`(2,10,3)++
1`(2,10,4)++
1`(2,10,5)++
1`(2,11,1)++
1`(2,11,2)++
1`(2,11,3)++
1`(2,11,4)++
1`(2,11,5)++
1`(2,12,1)++
1`(2,12,3)++
1`(2,12,4)++
1`(2,12,5)++
1`(2,13,2)++
1`(2,13,5)++
1`(3,2,1)++
1`(3,3,5)++
1`(3,4,1)++
1`(3,5,1)++
1`(3,5,4)++
1`(3,6,1)++
1`(3,6,2)++
1`(3,6,3)++
1`(3,6,5)++
1`(3,7,1)++
1`(3,7,2)++
1`(3,7,5)++
1`(3,8,1)++
1`(3,8,2)++
1`(3,8,3)++
1`(3,8,4)++
1`(3,9,1)++
1`(3,9,2)++
1`(3,9,4)++
1`(3,9,5)++
1`(3,10,1)++
1`(3,10,2)++
1`(3,10,4)++
1`(3,11,1)++
1`(3,11,2)++
1`(3,11,3)++
1`(3,11,5)++
1`(3,12,1)++
1`(3,12,2)++
1`(3,12,3)++
1`(3,12,5)++
1`(3,13,1)++
1`(3,13,2)++
1`(4,3,4)++
1`(4,5,4)++
1`(4,5,5)++
1`(4,6,1)++
1`(4,6,4)++
1`(4,6,5)++
1`(4,7,1)++
1`(4,7,3)++
1`(4,7,5)++
1`(4,8,2)++
1`(4,8,3)++
1`(4,8,4)++
1`(4,8,5)++
1`(4,9,1)++
1`(4,9,2)++
1`(4,9,3)++
1`(4,9,4)++
1`(4,9,5)++
1`(4,10,1)++
1`(4,10,2)++
1`(4,10,3)++
1`(4,10,5)++
1`(4,11,1)++
1`(4,11,2)++
1`(4,11,4)++
1`(4,12,1)++
1`(4,12,2)++
1`(4,12,3)++
1`(4,12,4)++
1`(4,12,5)++
1`(4,13,3)++
1`(4,13,4)++
1`(4,13,5)++
1`(5,2,1)++
1`(5,2,5)++
1`(5,4,1)++
1`(5,4,2)++
1`(5,4,4)++
1`(5,5,2)++
1`(5,5,4)++
1`(5,6,1)++
1`(5,6,2)++
1`(5,6,3)++
1`(5,6,5)++
1`(5,7,1)++
1`(5,7,2)++
1`(5,7,4)++
1`(5,7,5)++
1`(5,8,2)++
1`(5,8,4)++
1`(5,8,5)++
1`(5,9,2)++
1`(5,9,3)++
1`(5,9,4)++
1`(5,9,5)++
1`(5,10,1)++
1`(5,10,2)++
1`(5,10,4)++
1`(5,10,5)++
1`(5,11,3)++
1`(5,11,4)++
1`(5,12,1)++
1`(5,12,2)++
1`(5,12,3)++
1`(5,12,4)++
1`(5,12,5)++
```

```
1`(5,13,1)++
1`(5,13,3)++
1`(5,13,4)++
1`(5,13,5)++
1`(5,14,4)++
1`(6,1,1)++
1`(6,2,3)++
1`(6,4,2)++
1`(6,4,4)++
1`(6,4,5)++
1`(6,6,2)++
1`(6,6,3)++
1`(6,6,4)++
1`(6,6,5)++
1`(6,7,1)++
1`(6,7,2)++
1`(6,7,3)++
1`(6,7,4)++
1`(6,7,5)++
1`(6,8,1)++
1`(6,8,3)++
1`(6,8,4)++
1`(6,9,1)++
1`(6,9,2)++
1`(6,9,3)++
1`(6,9,4)++
1`(6,9,5)++
1`(6,10,1)++
1`(6,10,2)++
1`(6,10,3)++
1`(6,10,4)++
1`(6,10,5)++
1`(6,11,1)++
1`(6,11,2)++
1`(6,11,3)++
1`(6,11,4)++
1`(6,11,5)++
1`(6,12,1)++
1`(6,12,3)++
1`(6,12,4)++
1`(6,13,1)++
1`(6,13,2)++
1`(6,13,4)++
1`(7,2,5)++
1`(7,4,2)++
1`(7,4,3)++
1`(7,4,5)++
1`(7,5,3)++
1`(7,6,5)++
1`(7,7,4)++
1`(7,8,2)++
1`(7,8,3)++
1`(7,8,4)++
1`(7,8,5)++
1`(7,9,2)++
1`(7,9,3)++
1`(7,9,4)++
1`(7,10,1)++
1`(7,10,2)++
1`(7,10,3)++
1`(7,10,4)++
1`(7,11,1)++
1`(7,11,2)++
1`(7,11,3)++
1`(7,12,1)++
1`(7,12,2)++
1`(7,12,3)++
1`(7,12,4)++
1`(7,12,5)++
1`(7,13,1)++
1`(7,13,3)++
1`(7,13,4)++
1`(7,13,5)++
1`(7,14,1)++
1`(8,1,4)++
1`(8,3,2)++
1`(8,4,1)++
1`(8,4,2)++
1`(8,4,3)++
1`(8,6,1)++
1`(8,6,2)++
1`(8,6,4)++
1`(8,7,1)++
1`(8,7,2)++
1`(8,7,4)++
1`(8,7,5)++
1`(8,8,1)++
1`(8,8,2)++
1`(8,8,3)++
1`(8,8,4)++
1`(8,8,5)++
1`(8,9,1)++
1`(8,9,2)++
1`(8,9,3)++
1`(8,9,5)++
1`(8,10,1)++
1`(8,10,2)++
1`(8,10,3)++
1`(8,10,4)++
1`(8,10,5)++
1`(8,11,1)++
1`(8,11,2)++
1`(8,11,3)++
1`(8,11,5)++
1`(8,12,3)++
1`(8,12,4)++
1`(8,13,3)++
1`(8,13,4)++
1`(9,4,2)++
1`(9,4,4)++
1`(9,5,5)++
1`(9,6,3)++
1`(9,6,5)++
1`(9,7,1)++
1`(9,7,2)++
1`(9,7,4)++
1`(9,7,5)++
1`(9,8,1)++
1`(9,8,2)++
1`(9,8,3)++
1`(9,8,5)++
1`(9,9,2)++
1`(9,9,4)++
1`(9,9,5)++
1`(9,10,1)++
1`(9,10,2)++
1`(9,10,3)++
1`(9,10,4)++
1`(9,11,2)++
1`(9,11,4)++
1`(9,12,1)++
1`(9,12,3)++
```

1`(9,12,4)++
1`(9,13,1)++
1`(9,13,3)++
1`(9,13,4)++
1`(9,13,5)++
1`(9,14,3)++
1`(10,3,2)++
1`(10,5,1)++
1`(10,5,3)++
1`(10,6,1)++
1`(10,6,4)++
1`(10,6,5)++
1`(10,7,1)++
1`(10,7,3)++
1`(10,7,4)++
1`(10,7,5)++
1`(10,8,1)++
1`(10,8,2)++
1`(10,8,5)++
1`(10,9,1)++
1`(10,9,2)++
1`(10,9,3)++
1`(10,9,5)++
1`(10,10,1)++
1`(10,10,2)++
1`(10,10,3)++
1`(10,10,4)++
1`(10,10,5)++
1`(10,11,1)++
1`(10,11,2)++
1`(10,11,3)++
1`(10,11,4)++
1`(10,11,5)++
1`(10,12,2)++
1`(10,12,3)++
1`(10,12,5)++
1`(10,13,1)++
1`(10,13,2)++
1`(10,13,3)++
1`(10,13,4)

AgentD2'Signal1 1
1`(0,0,0)++
2`(1,3,2)++
2`(1,4,1)++
2`(1,4,3)++
2`(1,4,4)++
2`(1,4,5)++
2`(1,5,1)++
2`(1,5,3)++
2`(1,5,4)++
2`(1,5,5)++
2`(1,6,1)++
2`(1,6,2)++
2`(1,6,3)++
2`(1,6,5)++
2`(1,7,1)++
2`(1,7,2)++
2`(1,7,4)++
2`(1,7,5)++
2`(1,8,2)++
2`(1,8,3)++
2`(1,8,4)++
2`(1,8,5)++
2`(1,9,2)++
2`(1,9,3)++
2`(1,9,4)++
2`(1,9,5)++
2`(1,10,1)++
2`(1,10,2)++
2`(1,10,3)++
2`(1,10,5)++
2`(1,11,1)++
2`(1,11,2)++
2`(1,11,3)++
2`(1,11,5)++
2`(1,12,1)++
2`(1,12,5)++
2`(1,13,4)++
2`(2,1,4)++
2`(2,3,2)++
2`(2,3,4)++
2`(2,4,2)++
2`(2,4,3)++
2`(2,4,4)++
2`(2,5,1)++
2`(2,5,2)++
2`(2,5,3)++
2`(2,5,4)++
2`(2,5,5)++
2`(2,6,2)++
2`(2,6,3)++
2`(2,6,4)++
2`(2,6,5)++
2`(2,7,1)++
2`(2,7,2)++
2`(2,7,3)++
2`(2,7,4)++
2`(2,7,5)++
2`(2,8,2)++
2`(2,8,3)++
2`(2,8,5)++
2`(2,9,1)++
2`(2,9,3)++
2`(2,9,4)++
2`(2,10,1)++
2`(2,10,2)++
2`(2,10,4)++
2`(2,10,5)++
2`(2,11,1)++
2`(2,11,2)++
2`(2,11,3)++
2`(2,11,4)++
2`(2,11,5)++
2`(2,12,1)++
2`(2,12,3)++
2`(2,13,1)++
2`(2,13,2)++
2`(2,15,1)++
2`(3,4,4)++
2`(3,4,5)++
2`(3,5,5)++
2`(3,6,1)++
2`(3,6,3)++
2`(3,6,4)++
2`(3,6,5)++
2`(3,7,1)++
2`(3,7,2)++
2`(3,7,3)++
2`(3,7,4)++
2`(3,7,5)++
2`(3,8,2)++

2`(3,8,3)++
2`(3,8,4)++
2`(3,8,5)++
2`(3,9,1)++
2`(3,9,2)++
2`(3,9,3)++
2`(3,9,4)++
2`(3,9,5)++
2`(3,10,1)++
2`(3,10,2)++
2`(3,10,3)++
2`(3,10,5)++
2`(3,11,2)++
2`(3,11,3)++
2`(3,11,4)++
2`(3,12,1)++
2`(3,12,3)++
2`(4,2,2)++
2`(4,4,1)++
2`(4,4,3)++
2`(4,5,2)++
2`(4,5,3)++
2`(4,5,4)++
2`(4,5,5)++
2`(4,6,1)++
2`(4,6,2)++
2`(4,6,3)++
2`(4,6,5)++
2`(4,7,1)++
2`(4,7,3)++
2`(4,7,4)++
2`(4,7,5)++
2`(4,8,1)++
2`(4,8,2)++
2`(4,8,3)++
2`(4,8,4)++
2`(4,8,5)++
2`(4,9,1)++
2`(4,9,2)++
2`(4,9,3)++
2`(4,9,4)++
2`(4,9,5)++
2`(4,10,1)++
2`(4,10,3)++
2`(4,10,4)++
2`(4,11,1)++
2`(4,11,2)++
2`(4,11,3)++
2`(4,12,2)++
2`(4,12,3)++
2`(4,12,5)++
2`(5,3,1)++
2`(5,3,4)++
2`(5,4,1)++
2`(5,4,3)++
2`(5,5,3)++
2`(5,5,5)++
2`(5,6,1)++
2`(5,6,2)++
2`(5,6,4)++
2`(5,6,5)++
2`(5,7,1)++
2`(5,7,2)++
2`(5,7,3)++
2`(5,7,4)++
2`(5,8,1)++
2`(5,8,2)++
2`(5,8,3)++
2`(5,8,4)++
2`(5,8,5)++
2`(5,9,1)++
2`(5,9,2)++
2`(5,9,3)++
2`(5,9,4)++
2`(5,9,5)++
2`(5,10,1)++
2`(5,10,2)++
2`(5,10,4)++
2`(5,10,5)++
2`(5,11,1)++
2`(5,11,4)++
2`(5,11,5)++
2`(5,12,1)++
2`(5,12,2)++
2`(5,12,5)++
2`(5,13,2)++
2`(6,1,1)++
2`(6,4,1)++
2`(6,4,2)++
2`(6,4,3)++
2`(6,4,4)++
2`(6,5,1)++
2`(6,5,4)++
2`(6,5,5)++
2`(6,6,1)++
2`(6,6,3)++
2`(6,6,4)++
2`(6,7,1)++
2`(6,7,2)++
2`(6,7,4)++
2`(6,8,1)++
2`(6,8,2)++
2`(6,8,3)++
2`(6,8,4)++
2`(6,8,5)++
2`(6,9,1)++
2`(6,9,2)++
2`(6,9,3)++
2`(6,9,4)++
2`(6,9,5)++
2`(6,10,1)++
2`(6,10,2)++
2`(6,10,3)++
2`(6,11,1)++
2`(6,11,2)++
2`(6,11,3)++
2`(6,11,5)++
2`(6,12,2)++
2`(6,13,2)++
2`(7,4,2)++
2`(7,5,2)++
2`(7,5,5)++
2`(7,6,3)++
2`(7,6,4)++
2`(7,6,5)++
2`(7,7,1)++
2`(7,7,2)++
2`(7,7,3)++
2`(7,7,4)++
2`(7,8,1)++
2`(7,8,2)++
2`(7,8,3)++

```
2`(7,8,4)++
2`(7,8,5)++
2`(7,9,1)++
2`(7,9,2)++
2`(7,9,3)++
2`(7,9,4)++
2`(7,9,5)++
2`(7,10,2)++
2`(7,10,3)++
2`(7,10,4)++
2`(7,10,5)++
2`(7,11,1)++
2`(7,11,2)++
2`(7,11,3)++
2`(7,11,4)++
2`(7,11,5)++
2`(7,12,3)++
2`(7,13,2)++
2`(7,14,1)++
2`(7,15,2)++
2`(8,1,4)++
2`(8,2,3)++
2`(8,4,1)++
2`(8,4,2)++
2`(8,4,4)++
2`(8,4,5)++
2`(8,5,2)++
2`(8,5,5)++
2`(8,6,1)++
2`(8,6,2)++
2`(8,6,3)++
2`(8,7,2)++
2`(8,7,3)++
2`(8,7,4)++
2`(8,7,5)++
2`(8,8,1)++
2`(8,8,2)++
2`(8,8,4)++
2`(8,8,5)++
2`(8,9,1)++
2`(8,9,2)++
2`(8,9,4)++
2`(8,9,5)++
2`(8,10,1)++
2`(8,10,2)++
2`(8,10,3)++
2`(8,10,4)++
2`(8,10,5)++
2`(8,11,1)++
2`(8,11,3)++
2`(8,11,4)++
2`(8,12,5)++
2`(8,13,4)++
2`(8,15,3)++
2`(9,2,4)++
2`(9,3,1)++
2`(9,3,3)++
2`(9,4,1)++
2`(9,4,3)++
2`(9,4,5)++
2`(9,5,2)++
2`(9,5,3)++
2`(9,5,5)++
2`(9,6,2)++
2`(9,6,4)++
2`(9,6,5)++
2`(9,7,1)++
2`(9,7,2)++
2`(9,7,3)++
2`(9,7,5)++
2`(9,8,1)++
2`(9,8,2)++
2`(9,8,3)++
2`(9,8,4)++
2`(9,8,5)++
2`(9,9,1)++
2`(9,9,3)++
2`(9,9,4)++
2`(9,9,5)++
2`(9,10,1)++
2`(9,10,2)++
2`(9,10,3)++
2`(9,10,4)++
2`(9,10,5)++
2`(9,11,1)++
2`(9,11,4)++
2`(9,11,5)++
2`(9,15,3)++
2`(10,2,1)++
2`(10,2,3)++
2`(10,4,2)++
2`(10,4,4)++
2`(10,5,4)++
2`(10,5,5)++
2`(10,6,1)++
2`(10,6,2)++
2`(10,6,3)++
2`(10,6,4)++
2`(10,6,5)++
2`(10,7,1)++
2`(10,7,4)++
2`(10,8,1)++
2`(10,8,2)++
2`(10,8,3)++
2`(10,8,4)++
2`(10,8,5)++
2`(10,9,1)++
2`(10,9,2)++
2`(10,9,3)++
2`(10,9,4)++
2`(10,9,5)++
2`(10,10,1)++
2`(10,10,2)++
2`(10,10,5)++
2`(10,11,2)++
2`(10,11,3)++
2`(10,11,4)++
2`(10,12,4)++
2`(10,12,5)++
2`(10,13,4)

AgentD2'Signal2 1
1`(0,0,0)++
2`(1,3,4)++
2`(1,5,1)++
2`(1,5,5)++
2`(1,6,1)++
2`(1,6,2)++
2`(1,6,3)++
2`(1,6,4)++
2`(1,6,5)++
2`(1,7,2)++
```

2`(1,7,3)++
2`(1,7,4)++
2`(1,7,5)++
2`(1,8,1)++
2`(1,8,2)++
2`(1,8,3)++
2`(1,8,4)++
2`(1,8,5)++
2`(1,9,2)++
2`(1,9,3)++
2`(1,9,4)++
2`(1,9,5)++
2`(1,10,1)++
2`(1,10,3)++
2`(1,10,4)++
2`(1,10,5)++
2`(1,11,1)++
2`(1,11,3)++
2`(1,11,4)++
2`(1,11,5)++
2`(1,12,1)++
2`(1,12,2)++
2`(1,12,5)++
2`(1,13,5)++
2`(1,14,2)++
2`(2,2,2)++
2`(2,5,3)++
2`(2,5,5)++
2`(2,6,2)++
2`(2,6,3)++
2`(2,6,4)++
2`(2,6,5)++
2`(2,7,1)++
2`(2,7,2)++
2`(2,7,3)++
2`(2,7,4)++
2`(2,7,5)++
2`(2,8,1)++
2`(2,8,2)++
2`(2,8,3)++
2`(2,8,4)++
2`(2,9,1)++
2`(2,9,2)++
2`(2,9,3)++
2`(2,9,4)++
2`(2,9,5)++
2`(2,10,1)++
2`(2,10,2)++
2`(2,10,3)++
2`(2,10,4)++
2`(2,11,2)++
2`(2,11,3)++
2`(2,12,4)++
2`(2,13,1)++
2`(2,13,4)++
2`(3,3,5)++
2`(3,4,1)++
2`(3,4,3)++
2`(3,6,1)++
2`(3,6,2)++
2`(3,6,3)++
2`(3,6,4)++
2`(3,6,5)++
2`(3,7,1)++
2`(3,7,2)++
2`(3,7,3)++
2`(3,7,5)++
2`(3,8,1)++
2`(3,8,2)++
2`(3,8,3)++
2`(3,8,4)++
2`(3,8,5)++
2`(3,9,2)++
2`(3,9,4)++
2`(3,9,5)++
2`(3,10,1)++
2`(3,10,2)++
2`(3,10,3)++
2`(3,10,4)++
2`(3,10,5)++
2`(3,11,2)++
2`(3,11,3)++
2`(3,11,4)++
2`(3,12,1)++
2`(3,12,2)++
2`(3,12,4)++
2`(3,13,3)++
2`(3,13,4)++
2`(3,14,2)++
2`(3,14,3)++
2`(4,1,3)++
2`(4,3,2)++
2`(4,5,4)++
2`(4,5,5)++
2`(4,6,1)++
2`(4,6,2)++
2`(4,6,3)++
2`(4,6,4)++
2`(4,6,5)++
2`(4,7,2)++
2`(4,7,3)++
2`(4,7,4)++
2`(4,7,5)++
2`(4,8,1)++
2`(4,8,3)++
2`(4,8,4)++
2`(4,8,5)++
2`(4,9,1)++
2`(4,9,2)++
2`(4,9,5)++
2`(4,10,1)++
2`(4,10,2)++
2`(4,10,3)++
2`(4,10,4)++
2`(4,10,5)++
2`(4,11,2)++
2`(4,11,3)++
2`(4,11,5)++
2`(4,12,2)++
2`(4,13,2)++
2`(4,13,3)++
2`(5,1,4)++
2`(5,4,2)++
2`(5,5,5)++
2`(5,6,1)++
2`(5,6,3)++
2`(5,6,4)++
2`(5,6,5)++
2`(5,7,1)++
2`(5,7,2)++
2`(5,7,3)++
2`(5,7,4)++

2`(5,8,1)++
2`(5,8,2)++
2`(5,8,4)++
2`(5,8,5)++
2`(5,9,1)++
2`(5,9,2)++
2`(5,9,3)++
2`(5,9,4)++
2`(5,9,5)++
2`(5,10,1)++
2`(5,10,2)++
2`(5,10,3)++
2`(5,10,4)++
2`(5,10,5)++
2`(5,11,2)++
2`(5,11,4)++
2`(5,12,2)++
2`(5,12,3)++
2`(5,13,2)++
2`(5,13,3)++
2`(5,14,2)++
2`(6,5,2)++
2`(6,5,4)++
2`(6,5,5)++
2`(6,6,1)++
2`(6,6,2)++
2`(6,6,3)++
2`(6,6,4)++
2`(6,6,5)++
2`(6,7,1)++
2`(6,7,2)++
2`(6,7,3)++
2`(6,7,4)++
2`(6,8,1)++
2`(6,8,2)++
2`(6,8,4)++
2`(6,8,5)++
2`(6,9,1)++
2`(6,9,2)++
2`(6,9,3)++
2`(6,9,4)++
2`(6,9,5)++
2`(6,10,1)++
2`(6,10,2)++
2`(6,10,3)++
2`(6,10,4)++
2`(6,10,5)++
2`(6,11,2)++
2`(6,12,1)++
2`(6,12,2)++
2`(6,12,5)++
2`(6,13,1)++
2`(6,14,3)++
2`(7,1,1)++
2`(7,2,3)++
2`(7,4,3)++
2`(7,5,1)++
2`(7,5,2)++
2`(7,5,3)++
2`(7,5,4)++
2`(7,6,1)++
2`(7,6,2)++
2`(7,6,3)++
2`(7,6,5)++
2`(7,7,1)++
2`(7,7,3)++
2`(7,7,4)++
2`(7,7,5)++
2`(7,8,2)++
2`(7,8,3)++
2`(7,8,5)++
2`(7,9,2)++
2`(7,9,3)++
2`(7,9,4)++
2`(7,10,1)++
2`(7,10,2)++
2`(7,10,4)++
2`(7,10,5)++
2`(7,11,1)++
2`(7,11,3)++
2`(7,12,2)++
2`(7,12,3)++
2`(7,12,4)++
2`(7,13,4)++
2`(7,14,1)++
2`(8,1,3)++
2`(8,2,3)++
2`(8,3,5)++
2`(8,4,2)++
2`(8,5,3)++
2`(8,5,5)++
2`(8,6,1)++
2`(8,6,2)++
2`(8,6,4)++
2`(8,6,5)++
2`(8,7,1)++
2`(8,7,2)++
2`(8,7,4)++
2`(8,7,5)++
2`(8,8,1)++
2`(8,8,2)++
2`(8,8,3)++
2`(8,8,4)++
2`(8,8,5)++
2`(8,9,1)++
2`(8,9,2)++
2`(8,9,3)++
2`(8,9,4)++
2`(8,9,5)++
2`(8,10,1)++
2`(8,10,2)++
2`(8,10,3)++
2`(8,10,4)++
2`(8,11,1)++
2`(8,11,5)++
2`(8,12,2)++
2`(8,13,2)++
2`(8,13,3)++
2`(8,14,3)++
2`(9,5,1)++
2`(9,5,3)++
2`(9,5,4)++
2`(9,6,1)++
2`(9,6,2)++
2`(9,6,3)++
2`(9,6,4)++
2`(9,6,5)++
2`(9,7,1)++
2`(9,7,2)++
2`(9,7,4)++
2`(9,8,1)++
2`(9,8,2)++

2`(9,8,3)++
2`(9,8,4)++
2`(9,8,5)++
2`(9,9,2)++
2`(9,9,4)++
2`(9,9,5)++
2`(9,10,2)++
2`(9,10,3)++
2`(9,10,4)++
2`(9,10,5)++
2`(9,11,2)++
2`(9,11,3)++
2`(9,11,4)++
2`(9,12,1)++
2`(9,12,3)++
2`(9,13,2)++
2`(9,13,4)++
2`(9,14,5)++
2`(10,4,2)++
2`(10,5,2)++
2`(10,6,1)++
2`(10,6,3)++
2`(10,6,4)++
2`(10,6,5)++
2`(10,7,1)++
2`(10,7,3)++
2`(10,7,5)++
2`(10,8,2)++
2`(10,8,3)++
2`(10,8,4)++
2`(10,8,5)++
2`(10,9,1)++
2`(10,9,2)++
2`(10,9,3)++
2`(10,9,4)++
2`(10,9,5)++
2`(10,10,1)++
2`(10,10,3)++
2`(10,10,4)++
2`(10,10,5)++
2`(10,11,2)++
2`(10,11,3)++
2`(10,11,4)++
2`(10,12,3)++
2`(10,14,2)

AgentD2'Signal3 1
1`(0,0,0)++
2`(1,4,1)++
2`(1,4,3)++
2`(1,5,1)++
2`(1,5,2)++
2`(1,6,2)++
2`(1,6,4)++
2`(1,6,5)++
2`(1,7,1)++
2`(1,7,5)++
2`(1,8,1)++
2`(1,8,2)++
2`(1,8,3)++
2`(1,8,4)++
2`(1,8,5)++
2`(1,9,1)++
2`(1,9,2)++
2`(1,9,3)++
2`(1,9,4)++
2`(1,9,5)++
2`(1,10,1)++
2`(1,10,2)++
2`(1,10,4)++
2`(1,10,5)++
2`(1,11,1)++
2`(1,11,2)++
2`(1,11,3)++
2`(1,11,4)++
2`(1,11,5)++
2`(1,12,1)++
2`(1,12,2)++
2`(1,12,3)++
2`(1,12,4)++
2`(1,12,5)++
2`(1,13,1)++
2`(1,13,2)++
2`(1,13,3)++
2`(1,13,5)++
2`(1,14,1)++
2`(2,3,5)++
2`(2,4,4)++
2`(2,5,3)++
2`(2,5,4)++
2`(2,6,2)++
2`(2,6,3)++
2`(2,6,4)++
2`(2,7,1)++
2`(2,7,3)++
2`(2,7,5)++
2`(2,8,1)++
2`(2,8,2)++
2`(2,8,3)++
2`(2,8,4)++
2`(2,8,5)++
2`(2,9,1)++
2`(2,9,2)++
2`(2,9,3)++
2`(2,9,5)++
2`(2,10,1)++
2`(2,10,2)++
2`(2,10,3)++
2`(2,10,4)++
2`(2,10,5)++
2`(2,11,1)++
2`(2,11,2)++
2`(2,11,3)++
2`(2,11,4)++
2`(2,11,5)++
2`(2,12,1)++
2`(2,12,3)++
2`(2,12,4)++
2`(2,12,5)++
2`(2,13,2)++
2`(2,13,5)++
2`(3,2,1)++
2`(3,3,5)++
2`(3,4,1)++
2`(3,5,1)++
2`(3,5,4)++
2`(3,6,1)++
2`(3,6,2)++
2`(3,6,3)++
2`(3,6,5)++
2`(3,7,1)++
2`(3,7,2)++

2`(3,7,5)++
2`(3,8,1)++
2`(3,8,2)++
2`(3,8,3)++
2`(3,8,4)++
2`(3,9,1)++
2`(3,9,2)++
2`(3,9,4)++
2`(3,9,5)++
2`(3,10,1)++
2`(3,10,2)++
2`(3,10,4)++
2`(3,11,1)++
2`(3,11,2)++
2`(3,11,3)++
2`(3,11,5)++
2`(3,12,1)++
2`(3,12,2)++
2`(3,12,3)++
2`(3,12,5)++
2`(3,13,1)++
2`(3,13,2)++
2`(4,3,4)++
2`(4,5,4)++
2`(4,5,5)++
2`(4,6,1)++
2`(4,6,4)++
2`(4,6,5)++
2`(4,7,1)++
2`(4,7,3)++
2`(4,7,5)++
2`(4,8,2)++
2`(4,8,3)++
2`(4,8,4)++
2`(4,8,5)++
2`(4,9,1)++
2`(4,9,2)++
2`(4,9,3)++
2`(4,9,4)++
2`(4,9,5)++
2`(4,10,1)++
2`(4,10,2)++
2`(4,10,3)++
2`(4,10,5)++
2`(4,11,1)++
2`(4,11,2)++
2`(4,11,4)++
2`(4,12,1)++
2`(4,12,2)++
2`(4,12,3)++
2`(4,12,4)++
2`(4,12,5)++
2`(4,13,3)++
2`(4,13,4)++
2`(4,13,5)++
2`(5,2,1)++
2`(5,2,5)++
2`(5,4,1)++
2`(5,4,2)++
2`(5,4,4)++
2`(5,5,2)++
2`(5,5,4)++
2`(5,6,1)++
2`(5,6,2)++
2`(5,6,3)++
2`(5,6,5)++
2`(5,7,1)++
2`(5,7,2)++
2`(5,7,4)++
2`(5,7,5)++
2`(5,8,2)++
2`(5,8,4)++
2`(5,8,5)++
2`(5,9,2)++
2`(5,9,3)++
2`(5,9,4)++
2`(5,9,5)++
2`(5,10,1)++
2`(5,10,2)++
2`(5,10,4)++
2`(5,10,5)++
2`(5,11,3)++
2`(5,11,4)++
2`(5,12,1)++
2`(5,12,2)++
2`(5,12,3)++
2`(5,12,4)++
2`(5,12,5)++
2`(5,13,1)++
2`(5,13,3)++
2`(5,13,4)++
2`(5,13,5)++
2`(5,14,4)++
2`(6,1,1)++
2`(6,2,3)++
2`(6,4,2)++
2`(6,4,4)++
2`(6,4,5)++
2`(6,6,2)++
2`(6,6,3)++
2`(6,6,4)++
2`(6,6,5)++
2`(6,7,1)++
2`(6,7,2)++
2`(6,7,3)++
2`(6,7,4)++
2`(6,7,5)++
2`(6,8,1)++
2`(6,8,3)++
2`(6,8,4)++
2`(6,9,1)++
2`(6,9,2)++
2`(6,9,3)++
2`(6,9,4)++
2`(6,9,5)++
2`(6,10,1)++
2`(6,10,2)++
2`(6,10,3)++
2`(6,10,4)++
2`(6,10,5)++
2`(6,11,1)++
2`(6,11,2)++
2`(6,11,3)++
2`(6,11,4)++
2`(6,11,5)++
2`(6,12,1)++
2`(6,12,3)++
2`(6,12,4)++
2`(6,13,1)++
2`(6,13,2)++
2`(6,13,4)++
2`(7,2,5)++

2`(7,4,2)++
2`(7,4,3)++
2`(7,4,5)++
2`(7,5,3)++
2`(7,6,5)++
2`(7,7,4)++
2`(7,8,2)++
2`(7,8,3)++
2`(7,8,4)++
2`(7,8,5)++
2`(7,9,2)++
2`(7,9,3)++
2`(7,9,4)++
2`(7,10,1)++
2`(7,10,2)++
2`(7,10,3)++
2`(7,10,4)++
2`(7,11,1)++
2`(7,11,2)++
2`(7,11,3)++
2`(7,12,1)++
2`(7,12,2)++
2`(7,12,3)++
2`(7,12,4)++
2`(7,12,5)++
2`(7,13,1)++
2`(7,13,3)++
2`(7,13,4)++
2`(7,13,5)++
2`(7,14,1)++
2`(8,1,4)++
2`(8,3,2)++
2`(8,4,1)++
2`(8,4,2)++
2`(8,4,3)++
2`(8,6,1)++
2`(8,6,2)++
2`(8,6,4)++
2`(8,7,1)++
2`(8,7,2)++
2`(8,7,4)++
2`(8,7,5)++
2`(8,8,1)++
2`(8,8,2)++
2`(8,8,3)++
2`(8,8,4)++
2`(8,8,5)++
2`(8,9,1)++
2`(8,9,2)++
2`(8,9,3)++
2`(8,9,5)++
2`(8,10,1)++
2`(8,10,2)++
2`(8,10,3)++
2`(8,10,4)++
2`(8,10,5)++
2`(8,11,1)++
2`(8,11,2)++
2`(8,11,3)++
2`(8,11,5)++
2`(8,12,3)++
2`(8,12,4)++
2`(8,13,3)++
2`(8,13,4)++
2`(9,4,2)++
2`(9,4,4)++
2`(9,5,5)++
2`(9,6,3)++
2`(9,6,5)++
2`(9,7,1)++
2`(9,7,2)++
2`(9,7,4)++
2`(9,7,5)++
2`(9,8,1)++
2`(9,8,2)++
2`(9,8,3)++
2`(9,8,5)++
2`(9,9,2)++
2`(9,9,4)++
2`(9,9,5)++
2`(9,10,1)++
2`(9,10,2)++
2`(9,10,3)++
2`(9,10,4)++
2`(9,11,2)++
2`(9,11,4)++
2`(9,12,1)++
2`(9,12,3)++
2`(9,12,4)++
2`(9,13,1)++
2`(9,13,3)++
2`(9,13,4)++
2`(9,13,5)++
2`(9,14,3)++
2`(10,3,2)++
2`(10,5,1)++
2`(10,5,3)++
2`(10,6,1)++
2`(10,6,4)++
2`(10,6,5)++
2`(10,7,1)++
2`(10,7,3)++
2`(10,7,4)++
2`(10,7,5)++
2`(10,8,1)++
2`(10,8,2)++
2`(10,8,5)++
2`(10,9,1)++
2`(10,9,2)++
2`(10,9,3)++
2`(10,9,5)++
2`(10,10,1)++
2`(10,10,2)++
2`(10,10,3)++
2`(10,10,4)++
2`(10,10,5)++
2`(10,11,1)++
2`(10,11,2)++
2`(10,11,3)++
2`(10,11,4)++
2`(10,11,5)++
2`(10,12,2)++
2`(10,12,3)++
2`(10,12,5)++
2`(10,13,1)++
2`(10,13,2)++
2`(10,13,3)++
2`(10,13,4)

AgentD2'Stimul21 1
1`0++
1`6++

1`11++
1`15++
1`16++
1`17++
1`19++
1`20++
1`21++
1`22++
1`23++
1`24++
1`25++
1`27++
1`28

AgentD2'Stimul22 1
1`0++
1`6++
1`11++
1`15++
1`16++
1`17++
1`19++
1`20++
1`21++
1`22++
1`23++
1`24++
1`25++
1`27++
1`28

AgentD2'Stimul23 1
1`0++
1`6++
1`11++
1`15++
1`16++
1`17++
1`19++
1`20++
1`21++
1`22++
1`23++
1`24++

1`25++
1`27++
1`28

AgentD2'TableauN 1
1`0

AgentD2'Tache1 1
3`(t,r,0,0)++
3`(t,r,1,1)++
3`(t,r,2,2)++
3`(t,r,3,3)++
3`(t,r,5,5)

AgentD2'Tache2 1
1`(t,r,3,6)++
1`(t,r,3,11)++
1`(t,r,3,15)++
1`(t,r,3,16)++
1`(t,r,3,17)++
1`(t,r,3,19)++
1`(t,r,3,20)++
1`(t,r,3,21)++
1`(t,r,3,22)++
1`(t,r,3,23)++
1`(t,r,3,24)++
1`(t,r,3,25)++
1`(t,r,3,27)++
1`(t,r,3,28)

AgentD2'Tamp12 1
1`(t,r,1,3)++
1`(t,r,1,4)++
1`(t,r,1,8)++
1`(t,r,1,13)++
1`(t,r,1,20)++
1`(t,r,1,24)++
1`(t,r,1,36)++
1`(t,r,1,43)++
1`(t,r,1,63)++
1`(t,r,1,80)++
1`(t,r,1,113)++
1`(t,r,1,191)++
1`(t,r,1,201)++

1`(t,r,1,206)++
1`(t,r,1,217)++
1`(t,r,1,226)++
1`(t,r,1,232)++
1`(t,r,1,461)++
1`(t,r,2,2)++
1`(t,r,2,16)++
1`(t,r,2,21)++
1`(t,r,2,28)++
1`(t,r,2,29)++
1`(t,r,2,32)++
1`(t,r,2,33)++
1`(t,r,2,55)++
1`(t,r,2,79)++
1`(t,r,2,90)++
1`(t,r,2,94)++
1`(t,r,2,111)++
1`(t,r,2,125)++
1`(t,r,2,143)++
1`(t,r,2,149)++
1`(t,r,2,305)++
1`(t,r,2,636)++
1`(t,r,3,25)++
1`(t,r,3,28)++
1`(t,r,3,31)++
1`(t,r,3,34)++
1`(t,r,3,36)++
1`(t,r,3,48)++
1`(t,r,3,49)++
1`(t,r,3,58)++
1`(t,r,3,67)++
1`(t,r,3,69)++
1`(t,r,3,79)++
1`(t,r,3,89)++
1`(t,r,3,93)++
1`(t,r,3,107)++
1`(t,r,3,110)++
1`(t,r,3,115)++
1`(t,r,3,117)++
1`(t,r,3,128)++
1`(t,r,3,176)++
1`(t,r,3,316)++
1`(t,r,3,336)++
1`(t,r,3,451)++

1`(t,r,3,548)

AgentD2'Tamp14 1
3`(t,r,0,0)++
3`(t,r,1,1)++
3`(t,r,2,2)++
3`(t,r,3,3)++
3`(t,r,5,5)

AgentD2'Tamp15 1
1`1++
1`2++
1`3

AgentD2'Tamp16 1
1`1

AgentD2'Tamp22 1
1`(t,r,1,6)++
1`(t,r,1,15)++
1`(t,r,1,16)++
1`(t,r,1,17)++
1`(t,r,1,19)++
1`(t,r,1,20)++
1`(t,r,1,23)++
1`(t,r,1,24)++
1`(t,r,1,25)++
1`(t,r,1,28)++
1`(t,r,2,11)++
1`(t,r,2,15)++
1`(t,r,2,16)++
1`(t,r,2,19)++
1`(t,r,2,21)++
1`(t,r,2,22)++
1`(t,r,2,23)++
1`(t,r,2,24)++
1`(t,r,2,27)++
1`(t,r,2,28)++
1`(t,r,3,0)++
1`(t,r,3,15)++
1`(t,r,3,16)++
1`(t,r,3,17)++
1`(t,r,3,19)++
1`(t,r,3,20)++
1`(t,r,3,21)++
1`(t,r,3,22)++
1`(t,r,3,23)++
1`(t,r,3,24)++
1`(t,r,3,25)++
1`(t,r,3,27)

AgentD2'inhibet 1
1`(1,6)++
1`(1,11)++
1`(1,15)++
1`(1,16)++
1`(1,17)++
1`(1,19)++
1`(1,20)++
1`(1,21)++
1`(1,22)++
1`(1,23)++
1`(1,24)++
1`(1,25)++
1`(1,27)++
1`(1,28)

AgentD2'inhibm 1
1`(0,21)++
1`(0,22)++
1`(0,23)++
1`(0,24)++
1`(0,25)++
1`(0,27)++
1`(0,28)++
1`(1,6)++
1`(1,11)++
1`(1,15)++
1`(1,16)++
1`(1,17)++
1`(1,19)++
1`(1,20)

AgentD2'inhibr 1
1`(1,6)++
1`(1,11)++
1`(1,15)++
1`(1,16)++
1`(1,17)++
1`(1,19)++
1`(1,20)++
1`(1,21)++
1`(1,22)++
1`(1,23)++
1`(1,24)++
1`(1,25)++
1`(1,27)++
1`(1,28)

AgentD2'tamp21 1
1`(t,r,0,0)++
1`(t,r,3,6)++
1`(t,r,3,11)++
1`(t,r,3,15)++
1`(t,r,3,16)++
1`(t,r,3,17)++
1`(t,r,3,19)++
1`(t,r,3,20)++
1`(t,r,3,21)++
1`(t,r,3,22)++
1`(t,r,3,23)++
1`(t,r,3,24)++
1`(t,r,3,25)++
1`(t,r,3,27)++
1`(t,r,3,28)

AgentD2'tamp23 1
1`(t,r,1,6)++
1`(t,r,1,15)++
1`(t,r,1,16)++
1`(t,r,1,17)++
1`(t,r,1,19)++
1`(t,r,1,20)++
1`(t,r,1,23)++
1`(t,r,1,24)++
1`(t,r,1,25)++
1`(t,r,1,28)++
1`(t,r,2,11)++
1`(t,r,2,15)++
1`(t,r,2,16)++
1`(t,r,2,19)++
1`(t,r,2,21)++

1`(t,r,2,22)++
1`(t,r,2,23)++
1`(t,r,2,24)++
1`(t,r,2,27)++
1`(t,r,2,28)++
1`(t,r,3,0)++
1`(t,r,3,15)++
1`(t,r,3,16)++
1`(t,r,3,17)++
1`(t,r,3,19)++
1`(t,r,3,20)++
1`(t,r,3,21)++
1`(t,r,3,22)++
1`(t,r,3,23)++
1`(t,r,3,24)++
1`(t,r,3,25)++
1`(t,r,3,27)

AgentD2'tamp24 1
1`(t,r,3,6)++
1`(t,r,3,11)++
1`(t,r,3,15)++
1`(t,r,3,16)++
1`(t,r,3,17)++
1`(t,r,3,19)++
1`(t,r,3,20)++
1`(t,r,3,21)++
1`(t,r,3,22)++
1`(t,r,3,23)++
1`(t,r,3,24)++
1`(t,r,3,25)++
1`(t,r,3,27)++
1`(t,r,3,28)

AgentD2'verrou2 1 1`1

Best Lower Multi-set Bounds
AgentD1'BaseConn1 1 empty
AgentD1'FAct14 1 empty
AgentD1'FilAct22 1 empty
AgentD1'InhibM1 1 empty
AgentD1'InhibR1 1 empty
AgentD1'InhibW1 1 empty
AgentD1'MesgI1 1 empty
AgentD1'PercepM1 1 empty
AgentD1'PercepW1 1 empty
AgentD1'PerceptR1 1 empty
AgentD1'RapportT1 1 empty
AgentD1'SPercept 1 empty
AgentD1'Senarios 1 empty
AgentD1'SigneMM1 1 empty
AgentD1'Stimuli11 1 empty
AgentD1'Stimuli12 1 empty
AgentD1'Stimuli13 1 empty
AgentD1'directs1 1 empty
AgentD1'fAct11 1 empty
AgentD1'fAct12 1 empty
AgentD1'fAct13 1 empty
AgentD1'filAct21 1 empty
AgentD1'filAct23 1 empty
AgentD1'inhibET2 1 empty
AgentD1'inhibM 1 empty
AgentD1'inhibR 1 empty
AgentD1'inhibR2 1 empty
AgentD1'inhibW 1 empty
AgentD1'messageR 1 empty
AgentD1'msgI2 1 empty
AgentD1'percepC1 1 empty
AgentD1'percepC2 1 empty
AgentD1'plans1 1 empty
AgentD1'raportT2 1 empty
AgentD1'signal1 1 empty
AgentD1'signal2 1 empty
AgentD1'signal3 1 empty
AgentD1'tableauN 1 1`0
AgentD1'tache1 1 empty
AgentD1'tache2 1 empty
AgentD1'tamp11 1 empty
AgentD1'tamp12 1 empty
AgentD1'tamp13 1 empty
AgentD1'tamp14 1 empty
AgentD1'tamp15 1 empty
AgentD1'tamp16 1 empty
AgentD1'tamp22 1 empty

AgentD1'verrou1 1
empty
AgentD2'BaseCon2
1 empty
AgentD2'CPUmod 1
empty
AgentD2'CPUrate 1
empty
AgentD2'Directi2 1
empty
AgentD2'FACt11 1
empty
AgentD2'FACt12 1
empty
AgentD2'FACt13 1
empty
AgentD2'FAcT11 1
empty
AgentD2'FAcT12 1
empty
AgentD2'FAcT13 1
empty
AgentD2'FAct11 1
empty
AgentD2'FAct12 1
empty
AgentD2'FAct13 1
empty
AgentD2'FilAct21 1
empty
AgentD2'FilAct22 1
empty
AgentD2'FilAct23 1
empty
AgentD2'InhibET2 1
empty
AgentD2'InhibR2 1
empty
AgentD2'InhibW2 1
empty
AgentD2'MesgI2 1
empty
AgentD2'MessageR
1 empty

AgentD2'PerceET2 1
empty
AgentD2'PercepC1 1
empty
AgentD2'PercepC2 1
empty
AgentD2'PercepR2 1
empty
AgentD2'PercepW2
1 empty
AgentD2'Plans1 1
empty
AgentD2'Plans2 1
empty
AgentD2'RaportT2 1
empty
AgentD2'Scenario 1
empty
AgentD2'Sig1 1
empty
AgentD2'Sig2 1
empty
AgentD2'Sig3 1
empty
AgentD2'Signal1 1
empty
AgentD2'Signal2 1
empty
AgentD2'Signal3 1
empty
AgentD2'Stimul21 1
empty
AgentD2'Stimul22 1
empty
AgentD2'Stimul23 1
empty
AgentD2'TableauN 1
1`0
AgentD2'Tache1 1
empty
AgentD2'Tache2 1
empty
AgentD2'Tamp12 1
empty

AgentD2'Tamp14 1
empty
AgentD2'Tamp15 1
empty
AgentD2'Tamp16 1
empty
AgentD2'Tamp22 1
empty
AgentD2'inhibet 1
empty
AgentD2'inhibm 1
empty
AgentD2'inhibr 1
empty
AgentD2'tamp21 1
empty
AgentD2'tamp23 1
empty
AgentD2'tamp24 1
empty
AgentD2'verrou2 1
empty

Home Properties

Home Markings
Initial Marking is not a
home marking

Liveness Properties

Dead Markings
7915
[2708,2706,2704,2702,2
700,...]

Dead Transition
Instances
None

Live Transition Instances None	Fairness Properties --- No infinite occurrence sequences.

BIBLIOGRAPHIE

Abeillé, A., Blache, P. (2000). Grammaires et analyseurs syntaxiques. Informatique et Systèmes d'Information. H. Science**:** 51-76.

Accot, J., Chatty, S., Palanque, P. (1996). "A Formal Description of Low Level Interaction and its Application to Multimodal Interactive Systems." 3rd workshop on Design, Specification and Verification of Interactive systems(Springer-Verlag): 92-104.

Allen, R. J., Douence, R., Garlan, D. (1997). "Specifying Dynamism in Software Architectures." Proceedings of Foundations of Component-Based Systems Workshop.

Arens, Y., Hoy, E., Van Mulken, S. (1993). "Structure and Rules in Automated Multimedia Presentation Planning." Actes de la conférence IJCAI'93: 1253-1259.

Asterio, P., Guerra, C., Rubira, C. F., Romanovsky, A., de Lemos, R. (2003). "A fault-tolerant software architecture for COTS-based software systems." Proceedings of the 9th European software engineering conference held jointly with 11th ACM SIGSOFT international symposium on Foundations of software engineering (Helsinki, Finland) **11**: 375-378.

Azémard, F. (1995). Des références dans le dialogue Homme-Machine multimodal, une approche adaptée du formalisme des graphes conceptuels. Toulouse, Université Paul Sabatier.

Baillie, L., Schatz, R. (2005-a). Exploring Multimodality in the Laboratory and the Field. Seventh International Conference on Multimodal Interfaces, ICMI 2005, Trento, Italy.

Baillie, L., Simon, R., Schatz, R., Wegscheider, F., Anegg, H. (2005-b). Gathering Requirements for Multimodal Mobile Applications. Seventh International Conference on Information Technology Interfaces, ITI 2005, Dubrovnik, Croatia.

Balbo, S., Coutaz, J., Salber, D. (1993). "Toward automatic evaluation of multimodal user interfaces." Proceedings of the 1st International Conference on Intelligent User Interfaces, IUI'93, ACM Press: 201-208.

Barbacci, M., Weinstock, C., Doubleday, D., Gardner, M., Lichota, R. (1993). "Durra: A Structure Description Language for Developing Distributed Applications." IEEE Software Engineering Journal **8**(2): 83-94.

Bass, L., Clements, P., Kazman, R. (1998). Software Architecture in Practice.

Bass, L., Klein, M., Gabriel, M. (2001). Applicability of General Scenarios to the Architecture Tradeoff Analysis Method. Pittsburgh, PA:, Software Engineering Institute, Carnegie Mellon University.

Bastide, R. (1992). Objets coopératifs : un formalisme pour la modélisation des systèmes concurrents. Toulouse, France, Université Paul Sabatier**:** 263.

Bastide, R., Palanque, P. (1995). "A Petri-Net Based Environment for the Design of Event-Driven Interfaces." Lecture in Computer Science16 th ICATPN'95 **935**(66-83).

Bellik, Y. (1995). Interfaces multimodales : concepts modèles architectures. Département Informatique. Paris, Université de Paris XI.

Bellik, Y., Burger. D. (1994). "Multimodal Interfaces: New Solutions to the Problem of the Computer Accessibility for the Blind." Proc. CHI'94: 24-28.

Benarif, S. (2007). Plate-forme multiagent pour la reconfiguration dynamique des architectures logicielles. PRiSM. Versailles, France, UVSQ.

Bernsen, N. O. (1994). "Modality Theory in Support of Multimodal Interface Design." Working notes, AAAI spring symposium series. Symposium: Intelligent Multi-Media Multi-Modal Systems: 21-23.

Bisson, A., Maillé, C.-A., Gill, M.-A. (2006). Jeu de mémoire. Montréal, ÉTS, Département du GL et des TI.

Blattner, M. M., Dannenberg, R.B. (1990). "Workshop Report." CHI'90 Workshop on Multimedia and Multimodal Interface Design, ACM SIGCHI Bulletin **22**: 54-58.

Bloom, T., Day, M. (1993). "Reconfiguration and Module Replacement in Argus: Theory and Practice." IEEE Software Engineering Journal: 102-108.

Bolognesi, T., Brinksma, E. (1988). "Introduction to the ISO Specification Language LOTOS." Computer Networks and ISDN Systems **14**(1).

Bolt, R. (1980). "Put-that there: Voice and gesture at the graphics interface." Computer Graphics Journal of the Association of Computing and Machinery **14**(3): 262-270.

Bond, A. H., Gasser, L. (1998). Readings in Distributed Artificial Intelligence. SanMateo, CA.

Bourdot, P., Krus, M. Gherbi, R. (1995). "Management of non-standard devices for multimodal user interfaces under UNIX/X11." Proc. of the Intern. Conf. on Cooperative Multimodal Communication (CMC'95), Eindhoven **I**: 49-61.

Bourguet, M.-L. (1992). Conception et réalisation d'une interface de dialogue personne-machine multimodale. Institut National Polytechnique de Grenoble. Grenoble, France, INPG.

Brennan, S., Ohaeri, J. (1994). "Effect of message style on users' attributions toward agents." CHI' 94 Conference Companion Human Factors in Computing Systems(Boston. ACM Press): 281-282.

Brooks, R. A. (1991-a). "Intelligence without representation." Artificial Intelligence **47**(1-3): 139-159.

Brooks, R. A. (1991-b). " Intelligence without reason." Proceedings of the 12th International Joint Conference on Artificial Intelligence(Ray Myopoulos, John ; Reiter, editor, Morgan Kaufmann, Sydney, Australia): 569-595.

Cammarata, S., McArthur, D., Steeb, R. (1988). "Strategies of cooperation in distributed problem solving." Readings in Distributed Artificial Intelligence: 102-105.

Chaib-draa, B., Levesque, P. (1996). "Hierarchical model and communication by signs, signals and symbols in multiagent environments." Journal of Experimental and Theoretical AI (JETAI) **8**: 7-20.

Chaib-draa, B., Moulin, B., Jarras, I. (2001). Agent et Systèmes Multiagents. Principes et architecture des systèmes multiagents., J.P. Briot et Y. Demazeau.

Chibelushi, C. C., Deravi, F., Mason, J.S.D. (2002). "A review of speech-based bimodal recognition." IEEE Transactions on Multimedia **4**(1): 23-37.

Ciancarini, P., Mascolo, C. (1998). "Software Architecture and Mobility." Third International Software Architecture Workshop (ISAW-3), Orlando, Florida, ACM-Press: 21-24.

Clements, P., Kazman, R., Klein, M. (2002). Evaluating Software Architectures. Boston MA.

Cohen, P., Johnston, M., McGee, D., Oviatt, S., Pittman, J., Smith, I., Chen, L., Clow, J. (1997). "QuickSet: Multimodal interaction for distributed applications." Fifth ACM International Multimedia Conference MULTIMEDIA'97: 31-40.

Cohen, P. R., Johnston, M., McGee, D., Oviatt, S., Pittman, J., Smith, I., Chen, L., Clow, J. (1997). "QuickSet: Multimodal interaction for distributed applications." Proceedings of the Fifth International Multimedia Conference, Multimedia '97(ACM Press): 31-40.

Cosi, P., Magno, E., Caldognetto, K., Vagges, G., Mian, A., Contolini, M. (1994). "Bimodal Recognition Experiments With Recurrent Neural Networks." Proc. ICASSP '94 **19-22**(2): 553-556.

Coutaz, J., Nigay, L., Salber, D., Blandford, A., May, J., Young, R. (1994). "Les propriétés CARE dans les interfaces multimodales." IHM'94, Lilles.

Coutaz, J., Nigay, L., Salber, D., Blandford, A., May, J., Young, R. (1995). "Four Easy Pieces for Assessing the Usability of Multimodal in Interaction the CARE Properties." Human Computer Interaction, Interact' 95: 115-120.

Crowley, J. L., Bérard, F. (1997). Multimodal tracking of faces for video communications. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'97), San Juan, NY, IEEE Press.

Davies, J., Woodcock, J. (1996). Using Z. Specification Refinement and Proof. Series in Computer Science.

DeNicola, R., Ferrari, G., Pugliese, R. (1998). "KLAIM: A kernel Language for Agents Interaction and Mobility." IEEE Transactions on Software Engineering **24**(5): 315-330.

Dennett, D. (1983). "International systems in cognitive ethology : the panglossian paradigm defended." The behavioural and brain sciences **6**(3): 343-390.

Djenidi, H., Benarif, S., Ramdane-Cherif, A., Levy, N., Tadj, C. (2002-d). Dynamic Architecture Based Multi-Agent Paradigm for Multimedia Multimodal Dialogs. KMN'2002: IEEE International Workshop on Knowledge Media Networking, Kyoto, Japan,.

Djenidi, H., Benarif, S., Ramdane-Cherif, A., Levy, N., Tadj, C. (2003). InterAct Software 1.0: An Eye-Gaze-Speech Multimodal Environment Dedicated To Disabled People. International Symposium On Intelligent Signal Processing and Communication Systems, Awaji Island, Japan.

Djenidi, H., Benarif, S., Ramdane-Cherif, A., Levy, N., Tadj, C. (2005). Prototypage d'une application Internet multimédia multimodale évolutive générique en vue d'aide aux handicapés. Montréal Québec, Versailles France, ÉTS-UVSQ.

Djenidi, H., Benarif, S., Ramdane-Cherif, A., Tadj, C., Levy, N. (2004-a). "Generic Multimedia Multimodal Agents Paradigms and their Dynamic Based Agent Reconfiguration at the Architectural Level." Eurasip : European journal of Applied Signal Processing: 1688-1707.

Djenidi, H., Ramdane-Cherif, A., Tadj, C., Levy. N. (2002-a). Architectures multi-agents génériques à base de réseaux de Pétri temporisés colorés pour la fusion Multimodale en entrée. Proceedings of the 14th French-Speaking Conference on Human-Computer interaction (ConféRence Francophone Sur L'interaction Homme-Machine IHM'02), Poitiers, France, ACM Press, New York, NY, 33-40.

Djenidi, H., Ramdane-Cherif, A., Tadj, C., Levy. N. (2002-b). Generic Multi-Agent Architectures for Multimedia Multimodal Dialogs. MOCA'2002: Second Workshop on Modeling of Objects, Components, and Agents, Aarhus, Denmark.

Djenidi, H., Ramdane-Cherif, A., Tadj, C., Levy. N. (2002-c). Dynamic Based Agent Reconfiguration of Multimedia Multimodal Architecture. Fourth International Symposium on Multimedia Software Engineering, Newport Beach, California, USA.

Djenidi, H., Ramdane-Cherif, A., Tadj, C., Levy. N. (2002-e). Dynamic Based Agent Reconfiguration of Multimedia Multimodal Architecture. MSE'2002: Fourth International Symposium on Multimedia Software Engineering., Newport Beach, California, USA.

Djenidi, H., Ramdane-Cherif, A., Tadj, C., Levy. N. (2004-b). "Generic Multi-Agents Architecture for Multimedia Multimodal Software Environment." JOT'2004: International Journal of Object Technology **3**(8): 147-168.

Duke, D., Barnard, P., Duce, D., May J. (1994). Syndetic model for human-computer interaction. Amodeus-2.

Duke, D., Harrison, M.D. (1997). "Mapping User Requirements to Implementations." Software Engineering Journal **10**(1): 54-75.

Durfee, E. H., Lesser, V. (1989). "Negotiating task decomposition and allocation using partial global planning." Distributed Artificial Intelligence **II**: 229-244.

Emery, K. V., Edwards, P.J., Jacko, J.A., Moloney, K.P., Barnard, L., Kongnakorn, T., Sainfort, F., Scott, I.U. (2003). Toward Achieving Universal Usability for Older Adults Through Multimodal Feedback. CUU'03,, British Columbia, Canada.

Erman, L. D., Hayes-Roth, F., Lesser, V., Raj Reddy D. (1980). "The HERSAY II speech understanding system: Integrating knowledge to resolve uncertainty." Computing Surveys **12**(2): 213-253.

Farcy, R., Leroux, R., Damaschini, R., Legras, R., Bellik, Y., Jacquet, C., Greene, J., Pardo. P. (2004). "Perception de l'espace et locomotion des non-voyants par profilométrie laser: Aides électroniques à la locomotion." J3eA, Journal sur l'enseignement des sciences et technologies de l'information et des systèmes **3**(Hors-Série 1).

Faure, C., Julia, L. (1993). "Interaction homme-machine par la parole et le geste pour l'édition de documents: TAPAGE." International Conference on Interfaces to Real and Virtual Worlds: 171-180.

Ferber, J. (1994). "Simulating with Reactive Agents." Many Agent Simulation and Artificial Life: 8-28.

Ferber, J. (1995). Les systèmes multi-agents, vers une intelligence collective.

Ferguson, I. A. (1992). Touring Machines : An Architecture for Dynamic, Rational, Mobile Agents., University of Cambridge, UK.

Finin, T., Fritzson, R. (1994). "KQML: a language and protocol for knowledge and information exchange." Proceedings of the Thirteenth International Workshop on Distributed Artificial Intelligence, Lake Quinalt, WA: 126-136.

Fischer, K., Chaib-draa, B., Müller, H.J., Müller, J.P., Pischel, M. (1999). "A simulation approach based on negotiation and cooperation between agents." IEEE Trans. on Systems, Man, and Cybernetics **29**(4): 531-545.

Frohlich, D. (1991). "The Design Space of Interfaces, Multimedia System. Interaction and Applications." 1st Eurographics Workshop: 53-69.

Gacek, C., Abd-Allah, A., Clark, B., Boehm, B. (1995). "On the Definition of Software System Architecture." Proceedings of the First International Workshop on Architectures for Software System Architecture- In Cooperation with the 17th International Conference on Software Engineering.(In D. Garlan editor, Seattle, WA): 85-95.

Gagnon, P., Laberge, C., Héroux, S., Marois, P. (2006). Tux-Xmen. Montréal, ÉTS, Département du GL et des TI.

Garlan, D., Cheng. S.-W., Schmerl, B. (2003). Increasing System Dependability through Architecture-based Self-repair. Architecting Dependable Systems. C. G. R. de Lemos, A. Romanovsky, Springer-Verlag.

Garlan, D., Perry, D. (1995). "Introduction to the Special Issue on Software Architecture." IEEE Transactions on Software Engineering **21**(4).

Garlan, D., Schmerl, B.R., Chang, J. (2001). Using Gauges for Architecture-Based Monitoring and Adaptation. The 1st Working Conference on Complex and Dynamic System Architecture, Brisbane, Australia.

Garland, D., Shaw, M. (1993). " An Introduction to Software Architecture." Advances in Software Engineering and Knowledge Engineering.(World Scientific Publishing Co. ed.) **1**: 1-39.

Genesereth, M. R., Ketchpel, S.P. (1994). "Software Agents." Communication of the ACM **37**(7).

Gourdol, A., Nigay, L., Salber, D., Coutaz, J. (1992). "Two case Studies of Software Architecture for Multimodal Interactive Systems: VoicePaint and a Voice-enabled Graphical Notebook." Prooceedings of the IFIP TC2/WG2.7 Working Conference on Engineering for Human-Computer Interaction: 271-283.

Gray, R. (1998). Agent tcl : A flexible and secure mobile-agent systems. Hanover, NH, Dartmouth College, Computer Science Dept.

Guizzardi, R. S. S., Aroyo, L., Wagner, G. (2003). "Agent-oriented Knowledge Management in Learning Environments: A Peer-to-Peer Helpdesk Case Study."

Proceedings of AAAI Spring Simposium on Agent Mediated Knowledge Management (AMKM'03) In Lecture Notes in Computer Science Series **2926**: 57-72.

Hachette (2004). Dictionnaire de la langue française.

Hall, A. (1990). "Seven Myths of Formal Methods." IEEE Software **7**(5): 11-19.

Hill, W., Wroblewksi, D., McCandless, T. and Cohen, R. (1992). "Architectural Qualities and Principles for Multimodal and Multimedia Interfaces." Multimedia Interface Design: Blattner, M. and Dannenberg, R. B. ACM Press ed.

http://edwardtse.com/edwardtse-videos.html (2006). site consulté en mars 2006.

http://marioparty.com/ (2006). site consulté en mars 2006.

http://tux.crystalxp.net (2006). site consulté en décembre 2006.

http://www.cse.ogi.edu/CHCC/QuickSet/mainProj.html (2006). site consulté en décembre 2006.

http://www.fipa.org/ (2006). site consulté en novembre 2006.

http://www.ieee.org/web/standards/home/index.html (2006). site consulté en novembre 2006.

http://www.limsi.fr/Individu/bellik/ (2005). site consulté en décembre 2005.

http://www.rational.com/uml/index.jsp (2006). site consulté en décembre 2006.

Hutchins, E. L., Holland, J.D., Norman, D.A. (1986). Direct manipulation interfaces. User centered system design: new perspectives on human computer design. D. A. illsdale, Norman, and S.W., Draper,. NJ, Lawrence Erlbaum.

Jacko, J. A., Scott, I.U., Sainfort, F., Moloney, K.P., Kongnakorn, T., Zorich, B.S., Emery, V.K. (2003). "Effects of multimodal feedback on the performance of older adults with normal and impaired vision." Lecture Notes in Computer Science (LNCS) **2615**: 3-22.

Jacquet, C., Bellik, Y., Bourda, Y., Farcy, R. (2004). " A Context-Aware Locomotion Assistance Device for the Blind." ICCHP 2004, 9th International Conference on Computers Helping People with Special Needs, Paris, France: 438-445.

Jennings, N. R. (1995). "Controlling cooperative problem solving in industrial multi-agent systems using joint intentions." J. of Artificial Intelligence **74**(2).

Jennings, N. R., Wooldridge, M., Sycara, K. (1998-b). "A roadmap of agent research and development." Int Journal of Autonomous Agents and Multi-Agent Systems **1**(1): 7-38.

Jennings, N. R., Wooldridge, M.J. (1998-a). "Applications of Intelligent Agents. Agent." Technologies Foundations, Applications and Markets **3**(28).

Jensen, K. (1997-a). Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use, Basic Concepts, Monographs in Theoretical Computer Science.

Jensen, K. (1997-b). Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use, Analysis Methods., Monographs in Theoretical Computer Science.

Jensen, K., Christensen, S., Huber, P., Holla, M. (1995). Design/CPN Reference Manual,University of Aarhus, Denmark, Department of Computer Science.

Johnston, M., Cohen, P., McGee, D., Oviatt, S., Pittman, J., Smith, I. (1997). "Unification-based multimodal integration." 35th Annual Meeting of the Association for Computational Linguistics, Madrid, (Spain): 281-288.

Jourlin, P. (1998). Approche bimodale du traitement automatique de la parole : application à la reconnaissance du message et du locuteur. France, Université d'Avignon et des Pays de Vaucluse.

Kaiser, E., Olwal, A., McGee, D., Benko, H., Corradini, A., Li, X., Cohen, P., Feiner, S. (2003). "Mutual disambiguation of 3D multimodal interaction in augmented and virtual reality." Proceedings of the 5th international Conference on Multimodal interfaces, ICMI '03Vancouver, British Columbia, Canada, ACM Press, New York, NY: 12-19.

Kamio, M., Koorita, H., Matsuura, M., Tamura Nitta, T. (1994). "A UI design support tool for multimodal spoken dialogue system." International Conference on Spoken Language Processing, Philadelphia, IEEE Computer Society (PA) **3**: 1283-1286.

Karp, R. M., Miller R.E. (1969). "Parallel program schemata." Journal of Computing System Science **4**: 147-195.

Kazman, R. (1996). "Tool support for architecture analysis and design." In Joint Proceedings of the Second international Software Architecture Workshop (Isaw-2) and international Workshop on Multiple Perspectives in Software Development (Viewpoints '96) on SIGSOFT '96 Workshops, ACM Press, New York, NY,: 94-97.

Kornfeld, W. (1979). ETHER : a parallel problem solving system. Procedings of 6th IJCAI.

Kramer, J., Magee, J. (1990). "The Evolving Philosophers Problem: Dynamic Change Management." IEEE Trans. On Software Eng. **16**(11): 1293-1306.

Kramer, J., Magee, J. (1996-a). "Dynamic Structure in Software Architectures." Proceedings of the fourth ACM SIGSOFT Symposium on Foundations of Software Engineering (FSE'96), ACM-Press: 3-14.

Kramer, J., Magee, J. (1996-b). "Self Organizing Software Architectures." Joint Proceedings of the second International Software Architecture Workshop (ISAW-2) and International Workshop on Multiple Perspectives in Software Development (Viewpoints'96), ACM-Press: 35-38.

Kraus, S., Wilkenfeld, J. (1991). "Negotiations over time in a multi-agent environment." Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91), Sydney, Australia: 56-61.

Kraus, S., Wilkenfeld, J., Zlotkin, G. (1995). "Multiagent negotiation under time constraints." Artificial Intelligence **75**(2): 297-345.

Kreifelts, T., von Martial, F. (1991). "A negotiation framework for autonomous agents." Decentralized AI 2 - Proceedings of the Second European Workshop on Modelling Autonomous Agents and Multi-Agent Worlds (MAAMAW-90),(Y. Demazeau and J-P. Müller, editors, Elsevier Science Publishers B.V. : Amsterdam, The Netherlands): 71-88.

Kruchten, P. B. (1995). "The 4+1 view model of architecture." IEEE Software **12**(6): 42-50.

Krus, M. (1993). XSPECIMEN: Un Éditeur Interactif de Graphes de Transitions décrivant des Interfaces Multimodales. Paris, Université Paris XI.

Lesser, V. R. (1991). "A retrospective view of FA/C distributed problem solving." IEEE Transactions on Systems, Man, and Cybernetics, Special Issue on Distributed Artificial Intelligence **21**(6): 1347-1362.

Lesser, V. R., Corkill, D.D., (1988). The distributed vehicle monitoring testbed : A tool for investigating distributed problem solving networks. Readings From the AI Magazine, R. Engelmore, Ed. American Association for Artificial Intelligence, Menlo Park, CA**:** 69-85.

Lin, T. a. I., A. (2006). "Evaluating usability based on multimodal information: an empirical study." Proceedings of the 8th international Conference on Multimodal interfaces, ICMI '06, Banff, Alberta, ACM Press, New York, NY: 364-371.

Littlewood, B., Strigini L. (2000). "Software reliability and dependability: a roadmap." International Conference on Software Engineering archive. Proceedings of the Conference on The Future of Software Engineering, Limerick, Ireland **175-188**.

MacColl, D., Carrington, D. (1998). "Testing MATIS: a Case Study On Specification-Based Testing of Interactive Systems." FAHCI: 57-69.

Maes, P. (1994). "Agents that reduce work and information overload." Communications of the ACM **37**(7): 31-40.

Maes, P. (1994). "Social interface agents : Acquiring competence by learning from users and other agents." Software Agents - Papers from the 1994 Spring Symposium (Technical Report SS-94-03), In O. Etzioni, editor, AAAI Press.: 71-78.

Malone, T. W. (1990). Cognition, Computation and Cooperation. Organizing information processing systems : parallels between human organizations and computer systems. W. W. Z. a. S. P. Robertson**:** 56-83.

Manchón, P., del Solar, C., Amores, G., Pérez, G. (2006-a). "The MIMUS Corpus." LREC 2006." International Workshop on Multimodal Corpora From Multimodal Behaviour Theories to Usable Models **Genoa, Italy**: 56-59.

Manchón, P., Pérez, G., Amores, G. (2006-b). "Multimodal Fusion: a new hybrid strategy for dialogue systems." Proceedings of the 8th international conference on Multimodal interfaces, Banff, Alberta, Canada,: 357 - 363.

Martial, F. V. (1992). Coordinating Plans of Autonomous Agents. Heidelberg, Germany.

Martin, J. (1997). Towards "intelligent" cooperation between modalities. IJCAI Workshop on Intelligent Multimodal Systems, Nagoya ,Japan.

Martin, J., Veldman R., Beéroule, D. (1995-a). "Towards adequate representation technologies for multimodal interfaces." International Conference on Cooperative Multimodal Communication **1**: 207-223.

Martin, J.-C. (1995). Coopérations entre modalités et liage par synchronie dans les interfaces multimodales. Paris, École Nationale Supérieure des Télécommunications.

Martin, J. C., Béroule, D. (1993). "Type et buts de coopérations entre modalités." Cinquièmes journées sur l'ingénierie des interfaces Homme-Machine IHM'93: 19-20.

McGee, D. R., Cohen, P.R., Wu, L. (2000). "Something from nothing: Augmenting a paper-based work practice with multimodal interaction." Proceedings of the Conf. on Designing Augmented Reality Environments: 71-80.

Milner, R. (1989). Communication and Concurrency. NJ.

Moeschler, J. (1989). Modélisation du Dialogue. Représentation de l'inférence argumentative. Paris.

Moran, D. B., Cheyer, A.J. Julia, L.E., Martin, D.L., Park, S. (1997). "Multimodal User Interfaces in the Open Agent Architecture." Proceedings of the 1997 International Conference on Intelligent User Interfaces, IUI97(Orlando, Florida): 61-68.

Muller, H. J. (1996). "Negotiation principles." Foundations of Distributed Artificial Intelligence,G.M.P., O'Hare and N.R., Jennings, (eds.) Wiley: 211-229.

Muller, P.-A. (1997). Modélisation objet avec UML.

Navarre, D., Palanque, P., Bastide, R. (2002). "Model based interactive prototyping of highly interactive applications." CADUI, Kluwer academic publishers: 205-216.

Nigay, L. (1994). Conception et modélisation logicielles des systèmes interactifs. Grenoble, Université Joseph Fourier.

Nigay, L. (2001). Modalité d'interaction et multimodalité. Université Joseph-Fourier.

Nigay, L., Coutaz, J. (1993). "A design space for multimodal systems: Concurrent processing and data fusion." International Conference on Computer-Human Interaction, ACM: 172-178.

Nigay, L., Coutaz, J. (1996). "Espaces de conception des interfaces multimédia et multimodales." TSI, numéro spécial Multimédia et collecticiel **15**: 1195-1225.

Nigay, L., Coutaz., J. (1995). "A generic platform for addressing the multimodal challenge." International Conference on Computer-Human Interaction, ACM, Denver (CO): 98-105.

Nii, P. (1989). "Blackboard systems." In A. Bat, r, P. R. Cohen & E, A, Feigenbaum (eds.), Handbook of Artificial Intelligence **IV**: 1-74.

Nwana, H. S. (1996). "Software Agents: An Overview." Knowledge Engineering Review **11**(3): 205-244.

Odell, J., Parunak, H.V.D., Bauer, B. (2000). "Extending UML for Agents." Proc. of the Agent-Oriented Information Systems Workshop at the 17th National conference on Artificial Intelligence(Gerd Wagner, Yves Lesperance, and Eric Yu eds., Austin, TX, AOIS Workshop at AAAI 2000,): 3-17.

Oreizy, P., Medvidovic, N., Taylor, R.N. (1998). "Architecture-Based Runtime Software Evolution." Proc. 20th Int'l Conf. On Soft. Eng. (ICSE'98), Kyoto, Japan: 177-186.

Oviatt, S. L. (2000-b). "Multimodal System Processing in Mobile Environments." Proceedings of the Thirteenth Annual ACM Symposium on User Interface Software Technology, UIST'2000,(ACM Press, New York): 21-30.

Oviatt, S. L. (2003). Multimodal interfaces, chap. 14. The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications. L. E. A. J. Jacko and A. Sears editor, Mahwah, NJ. Mahwah, NJ**:** 286-304.

Oviatt, S. L., & al. (2000-a). "Designing the user interface for multimodal speech and gesture applications: State-of-the-art systems and research directions." Journal of Human Computer Interaction **15**(4): 263-322.

Parnas, D. L. (1969). "On the Use of Transition Diagram in the Design of a User Interface for Interactive Computer System." Proceedings of the 24th ACM Conference, ACM Press.

Parunak, H. V. D. (1996). Applications of distributed artificial intelligence in industry. Foundations of Distributed AI,. O. H. a. N. R. G.M.P., Jennings. Chichester, England., John Wiley & Sons.

Pérez, G., Amores G., Manchón, P. (2005). Two strategies for multimodal fusion. Proceedings of ICMI'05 Workshop on Multimodal Interaction for the Visualisation and Exploration of Scientific Data., Trento, Italy.

Perry, D. E., Wolf, A.L. (1992). " Foundations for the Study of Software Architecture." ACM Software Eng. Notes **17**(4): 40-52.

Petri, C. A. (1962). Kommunikation Mit Automaten. Institute fur instrumentelle mathematik, schriften des IIM Nr. 2, German.

Pierrel, J. M. (1987). Dialogue oral Homme-Machine.

Purtilo, J. M. (1994). "The Polylith Software Bus." ACM TOPLAS **16**(1): 151-174.

Rabiner, L., Juang, B. (1993). Time Alignment and Normalization. Fundamentals of speech recognition.**:** 200-238.

Ramdane-Cherif, A., Levy, N. (2001). "Agent-Based Dynamic Architecture." NOSA'01: Fourth Nordic Symposium on Software Architecture. Odense, Dehnmark.

Ramdane-Cherif, A., Levy, N. (2002). "An Approach for Dynamic Reconfigurable Software Architectures." IDPT'02: The Sixth World conference on Integrated Design and Process Technology, Pasadena, California, USA.

Ramdane-Cherif, A., Levy, N. (2002-a). "An Approach for Dynamic Reconfigurable Software Architectures." IDPT'02: The Sixth World conference on Integrated Design and Process Technology, Pasadena, California, USA.

Ramdane-Cherif, A., Levy, N., Djenidi, H., Tadj, C. (2002-b). Agents for Resolving a Complex Optimization Problem. In ICCI 2002. The 1st IEEE International Conference on Cognitive Informatics, Calgary, Alberta, Canada.

Reisig, W. (1985). Petri nets : an introduction. EATCS monographs on theoretical computer science.

Ricordel, P.-M. (2001). Programmation Orientée Multi-Agents : Développement et Déploiement de Systèmes Multi-Agents Voyelles. Grenoble, INPG.

Rousseau, D., Moulin, G., Lapalme, B. (1995). "A language for describing coordination in multiagent systems." Actes des Deuxièmes journées francophones Intelligence Artificielle Distribuée et Systèmes Multi-Agents, Voiron, (France): 3-14.

Salber, D. (1994). "Observing users in multimodal interaction." Conference Companion on Human Factors in computing System, CHI'94, C. Plaisant Ed., ACM Press New York, NY: 103-104.

Sarukkai, R., Hunter, C. (1997). "Integration of eye fixation information with speech recognition systems." European Conference on Speech Communication and Technology,Rhodes (Greece): 97-100.

Schneider, S. (2000). Concurrent and Real-time Systems: The CSP Approach.

Schomaker, L., Nijtmans, J., Camurri, A., Lavagetto, F., Morasso, P., Benoît, C., Guiard-Marigny, T., LeGoff, B., Robert-Ribes, J., Adjoudani, A., Defée, I.,

Münch, S., Hartung, K., Blauert, J. (1995). Common concepts and software tools. Esprit Project 8579 MIAMI, WP 1.

Serrano, M., Nigay, L., Demumieux, R., Descos, J., and Losquin, P. (2006). "Multimodal interaction on mobile phones: development and evaluation using ACICARE." Proceedings of the 8th Conference on Human-Computer interaction with Mobile Devices and Services, MobileHCI '06, Helsinki, Finland, ACM Press, New York, NY, **159**: 129-136.

Shannon, C., Weaver, W. (1963). The mathematical theory of communication. Urbana, University of Illinois press.

Shoham, Y. (1993). "Agent-Oriented Programming." Artificial Intelligence **60**: 51-92.

Sycara, K. P. (1989). "Multiagent compromise via negotiation." Distributed Artificial Intelligence **2**: 119-138.

Sycara, K. P. (1990-a). "Negotiation planning : An AI approach." European Journal of Operational Research **46**(216-234).

Sycara, K. P. (1990-b). "Persuasive argumentation in negotiation." Theory and Decision **28**: 203-242.

Tabeling, P. (2002). "Multilevel Modeling of Concurrent and Distributed Systems." SERP'02 International Conference: 94-100.

Takeuchi, A., Naito, T. (1995). "Situated facial displays: Towards social interaction." International Conference on Computer Human Interaction CHI, Denver (CO) ACM: 450-455.

Taylor, R. N., Medvidovic, N., Anderson, K.N., Whitehead, E. J., Robbins, Jr., Nies, J.E., Oreizy, K.A., Dubrow, P.D.L. (1996). "A Component and Message-Based Architectural Style for GUI Software." IEEE Transactions on Software Engineering. **22**(6): 390-406.

Uchihira, N. e. H. S. (1990). "Verification and synthesis of concurrent programs using Petri nets and temporal logic." The Transactions of the IEICE **E73**(12): 2001-2010.

University of Aarhus, D. (2006). "CPN-TOOLS version 2.2.0." http://wiki.daimi.au.dk/cpntools/cpntools.wiki: (site consulté en Décembre 2006).

Van Belle, W., Fabry, J. (2001). "Experience in Mobile computing/ The Cborg Mobile Multi-Agent System." Technology of Object-Oriented Languages and Systems Tools **38**: 7-18.

Vauclair, J. (1992). L'intelligence de l'animal.

Vidal-Naquet G., C. G., A. (1992). Réseaux de Petri et systèmes parallèles.

Villing, J., Larsson, S. (2006). "Dico - A Multimodal Menu-based In-vehicle Dialogue System." BRANDIAL.

Vinoski, S. (1997). " CORBA: Integrating diverse applications within distributed heterogeneous environments." IEEE Communications **14**(2).

Vitense, H. S., Jacko, J.A., Emery, V.K. (2003). "Multimodal feedback: An assessment of performance and mental workload." Ergonomics **46**(1-3): 68-87.

Vlassis, N. A. (2003). A Concise Introduction to Multiagent Systems and Distributed AI.

Vo, M. (1998). A Framework and Toolkit for the Construction of Multimodal Learning Interfaces. Pittsburgh, USA, Carnegie Mellon University.

Vo, M., Waibel, A. (1997). Modeling and interpreting multimodal inputs: A semantic integration approach., Carnegie Mellon University.

Vo, M., Wood, C. (1996). "Building an application framework for speech and pen input integration in multimodal learning interfaces." Proceedings of the International Conference on Acoustics Speech and Signal Processing (IEEE-ICASSP) **6**: 3545-3548.

Von martial, F. (1992). "Coordinating plans of autonomous agents." **160**(Springer Verlag eds.).

Vongkasem, L., Chaib-draa, B. (1999). "ACL as a joint project between participants : A preliminary report." First Workshop on Agent Communication Language (ACL'99), Stokholm, Sweden.

Walker, J., Sproull, L., Subramani, R. (1994). "Using a human face in an interface." Human Factors in Computing Systems: 85-91.

Watzlawick, P., Helmick-BeaVin, J., Jackson, D. (1972). Une logique de la communication. Paris.

Wayner, P. (1995). Agents Unleashed: A Public Domain Look at Agent Technology. Boston, MA : AP Professional.

Weiss, G. (1999). Multiagent Systems.

Xavier, N. (1992). ATP : une algèbre pour la spécification et l'analyse des systèmes temps réel. Institut National Polytechnique de Grenoble. Grenoble, France.

Zouinar, M., Relieu, M., Salembier, P., Calvet, G. (2004). "Conception d'un dispositif d'observation et d'analyse de l'usage des technologies mobiles." Mobiquité'04, Antibes.